import React, { useState, useRef, useEffect } from 'react';

import { Mic, Square, Play, Pause, Download, Share2, Package } from 'lucide-react';

const VocalPainter = () => {

const [isRecording, setIsRecording] = useState(false);

const [audioBlob, setAudioBlob] = useState(null);

const [audioUrl, setAudioUrl] = useState(null);

const [isPlaying, setIsPlaying] = useState(false);

const [currentView, setCurrentView] = useState('home');

const [cymatics, setCymatics] = useState(null);

const [selectedColor, setSelectedColor] = useState('#1e3a8a'); // deep blue default

const mediaRecorderRef = useRef(null);

const audioRef = useRef(null);

const canvasRef = useRef(null);

// Emotion-color mappings

const emotionColors = {

sorrow: { color: '#1e3a8a', name: 'Deep Blue' },

joy: { color: '#eab308', name: 'Gold' },

silence: { color: '#6b7280', name: 'Silver' },

fragility: { color: '#fce7f3', name: 'Pale Pink' },

love: { color: '#d97706', name: 'Warm Amber' },

passion: { color: '#dc2626', name: 'Red' },

calm: { color: '#2563eb', name: 'Blue' },

hope: { color: '#eab308', name: 'Yellow' },

absence: { color: '#000000', name: 'Black' },

stillness: { color: '#ffffff', name: 'White' }

};

// Start recording

const startRecording = async () => {

try {

const stream = await navigator.mediaDevices.getUserMedia({ audio: true });

const mediaRecorder = new MediaRecorder(stream);

mediaRecorderRef.current = mediaRecorder;

const chunks = [];

mediaRecorder.ondataavailable = (e) => chunks.push(e.data);

mediaRecorder.onstop = () => {

const blob = new Blob(chunks, { type: 'audio/wav' });

setAudioBlob(blob);

setAudioUrl(URL.createObjectURL(blob));

generateCymatics();

};

mediaRecorder.start();

setIsRecording(true);

// Stop after 60 seconds max

setTimeout(() => {

if (mediaRecorderRef.current && isRecording) {

stopRecording();

}

}, 60000);

} catch (err) {

console.error('Error accessing microphone:', err);

}

};

// Stop recording

const stopRecording = () => {

if (mediaRecorderRef.current) {

mediaRecorderRef.current.stop();

mediaRecorderRef.current.stream.getTracks().forEach(track => track.stop());

}

setIsRecording(false);

};

// Play/pause audio

const togglePlayback = () => {

if (audioRef.current) {

if (isPlaying) {

audioRef.current.pause();

} else {

audioRef.current.play();

}

setIsPlaying(!isPlaying);

}

};

// Generate cymatic pattern (simplified visualization)

const generateCymatics = () => {

const canvas = canvasRef.current;

if (!canvas) return;

const ctx = canvas.getContext('2d');

const width = canvas.width = 400;

const height = canvas.height = 400;

ctx.clearRect(0, 0, width, height);

ctx.fillStyle = selectedColor;

// Generate organic, cymatic-inspired pattern

const centerX = width / 2;

const centerY = height / 2;

const rings = 8;

const points = 12;

for (let ring = 1; ring <= rings; ring++) {

const radius = (ring / rings) \* 150;

const amplitude = 20 / ring;

ctx.beginPath();

for (let i = 0; i <= points \* 2; i++) {

const angle = (i / points) \* Math.PI;

const variation = Math.sin(angle \* 3) \* amplitude + Math.cos(angle \* 5) \* amplitude \* 0.5;

const x = centerX + Math.cos(angle) \* (radius + variation);

const y = centerY + Math.sin(angle) \* (radius + variation);

if (i === 0) {

ctx.moveTo(x, y);

} else {

ctx.lineTo(x, y);

}

}

ctx.closePath();

ctx.globalAlpha = 0.1 + (0.1 \* ring);

ctx.fill();

}

// Add central resonance point

ctx.beginPath();

ctx.arc(centerX, centerY, 5, 0, 2 \* Math.PI);

ctx.globalAlpha = 1;

ctx.fill();

setCymatics(canvas.toDataURL());

};

// Generate QR code (simplified - would use actual QR library)

const generateQRCode = () => {

// In real implementation, this would generate actual QR code linking to audio

return "data:image/svg+xml;base64," + btoa(`

<svg width="100" height="100" xmlns="http://www.w3.org/2000/svg">

<rect width="100" height="100" fill="white"/>

<rect x="10" y="10" width="10" height="10" fill="black"/>

<rect x="30" y="10" width="10" height="10" fill="black"/>

<rect x="50" y="10" width="10" height="10" fill="black"/>

<rect x="70" y="10" width="10" height="10" fill="black"/>

<rect x="10" y="30" width="10" height="10" fill="black"/>

<rect x="50" y="30" width="10" height="10" fill="black"/>

<rect x="10" y="50" width="10" height="10" fill="black"/>

<rect x="30" y="50" width="10" height="10" fill="black"/>

<rect x="70" y="50" width="10" height="10" fill="black"/>

<rect x="10" y="70" width="10" height="10" fill="black"/>

<rect x="30" y="70" width="10" height="10" fill="black"/>

<rect x="50" y="70" width="10" height="10" fill="black"/>

<rect x="70" y="70" width="10" height="10" fill="black"/>

<text x="50" y="95" text-anchor="middle" font-size="8" fill="black">SCAN</text>

</svg>

`);

};

// Download the memory

const downloadMemory = () => {

if (cymatics) {

const link = document.createElement('a');

link.download = 'voice-memory.png';

link.href = cymatics;

link.click();

}

};

if (currentView === 'home') {

return (

<div className="min-h-screen bg-black text-white flex flex-col items-center justify-center p-6">

<div className="max-w-md w-full space-y-8 text-center">

<div className="space-y-4">

<h1 className="text-2xl font-light tracking-wide">VocalPainter</h1>

<p className="text-sm text-gray-400 leading-relaxed">

Turn voice into memory.<br/>

Not as data, but as emotion made visible.

</p>

</div>

<div className="mt-6 text-center max-w-md mx-auto">

<p className="text-sm text-gray-500 leading-relaxed">

Your voice has a shape — not in space, but in feeling.

We turn it into a visual pattern based on tone, rhythm, and emotion.

Colors reflect how you felt: deep blue for sorrow, gold for joy, silver for silence.

This isn't art.

It's a fingerprint of your voice — made visible.

</p>

</div>

<div className="space-y-6">

<div className="w-32 h-32 mx-auto border border-gray-800 rounded-full flex items-center justify-center">

{!isRecording ? (

<button

onClick={startRecording}

className="w-16 h-16 bg-white text-black rounded-full flex items-center justify-center hover:bg-gray-200 transition-colors"

>

<Mic size={24} />

</button>

) : (

<button

onClick={stopRecording}

className="w-16 h-16 bg-red-600 text-white rounded-full flex items-center justify-center animate-pulse"

>

<Square size={24} />

</button>

)}

</div>

{isRecording && (

<p className="text-sm text-red-400 animate-pulse">

Recording... (up to 60 seconds)

</p>

)}

{audioUrl && !isRecording && (

<div className="space-y-4">

<button

onClick={togglePlayback}

className="flex items-center justify-center space-x-2 mx-auto px-6 py-2 border border-gray-600 rounded hover:bg-gray-900 transition-colors"

>

{isPlaying ? <Pause size={16} /> : <Play size={16} />}

<span className="text-sm">Your Voice</span>

</button>

<audio

ref={audioRef}

src={audioUrl}

onEnded={() => setIsPlaying(false)}

hidden

/>

<button

onClick={() => setCurrentView('create')}

className="w-full py-3 bg-white text-black font-medium rounded hover:bg-gray-200 transition-colors"

>

Create Memory

</button>

</div>

)}

</div>

<div className="text-xs text-gray-600 space-y-2">

<p>No login. No account. No tracking.</p>

<p>This moment matters. You are not alone.</p>

</div>

<p className="text-sm text-gray-500 mt-4 leading-relaxed">

This is not a product.<br/>

It's a memory.<br/>

If you're grieving —<br/>

take your time.<br/>

You don't have to share.<br/>

You don't have to speak.<br/>

Just be here.

</p>

</div>

</div>

);

}

if (currentView === 'create') {

return (

<div className="min-h-screen bg-black text-white p-6">

<div className="max-w-2xl mx-auto space-y-8">

<div className="text-center space-y-2">

<h2 className="text-xl font-light">Choose Your Feeling</h2>

<p className="text-sm text-gray-400">Colors are vibrations — like sound.</p>

</div>

<div className="grid grid-cols-2 gap-4">

{Object.entries(emotionColors).map(([emotion, { color, name }]) => (

<button

key={emotion}

onClick={() => {

setSelectedColor(color);

generateCymatics();

}}

className={`p-4 border rounded-lg transition-colors ${

selectedColor === color

? 'border-white bg-gray-900'

: 'border-gray-700 hover:border-gray-500'

}`}

>

<div

className="w-8 h-8 rounded-full mx-auto mb-2"

style={{ backgroundColor: color }}

/>

<div className="text-sm font-medium capitalize">{emotion}</div>

<div className="text-xs text-gray-400">{name}</div>

</button>

))}

</div>

<div className="bg-gray-900 rounded-lg p-6 space-y-4">

<h3 className="text-lg font-light text-center">Your Memory</h3>

<div className="flex justify-center">

<canvas

ref={canvasRef}

className="border border-gray-700 rounded"

style={{ maxWidth: '100%', height: 'auto' }}

/>

</div>

{cymatics && (

<div className="flex justify-center">

<img

src={generateQRCode()}

alt="QR Code to hear voice"

className="w-20 h-20 bg-white rounded p-1"

/>

</div>

)}

</div>

{cymatics && (

<div className="flex space-x-4">

<button

onClick={downloadMemory}

className="flex-1 flex items-center justify-center space-x-2 py-3 border border-gray-600 rounded hover:bg-gray-900 transition-colors"

>

<Download size={16} />

<span>Save</span>

</button>

<button

onClick={() => setCurrentView('options')}

className="flex-1 flex items-center justify-center space-x-2 py-3 bg-white text-black rounded hover:bg-gray-200 transition-colors"

>

<Package size={16} />

<span>3D Print</span>

</button>

</div>

)}

<button

onClick={() => setCurrentView('home')}

className="w-full py-2 text-sm text-gray-400 hover:text-white transition-colors"

>

Start Over

</button>

</div>

</div>

);

}

if (currentView === 'options') {

return (

<div className="min-h-screen bg-black text-white p-6">

<div className="max-w-md mx-auto space-y-8">

<div className="text-center space-y-2">

<h2 className="text-xl font-light">Send Your Memory</h2>

<p className="text-sm text-gray-400">Physical voice. Real presence.</p>

</div>

<div className="bg-red-900 border border-red-700 rounded-lg p-4">

<p className="text-sm text-red-200">

⚠️ This is irreversible.<br/>

Only confirm if you're ready.

</p>

</div>

<div className="space-y-4">

<div className="border border-gray-700 rounded-lg p-4">

<h3 className="font-medium mb-2">Material: PLA Plastic</h3>

<p className="text-sm text-gray-400">Durable, lightweight, biodegradable</p>

<p className="text-lg font-light mt-2">$24 + shipping</p>

</div>

<div className="border border-gray-700 rounded-lg p-4">

<h3 className="font-medium mb-2">Material: Resin</h3>

<p className="text-sm text-gray-400">Smooth finish, fine details</p>

<p className="text-lg font-light mt-2">$32 + shipping</p>

</div>

</div>

<div className="space-y-3">

<input

type="email"

placeholder="Recipient's email"

className="w-full p-3 bg-gray-900 border border-gray-700 rounded text-white placeholder-gray-500"

/>

<textarea

placeholder="Address (optional message)"

rows="3"

className="w-full p-3 bg-gray-900 border border-gray-700 rounded text-white placeholder-gray-500"

/>

</div>

<button className="w-full py-3 bg-white text-black font-medium rounded hover:bg-gray-200 transition-colors">

Confirm & Send

</button>

<div className="text-center space-y-2">

<button

onClick={() => setCurrentView('create')}

className="text-sm text-gray-400 hover:text-white transition-colors"

>

Back to Memory

</button>

</div>

<div className="text-xs text-gray-600 text-center space-y-1">

<p>QR code will be on the base.</p>

<p>Scan to hear the original voice.</p>

<p>A conversation across time.</p>

</div>

</div>

</div>

);

}

};

export default VocalPainter;