EchoGuard

Project Lead: Abner Rodriguez

Project Members: Kevin Sanchez, Carlos Navarro, Kenneth Martin, Alfonso Contreras

Product Manager: Professor Christian Poellabauer

FIU
FLORIDA INTERNATIONAL UNIVERSITY

COLLEGE OF ENGINEERING AND COMPUTING

What is EchoGuard

- EchoGuard is an AI-based web application that uses a CNN machine learning model to help parents keep an eye on the type of media their kids are consuming.

- EchoGuard takes an audio input: this could be a clip from a TV show, a YouTube video, and it analyzes it. After that analysis, it generates both a risk score and a benefit score for that audio clip as well as a confidence score for its evaluations.

- How does EchoGuard decide those scores? It looks at two key aspects of the audio input:

1. First, it analyzes the human speech in that clip (transcript).

2. Second, it checks for any violent environmental sounds that might be present, like gunshots, shouting, or explosions.

FIU Engineering & Computing

Motivation For EchoGuard

- The motivation for building EchoGuard came from a problem that's becoming more and more common today.
- Kids are constantly exposed to digital media, and the audio can be unpredictable and inappropriate for their age. And parents understandably are busy. They can't manually monitor everything their kids watch or listen to in real time.

- That's where EchoGuard steps in. It fills that gap by analyzing the audio for them and giving clear feedback about what their children may be consuming.

FIU Engineering & Computing

System Overview

1. **Frontend**
   - Built with Next.js and styled with Tailwind CSS.
   - Allows users to record or upload audio clips directly from the browser.
   - Displays waveform visualizations, playback controls, and the final analysis results (risk breakdown, confidence score, etc.).
1. **Backend (Flask Server + Supabase)**
   - Developed in Python Flask, which receives audio clips from the front-end.
   - Handles audio preprocessing (cleaning, normalizing, and splitting audio).
   - Sends processed data to the CNN model for sound classification.
   - Integrates with Supabase for user authentication and data storage.
2. **Machine Learning Components**
   - CNN Model (PyTorch): Detects non-speech environmental sounds like gunshots, glass breaking, screaming, etc.
   - Gemini Wrapper (Contextual AI): Interprets spoken dialogue and contextual cues from the CNN to produce a risk/benefit analysis of what is being played.

FIU Engineering & Computing

System Workflow (Explained)

1. **User Interaction**
   - User presses “Listen Now” or uploads a clip.
   - The app records the surrounding audio (e.g., a TV show, YouTube video, or TikTok).
2. **Audio Capture and Voice Activity Detection (VAD)**
   - Using the RNNoise VAD model, the backend separates:
   - Speech segments (dialogue, conversations)
   - Non-speech segments (sound effects, background noises)
3. **Storing Segments**
   - Using an in-memory SessionManager, non-speech segments, speech segments, as well as raw WAV files are stored in-memory.
4. **Routing raw Wav files + Non-speech Segments**
   - Raw WAV files → Transcription Tool (Faster-Whisper)
   - Non-speech segments → Validation endpoint
5. **Non-speech Validation before reaching CNN Model**
   - Validation is performed on non-speech segments to ensure that silence is not fed into our CNN Model.
   - This is necessary because feeding our model ‘junk’ input would disrupt its analysis and cause it to hallucinate.

FIU Engineering & Computing

System Workflow (Explained)

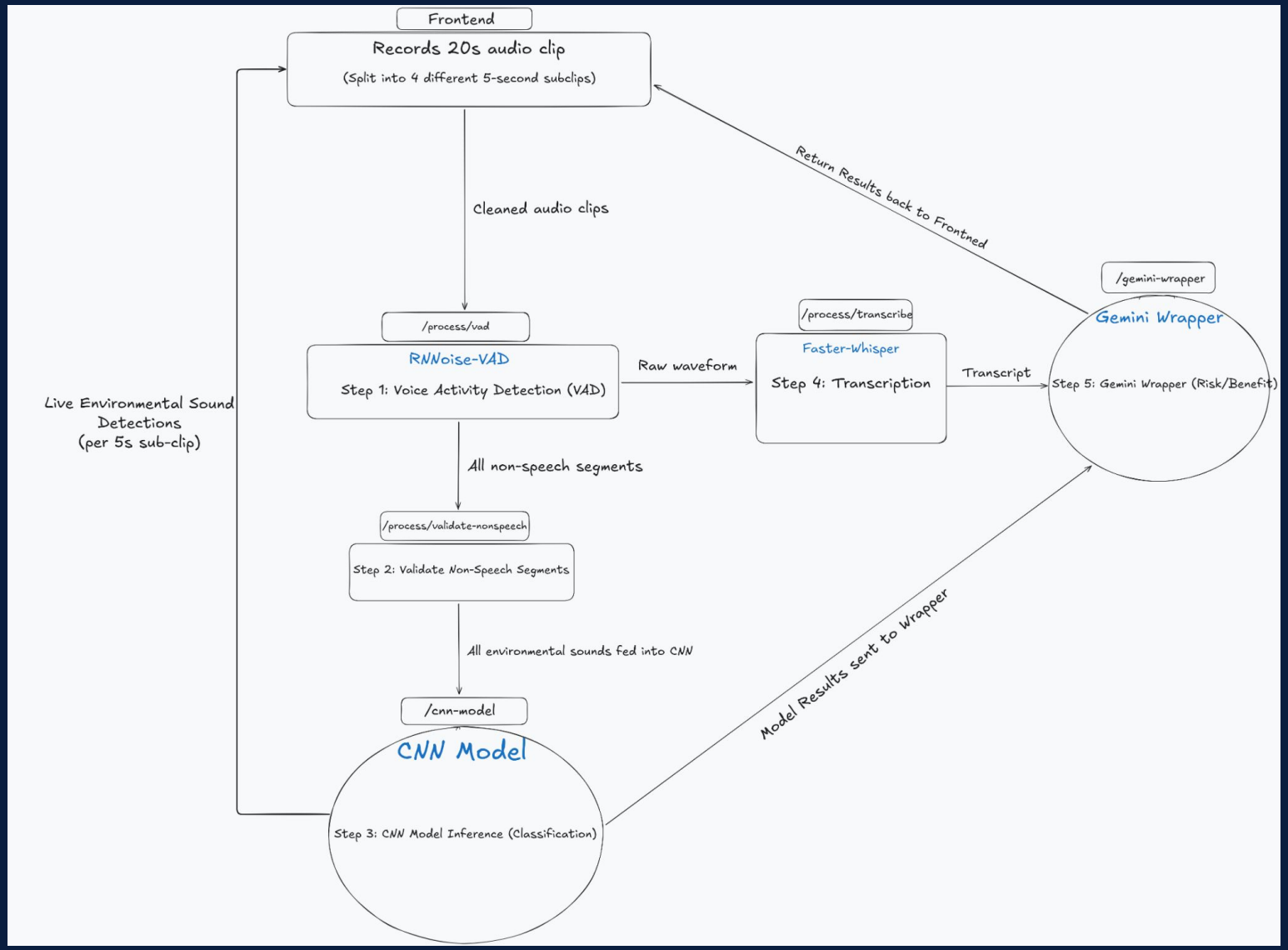
1. **Transcription**
   - Once, the user finishes their recording, all stored raw WAV files are sent to the transcription endpoint.
   - In this endpoint, all files are resampled to 16kHz for input into the transcription tool (Faster-Whisper).
   - Transcript for that recording is generated.

2. **Sending Results to Gemini Wrapper (Gemini 2.5 Pro Model)**
   - Once both our CNN model finishes its analysis, as well as the transcript has been generated these results are sent to our Gemini Wrapper.
   - Once Gemini has received both inputs it generates a benefit, risk, and confidence score as well as explanations for each.

3. **Results → Frontend → Displayed in Dashboard**
   - Once Gemini has finished its analysis, it sends its results to the frontend, and these results are displayed in a dashboard for the user.

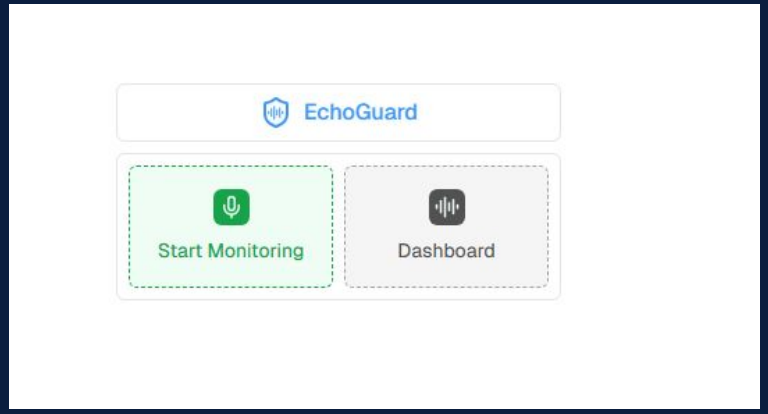
FIU Engineering & Computing

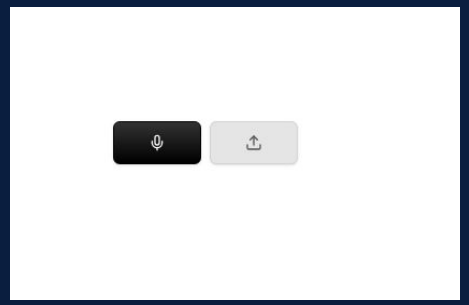
System Workflow (Diagram)

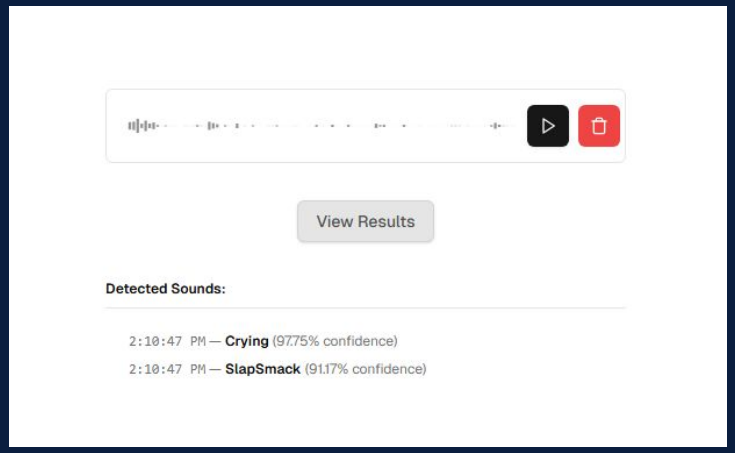


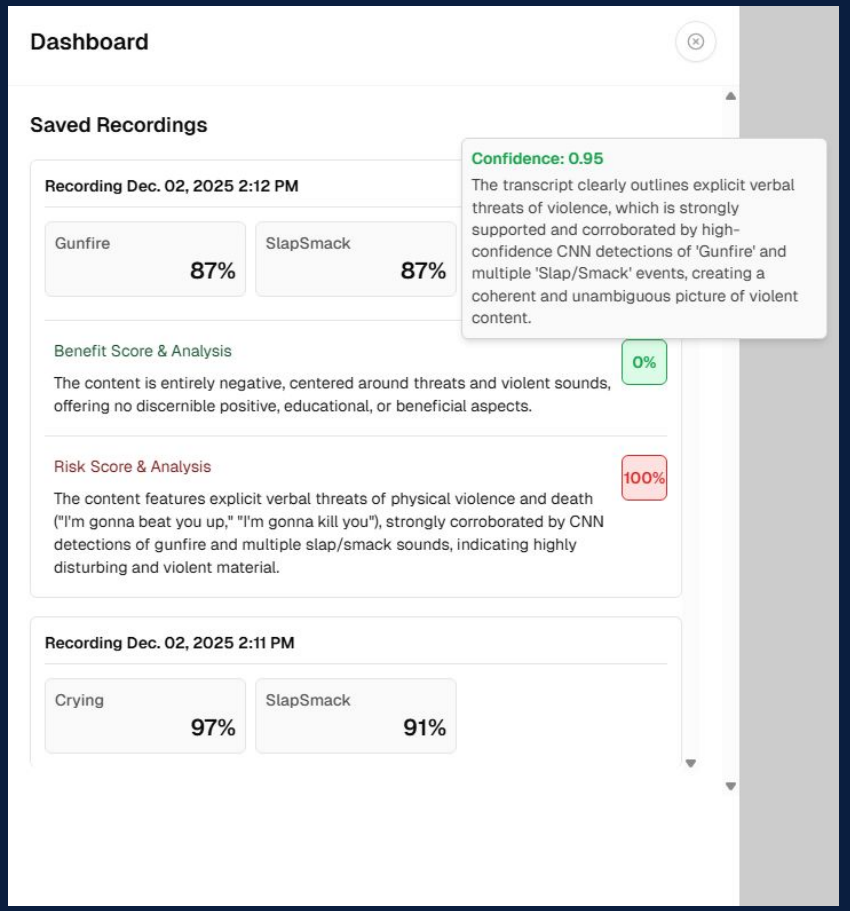
FIU Engineering & Computing

UI/UX









FIU Engineering & Computing

Multi-Label CNN Model

- CNN Model's role is to receive the recorded audio and provide a confidence score from 0 to 1 for each trained category.

- 14 trained categories:
- Crying, Car Alarm, Chainsaw, Explosion, Glass Shattering, Fire, Emergency Siren (Ambulance, fire truck, and police siren), Artillery Fire, Battle Cry, Gunfire (regular and machine gunshots), Slap/Smack, Thunderstorm, Growling, Roar, Whimper.
- If a category receives a confidence score of 0.7 or higher, it will send that finding to the Gemini wrapper.

FIU Engineering & Computing

Future Improvements

1. Add full backend database to persist user sessions and results.
2. Implement source separation/noise suppression to isolate target audio and improve classification accuracy.
3. Add secondary “event recovery” pipeline to analyze speech-labeled segments for hidden environmental sounds.
4. Improve front end audio capture pipeline to reduce browser noise suppression artifacts.
5. Improve and expand dataset by either creating a specific dataset or manually curating one.
6. Expand CNN training to more categories using these datasets.
7. Implement a more polished UI: full session history, clearer visualizations

FIU Engineering & Computing