PROJECT ALG

In [ ]:
```
!python -m ensurepip --upgrade # install pip on jupyter lab
```

In [ ]:
```
pip install gradio # install graio for UI
```

In [1]:
```
import openai
import os
import gradio as gr
from dotenv import load_dotenv, find_dotenv

# Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.getenv('OPENAI_API_KEY')

# Function to generate lyrics based on a description
def generate_lyrics(description):
    # Create a prompt for lyrics generation
    prompt = f"Write a song lyrics based on the following scene or description:\n{d

    # Create a chat completion
    chat_completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        max_tokens=300,  # Adjust as needed for response length
        temperature=0.7  # Adjust for creativity vs. accuracy
    )

    # Return the generated lyrics
    return chat_completion.choices[0].message['content']

# Gradio Interface
demo_lyrics_generator = gr.Interface(
    fn=generate_lyrics,
    inputs=gr.Textbox(label="Song Description", placeholder="Describe the scene or
    outputs="text",
    title="Lyrics Generator",
    description="Enter a description to generate lyrics based on the scene or song
)

# Launch the interface with a public URL
demo_lyrics_generator.launch(share=True)
```

```
/usr/local/lib/python3.9/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not f
ound. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/e
n/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Running on local URL:  http://127.0.0.1:7860
Running on public URL: https://8d15b95de679405c14.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, ru
n `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Loading...

Use via API · Built with Gradio

Out[1]:

In [2]:
```
import openai
import os
import gradio as gr
from dotenv import load_dotenv, find_dotenv

# Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.getenv('OPENAI_API_KEY')

# Function to generate lyrics based on description, genre, emotion, and language
def generate_lyrics(description, genre, emotion, language):
    # Create a more structured prompt for musical synchronization, rhyming, and mel
    prompt = (
        f"Write song lyrics in {language} based on the following scene or descripti
        f"Make sure the lyrics follow a {genre} style, evoke the emotion of {emotio
        f"include a melodic rhythm, and have a rhyming structure.\n\n"
        "Lyrics:"
    )

    # Create a chat completion
    chat_completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
```

```
        max_tokens=300,  # Adjust for response length
        temperature=0.8  # Higher creativity for lyrical flow
    )

    # Return the generated lyrics
    return chat_completion.choices[0].message['content']

# Gradio Interface with enhanced inputs
demo_lyrics_generator = gr.Interface(
    fn=generate_lyrics,
    inputs=[
        gr.Textbox(label="Song Description", placeholder="Describe the scene or the
        gr.Dropdown(choices=["Pop", "Rock", "Classical", "Hip-hop", "Jazz", "Countr
        gr.Textbox(label="Emotion", placeholder="Enter the emotion (e.g., love, sad
        gr.Dropdown(choices=["English", "Telugu", "Hindi", "Tamil", "Kannada", "Mal
    ],
    outputs="text",
    title="AI Lyrics Generator",
    description="Enter a description, genre, emotion, and language to generate musi
)

# Launch the interface with a public URL
demo_lyrics_generator.launch(share=True)
```

```
Running on local URL:  http://127.0.0.1:7861
Running on public URL: https://e2023eb4eafe8e9ad8.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, ru
n `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Loading...

Use via API · Built with Gradio

Out[2]:

In [4]:
```
import openai
import os
import gradio as gr
from dotenv import load_dotenv, find_dotenv

# Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.getenv('OPENAI_API_KEY')

# Define a fixed context for the application
fixed_context = "You are a lyrics generator."

# Function to generate lyrics based on a description
def generate_lyrics(description):
    prompt = f"Description: {description}\nGenerate song lyrics:"

    chat_completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        max_tokens=150,
        temperature=0.7
    )
```

```
    return chat_completion.choices[0].message['content']

# Gradio Interface
demo_lyrics_generator = gr.Interface(
    fn=generate_lyrics,
    inputs=gr.Textbox(label="Song Description", placeholder="Describe the scene or
    outputs="text",
    title="AI Lyrics Generator",
    description="Enter a description to generate song lyrics based on the scene.",
    theme="compact"  # Keep this if your version supports it
)

# Launch the interface with a public URL
demo_lyrics_generator.launch(share=True)
```

```
Running on local URL:  http://127.0.0.1:7862
Running on public URL: https://6d6c3874200787eaa0.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, ru
n `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Loading...

Use via API · Built with Gradio

Out[4]:

In [5]:

```
import openai
import os
import gradio as gr
```

```
from dotenv import load_dotenv, find_dotenv

# Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.getenv('OPENAI_API_KEY')

# Function to generate lyrics based on description, genre, emotion, and languages
def generate_lyrics(description, genre, emotion, languages, english_script):
    # Prepare the languages as a comma-separated string
    languages_str = ", ".join(languages)

    # Create a prompt for structured lyrics generation with multiple languages and
    prompt = (
        f"Write song lyrics in {languages_str} based on the following scene or desc
        f"Make sure the lyrics follow a {genre} style, evoke the emotion of {emotio
        f"include a melodic rhythm, and have a rhyming structure.\n"
    )

    # Add a note to generate lyrics in the English script if required
    if english_script:
        prompt += (
            f"Please provide the text in {languages_str} but written using the Engl
            f"maintain the original {languages_str} meaning while using English let
        )

    prompt += "Lyrics:"

    # Create a chat completion
    chat_completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        max_tokens=300,  # Adjust for response length
        temperature=0.8  # Higher creativity for lyrical flow
    )

    # Return the generated lyrics
    return chat_completion.choices[0].message['content']

# Gradio Interface with enhanced inputs
demo_lyrics_generator = gr.Interface(
    fn=generate_lyrics,
    inputs=[
        gr.Textbox(label="Song Description", placeholder="Describe the scene or the
        gr.Dropdown(choices=["Pop", "Rock", "Classical", "Hip-hop", "Jazz", "Countr
        gr.Textbox(label="Emotion", placeholder="Enter the emotion (e.g., love, sad
        gr.CheckboxGroup(choices=["English", "Telugu", "Hindi", "Tamil", "Kannada",
                         label="Languages", value=["English"]),
        gr.Checkbox(label="Provide in English Script", value=False)
    ],
    outputs="text",
    title="AI Lyrics Generator",
    description="Enter a description, genre, emotion, and select one or more langua
)

# Launch the interface with a public URL
demo_lyrics_generator.launch(share=True)
```

```
Running on local URL:  http://127.0.0.1:7863
Running on public URL: https://ab1d63dce3eca809cd.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, ru
n `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Loading...

Use via API · Built with Gradio

Out[5]:

In [7]:

```python
import openai
import os
import gradio as gr
from dotenv import load_dotenv, find_dotenv

# Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.getenv('OPENAI_API_KEY')

# Function to generate lyrics based on description, genre, emotion, and languages
def generate_lyrics(description, genre, emotion, languages, english_script):
    output_lyrics = ""

    for language in languages:
        # Create a prompt for each selected language
        prompt = (
            f"Write song lyrics in {language} based on the following scene or descr
            f"Make sure the lyrics follow a {genre} style, evoke the emotion of {em
```

```
            f"include a melodic rhythm, and have a rhyming structure.\n\n"
            "Lyrics:"
        )

        # Create a chat completion for each language
        chat_completion = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": prompt}],
            max_tokens=3000,  # Adjust for response length
            temperature=0.8  # Higher creativity for lyrical flow
        )

        # Get the generated lyrics
        lyrics = chat_completion.choices[0].message['content']

        # Append the language and its generated lyrics to the output
        output_lyrics += f"Language: {language}\n{lyrics}\n\n"

        # If the English script is requested and the language is not English
        if english_script and language != "English":
            english_script_prompt = (
                f"Provide the following lyrics in {language} but written using the
                "The text should maintain the original meaning while using English
                f"{lyrics}"
            )

            # Get the English transliteration
            english_script_completion = openai.ChatCompletion.create(
                model="gpt-3.5-turbo",
                messages=[{"role": "user", "content": english_script_prompt}],
                max_tokens=300,
                temperature=0.7
            )

            # Get the English script version and append it
            english_script_lyrics = english_script_completion.choices[0].message['c
            output_lyrics += f"English Script for {language}:\n{english_script_lyri

    return output_lyrics

# Gradio Interface with enhanced inputs
demo_lyrics_generator = gr.Interface(
    fn=generate_lyrics,
    inputs=[
        gr.Textbox(label="Song Description", placeholder="Describe the scene or the
        gr.Dropdown(choices=["Pop", "Rock", "Classical", "Hip-hop", "Jazz", "Countr
        gr.Textbox(label="Emotion", placeholder="Enter the emotion (e.g., love, sad
        gr.CheckboxGroup(choices=["English", "Telugu", "Hindi", "Tamil", "Kannada",
                         label="Languages", value=["English"]),
        gr.Checkbox(label="Provide in English Script", value=False)
    ],
    outputs="text",
    title="AI Lyrics Generator",
    description="Enter a description, genre, emotion, and select one or more langua
)
```

```
# Launch the interface with a public URL
demo_lyrics_generator.launch(share=True)
```

```
Running on local URL:  http://127.0.0.1:7865
Running on public URL: https://fc9a83d4b9e74cbe86.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, ru
n `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Loading...

Use via API · Built with Gradio

Out[7]: