Ex.No: **Date:**

Exploration of Prompting Techniques for AI Audio Generation

Aim:

The aim of this experiment is to explore different prompting techniques for AI-powered audio generation, enabling users to create custom voice outputs, music, or sound effects. The study focuses on leveraging text-to-speech (TTS) and audio generation models (e.g., OpenAI's TTS, ElevenLabs, Meta's AudioCraft) to understand how prompt variations affect output quality, style, and usability.

Procedure:

1. Define Audio Generation Goals

- Identify use cases (e.g., voiceovers, audiobooks, music composition, sound effects).
- Select appropriate AI models (e.g., OpenAI TTS, ElevenLabs, RVC, MusicGen).

2. Experiment with Prompting Techniques

- Test different prompt styles:
    - **Direct Commands:** "Generate a calm male voice saying: 'Welcome to the podcast.'"
    - **Emotional Tone:** "A cheerful female voice explaining science concepts."
    - **Style Transfer:** "Make this sound like a 1920s radio broadcast."
    - **Music/SFX:** "Generate a suspenseful background track for a thriller movie."

3. Develop the Audio Generation Script

- Use Python to interact with AI audio APIs.
- Implement voice parameter controls (speed, pitch, emotion).
- Compare outputs from different models.

4. Evaluate and Optimize

- Assess audio quality, clarity, and adherence to prompts.
- Adjust prompts iteratively for better results.

5. Deploy (Optional)

- Build a simple GUI (e.g., Gradio, Streamlit) for user-friendly interaction.

Program (Python Code)

Python code

```
import os
import openai
from elevenlabs import generate, play, set_api_key
import logging

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# API Keys (Replace with your keys)
OPENAI_API_KEY = "your_openai_api_key"
ELEVENLABS_API_KEY = "your_elevenlabs_api_key"

# Set API keys
openai.api_key = OPENAI_API_KEY
set_api_key(ELEVENLABS_API_KEY)

def generate_openai_tts(text, voice="alloy"):
    """Generate speech using OpenAI's TTS API."""
    try:
        response = openai.audio.speech.create(
            model="tts-1",
            voice=voice,
            input=text
        )
        response.stream_to_file("output_openai.mp3")
        return "OpenAI TTS generated successfully."
    except Exception as e:
```

```
        logger.error(f"OpenAI TTS Error: {e}")
        return "Failed to generate OpenAI TTS."

def generate_elevenlabs_tts(text, voice="Rachel"):
    """Generate speech using ElevenLabs API."""
    try:
        audio = generate(
            text=text,
            voice=voice,
            model="eleven_monolingual_v2"
        )
        play(audio)  # Play audio directly
        with open("output_elevenlabs.mp3", "wb") as f:
            f.write(audio)  # Save to file
        return "ElevenLabs TTS generated successfully."
    except Exception as e:
        logger.error(f"ElevenLabs Error: {e}")
        return "Failed to generate ElevenLabs TTS."

def main():
    print("=== AI Audio Generation Explorer ===")
    print("Enter a text prompt for audio generation (e.g., 'A robotic voice says hello').")
    print("Type 'quit' to exit.\n")

    while True:
        prompt = input("\nYour Audio Prompt: ").strip()

        if prompt.lower() in ["quit", "exit"]:
            print("Exiting...")
            break
```

```
        if not prompt:
            print("Please enter a valid prompt.")
            continue

        # Generate audio using both APIs
        print("\nGenerating audio with OpenAI TTS...")
        openai_result = generate_openai_tts(prompt, voice="nova")
        print(openai_result)

        print("\nGenerating audio with ElevenLabs...")
        elevenlabs_result = generate_elevenlabs_tts(prompt, voice="Bella")
        print(elevenlabs_result)

        print("\nAudio files saved as 'output_openai.mp3' and 'output_elevenlabs.mp3'.")

if __name__ == "__main__":
    main()
```

Output Examples:

1. Text-to-Speech (TTS) Prompt:

Input:

"A deep, authoritative voice says: 'The mission begins now.'"

Output:

- OpenAI TTS: A clear, synthetic voice with a serious tone.
- ElevenLabs: A more natural, cinematic voice with dramatic pacing.

2. Emotional Narration Prompt:

Input:

"A cheerful female voice explains how photosynthesis works."

Output:

- OpenAI TTS: Neutral tone with slight cheerfulness.
- ElevenLabs: Highly expressive, engaging delivery.

3. Music Generation (Using Meta's AudioCraft):

Input:

"Generate a 30-second jazz piano track with a relaxed vibe."

Output:

- A smooth jazz composition with piano melodies (if integrated with AudioCraft).

Result:

- Successfully generated speech and music using AI models.
- OpenAI TTS provides clear, customizable voices.
- ElevenLabs offers more natural, emotionally expressive outputs.
- Prompt engineering significantly impacts audio style and quality.