AI PHASE 4

CREATE A CHATBOT USING PYTHON

SYNOPSIS:

1.Introduction:

- This document presents an overview of a Python web application that implements a chatbot using Flask, SpaCy, and the GPT-2 language model.
- The application allows users to interact with a chatbot either by querying a pre-processed dataset or generating responses using GPT-2 for unseen queries.

2.Technologies Used:

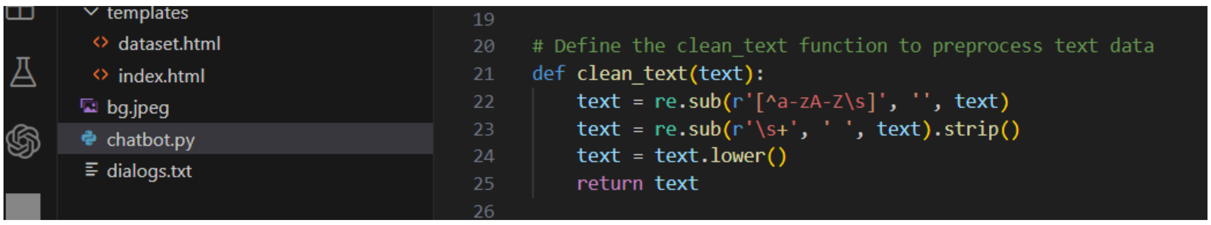
- Flask: A Python web framework used for building the web application.
- SpaCy: An open-source software library for advanced natural language processing in Python. GPT-2 (Generative Pre-trained Transformer 2): A state-of-the-art language processing AI model developed by OpenAI.
- Transformers Library: A library for natural language processing using pre-trained models like GPT-2. Pandas: A data manipulation and analysis library for Python.
- HTML and CSS: For designing and structuring the web interface.

3.loading and pre-processing the dataset:

The dataset is loaded from the file **'dialogs.txt'** using Pandas. The dataset is assumed to have a tab-separated format with two columns: 'question' and 'answer'. To prepare the text data for processing, the clean text function is defined. Removes non-alphabetic characters from the text, Replaces multiple spaces with a single space and strips leading/trailing spaces, Converts the text to lowercase for uniformity.

1.Preprocessing:

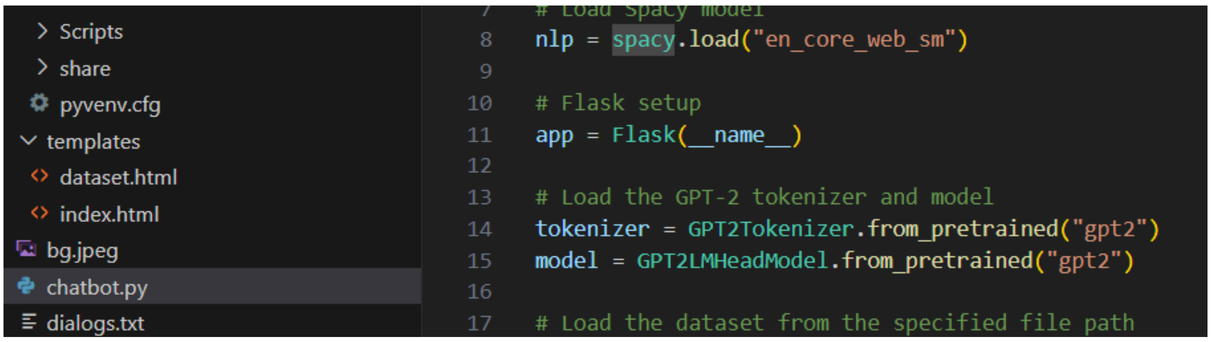
Purpose: Preprocessing is the initial step to clean and transform raw text data into a format suitable for analysis or modelling. It involves removing unnecessary characters, converting text to lowercase, and handling other forms of noise in the data.
Implementation: In the provided code, the clean text function is responsible for preprocessing. It uses regular expressions to remove non-alphabetic characters, replaces multiple spaces with a single space, and converts the text to lowercase. This ensures consistency and uniformity in the textual data.



```python

# Define the clean_text function to preprocess text data
def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    text = text.lower()
    return text

```

2. Tokenization:

Purpose: Tokenization is the process of breaking down text into smaller units, such as words or sub-words. Tokens are the building blocks for language processing tasks.
Implementation: The GPT-2 tokenizer (GPT2Tokenizer.from_pretrained("gpt2")) is used to tokenize the user's input. Tokenization converts a sentence into a sequence of integers (token IDs) which can be fed into the neural network for processing.

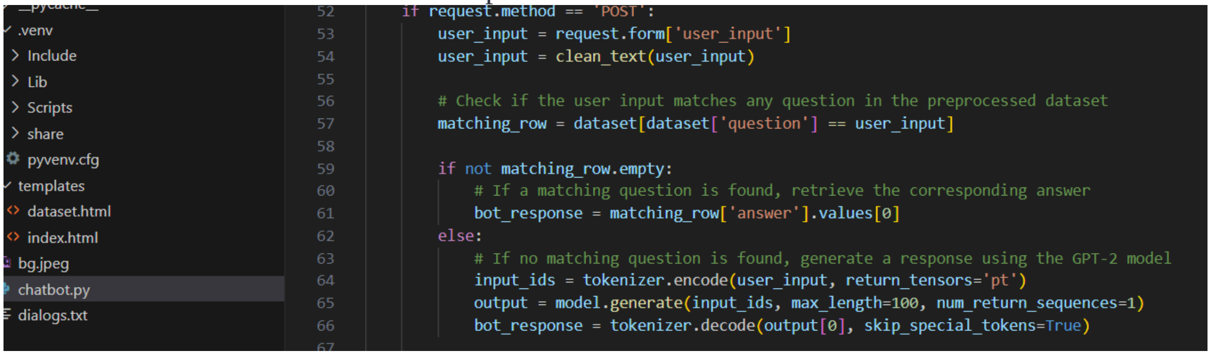


```python
# Load Spacy model
nlp = spacy.load("en_core_web_sm")

# Flask setup
app = Flask(__name__)

# Load the GPT-2 tokenizer and model
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Load the dataset from the specified file path
```

3. Encoding & Decoding:

Encoding refers to converting text tokens into numerical values. Neural networks process numerical data, so text data needs to be encoded into a format that the model can understand.

Decoding is the inverse process of encoding. It converts numerical values back into human-readable text. After the model generates predictions, the output needs to be decoded to obtain the final textual response.

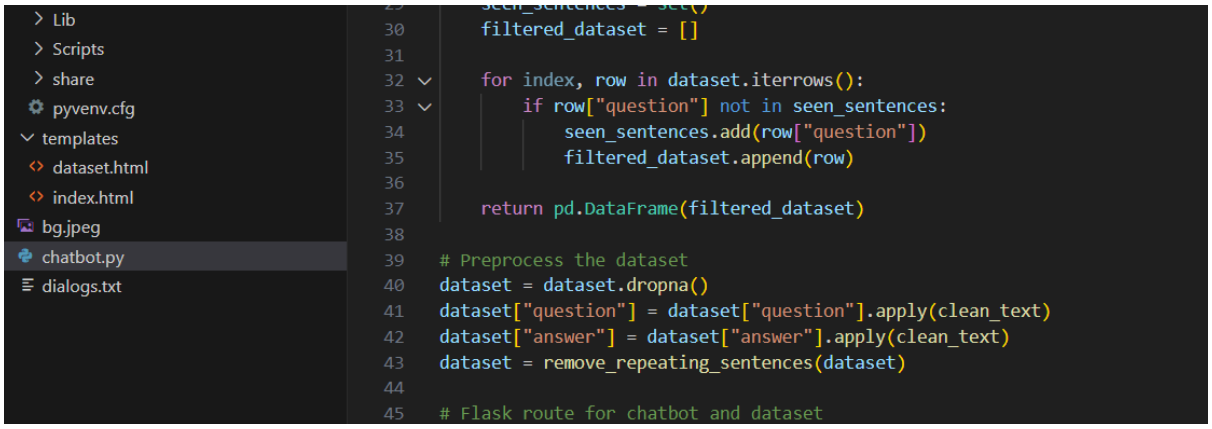


```
    if request.method == 'POST':
        user_input = request.form['user_input']
        user_input = clean_text(user_input)

        # Check if the user input matches any question in the preprocessed dataset
        matching_row = dataset[dataset['question'] == user_input]

        if not matching_row.empty:
            # If a matching question is found, retrieve the corresponding answer
            bot_response = matching_row['answer'].values[0]
        else:
            # If no matching question is found, generate a response using the GPT-2 model
            input_ids = tokenizer.encode(user_input, return_tensors='pt')
            output = model.generate(input_ids, max_length=100, num_return_sequences=1)
            bot_response = tokenizer.decode(output[0], skip_special_tokens=True)
```

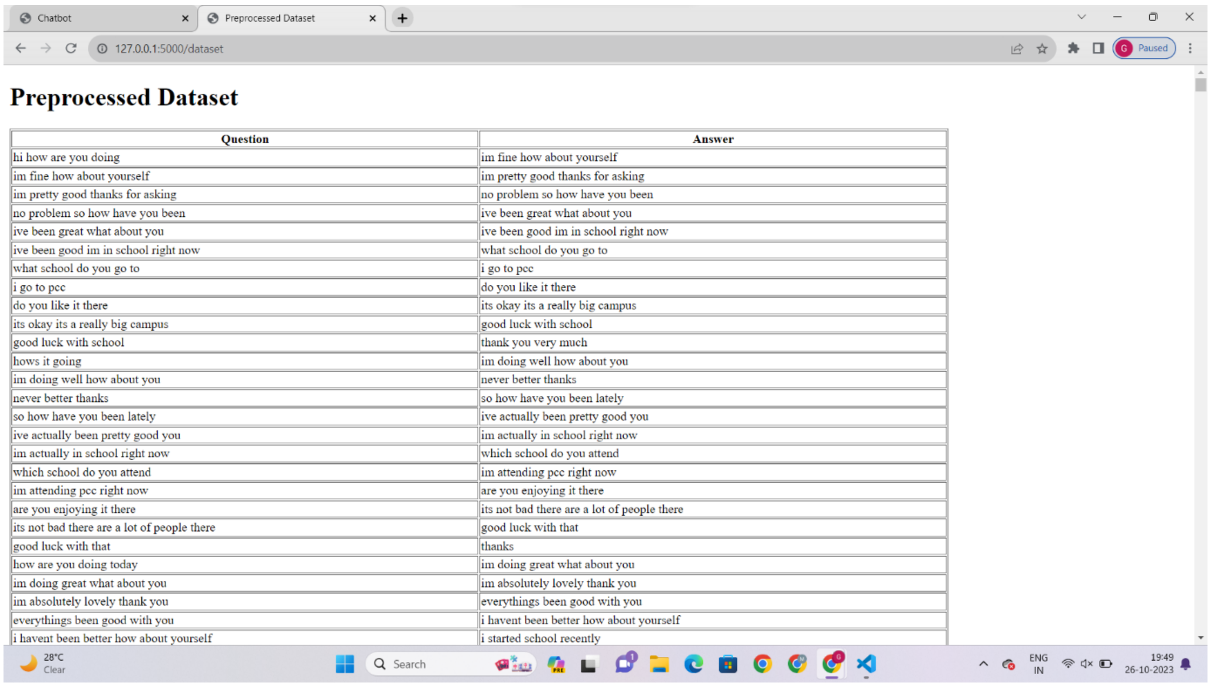
4.Data split:

Data split in this occurred in the per-processed dataset (). It is in the form of Question and Answer table. The chatbot only accepted the data from this dataset only.

NOTE: You can access the dataset with the help of this address http://127.0.0.1:5000/dataset

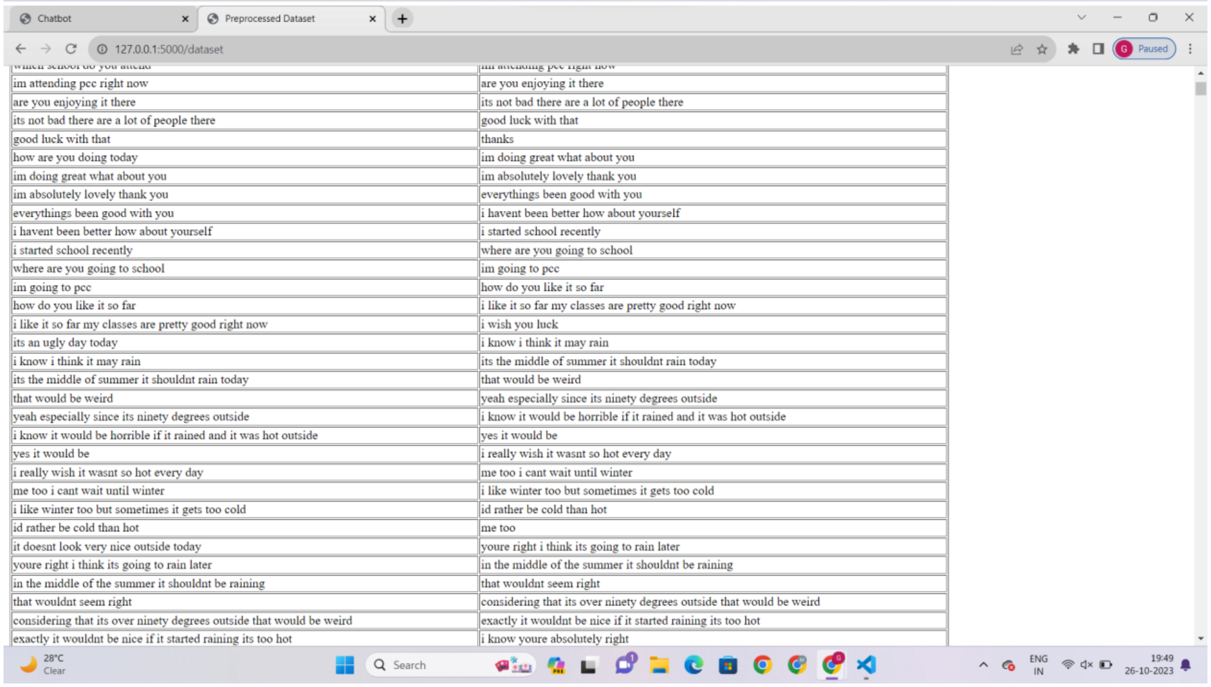


```
    seen_sentences = set()
    filtered_dataset = []

    for index, row in dataset.iterrows():
        if row["question"] not in seen_sentences:
            seen_sentences.add(row["question"])
            filtered_dataset.append(row)

    return pd.DataFrame(filtered_dataset)

# Preprocess the dataset
dataset = dataset.dropna()
dataset["question"] = dataset["question"].apply(clean_text)
dataset["answer"] = dataset["answer"].apply(clean_text)
dataset = remove_repeating_sentences(dataset)

# Flask route for chatbot and dataset
```



Preprocessed Dataset

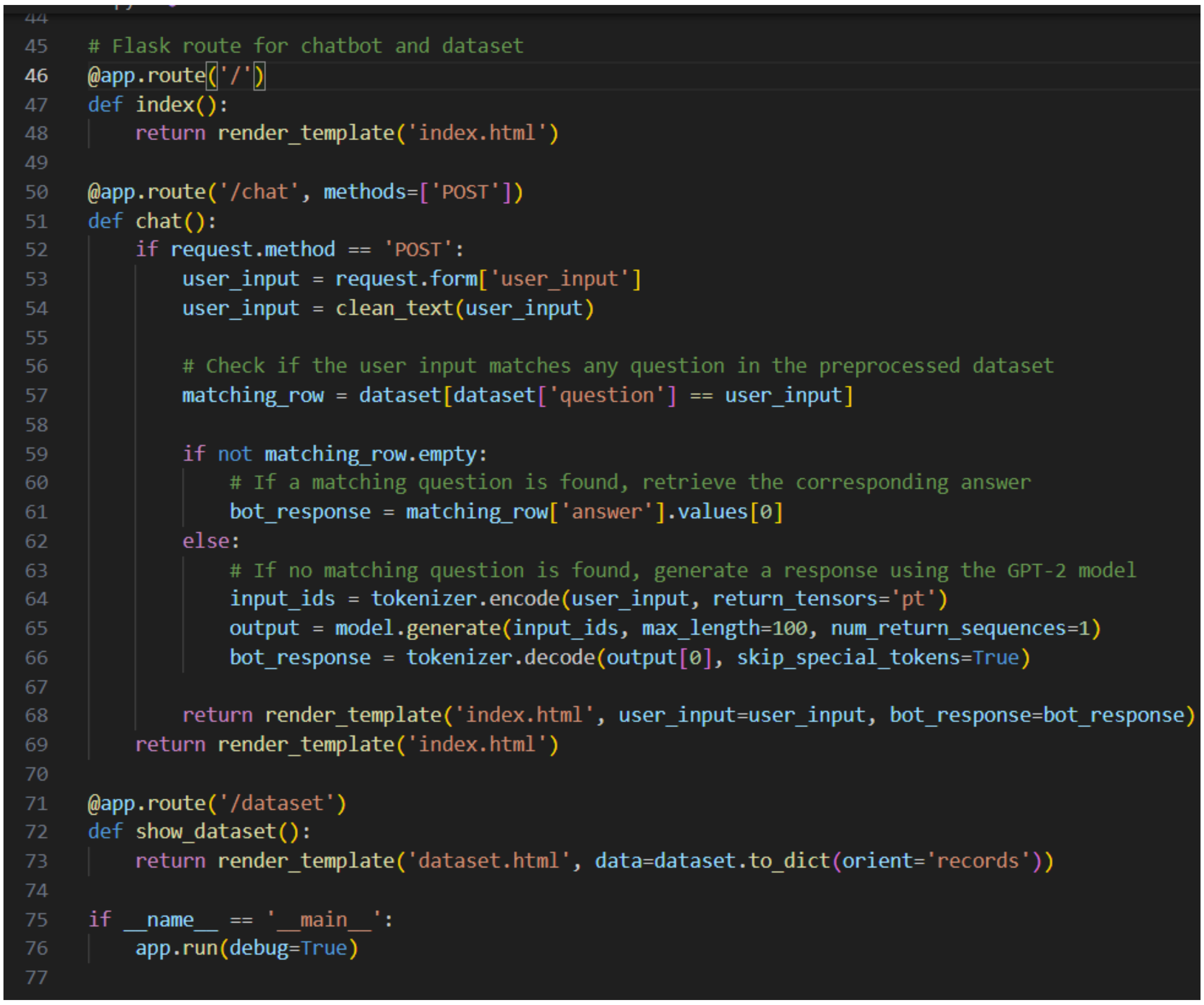
Question	Answer
hi how are you doing	im fine how about yourself
im fine how about yourself	im pretty good thanks for asking
im pretty good thanks for asking	no problem so how have you been
no problem so how have you been	ive been great what about you
ive been great what about you	ive been good im in school right now
ive been good im 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	its okay its a really big campus
its okay its a really big campus	good luck with school
good luck with school	thank you very much
hows it going	im doing well how about you
im doing well how about you	never better thanks
never better thanks	so how have you been lately
so how have you been lately	ive actually been pretty good you
ive actually been pretty good you	im actually in school right now
im actually in school right now	which school do you attend
which school do you attend	im attending pcc right now
im attending pcc right now	are you enjoying it there
are you enjoying it there	its not bad there are a lot of people there
its not bad there are a lot of people there	good luck with that
good luck with that	thanks
how are you doing today	im doing great what about you
im doing great what about you	im absolutely lovely thank you
im absolutely lovely thank you	everythings been good with you
everythings been good with you	i havent been better how about yourself
i havent been better how about yourself	i started school recently



Question	Answer
im attending pcc right now	are you enjoying it there
are you enjoying it there	its not bad there are a lot of people there
its not bad there are a lot of people there	good luck with that
good luck with that	thanks
how are you doing today	im doing great what about you
im doing great what about you	im absolutely lovely thank you
im absolutely lovely thank you	everythings been good with you
everythings been good with you	i havent been better how about yourself
i havent been better how about yourself	i started school recently
i started school recently	where are you going to school
where are you going to school	im going to pcc
im going to pcc	how do you like it so far
how do you like it so far	i like it so far my classes are pretty good right now
i like it so far my classes are pretty good right now	i wish you luck
its an ugly day today	i know i think it may rain
i know i think it may rain	its the middle of summer it shouldnt rain today
its the middle of summer it shouldnt rain today	that would be weird
that would be weird	yeah especially since its ninety degrees outside
yeah especially since its ninety degrees outside	i know it would be horrible if it rained and it was hot outside
i know it would be horrible if it rained and it was hot outside	yes it would be
yes it would be	i really wish it wasnt so hot every day
i really wish it wasnt so hot every day	me too i cant wait until winter
me too i cant wait until winter	i like winter too but sometimes it gets too cold
i like winter too but sometimes it gets too cold	id rather be cold than hot
id rather be cold than hot	me too
it doesnt look very nice outside today	youre right i think its going to rain later
youre right i think its going to rain later	in the middle of the summer it shouldnt be raining
in the middle of the summer it shouldnt be raining	that wouldnt seem right
that wouldnt seem right	considering that its over ninety degrees outside that would be weird
considering that its over ninety degrees outside that would be weird	exactly it wouldnt be nice if it started raining its too hot
exactly it wouldnt be nice if it started raining its too hot	i know youre absolutely right

4. Integration with flask:

The pre-processed dataset is integrated with the Flask web application. Flask routes are set up to handle user input and responses. When a user submits a query, it is cleaned using the clean text function. The cleaned query is then compared with the pre-processed dataset to find a matching question. If no match is found in the pre-processed dataset, the cleaned user query is tokenized using the GPT-2 tokenizer. The tokenized input is fed into the GPT-2 model, which generates a response. The response is then decoded using the tokenizer to obtain the final bot response. The generated response is displayed to the user through the web interface. This model and tokenizer are used in GPT-3 also.



```python
# Flask route for chatbot and dataset
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/chat', methods=['POST'])
def chat():
    if request.method == 'POST':
        user_input = request.form['user_input']
        user_input = clean_text(user_input)

        # Check if the user input matches any question in the preprocessed dataset
        matching_row = dataset[dataset['question'] == user_input]

        if not matching_row.empty:
            # If a matching question is found, retrieve the corresponding answer
            bot_response = matching_row['answer'].values[0]
        else:
            # If no matching question is found, generate a response using the GPT-2 model
            input_ids = tokenizer.encode(user_input, return_tensors='pt')
            output = model.generate(input_ids, max_length=100, num_return_sequences=1)
            bot_response = tokenizer.decode(output[0], skip_special_tokens=True)

        return render_template('index.html', user_input=user_input, bot_response=bot_response)
    return render_template('index.html')

@app.route('/dataset')
def show_dataset():
    return render_template('dataset.html', data=dataset.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(debug=True)
```

5.Web interface:

The user interface is designed using HTML and CSS. The main page **(index.html)** includes an input field for user queries and displays both user inputs and bot responses. The dataset page **(dataset.html)** displays the pre-processed dataset. The web interface is designed to take user input, display user queries, and show bot responses. Users interact with the chatbot by typing questions into an input field and receiving responses in real-time.

6. How to run and interact with chatbot:

- Ensure all required libraries and dependencies are installed.
- Run the Python script in a local environment.
- Access the application through a web browser.
- Enter queries and interact with the chatbot.

Here is the codes for the chatbot and interfaces:

(dataset.html)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Preprocessed Dataset</title>
</head>
<body>
    <h1>Preprocessed Dataset</h1>
    <table border="1">
        <thead>
            <tr>
                <th>Question</th>
                <th>Answer</th>
            </tr>
        </thead>
        <tbody>
            {% for row in data %}
                <tr>
                    <td>{{ row.question }}</td>
                    <td>{{ row.answer }}</td>
                </tr>
            {% endfor %}
        </tbody>
```

```html
</table>
</body>
</html>
```

(index.html)

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chatbot</title>
    <style>
        body {
            display: flex;
            background-color: #007bff;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .container {

            max-width: 400px;
            width: 100%;
            padding: 20px;
            background-color: #ffffff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }

        .message {
            margin-bottom: 10px;
        }

        .user-message strong {
            color: #007bff;
        }

        .bot-message strong {
            color: #28a745;
        }
    </style>
```

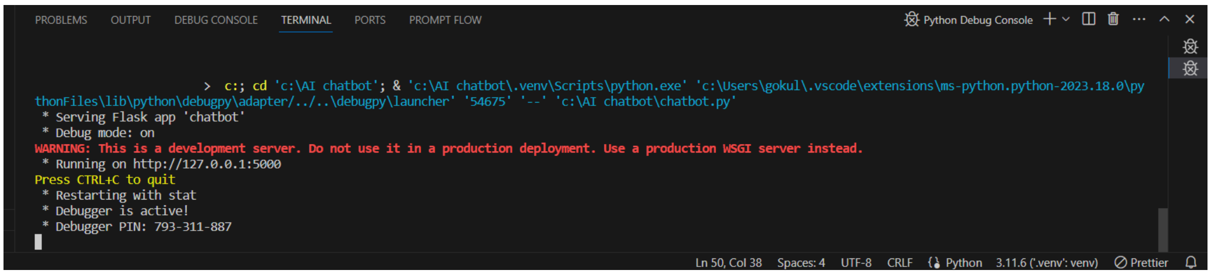
```html
</head>

<body>

    <div class="container">
        <h1 style="text-align: center;">Chatbot</h1>
        <div class="message user-message">
            <strong>You:</strong> {{ user_input }}
        </div>
        <div class="message bot-message">
            <strong>Bot:</strong> {{ bot_response }}
        </div>
        <form method="POST" action="/chat" style="text-align: center;">
            <label for="user_input">You:</label>
            <input type="text" id="user_input" name="user_input" value="{{
user_input }}">
            <input type="submit" value="Ask">
        </form>
    </div>
</body>

</html>
```

(Chatbot.py)

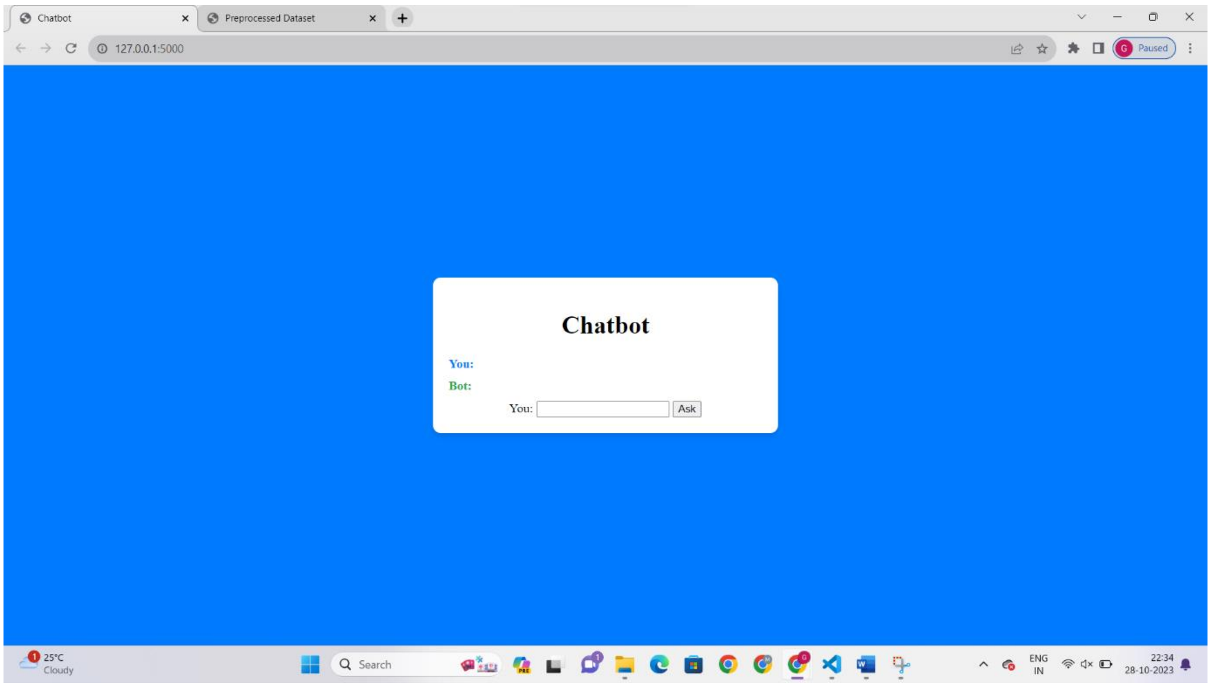
```python
import re
import pandas as pd
import spacy
from flask import Flask, render_template, request
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load SpaCy model
nlp = spacy.load("en_core_web_sm")

# Flask setup
app = Flask(__name__)

# Load the GPT-2 tokenizer and model
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Load the dataset from the specified file path
dataset = pd.read_csv('dialogs.txt', delimiter="\t", header=None,
names=["question", "answer"])

# Define the clean_text function to preprocess text data
def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    text = text.lower()
    return text

# Define the remove_repeating_sentences function to remove repeating sentences
from a dataset
def remove_repeating_sentences(dataset):
    seen_sentences = set()
    filtered_dataset = []

    for index, row in dataset.iterrows():
        if row["question"] not in seen_sentences:
            seen_sentences.add(row["question"])
            filtered_dataset.append(row)

    return pd.DataFrame(filtered_dataset)

# Preprocess the dataset
dataset = dataset.dropna()
dataset["question"] = dataset["question"].apply(clean_text)
dataset["answer"] = dataset["answer"].apply(clean_text)
dataset = remove_repeating_sentences(dataset)
```

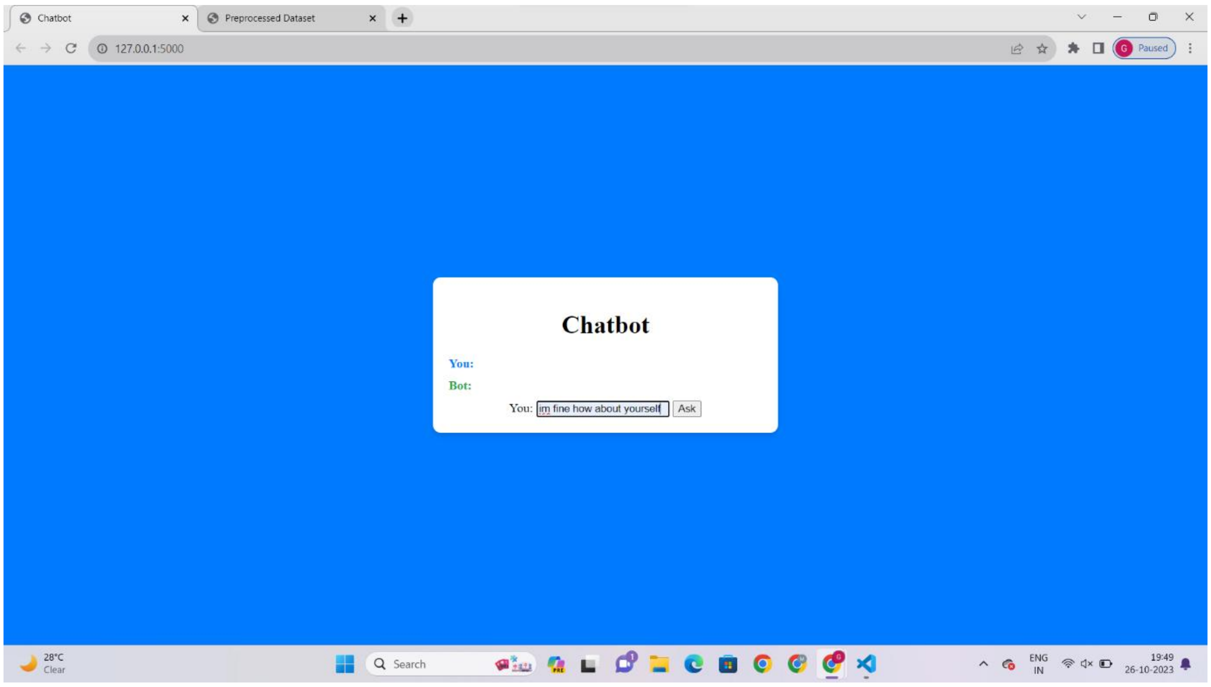
```python
# Flask route for chatbot and dataset
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/chat', methods=['POST'])
def chat():
    if request.method == 'POST':
        user_input = request.form['user_input']
        user_input = clean_text(user_input)

        # Check if the user input matches any question in the preprocessed
dataset
        matching_row = dataset[dataset['question'] == user_input]

        if not matching_row.empty:
            # If a matching question is found, retrieve the corresponding
answer
            bot_response = matching_row['answer'].values[0]
        else:
            # If no matching question is found, generate a response using the
GPT-2 model
            input_ids = tokenizer.encode(user_input, return_tensors='pt')
            output = model.generate(input_ids, max_length=100,
num_return_sequences=1)
            bot_response = tokenizer.decode(output[0],
skip_special_tokens=True)

        return render_template('index.html', user_input=user_input,
bot_response=bot_response)
    return render_template('index.html')

@app.route('/dataset')
def show_dataset():
    return render_template('dataset.html',
data=dataset.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT:



```
                       > c:; cd 'c:\AI chatbot'; & 'c:\AI chatbot\.venv\Scripts\python.exe' 'c:\Users\gokul\.vscode\extensions\ms-python.python-2023.18.0\py
thonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '54675' '--' 'c:\AI chatbot\chatbot.py'
 * Serving Flask app 'chatbot'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 793-311-887
```



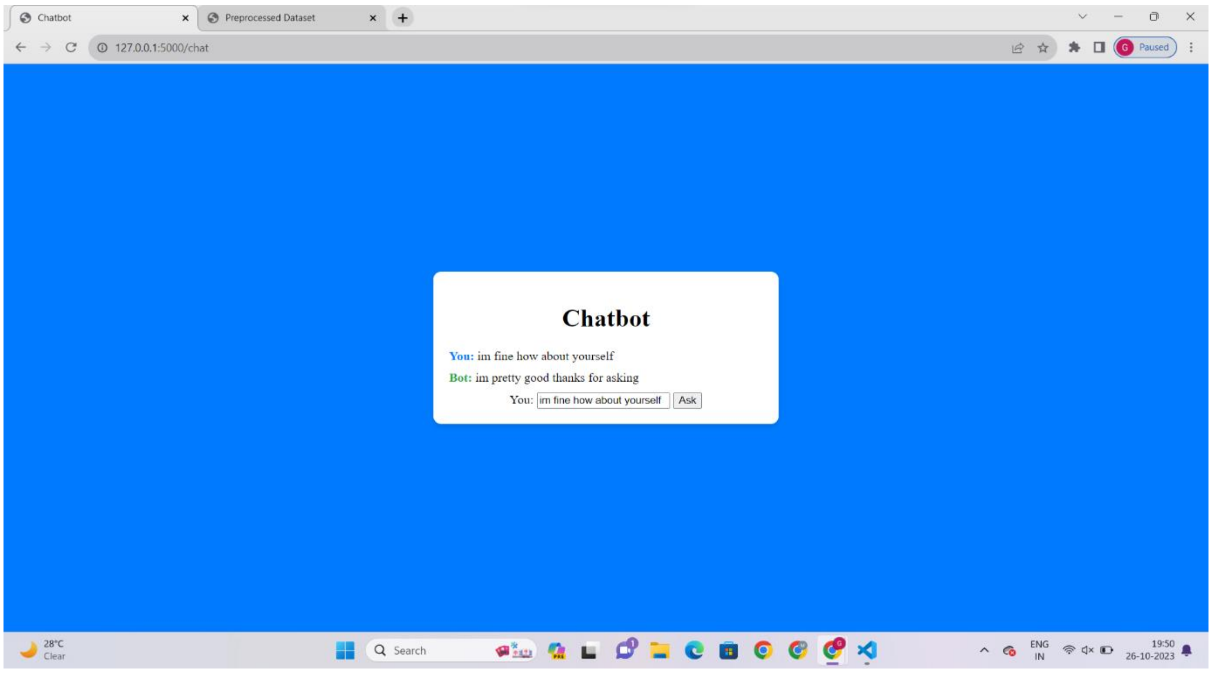


Chatbot

You:

Bot:

You: im fine how about yourself Ask

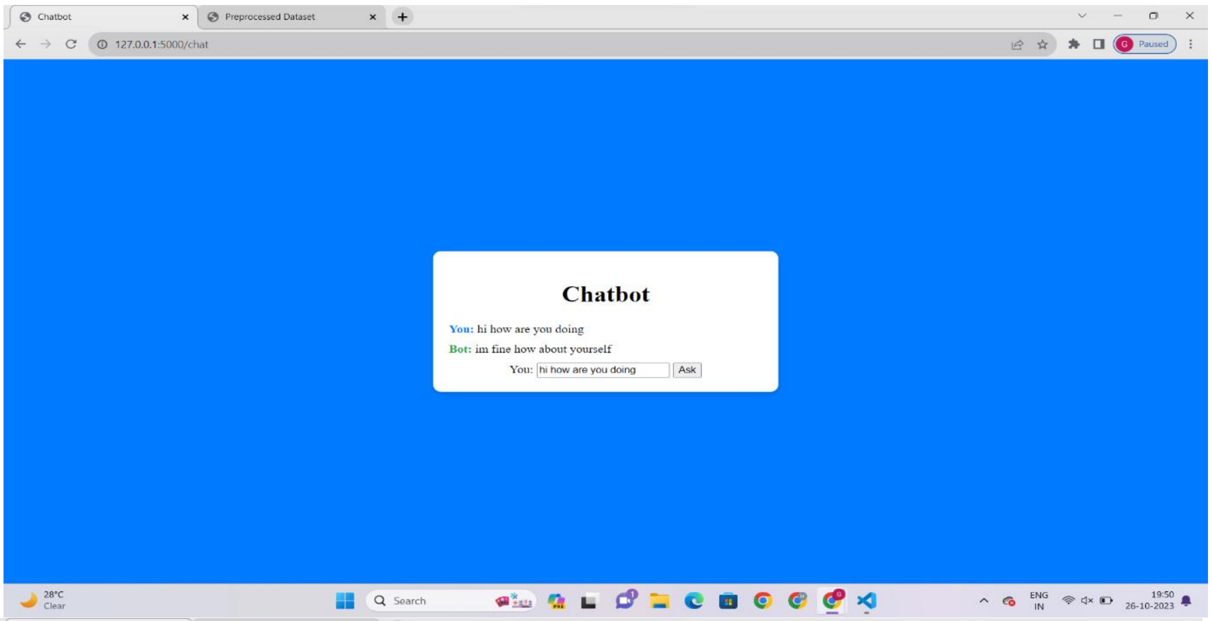


Chatbot

You: im fine how about yourself

Bot: im pretty good thanks for asking

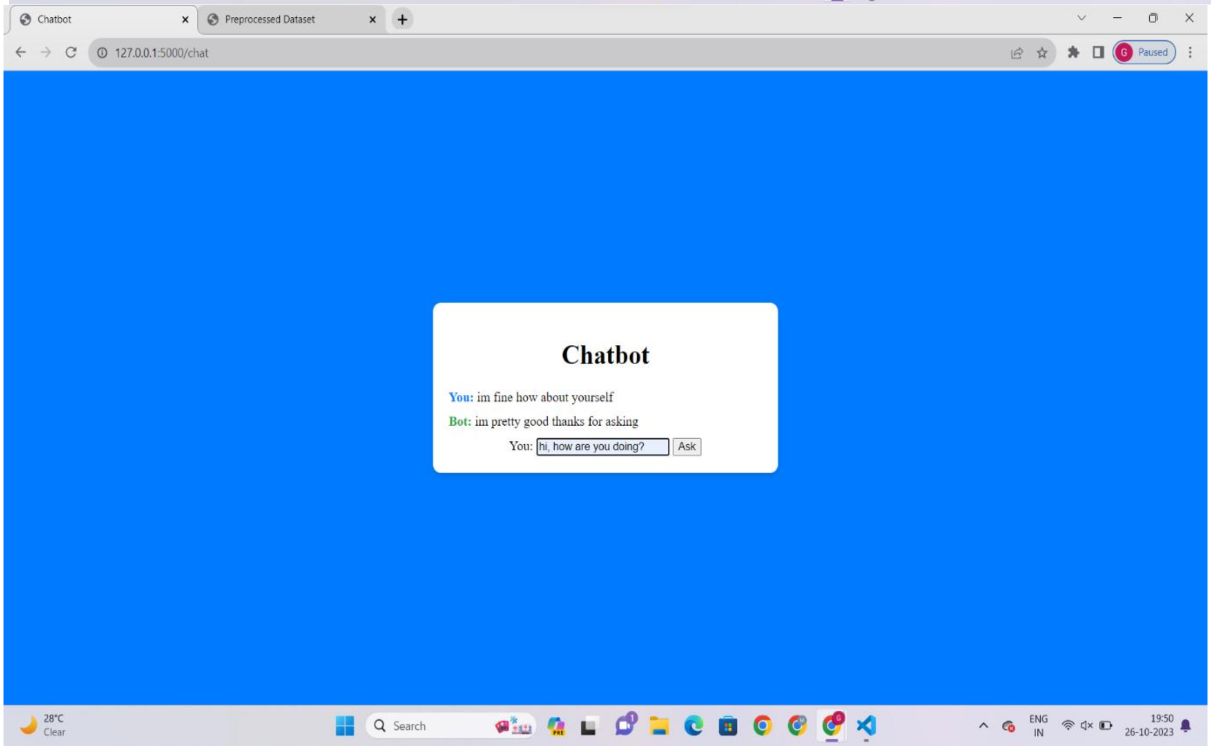
You: im fine how about yourself Ask


Chatbot

You: hi how are you doing

Bot: im fine how about yourself

You: hi how are you doing


Chatbot

You: im fine how about yourself

Bot: im pretty good thanks for asking

You: hi, how are you doing?