***TITLE : CREATE A CHATBOT IN PYTHON***

***PROJECT OVERVIEW :***

Here is a basic project overview for a chatbot using Python:

To build an interactive chatbot capable of understanding and responding to user queries effectively.

Individuals seeking a user-friendly interface for obtaining information or assistance in a specific domain.

Natural Language Processing (NLP) for understanding user input.

- Responses tailored to user queries using predefined knowledge base or machine learning models.

- Integration with external services for data retrieval or real-time information.

- Seamless user interaction and intuitive conversation flow.

- Data collection and preprocessing for training and testing.

- NLP model development (e.g., rule-based systems, retrieval-based models, or generative models).

- Integration with relevant APIs for external data handling.

- User interface development for a smooth interaction experience.

Milestones:

- Data gathering and preprocessing completion.

- Initial NLP model development and testing.

- Integration with external services for enhanced functionality.

- User interface implementation and testing.

- Deployment and beta testing for user feedback.

Risks and Mitigation:

- Data security and privacy concerns, addressed by implementing secure data handling protocols.

- Potential limitations in NLP model performance, mitigated through rigorous testing and model refinement.

- Integration challenges with external APIs, managed through thorough API documentation study and error handling procedures.

Deliverables:

- Fully functional chatbot script.

- Documentation detailing the development process, including data preprocessing, model training, and integration steps.

- User guide for the chatbot's functionalities and usage.

Timeline:

- Data collection and preprocessing: 1 week.

- NLP model development and testing: 2-3 weeks.

- Integration and user interface development: 1-2 weeks.

- Deployment and testing: 1 week.

- Beta testing and feedback incorporation: 1-2 weeks.

Stakeholders:

- Development team members, including NLP experts, Python developers, and UI/UX designers.

- End users providing feedback during the beta testing phase.

Budget:

Allocation for resources like human capital, cloud services, and potential external API usage, with regular reviews to ensure cost-effectiveness.

By following this project overview, you can structure the development process systematically and ensure a successful outcome for your chatbot project.

***PROGRAM:***

main.py:

import json

import re

import random\_responses

# Load JSON data

def load\_json(file):

with open(file) as bot\_responses:

print(f"Loaded '{file}' successfully!")

return json.load(bot\_responses)

# Store JSON data

response\_data = load\_json("bot.json")

def get\_response(input\_string):

split\_message = re.split(r'\s+|[,;?!.-]\s\*', input\_string.lower())

score\_list = []

# Check all the responses

for response in response\_data:

response\_score = 0

required\_score = 0

required\_words = response["required\_words"]

# Check if there are any required words

if required\_words:

for word in split\_message:

if word in required\_words:

required\_score += 1

# Amount of required words should match the required score

if required\_score == len(required\_words):

# print(required\_score == len(required\_words))

# Check each word the user has typed

for word in split\_message:

# If the word is in the response, add to the score

if word in response["user\_input"]:

response\_score += 1

# Add score to list

score\_list.append(response\_score)

# Debugging: Find the best phrase

# print(response\_score, response["user\_input"])

# Find the best response and return it if they're not all 0

best\_response = max(score\_list)

response\_index = score\_list.index(best\_response)

# Check if input is empty

if input\_string == "":

return "Please type something so we can chat :("

# If there is no good response, return a random one.

if best\_response != 0:

return response\_data[response\_index]["bot\_response"]

return random\_responses.random\_string()

while True:

user\_input = input("You: ")

print("Bot:", get\_response(user\_input))

bot.json:

[

{

"response\_type": "greeting",

"user\_input": ["hello", "hi", "hey"],

"bot\_response": "Hey there!",

"required\_words": []

},

{

"response\_type": "greeting",

"user\_input": ["see you", "goodbye", "bye"],

"bot\_response": "See you later!",

"required\_words": []

},

{

"response\_type": "greeting",

"user\_input": ["nice", "to", "meet", "you"],

"bot\_response": "The pleasure is all mine!",

"required\_words": ["nice", "meet", "you"]

},

{

"response\_type": "question",

"user\_input": ["how", "to", "learn", "code", "coding", "apps"],

"bot\_response": "Start by typing: 'How to learn coding' on Google.",

"required\_words": ["learn", "code"]

},

{

"response\_type": "question",

"user\_input": ["refund", "how", "can", "I", "get"],

"bot\_response": "We don't offer refunds for free education.",

"required\_words": ["refund", "I"]

"required\_words": ["refund", "i"]

},

{

"response\_type": "question",

"user\_input": ["how", "are", "you"],

"bot\_response": "I'm great! Thanks for asking.",

"required\_words": ["how", "are", "you"]

}

]

Random\_responses.py:

import random

def random\_string():

random\_list = [

"Please try writing something more descriptive.",

"Oh! It appears you wrote something I don't understand yet",

"Do you mind trying to rephrase that?",

"I'm terribly sorry, I didn't quite catch that.",

"I can't answer that yet, please try asking something else."

]

list\_count = len(random\_list)

random\_item = random.randrange(list\_count)

return random\_list[random\_item]

***OUTPUT:***

Loaded ‘bot.json’ successfully!

You : How are you?

Bot : I’m great! Thanks for asking.

You : I want to have a refund please!

Bot : We don’t offer refunds for free education.