TEAM RED

Sahasra 23BCE1838
Anuska 23BCE1045
NagaSujatha 23BCE1927
Mani shankar 23BME1026

PROJECT STATEMENT:

2. Problem Statement (Elaborated)

In various industries, such as content creation, customer support, and entertainment, personalized voice experiences are becoming increasingly valuable. However, traditional voicerecording and synthesis require extensive training, expensive studio equipment, and professional voice actors.

Users who want to create personalized, high-quality voiceovers often face challenges such as:

- A lack of technical expertise in voice synthesis.
- Limited access to professional voice actors.
- Difficulty in maintaining consistency in tone and style across different scripts.
- The time-consuming nature of recording and editing voice content manually.

A solution is needed that enables users to create a digital replica of their voice using AI. This system should allow users to input text, which is then converted into speech that sounds just like their real voice, while maintaining tone, emotion, and natural inflections.

User Story

Title: Personalized Voice Cloning for Content Creation

As a content creator,
I want to generate realistic speech using a cloned version of my voice,
So that I can create consistent, high-quality voiceovers without needing to record manually every time.

Acceptance Criteria:

- Users should be able to provide voice samples for training the AI model.
- The system should analyze and learn the voice's unique characteristics, such as tone, pitch, and speaking style.
- Users should be able to input text, and the system will generate speech in their cloned voice.
- The generated speech should sound natural, with variations in tone and emotion.
- Users should have options to fine-tune pacing, emphasis, and expression.
- The output should be downloadable in multiple formats for integration into videos, podcasts, and virtual assistants.

CODE:

```
import torch
import torchaudio
import soundfile as sf
from TTS.api import TTS
import os

class VoiceCloner:
    def __init__(self, model="tts_models/multilingual/multi-dataset/xtts_v2"):
        """Initialize the TTS model with GPU support if available."""
        self.device = "cuda" if torch.cuda.is_available() else "cpu"
        self.tts = TTS(model).to(self.device)
        print(f"Model loaded on {self.device}")

    def train(self, voice_sample):
        """Train AI on a given voice sample to clone it."""
        if not os.path.exists(voice_sample):
            raise FileNotFoundError("Voice sample file not found!")

        print(f"Training on voice sample: {voice_sample}")
        return self.tts.tts_clone(voice_sample)

    def generate(self, text, cloned_voice, output_file="output.wav", speed=1.0,
emotion="neutral"):
        """Generate speech using the cloned voice with optional fine-tuning."""
        print(f"Generating speech with emotion: {emotion} and speed: {speed}")

        self.tts.tts_to_file(
            text=text,
            speaker_wav=cloned_voice,
            language="en",
            speed=speed,
            emotion=emotion,
            file_path=output_file
        )
```

```
        print(f"Generated speech saved as: {output_file}")
        return output_file

    def convert_format(self, input_file, output_file, format="mp3"):
        """Convert the generated speech to different audio formats."""
        if not os.path.exists(input_file):
            raise FileNotFoundError("Generated audio file not found!")

        waveform, sample_rate = torchaudio.load(input_file)
        sf.write(output_file, waveform.numpy().T, sample_rate)
        print(f"Converted {input_file} to {output_file}")

# ======= USAGE EXAMPLE =======
if __name__ == "__main__":
    vc = VoiceCloner()

    # Step 1: Train AI on the given voice sample
    trained_voice = vc.train("voice_sample.wav")

    # Step 2: Generate speech using the cloned voice
    generated_file = vc.generate(
        "Hello, this is my AI-generated voice! It sounds just like me.",
        trained_voice,
        "cloned_voice.wav",
        speed=1.1,
        emotion="happy"
    )

    # Step 3: Convert output to MP3 format
    vc.convert_format(generated_file, "cloned_voice.mp3", "mp3")
```