Kameswara Naga Sai Siva Dheeraj Mantha

Research Journal Assignment: Integrating AI Tools with HTML, CSS, and JavaScript for Responsive Web Development

Research Question

How effectively can a **sound-based AI model** trained with **Teachable Machine** classify audio inputs like different button and click sounds, and reflect those predictions in real-time on a responsive website using basic **HTML**, **CSS**, and **JavaScript?**

Hypothesis

I recently worked with Google's Teachable Machine's hand-tracking project model and implemented it in Unity Game Engine for Interactivity. But, this time I wanted to test the audio project to test sound sound-based model. I believe, when the model is trained properly and integrated into a responsive webpage, it can accurately detect and differentiate between certain sounds, background noises, and use that detection to control the website's appearance through live UI feedback. I wanted to see how far I could push this with no previous coding knowledge.

AI Tools Researched and Used

Teachable Machine by Google

Teachable Machine is a no-code model training tool that lets anyone create machine learning models quickly. It supports image, pose, and audio classification, and gives an easy way to export the trained model to TensorFlow.js. I used it for its simplicity and for how fast I could build and test sound classification models without having to dive deep into backend ML.

I chose to experiment on Teachable Machine because I had a previous experience working with it for one of my interactive projects. Since I had no coding experience, especially in **HTML**, **CSS,** and **JavaScript**, I had to stick to **ChatGPT** as well to create the website and **integrate the Teachable Machine's Tensorflow.js into the script.**

Replit

For designing, testing, and running the webpage, I used **Replit**, which is an online IDE and hosting platform that supports live previews for frontend projects. I **used** the **HTML, CSS, JS template** so I didn't have to worry about setting up a local server. This saved me a lot of time and made the whole testing process much faster. Also, for some reason, **Replit's** live preview feature wasn't working as expected in a few scenarios, so I had to download the **HTML** file and run it locally.

Features & Capabilities

Teachable Machine

No-Code Training: I didn't need to write a single line of code to train the model. I just recorded audio samples directly through the browser and labeled each class.
Audio Classification Support: I was able to create a model specifically for sound-based detection. It supported real-time mic input and gave consistent prediction updates.
TensorFlow.js Export: It gave me a direct way to plug the trained model into my JavaScript code using just a hosted URL. This helped me skip a lot of complex setup.

Quick Testing & Deployment: I could train and test new audio models in under 10 minutes. This helped me iterate faster and fine-tune my training samples.

TensorFlow.js

Runs in the Browser: I didn't need any backend or server. The whole model runs directly in the browser, which helped keep the project light and fast.
Speech Commands Library: It provided a clean way to connect the audio model to real-time sound detection using recognizer.listen().
Custom Thresholds: I could set how confident the model needed to be before updating the UI. This helped me reduce false positives.
Highly Responsive: The predictions are updated in milliseconds, so it was easy to build a real-time, animated interface that reacts instantly to different sounds.

Replit

Browser-Based IDE: I didn't need to install anything. Everything runs in the browser, and the interface is clean and fast.
Live Preview: Every time I saved changes, the preview updated instantly. This made testing the AI interaction feel smooth and real-time.
HTML/CSS/JS Friendly: I could use plain HTML, CSS, and JavaScript, no framework required. This kept my setup simple and clean.
No Server Setup Needed: Just hitting "Run" gave me a working webpage with mic permissions, which made testing the AI model a lot easier.

Relevance

All three tools, **Teachable Machine**, **TensorFlow.js**, and **Replit**, were directly tied to my main research goal. I wanted to create a live, responsive webpage that reacts to different real-world sounds using AI. I also didn't want to spend too much time setting up servers or complicated backend logic.

- **Teachable Machine** helped me train the actual AI model quickly and visually, without needing ML experience.
- **TensorFlow.js** allowed that model to work directly in my website, using only frontend code. This matched my goal of real-time sound-based interaction in the browser.
- **Replit** made it easy to design and test the entire UI, with animations, transitions, and mic input, in one place, with no need for deployment or hosting.

Together, they helped me take an idea from training to testing to final design, all within a browser-based workflow.

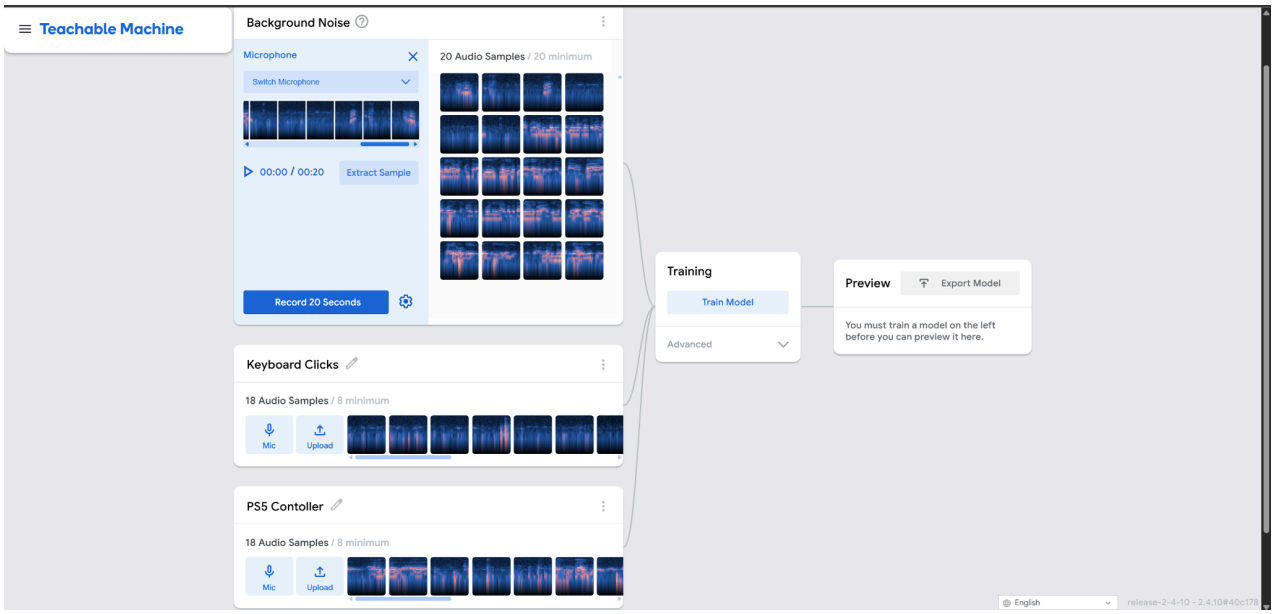
Implementation and Testing

Tools & Setup

Audio Model Setup: I trained the Teachable Machine with a sound classifier **with 3 classes**:

- **Background Noise**
- **Keyboard Typing**
- **PS5 Controller Button Presses**

The export type used was TensorFlow.js, hosted, which provides a link to a .json model I could plug directly into my frontend.

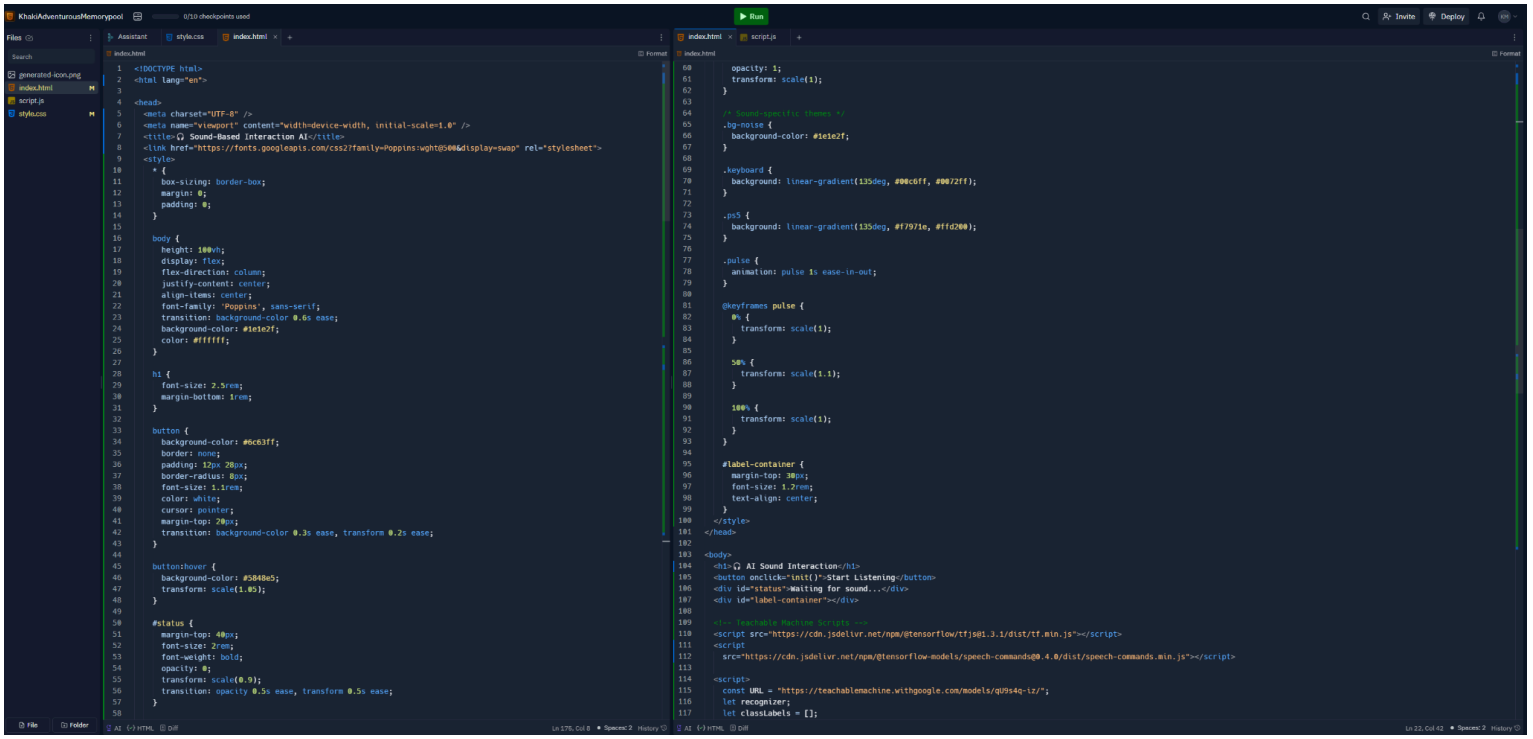


Prompting & Design Logic

I trained the model using my mic, with all the noise cancellation features turned off, and recorded the three sounds. I gave the model **18 samples**, **recorded 4 - 5 times**(3 seconds each) of audio per class, and made sure to vary the distance and angle a bit for better accuracy. Once an audio sample is recorded, it generates 5-6 samples **per record**.

After training, I uploaded the Tensorflow.js to the cloud and saved the link to the clipboard. **After exporting**, I had to rely on ChatGPT to get the **JavaScript and other codes to setup the website**. I wanted a basic website design that can access the user's microphone, and display the text of detected sound while also changing the background colour. I prompted **ChatGPT** multiple times, and it gave me the design in a single code.

Replit Workflow: A big 130 lines of code have been implemented in the index.html file, and the JavaScript, style.css are not touched.



Experience:

Google Teachable Machine

The initial setup was smooth. It was easy to set up the classes and set the desired microphone before starting to record. But the real problem was the unexpected bugs. Sometimes, when I try to stop the recording and export the sample, Teachable Machine stops the recording, but still records the audio infinitely, restricting the user from starting a new recording again. It took me a few attempts before I successfully captured all three classes**(background noises, keyboard buttons, and PS5 controller clicks).**

Experience with Replit and ChatGPT

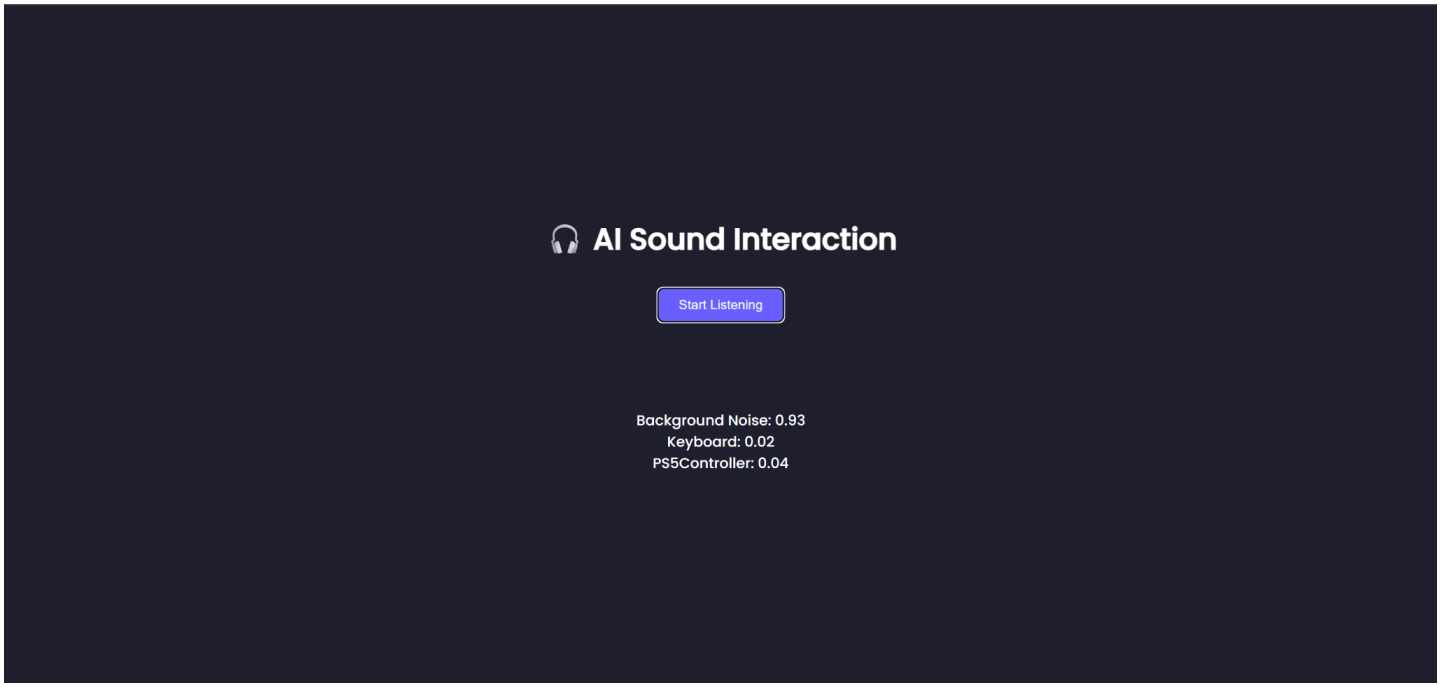
Since I have zero coding knowledge, I followed **ChatGPT** for the coding part, and realized that ChatGPT generated a different JavaScript code. I ended up pasting it in my Replit workflow, and the website didn't work(of course, it won't).

Later, I had to copy the Tensorflow.js and prompt **ChatGPT** to generate code based on that. I got the working code into index.html. Now, the **"Preview"** button was not working. I thought something went wrong with my code, and was trying to debug it with ChatGPT's help. Apparently, I found out that the preview button is just not working, and I decided to run the website locally.

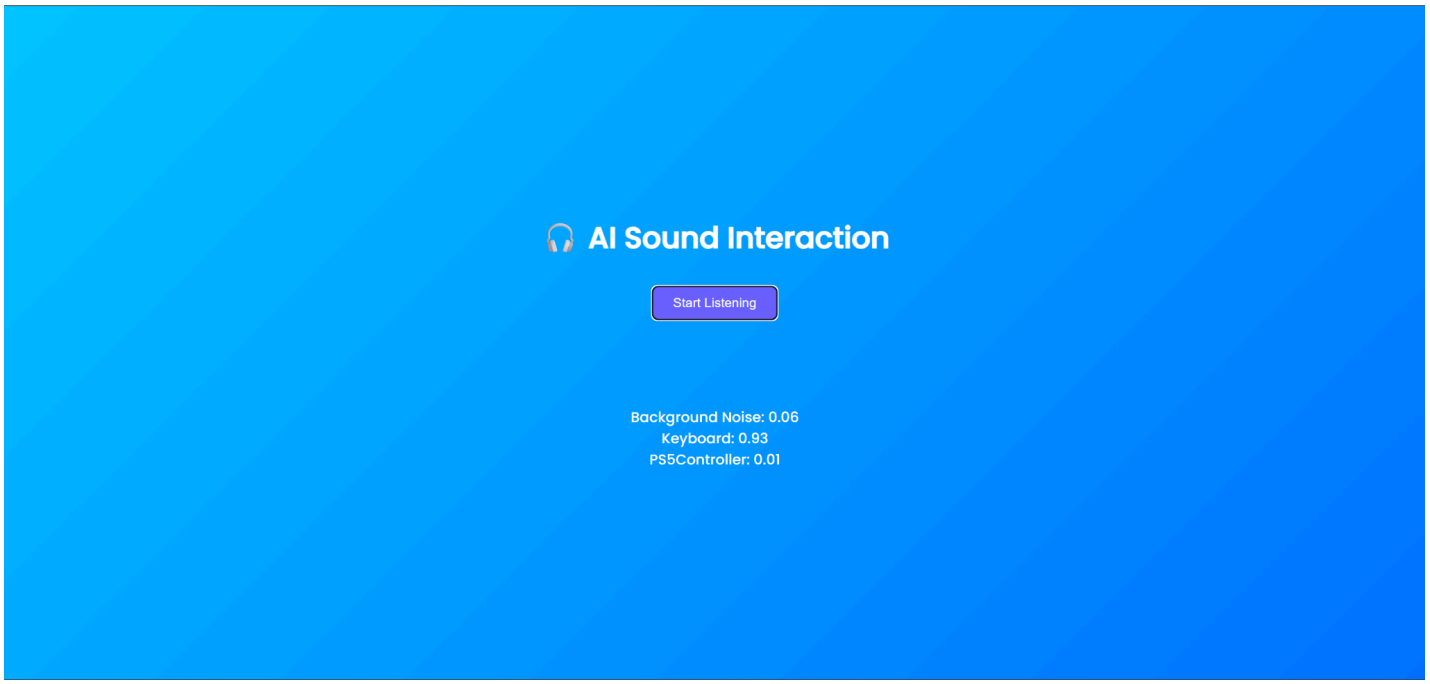
I downloaded the HTML file and ran it.

Experience with the Website

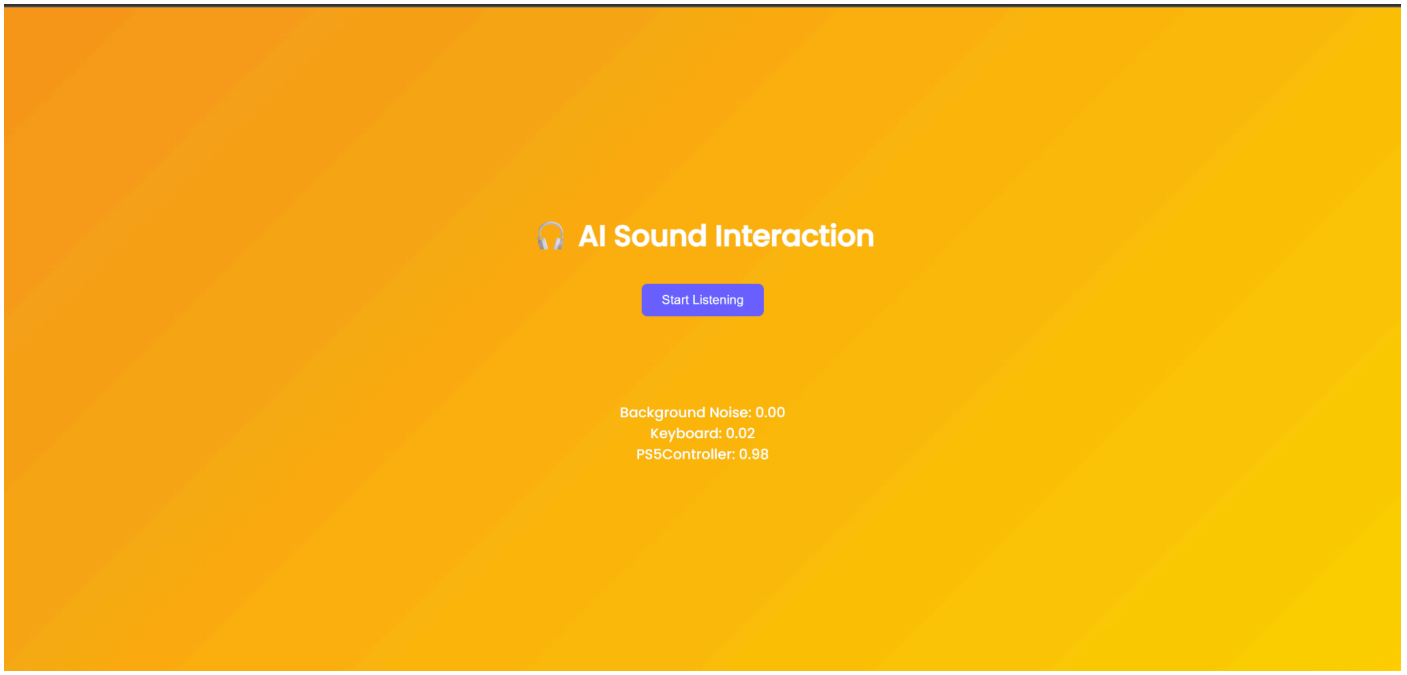
Background Noise: Sounds and voices other than keyboards and controller sounds.



Keyboard Typing: Blue theme with a pulsing animation.



PS5 Controller Press: Yellow-orange theme with a shake animation.



I was not able to upload the recorded video, and I will attach the video along with the journal document

Upon setting up the **correct microphone device in Google Chrome**, and successfully able to access the website. As soon as I clicked on **"Start Listening",** Google Chrome asked me for **microphone access**. I approved, and the prompt kept appearing again and again. That was one more frustrating bug I encountered. **Upon restarting Google Chrome,** I was able to get the website working.

The pulse and shake animations were not working as expected. The transitions were not smooth. Though Teachable Machine sometimes made minor errors in recognizing the correct sound, **the overall behaviour of the website and Teachable Machine met my expectations.**

Evaluation

Even with a few setup hurdles, **Teachable Machine** and **Replit** worked pretty well. The AI did a decent job of telling the difference between different sounds like **background chatte**r, **typing**, and **PS5** button presses. It was not perfect, but for an easy-to-use tool for beginners, it was accurate enough. Interacting with sound felt smooth, and seeing the website's colour change made it feel like the AI was listening. The shaking and pulsing did not work as expected, but the color changes responded okay. Overall, the website was interactive enough for a simple AI-powered page.

Performance Against Hypothesis

I expected the AI model to differentiate between different **non-verbal sound inputs** and help trigger visual changes on the webpage. **This worked as planned**. Even though I had no experience in **HTML**, **CSS**, or **JavaScript**, **the help from ChatGPT** and the easy export from Teachable Machine helped me reach my goal. **The AI was not always 100% accurate,** but for a simple web project that listens and responds to real-time sound inputs, it met the expectations I had when I started. The UI reaction felt good, and I was happy to see a working output after all the issues I faced during setup.

Challenges

There were multiple bugs I had to fix. **First**, while recording audio samples in **Teachable Machine**, the **tool stopped working** a few times, especially when trying to **stop a recording**. I had to refresh the page several times before it allowed me to restart. Then, when I moved to Replit, the code generated by **ChatGPT** didn't work the first few times. I pasted it blindly, and nothing showed up in the preview. I thought there was something wrong with the code, but it turned out the preview button in **Replit** was not working properly(**atleast for me**). Even after running the website locally, I faced another issue. **Chrome** kept asking for microphone permission again and again.

Resolution

I had to try different things to solve each problem. For the **Teachable Machine** issue, I simply refreshed and **re-recorded** the samples multiple times until it worked. For the **broken code** issue in **Replit**, I went back to ChatGPT and asked it to generate the **JavaScript** again, and this time, I made sure I copied the TensorFlow.js code from the **Teachable Machine** website into **ChatGPT** and let it generate the **correct code** before I took it to Replit's **index.html** file. And for the microphone bug in **Chrome**, I restarted the browser and selected the right mic input in the site settings. After these steps, I was finally able to get the entire project working.

Insights

From this project, I learned that sometimes, simple AI tools can be **very powerful** when used creatively. I also saw how important **UI** design is when it comes to **user feedback**. Without **animations** and **text**, the overall experience would've been less responsive, even if the AI was detecting the right sound. Having no prior knowledge of coding, I now feel more confident in working with AI models in basic web projects. I also learned that **small changes** like setting or changing animation timing can make a big difference in the final output.

Real-World Application

This idea can be useful in multiple scenarios. For example, it could be used in accessibility focused websites where touch or mouse control is not possible. It could also be used in interactive

installations, gaming websites, or smart interfaces where simple sound inputs can trigger events. I can see this being used in an **interactive portfolio**, or even in **classrooms** where background **noise detection** can be used to control certain UI actions automatically.

Future Improvement

In the future, I plan to try adding more sound classes like **claps**, **whistles**, or **even voice commands**. I also want to improve the animations and make them smoother. Right now, the pulse and shake effects do not feel that dynamic, so I would want to try generating better code from AI models. I may also try Replit's AI generator and see how it works. Another idea is to save the prediction results in a log and show how the user interacted over time. **But for now**, I am satisfied with what **I was able to achieve with this project.**