

Audio Ninja

COMP 523 - Team E

By Ethan Ahdout, Jacob Brown, Jake Terrill and Ryder Klein

Client: John McHugh



Agenda

Project Overview

[01]

Live Demo

[02]

Platform and Tools

[03]

Final Architecture

[04]

Codebase Review

[05]

Reflection

[06]

What is Audio Ninja?

- Desktop application for marking and exporting precise audio segments
- Users can load media files, place timestamp markers, name segments and export results
- Keyboard-centric design with customizable shortcuts for all commands and features
- Built for users who work with large audio files who need accurate, repeatable media annotation



Features



Media Markup

- Place and delete markers using keyboard or UX
- Name and fine-tune segments



Media Interaction

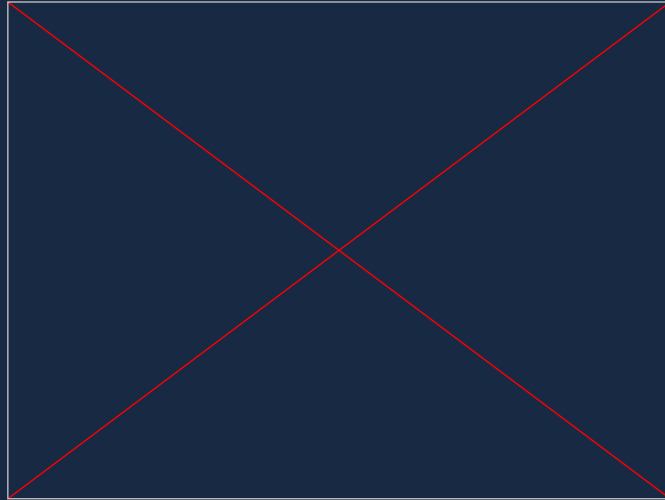
- Load media files of various formats
- Render an interactive waveform
- Playback controls



Export/Segmentation

- Export segments directly from Audio Ninja (same format as original file)
- Export CSV with segments

DEMO



<https://github.com/Comp523-Team-E/audio-ninja>

Platform and Tools

Platform: Desktop app compatible with **MacOS** (Intel and Silicon), **Windows** and **Linux** operating systems

Frontend: SvelteKit, Svelte5, **Typescript**; **Wavesurfer.js** for audio visualization

Backend: Rust, communicates with frontend via **IPC**

Testing

- **Vitest** for frontend
- **Cargo** for backend

Audio Processing/Export

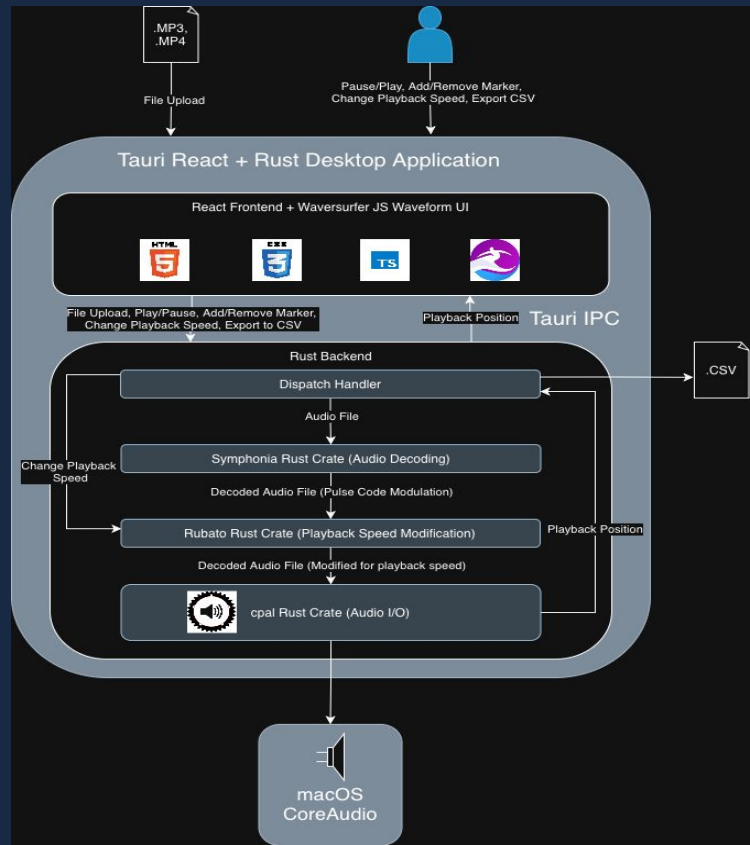
- **FFmpeg** (bundled as a Rust sidecar)

Development Tools

- IDE: **VSCode**
- SCM: **Git/GitHub**
- CI/CD: **GitHub Actions**

Final System Architecture

- Local-first desktop app
 - No external server
 - Media files stay on users machine
 - Lightweight
- Svelte Frontend
 - Renders the UI, waveform, marker panels, shortcuts and export controls
 - More lightweight than React
- Tauri IPC Bridge
 - Frontend calls Rust commands via `invoke[...]`
- Rust backend services
 - Handles playback, marker validation, CSV/FFmpeg export logic



Reflection

Overall, we are very happy with how our project turned out, and so is our client. We've all gained valuable experience in managing the full development lifecycle, from inception, to specifications, to iterative development and testing and deployment.



What went well:

- All functional requirements met
- Lightweight, cross-platform distributable desktop application
- All deadlines met
- Development tools and coding agents properly utilized

What was challenging:

- Features such as zoom were not as straightforward as expected to implement
- Code signing/notarization for macOS had extra steps we were unfamiliar with

What we learned:

- App development using Tauri shell
- Managing project state and CI/CD via GitHub actions
- Code signing and notarization for macOS

Thank you!

