**PROJECT TITLE: CREATE A CHATBOT USING PYTHON**

**PHASE 4 DEVELOPING PART-2**

**PRESENTED BY**

**SATHISH KUMAR N**

**710021106307**

**CONTINUE BUILDING THE CHATBOT INTEGRATING IT INTO A WEB APP USING FLASK**

**STEPS TO FOLLOW THE PROCEDURE TO CREATE WEB APP USING FLASK:**

**STEP:1** Install the visual studio code and python latest version.

**STEP:2** Install required libraries and extension in the vscode

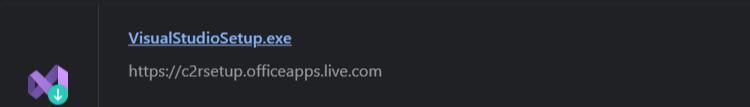
**STEP:3** Create the folder in the system and open the folder in the vscode and install the virtual env in the vs code.

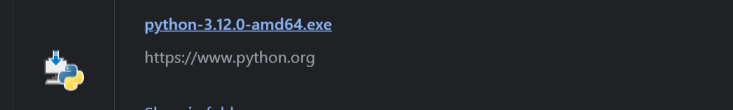
**STEP:4** Install required libraries to import the code in the vscode application.

**STEP:5** create a code for the integrating the chatbot in the web applications.

**STEP1:**

Install the visual studio code and python latest version.





**STEP 2:**

Install required libraries and extension in the vscode

**Libraries for the flask are**

1.pip install flask

2.pip install flask-login

3.pip install nltk

4.pip install transformer

5.pip install torch’

**Extension for web app create in vscode:**

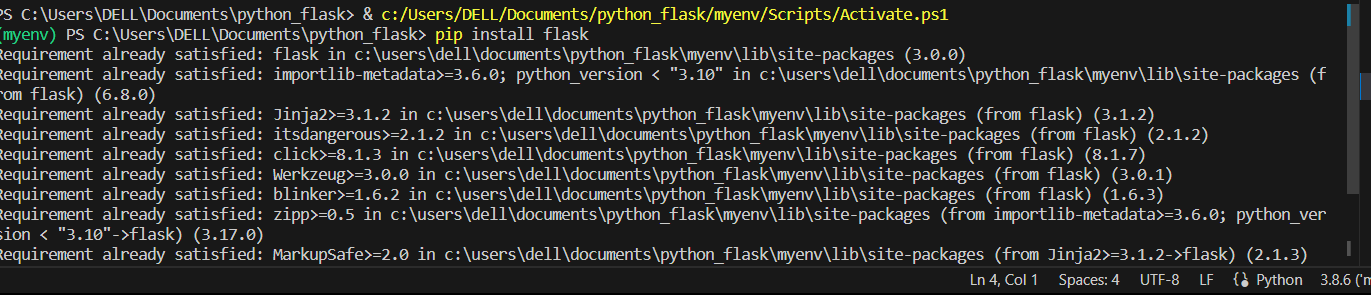
1**.**Flask

2.torch

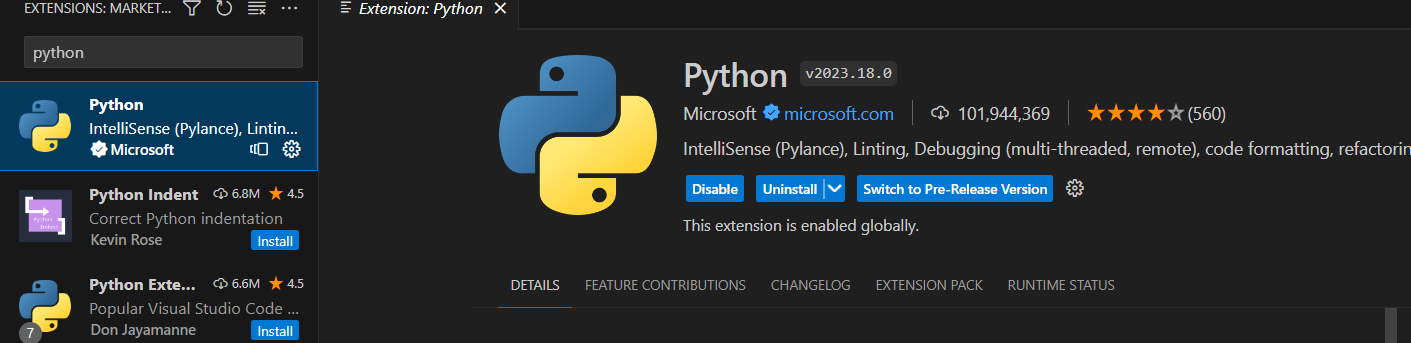
3.python

4.transformer

5.nltk

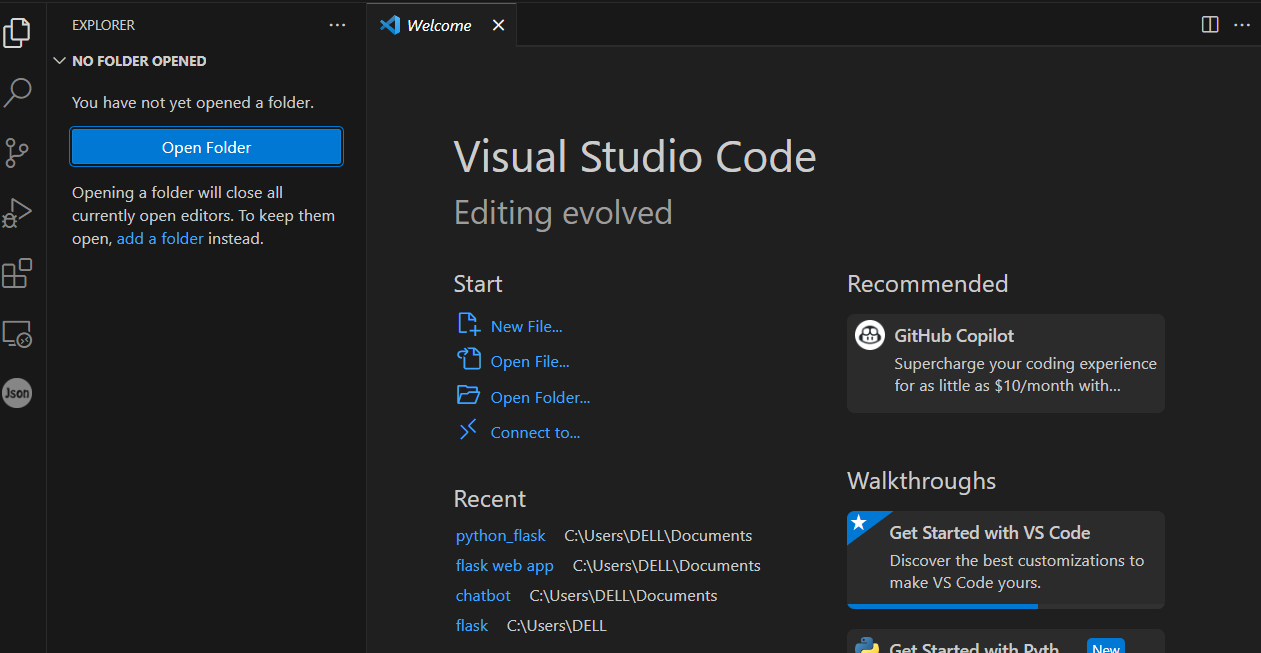
**Libraries** 

**Extension**



**STEP 3:**

Create the folder in the system and open the folder in the vscode and install the virtual env in the vs code.



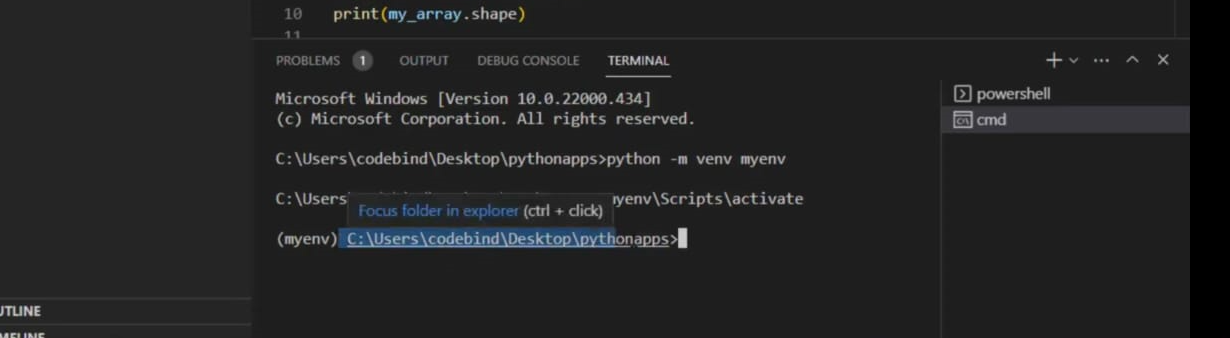
**Install the virtualenv in the vs code:**

Python -m venv -create the virtaulenv in the Terminal in the vscode

**Venv** file in the folder and activate the scripts in the folder for this application.

Venv\Scripts\activate- virtualenv terminal code.

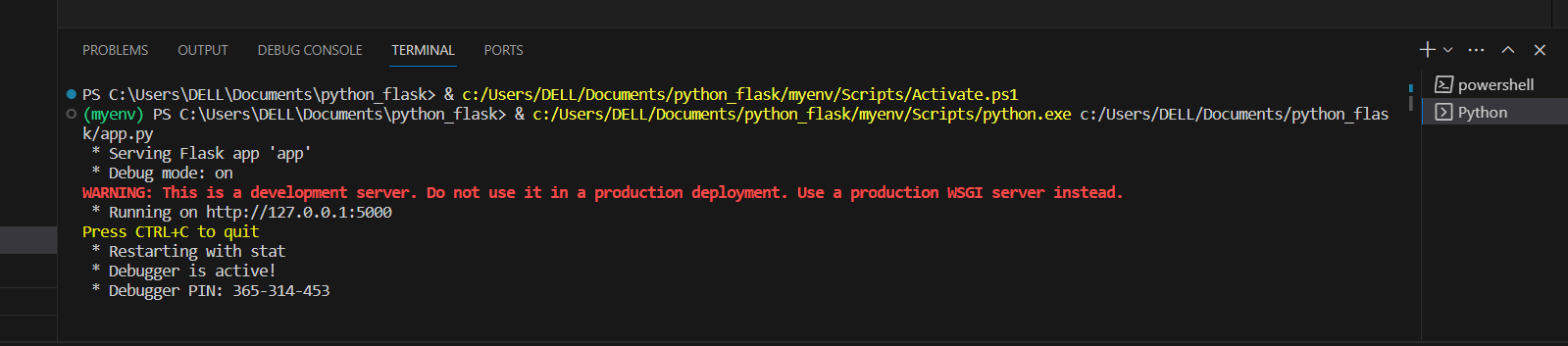
Venv virtualenv created in that folder.



**create a code for the integrating the chatbot in the web applications:**

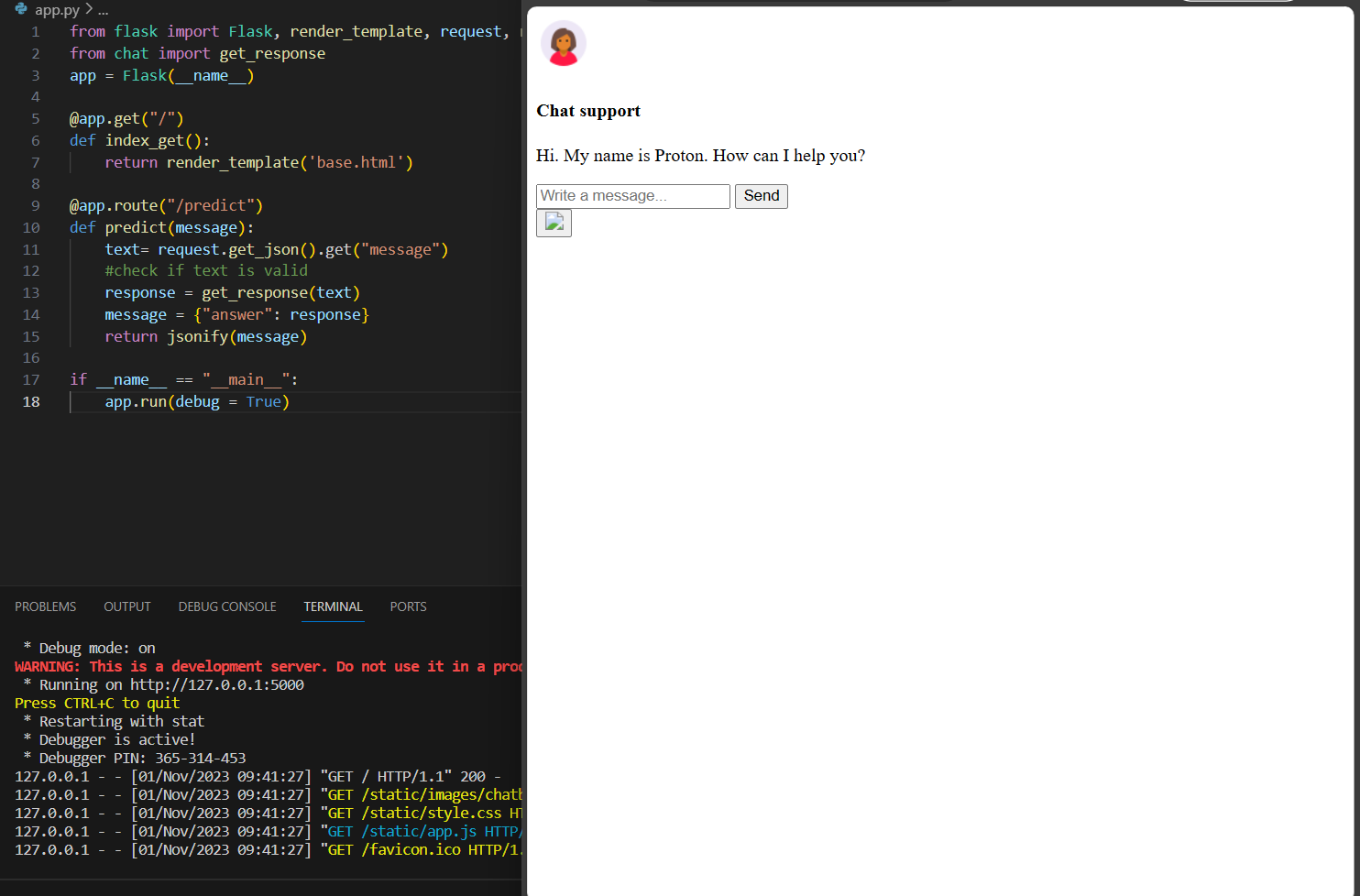
* Files are saved in app.py(python formate):
* Create the web design html code and the images in the code in vscode
* In the code was successfully in the terminal display the localhost address for the web application output check
* The follow the localhost the travel the web application to access the chatbot users.
* Integrating the chatbot to the web using flask
* The chatbot receive the message from user and response through the bot.
* The dataset attached the file in this project

Localhost : http://127.0.0.1:5000

The localhost are the followed link to the web page.

The localhost are follow link are go to the web page

* The static file have the json file in the flask application
* The templates are have the web page file base.html file in the templates.
* The base.html file are run the web page in the localhost
* Localhost follow this webpage link.



The above image are follow link of webpage integrating the chatbot to the website.

PROGRAM:

**App.py code:**

from flask import Flask, render\_template, request, redirect, url\_for, flash,jsonify

from chat import get\_response

app = Flask(\_\_name\_\_)

@app.get("/")

def index\_get():

return render\_template('base.html')

@app.route("/predict")

def predict(message):

text= request.get\_json().get("message")

#check if text is valid

response = get\_response(text)

message = {"answer": response}

return jsonify(message)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug = True)

**NLTK.UTILITIES CODE:**

mport nltk

#nltk.download('punkt') #uncomment to download punkt package

from nltk.stem.porter import PorterStemmer

import numpy as np

#initialise stemmer

stemmer = PorterStemmer()

#define tokenisation function

def tokenize(sentence):

return nltk.word\_tokenize(sentence)

#define stemming function

def stem(word):

return stemmer.stem(word.lower())

#define bag of words function

def bag\_of\_words(tokenized\_sentence, all\_words):

"""

sentence = ["hello", "how", "are", "you"]

words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]

bag = [ 0, 1, 0, 1, 0, 0, 0]

"""

tokenized\_sentence = [stem(w) for w in tokenized\_sentence]

bag =np.zeros(len(all\_words), dtype=np.float32)

for idx, w in enumerate(all\_words):

if w in tokenized\_sentence:

bag[idx] = 1.0

return bag

**Chat.PY:**

import random

import json

import torch

from model import NeuralNet

from nltk\_utilities import bag\_of\_words, tokenize

#check for gpu availability

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

with open('intents.json', 'r') as json\_data:

intents = json.load(json\_data)

FILE = "data.pth"

data = torch.load(FILE)

input\_size = data["input\_size"]

hidden\_size = data["hidden\_size"]

output\_size = data["output\_size"]

all\_words = data['all\_words']

tags = data['tags']

model\_state = data["model\_state"]

model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)

model.load\_state\_dict(model\_state)

model.eval()

bot\_name = "Proton"

def get\_response(msg):

sentence = tokenize(msg)

X = bag\_of\_words(sentence, all\_words)

X = X.reshape(1, X.shape[0])

X = torch.from\_numpy(X).to(device)

output = model(X)

\_, predicted = torch.max(output, dim=1)

tag = tags[predicted.item()]

probs = torch.softmax(output, dim=1)

prob = probs[0][predicted.item()]

if prob.item() > 0.75:

for intent in intents['intents']:

if tag == intent["tag"]:

return random.choice(intent['responses'])

return "I do not understand..."

if \_name\_ == "\_main\_":

print("Let's chat! (Type 'quit' to exit)")

while True:

# sentence = "do you use credit cards?"

sentence = input("You: ")

if sentence == "quit":

break

resp = get\_response(sentence)

print(resp)

**HTML CODE:**

<!DOCTYPE html>

<html lang="en">

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<head>

<meta charset="UTF-8">

<title>Chatbot</title>

</head>

<body>

<div class="container">

<div class="chatbox">

<div class="chatbox\_\_support">

<div class="chatbox\_\_header">

<div class="chatbox\_\_image--header">

<img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image">

</div>

<div class="chatbox\_\_content--header">

<h4 class="chatbox\_\_heading--header">Chat support</h4>

<p class="chatbox\_\_description--header">Hi. My name is Proton. How can I help you?</p>

</div>

</div>

<div class="chatbox\_\_messages">

<div></div>

</div>

<div class="chatbox\_\_footer">

<input type="text" placeholder="Write a message...">

<button class="chatbox\_\_send--footer send\_\_button">Send</button>

</div>

</div>

<div class="chatbox\_\_button">

<button><img src="{{ url\_for('static', filename='images/chatbox-icon.svg') }}" /></button>

</div>

</div>

</div>

<script>

$SCRIPT\_ROOT = {{ request.script\_root|tojson }};

</script>

<script type="text/javascript" src="{{ url\_for('static', filename='app.js') }}"></script>

</body>

</html>

**Train.py**

#import necessary libraries

import json

from nltk\_utilities import tokenize, stem, bag\_of\_words

import numpy as np

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader

from model import NeuralNet

#import json file and open it

with open('intents.json') as f:

intents = json.load(f)

#tokenisation

all\_words = []

tags = []

xy = []

for intent in intents['intents']:

tag = intent['tag']

tags.append(tag)

for pattern in intent['patterns']:

w = tokenize(pattern)

all\_words.extend(w)

xy.append((w,tag))

#ignore the following words

ignore\_words = ['?', '!', '.', ',']

#stemming

all\_words = [stem(w) for w in all\_words if w not in ignore\_words]

all\_words = sorted(set(all\_words))

tags = sorted(set(tags))

print(tags)

#training data

X\_train = []

y\_train = []

for (pattern\_sentence, tag) in xy:

bag = bag\_of\_words(pattern\_sentence, all\_words)

X\_train.append(bag)

label = tags.index(tag)

y\_train.append(label)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

#build pytorch dataset

class ChatDataset(Dataset):

def \_\_init\_\_(self):

self.n\_samples = len(X\_train)

self.x\_data = X\_train

self.y\_data = y\_train

#dataset[idx]

def \_\_getitem\_\_(self, index):

return self.x\_data[index], self.y\_data[index]

def \_\_len\_\_(self):

return self.n\_samples

#hyperparameters

batch\_size = 8

hidden\_size = 8

output\_size = len(tags)

input\_size = len(X\_train[0])

learning\_rate = 0.001

num\_epochs = 1000

dataset = ChatDataset()

train\_loader = DataLoader(dataset=dataset, batch\_size=batch\_size, shuffle=True, num\_workers=0)

#check for gpu support

device=torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)

#loss and optimiser

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)

#training loop

for epoch in range(num\_epochs):

for (words, labels) in train\_loader:

words=words.to(device)

labels=labels.to(device)

#forward prop

outputs = model(words)

loss = criterion(outputs, labels.long()) #long dtype is expected for crossentropyloss

#backward prop and optimisation

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (epoch+1)%100 == 0:

print(f'epoch {epoch+1}/{num\_epochs}, loss={loss.item():.4f}')

print(f'final loss, loss={loss.item():.4f}')

data ={

"model\_state": model.state\_dict(),

"input\_size": input\_size,

"output\_size": output\_size,

"hidden\_size": hidden\_size,

"all\_words": all\_words,

"tags": tags

}

FILE = "data.pth"

torch.save(data, FILE)

print(f'Training complete. File saved to {FILE}')

the code for the chatbot interfacing with web page application the above codes created in the flask.