Code in app.py

```
from flask import Flask, render_template, jsonify, request

from flask_socketio import SocketIO, emit

import azure.cognitiveservices.speech as speechsdk

import openai

import threading

import os


app = Flask(__name__)

socketio = SocketIO(app)


SPEECH_KEY = 'speech-key'

SPEECH_REGION = 'speech-region'

OPENAI_API_KEY = 'openai-api-key'

openai.api_key = OPENAI_API_KEY


# Function to handle continuous speech recognition

def continuous_recognition(stop_event):

    speech_config = speechsdk.SpeechConfig(subscription=SPEECH_KEY, region=SPEECH_REGION)

    speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config)


    def recognized(args):

        if args.result.text:

            text = args.result.text.lower().strip()

            print(f"Recognized: {text}")

            socketio.emit('recognized_text', {'text': text})  # Emit recognized text to the client

            if text in ["exit", "bye"]:

                stop_event.set()  # Signal to stop listening
```

```
            speech_recognizer.stop_continuous_recognition()
        else:
            socketio.emit('listening_state', {'state': False})  # Indicate that mic is not listening
            response = generate_response_with_gpt_and_speak(text)
            print("GPT-3 Response:", response)
            socketio.emit('gpt_response', {'response': response})  # Emit GPT-3 response to the client
            speak_text(response)
            socketio.emit('listening_state', {'state': True})  # Indicate that mic can resume listening

    speech_recognizer.recognized.connect(recognized)
    speech_recognizer.start_continuous_recognition()

    while not stop_event.is_set():
        pass  # Keep running until stop event is triggered

    speech_recognizer.stop_continuous_recognition()

# Rest of your code...

def generate_response_with_gpt_and_speak(text):
    # Define the prompt for GPT
    full_prompt = f"""
    only give the correct answer to my questions. Do not answer irrelevant. Answer each question to the point.
    Question: {text}
    Answer:"""

    try:
        response = openai.Completion.create(
```

```
            engine="gpt-3.5-turbo-instruct",
            prompt=full_prompt,
            temperature=0.7,
            max_tokens=150,
            top_p=1.0,
            frequency_penalty=0.0,
            presence_penalty=0.0
        )
        response_text = response.choices[0].text.strip()
        print(f"GPT-3 response: {response_text}")
        return response_text
    except Exception as e:
        print(f"An error occurred while getting the response from GPT-3: {e}")
        return "I'm sorry, I couldn't process that."

def speak_text(text):
    speech_config = speechsdk.SpeechConfig(subscription=SPEECH_KEY, region=SPEECH_REGION)
    speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config)
    speech_synthesizer.speak_text_async(text).get()

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/start_listening', methods=['GET'])
def start_listening():
    stop_event = threading.Event()
    thread = threading.Thread(target=continuous_recognition, args=(stop_event,))
```

```
    thread.start()

    return jsonify({'message': 'Listening started. Say "exit" or "bye" to stop.'})

if __name__ == '__main__':

    socketio.run(app, debug=True, host="0.0.0.0", port=8000)
```

Requirement.txt

```
aiohttp==3.8.6

aiosignal==1.3.1

annotated-types==0.5.0

anyio==3.7.1

async-timeout==4.0.3

asynctest==0.13.0

attrs==23.2.0

azure-cognitiveservices-speech==1.36.0

bidict==0.22.1

cached-property==1.5.2

certifi==2024.2.2

charset-normalizer==3.3.2

click==8.1.7

colorama==0.4.6

distro==1.9.0

exceptiongroup==1.2.0

Flask==2.2.5

Flask-SocketIO==5.3.6

frozenlist==1.3.3
```

h11==0.14.0

httpcore==0.17.3

httpx==0.24.1

idna==3.6

importlib-metadata==6.7.0

itsdangerous==2.1.2

Jinja2==3.1.3

MarkupSafe==2.1.5

multidict==6.0.5

openai==0.28.0

pydantic==2.5.3

pydantic-core==2.14.6

python-engineio==4.9.0

python-socketio==5.11.0

pytz==2024.1

requests==2.31.0

simple-websocket==1.0.0

sniffio==1.3.1

tqdm==4.66.2

typing-extensions==4.7.1

urllib3==2.0.7

Werkzeug==2.2.3

wsproto==1.2.0

yarl==1.9.4

zipp==3.15.0

HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Speech Interaction</title>
    <style>
        body, html {
            height: 100%;
            margin: 0;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            text-align: center;
        }
        .container {
            width: 80%;
            max-width: 600px;
        }
        h2 {
            margin-bottom: 20px;
        }
        button {
            padding: 10px 20px;
```

```
            font-size: 16px;
            border: none;
            background-color: #007bff;
            color: white;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
        #messages {
            list-style-type: none;
            text-align: left;
            padding: 0;
        }
        #messages li {
            margin-bottom: 10px;
        }
        .recognized-text {
            color: #28a745; /* Green color for recognized text */
        }
        .gpt-response {
            color: #007bff; /* Blue color for GPT-3 response */
        }
    </style>
</head>
<body>
    <div class="container">
```

```
    <h2>To stop the process/listening, say "exit" or "bye".</h2>
    <button onclick="startListening()">Start Listening</button>
    <p id="status">Click "Start Listening" to begin.</p>
    <ul id="messages"></ul>
</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.1/socket.io.js"></script>
<script>
    var socket = io();

    socket.on('recognized_text', function(data) {
        var message = document.createElement('li');
        message.textContent = 'Recognized: ' + data.text;
        message.classList.add('recognized-text');
        document.getElementById('messages').appendChild(message);
    });

    socket.on('gpt_response', function(data) {
        var message = document.createElement('li');
        message.textContent = 'GPT-3 Response: ' + data.response;
        message.classList.add('gpt-response');
        document.getElementById('messages').appendChild(message);
    });

    // Listen for listening_state events to update the UI
    socket.on('listening_state', function(data) {
        if (data.state) {
            document.getElementById('status').innerText = 'Listening... Say "exit" or "bye" to stop.';
```

```
        } else {
            document.getElementById('status').innerText = 'Processing... Mic is not listening.';
        }
    });

    function startListening() {
        fetch("/start_listening").then(response => response.json()).then(data => {
            document.getElementById('status').innerText = 'Listening... Say "exit" or "bye" to stop.';
        }).catch(error => {
            console.log(error);
        });
    }
  </script>
</body>
</html>
```