```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Lovely Rhythmic Melody 🎵</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&amp;display=swap"
rel="stylesheet" />
    <link
href="https://fonts.googleapis.com/css2?family=Comfortaa:wght@400;700&amp;display=swap"
rel="stylesheet" />
    <script src="https://cdnjs.cloudflare.com/ajax/libs/tone/14.8.49/Tone.min.js"></script>
    <style>
        :root {
            --bg-hue: 240;
            --logo-height: 60px;
        }
        body {
            font-family: 'Inter', sans-serif;
            overscroll-behavior: none;
            background-color: hsl(var(--bg-hue), 15%, 10%);
            transition: background-color 0.5s ease; /* Faster background transition */
            color: hsl(var(--bg-hue), 15%, 85%);
        }
        .dynamic-text-color { color: hsl(var(--bg-hue), 100%, 90%); }
        .dynamic-accent-bg { background-color: hsl(var(--bg-hue), 50%, 30%, 0.3); }
        .dynamic-accent-border { border-color: hsl(var(--bg-hue), 50%, 50%); }

        .custom-scrollbar::-webkit-scrollbar { width: 8px; height: 8px; }
        .custom-scrollbar::-webkit-scrollbar-track { background: hsl(var(--bg-hue), 15%, 15%);
border-radius: 4px;}
        .custom-scrollbar::-webkit-scrollbar-thumb { background: hsl(var(--bg-hue), 20%, 35%);
border-radius: 4px; }
        .custom-scrollbar::-webkit-scrollbar-thumb:hover { background: hsl(var(--bg-hue), 25%,
45%); }

        canvas { display: block; max-width: 100%; border-radius: 0.5rem; cursor: crosshair; }

        .lyric-line {
            transition: color 0.5s ease-in-out, background-color 0.5s ease-in-out;
            padding: 6px 10px;
            border-radius: 6px;
```

```
        }
        .lyric-line.highlighted {
            color: hsl(var(--bg-hue), 90%, 85%);
            background-color: hsl(var(--bg-hue), 50%, 30%, 0.4);
        }

        .kanban-column { min-width: 260px; scroll-snap-align: start; }
        .kanban-node {
            border-left-width: 4px;
            transition: transform 0.2s ease-out, box-shadow 0.2s ease-out;
            background-color: hsl(var(--bg-hue), 20%, 20%);
        }
        .kanban-node:hover {
            transform: translateY(-3px) scale(1.02);
            box-shadow: 0 10px 15px -3px hsl(var(--bg-hue), 50%, 10%, 0.3), 0 4px 6px -2px
hsl(var(--bg-hue), 50%, 10%, 0.2);
        }
        .node-group-1 { border-color: hsl(var(--bg-hue), 70%, 60%); }
        .node-group-2 { border-color: hsl((var(--bg-hue) + 60), 70%, 60%); }
        .node-group-3 { border-color: hsl((var(--bg-hue) + 120), 70%, 60%); }

        #logoContainer {
            height: var(--logo-height);
            display: flex;
            align-items: center;
            justify-content: center;
            cursor: pointer;
            user-select: none;
            background-color: hsl(var(--bg-hue), 20%, 15%);
            border-bottom: 1px solid hsl(var(--bg-hue), 25%, 25%);
            overflow: hidden;
            position: sticky;
            top: 0;
            z-index: 20;
        }
        #logoText {
            font-family: 'Comfortaa', cursive;
            font-size: 1.75rem;
            font-weight: 700;
            display: flex;
        }
        #logoText span {
            display: inline-block;
            transition: transform 0.3s cubic-bezier(0.68, -0.55, 0.27, 1.55), color 0.5s ease;
```

```
            padding: 0 1px;
            animation: gentleWave 5s infinite ease-in-out;
        }
        #logoText span:nth-child(2n) { animation-delay: 0.2s; }
        #logoText span:nth-child(3n) { animation-delay: 0.4s; }
        #logoText span:nth-child(4n) { animation-delay: 0.1s; }
        #logoText span:nth-child(5n) { animation-delay: 0.3s; }

        @keyframes gentleWave { 0%, 100% { transform: translateY(0); } 50% { transform:
translateY(-3px); } }
        #logoText.touched span { animation: jumpAndColor 0.8s ease-in-out; }
        @keyframes jumpAndColor {
            0% { transform: translateY(0) scale(1); }
            25% { transform: translateY(-8px) scale(1.2); color: hsl(var(--bg-hue), 100%, 70%); }
            50% { transform: translateY(2px) scale(0.9); }
            75% { transform: translateY(-4px) scale(1.1); color: hsl((var(--bg-hue) + 40), 100%,
75%); }
            100% { transform: translateY(0) scale(1); color: inherit; }
        }

        .sticky-header {
            position: sticky; top: 0; z-index: 10;
            padding-top: 0.75rem; padding-bottom: 0.75rem;
            background-color: hsl(var(--bg-hue), 15%, 18%);
        }
        #lyricsWrapper .sticky-header, #mindmapWrapper .sticky-header,
#socialGuestbookWrapper .sticky-header {
             background-color: hsl(var(--bg-hue), 15%, 18%) !important;
        }

        .buy-button, .social-button, .submit-button {
            background-image: linear-gradient(to right, hsl(var(--bg-hue), 60%, 55%),
hsl((var(--bg-hue) + 20), 70%, 50%));
            transition: all 0.3s ease;
            box-shadow: 0 4px 6px rgba(0,0,0,0.1), 0 1px 3px rgba(0,0,0,0.08);
            color: white;
            padding: 0.5rem 1rem;
            border-radius: 0.375rem; /* rounded-md */
            font-weight: 600;
        }
        .buy-button:hover, .social-button:hover, .submit-button:hover {
            transform: translateY(-2px) scale(1.02);
            box-shadow: 0 10px 15px -3px hsl(var(--bg-hue), 50%, 30%, 0.3), 0 4px 6px -2px
hsl(var(--bg-hue), 50%, 30%, 0.2);
```

```
}

.like-button.liked svg {
    fill: hsl(var(--bg-hue), 90%, 65%);
    stroke: hsl(var(--bg-hue), 90%, 75%);
}
.like-button svg {
    transition: fill 0.2s ease, stroke 0.2s ease;
}

.form-input, .form-textarea {
    background-color: hsl(var(--bg-hue), 20%, 15%);
    border: 1px solid hsl(var(--bg-hue), 25%, 30%);
    color: hsl(var(--bg-hue), 15%, 80%);
    border-radius: 0.375rem;
    padding: 0.5rem 0.75rem;
    transition: border-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out;
}
.form-input:focus, .form-textarea:focus {
    outline: none;
    border-color: hsl(var(--bg-hue), 70%, 60%);
    box-shadow: 0 0 0 3px hsl(var(--bg-hue), 70%, 50%, 0.3);
}
.form-textarea {
    min-height: 80px;
}

#feedbackMessage {
    position: fixed;
    bottom: 20px;
    left: 50%;
    transform: translateX(-50%);
    padding: 10px 20px;
    border-radius: 8px;
    font-weight: 600;
    z-index: 1000;
    opacity: 0;
    transition: opacity 0.5s ease-in-out, transform 0.5s ease-in-out;
    pointer-events: none;
}
#feedbackMessage.show {
    opacity: 1;
    transform: translateX(-50%) translateY(0);
}
```

```
        #feedbackMessage.success {
            background-color: hsl(var(--bg-hue), 60%, 40%);
            color: hsl(var(--bg-hue), 60%, 90%);
        }
        #feedbackMessage.error {
             background-color: hsl(0, 60%, 40%);
            color: hsl(0, 60%, 90%);
        }
    </style>
</head>
<body class="text-gray-300 flex flex-col min-h-screen">
    <div id="logoContainer">
        <div id="logoText"></div>
    </div>

    <div id="mainContentWrapper" class="flex flex-col">

        <div id="lyricsWrapper" class="w-full p-6 lg:p-8 dynamic-bg-section">
            <h1 class="text-xl sm:text-2xl font-bold mb-4 dynamic-accent-border border-b
dynamic-text-color sticky-header">Song Lyrics 🎶</h1>

            <div class="flex flex-col sm:flex-row gap-4 mb-6">
                <img id="songCoverArt"
src="https://placehold.co/150x150/2c3e50/ecf0f1?text=Cover+Art" alt="Song Cover Art"
class="w-32 h-32 sm:w-36 sm:h-36 rounded-lg object-cover shadow-lg mx-auto sm:mx-0"/>
                <div class="flex-1 text-center sm:text-left">
                    <h2 id="songTitle" class="text-2xl font-bold dynamic-text-color mb-1">Lovely
Rhythmic Melody</h2>
                    <p id="artistName" class="text-lg mb-3" style="color: hsl(var(--bg-hue), 40%,
75%);">Your Artist Name</p>
                    <a href="#" id="buySongLink" target="_blank"
                       class="buy-button inline-flex items-center justify-center px-6 py-3 text-base
font-medium rounded-md hover:bg-opacity-90 focus:outline-none focus:ring-2 focus:ring-offset-2
dynamic-accent-border">
                        <svg class="w-5 h-5 mr-2" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M3 17a1 1 0 011-1h12a1 1 0
110 2H4a1 1 0 01-1-1zm3.293-7.707a1 1 0 011.414 0L9 10.586V3a1 1 0 112
0v7.586l1.293-1.293a1 1 0 111.414 1.414l-3 3a1 1 0 01-1.414 0l-3-3a1 1 0 010-1.414z"
clip-rule="evenodd"></path></svg>
                        Buy & Download
                    </a>
                    <p class="text-xs mt-2" style="color: hsl(var(--bg-hue), 30%, 60%);">Link to your
Shopify product page</p>
                </div>
```

```
        </div>

        <div class="space-y-3 text-sm sm:text-base leading-relaxed">
            <p class="lyric-line" id="lyric-0">"Distant by the lies in time you said to me, because you never held my broken back." 💔</p>
            <p class="lyric-line" id="lyric-1">"Time is wasted on the thought of little things, like the pain of a broken back." ⏳</p>
            <p class="lyric-line" id="lyric-2">"You will never let a memory free, drowning me took care of that." 🌊</p>
            <p class="lyric-line" id="lyric-3">"I didn’t like you bashing my brains brains brains brains brains brains brains brains." 🤯</p>
            <p class="lyric-line" id="lyric-4">"Distant eyes. I've been thinkin' 'bout: takin' life. I've been thinkin' 'bout wastin' time. I've been thinkin' 'bout a lot...." 🤔💭</p>
            <p class="lyric-line" id="lyric-5">"Sexualize. They touched my butt, my legs, my hips, and thighs. They said fuck his feelings 'cause he'll be alright, but, i really really won't..." 😔</p>
            <p class="lyric-line" id="lyric-6">"'Cause in my head, lives the dead, from what you said, and what they did. But that's how it goes... another story untold.... about the rich and the poor.... about the young and the old! About the love I never had inside, every fear that lived I wanted to die, every second of day and night, not allow to lay my head down and cry, but when I fear there's a little tone and they say: 'hello.....'" 📞</p>
        </div>
    </div>

    <div class="w-full h-[50vh] md:h-[60vh] flex items-center justify-center p-4 sm:p-6 dynamic-bg-section border-t border-b dynamic-accent-border">
        <canvas id="audiowaveCanvas"></canvas>
    </div>

    <div id="mindmapWrapper" class="w-full p-6 lg:p-8 dynamic-bg-section">
        <h2 class="text-xl sm:text-2xl font-bold mb-6 dynamic-accent-border border-b dynamic-text-color sticky-header">Thematic Mindmap 🧠</h2>
        <div class="flex flex-col lg:flex-row gap-6">
            <div class="kanban-column flex-1 p-4 rounded-lg shadow-lg dynamic-accent-bg">
                <h3 class="font-semibold mb-3 text-lg" style="color: hsl(var(--bg-hue), 70%, 70%);">Core Pain & Betrayal</h3>
                <div class="space-y-3">
                    <div class="kanban-node node-group-1 p-3 rounded shadow-md">"Broken back" (Physical & Emotional Neglect)</div>
                    <div class="kanban-node node-group-1 p-3 rounded shadow-md">"Bashing brains" (Mental Anguish)</div>
                    <div class="kanban-node node-group-1 p-3 rounded shadow-md">"Lies in time" (Deception)</div>
                </div>
            </div>
```

```
            <div class="kanban-column flex-1 p-4 rounded-lg shadow-lg dynamic-accent-bg">
                <h3 class="font-semibold mb-3 text-lg" style="color: hsl((var(--bg-hue) + 60), 70%,
70%);">Internal Turmoil</h3>
                <div class="space-y-3">
                    <div class="kanban-node node-group-2 p-3 rounded shadow-md">"Drowning
memory" (Trapped by the Past)</div>
                    <div class="kanban-node node-group-2 p-3 rounded shadow-md">"In my head,
lives the dead" (Internalized Trauma)</div>
                    <div class="kanban-node node-group-2 p-3 rounded shadow-md">"Love I never
had" (Emptiness)</div>
                    <div class="kanban-node node-group-2 p-3 rounded shadow-md">"Fear...
wanted to die" (Despair)</div>
                </div>
            </div>
            <div class="kanban-column flex-1 p-4 rounded-lg shadow-lg dynamic-accent-bg">
                <h3 class="font-semibold mb-3 text-lg" style="color: hsl((var(--bg-hue) + 120), 70%,
70%);">Reflections & Unseen Worlds</h3>
                <div class="space-y-3">
                    <div class="kanban-node node-group-3 p-3 rounded shadow-md">"Thinkin'
'bout: takin' life" (Suicidal Ideation)</div>
                    <div class="kanban-node node-group-3 p-3 rounded shadow-md">"Wastin' time"
(Regret/Futility)</div>
                    <div class="kanban-node node-group-3 p-3 rounded shadow-md">"Another story
untold" (Hidden Narratives)</div>
                    <div class="kanban-node node-group-3 p-3 rounded shadow-md">"Hello..." (A
Glimmer? A Question?)</div>
                </div>
            </div>
        </div>
    </div>

    <div id="socialGuestbookWrapper" class="w-full p-6 lg:p-8 dynamic-bg-section border-t
dynamic-accent-border">
        <h2 class="text-xl sm:text-2xl font-bold mb-6 dynamic-accent-border border-b
dynamic-text-color sticky-header">Connect & Share 💬</h2>

        <div class="grid grid-cols-1 md:grid-cols-2 gap-8">
            <div>
                <h3 class="text-lg font-semibold dynamic-text-color mb-3">Reactions &
Comments</h3>
                <div class="flex items-center space-x-4 mb-6">
                    <button id="likeButton" title="Like this page" class="like-button p-2 rounded-full
hover:bg-white/10 transition-colors">
```

```
                    <svg class="w-7 h-7" fill="none" stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0
00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"></path></svg>
                </button>
                <span id="likeCount" class="text-sm">0 Likes</span>

                <div class="border-l h-6 dynamic-accent-border mx-2"></div>

                <button id="copyLinkButton" title="Copy Link" class="social-button p-2
rounded-full text-white">
                    <svg class="w-6 h-6" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path d="M8 3a1 1 0 011-1h2a1 1 0 110 2H9a1 1 0
01-1-1z"></path><path d="M6 3a2 2 0 00-2 2v11a2 2 0 002 2h8a2 2 0 002-2V5a2 2 0 00-2-2 3
3 0 01-3 3H9a3 3 0 01-3-3z"></path></svg>
                </button>
                <a href="#" target="_blank" title="Share on Twitter (X)" class="social-button p-2
rounded-full text-white">
                    <svg class="w-6 h-6" viewBox="0 0 24 24" fill="currentColor"><path
d="M18.244 2.25h3.308l-7.227 8.26 8.502 11.24H16.17l-5.214-6.817L4.99
21.75H1.68l7.73-8.835L1.254 2.25H8.08l4.713 6.231zm-1.161 17.52h1.833L7.084
4.126H5.117z"></path></svg>
                </a>
                <a href="#" target="_blank" title="Share on Facebook" class="social-button p-2
rounded-full text-white">
                  <svg class="w-6 h-6" fill="currentColor" viewBox="0 0 24 24"><path d="M12
2.04c-5.5 0-9.96 4.46-9.96 9.96s4.46 9.96 9.96 9.96c5.5 0 9.96-4.46 9.96-9.96S17.5 2.04 12
2.04zm0 17.92c-4.41 0-7.96-3.55-7.96-7.96s3.55-7.96 7.96-7.96 7.96 3.55 7.96 7.96 3.55
7.96-7.96 7.96zm.23-5.99h-1.62v-2.41h1.62v-1.7c0-1.3.63-2.12
2.21-2.12h1.58v2.31h-1.09c-.48 0-.57.23-.57.58v.93h1.68l-.22
2.41h-1.46V18h-2.53v-3.99z"></path></svg>
                </a>
            </div>

            <form id="commentForm" class="space-y-3">
                <div>
                    <label for="commentText" class="block text-sm font-medium mb-1">Leave a
Comment:</label>
                    <textarea id="commentText" name="commentText" rows="3"
class="form-textarea w-full" placeholder="Your thoughts..."></textarea>
                </div>
                <button type="submit" class="submit-button px-4 py-2 text-sm font-semibold
rounded-md">Post Comment</button>
            </form>
```

```
            <div id="commentDisplay" class="mt-4 text-xs text-gray-400 italic">
                (Comments will appear here - backend functionality not yet implemented)
            </div>
        </div>

        <div>
            <h3 class="text-lg font-semibold dynamic-text-color mb-3">Guest Book</h3>
             <form id="guestbookForm" class="space-y-3">
                <div>
                    <label for="guestName" class="block text-sm font-medium mb-1">Your
Name:</label>
                    <input type="text" id="guestName" name="guestName" class="form-input
w-full" placeholder="Enter your name"/>
                </div>
                <button type="submit" class="submit-button px-4 py-2 text-sm font-semibold
rounded-md">Sign Guestbook</button>
            </form>
            <div id="guestbookDisplay" class="mt-4 text-xs text-gray-400 italic">
                (Guestbook entries will appear here - backend functionality not yet implemented)
            </div>
        </div>
    </div>
</div>

</div>

<div id="feedbackMessage"></div>


<script>
    const animationsEnabled = true;

    const canvas = document.getElementById('audiowaveCanvas');
    const ctx = canvas.getContext('2d');
    const canvasContainer = canvas.parentElement;
    const lyricLines = document.querySelectorAll('.lyric-line');
    const bodyEl = document.body;
    const dynamicBgSections = document.querySelectorAll('.dynamic-bg-section');
    const logoContainer = document.getElementById('logoContainer');
    const logoTextEl = document.getElementById('logoText');
    const stickyHeaders = document.querySelectorAll('.sticky-header');

    let animationFrameId;
    let time = 0;
```

```
        let currentHue = 240;
        let currentLyricIndex = 0;
        let lyricHighlightInterval;
        const HIGHLIGHT_DURATION = 4500;

        let interactionEffect = 0;
        const INTERACTION_DECAY_RATE = 0.05;
        const INTERACTION_STRENGTH = 1.2;

        let mouseX = null;
        let mouseY = null;
        let isMouseOverCanvas = false;
        const SPREAD_RADIUS = 100;
        const SPREAD_AMPLITUDE_DAMPEN = 0.6;
        const SPREAD_Y_FORCE = 20;

        let activePulses = [];
        const PULSE_MAX_RADIUS = 70;
        const PULSE_DURATION = 600;
        const PULSE_LINE_WIDTH = 2.5;

        let hyperParticles = [];
        const NUM_WAVES_FOR_PARTICLES = 5;
        const TOTAL_HYPER_PARTICLES = 4;
        const PARTICLE_TRAIL_LENGTH = 8;

        let stars = [];
        const NUM_STARS = 100;
        const STAR_TWINKLE_SPEED = 0.001;

        let nebulaCanvas = document.createElement('canvas');
        let nebulaCtx = nebulaCanvas.getContext('2d');
        let nebulaNoiseSeed = Math.random() * 1000;
        let nebulaTime = 0;
        const NEBULA_SCALE = 0.002;
        const NEBULA_SPEED = 0.00005;
        let nebulaFrameCount = 0;
        const NEBULA_UPDATE_INTERVAL = 5;

        let synth;
        let reverb;
        let audioInitialized = false;
        const musicalScale = ["C3", "Eb3", "F3", "G3", "Bb3", "C4", "Eb4", "F4", "G4", "Bb4", "C5",
"Eb5"];
```

```
let flungNotes = [];
const FLUNG_NOTE_SYMBOLS = ['♩', '♪', '♫', '♬'];
const FLUNG_NOTE_LIFESPAN = 100;
const FLUNG_NOTE_GRAVITY = 0.035;
const FLUNG_NOTE_INITIAL_VELOCITY_Y = -1.8;
const FLUNG_NOTE_SPREAD_X = 1.2;

const likeButton = document.getElementById('likeButton');
const likeCountDisplay = document.getElementById('likeCount');
let currentLikes = 0;
let isLiked = false;

const copyLinkButton = document.getElementById('copyLinkButton');
const commentForm = document.getElementById('commentForm');
const commentText = document.getElementById('commentText');
const guestbookForm = document.getElementById('guestbookForm');
const guestNameInput = document.getElementById('guestName');
const feedbackMessageDiv = document.getElementById('feedbackMessage');

function showFeedbackMessage(message, type = 'success') {
    feedbackMessageDiv.textContent = message;
    feedbackMessageDiv.className = '';
    feedbackMessageDiv.classList.add('show', type);
    setTimeout(() => {
        feedbackMessageDiv.classList.remove('show');
    }, 3000);
}

if (likeButton) {
    likeButton.addEventListener('click', () => {
        isLiked = !isLiked;
        if (isLiked) {
            currentLikes++;
            likeButton.classList.add('liked');
            showFeedbackMessage('You liked this!', 'success');
        } else {
            currentLikes--;
            likeButton.classList.remove('liked');
        }
        likeCountDisplay.textContent = `${currentLikes} Likes`;
    });
}
```

```
    if (copyLinkButton) {
        copyLinkButton.addEventListener('click', () => {
            // For copying to clipboard, using document.execCommand as navigator.clipboard
might be restricted in iframes
            const dummy = document.createElement('textarea');
            document.body.appendChild(dummy);
            dummy.value = window.location.href;
            dummy.select();
            try {
                document.execCommand('copy');
                showFeedbackMessage('Link copied to clipboard!', 'success');
            } catch (err) {
                console.error('Failed to copy link: ', err);
                showFeedbackMessage('Failed to copy link. Please copy manually.', 'error');
            }
            document.body.removeChild(dummy);
        });
    }

    if (commentForm) {
        commentForm.addEventListener('submit', (e) => {
            e.preventDefault();
            const comment = commentText.value.trim();
            if (comment) {
                showFeedbackMessage('Comment submitted (not saved)!', 'success');
                commentText.value = '';
            } else {
                showFeedbackMessage('Comment cannot be empty.', 'error');
            }
        });
    }

    if (guestbookForm) {
        guestbookForm.addEventListener('submit', (e) => {
            e.preventDefault();
            const guestName = guestNameInput.value.trim();
            if (guestName) {
                showFeedbackMessage(`Thanks for signing, ${guestName}! (not saved)`,
'success');
                guestNameInput.value = '';
            } else {
                 showFeedbackMessage('Please enter your name.', 'error');
            }
        });
```

```
    }

    function setupAudio() {
        if (audioInitialized || typeof Tone === 'undefined') return;
        try {
            if (Tone.context.state !== 'running') {
                console.warn("Tone.js context not running. Audio setup deferred until user
interaction.");
                return;
            }
            reverb = new Tone.Reverb(0.6).toDestination();
            synth = new Tone.FMSynth({
                harmonicity: 1.2,
                modulationIndex: 8,
                volume: -12,
                envelope: { attack: 0.005, decay: 0.15, sustain: 0.02, release: 0.3, },
                modulationEnvelope: { attack: 0.005, decay: 0.08, sustain: 0.02, release: 0.15, }
            }).connect(reverb);
            audioInitialized = true;
            console.log("Audio components initialized successfully.");
        } catch (e) {
            console.error("Error setting up audio components:", e);
            audioInitialized = false;
        }
    }

    function createHyperParticle(type) {
        let particle = {
            type: type, isActive: false, x: 0, y: 0, waveIndex: 0, progress: 0,
            speed: (type === 'tesseract' ? 0.0022 : 0.0018),
            direction: 1,
            size: (type === 'tesseract' ? 3 : 4),
            glowSize: (type === 'tesseract' ? 10 : 12),
            respawnTimer: 0, respawnDelay: Math.random() * 100 + 100,
            color: '', glowColor: '',
            hueOffset: (type === 'tesseract' ? 150 : 180),
            history: []
        };
        return particle;
    }

    function initializeParticle(particle, canvasWidth) {
        particle.isActive = true;
        particle.waveIndex = Math.floor(Math.random() * NUM_WAVES_FOR_PARTICLES);
```

```
            particle.direction = (Math.random() < 0.5) ? 1 : -1;
            particle.progress = (particle.direction === 1) ? 0.01 : 0.99;
            particle.x = particle.progress * canvasWidth;
            particle.history = [];
            updateParticleColors(particle);
        }

        function updateParticleColors(particle) {
            const particleHue = (currentHue + particle.hueOffset) % 360;
            particle.color = `hsla(${particleHue}, 100%, 90%, 0.95)`;
            particle.glowColor = `hsla(${particleHue}, 100%, 80%, 0.3)`;
        }

        const logoString = "lovely rhythmic Melody";
        logoString.split('').forEach(char => {
            const span = document.createElement('span');
            span.textContent = char === ' ' ? '\u00A0' : char;
            logoTextEl.appendChild(span);
        });
        const logoSpans = logoTextEl.querySelectorAll('span');

        function resizeCanvas() {
            const computedStyle = getComputedStyle(canvasContainer);
            const paddingX = parseFloat(computedStyle.paddingLeft) +
parseFloat(computedStyle.paddingRight);
            const paddingY = parseFloat(computedStyle.paddingTop) +
parseFloat(computedStyle.paddingBottom);
            let newWidth = canvasContainer.clientWidth - paddingX;
            let newHeight = canvasContainer.clientHeight - paddingY;
            newWidth = Math.max(newWidth, 150);
            newHeight = Math.max(newHeight, 100);
            canvas.width = newWidth;
            canvas.height = newHeight;

            nebulaCanvas.width = newWidth;
            nebulaCanvas.height = newHeight;

            initializeStars(newWidth, newHeight);
        }

        function updateColors() {
            currentHue = (currentHue + 0.35) % 360;
            bodyEl.style.setProperty('--bg-hue', currentHue.toFixed(0));
```

```
        const sectionBaseBg = `hsl(${currentHue}, 18%, 12%)`;
        const panelBg = `hsl(${currentHue}, 15%, 18%)`;

        dynamicBgSections.forEach(section => {
            if (section.id === 'lyricsWrapper' || section.id === 'mindmapWrapper' || section.id ===
'socialGuestbookWrapper') {
                section.style.backgroundColor = panelBg;
            } else {
                section.style.backgroundColor = sectionBaseBg;
            }
        });

        stickyHeaders.forEach(header => {
            const parent = header.parentElement;
            if (parent && (parent.id === 'lyricsWrapper' || parent.id === 'mindmapWrapper' ||
parent.id === 'socialGuestbookWrapper')) {
                header.style.backgroundColor = panelBg;
            }
        });

        if (logoContainer) {
            logoContainer.style.backgroundColor = `hsl(${currentHue}, 20%, 15%)`;
            logoContainer.style.borderColor = `hsl(${currentHue}, 25%, 25%)`;
        }

        logoSpans.forEach((span, index) => {
            if (!logoTextEl.classList.contains('touched')) {
                span.style.color = `hsl(${(currentHue + index * 10) % 360}, 85%, 78%)`;
            }
        });

        lyricLines.forEach((line) => {
            if (line.classList.contains('highlighted')) {
                line.style.color = `hsl(${currentHue}, 90%, 85%)`;
                line.style.backgroundColor = `hsl(${currentHue}, 50%, 30%, 0.4)`;
            } else {
                line.style.color = '';
                line.style.backgroundColor = 'transparent';
            }
        });

        hyperParticles.forEach(particle => updateParticleColors(particle));
    }
```

```
      function calculateWaveY(targetX, waveLayerIndex, canvasWidth, canvasHeight,
currentTime) {
          const centerY = canvasHeight / 2;
          const baseAmplitude = Math.min(canvasHeight / 3.5, 70);
          const clickPulseAmplitudeEffect = baseAmplitude * 0.5 * interactionEffect;

          const frequency = 0.025;
          const segments = Math.max(50, Math.floor(canvasWidth / 10));
          const segmentWidth = canvasWidth / segments;

          let layerBaseAmplitude = (baseAmplitude + clickPulseAmplitudeEffect) * (1 -
waveLayerIndex * 0.18);
          const layerFrequency = frequency * (1 + waveLayerIndex * 0.20);
          const timeOffset = waveLayerIndex * 0.45;

          const segmentIndexFloat = targetX / segmentWidth;

          let yOffset = 0;
          let currentSegmentAmplitude = layerBaseAmplitude;

          if (isMouseOverCanvas && mouseX !== null && mouseY !== null) {
              const originalWavePointY = centerY + Math.sin(segmentIndexFloat * layerFrequency
+ currentTime + timeOffset) *
                                  (0.65 + 0.35 * Math.sin(currentTime * 0.25 + segmentIndexFloat *
0.018)) * layerBaseAmplitude;
              const dxToMouse = targetX - mouseX;
              const dyToMouse = originalWavePointY - mouseY;
              const distanceToMouse = Math.sqrt(dxToMouse * dxToMouse + dyToMouse *
dyToMouse);

              if (distanceToMouse < SPREAD_RADIUS) {
                  const influence = 1 - (distanceToMouse / SPREAD_RADIUS);
                  currentSegmentAmplitude *= (1 - influence * SPREAD_AMPLITUDE_DAMPEN);
                  const pushDirection = Math.sign(dyToMouse) || 1;
                  yOffset = pushDirection * influence * SPREAD_Y_FORCE * (1 - waveLayerIndex *
0.3);
              }
          }

          const sinValue = Math.sin(segmentIndexFloat * layerFrequency + currentTime +
timeOffset) *
                           (0.65 + 0.35 * Math.sin(currentTime * 0.25 + segmentIndexFloat * 0.018));
          return centerY + sinValue * currentSegmentAmplitude + yOffset;
      }
```

```
function initializeStars(canvasWidth, canvasHeight) {
    stars = [];
    for (let i = 0; i < NUM_STARS; i++) {
        stars.push({
            x: Math.random() * canvasWidth,
            y: Math.random() * canvasHeight,
            size: Math.random() * 1.5 + 0.5,
            opacity: Math.random() * 0.3 + 0.1,
            opacityTarget: Math.random() * 0.3 + 0.1
        });
    }
}

function drawStars(canvasWidth, canvasHeight) {
    stars.forEach(star => {
        if (Math.abs(star.opacity - star.opacityTarget) < 0.01) {
            star.opacityTarget = Math.random() * 0.3 + 0.1;
        }
        if (star.opacity < star.opacityTarget) {
            star.opacity += STAR_TWINKLE_SPEED;
        } else {
            star.opacity -= STAR_TWINKLE_SPEED;
        }
        star.opacity = Math.max(0.1, Math.min(0.4, star.opacity));

        let starX = star.x;
        let starY = star.y;
        let starColor = `hsla(${currentHue}, 20%, 90%, ${star.opacity})`;
        // Time Dilation Lensing for stars removed

        ctx.beginPath();
        ctx.arc(starX, starY, star.size, 0, Math.PI * 2);
        ctx.fillStyle = starColor;
        ctx.fill();
    });
}

function seededRandom(seed) {
    var x = Math.sin(seed++) * 10000;
    return x - Math.floor(x);
}

function drawNebula(canvasWidth, canvasHeight) {
```

```
        nebulaFrameCount++;
        if (nebulaFrameCount % NEBULA_UPDATE_INTERVAL === 0) {
            nebulaTime += NEBULA_SPEED * NEBULA_UPDATE_INTERVAL;
            const imageData = nebulaCtx.createImageData(canvasWidth, canvasHeight);
            const data = imageData.data;

            for (let x = 0; x < canvasWidth; x++) {
                for (let y = 0; y < canvasHeight; y++) {
                    const val1 = seededRandom( (x * NEBULA_SCALE) + (y * NEBULA_SCALE *
0.5) + nebulaTime + nebulaNoiseSeed);
                    const val2 = seededRandom( (x * NEBULA_SCALE * 0.8) - (y *
NEBULA_SCALE * 1.2) + nebulaTime * 0.5 + nebulaNoiseSeed + 100);
                    const combinedVal = (val1 + val2) * 0.5;

                    const index = (y * canvasWidth + x) * 4;
                    const hue1 = (currentHue + 30) % 360;
                    const hue2 = (currentHue - 30 + 360) % 360;

                    const r1 = Math.sin(hue1 * Math.PI / 180 + 0) * 127 + 128;
                    const g1 = Math.sin(hue1 * Math.PI / 180 + 2 * Math.PI / 3) * 127 + 128;
                    const b1 = Math.sin(hue1 * Math.PI / 180 + 4 * Math.PI / 3) * 127 + 128;

                    const r2 = Math.sin(hue2 * Math.PI / 180 + 0) * 127 + 128;
                    const g2 = Math.sin(hue2 * Math.PI / 180 + 2 * Math.PI / 3) * 127 + 128;
                    const b2 = Math.sin(hue2 * Math.PI / 180 + 4 * Math.PI / 3) * 127 + 128;

                    data[index]     = r1 * combinedVal + r2 * (1 - combinedVal);
                    data[index + 1] = g1 * combinedVal + g2 * (1 - combinedVal);
                    data[index + 2] = b1 * combinedVal + b2 * (1 - combinedVal);
                    data[index + 3] = combinedVal * 40 + 5;
                }
            }
            nebulaCtx.putImageData(imageData, 0, 0);
        }
        ctx.globalAlpha = 0.25;
        ctx.drawImage(nebulaCanvas, 0, 0);
        ctx.globalAlpha = 1.0;
    }


    function drawWave() {
        updateColors();
        ctx.clearRect(0, 0, canvas.width, canvas.height);
```

```
        const width = canvas.width;
        const height = canvas.height;
        const nowMs = performance.now();

        if (width === 0 || height === 0) {
            animationFrameId = requestAnimationFrame(drawWave);
            return;
        }

        drawNebula(width, height);
        drawStars(width, height);

        // Time Dilation Manager logic removed

        const centerY = height / 2;
        const baseAmplitude = Math.min(height / 3.5, 70);
        const clickPulseAmplitudeEffect = baseAmplitude * 0.5 * interactionEffect;

        const frequency = 0.028;
        const segments = Math.max(50, Math.floor(width / 10));
        const segmentWidth = width / segments;
        const originalLineWidth = Math.max(1, width / 400);

        const numWaves = NUM_WAVES_FOR_PARTICLES;
        for (let j = 0; j < numWaves; j++) {
            ctx.beginPath();
            const waveHue = (currentHue + j * 45 + Math.sin(time * 0.1 + j) * 20) % 360;
            let baseLightness = 55 + j * 6;
            const currentLightness = baseLightness + (20 * interactionEffect);
            let waveStrokeStyle = `hsl(${waveHue}, ${80 + j*4}%, ${Math.min(100,
currentLightness)}%)`;
            ctx.lineWidth = originalLineWidth;

            let layerBaseAmplitude = (baseAmplitude + clickPulseAmplitudeEffect) * (1 - j * 0.18);
            const layerFrequency = frequency * (1 + j * 0.20);
            const timeOffset = j * 0.45;

            for (let i = 0; i <= segments; i++) {
                let x = i * segmentWidth;
                let y = 0;
                let yOffset = 0;
                let currentSegmentAmplitude = layerBaseAmplitude;
                let segmentStrokeStyle = waveStrokeStyle;
```

```
            const originalSinValue = Math.sin(i * layerFrequency + time + timeOffset) *
                    (0.65 + 0.35 * Math.sin(time * 0.25 + i * 0.018));
            let originalY = centerY + originalSinValue * currentSegmentAmplitude;


            if (isMouseOverCanvas && mouseX !== null && mouseY !== null) {
                const dxToMouse = x - mouseX;
                const dyToMouse = originalY - mouseY;
                const distanceToMouse = Math.sqrt(dxToMouse * dxToMouse + dyToMouse *
dyToMouse);

                if (distanceToMouse < SPREAD_RADIUS) {
                    const influence = 1 - (distanceToMouse / SPREAD_RADIUS);
                    currentSegmentAmplitude *= (1 - influence *
SPREAD_AMPLITUDE_DAMPEN);
                    const pushDirection = Math.sign(dyToMouse) || 1;
                    yOffset = pushDirection * influence * SPREAD_Y_FORCE * (1 - j * 0.3);
                }
            }

            // Time Dilation distortion for waves removed
            ctx.strokeStyle = segmentStrokeStyle;

            const finalSinValue = Math.sin(i * layerFrequency + time + timeOffset) *
                    (0.65 + 0.35 * Math.sin(time * 0.25 + i * 0.018));
            y = centerY + finalSinValue * currentSegmentAmplitude + yOffset;

            if (i === 0) ctx.moveTo(x, y);
            else ctx.lineTo(x, y);
        }
        ctx.stroke();
    }

    // Time Dilation Symbol drawing removed

    hyperParticles.forEach(particle => {
        if (particle.isActive) {
            particle.history.push({ x: particle.x, y: particle.y });
            if (particle.history.length > PARTICLE_TRAIL_LENGTH) {
                particle.history.shift();
            }

            particle.progress += particle.speed * particle.direction;
            particle.x = particle.progress * width;
```

```
                if (particle.progress > 1.02 || particle.progress < -0.02) {
                    particle.isActive = false;
                    particle.respawnTimer = particle.respawnDelay;
                } else {
                    particle.y = calculateWaveY(particle.x, particle.waveIndex, width, height, time);

                    // Time Dilation distortion for particles removed

                    if (particle.history.length > 1) {
                        ctx.beginPath();
                        ctx.moveTo(particle.history[0].x, particle.history[0].y);
                        for (let k = 1; k < particle.history.length; k++) {
                            const trailPoint = particle.history[k];
                            const opacity = (k / particle.history.length) * 0.5;
                            const trailWidth = particle.size * (k / particle.history.length) * 0.7;

                            const prevPoint = particle.history[k-1];
                            const grad = ctx.createLinearGradient(prevPoint.x, prevPoint.y, trailPoint.x,
trailPoint.y);
                            grad.addColorStop(0, `hsla(${(currentHue + particle.hueOffset) % 360},
100%, 85%, ${opacity * 0.5})`);
                            grad.addColorStop(1, `hsla(${(currentHue + particle.hueOffset) % 360},
100%, 90%, ${opacity})`);

                            ctx.strokeStyle = grad;
                            ctx.lineWidth = Math.max(0.5, trailWidth);
                            ctx.lineTo(trailPoint.x, trailPoint.y);
                        }
                        ctx.stroke();
                    }

                    ctx.beginPath();
                    ctx.arc(particle.x, particle.y, particle.glowSize, 0, Math.PI * 2);
                    ctx.fillStyle = particle.glowColor;
                    ctx.fill();

                    ctx.beginPath();
                    ctx.arc(particle.x, particle.y, particle.size, 0, Math.PI * 2);
                    ctx.fillStyle = particle.color;
                    ctx.fill();
                }
            } else {
                particle.respawnTimer--;
```

```
        if (particle.respawnTimer <= 0) {
            initializeParticle(particle, width);
        }
    }
});

flungNotes = flungNotes.filter(note => {
    note.x += note.vx * 1.2;
    note.y += note.vy * 1.2;
    note.vy += FLUNG_NOTE_GRAVITY;
    note.life--;
    note.opacity = Math.max(0, note.life / FLUNG_NOTE_LIFESPAN);
    note.rotation += note.rotationSpeed;

    const shimmerX = (Math.random() - 0.5) * 1 * (1 - note.opacity);
    const shimmerY = (Math.random() - 0.5) * 1 * (1 - note.opacity);

    if (note.life > 0) {
        ctx.save();
        ctx.translate(note.x + shimmerX, note.y + shimmerY);
        ctx.rotate(note.rotation);
        ctx.font = `${note.size * (0.5 + note.opacity * 0.5)}px Comfortaa, cursive`;
        ctx.fillStyle = `hsla(${(currentHue + 60) % 360}, 100%, 80%, ${note.opacity * 0.8})`;
        ctx.textAlign = 'center';
        ctx.textBaseline = 'middle';
        ctx.fillText(note.symbol, 0, 0);
        ctx.restore();
        return true;
    }
    return false;
});


activePulses = activePulses.filter(pulse => {
    const elapsedTime = nowMs - pulse.startTime;
    if (elapsedTime >= pulse.duration) return false;

    const progress = elapsedTime / pulse.duration;
    pulse.currentRadius = progress * pulse.maxRadius;
    pulse.opacity = 1 - progress;

    ctx.beginPath();
    ctx.arc(pulse.x, pulse.y, pulse.currentRadius, 0, Math.PI * 2);
```

```
            ctx.strokeStyle = `hsla(${(currentHue + 90) % 360}, 100%, 85%, ${pulse.opacity *
0.8})`;
            ctx.lineWidth = PULSE_LINE_WIDTH * (1 - progress) + 0.5;
            ctx.stroke();
            return true;
        });


        if (animationsEnabled) {
            time += 0.055;
        }

        if (interactionEffect > 0) {
            interactionEffect -= INTERACTION_DECAY_RATE;
            if (interactionEffect < 0) interactionEffect = 0;
        }

        animationFrameId = requestAnimationFrame(drawWave);
    }

    function updateLyricHighlight() {
        lyricLines.forEach(line => {
            line.classList.remove('highlighted');
            line.style.backgroundColor = 'transparent';
            line.style.color = '';
        });

        const currentLine = lyricLines[currentLyricIndex];
        if (currentLine) {
            currentLine.classList.add('highlighted');
            currentLine.style.color = `hsl(${currentHue}, 90%, 85%)`;
            currentLine.style.backgroundColor = `hsl(${currentHue}, 50%, 30%, 0.4)`;
        }
        currentLyricIndex = (currentLyricIndex + 1) % lyricLines.length;
    }

    function startLyricHighlighting() {
        if (lyricLines.length === 0) return;
        stopLyricHighlighting();
        updateLyricHighlight();
        lyricHighlightInterval = setInterval(updateLyricHighlight, HIGHLIGHT_DURATION);
    }
    function stopLyricHighlighting() {
        clearInterval(lyricHighlightInterval);
```

```
    lyricHighlightInterval = null;
}

logoContainer.addEventListener('click', () => {
    if (logoTextEl.classList.contains('touched')) return;
    logoTextEl.classList.add('touched');
    logoSpans.forEach((span, index) => { span.style.animationDelay = `${index * 0.05}s`; });
    setTimeout(() => {
        logoTextEl.classList.remove('touched');
        logoSpans.forEach((span) => { span.style.animationDelay = ''; });
         updateColors();
    }, 800);
});

function handleCanvasInteraction(event) {
    interactionEffect = INTERACTION_STRENGTH;
    const pos = getMousePos(canvas, event);

    if (typeof Tone !== 'undefined') {
        if (Tone.context.state !== 'running') {
            Tone.start().then(() => {
                console.log("AudioContext started by user gesture.");
                if (!audioInitialized) setupAudio();
                if (audioInitialized) {
                  triggerNote(pos.y);
                  spawnFlungNotes(pos.x, pos.y);
                }
            }).catch(e => console.error("Error starting Tone.js context:", e));
        } else if (audioInitialized) {
            triggerNote(pos.y);
            spawnFlungNotes(pos.x, pos.y);
        } else {
            setupAudio();
            if (audioInitialized) {
                triggerNote(pos.y);
                spawnFlungNotes(pos.x, pos.y);
            }
        }
    }

    activePulses.push({
        x: pos.x, y: pos.y, startTime: performance.now(),
        maxRadius: PULSE_MAX_RADIUS, duration: PULSE_DURATION,
        currentRadius: 0, opacity: 1
```

```
        });
    }

    function triggerNote(yPosition) {
        if (!synth || !audioInitialized) {
            return;
        }
        const noteIndex = musicalScale.length - 1 - Math.floor((yPosition / canvas.height) *
musicalScale.length);
        const note = musicalScale[Math.max(0, Math.min(noteIndex, musicalScale.length - 1))];
        try {
            synth.triggerAttackRelease(note, "8n", Tone.now());
        } catch (e) {
            console.error("Error triggering note:", e);
        }
    }

    function spawnFlungNotes(x, y) {
        const numNotesToSpawn = Math.floor(Math.random() * 2) + 1;
        for (let i = 0; i < numNotesToSpawn; i++) {
            flungNotes.push({
                x: x, y: y,
                vx: (Math.random() - 0.5) * FLUNG_NOTE_SPREAD_X * 2,
                vy: FLUNG_NOTE_INITIAL_VELOCITY_Y + (Math.random() - 0.5) * 0.5,
                life: FLUNG_NOTE_LIFESPAN + Math.random() * 30 - 15,
                symbol: FLUNG_NOTE_SYMBOLS[Math.floor(Math.random() *
FLUNG_NOTE_SYMBOLS.length)],
                size: 12 + Math.random() * 8,
                opacity: 1,
                rotation: (Math.random() - 0.5) * Math.PI * 0.2,
                rotationSpeed: (Math.random() - 0.5) * 0.05
            });
        }
    }


    canvas.addEventListener('click', handleCanvasInteraction);
    canvas.addEventListener('touchstart', (event) => {
        event.preventDefault();
        handleCanvasInteraction(event.touches[0] ? event.touches[0] : event);
    }, { passive: false });

    function getMousePos(canvasEl, evt) {
        const rect = canvasEl.getBoundingClientRect();
```

```
        const clientX = evt.clientX !== undefined ? evt.clientX : (evt.touches &&
evt.touches.length > 0 ? evt.touches[0].clientX : 0);
        const clientY = evt.clientY !== undefined ? evt.clientY : (evt.touches &&
evt.touches.length > 0 ? evt.touches[0].clientY : 0);
        return {
            x: clientX - rect.left,
            y: clientY - rect.top
        };
    }

    canvas.addEventListener('mousemove', (event) => {
        const pos = getMousePos(canvas, event);
        mouseX = pos.x;
        mouseY = pos.y;
        isMouseOverCanvas = true;
    });

    canvas.addEventListener('touchmove', (event) => {
        event.preventDefault();
        if (event.touches && event.touches.length > 0) {
            const pos = getMousePos(canvas, event.touches[0]);
            mouseX = pos.x;
            mouseY = pos.y;
            isMouseOverCanvas = true;
        }
    }, { passive: false });

    canvas.addEventListener('mouseenter', () => { isMouseOverCanvas = true; });
    canvas.addEventListener('mouseleave', () => {
        isMouseOverCanvas = false;
        mouseX = null;
        mouseY = null;
    });
    canvas.addEventListener('touchend', () => {
        isMouseOverCanvas = false;
    });
     canvas.addEventListener('touchcancel', () => {
        isMouseOverCanvas = false;
    });


    let resizeTimeout;
    window.addEventListener('resize', () => {
        clearTimeout(resizeTimeout);
```

```
    resizeTimeout = setTimeout(() => { resizeCanvas(); }, 100);
});

function initializeHyperParticles() {
    hyperParticles = [];
    for (let i = 0; i < TOTAL_HYPER_PARTICLES; i++) {
        const type = i < Math.floor(TOTAL_HYPER_PARTICLES / 2) ? 'penteract' : 'tesseract';
        const particle = createHyperParticle(type);
        hyperParticles.push(particle);
        setTimeout(() => initializeParticle(particle, canvas.width), i * 500);
    }
}

// initializeTimeDilation function removed

function initializeApp() {
    resizeCanvas();
    initializeHyperParticles();
    // initializeTimeDilation removed

    if (typeof Tone !== 'undefined') {
         audioInitialized = false;
         console.log("Tone.js loaded. Audio setup will be completed on first user interaction.");
    } else {
        console.warn("Tone.js not loaded. Interactive audio will not be available.");
    }

    logoTextEl.classList.add('animated');
    drawWave();
    startLyricHighlighting();
}

if (document.readyState === 'loading') {
    window.addEventListener('DOMContentLoaded', initializeApp);
} else {
    initializeApp();
}

document.addEventListener('visibilitychange', () => {
    if (document.hidden) {
        if (animationFrameId) {
            cancelAnimationFrame(animationFrameId);
            animationFrameId = null;
        }
```

```
            stopLyricHighlighting();
        } else {
            if (!animationFrameId) {
                resizeCanvas();
                time = performance.now() / 200;
                hyperParticles.forEach(particle => {
                    if (!particle.isActive && particle.respawnTimer <= 0) {
                        initializeParticle(particle, canvas.width);
                    }
                });
                // Time Dilation re-activation logic removed
                drawWave();
                startLyricHighlighting();
            }
        }
    });

  </script>
</body>
</html>
```