Simon Tesfatsion

SFBU Customer Support System
Speech to Text to Speech

GItHUB LInk

Google Slides Link

OVERVIEW

Introduction

In this project, I developed an interactive assistant capable of:

- Converting user speech to text.
- Processing the text using OpenAI's GPT model for intelligent responses.
- Converting the generated text into speech for conversational interaction.

The system leverages a React-based frontend and a Python-powered backend to provide a seamless user experience.

Environment Setup

Frontend:

- Built using React with Node.js for dependency management.
- Libraries: axios for API communication, CSS for styling.

Backend:

- Python-based backend using Flask for API endpoints.
- Key Python libraries:
  - Speech Recognition and OpenAI's Whisper for speech-to-text.
  - gTTS for text-to-speech.
  - OpenAI for GPT API integration.

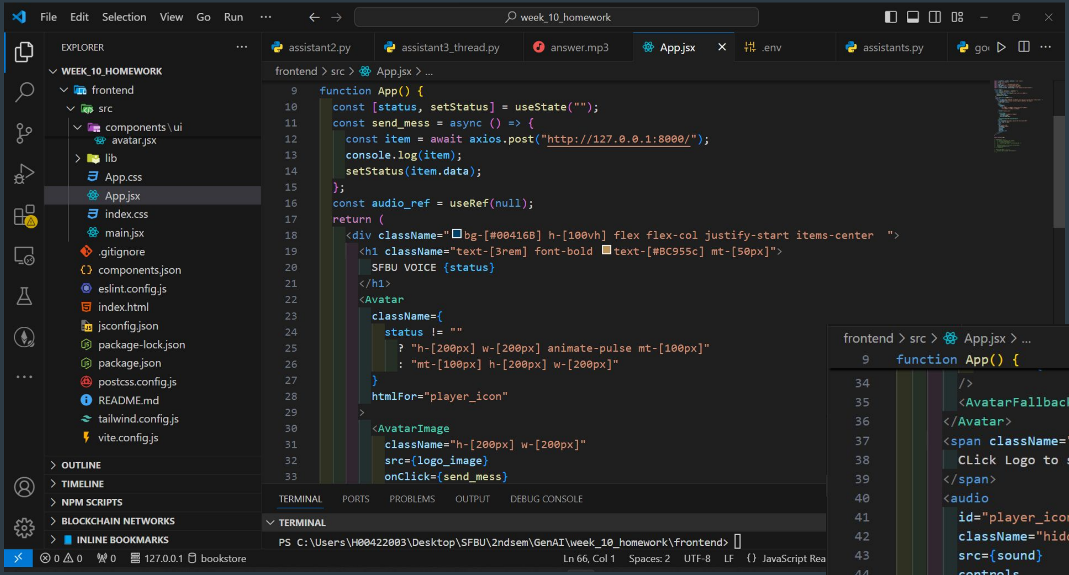
Installation:

  - Frontend: npm install
  - Backend: pip install -r requirements.txt

Frontend

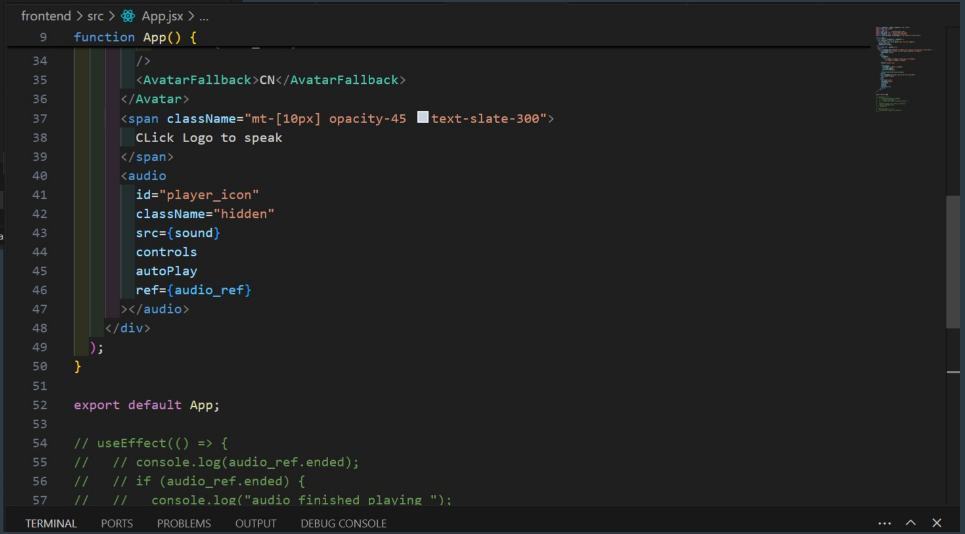
- Speech Recording Button: Allows users to start and stop recording speech.
- Technologies Used:
  - React Components: Modular design for easy maintenance.
  - Axios: For communicating with the backend API.
  - Web Audio API: To capture and send user audio input.

Front end code sample



```jsx
function App() {
  const [status, setStatus] = useState("");
  const send_mess = async () => {
    const item = await axios.post("http://127.0.0.1:8000/");
    console.log(item);
    setStatus(item.data);
  };
  const audio_ref = useRef(null);
  return (
    <div className="bg-[#00416B] h-[100vh] flex flex-col justify-start items-center  ">
      <h1 className="text-[3rem] font-bold text-[#BC955c] mt-[50px]">
        SFBU VOICE {status}
      </h1>
      <Avatar
        className={
          status != ""
            ? "h-[200px] w-[200px] animate-pulse mt-[100px]"
            : "mt-[100px] h-[200px] w-[200px]"
        }
        htmlFor="player_icon"
      >
        <AvatarImage
          className="h-[200px] w-[200px]"
          src={logo_image}
          onClick={send_mess}
```

```
TERMINAL
PS C:\Users\H00422003\Desktop\SFBU\2ndsem\GenAI\week_10_homework\frontend>
```



frontend > src > App.jsx > ...
TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE

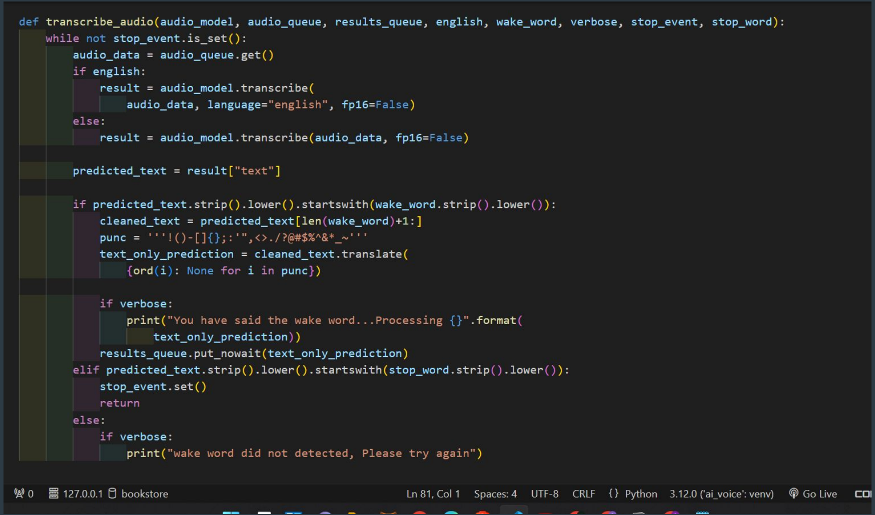
```jsx
function App() {
          />
          <AvatarFallback>CN</AvatarFallback>
        </Avatar>
        <span className="mt-[10px] opacity-45 text-slate-300">
          CLick Logo to speak
        </span>
        <audio
          id="player_icon"
          className="hidden"
          src={sound}
          controls
          autoPlay
          ref={audio_ref}
        ></audio>
      </div>
    );
}

export default App;

// useEffect(() => {
//   // console.log(audio_ref.ended);
//   // if (audio_ref.ended) {
//   //   console.log("audio finished playing ");
```

Backend

- Manages the flow of data between the frontend, OpenAI API, and the text-to-speech system.
- Key Responsibilities:
  - Handle audio files sent from the frontend.
  - Convert audio to text using the Whisper model.
  - Generate responses using OpenAI's GPT model.
  - Convert GPT responses to audio and send them back to the frontend.
- Tools and Libraries:
  - Flask for API endpoints.
  - Queue for handling audio processing tasks.
  - Multithreading for real-time processing.

Speech to Text

- Library: OpenAI’s Whisper model.
- Process:
  - Accepts audio from the frontend.
  - Converts it into text using Whisper's high-accuracy transcription.
- Code :

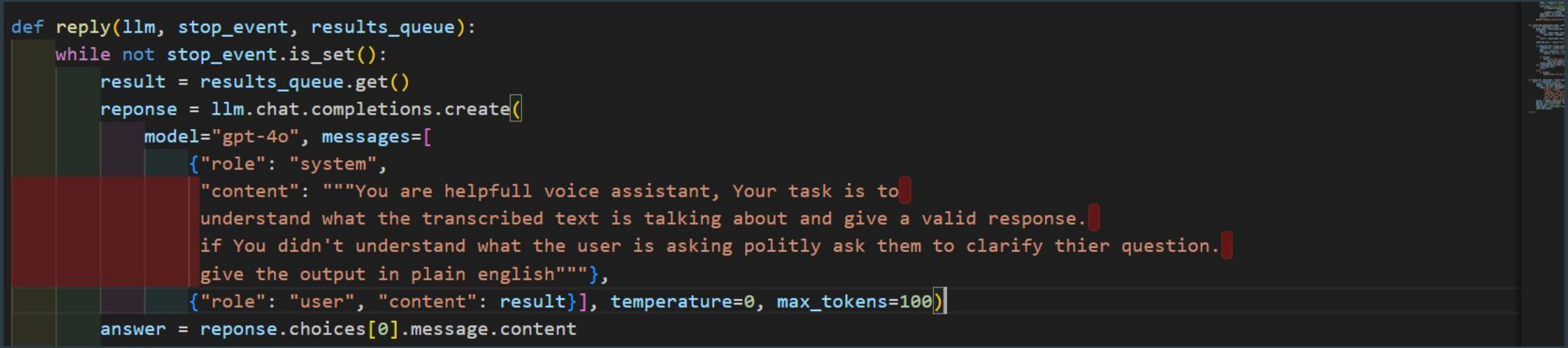


```python
def transcribe_audio(audio_model, audio_queue, results_queue, english, wake_word, verbose, stop_event, stop_word):
    while not stop_event.is_set():
        audio_data = audio_queue.get()
        if english:
            result = audio_model.transcribe(
                audio_data, language="english", fp16=False)
        else:
            result = audio_model.transcribe(audio_data, fp16=False)

        predicted_text = result["text"]

        if predicted_text.strip().lower().startswith(wake_word.strip().lower()):
            cleaned_text = predicted_text[len(wake_word)+1:]
            punc = '''!()-[]{};:'",<>./?@#$%^&*_~'''
            text_only_prediction = cleaned_text.translate(
                {ord(i): None for i in punc})

            if verbose:
                print("You have said the wake word...Processing {}".format(
                    text_only_prediction))
            results_queue.put_nowait(text_only_prediction)
        elif predicted_text.strip().lower().startswith(stop_word.strip().lower()):
            stop_event.set()
            return
        else:
            if verbose:
                print("wake word did not detected, Please try again")
```

```
0   127.0.0.1   bookstore          Ln 81, Col 1   Spaces: 4   UTF-8   CRLF   {} Python   3.12.0 ('ai_voice': venv)   Go Live
```

LLM Response

- Library: OpenAI GPT model via OpenAI API.
- Process:
    - The transcribed text from Whisper is sent as a query to the GPT model.
    - The model generates a conversational and contextually appropriate response.
- Code :

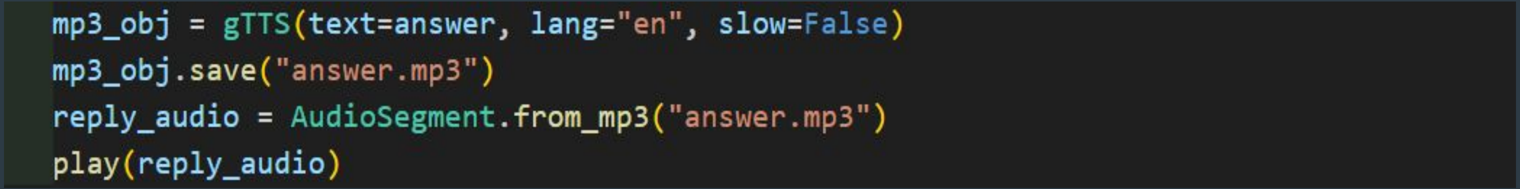


```python
def reply(llm, stop_event, results_queue):
    while not stop_event.is_set():
        result = results_queue.get()
        reponse = llm.chat.completions.create(
            model="gpt-4o", messages=[
                {"role": "system",
                 "content": """You are helpfull voice assistant, Your task is to
                 understand what the transcribed text is talking about and give a valid response.
                 if You didn't understand what the user is asking politly ask them to clarify thier question.
                 give the output in plain english"""},
                {"role": "user", "content": result}], temperature=0, max_tokens=100)
        answer = reponse.choices[0].message.content
```

Text to Speech

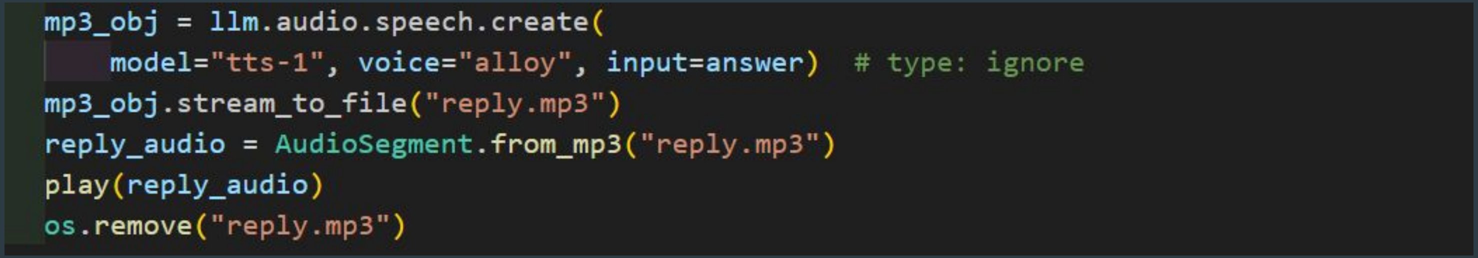
- Library: Google Text-to-Speech (gTTS) or Openai TTS.
- Process:
  - The response text from GPT is converted into speech using gTTS.
  - The audio is sent back to the frontend for playback.
- Code :

GTTS



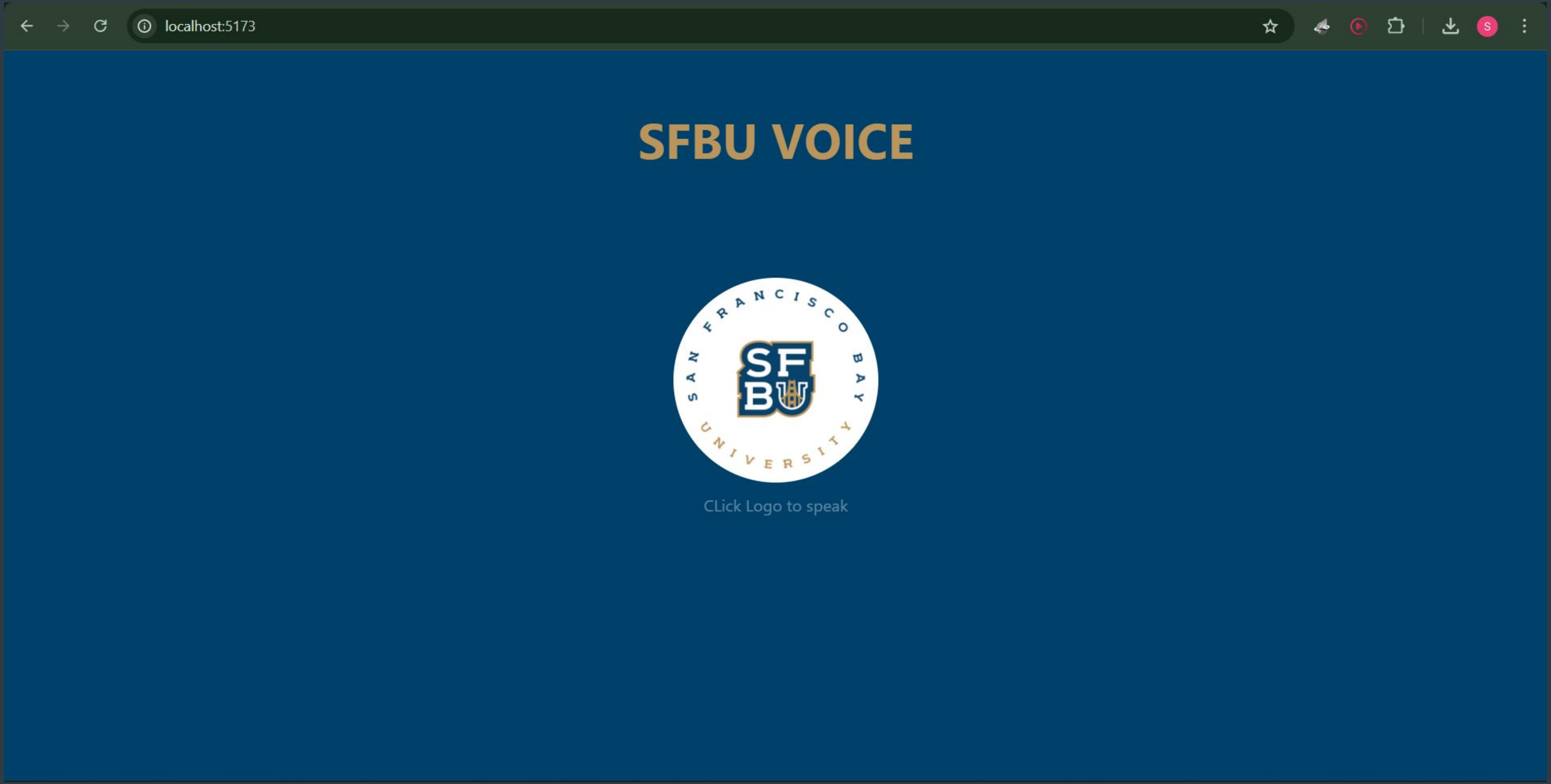
```python
mp3_obj = gTTS(text=answer, lang="en", slow=False)
mp3_obj.save("answer.mp3")
reply_audio = AudioSegment.from_mp3("answer.mp3")
play(reply_audio)
```

OTTS



```python
mp3_obj = llm.audio.speech.create(
    model="tts-1", voice="alloy", input=answer)  # type: ignore
mp3_obj.stream_to_file("reply.mp3")
reply_audio = AudioSegment.from_mp3("reply.mp3")
play(reply_audio)
os.remove("reply.mp3")
```

Slide 8: Demo


localhost:5173
SFBU VOICE
SAN FRANCISCO BAY UNIVERSITY
SFBU
CLick Logo to speak

Project Reference Materials

GitHub Link :

https://github.com/Montegan/SFBU_STT_TTS

Google Slides Link :

https://docs.google.com/presentation/d/17UKPttav2GZR-TxPyG7hNnxGE9t49ttK-uhu0n2j1Ms/edit?usp=sharing

“

Thank You

Simon Tesfatsion