Date:10.10.25

TASK:9

To Build an Intelligent **Chatbot system** with Python and Dialog-flow using Interactive Text Mining Framework for Exploration of Semantic Flows in Large Corpus of Text.

To Build an Intelligent Chatbot system with Python and Dialog-flow using Interactive Text Mining Framework for Exploration of Semantic Flows in Large Corpus of Text. **CO4 S3**

- To integrate with Google Cloud Speech-to-Text and third-party services such as Google Assistant, Amazon Alexa, and Facebook Messenger.
- Configure Dialogflow to manage your data across GCP services and let you optionally integrate Google Assistant.

Tools- Python, Dialog-flow Framework

TO BUILD AN INTELLIGENT **CHATBOT SYSTEM** WITH PYTHON AND DIALOG-FLOW USING INTERACTIVE TEXT MINING FRAMEWORK FOR EXPLORATION OF SEMANTIC FLOWS IN LARGE CORPUS OF TEXT

AIM:

To build an intelligent chatbox system with Python and dialog-flow using interactive text mining framework for exploration of semantic flow in large corpus of Text

ALGORITHM:

Steps to create an intelligent chatbot using OpenAI APIs:

1. Sign up for OpenAI API access at https://beta.openai.com/signup/. Once you sign up, you will receive your API key.

2. Choose the type of chatbot you want to create. For example, you can create an FAQ chatbot, a customer support chatbot, or a conversational chatbot.

3. Use OpenAI's GPT-3 language model to generate responses to user input. You can use the API to train the language model on your chatbot's intended use case/s.

4. Use Natural Language Processing (NLP) techniques to understand user input and provide relevant responses. You can use OpenAI's API to extract entities (such as dates and names) from user input.

5. Use Machine Learning to continually improve the chatbot's ability to understand and respond to user input.

6. Integrate the chatbot with your preferred messaging platform or channel (e.g., web chat, social media, etc.) using API connectors.

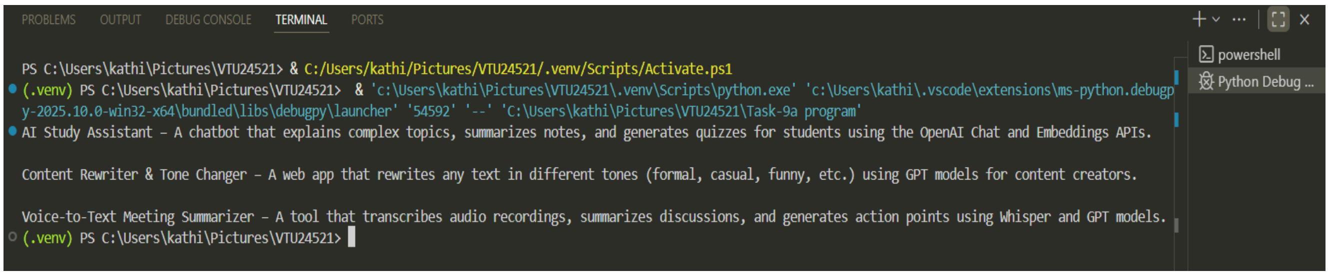
7. Test your chatbot frequently, and use user feedback to improve its performance and provide the best possible experience for your users.

A. **SIMPLE CHATGPT USING OPENAI**

CODE:

```
Pip install openai

import openai

openai.api_key = "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"

completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[{"role": "user", "content": "Give me 3 ideas that i could build using openai apis"}])

print(completion.choices[0].message.content)
```

OUTPUT:



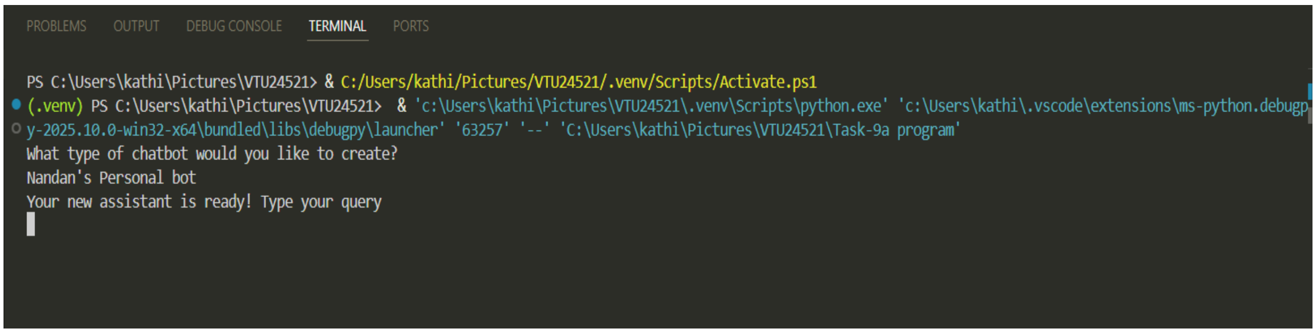
```
PS C:\Users\kathi\Pictures\VTU24521> & C:/Users/kathi/Pictures/VTU24521/.venv/Scripts/Activate.ps1
(.venv) PS C:\Users\kathi\Pictures\VTU24521>  & 'c:\Users\kathi\Pictures\VTU24521\.venv\Scripts\python.exe' 'c:\Users\kathi\.vscode\extensions\ms-python.debugp
y-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '54592' '--' 'C:\Users\kathi\Pictures\VTU24521\Task-9a program'
AI Study Assistant – A chatbot that explains complex topics, summarizes notes, and generates quizzes for students using the OpenAI Chat and Embeddings APIs.

Content Rewriter & Tone Changer – A web app that rewrites any text in different tones (formal, casual, funny, etc.) using GPT models for content creators.

Voice-to-Text Meeting Summarizer – A tool that transcribes audio recordings, summarizes discussions, and generates action points using Whisper and GPT models.
(.venv) PS C:\Users\kathi\Pictures\VTU24521>
```

B. CHATGPT ASSISTANT USING OPENAI

CODE:

```
import openai


openai.api_key = "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"


messages = []
system_msg = input("What type of chatbot would you like to create?\n")
messages.append({"role": "system", "content": system_msg})


print("Your new assistant is ready! Type your query")
while input != "quit()":
    message = input()
    messages.append({"role": "user", "content": message})
    response = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=messages)
    reply = response["choices"][0]["message"]["content"]
    messages.append({"role": "assistant", "content": reply})
    print("\n" + reply + "\n")
```

OUTPUT:



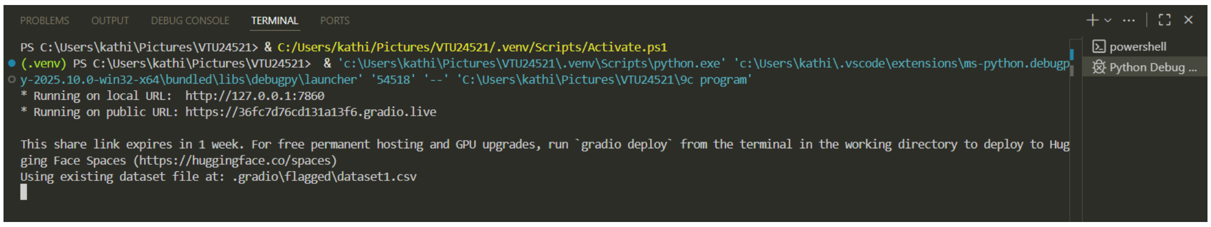
```
PS C:\Users\kathi\Pictures\VTU24521> & C:/Users/kathi/Pictures/VTU24521/.venv/Scripts/Activate.ps1
(.venv) PS C:\Users\kathi\Pictures\VTU24521>  & 'c:\Users\kathi\Pictures\VTU24521\.venv\Scripts\python.exe' 'c:\Users\kathi\.vscode\extensions\ms-python.debugp
y-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63257' '--' 'C:\Users\kathi\Pictures\VTU24521\Task-9a program'
What type of chatbot would you like to create?
Nandan's Personal bot
Your new assistant is ready! Type your query
█
```

C. **CHATBOT CHAT ASSISTANT WEBSITE**

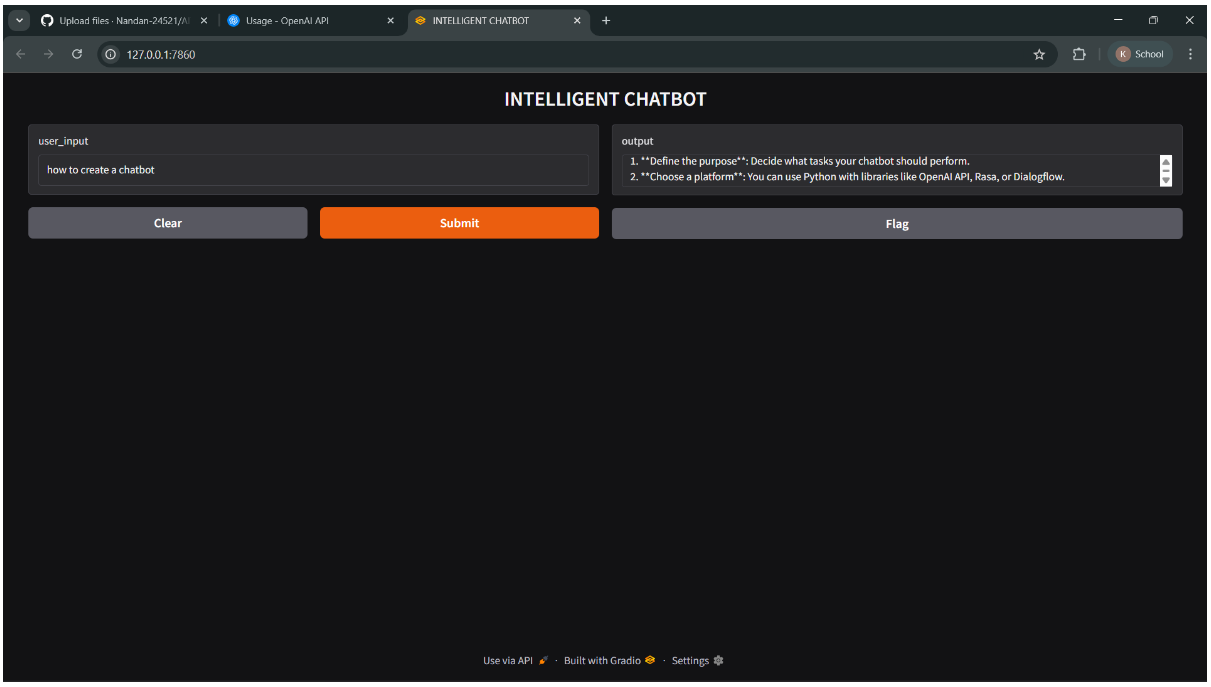
CODE:

```
import openai

import gradio

openai.api_key = "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"

messages = [{"role": "system", "content": "You are a financial experts that specializes in real estate investment and negotiation"}]

def CustomChatGPT(user_input):

    messages.append({"role": "user", "content": user_input})

    response = openai.ChatCompletion.create(

        model = "gpt-3.5-turbo",

        messages = messages

    )

    ChatGPT_reply = response["choices"][0]["message"]["content"]

    messages.append({"role": "assistant", "content": ChatGPT_reply})

    return ChatGPT_reply

demo = gradio.Interface(fn=CustomChatGPT, inputs = "text", outputs = "text", title = "INTELLIGENT CHATBOT")

demo.launch(share=True)
```

OUTPUT:



```
PS C:\Users\kathi\Pictures\VTU24521> & C:/Users/kathi/Pictures/VTU24521/.venv/Scripts/Activate.ps1
(.venv) PS C:\Users\kathi\Pictures\VTU24521>  & 'c:\Users\kathi\Pictures\VTU24521\.venv\Scripts\python.exe' 'c:\Users\kathi\.vscode\extensions\ms-python.debugp
y-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '54518' '--' 'C:\Users\kathi\Pictures\VTU24521\9c program'
 * Running on local URL:  http://127.0.0.1:7860
 * Running on public URL: https://36fc7d76cd131a13f6.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hug
ging Face Spaces (https://huggingface.co/spaces)
Using existing dataset file at: .gradio\flagged\dataset1.csv
```


Upload files · Nandan-24521/AI
Usage - OpenAI API
INTELLIGENT CHATBOT
127.0.0.1:7860
School
Clear
Submit
Flag
Use via API · Built with Gradio · Settings

INTELLIGENT CHATBOT

user_input

how to create a chatbot

output

1. **Define the purpose**: Decide what tasks your chatbot should perform.
2. **Choose a platform**: You can use Python with libraries like OpenAI API, Rasa, or Dialogflow.

RESULT:

Thus, to build an intelligent chatbox system with Python and dialogue flow was successfully completed and output was verified.