Strategic Architecture and Product Roadmap: AudioEmotion

1. Executive Summary and Strategic Vision

1.1 The Paradigm Shift in Affective Computing

The digital landscape is currently undergoing a fundamental transformation in how applications interact with human users. We are moving beyond the era of explicit command-based interfaces—where users must clearly state their intent via text or buttons—into an era of implicit, affective computing. In this new paradigm, applications must possess the capability to "read the room," understanding not just *what* is being said, but *how* it is being said. "AudioEmotion" represents a flagship implementation of this capability, designed to serve as a real-time, browser-based Speech Emotion Recognition (SER) utility. The core mandate for AudioEmotion is to detect and visualize eight distinct emotional states—Neutral, Calm, Happy, Sad, Angry, Fearful, Disgust, and Surprised—with zero latency and absolute privacy. This requirement dictates a radical departure from traditional cloud-based AI architectures. The prevailing model of streaming audio to a remote server for inference is fundamentally flawed for this use case due to two critical bottlenecks: network latency, which disrupts the cognitive feedback loop required for real-time interaction; and data sovereignty, as streaming biometric voice data to the cloud invites immense regulatory liability under GDPR, CCPA, and emerging AI governance frameworks.[1]
Therefore, the strategic direction for AudioEmotion is a "Local-First" architecture. By leveraging the convergence of the Web Audio API's AudioWorklet for glitch-free signal processing [3] and ONNX Runtime Web for efficient client-side neural network execution [5], we can deliver a professional-grade affective computing tool that runs entirely within the user's browser. This report outlines the exhaustive technical specifications, user experience strategies, and visual identity systems required to bring this vision to life.

1.2 Value Proposition and Market Fit

The utility of AudioEmotion extends across multiple high-value verticals. For tele-health and remote therapy, it provides clinicians with an objective, longitudinal record of patient affect, distinguishing between clinically significant states like "Calm" (low arousal, positive valence) versus "Neutral" (medium arousal, zero valence).[6] In Customer Experience (CX) management, the ability to instantly flag "Anger" or "Disgust" allows for proactive de-escalation in support scenarios. Furthermore, for the AI research community, AudioEmotion serves as a frictionless benchmark tool, allowing developers to test model performance against standard datasets without the overhead of Python environment management.
The following table summarizes the strategic differentiation of AudioEmotion compared to incumbent cloud-based solutions:

Feature	Cloud-Based Legacy Solutions	AudioEmotion (Local-First Strategy)	Strategic Advantage
Latency	200ms - 1500ms (Network dependent)	< 50ms (Inference time only)	Enables usage in real-time conversation coaching and interruption handling.
Privacy	Audio leaves device; GDPR compliance is complex.	Audio never leaves RAM; inherently GDPR compliant.	Eliminates data storage liability; high trust for enterprise adoption.[1]
Cost	Per-minute API metering (e.g., Google/AWS).	Zero marginal cost (Client compute).	Infinite scalability without linear operational expenditure growth.
Availability	Requires active internet connection.	Fully functional offline (PWA).	Critical for field work or low-bandwidth environments.

---

2. Theoretical Framework: The Taxonomy of Emotion

To build a robust detection system, we must first rigorously define the signals we are detecting. The requirement to distinguish eight specific emotions—Neutral, Calm, Happy, Sad, Angry, Fearful, Disgust, and Surprised—requires a nuanced understanding of acoustic psychology. These are not merely labels; they are distinct physiological states that manifest in specific prosodic features (pitch, energy, timbre) which our architecture must extract and classify.

2.1 The Circumplex Model vs. Categorical Classification

While the user interface will present categorical labels (e.g., "Happy"), the underlying machine learning strategy must account for the dimensional nature of emotion. Psychological research, particularly the work of Russell and Plutchik, organizes emotions along two primary axes: Valence (Positivity/Negativity) and Arousal (Energy/Intensity).[6]

- **Valence:** The intrinsic attractiveness (positive) or aversiveness (negative) of an event.
- **Arousal:** The state of being awoken or stimulated.

Understanding this dimensionality is critical for handling edge cases. For instance, "Angry" and "Fearful" are both High-Arousal/Negative-Valence states. To distinguish them, our model must detect subtle "Dominance" cues—Anger is dominant/aggressive, while Fear is submissive/avoidant.[9] Similarly, "Calm" and "Neutral" are often confused by basic models. "Calm" implies a positive valence with low arousal, whereas "Neutral" implies a lack of significant valence or arousal.

2.2 Acoustic Signatures of the Target Spectrum

The following analysis maps the eight target emotions to their specific acoustic footprints, guiding both our feature extraction strategy and our visual identity system.

Emotion ID	**Label**	**Acoustic Features (Prosody)**	**Valence**	**Arousal**	**Dominance**
01	**Neutral**	Flat fundamental frequency (F_0), steady rhythm, lack of distinct inflection.	Medium	Medium	Medium
02	**Calm**	Slower tempo, softer volume, lower pitch variance than neutral; often breathy.	Positive	Low	Low
03	**Happy**	Higher mean pitch, increased energy, faster tempo, upward inflections.	Positive	High	High
04	**Sad**	Low mean pitch, decreased energy, slow tempo, downward inflections, pauses.	Negative	Low	Low
05	**Angry**	High energy, loud volume, sharp/abrupt changes, high frequency energy (shimmer).	Negative	High	High
06	**Fearful**	High pitch,	Negative	High	Low

		rapid tempo, breathiness, tremors (jitter) in vocal cords.			
07	**Disgust**	Lower pitch, glottal stops, slower tempo, "throaty" quality, prolonged vowels.	Negative	Medium	Medium
08	**Surprised**	Sudden pitch spike, sharp inhalation, short duration, high initial energy.	Positive	High	Medium

This mapping reveals that simple energy detection (RMS) is insufficient. To distinguish "Disgust" (throaty, low pitch) from "Sad" (low pitch, low energy), the system must analyze spectral texture features like MFCCs (Mel-Frequency Cepstral Coefficients) which represent the shape of the vocal tract.[10]

3. Technical Architecture: The Client-Side Signal Processing Pipeline

The technical success of AudioEmotion hinges on its ability to process high-fidelity audio (typically 44.1kHz or 48kHz) in real-time without blocking the main UI thread. A blocked UI thread results in "janky" visualizations, destroying the user's perception of responsiveness. To achieve this, we will implement a multi-threaded architecture leveraging the specialized AudioWorklet and Web Worker interfaces.

3.1 The AudioWorklet and SharedArrayBuffer Pattern

Traditional web audio processing relied on ScriptProcessorNode, which ran on the main thread and introduced significant latency. The modern standard is the AudioWorklet, which runs audio processing code in a separate, high-priority thread synchronous with the audio hardware clock.[3]
However, passing data from the AudioWorklet (the producer) to the Machine Learning worker (the consumer) typically involves postMessage, which serializes data and causes garbage collection overhead. For AudioEmotion, we will implement a **Lock-Free Ring Buffer** using

SharedArrayBuffer (SAB). This allows both threads to access the same block of memory simultaneously, eliminating copying overhead.[4]

3.1.1 Implementation of the Circular Buffer

The circular buffer acts as a FIFO (First-In-First-Out) queue.

- **Capacity:** We require a buffer large enough to hold at least 10 seconds of audio to handle any inference jitter. At 16kHz (the native sampling rate of our ML models), this requires 16000 * 10 * 4 bytes (Float32) = ~640KB.
- **State Management:** We use Atomics to manage the writePointer (controlled by the AudioWorklet) and the readPointer (controlled by the ML Worker). This prevents race conditions where the ML model might read data that is currently being written.[4]
- **Security Requirements:** To utilize SharedArrayBuffer, the hosting environment must be isolated. The application must be served with the following HTTP headers to bypass the browser's cross-origin protections against Spectre vulnerabilities [13]:
  - Cross-Origin-Opener-Policy: same-origin
  - Cross-Origin-Embedder-Policy: require-corp

3.2 Digital Signal Processing (DSP) & Feature Extraction

While our primary classification will be performed by a deep learning model, the visualization layer (Siri-style waveform and spectral analysis) requires efficient, low-level feature extraction.

3.2.1 Library Selection: Essentia.js vs. Meyda vs. AudioFlux

We evaluated three primary JavaScript libraries for browser-based DSP:

Library	Architecture	Performance (MFCC Extraction)	Pros	Cons	Recommendation
Meyda.js	Pure JavaScript	Moderate	Easy to use; standard Web Audio API integration.[10]	Slower on large buffers; inconsistent with Python baselines (Librosa).[11]	Use for simple UI volume meters only.
Essentia.js	C++ compiled to WASM	**High** (Near native)	comprehensive feature set; excellent parity with offline analysis.[14]	Larger bundle size; steeper learning curve.	**Primary Choice for Visualizer DSP.**
AudioFlux	Python/C bound	Variable	High feature count.	Less mature JS support compared to	Secondary alternative.

				Essentia.	

Decision: We will utilize **Essentia.js**. Its WebAssembly backend allows it to compute complex spectral features (Spectral Centroid, MFCCs) faster than real-time, which is essential for driving the high-frame-rate visualizations required by the UI.[14] Specifically, the Essentia.js FrameCutter and FrameGenerator algorithms offer robust windowing functions that align closely with librosa, ensuring that the visual data accurately represents the acoustic reality.[11]

3.3 The Frontend Application Framework

The user interface is built on **React 18**, leveraging its concurrent rendering capabilities to handle high-frequency state updates without UI freezing.

- **State Management: Zustand**. Unlike Context API or Redux, which can trigger unnecessary re-renders of the entire component tree, Zustand allows for transient state updates. We can bind the high-frequency emotion probabilities directly to the visualization components (Radar Chart) via subscribers, bypassing the React virtual DOM diffing for the most performance-critical updates.[17]
- **Component Architecture:** The application will be structured as a Single Page Application (SPA) wrapped in an AudioContext provider.
    - AudioProvider: Manages the lifecycle of the AudioContext and microphone permissions.
    - InferenceEngine: A headless component that manages the Web Worker and ONNX runtime.
    - VisualizerCanvas: A React component wrapping a standard HTML5 <canvas> element, driven by a requestAnimationFrame loop to ensure 60fps rendering independent of React state.[18]

---

4. Machine Learning Pipeline: The Speech Emotion Recognition (SER) Engine

The heart of AudioEmotion is the deep learning model. The requirement to run this client-side imposes strict constraints on model size and computational complexity. We cannot simply deploy a massive 10GB parameter model; we must balance accuracy with the constraints of consumer hardware (laptops, tablets).

4.1 Model Architecture Selection

Traditional SER models relied on CNNs (Convolutional Neural Networks) taking spectrogram images as input. However, current State-of-the-Art (SOTA) performance is achieved by **Transformer-based models** which process raw audio waveforms and capture long-range temporal dependencies using self-attention mechanisms.[19]

Primary Candidate: Wav2Vec2-Large-Robust (Fine-Tuned)

We will utilize a derivative of the wav2vec2-large-robust architecture, specifically fine-tuned on emotion datasets such as MSP-Podcast or IEMOCAP.7

- **Justification:** The "Robust" variant of Wav2Vec2 is pre-trained on noisy audio data, making it resilient to the varied microphone quality and background noise inherent in web-based deployment.[9]
- **Performance:** Benchmarks indicate that this architecture achieves approximately 75-78% accuracy on standard datasets like IEMOCAP, significantly outperforming legacy statistical methods.[20]
- **Alternative: SenseVoice**. Recent benchmarks suggest SenseVoice (specifically SenseVoice-Small) offers competitive performance and speed, potentially surpassing Wav2Vec2 in multilingual contexts. We will maintain a modular interface to swap this model in if ONNX conversion proves more efficient.[21]

4.2 The Inference Runtime: ONNX Runtime Web

Running PyTorch models directly in the browser is not feasible. We will convert the model to the **Open Neural Network Exchange (ONNX)** format and execute it using **ONNX Runtime Web (ORT-Web)**.

- **Runtime Backend:** ORT-Web utilizes WebAssembly (WASM) with SIMD (Single Instruction, Multiple Data) optimizations to accelerate matrix multiplications on the CPU. It also supports WebGL and the emerging WebGPU standard for hardware acceleration.[5]
- **Framework Compatibility:** While TensorFlow.js is an option, the vast majority of SOTA audio models (including Wav2Vec2) are native to PyTorch/Hugging Face. The pipeline from PyTorch $\rightarrow$ ONNX is robust and well-documented (torch.onnx.export), whereas converting complex Transformers to TensorFlow.js format often leads to operator incompatibility.[22]

4.3 Quantization Strategy

A standard Wav2Vec2 model is approximately 360MB in size (Float32). This is too large for a web application, causing slow load times and high memory consumption.

- **Solution:** We will apply **Dynamic Quantization** to Int8 (8-bit integers).
- **Impact:** This reduces the model size by ~75% (down to ~90MB) and improves inference speed by 2x-3x on CPUs, with a negligible drop in accuracy (< 1%).[23]
- **Implementation:** The ONNX runtime supports quantized operators natively, allowing us to ship a lightweight binary that respects the user's bandwidth and RAM.[24]

4.4 Sliding Window and Hysteresis

Real-time conversation is continuous, but the model requires fixed-length inputs.

- **Window Logic:** We analyze a **3-second sliding window** of audio, advancing every **1 second** (overlap of 2 seconds). This ensures that emotional transitions are captured smoothly.
- **Hysteresis (Smoothing):** Raw model outputs can be jittery (flickering between "Calm" and "Neutral"). We will implement a temporal smoothing filter (Exponential Moving Average) on the output probabilities. A new emotion is only registered as "Dominant" if

it maintains the highest probability for 2 consecutive frames, preventing the UI from flashing erratically.[25]

---

5. User Experience (UX) and Visual Identity System (VIS)

The user interface of AudioEmotion must balance scientific precision with accessibility. It is not just a dashboard; it is an interpretive layer that translates complex biometric data into intuitive visual cues.

5.1 Visual Identity: The Plutchik-Derived Palette

Color is the primary signifier of emotion. To ensure the interface is intuitive, we will adapt Robert Plutchik's "Wheel of Emotions" into a web-accessible color system. Standard primary colors are insufficient; we need specific hues that convey the nuance of the 8 target states.[6]

5.1.1 The Emotion Color Map

The following hex codes have been selected based on color psychology research and contrast accessibility standards:

Emotion	**Hex Code**	**Color Name**	**Psychological Rationale**
Neutral	#9E9E9E	Slate Grey	Represents balance, lack of bias, and the "zero state".[8]
Calm	#5DADE2	Serene Blue	Low arousal, positive valence. Associated with water, sky, and stability.[26]
Happy	#F1C40F	Sunflower	High arousal, positive valence. Universally linked to sun, energy, and joy.[6]
Sad	#2E86C1	Deep Indigo	Low arousal, negative valence. Linked to depth, heaviness, and cold.[6]
Angry	#E74C3C	Crimson	High arousal, negative valence. Signals danger, heat, and blood flow

			(aggression).[8]
Fearful	#8E44AD	Royal Violet	High arousal, negative valence. Historically linked to mystery and the unknown; distinct from the red of anger.[6]
Disgust	#27AE60	Nephritis	Medium arousal, negative valence. Biologically linked to toxicity, sickness, and aversion.[27]
Surprised	#E67E22	Carrot Orange	High arousal, positive/ambivalent. A "shock" color that bridges the energy of red and the joy of yellow.[6]

5.1.2 Typography and Readability

- **Primary Typeface: Inclusive Sans**. Designed specifically for accessibility, this font ensures that text is legible for users with dyslexia and visual impairments. Its open apertures and distinct character shapes (e.g., differentiating 'I', 'l', and '1') are crucial for reading confidence scores accurately.[28]
- **Data Typeface: JetBrains Mono**. For all numerical data (probabilities, timestamps), a monospace font is mandatory. It ensures that tabular data aligns perfectly, preventing the interface from "jittering" as numbers update rapidly.

5.2 Interface Components and Wireframe Strategy

5.2.1 The "Siri-Style" Waveform Visualizer

To confirm the system is active, the central stage of the application will feature a dynamic waveform.

- **Implementation:** We will use a custom Canvas implementation rather than a heavy library.
- **Aesthetic:** Three overlapping sine waves with varying phase, amplitude, and opacity.
- **Behavior:** The amplitude (A) of the waves is modulated by the RMS energy of the input audio. The frequency (f) is modulated by the Spectral Centroid. This creates an organic, "living" visualization that responds instantly to the user's voice, building trust that the system is listening.[29]

5.2.2 The Probability Radar (Spider Chart)

Emotions are rarely singular. A person might be 60% Happy and 40% Surprised. A simple bar chart fails to show this blend.

- **Solution:** A **Radar Chart** (Spider Graph) plotting the 8 emotions on radial axes.
- **Library: react-chartjs-2**. While Recharts is excellent for SVG-based charts, Chart.js (Canvas-based) offers superior performance for real-time animations involving fills and transparency.[17]
- **Optimization:** We will disable the default chart animations (which are too slow) and implement a custom linear interpolation (Lerp) on the dataset values to smooth the transitions between inference frames.[18]

5.2.3 The "Control Tower" Layout

The wireframe layout is designed for single-screen utility [33]:

1. **Header:** Contains microphone selector, privacy indicator (Green padlock icon), and "Export JSON" button.
2. **Main Viewport (Left 66%):** Large Radar Chart overlaid on the Waveform. This is the "Focus Mode."
3. **Data Rail (Right 33%):** A vertical list of "Emotion Cards." The cards re-sort dynamically, with the dominant emotion always at the top. Each card displays the label, the confidence bar, and the mapped color.

---

6. Detailed Feature Specifications

6.1 Feature Set Overview

Feature ID	Name	Description	Technical Dependency
F-01	**Device Selection**	User selects specific audio input (e.g., Bluetooth Headset vs. Internal Mic).	navigator.mediaDevices.enumerateDevices
F-02	**Noise Gate**	System ignores audio below a decibel threshold (e.g., -50dB) to prevent hallucination in silence.	Essentia.RMS or Meyda.energy
F-03	**Live Classification**	Real-time inference of the 8 emotions.	ONNX Runtime Web + Wav2Vec2
F-04	**Dominant Smoothing**	Hysteresis logic to prevent rapid label switching.	Zustand Store Logic
F-05	**Session Recording**	Record the *analysis data* (not audio) for	IndexedDB or In-Memory Array

		export.	
F-06	**Sentiment Trending**	A secondary line chart tracking "Positive vs. Negative" valence over the session.	react-chartjs-2 Line Chart
F-07	**Privacy Shield**	Visual indicator that data is local.	Static Asset / Tooltip

6.2 Deep Dive: The Noise Gate (F-02)

A critical failure mode for SER apps is "Silence Hallucination," where the model forces a prediction (often "Neutral" or "Sad") on background noise.

- **Specification:** The AudioWorklet will compute the RMS (Root Mean Square) amplitude of every frame. If RMS < Threshold, the frame is discarded *before* it reaches the Ring Buffer. The UI will dim the visualizer to indicate "Waiting for Speech."
- **Benefit:** This saves CPU cycles (no inference on silence) and improves the accuracy of the session report.

---

7. Privacy, Compliance, and Security

In the current regulatory climate, "Privacy by Design" is not optional; it is the primary shield against liability. AudioEmotion deals with biometric data (voice), which is protected under the highest tiers of GDPR (Article 9) and CCPA.

7.1 The "Zero-Knowledge" Architecture

Our strongest compliance asset is our architecture.

- **No Server-Side Inference:** Traditional architectures stream audio to an API (like Google Cloud Speech-to-Text). This requires explicit user consent, complex data retention policies, and robust encryption in transit and at rest.[35]
- **AudioEmotion Implementation:** The Python model is compiled to ONNX and delivered as a static asset. Once the web page loads, the internet connection can be severed, and the application will continue to function. The audio data flows from Microphone $\rightarrow$ RAM $\rightarrow$ CPU/GPU $\rightarrow$ Visualization. It never touches a network socket.
- **Privacy Policy Terminology:** The application's privacy policy can uniquely state: *"We do not collect, store, or transmit your voice data. All processing occurs locally on your device."* This drastically simplifies legal requirements.[2]

7.2 Security Headers (COOP/COEP)

To enable the high-performance SharedArrayBuffer required for our Ring Buffer, the application must be served in a "Secure Context" with cross-origin isolation. This protects the user's memory from side-channel attacks (Spectre/Meltdown).[13]

- **Implementation:** The production server (e.g., Nginx, Vercel, Netlify) must return these headers:
  ```
  HTTP
  Cross-Origin-Opener-Policy: same-origin
  Cross-Origin-Embedder-Policy: require-corp
  ```
- **Constraint:** This isolates the app. We cannot easily embed iframes from other domains (e.g., YouTube embeds) unless they are also credentialed, which is acceptable for a focused utility tool.

---

8. Implementation Roadmap

Phase 1: Foundation (Weeks 1-4)

- **Objective:** Establish the high-performance audio loop.
- **Deliverables:**
  - Repo setup (Vite + React + TypeScript).
  - AudioWorkletProcessor implementation using SharedArrayBuffer.[4]
  - Integration of Essentia.js for basic RMS and Spectral Centroid extraction.[15]
  - Basic "Siri-Wave" canvas visualizer driven by live audio.[29]

Phase 2: The Brain (Weeks 5-8)

- **Objective:** Client-side Inference Integration.
- **Deliverables:**
  - Export wav2vec2-large-robust-12-ft-emotion-msp-dim to ONNX with Int8 quantization.[22]
  - Web Worker setup with ONNX Runtime Web.[5]
  - Implementation of the "Sliding Window" logic in the Worker.[25]
  - Benchmark testing: Verify browser inference latency is < 100ms.

Phase 3: The Experience (Weeks 9-12)

- **Objective:** UI/UX and Visualization.
- **Deliverables:**
  - Implementation of the Plutchik Color System in CSS variables.
  - Development of the Radar Chart using react-chartjs-2.[31]
  - "Control Tower" layout implementation with Zustand state management.
  - Accessibility audit (Contrast ratios, ARIA labels).

Phase 4: Polish & Launch (Weeks 13-14)

- **Objective:** Optimization and Compliance.
- **Deliverables:**
  - Bundle size optimization (Lazy loading the 90MB ONNX model).
  - Security Header configuration (COOP/COEP).
  - Drafting of the "Local-Processing" Privacy Policy.
  - Final QA across Chrome, Firefox, and Edge.

---

9. Conclusion

AudioEmotion represents a new breed of web application: the "Heavy Client." By rejecting the traditional dependency on cloud APIs, we unlock a user experience that is instantaneous, private, and robust. The technical complexity of managing SharedArrayBuffers, AudioWorklets, and WASM runtimes is significant, but the payoff is a product that can serve sensitive industries (Healthcare, HR) with a level of trust that cloud-native competitors cannot match.
The combination of **Essentia.js** for DSP, **Wav2Vec2 (ONNX)** for cognition, and **React** for presentation creates a "Local-First" stack that is not only viable but optimal for the future of affective computing. This roadmap provides the definitive path to executing that vision.

Works cited

1. Build an emotion recognition application with Tensorflow.js | Pusher tutorials, accessed on December 24, 2025, https://pusher.com/tutorials/emotion-recognition-tensorflow/
2. 23 Privacy Policy Examples (+ Free Privacy Policy Template) - Enzuzo, accessed on December 24, 2025, https://www.enzuzo.com/blog/best-privacy-policy-examples
3. Background audio processing using AudioWorklet - Web APIs | MDN, accessed on December 24, 2025, https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Using_AudioWorklet
4. Audio worklet design pattern | Blog - Chrome for Developers, accessed on December 24, 2025, https://developer.chrome.com/blog/audio-worklet-design-pattern
5. Run PyTorch models on the edge - ONNX Runtime, accessed on December 24, 2025, https://onnxruntime.ai/blogs/pytorch-on-the-edge
6. Colors & Emotions | Overview, Theories & Connections - Lesson - Study.com, accessed on December 24, 2025, https://study.com/academy/lesson/color-theory-emotions.html
7. Wav2vec2 Large Robust 12 Ft Emotion Msp Dim · Models - Dataloop, accessed on December 24, 2025, https://dataloop.ai/library/model/audeering_wav2vec2-large-robust-12-ft-emotion-msp-dim/
8. Frequency of selection of emotions for each color: study 1. - ResearchGate,

accessed on December 24, 2025, https://www.researchgate.net/figure/Frequency-of-selection-of-emotions-for-each-color-study-1_fig2_331353761
9. audeering/wav2vec2-large-robust-12-ft-emotion-msp-dim - Hugging Face, accessed on December 24, 2025, https://huggingface.co/audeering/wav2vec2-large-robust-12-ft-emotion-msp-dim
10. Live Audio Feature Visualization | Live Audio MFCC (Web Audio API) - GitHub Pages, accessed on December 24, 2025, https://pulakk.github.io/Live-Audio-MFCC/tutorial.html
11. What is the difference between the way Essentia and Librosa generate MFCCs?, accessed on December 24, 2025, https://dev.to/enutrof/what-is-the-difference-between-the-way-essentia-and-librosa-generate-mfccs-13n3
12. AudioWorklet, SharedArrayBuffer, and Worker | Web Audio Samples, accessed on December 24, 2025, https://googlechromelabs.github.io/web-audio-samples/audio-worklet/design-pattern/shared-buffer/
13. WebAudioWorklet in webassembly Shared Array Buffer not defined - Stack Overflow, accessed on December 24, 2025, https://stackoverflow.com/questions/76920200/webaudioworklet-in-webassembly-shared-array-buffer-not-defined
14. ESSENTIA.JS: A JAVASCRIPT LIBRARY FOR MUSIC AND AUDIO ANALYSIS ON THE WEB - ISMIR 2020, accessed on December 24, 2025, https://program.ismir2020.net/static/final_papers/260.pdf
15. Essentia's Benchmarking, accessed on December 24, 2025, https://mtg.github.io/essentia.js-benchmarks/
16. Audio and Music Analysis on the Web using Essentia.js - Repositori UPF, accessed on December 24, 2025, https://repositori.upf.edu/bitstream/handle/10230/49060/correya_tismir_audio.pdf
17. Best React chart libraries (2025 update): Features, performance & use cases, accessed on December 24, 2025, https://blog.logrocket.com/best-react-chart-libraries-2025/
18. How does chart updating works within the context of react? - Stack Overflow, accessed on December 24, 2025, https://stackoverflow.com/questions/72017379/how-does-chart-updating-works-within-the-context-of-react
19. Development of Smart Emotion Recognition System based on Hybrid Deep Learning Models, accessed on December 24, 2025, https://www.ijsat.org/papers/2025/3/7449.pdf
20. speechbrain/emotion-recognition-wav2vec2-IEMOCAP - Hugging Face, accessed on December 24, 2025, https://huggingface.co/speechbrain/emotion-recognition-wav2vec2-IEMOCAP
21. FunAudioLLM/SenseVoice: Multilingual Voice Understanding Model - GitHub, accessed on December 24, 2025, https://github.com/FunAudioLLM/SenseVoice

22. Convert your PyTorch training model to ONNX - Microsoft Learn, accessed on December 24, 2025, https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-convert-model
23. Transformers.js - Hugging Face, accessed on December 24, 2025, https://huggingface.co/docs/transformers.js/en/index
24. Tutorial: Converting a PyTorch Model to ONNX Format | by Deci AI | Medium, accessed on December 24, 2025, https://medium.com/@deciai/tutorial-converting-a-pytorch-model-to-onnx-format-f1bbce156d2a
25. How to implement a sliding window in tensorflow? - Stack Overflow, accessed on December 24, 2025, https://stackoverflow.com/questions/41879440/how-to-implement-a-sliding-window-in-tensorflow
26. Color Palette: Peaceful Palettes - Paper Heart Design Co., accessed on December 24, 2025, https://paperheartdesign.com/blog/color-palette-peaceful-palettes
27. Color Psychology Chart: Guide for Designers - Page Flows, accessed on December 24, 2025, https://pageflows.com/resources/color-psychology-chart/
28. Inclusive Sans - Google Fonts, accessed on December 24, 2025, https://fonts.google.com/specimen/Inclusive+Sans
29. Siri-style audio visualizer - GitHub Gist, accessed on December 24, 2025, https://gist.github.com/jango-blockchained/24a4a0b8ac7ddf49843bee403d02a3d6
30. How to make an Audio Visualizer with React Native Audio API and Skia - YouTube, accessed on December 24, 2025, https://www.youtube.com/watch?v=gmW8KeMKXok
31. Radar Chart - react-chartjs-2, accessed on December 24, 2025, https://react-chartjs-2.js.org/examples/radar-chart/
32. Performance | Chart.js, accessed on December 24, 2025, https://www.chartjs.org/docs/latest/general/performance.html
33. Free Online Wireframe Tool - Canva, accessed on December 24, 2025, https://www.canva.com/online-whiteboard/wireframes/
34. Voice recording ui Images - Free Download on Freepik, accessed on December 24, 2025, https://www.freepik.com/free-photos-vectors/voice-recording-ui
35. Protecting Personal Information: A Guide for Business | Federal Trade Commission, accessed on December 24, 2025, https://www.ftc.gov/business-guidance/resources/protecting-personal-information-guide-business-0
36. Sample Privacy Policy Template for Website (with Examples) - Termly, accessed on December 24, 2025, https://termly.io/resources/templates/privacy-policy-template/