Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no">
  <title>VibePick</title>
  <link rel="icon" type="image/png" href="favicon.png">
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300&family=Playfair+Display:ital@1&di
splay=swap" rel="stylesheet">
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="onboarding-overlay" class="hidden">
  <div class="onboarding-card">
    <h2>Welcome to VibePick</h2>
    <p>Enter your location to let the environment curate your music.</p>

    <input type="text" id="city-input" placeholder="City (e.g., Seattle, WA)">

    <select id="lang-input">
      <option value="English">English</option>
      <option value="Hindi">Hindi</option>
      <option value="Kannada">Kannada</option>
      <option value="Bengali">Bengali</option>
      <option value="Tamil">Tamil</option>
      <option value="Telugu">Telugu</option>
      <option value="Malayalam">Malayalam</option>
      <option value="Punjabi">Punjabi</option>
      <option value="Spanish">Spanish</option>
      <option value="French">French</option>
      <option value="Japanese">Japanese</option>
    </select>

    <button id="save-btn">Unlock</button>
  </div>
  </div>

    <div id="app-container">
  <div id="text-display">Swipe to unlock the vibe.</div>

  <div id="strings-container">
    <div id="sound-hole"></div>

    <div class="string" data-index="1"></div>
    <div class="string" data-index="2"></div>
    <div class="string" data-index="3"></div>
    <div class="string" data-index="4"></div>
    <div class="string" data-index="5"></div>
```

```
      <div class="string" data-index="6"></div>
      </div>
   </div>
   </div>

   <div id="player"></div>

<div id="youtube-player" style="position: absolute; width: 0; height: 0; pointer-events: none;
opacity: 0;"></div>

   <script src="https://www.youtube.com/iframe_api"></script>

   <script src="app.js"></script>
</body>
</html>
```

app.js

```
// --- 1. YouTube Player Setup ---
let ytPlayer;
let isPlayerReady = false;

// This is a special function that the YouTube API looks for automatically
function onYouTubeIframeAPIReady() {
   ytPlayer = new YT.Player('youtube-player', {
      height: '0',
      width: '0',
      playerVars: {
         'autoplay': 0,
         'controls': 0,
         'playsinline': 1 // Prevents it from going full-screen on mobile
      },
      events: {
         'onReady': () => {
            isPlayerReady = true;
            console.log("📺 Hidden YouTube Player is locked and loaded!");
         }
      }
   });
}

// --- Onboarding & Local Storage Logic ---
const onboardingOverlay = document.getElementById('onboarding-overlay');
const cityInput = document.getElementById('city-input');
const langInput = document.getElementById('lang-input');
const saveBtn = document.getElementById('save-btn');

document.addEventListener('DOMContentLoaded', () => {
   const savedCity = localStorage.getItem('vibe_city');
   const savedLang = localStorage.getItem('vibe_lang');

   if (!savedCity || !savedLang) {
```

```
        onboardingOverlay.classList.remove('hidden');
    }
});

saveBtn.addEventListener('click', () => {
    const city = cityInput.value.trim();
    const lang = langInput.value;

    if (city) {
        localStorage.setItem('vibe_city', city);
        localStorage.setItem('vibe_lang', lang);
        onboardingOverlay.classList.add('hidden');
    } else {
        cityInput.style.border = "1px solid red";
        cityInput.placeholder = "City is required to find your vibe...";
    }
});

// --- The Strumming & Magic Loop Logic ---
const strings = document.querySelectorAll('.string');
const textDisplay = document.getElementById('text-display');

// 1. Pre-load the audio file so there is zero delay when swiping
const strumAudio = new Audio('assets/chord.wav');
strumAudio.volume = 1.0; // Ensure it is at full volume

let hasStrummed = false;
let currentStrum = new Set();

function handleStrum(event) {
    if (hasStrummed) return;

    event.preventDefault();

    let clientX, clientY;
    if (event.type.includes('touch')) {
        clientX = event.touches[0].clientX;
        clientY = event.touches[0].clientY;
    } else {
        clientX = event.clientX;
        clientY = event.clientY;
    }

    strings.forEach(string => {
        const rect = string.getBoundingClientRect();
        const index = string.getAttribute('data-index');

        if (clientX >= rect.left - 20 && clientX <= rect.right + 20 &&
            clientY >= rect.top && clientY <= rect.bottom) {

            if (!currentStrum.has(index)) {
```

```
            currentStrum.add(index);

            string.classList.add('strummed');
            if (navigator.vibrate) navigator.vibrate(15);
            setTimeout(() => string.classList.remove('strummed'), 200);

            // 2. Play the audio the millisecond the first string is hit
            if (currentStrum.size === 1) {
               playAcousticChord();
            }

            if (currentStrum.size >= 5) {
               hasStrummed = true;
               startMagicLoop();
            }
         }
      }
   });
}

function playAcousticChord() {
   // Reset the audio to the beginning just in case it was already playing
   strumAudio.currentTime = 0;

   // Play the beautiful chord!
   strumAudio.play().catch(error => {
      console.log("Audio playback blocked by browser:", error);
   });

   console.log("Audio playing: Acoustic chord is ringing out!");
}

async function startMagicLoop() {
   const stringsContainer = document.getElementById('strings-container');

   // 1. Fade out the UI elements
   textDisplay.style.opacity = 0;
   stringsContainer.classList.add('fade-out');

   // Get the saved data from the onboarding step
   const city = localStorage.getItem('vibe_city');
   const language = localStorage.getItem('vibe_lang');

   try {
      // 2. Call your new Vercel Serverless Function
      const response = await fetch('/api/get-song', {
         method: 'POST',
         headers: { 'Content-Type': 'application/json' },
         body: JSON.stringify({ city, language })
      });
```

```
      const data = await response.json();

      // 3. Display the result and PLAY THE SONG!
      setTimeout(() => {
        textDisplay.classList.add('playing-state');

        if (data.success) {
          // Play the YouTube Video!
          if (isPlayerReady && ytPlayer) {
            ytPlayer.loadVideoById(data.youtube_id);
            ytPlayer.setVolume(100); // Make sure it's loud enough
          }

          // Update the screen with the Vibe Reason, Intended Song, and Actual Link
          textDisplay.innerHTML = `
            <div style="font-size: 1.5rem; margin-bottom: 15px; font-weight: bold;">
              "${data.vibe_reason}"
            </div>
            <div style="font-size: 1rem; opacity: 0.8; font-style: italic; margin-bottom: 15px;">
              Playing: ${data.youtube_title}
            </div>
          `;
        } else {
          textDisplay.innerText = "Error: " + data.error;
        }

        textDisplay.style.opacity = 1;
      }, 1000); // Wait for initial text to fade

    } catch (error) {
      console.error("Fetch error:", error);
      setTimeout(() => {
        textDisplay.classList.add('playing-state');
        textDisplay.innerText = "Connection lost to the vibe.";
        textDisplay.style.opacity = 1;
      }, 1000);
    }
}

function resetStrum() {
    if (!hasStrummed) {
        currentStrum.clear();
    }
}

// Event Listeners
const container = document.getElementById('strings-container');
container.addEventListener('touchmove', handleStrum, { passive: false });
container.addEventListener('mousemove', (e) => {
    if (e.buttons === 1) handleStrum(e);
```

```
});

document.addEventListener('touchend', resetStrum);
document.addEventListener('mouseup', resetStrum);
```

style.css

```
/* Reset and Base Theme */
body, html {
    margin: 0;
    padding: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(135deg, #0f172a, #020617);
    color: white;
    overflow: hidden; /* Crucial: Prevents the screen from scrolling when swiping */
    display: flex;
    justify-content: center;
    align-items: center;
}

#app-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    height: 75vh;
    justify-content: space-between;
    width: 100%;
}

/* Typography */
#text-display {
    font-family: 'Playfair Display', serif;
    font-size: 1.8rem;
    text-align: center;
    opacity: 0.8;
    padding: 0 20px;
    transition: opacity 1s ease;
}

/* The Guitar Strings */
#strings-container {
    position: relative; /* Required to center the sound hole inside it */
    display: flex;
    gap: 24px;
    height: 50vh;
    align-items: center;
    padding: 20px;
    width: 80%;
    justify-content: center;
    z-index: 1;
}
```

```
/*The Sound Hole */
#sound-hole {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%, -50%);
   width: 220px;
   height: 220px;
   border-radius: 50%;
   background-color: #000000; /* Pure black for depth */
   /* An inset shadow creates the illusion of a deep cutout, and a subtle outer glow frames it */
   box-shadow: inset 0px 10px 30px rgba(0, 0, 0, 0.9),
            0px 0px 15px rgba(255, 255, 255, 0.03);
   z-index: -1; /* Pushes it behind the strings */
}

/* The Base String */
.string {
   height: 100%;
   /* A very subtle metallic gradient */
   background: linear-gradient(to right, rgba(255,255,255,0.1), rgba(255,255,255,0.3),
rgba(255,255,255,0.1));
   border-radius: 2px;
   transition: background-color 0.1s ease, box-shadow 0.1s ease, transform 0.1s ease;
   box-shadow: 2px 0px 5px rgba(0,0,0,0.5); /* Cast a slight shadow over the sound hole */
}

/* Realistic String Thicknesses */
.string[data-index="1"] { width: 5.5px; } /* Low E */
.string[data-index="2"] { width: 4.5px; } /* A */
.string[data-index="3"] { width: 3.5px; } /* D */
.string[data-index="4"] { width: 2.5px; } /* G */
.string[data-index="5"] { width: 1.5px; } /* B */
.string[data-index="6"] { width: 1px; }   /* High E */

/* The Glow Effect */
.string.strummed {
   background: rgba(255, 255, 255, 0.9);
   box-shadow: 0 0 15px 4px rgba(255, 255, 255, 0.5);
   transform: scaleX(1.4);
}

/* --- First-Time Onboarding UI --- */
#onboarding-overlay {
   position: fixed;
   top: 0;
   left: 0;
   width: 100%;
   height: 100%;
   background: rgba(2, 6, 23, 0.95); /* Deep midnight background */
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 50; /* Ensures it sits on top of the guitar strings */
    transition: opacity 0.5s ease;
}

/* Will toggle this class in JavaScript */
#onboarding-overlay.hidden {
    opacity: 0;
    pointer-events: none;
}

.onboarding-card {
    background: rgba(255, 255, 255, 0.05);
    padding: 40px 30px;
    border-radius: 16px;
    backdrop-filter: blur(12px); /* The frosted glass effect */
    border: 1px solid rgba(255, 255, 255, 0.1);
    text-align: center;
    font-family: 'Inter', sans-serif;
    display: flex;
    flex-direction: column;
    gap: 15px;
    width: 80%;
    max-width: 320px;
}

.onboarding-card h2 {
    font-family: 'Playfair Display', serif;
    margin: 0;
    font-size: 1.8rem;
}

.onboarding-card p {
    font-size: 0.9rem;
    opacity: 0.7;
    margin-bottom: 10px;
}

.onboarding-card input, .onboarding-card select {
    padding: 14px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.2);
    background: rgba(0, 0, 0, 0.4);
    color: white;
    font-family: 'Inter', sans-serif;
    font-size: 1rem;
    outline: none;
}
```

```
.onboarding-card button {
    padding: 14px;
    border-radius: 8px;
    border: none;
    background: white;
    color: black;
    font-family: 'Inter', sans-serif;
    font-weight: 600;
    font-size: 1rem;
    cursor: pointer;
    margin-top: 10px;
    transition: transform 0.1s ease;
}

.onboarding-card button:active {
    transform: scale(0.95);
}

/* --- UI Transitions --- */
.fade-out {
    opacity: 0 !important;
    pointer-events: none; /* Prevents any further touches or swipes */
    transition: opacity 2s ease; /* A nice, slow, dramatic fade */
}

/* make the AI text bigger and more central when it's the only thing on screen */
#text-display.playing-state {
    font-size: 2.2rem;
    line-height: 1.4;
    text-shadow: 0 0 20px rgba(255, 255, 255, 0.3);
    transform: translateY(20vh); /* Moves it down to the center of the screen */
    transition: all 2s ease;
}
```

api/get-song.js

```
export default async function handler(req, res) {
    if (req.method !== 'POST') {
        return res.status(405).json({ error: 'Method not allowed' });
    }

    const { city, language } = req.body;
    if (!city) return res.status(400).json({ error: 'City is required' });

    const WEATHER_API_KEY = process.env.WEATHER_API_KEY;
    const GROQ_API_KEY = process.env.GROQ_API_KEY;
    const YOUTUBE_API_KEY = process.env.YOUTUBE_API_KEY;

    try {
        // --- 1. FETCH WEATHER ---
```

```
    const weatherUrl =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${WEATHER_API_KEY}&units
=metric`;
    const weatherRes = await fetch(weatherUrl);
    const weatherData = await weatherRes.json();

    if (weatherData.cod != 200) {
      return res.status(400).json({ error: weatherData.message || 'City not found' });
    }

    const weatherDesc = weatherData.weather[0].description;
    const temp = Math.round(weatherData.main.temp);

    const utcTime = new Date().getTime() + (new Date().getTimezoneOffset() * 60000);
    const cityTime = new Date(utcTime + (weatherData.timezone * 1000));
    const hour = cityTime.getHours();

    let timeOfDay = "night";
    if (hour >= 5 && hour < 12) timeOfDay = "morning";
    else if (hour >= 12 && hour < 17) timeOfDay = "afternoon";
    else if (hour >= 17 && hour < 20) timeOfDay = "evening";

    const month = cityTime.toLocaleString('default', { month: 'long' });

    const vibeString = `It is currently ${timeOfDay}, ${weatherDesc}, ${temp}°C, in ${month}.
Location: ${city}. The user's language preference for the song is ${language}.`;

    // --- 2. ASK GROQ (THE CURATOR) ---
    const prompt = `You are an elite, avant-garde cinematic music curator. Your job is to select ONE
breathtaking, mood-defining song based on this exact environment and user preference:
"${vibeString}".

    CRITICAL INSTRUCTIONS:
    1. AVOID CLICHÉS: Absolutely NO obvious, overplayed, or stereotypical top-40 hits. (e.g., do not
pick 'Here Comes the Sun' for mornings, or generic Lo-Fi beats).
    2. DIG DEEPER: Choose high-quality, atmospheric tracks. Think indie, alternative, shoegaze, neo-
soul, breathtaking acoustic, or deep cuts from respected artists. The song should make the user feel
like the main character in a beautifully shot film.
    3. RESPECT THE LANGUAGE: The user requested the language: ${language}. You MUST provide a
culturally authentic, highly-regarded, atmospheric song in that exact language. Do not just pick an
English song.
    4. THE VIBE REASON: Write exactly a 5-word poetic reason that captures the sensory feeling of
the weather, time, and song.

    Return ONLY a raw JSON object. Do not include markdown formatting, backticks, or any
conversational text.
    Format must be exactly: {"song_title": "Title", "artist": "Artist", "vibe_reason": "Your 5-word
poetic reason."}`;

    const groqUrl = "https://api.groq.com/openai/v1/chat/completions";
```

```
    const groqRes = await fetch(groqUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${GROQ_API_KEY}`
      },
      body: JSON.stringify({
        model: "openai/gpt-oss-120b", // (Or whichever exact model ID you are using!)
        messages: [{ role: "user", content: prompt }],
        response_format: { type: "json_object" },
        // TEMPERATURE is the secret sauce for variety.
        // 0.85 gives us highly creative but accurate results.
        temperature: 0.85
      })
    });

    const groqData = await groqRes.json();

    // Parse the AI's response
    const aiText = groqData.choices[0].message.content;
    const aiJson = JSON.parse(aiText);

    console.log("GROQ CHOSE:", aiJson);

    // --- 3. SEARCH YOUTUBE FOR THE VIDEO ID ---
    const searchQuery = `"${aiJson.song_title}" "${aiJson.artist}" official audio`;

    const ytUrl =
`https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=1&q=${encodeURICom
ponent(searchQuery)}&type=video&key=${YOUTUBE_API_KEY}`;

    const ytRes = await fetch(ytUrl);
    const ytData = await ytRes.json();

    if (!ytData.items || ytData.items.length === 0) {
        return res.status(404).json({ error: 'Song found by AI, but no YouTube video found.' });
    }

    const videoId = ytData.items[0].id.videoId;
    // Grab the actual title of the video YouTube decided to give us!
    const ytTitle = ytData.items[0].snippet.title;

    console.log("YOUTUBE VIDEO ID:", videoId);
    console.log("YOUTUBE ACTUAL TITLE:", ytTitle);

    // --- 4. SEND EVERYTHING BACK TO FRONTEND ---
    return res.status(200).json({
      success: true,
      song_title: aiJson.song_title,
      artist: aiJson.artist,
      vibe_reason: aiJson.vibe_reason,
```

```
            youtube_id: videoId,
            youtube_title: ytTitle  // <-- Send the real title to the browser
        });

    } catch (error) {
        console.error("Backend Error:", error);
        return res.status(500).json({ error: 'Internal Server Error' });
    }
}
```