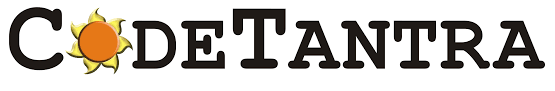
­



**INTERNSHIP REPORT ON**

**"PlyPluse (Digital Form of Music)"**

Submitted in partial fulfilment of requirements to CSE

**Submitted by:**

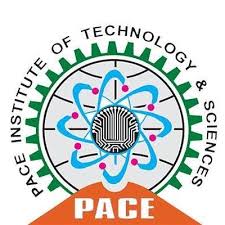
**G RAGHU BABU (22KQ1A0544),**

**K GOUTHAM (22KQ1A0552),**

**G SHASHANK (22KQ1A0543),**

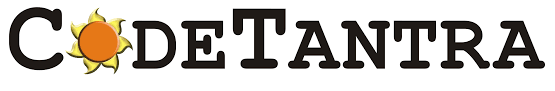
**SK SAMEER BASHA (22KQ1A0562).**

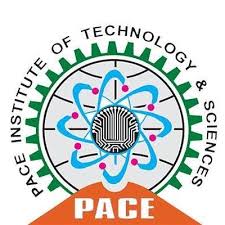
**III B. Tech CSE-A**

****

**PACE INSTITUTE OF TECHNOLOGY AND SCIENCES**

**JUNE 2025**

**PACE INSTITUTE OF TECHNOLOGY AND SCIENCE**

****DEPARTMENT OF CSE

**CERTIFICATE**

This is to certify that this internship report on **" PlyPluse (Digital Form of Music)"** is the Bonafide work of **"G RAGHU BABU (22KQ1A0544)"** who has carried out the work under my supervision and submitted in partial fulfilment for the award of **WEB TECHNOLOGY** Internship during the year 2025 (May) - 2025 (June).

**Signature of External Trainer Signature of Internal Trainer**

**Prof. & HOD**

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to all those who supported and guided me throughout my internship journey titled **"PlyPluse (Digital Form of Music)"**

First and foremost, I extend my heartfelt thanks to **Dr. G.V.S. Murthy**, Principal of **PACE Institute of Technology and Sciences**, for providing me with an encouraging environment and the opportunity to undertake this internship.

I am deeply grateful to **Dr.** **Giri Babu**, Head of the CSE, for paving the path and supporting me with all the necessary resources and guidance required to complete this internship.

My sincere thanks to **Mrs. P. Prisilla**, Internship In-charge, for his valuable suggestions, constant encouragement, and insightful feedback that played a significant role in the successful completion of this internship.

I would like to place on record my sincere appreciation and gratitude to Technical Trainer **P Sampath** from **CodeTantra**, the organization where I had the opportunity to pursue my internship. I am thankful for the exposure, technical learning, and real-time insights into web development, including **Web Development**, that I gained during this period.

Finally, I am immensely thankful to my friends and family for their unwavering moral support throughout this journey.

**G RAGHU BABU**

**(22KQ1A0544)**

**Abstract**

This project focuses on developing an **Interactive Music Visualization** website, leveraging fundamental web technologies: HTML for structure, CSS for styling, JavaScript for interactivity, and MongoDB for data persistence (if implemented for features like user preferences or music library management). The core of the project involves creating dynamic visual representations that respond to audio input, enhancing the user's music listening experience. This includes implementing various visualization types (e.g., bar graphs, waveforms, particle effects) that synchronize with audio frequencies and amplitudes.

The project aims to bridge academic learning with practical web development skills. By tackling a real-world inspired problem like interactive visualization, the internship strengthens foundational knowledge in front-end development, user experience design, and potentially back-end integration. Moreover, it fosters independent thinking, encourages clean code practices, and introduces basic software design strategies for web applications.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| S.NO | Name of the Topics | Page No |
| 1. | Introduction | 05 |
| 2. | Technologies Used | 06 |
| 3. | Problem Statement: Interactive Music Visualization | 07-12 |
| 4. | Learnings and Skills Gained | 13-16 |
| 5. | Conclusion | 17 |
| 6. | References | 17 |
| 7. | Prototype & Source Code | 18-47 |

**Introduction**

**1.1 Objectives**

* To apply HTML for structuring dynamic web content.
* To utilize CSS for creating visually appealing and responsive layouts.
* To implement JavaScript for client-side interactivity and audio processing.
* To understand and apply Web Audio API for analyzing audio data.
* To design and develop various real-time music visualization effects.
* To potentially integrate MongoDB for user-specific data storage (e.g., preferred visualizations, uploaded music).
* To develop problem-solving skills required for front-end and full-stack web development.
* To gain confidence in building interactive and media-rich web applications.
* To learn the process of designing, testing, and documenting web projects.

**1.2 Scope of the Project**

This project involves implementing an **Interactive Music Visualization** website based on a unique problem statement simulating real-world use cases. It includes:

* A front-end built with HTML, CSS, and JavaScript to create a user interface for music playback and visualization selection.
* Integration with the Web Audio API to capture and analyze audio data from a playing music source.
* Development of multiple visualization algorithms (e.g., frequency bars, waveform, circular visualizers) that dynamically react to audio properties.
* Responsive design to ensure optimal viewing and interaction across various devices (desktop, tablet, mobile).
* Potential back-end integration using Node.js and MongoDB for features like storing user playlists, visualization preferences, or user-uploaded music metadata.

Each solution aspect includes detailed input/output expectations, edge case considerations, code explanation, and test examples. This project strengthens practical web development skills and introduces structured thinking for building interactive web applications.

The project highlights different aspects of web programming:

1. Front-end structure and styling (HTML/CSS).
2. Client-side scripting, audio processing, and dynamic content generation (JavaScript/Web Audio API).
3. (Optional) Data persistence and back-end integration (MongoDB).

The scope also includes edge case handling and comprehensive code documentation.

**Technologies Used**

**2.1 Technologies Used**

* **Front-end Languages:**
  + HTML5 (for semantic structure)
  + CSS3 (for styling and responsiveness)
  + JavaScript (ES6+) (for interactivity, audio processing, and DOM manipulation)
* **JavaScript APIs/Libraries:**
  + Web Audio API (for audio input, analysis, and processing)
  + Canvas API (for drawing dynamic visualizations)
  + (Optional) D3.js or Three.js (for more complex visualizations, if applicable)
* **Back-end (Optional, for data persistence):**
  + Node.js (for server-side logic)
  + Express.js (for building RESTful APIs)
  + MongoDB (NoSQL database for data storage)
  + Mongoose (ODM for MongoDB in Node.js)
* **Development Tools:**
  + IDE: VS Code
  + Browser Developer Tools (for debugging)
  + Version Control: Git/GitHub

**5. Project Categorization**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Category** | **Key Concepts** |
| **HTML/CSS** | Front-end Development | Semantic HTML, CSS Flexbox/Grid, Responsive Design, CSS Animations/Transitions |
| **JavaScript** | Client-side Scripting / Audio Processing | DOM Manipulation, Event Handling, Asynchronous Programming (Promises/Async-Await), Web Audio API, Canvas API, Data Visualization Algorithms |
| **MongoDB** | Database Management / Back-end Development (Optional) | NoSQL Database, CRUD Operations, Data Modelling, Mongoose Schema, RESTful APIs |

This project maps directly to foundational topics in modern web development and interactive media.

* Interactive visualization resembles digital signal processing and real-time graphics.
* Front-end development reinforces user experience (UX) and user interface (UI) principles.
* Database integration (if included) models data management for web applications.

**Interactive Music Visualization**

**3. Project: Interactive Music Visualization**

**3.1.1 Problem Statement**

Develop a web application that provides interactive visual feedback to playing music. The application should allow users to upload or select music files and display dynamic visualizations that respond to the audio's frequency and amplitude changes in real-time. The solution should be responsive, aesthetically pleasing, and offer various visualization options.

**3.1.2 Input Format**

* User interaction via web interface (e.g., button clicks for play/pause, slider for volume).
* Music file input (e.g., audio element for pre-selected tracks, file input for user uploads).
* User selection of visualization type (e.g., dropdown, radio buttons).

**3.1.3 Output Format**

* Real-time graphical visualizations drawn on an HTML <canvas> element.
* Audio playback.
* User interface elements for control (play/pause button, volume slider, visualization selector).

**3.1.4 Constraints**

* Visualizations must be generated in real-time and synchronize accurately with the audio.
* The application should be responsive and function well on various screen sizes.
* Performance should be optimized to prevent lag or stuttering during visualization.
* Cross-browser compatibility is desired.
* (If MongoDB is used) Secure handling of user data and efficient database queries.

**3.1.5 Sample Interface/Features (Inspired by plypluse.rf.gd)**

* **Music Selection:** Ability to load different music tracks.
* **Playback Controls:** Play, pause, volume.
* **Visualization Modes:**
  + **Bar Graph:** Vertical bars reacting to frequency bands.
  + **Waveform:** Line drawing of the audio's amplitude over time.
  + **Circular/Radial:** Visuals expanding from a canter point.
  + **Particle Effects:** Abstract shapes or particles moving with the music.
* **Customization:** (Optional) Color schemes, intensity controls for visualizations.

**3.1.6 Algorithm Used (General)**

1. **Audio Source Loading:** Load an audio file (e.g., MP3, WAV) into an HTML <audio> element or via FileReader for user uploads.
2. **Web Audio API Context:** Create an AudioContext to process audio.
3. **Source Node:** Create an AudioBufferSourceNode (for loaded audio) or MediaElementSourceNode (for HTML audio element) and connect it to the AudioContext.
4. **Analyzer Node:** Create an AnalyserNode to retrieve frequency and time-domain data. Configure its FFT size and smoothing time constant.
5. **Data Retrieval:** Continuously fetch frequency data (e.g., getByteFrequencyData()) or waveform data (e.g., getByteTimeDomainData()) from the AnalyserNode in an animation loop (requestAnimationFrame).
6. **Canvas Drawing:** Use the HTML Canvas 2D API to clear the canvas and draw dynamic visual elements (lines, rectangles, circles, particles) based on the retrieved audio data.
7. **Looping:** The animation loop calls itself repeatedly, ensuring smooth and continuous visualization.

**3.1.7 Time & Space Complexity**

* **Time Complexity:** Predominantly O(N) per frame, where N is the FFT size/sample count from the AnalyserNode. Drawing operations on canvas also contribute. Optimized rendering is crucial for smooth performance.
* **Space Complexity:** O(N) for storing audio data arrays (frequency/time domain data) and potentially O(M) for visualization elements or particles on canvas, where M is the number of elements.

**3.1.8 Solution and Criteria**

A valid solution should:

* Successfully load and play music.
* Render real-time visualizations that accurately reflect audio characteristics.
* Provide a user-friendly interface for music control and visualization selection.
* Be responsive and performant across devices.
* (If MongoDB is used) Store and retrieve data reliably.

**3.1.9 Solution Design and Implementation**

* **HTML:** Minimal and semantic structure for audio element, canvas, and controls.
* **CSS:** Styling for layout, responsive design using media queries, and potentially CSS animations for UI elements.
* **JavaScript:**
  + **AudioManager module/class:** Handles audio loading, playback, Web Audio API setup, and analyzer node creation.
  + **VisualizationManager module/class:** Manages different visualization types. Each visualization type could be its own class/function.
  + **CanvasRenderer module/class:** Handles drawing operations on the canvas.
  + **Main App script:** Orchestrates interactions between AudioManager, VisualizationManager, and UI elements.
  + **Event Listeners:** For play/pause, volume change, visualization type selection.
  + **(Optional) Backend API:** Node.js/Express.js routes for handling music file uploads, user preferences, and MongoDB interactions.

**3.1.10 Error Handling and Edge Cases**

* **Audio Loading Errors:** Handle cases where audio files fail to load (e.g., network issues, unsupported format).
* **Web Audio API Support:** Check for browser compatibility of Web Audio API.
* **User Input Validation:** Ensure valid music file types are uploaded.
* **Empty Audio:** Gracefully handle scenarios where no music is playing.
* **Performance Degradation:** Implement safeguards or optimizations if performance drops (e.g., reduce visualization complexity).
* **(If MongoDB is used):** Database connection errors, invalid data writes/reads, schema validation.

**3.1.11 Limitations**

* Reliance on Web Audio API, which might have minor browser inconsistencies.
* Complexity of advanced visualizations can lead to performance issues on low-end devices.
* Copyright considerations for distributing music files for testing/demo.
* (If MongoDB is used) Requires server-side infrastructure.
* Limited to browser-supported audio formats.

**3.1.12 Code Optimization Techniques**

* **Efficient Canvas Drawing:** Minimize redraws, use requestAnimationFrame, avoid complex calculations in the drawing loop.
* **Web Audio API Optimization:** Reuse AudioContext, AnalyserNode.
* **CSS Optimizations:** Use hardware-accelerated CSS properties (e.g., transform, opacity).
* **JavaScript Performance:** Cache DOM elements, avoid unnecessary reflows/repaints, optimize loops.
* **(If MongoDB is used):** Indexing for faster queries, efficient data modeling to reduce document size.

**3.1.13 Methodology / Code (Conceptual)**

// Example JavaScript structure (simplified)

// 1. Audio Manager

class AudioManager {

constructor(audioElement) {

this.audioElement = audioElement;

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

this.sourceNode = this.audioContext.createMediaElementSource(this.audioElement);

this.analyser = this.audioContext.createAnalyser();

this.sourceNode.connect(this.analyser);

this.analyser.connect(this.audioContext.destination);

this.analyser.fftSize = 2048; // Adjust for desired frequency resolution

this.bufferLength = this.analyser.frequencyBinCount;

this.dataArray = new Uint8Array(this.bufferLength);

}

play() {

if (this.audioContext.state === 'suspended') {

this.audioContext.resume();

}

this.audioElement.play();

}

pause() {

this.audioElement.pause();

}

getFrequencyData() {

this.analyser.getByteFrequencyData(this.dataArray);

return this.dataArray;

}

getWaveformData() {

this.analyser.getByteTimeDomainData(this.dataArray);

return this.dataArray;

}

}

// 2. Visualization Renderer (example for bar graph)

class BarVisualization {

constructor(canvas, audioManager) {

this.canvas = canvas;

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

this.audioManager = audioManager;

this.WIDTH = canvas.width;

this.HEIGHT = canvas.height;

}

draw() {

requestAnimationFrame(() => this.draw()); // Loop for continuous animation

const dataArray = this.audioManager.getFrequencyData();

this.ctx.clearRect(0, 0, this.WIDTH, this.HEIGHT); // Clear canvas

const barWidth = (this.WIDTH / dataArray.length) \* 2.5; // Adjust for spacing

let x = 0;

for (let i = 0; i < dataArray.length; i++) {

const barHeight = dataArray[i]; // Value from 0-255

this.ctx.fillStyle = `rgb(${barHeight + 100}, 50, 50)`; // Example color

this.ctx.fillRect(x, this.HEIGHT - barHeight / 2, barWidth, barHeight / 2); // Draw bars

x += barWidth + 1; // Spacing

}

}

}

// 3. Main Application Logic

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

const audioElement = document.getElementById('audioPlayer');

const playPauseBtn = document.getElementById('playPauseBtn');

const canvas = document.getElementById('visualizerCanvas');

const audioManager = new AudioManager(audioElement);

const barViz = new BarVisualization(canvas, audioManager);

playPauseBtn.addEventListener('click', () => {

if (audioElement.paused) {

audioManager.play();

playPauseBtn.textContent = 'Pause';

barViz.draw(); // Start visualization

} else {

audioManager.pause();

playPauseBtn.textContent = 'Play';

}

});

// Load initial music

audioElement.src = 'path/to/your/music.mp3';

});

// HTML Structure (simplified)

/\*

<audio id="audioPlayer" controls></audio>

<button id="playPauseBtn">Play</button>

<canvas id="visualizerCanvas" width="800" height="400"></canvas>

\*/

**3.1.14 Final Output (Conceptual)**

The final output would be a fully functional web page that, upon loading, displays a music player interface. When a song is played, a dynamic visualization (e.g., frequency bars or waveform) appears on the canvas, reacting in real-time to the music.

**3.1.15 Explanation of the Problem**

The **Interactive Music Visualization** project involves translating abstract audio data into engaging visual experiences. It utilizes the power of the Web Audio API to tap into the raw sound information (frequencies, amplitudes) and the HTML Canvas API to render dynamic graphics. This project demonstrates how front-end technologies can be combined to create rich, interactive multimedia applications, reflecting real-world use cases in media players, live event visualizers, and educational tools for sound analysis. It emphasizes the importance of performance optimization and creative design in web development.

**3.1.16 Objective**

* To implement a fully functional interactive music visualization website.
* To gain a deep understanding of the Web Audio API and HTML Canvas for real-time graphics.
* To enhance problem-solving skills in integrating different web technologies.
* To develop a portfolio-worthy project demonstrating proficiency in front-end development.

**Learnings and Skills Gained**

**4. Real-World Scenarios and Analysis**

**4.1.1 Interactive Music Visualization**

In real-world applications, interactive music visualization is used in various contexts to enhance the user's audio experience, provide entertainment, or even for analytical purposes.

**Use Case Examples:**

* **Music Streaming Platforms:** Services like Spotify or Apple Music could integrate advanced visualizers to provide a more immersive listening experience, especially for premium users or during curated playlists.
* **Live Performance Visuals:** DJs and music producers use real-time visualizers during concerts and online streams to create dynamic backdrops that synchronize with their music, enhancing audience engagement.
* **Educational Tools:** Applications that teach audio engineering or music theory can use visualizers to help students understand concepts like frequency spectrum, amplitude, and waveforms.
* **Gaming:** Background music in games could be visually represented, adding to the atmosphere and player immersion.
* **Accessibility:** Visualizations can offer an alternative way for hearing-impaired individuals to "experience" music.

Preprocessing audio data for visualization is a fundamental task in multimedia applications and digital signal processing.

**4.2 Explanation of Project**

The **Interactive Music Visualization** project touches on several key areas of web development:

**HTML & CSS (Structure & Style)**

The project leverages HTML to define the core structure of the web page, including the audio player, control buttons, and the <canvas> element where visualizations are rendered. CSS is used extensively to style these elements, ensuring a responsive layout that adapts to different screen sizes (using Flexbox, Grid, and Media Queries) and providing an aesthetically pleasing user interface. This foundation is crucial for any modern web application, ensuring usability and visual appeal.

**Learning Highlights:**

* Demonstrates how semantic HTML creates a clear and accessible page structure.
* Reinforces responsive design principles for cross-device compatibility.
* Encourages creative CSS styling to enhance user experience.

**JavaScript & Web Audio API (Interactivity & Audio Processing)**

This is the core of the project. JavaScript is responsible for all client-side interactivity, including handling music playback (play/pause, volume), managing different visualization modes, and, most importantly, interacting with the Web Audio API. The Web Audio API allows direct access to audio data (frequencies, waveforms) in real-time. An AnalyserNode is used to capture this data, which is then passed to drawing functions that render the visualizations on the HTML <canvas> element using the Canvas 2D API. The use of requestAnimationFrame ensures smooth, browser-optimized animation loops.

**Learning Highlights:**

* Teaches real-time audio processing and analysis using the Web Audio API.
* Reinforces dynamic content generation and drawing on the HTML Canvas.
* Simulates principles of digital signal processing and interactive graphics.
* Encourages asynchronous programming and event-driven architecture.

**MongoDB (Optional - Data Persistence)**

While not strictly required for basic visualization, integrating MongoDB (via Node.js/Express.js backend) would elevate the project to a full-stack application. It would allow for features like:

* **User Profiles:** Storing user preferences for visualizations.
* **Music Library Management:** Enabling users to upload their own music files and storing metadata (artist, title, album).
* **Playlists:** Creating and saving custom playlists.

**Learning Highlights (if implemented):**

* Introduces server-side development with Node.js and Express.js.
* Teaches NoSQL database concepts and CRUD operations with MongoDB.
* Reinforces API design and full-stack application architecture.

**4.3 Learnings and Skills Gained**

* **1. Front-end Development Mastery:** Proficient in HTML5, CSS3, and JavaScript for building responsive and interactive web interfaces.
* **2. Real-time Audio Processing:** Acquired in-depth knowledge and practical experience with the Web Audio API for analyzing and visualizing audio data.
* **3. Dynamic Graphics Rendering:** Developed skills in using the HTML Canvas API to create complex, animated, and data-driven visualizations.
* **4. User Experience (UX) Design:** Learned to design intuitive interfaces and create engaging visual feedback for users.
* **5. Performance Optimization:** Understood techniques to ensure smooth and efficient real-time animations in web browsers.
* **6. (Optional) Full-Stack Integration:** Gained experience in connecting front-end applications with back-end services (Node.js/Express.js) and databases (MongoDB) for data persistence.
* **7. Problem Decomposition & Debugging:** Improved ability to break down complex problems into manageable components and systematically debug web applications.

**Additional Soft Skills Gained:**

* **Analytical Thinking:** Breaking down the process of audio visualization into distinct technical steps.
* **Creative Problem Solving:** Designing various visual effects that effectively represent audio properties.
* **Attention to Detail:** Ensuring synchronization, responsiveness, and aesthetic consistency across the application.
* **Technical Communication:** Clearly explaining the project's architecture, implementation, and future potential in documentation.

**4.4. Challenges Faced**

During the development of this project, we encountered a range of technical and logical challenges that contributed significantly to our learning experience.

* **Web Audio API Complexity:**
  + Understanding the audio graph, connecting nodes (source, analyser, destination), and correctly extracting frequency/time domain data proved challenging initially.
  + Synchronizing visualizations precisely with audio playback required careful timing and use of requestAnimationFrame.
* **Real-time Canvas Drawing:**
  + Optimizing canvas rendering for smooth performance across different devices, especially for complex visualizations, was a significant hurdle. Avoiding redraw flickers and ensuring efficient drawing operations required careful planning.
  + Implementing various visual patterns (e.g., bars, waveforms, particles) and mapping audio data to visual attributes required creative algorithmic thinking.
* **Cross-Browser Compatibility:**
  + Ensuring the Web Audio API and Canvas operations behaved consistently across different browsers sometimes led to unexpected issues.
* **Responsive Design:**
  + Making the visualizations and the UI elements adapt seamlessly to various screen sizes and orientations, especially for dynamic canvas elements, required meticulous CSS and JavaScript adjustments.
* **(If MongoDB was used):**
  + Setting up the Node.js server and connecting to MongoDB, understanding schema design, and implementing secure API endpoints.
  + Handling asynchronous operations and error handling in the back-end.

**General Challenges:**

* Debugging issues across the front-end (HTML, CSS, JavaScript) and potentially the back-end.
* Ensuring code readability, modularity, and maintainability for a growing codebase.
* Aligning the technical implementation with the desired user experience and visual appeal.

**4.5. Suggestions and Future Enhancements**

To further improve and expand the capabilities of the Interactive Music Visualization project, the following suggestions are proposed:

* **Enhanced Visualization Library:**
  + Integrate popular JavaScript visualization libraries like p5.js, D3.js, or Three.js for more sophisticated and diverse visual effects (e.g., 3D visualizations, interactive particle systems).
  + Add user-customizable parameters for visualizations (e.g., color gradients, intensity, shape variations, responsiveness to different frequency ranges).
* **User Account & Personalization (with MongoDB):**
  + Implement user authentication and profiles.
  + Allow users to save their preferred visualization settings, upload their own music, and create custom playlists stored in MongoDB.
* **Microphone Input:**
  + Extend the application to visualize audio input from the user's microphone, turning it into a real-time audio analysis tool.
* **External Music Source Integration:**
  + Explore integration with external music APIs (e.g., Spotify API) to stream music directly and visualize it (subject to API terms of service and authorization).
* **Export Options:**
  + Enable users to capture screenshots or short video clips of their favorite visualizations.
* **Performance Monitoring & Optimization:**
  + Integrate tools to monitor real-time performance and provide feedback or automatically adjust visualization complexity based on device capabilities.
* **Web Workers for Heavy Processing:**
  + Offload heavy audio analysis or complex visualization calculations to Web Workers to keep the main thread free and improve responsiveness.

**Conclusion**

**5. Conclusion**

This internship has provided hands-on exposure to building, debugging, and documenting a complete web solution for **Interactive Music Visualization**. The project reinforced fundamental web development skills in HTML, CSS, and JavaScript, while deeply exploring advanced browser APIs like the Web Audio API and HTML Canvas. It also encouraged systematic problem decomposition, emphasized the importance of code clarity, reusability, and maintainability, and introduced concepts of real-time data processing and interactive graphics. This experience has been invaluable in bridging theoretical knowledge with practical application in modern web development.

**6. References**

* **MDN Web Docs:** For comprehensive documentation on HTML, CSS, JavaScript, Web Audio API, and Canvas API.
  + <https://developer.mozilla.org/en-US/docs/Web/HTML>
  + <https://developer.mozilla.org/en-US/docs/Web/CSS>
  + <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
  + <https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API>
  + <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API>
* **(If MongoDB used) MongoDB Official Documentation:** For understanding database concepts and Mongoose.
  + <https://docs.mongodb.com/>
  + <https://mongoosejs.com/>
* **Stack Overflow & GitHub:** For troubleshooting specific implementation challenges and exploring open-source examples.

**Prototype & Source Code**

**Index.html**

<!-- Developed By G Raghu

K Goutham

G Shashank

Sk Sameer-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>PlyPluse</title>

<!-- Favicon -->

<link href="favicon\_io/1.png" rel="icon" />

<meta

content="Music Wave, Keytone Wave, Voice Wave, audio visualization, music analysis, interactive music, sound visualization, real-time audio processing"

name="keywords"

/>

<meta

content="PlyPluse offers interactive audio visualization experiences. Explore Music Wave for uploaded audio, Keytone Wave for keyboard input, and Voice Wave for real-time voice visualization."

name="description"

/>

<!-- Tailwind CSS CDN -->

<script src="https://cdn.tailwindcss.com"></script>

<style>

.footer {

margin-top: 3rem;

text-align: center;

color: #ffffff; /\* gray-400 \*/

font-size: 0.875rem;

}

/\* Custom styles for glassmorphism effect \*/

.glassmorphic {

background: rgba(

255,

255,

255,

0.1

); /\* Slightly transparent white background \*/

backdrop-filter: blur(10px); /\* Blur effect \*/

border: 1px solid rgba(255, 255, 255, 0.2); /\* Subtle white border \*/

box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37); /\* Shadow effect \*/

border-radius: 15px; /\* Rounded corners \*/

}

body {

font-family: "Inter", sans-serif; /\* Using Inter font \*/

/\* Light colorful gradient background for the entire page \*/

background: linear-gradient(

135deg,

#a8c0ff,

#3f2b96

); /\* Example gradient \*/

min-height: 100vh; /\* Ensure body takes full viewport height \*/

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

}

.text-shadow {

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3); /\* Add subtle text shadow for readability \*/

}

</style>

</head>

<body class="p-4 sm:p-6 md:p-8">

<!-- Navigation Bar -->

<nav

class="glassmorphic w-full max-w-7xl mx-auto p-4 sm:p-6 mb-8 flex flex-col sm:flex-row items-center justify-between space-y-4 sm:space-y-0 shadow-lg"

>

<!-- Logo -->

<div class="flex items-center">

<span

class="text-white text-3xl font-extrabold tracking-wide text-shadow"

>

<span class="text-indigo-200">P</span>LY

<span class="text-indigo-200">P</span>LUSE

</span>

</div>

<!-- Navigation Blocks -->

<div

class="flex flex-col sm:flex-row space-y-3 sm:space-y-0 sm:space-x-8 text-white text-lg font-medium"

>

<a

href="index.html"

class="hover:text-indigo-200 transition duration-300"

>Home</a

>

<a

href="Music\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Music</a

>

<a

href="Keytone\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Keytone</a

>

<a

href="Voice\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Voice</a

>

</div>

</nav>

<!-- Main Content - Horizontal Card Blocks -->

<main

class="w-full max-w-7xl mx-auto grid grid-cols-1 md:grid-cols-3 gap-6 sm:gap-8 justify-center items-stretch"

>

<!-- Card 1 -->

<div

class="glassmorphic p-6 sm:p-8 flex flex-col items-center text-center shadow-xl"

>

<h2 class="text-white text-2xl sm:text-3xl font-bold mb-4 text-shadow">

Music\_Wave

</h2>

<p class="text-white text-opacity-80 leading-relaxed text-lg mb-4">

Upload an audio file and watch the music come alive!

</p>

<a href="Music\_Wave.html"

><button

class="mt-auto px-6 py-3 bg-white bg-opacity-20 text-white font-semibold rounded-full hover:bg-opacity-30 transition duration-300 text-sm sm:text-base"

>

Click to Go

</button></a

>

</div>

<!-- Card 2 -->

<div

class="glassmorphic p-6 sm:p-8 flex flex-col items-center text-center shadow-xl"

>

<h2 class="text-white text-2xl sm:text-3xl font-bold mb-4 text-shadow">

Keytone\_Wave

</h2>

<p class="text-white text-opacity-80 leading-relaxed text-lg mb-4">

Click Keys and watch the music come alive!

</p>

<a href="Keytone\_Wave.html">

<button

class="mt-auto px-6 py-3 bg-white bg-opacity-20 text-white font-semibold rounded-full hover:bg-opacity-30 transition duration-300 text-sm sm:text-base"

>

Click to Go

</button></a

>

</div>

<!-- Card 3 -->

<div

class="glassmorphic p-6 sm:p-8 flex flex-col items-center text-center shadow-xl"

>

<h2 class="text-white text-2xl sm:text-3xl font-bold mb-4 text-shadow">

Voice\_Wave

</h2>

<p class="text-white text-opacity-80 leading-relaxed text-lg mb-4">

Speak and watch the music come alive!

</p>

<a href="Voice\_Wave.html"

><button

class="mt-auto px-6 py-3 bg-white bg-opacity-20 text-white font-semibold rounded-full hover:bg-opacity-30 transition duration-300 text-sm sm:text-base"

>

Click to Go

</button></a

>

</div>

</main>

<footer class="footer">

<p>&copy; 2025 PlyPluse. All rights reserved.</p>

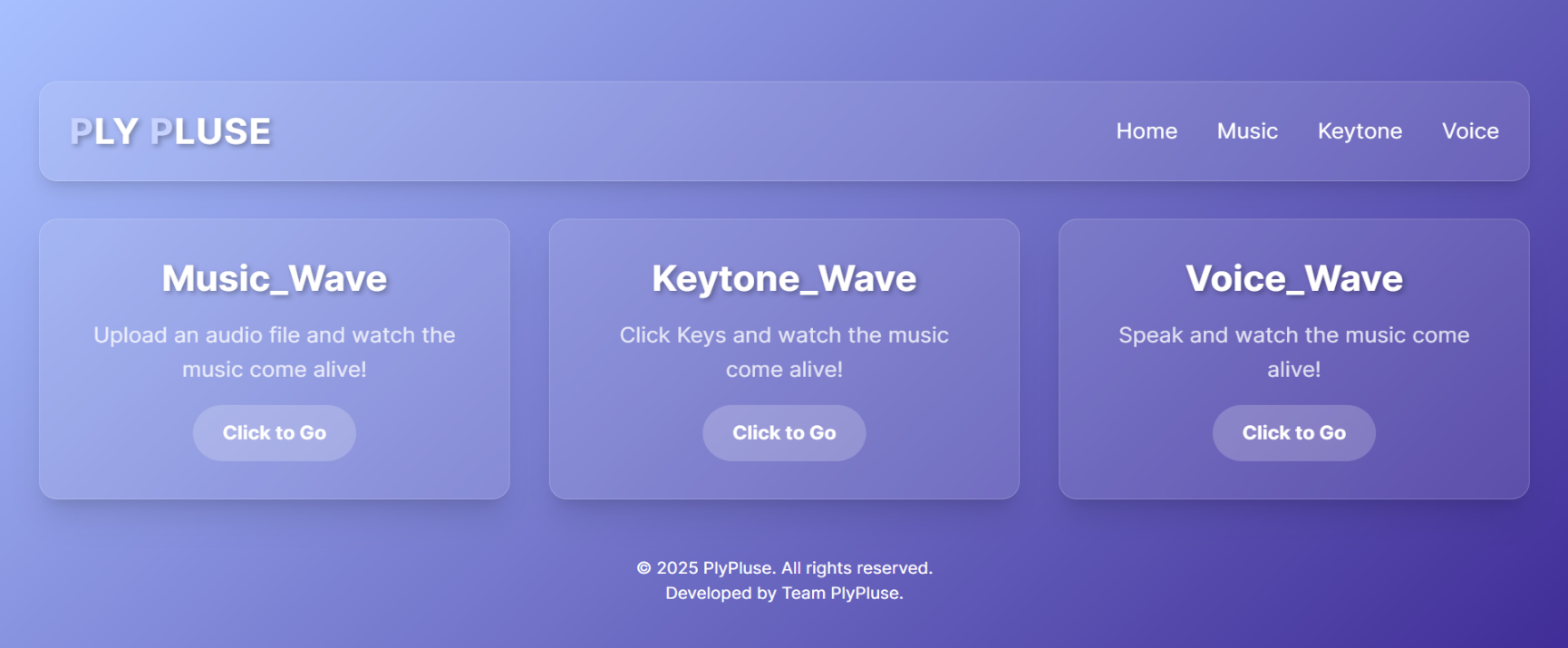
<p>Developed by Team PlyPluse.</p>

</footer>

</body>

</html>

**Output:**

****

**Keytone\_wave.html**

<!-- Developed By G Raghu

K Goutham

G Shashank

Sk Sameer-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Keytone Wave</title>

<!-- Favicon -->

<link href="favicon\_io/1.png" rel="icon" />

<meta

content="Music Wave, Keytone Wave, Voice Wave, audio visualization, music analysis, interactive music, sound visualization, real-time audio processing"

name="keywords"

/>

<meta

content="PlyPluse offers interactive audio visualization experiences. Explore Music Wave for uploaded audio, Keytone Wave for keyboard input, and Voice Wave for real-time voice visualization."

name="description"

/>

<script src="https://cdn.tailwindcss.com"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/tone/14.8.49/Tone.js"></script>

<style>

body {

font-family: "Inter", sans-serif;

}

canvas {

display: block;

width: 100%;

height: 200px;

}

.footer {

margin-top: 3rem;

text-align: center;

color: #ffffff; /\* gray-400 \*/

font-size: 0.875rem;

}

.glassmorphic {

background: rgba(

255,

255,

255,

0.1

); /\* Slightly transparent white background \*/

backdrop-filter: blur(10px); /\* Blur effect \*/

border: 1px solid rgba(255, 255, 255, 0.2); /\* Subtle white border \*/

box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37); /\* Shadow effect \*/

border-radius: 15px; /\* Rounded corners \*/

}

</style>

</head>

<body

class="bg-gradient-to-br from-gray-900 via-purple-900 to-indigo-900 text-white p-6 space-y-10"

style="padding-bottom: 6.5%"

>

<!-- Navigation Bar -->

<nav

class="glassmorphic w-full max-w-7xl mx-auto p-4 sm:p-6 mb-8 flex flex-col sm:flex-row items-center justify-between space-y-4 sm:space-y-0 shadow-lg"

>

<!-- Logo -->

<div class="flex items-center">

<span

class="text-white text-3xl font-extrabold tracking-wide text-shadow"

>

<span class="text-indigo-200">P</span>LY

<span class="text-indigo-200">P</span>LUSE

</span>

</div>

<!-- Navigation Blocks -->

<div

class="flex flex-col sm:flex-row space-y-3 sm:space-y-0 sm:space-x-8 text-white text-lg font-medium"

>

<a

href="index.html"

class="hover:text-indigo-200 transition duration-300"

>Home</a

>

<a

href="Music\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Music</a

>

<a

href="Keytone\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Keytone</a

>

<a

href="Voice\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Voice</a

>

</div>

</nav>

<!-- Main Content - Horizontal Card Blocks -->

<h1

class="text-5xl text-center font-extrabold text-transparent bg-clip-text bg-gradient-to-r from-purple-400 to-indigo-600"

>

Keytone Wave

</h1>

<!-- Play/Pause Button -->

<div class="text-center">

<button

id="togglePlaybackButton"

class="px-6 py-3 rounded-full bg-indigo-600 hover:bg-indigo-700"

>

<span id="playPauseIcon">▶</span>

<span id="playPauseText">Start Audio</span>

</button>

</div>

<!-- Visualizer -->

<div

class="max-w-4xl mx-auto bg-gray-800 rounded-xl p-4 border border-indigo-700"

>

<h2 class="text-2xl text-indigo-300 text-center mb-2">

Synth Visualizer

</h2>

<canvas id="visualizerCanvas"></canvas>

</div>

<!-- Controls -->

<div

class="max-w-4xl mx-auto bg-gray-800 rounded-xl p-4 border border-purple-700 grid grid-cols-1 md:grid-cols-3 gap-6"

>

<div>

<label class="block text-gray-300">Volume</label>

<input

id="volume"

type="range"

min="-40"

max="0"

value="0"

class="w-full"

/>

<span id="volumeValue" class="text-sm text-gray-400">0 dB</span>

</div>

<div>

<label class="block text-gray-300">Detune</label>

<input

id="detune"

type="range"

min="-1200"

max="1200"

value="0"

class="w-full"

/>

<span id="detuneValue" class="text-sm text-gray-400">0 cents</span>

</div>

<div>

<label class="block text-gray-300">Oscillator Type</label>

<select

id="oscType"

class="w-full bg-gray-700 border border-gray-600 text-white rounded px-2 py-1"

>

<option value="sine">Sine</option>

<option value="sawtooth">Sawtooth</option>

<option value="triangle">Triangle</option>

<option value="square">Square</option>

</select>

</div>

</div>

<!-- Keyboard -->

<div

class="max-w-4xl mx-auto bg-gray-800 rounded-xl p-4 border border-green-700"

>

<h2 class="text-2xl text-green-300 text-center mb-2">Play the Synth</h2>

<div id="keyboard" class="flex flex-wrap justify-center gap-2"></div>

<footer class="footer">

<p>&copy; 2025 PlyPluse. All rights reserved.</p>

<p>Developed by Team PlyPluse.</p>

</footer>

</div>

<script>

const notes = [

{ name: "C4", key: "a" },

{ name: "D4", key: "s" },

{ name: "E4", key: "d" },

{ name: "F4", key: "f" },

{ name: "G4", key: "g" },

{ name: "A4", key: "h" },

{ name: "B4", key: "j" },

{ name: "C5", key: "k" },

{ name: "D5", key: "l" },

{ name: "E5", key: ";" },

{ name: "F5", key: "'" },

{ name: "G5", key: "z" },

{ name: "A5", key: "x" },

{ name: "B5", key: "c" },

{ name: "C6", key: "v" },

];

let synth,

analyser,

isPlaying = false,

activeNotes = new Set(),

animationFrameId = null;

const canvas = document.getElementById("visualizerCanvas");

const keyboard = document.getElementById("keyboard");

const togglePlaybackButton = document.getElementById(

"togglePlaybackButton"

);

const playPauseIcon = document.getElementById("playPauseIcon");

const playPauseText = document.getElementById("playPauseText");

const volumeSlider = document.getElementById("volume");

const detuneSlider = document.getElementById("detune");

const oscType = document.getElementById("oscType");

const volumeValue = document.getElementById("volumeValue");

const detuneValue = document.getElementById("detuneValue");

function initSynth() {

synth = new Tone.PolySynth(Tone.Synth, {

oscillator: { type: oscType.value },

envelope: { attack: 0.005, decay: 0.1, sustain: 0.3, release: 1 },

volume: volumeSlider.value,

detune: detuneSlider.value,

}).toDestination();

analyser = new Tone.Analyser("waveform", 1024);

synth.connect(analyser);

}

function drawVisualizer() {

if (!analyser || !canvas || !isPlaying) return;

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

const buffer = analyser.getValue();

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

ctx.lineWidth = 2;

ctx.strokeStyle = "#6366f1";

ctx.beginPath();

const sliceWidth = canvas.width / buffer.length;

let x = 0;

for (let i = 0; i < buffer.length; i++) {

const v = buffer[i];

const y = (v \* canvas.height) / 2 + canvas.height / 2;

i === 0 ? ctx.moveTo(x, y) : ctx.lineTo(x, y);

x += sliceWidth;

}

ctx.lineTo(canvas.width, canvas.height / 2);

ctx.stroke();

animationFrameId = requestAnimationFrame(drawVisualizer);

}

function playNote(note) {

if (synth && Tone.context.state === "running") {

synth.triggerAttack(note);

activeNotes.add(note);

}

}

function releaseNote(note) {

if (synth && activeNotes.has(note)) {

synth.triggerRelease(note);

activeNotes.delete(note);

}

}

function createKeyboard() {

notes.forEach((n) => {

const btn = document.createElement("button");

btn.textContent = `${n.name}\n(${n.key.toUpperCase()})`;

btn.className =

"w-16 h-32 bg-gray-700 rounded text-xs p-2 hover:bg-gray-600";

btn.dataset.note = n.name;

btn.onmousedown = () => playNote(n.name);

btn.onmouseup = () => releaseNote(n.name);

btn.onmouseleave = () => releaseNote(n.name);

btn.ontouchstart = (e) => {

e.preventDefault();

playNote(n.name);

};

btn.ontouchend = () => releaseNote(n.name);

keyboard.appendChild(btn);

});

}

togglePlaybackButton.onclick = async () => {

if (!synth) initSynth();

if (Tone.context.state === "suspended") {

await Tone.context.resume();

isPlaying = true;

playPauseIcon.textContent = "⏸";

playPauseText.textContent = "Stop Audio";

drawVisualizer();

} else {

// Stop all active notes

activeNotes.forEach((note) => {

synth.triggerRelease(note);

});

activeNotes.clear();

Tone.context.suspend();

isPlaying = false;

playPauseIcon.textContent = "▶";

playPauseText.textContent = "Start Audio";

cancelAnimationFrame(animationFrameId);

}

};

volumeSlider.oninput = () => {

volumeValue.textContent = `${volumeSlider.value} dB`;

if (synth) synth.volume.value = volumeSlider.value;

};

detuneSlider.oninput = () => {

detuneValue.textContent = `${detuneSlider.value} cents`;

if (synth) synth.detune.value = detuneSlider.value;

};

oscType.onchange = () => {

if (synth) synth.dispose();

initSynth();

};

window.onkeydown = (e) => {

const note = notes.find((n) => n.key === e.key.toLowerCase());

if (note) playNote(note.name);

};

window.onkeyup = (e) => {

const note = notes.find((n) => n.key === e.key.toLowerCase());

if (note) releaseNote(note.name);

};

window.onload = () => {

initSynth();

createKeyboard();

canvas.width = canvas.offsetWidth;

};

</script>

</body>

</html>

**Output:**

**Music\_wave.html**

<!-- Developed By G Raghu

K Goutham

G Shashank

Sk Sameer-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Music Wave</title>

<!-- Favicon -->

<link href="favicon\_io/1.png" rel="icon" />

<meta

content="Music Wave, Keytone Wave, Voice Wave, audio visualization, music analysis, interactive music, sound visualization, real-time audio processing"

name="keywords"

/>

<meta

content="PlyPluse offers interactive audio visualization experiences. Explore Music Wave for uploaded audio, Keytone Wave for keyboard input, and Voice Wave for real-time voice visualization."

name="description"

/>

<script src="https://cdn.tailwindcss.com"></script>

<link rel="preconnect" href="https://fonts.googleapis.com" />

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<link

href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;800&display=swap"

rel="stylesheet"

/>

<style>

/\* General Setup & Font Import \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: "Inter", sans-serif;

color: white;

background: linear-gradient(

135deg,

#11142d,

#321c94

); /\* Example gradient from index.html \*/

min-height: 100vh; /\* Ensure body takes full viewport height \*/

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between; /\* Space out content and footer \*/

padding: 0 1rem; /\* Add horizontal padding to body \*/

}

/\* Custom styles for glassmorphism effect from index.html \*/

.glassmorphic {

background: rgba(

255,

255,

255,

0.1

); /\* Slightly transparent white background \*/

backdrop-filter: blur(10px); /\* Blur effect \*/

border: 1px solid rgba(255, 255, 255, 0.2); /\* Subtle white border \*/

box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37); /\* Shadow effect \*/

border-radius: 15px; /\* Rounded corners \*/

}

.text-shadow {

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3); /\* Add subtle text shadow for readability \*/

}

/\* Main container adjusted for flex column and content scrolling \*/

.app-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

width: 100%;

flex-grow: 1; /\* Allow container to grow and push footer down \*/

padding-top: 2rem; /\* Space from navbar \*/

padding-bottom: 2rem; /\* Space before footer \*/

}

/\* Header styles \*/

.header {

margin-bottom: 2rem;

text-align: center;

}

.title {

font-size: 3.75rem; /\* 5xl \*/

font-weight: 800; /\* extrabold \*/

margin-bottom: 1rem;

text-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);

}

.subtitle {

font-size: 1.25rem; /\* xl \*/

color: #c7d2fe; /\* indigo-200 \*/

}

/\* Main content card with glass effect \*/

.main-content {

background-color: rgba(31, 41, 55, 0.7); /\* gray-800 with opacity \*/

backdrop-filter: blur(10px);

border-radius: 0.75rem; /\* rounded-xl \*/

box-shadow: 0 25px 50px -12px rgba(0, 0, 0, 0.25);

padding: 2rem;

max-width: 56rem; /\* max-w-4xl \*/

width: 100%;

display: flex;

flex-direction: column;

align-items: center;

gap: 2rem;

}

/\* File input styling \*/

.file-input-container {

width: 100%;

text-align: center;

}

.file-label {

display: block;

font-size: 1.25rem;

font-weight: 600;

margin-bottom: 0.75rem;

color: #a5b4fc; /\* indigo-300 \*/

}

#audio-file {

color: #d1d5db; /\* gray-300 \*/

font-size: 0.875rem;

border-radius: 9999px;

width: 100%;

}

/\* Styling the button part of the file input \*/

#audio-file::file-selector-button {

margin-right: 1rem;

padding: 0.5rem 1rem;

border-radius: 9999px;

border: 0;

font-size: 0.875rem;

font-weight: 600;

background-color: #8b5cf6; /\* purple-500 \*/

color: white;

cursor: pointer;

transition: background-color 0.2s ease-in-out;

}

#audio-file::file-selector-button:hover {

background-color: #7c3aed; /\* purple-600 \*/

}

/\* Canvas container \*/

.canvas-container {

width: 100%;

height: 20rem; /\* h-80 \*/

display: flex;

align-items: center;

justify-content: center;

background-color: #111827; /\* gray-900 \*/

border-radius: 0.5rem;

overflow: hidden;

box-shadow: inset 0 2px 4px 0 rgba(0, 0, 0, 0.05);

}

#visualizer-canvas {

border-radius: 0.5rem;

border: 1px solid #8b5cf6; /\* purple-500 \*/

box-shadow: 0 2px 10px rgba(139, 92, 246, 0.3);

width: 100%;

height: 100%;

}

/\* Controls styling \*/

.controls-container {

width: 100%;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

gap: 1.5rem;

margin-top: 1rem;

}

.play-pause-btn {

padding: 1rem 2rem;

border-radius: 9999px;

font-size: 1.125rem;

font-weight: 700;

transition: all 0.3s ease-in-out;

transform-origin: center;

border: none;

cursor: pointer;

background-image: linear-gradient(

to right,

#22c55e,

#14b8a6

); /\* from-green-500 to-teal-500 \*/

color: white;

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);

display: flex;

align-items: center;

gap: 0.5rem;

}

.play-pause-btn:hover {

background-image: linear-gradient(

to right,

#16a34a,

#0d9488

); /\* from-green-600 to-teal-600 \*/

}

.play-pause-btn:active {

transform: scale(0.95);

}

.play-pause-btn:disabled {

background: #4b5563; /\* gray-600 \*/

color: #9ca3af; /\* gray-400 \*/

cursor: not-allowed;

box-shadow: none;

}

.play-pause-btn .icon {

width: 1.5rem;

height: 1.5rem;

}

.volume-control {

display: flex;

align-items: center;

gap: 1rem;

width: 100%;

max-width: 250px;

}

.volume-label {

font-size: 1.125rem;

font-weight: 600;

color: #a5b4fc; /\* indigo-300 \*/

}

#volume-slider {

width: 100%;

-webkit-appearance: none;

appearance: none;

height: 0.5rem;

border-radius: 0.5rem;

background-color: #374151; /\* gray-700 \*/

cursor: pointer;

}

/\* Styling for Webkit (Chrome, Safari) \*/

#volume-slider::-webkit-slider-thumb {

-webkit-appearance: none;

appearance: none;

width: 1rem;

height: 1rem;

border-radius: 50%;

background-color: #8b5cf6; /\* purple-500 \*/

box-shadow: 0 0 5px rgba(139, 92, 246, 0.5);

transition: background-color 0.2s ease-in-out;

}

#volume-slider::-webkit-slider-thumb:hover {

background-color: #7c3aed; /\* purple-600 \*/

}

/\* Styling for Firefox \*/

#volume-slider::-moz-range-thumb {

width: 1rem;

height: 1rem;

border-radius: 50%;

background-color: #8b5cf6;

border: none;

box-shadow: 0 0 5px rgba(139, 92, 246, 0.5);

transition: background-color 0.2s ease-in-out;

}

#volume-slider::-moz-range-thumb:hover {

background-color: #7c3aed;

}

/\* Footer styling \*/

.footer {

margin-top: 3rem;

text-align: center;

color: #fefefe; /\* gray-400 \*/

font-size: 0.875rem;

}

/\* Responsive adjustments for smaller screens \*/

@media (min-width: 640px) {

.controls-container {

flex-direction: row;

}

.volume-control {

width: auto;

}

}

.visualization-select {

display: flex;

align-items: center;

gap: 1rem;

font-size: 1rem;

font-weight: 600;

color: #a5b4fc;

}

#visual-style {

padding: 0.5rem 1rem;

border-radius: 0.5rem;

border: none;

background-color: #374151;

color: white;

font-weight: 600;

cursor: pointer;

}

</style>

<script>

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

// DOM Element References

let startTime = 0;

let pauseTime = 0;

const canvas = document.getElementById("visualizer-canvas");

const canvasCtx = canvas.getContext("2d");

const audioFileInput = document.getElementById("audio-file");

const playPauseBtn = document.getElementById("play-pause-btn");

const playPauseText = document.getElementById("play-pause-text");

const playIcon = document.getElementById("play-icon");

const pauseIcon = document.getElementById("pause-icon");

const volumeSlider = document.getElementById("volume-slider");

// Web Audio API State

let audioContext;

let analyser;

let source;

let gainNode;

// Application State

let audioBuffer = null;

let isPlaying = false;

let animationFrameId;

// Initialize Web Audio API components

function initAudioContext() {

if (!audioContext) {

const AudioContext =

window.AudioContext || window.webkitAudioContext;

audioContext = new AudioContext();

analyser = audioContext.createAnalyser();

analyser.fftSize = 256;

gainNode = audioContext.createGain();

gainNode.gain.value = volumeSlider.value;

analyser.connect(gainNode);

gainNode.connect(audioContext.destination);

}

}

// Handle file selection

audioFileInput.addEventListener("change", async (event) => {

const file = event.target.files[0];

if (!file) return;

// Initialize AudioContext on first user interaction

initAudioContext();

// Stop any currently playing audio

if (source) {

stopAudio();

}

pauseTime = 0;

startTime = 0;

const reader = new FileReader();

reader.onload = async (e) => {

try {

audioBuffer = await audioContext.decodeAudioData(e.target.result);

playPauseBtn.disabled = false;

playAudio();

} catch (error) {

console.error("Error decoding audio data:", error);

alert("Could not decode audio file. Please try another file.");

playPauseBtn.disabled = true;

}

};

reader.readAsArrayBuffer(file);

});

// Play/Pause button handler

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

if (!audioBuffer) return;

if (isPlaying) {

stopAudio();

} else {

// Resume context if it was suspended

if (audioContext.state === "suspended") {

audioContext.resume().then(playAudio);

} else {

playAudio();

}

}

});

// Volume slider handler

volumeSlider.addEventListener("input", (event) => {

if (gainNode) {

gainNode.gain.value = event.target.value;

}

});

// Function to play audio

function playAudio() {

if (!audioBuffer || !audioContext) return;

source = audioContext.createBufferSource();

source.buffer = audioBuffer;

source.connect(analyser);

source.onended = () => {

if (isPlaying) stopAudio();

};

const offset = pauseTime || 0;

startTime = audioContext.currentTime - offset;

source.start(0, offset);

isPlaying = true;

updatePlayPauseUI();

draw();

}

// Function to stop audio

function stopAudio() {

if (source) {

try {

source.stop();

source.disconnect();

} catch (e) {

console.warn("Audio already stopped.");

}

source = null;

}

pauseTime = audioContext.currentTime - startTime; // Track paused time

isPlaying = false;

updatePlayPauseUI();

cancelAnimationFrame(animationFrameId);

canvasCtx.clearRect(0, 0, canvas.width, canvas.height);

}

// Update button text and icon

function updatePlayPauseUI() {

if (isPlaying) {

playPauseText.textContent = "Pause";

playIcon.style.display = "none";

pauseIcon.style.display = "inline";

} else {

playPauseText.textContent = "Play";

playIcon.style.display = "inline";

pauseIcon.style.display = "none";

}

}

// Visualization drawing function

const styleSelect = document.getElementById("visual-style");

let currentStyle = "radial";

styleSelect.addEventListener("change", (e) => {

currentStyle = e.target.value;

});

function draw() {

if (!isPlaying) return;

animationFrameId = requestAnimationFrame(draw);

const bufferLength = analyser.frequencyBinCount;

const dataArray = new Uint8Array(bufferLength);

analyser.getByteFrequencyData(dataArray);

canvasCtx.clearRect(0, 0, canvas.width, canvas.height);

switch (currentStyle) {

case "sine":

drawSineWave(dataArray, bufferLength);

break;

case "bars":

drawBars(dataArray, bufferLength);

break;

case "circle":

drawCircleWave(dataArray, bufferLength);

break;

default:

drawRadial(dataArray, bufferLength);

}

}

function drawSineWave(dataArray, bufferLength) {

canvasCtx.beginPath();

canvasCtx.lineWidth = 2;

const sliceWidth = canvas.width / bufferLength;

let x = 0;

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

const v = dataArray[i] / 255;

const y = canvas.height / 2 + v \* 100 \* Math.sin(i \* 0.2);

// Dynamic hue based on index and frequency

const hue = (i + dataArray[i] \* 0.5) % 360;

canvasCtx.strokeStyle = `hsl(${hue}, 80%, 70%)`; // Use HSL for vibrant colors

if (i === 0) {

canvasCtx.moveTo(x, y);

} else {

canvasCtx.lineTo(x, y);

}

x += sliceWidth;

}

canvasCtx.stroke();

}

function drawBars(dataArray, bufferLength) {

const barWidth = (canvas.width / bufferLength) \* 2.5;

let x = 0;

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

const barHeight = dataArray[i];

// Cycle through hues and adjust saturation/lightness

const hue = (i \* 5) % 360;

const saturation = 70 + (barHeight / 255) \* 30;

const lightness = 50 + (barHeight / 255) \* 30;

canvasCtx.fillStyle = `hsl(${hue}, ${saturation}%, ${lightness}%)`;

canvasCtx.fillRect(

x,

canvas.height - barHeight,

barWidth,

barHeight

);

x += barWidth + 1;

}

}

function drawCircleWave(dataArray, bufferLength) {

const centerX = canvas.width / 2;

const centerY = canvas.height / 2;

canvasCtx.beginPath();

canvasCtx.lineWidth = 2;

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

const angle = (i / bufferLength) \* Math.PI \* 2;

const radius = 60 + dataArray[i] / 2;

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

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

// Dynamic hue for vibrant circle

const hue = (i \* 2 + dataArray[i]) % 360;

canvasCtx.strokeStyle = `hsl(${hue}, 90%, 60%)`;

if (i === 0) {

canvasCtx.moveTo(x, y);

} else {

canvasCtx.lineTo(x, y);

}

}

canvasCtx.closePath();

canvasCtx.stroke();

}

function drawRadial(dataArray, bufferLength) {

const centerX = canvas.width / 2;

const centerY = canvas.height / 2;

const radius = 80;

const angleStep = (Math.PI \* 2) / bufferLength;

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

const value = dataArray[i];

const barLength = value \* 1.2;

const angle = i \* angleStep;

const x1 = centerX + Math.cos(angle) \* radius;

const y1 = centerY + Math.sin(angle) \* radius;

const x2 = centerX + Math.cos(angle) \* (radius + barLength);

const y2 = centerY + Math.sin(angle) \* (radius + barLength);

// Dynamic HSL colors for radial bars

const hue = (i \* 3 + value) % 360;

const saturation = 80 + (value / 255) \* 20;

const lightness = 60 + (value / 255) \* 20;

canvasCtx.strokeStyle = `hsl(${hue}, ${saturation}%, ${lightness}%)`;

canvasCtx.lineWidth = 2;

canvasCtx.beginPath();

canvasCtx.moveTo(x1, y1);

canvasCtx.lineTo(x2, y2);

canvasCtx.stroke();

}

}

// Responsive Canvas Resizing

function resizeCanvas() {

const container = document.querySelector(".canvas-container");

canvas.width = container.clientWidth;

canvas.height = container.clientHeight;

}

window.addEventListener("resize", resizeCanvas);

resizeCanvas(); // Initial resize

});

</script>

</head>

<body>

<nav

class="glassmorphic w-full max-w-7xl mx-auto p-4 sm:p-6 mt-8 flex flex-col sm:flex-row items-center justify-between space-y-4 sm:space-y-0 shadow-lg"

>

<div class="flex items-center">

<span

class="text-white text-3xl font-extrabold tracking-wide text-shadow"

>

<span class="text-indigo-200">P</span>LY

<span class="text-indigo-200">P</span>LUSE

</span>

</div>

<div

class="flex flex-col sm:flex-row space-y-3 sm:space-y-0 sm:space-x-8 text-white text-lg font-medium"

>

<a

href="index.html"

class="hover:text-indigo-200 transition duration-300"

>Home</a

>

<a

href="Music\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Music</a

>

<a

href="Keytone\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Keytone</a

>

<a

href="Voice\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Voice</a

>

</div>

</nav>

<div class="app-container">

<header class="header">

<h1 class="title">Music Wave</h1>

<p class="subtitle">

Upload an audio file and watch the music come alive!

</p>

</header>

<main class="main-content">

<div class="file-input-container">

<label for="audio-file" class="file-label"

>Select Your Audio File:</label

>

<input type="file" id="audio-file" accept="audio/\*" />

</div>

<div class="canvas-container">

<canvas id="visualizer-canvas"></canvas>

</div>

<div class="controls-container">

<div class="visualization-select">

<label for="visual-style">Visualization Style:</label>

<select id="visual-style">

<option value="radial">Radial Bars</option>

<option value="sine">Sine Wave</option>

<option value="bars">Bars</option>

<option value="circle">Circle Wave</option>

</select>

</div>

<button id="play-pause-btn" class="play-pause-btn" disabled>

<span id="play-icon">

<svg class="icon" fill="currentColor" viewBox="0 0 20 20">

<path

d="M10 18a8 8 0 100-16 8 8 0 000 16zM9.5 8H8V6.5h1.5V8zm2.5 0H11V6.5h1.5V8zM10 4a6 6 0 110 12 6 6 0 010-12z"

/>

</svg>

</span>

<span id="pause-icon" style="display: none">

<svg class="icon" fill="currentColor" viewBox="0 0 20 20">

<path

d="M10 18a8 8 0 100-16 8 8 0 000 16zM9.5 8H8V6.5h1.5V8zm2.5 0H11V6.5h1.5V8zM10 4a6 6 0 110 12 6 6 0 010-12z"

/>

</svg>

</span>

<span id="play-pause-text">Play</span>

</button>

<div class="volume-control">

<label for="volume-slider" class="volume-label">Volume:</label>

<input

type="range"

id="volume-slider"

min="0"

max="1"

step="0.01"

value="0.7"

/>

</div>

</div>

</main>

<footer class="footer">

<p>&copy; 2025 PlyPluse. All rights reserved.</p>

<p>Developed by Team PlyPluse.</p>

</footer>

</div>

</body>

</html>

**Output:**

**Voice\_wave.html**

<!-- Developed By G Raghu

K Goutham

G Shashank

Sk Sameer-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Voice Wave</title>

<!-- Favicon -->

<link href="favicon\_io/1.png" rel="icon" />

<link href="../favicon.png" rel="icon" />

<meta

content="Music Wave, Keytone Wave, Voice Wave, audio visualization, music analysis, interactive music, sound visualization, real-time audio processing"

name="keywords"

/>

<meta

content="PlyPluse offers interactive audio visualization experiences. Explore Music Wave for uploaded audio, Keytone Wave for keyboard input, and Voice Wave for real-time voice visualization."

name="description"

/>

<script src="https://cdn.tailwindcss.com"></script>

<style>

@import url("https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap");

/\* Body & background \*/

body {

margin: 0;

min-height: 100vh; /\* Ensure body takes full viewport height \*/

background: linear-gradient(135deg, #52002d, #5c3db9);

font-family: "Inter", sans-serif;

display: flex;

flex-direction: column;

justify-content: space-between; /\* Changed to space-between to push footer to bottom \*/

align-items: center;

color: #e0e6f0;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

/\* Removed overflow: hidden; to allow scrolling \*/

}

/\* Container with glassmorphism \*/

.glass-container {

background: rgba(255, 255, 255, 0.1);

/\*box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);\*/

backdrop-filter: blur(10px);

-webkit-backdrop-filter: blur(10px);

border-radius: 20px;

border: 1px solid rgba(255, 255, 255, 0.18);

width: 90vw;

max-width: 900px;

padding: 30px 40px;

display: flex;

flex-direction: column;

align-items: center;

user-select: none;

/\* Removed margin-top: 2rem; as space-between will handle vertical spacing \*/

flex-grow: 1; /\* Allow the container to grow and push the footer down \*/

margin-top: 2rem; /\* Add some space from the navbar \*/

margin-bottom: 2rem; /\* Add some space before the footer \*/

}

h2 {

font-weight: 700;

font-size: 2rem;

margin-bottom: 1.5rem;

letter-spacing: 1.3px;

text-transform: uppercase;

color: #bcd4e6cc;

}

canvas {

width: 100%;

height: 300px;

border-radius: 15px;

box-shadow: 0 0 20px #4ac1ffaa;

background: rgba(0, 0, 0, 0.836);

margin-bottom: 1.8rem;

display: block;

}

button {

background: rgba(255 255 255 / 0.15);

border: 1.5px solid rgba(255 255 255 / 0.4);

padding: 14px 50px;

font-size: 1.1rem;

font-weight: 700;

color: #d0e8ffcc;

border-radius: 40px;

cursor: pointer;

transition: background-color 0.3s ease, box-shadow 0.3s ease;

/\* box-shadow: 0 8px 20px rgba(74, 193, 255, 0.3);\*/

backdrop-filter: blur(8px);

-webkit-backdrop-filter: blur(8px);

user-select: none;

/\*text-shadow: 0 0 4px rgba(74, 193, 255, 0.7);\*/

}

button:hover {

background: rgba(255 255 255 / 0.3);

/\*box-shadow: 0 12px 35px rgba(74, 193, 255, 0.6);\*/

color: #ffffffdd;

}

/\* Responsive font size \*/

@media (max-width: 480px) {

h2 {

font-size: 1.5rem;

}

button {

padding: 12px 30px;

font-size: 1rem;

}

}

.footer {

margin-top: 3rem;

text-align: center;

color: #ffffff; /\* gray-400 \*/

font-size: 0.875rem;

}

/\* Custom styles for glassmorphism effect \*/

.glassmorphic {

background: rgba(

255,

255,

255,

0.1

); /\* Slightly transparent white background \*/

backdrop-filter: blur(10px); /\* Blur effect \*/

border: 1px solid rgba(255, 255, 255, 0.2); /\* Subtle white border \*/

box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37); /\* Shadow effect \*/

border-radius: 15px; /\* Rounded corners \*/

}

.text-shadow {

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3); /\* Add subtle text shadow for readability \*/

}

</style>

</head>

<body class="p-4 sm:p-6 md:p-8">

<nav

class="glassmorphic w-full max-w-7xl mx-auto p-4 sm:p-6 mb-8 flex flex-col sm:flex-row items-center justify-between space-y-4 sm:space-y-0 shadow-lg"

>

<div class="flex items-center">

<span

class="text-white text-3xl font-extrabold tracking-wide text-shadow"

>

<span class="text-indigo-200">P</span>LY

<span class="text-indigo-200">P</span>LUSE

</span>

</div>

<div

class="flex flex-col sm:flex-row space-y-3 sm:space-y-0 sm:space-x-8 text-white text-lg font-medium"

>

<a

href="index.html"

class="hover:text-indigo-200 transition duration-300"

>Home</a

>

<a

href="Music\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Music</a

>

<a

href="Keytone\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Keytone</a

>

<a

href="Voice\_Wave.html"

class="hover:text-indigo-200 transition duration-300"

>Voice</a

>

</div>

</nav>

<div class="glass-container">

<h2

style="

font-weight: 700;

font-size: 2rem;

margin-bottom: 1.5rem;

letter-spacing: 1.3px;

text-transform: uppercase;

color: #ffffffcc;

"

>

Speak to Visualize

</h2>

<canvas id="visualizer"></canvas>

<button

id="startBtn"

style="

background: rgba(255 255 255 / 0.15);

border: 1.5px solid rgba(255 255 255 / 0.4);

padding: 14px 50px;

font-size: 1.1rem;

font-weight: 700;

color: #ffffffcc;

border-radius: 40px;

cursor: pointer;

transition: background-color 0.3s ease, box-shadow 0.3s ease;

backdrop-filter: blur(8px);

-webkit-backdrop-filter: blur(8px);

user-select: none;

"

>

Start Microphone

</button>

</div>

<footer class="footer">

<p>&copy; 2025 PlyPluse. All rights reserved.</p>

<p>Developed by Team PlyPluse.</p>

</footer>

<script>

const canvas = document.getElementById("visualizer");

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

let audioCtx;

let analyser;

let dataArray;

let bufferLength;

// Resize canvas to full width and fixed height

function resizeCanvas() {

canvas.width = canvas.clientWidth;

canvas.height = 300;

}

resizeCanvas();

window.addEventListener("resize", resizeCanvas);

const startBtn = document.getElementById("startBtn");

startBtn.addEventListener("click", async () => {

if (!navigator.mediaDevices.getUserMedia) {

alert("Your browser does not support microphone access.");

return;

}

startBtn.disabled = true;

startBtn.textContent = "Listening...";

const stream = await navigator.mediaDevices.getUserMedia({

audio: true,

});

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

const source = audioCtx.createMediaStreamSource(stream);

analyser = audioCtx.createAnalyser();

analyser.fftSize = 2048;

bufferLength = analyser.fftSize;

dataArray = new Uint8Array(bufferLength);

source.connect(analyser);

visualize();

});

function visualize() {

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

function draw() {

requestAnimationFrame(draw);

analyser.getByteTimeDomainData(dataArray);

// Calculate volume (RMS)

let sum = 0;

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

let val = (dataArray[i] - 128) / 128;

sum += val \* val;

}

let rms = Math.sqrt(sum / bufferLength);

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

if (rms > 0.02) {

// Threshold to only draw when speaking

// Gradient for waveform

const gradient = ctx.createLinearGradient(0, 0, canvas.width, 0);

gradient.addColorStop(0, "rgba(74, 193, 255, 0.9)");

gradient.addColorStop(0.5, "rgba(0, 255, 255, 0.8)");

gradient.addColorStop(1, "rgba(74, 193, 255, 0.9)");

ctx.lineWidth = 4;

ctx.strokeStyle = gradient;

ctx.shadowColor = "rgba(74, 193, 255, 0.7)";

ctx.shadowBlur = 20;

ctx.beginPath();

let sliceWidth = canvas.width / bufferLength;

let x = 0;

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

let v = dataArray[i] / 128.0; // normalized between 0 and 2

let y = v \* (canvas.height / 2);

if (i === 0) {

ctx.moveTo(x, y);

} else {

ctx.lineTo(x, y);

}

x += sliceWidth;

}

ctx.lineTo(canvas.width, canvas.height / 2);

ctx.stroke();

// Glow line below waveform

ctx.beginPath();

ctx.moveTo(0, canvas.height / 2 + 12);

ctx.lineTo(canvas.width, canvas.height / 2 + 12);

/\* ctx.strokeStyle = "rgba(0, 255, 255, 0.15)";

ctx.lineWidth = 16;

ctx.shadowBlur = 30;

ctx.stroke();\*/

} else {

// Subtle "waiting" text

ctx.font = "24px Inter, sans-serif";

ctx.fillStyle = "rgba(74, 193, 255, 0.3)";

ctx.textAlign = "center";

ctx.fillText(

"Speak to see the magic...",

canvas.width / 2,

canvas.height / 2

);

}

}

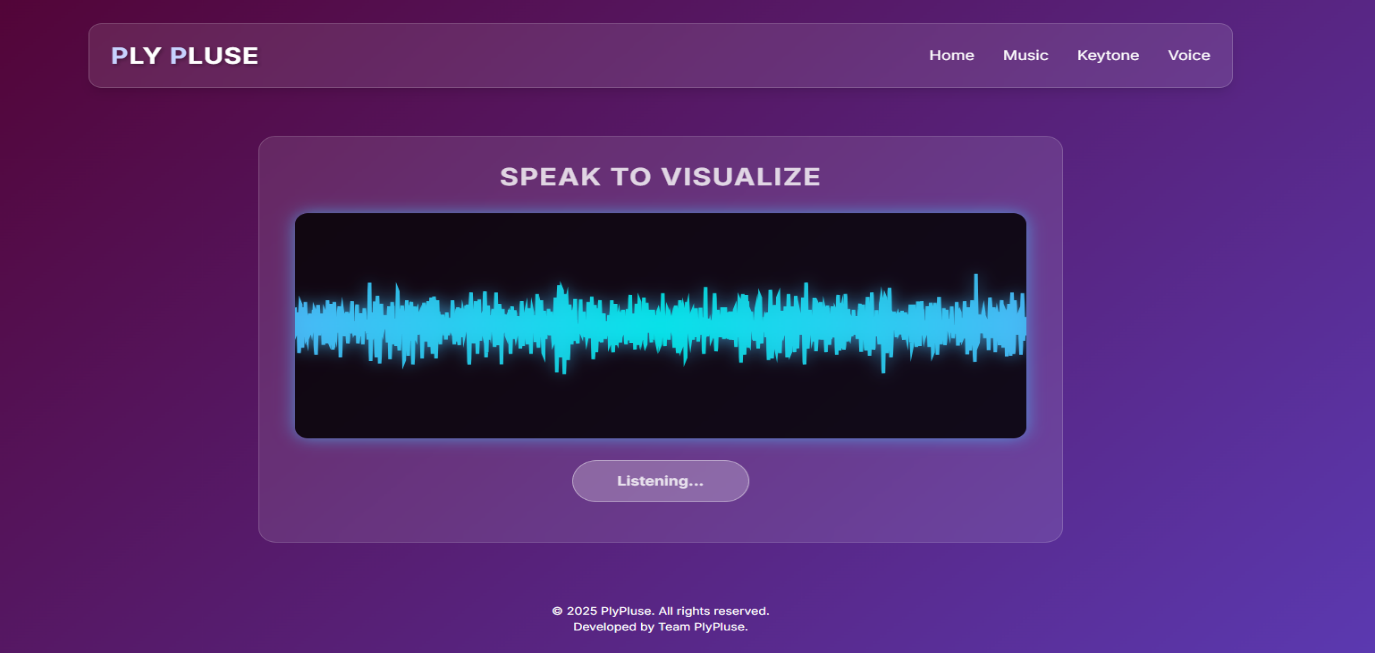
draw();

}

</script>

</body>

</html>

**Output:**