HTML code :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>MELODY PIECES</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>MELODY PIECES</h1>

<input type="file" accept=".mp3" id="song-input">

<div class="time-inputs">

<label for="start">Start Time (seconds):</label>

<input type="number" id="start" step="0.1" value="0">

<label for="end">End Time (seconds):</label>

<input type="number" id="end" step="0.1">

</div>

<div class="button-container">

<button id="cut-btn">Cut</button>

<button id="cut-play-btn">Cut and Play</button>

</div>

<script src="script.js"></script>

</body>

</html>

CSS CODE:

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f2f2f2;

margin: 0;

padding: 0;

}

h1 {

font-size: 24px;

margin-top: 20px;

}

input[type="file"] {

display: block;

margin: 20px auto;

}

.time-inputs {

display: flex;

justify-content: center;

align-items: center;

margin-top: 20px;

}

label {

font-size: 16px;

margin-right: 10px;

}

input[type="number"] {

width: 80px;

padding: 5px;

font-size: 16px;

}

.button-container {

margin-top: 20px;

}

button {

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 4px;

margin-right: 10px;

}

button:hover {

background-color: #0056b3;

}

button:disabled {

background-color: #ccc;

cursor: not-allowed;

}

JS CODE:

document.addEventListener("DOMContentLoaded", () => {

const songInput = document.getElementById("song-input");

const cutButton = document.getElementById("cut-btn");

const cutPlayButton = document.getElementById("cut-play-btn");

let audioContext;

let audioBuffer;

songInput.addEventListener("change", () => {

const file = songInput.files[0];

if (file) {

const reader = new FileReader();

reader.onload = async (e) => {

const fileResult = e.target.result;

audioBuffer = await decodeAudioData(fileResult);

};

reader.readAsArrayBuffer(file);

}

});

cutButton.addEventListener("click", () => {

if (audioBuffer) {

const startTime = parseFloat(document.getElementById("start").value) || 0;

const endTime = parseFloat(document.getElementById("end").value) || audioBuffer.duration;

const cutBuffer = getCutBuffer(audioBuffer, startTime, endTime);

// Save or process the cut buffer as required

console.log("Song cut!");

}

});

cutPlayButton.addEventListener("click", () => {

if (audioBuffer) {

const startTime = parseFloat(document.getElementById("start").value) || 0;

const endTime = parseFloat(document.getElementById("end").value) || audioBuffer.duration;

const cutBuffer = getCutBuffer(audioBuffer, startTime, endTime);

const audioSource = audioContext.createBufferSource();

audioSource.buffer = cutBuffer;

audioSource.connect(audioContext.destination);

audioSource.start();

console.log("Song cut and playing!");

}

});

function decodeAudioData(data) {

return new Promise((resolve, reject) => {

audioContext = new (window.AudioContext || window.webkitAudioContext)();

audioContext.decodeAudioData(data, resolve, reject);

});

}

function getCutBuffer(buffer, startTime, endTime) {

const startSample = Math.floor(startTime \* buffer.sampleRate);

const endSample = Math.floor(endTime \* buffer.sampleRate);

const channels = buffer.numberOfChannels;

const duration = endSample - startSample;

const cutBuffer = audioContext.createBuffer(channels, duration, buffer.sampleRate);

for (let channel = 0; channel < channels; channel++) {

const sourceData = buffer.getChannelData(channel);

const cutData = cutBuffer.getChannelData(channel);

for (let i = 0; i < duration; i++) {

cutData[i] = sourceData[startSample + i];

}

}

return cutBuffer;

}

});