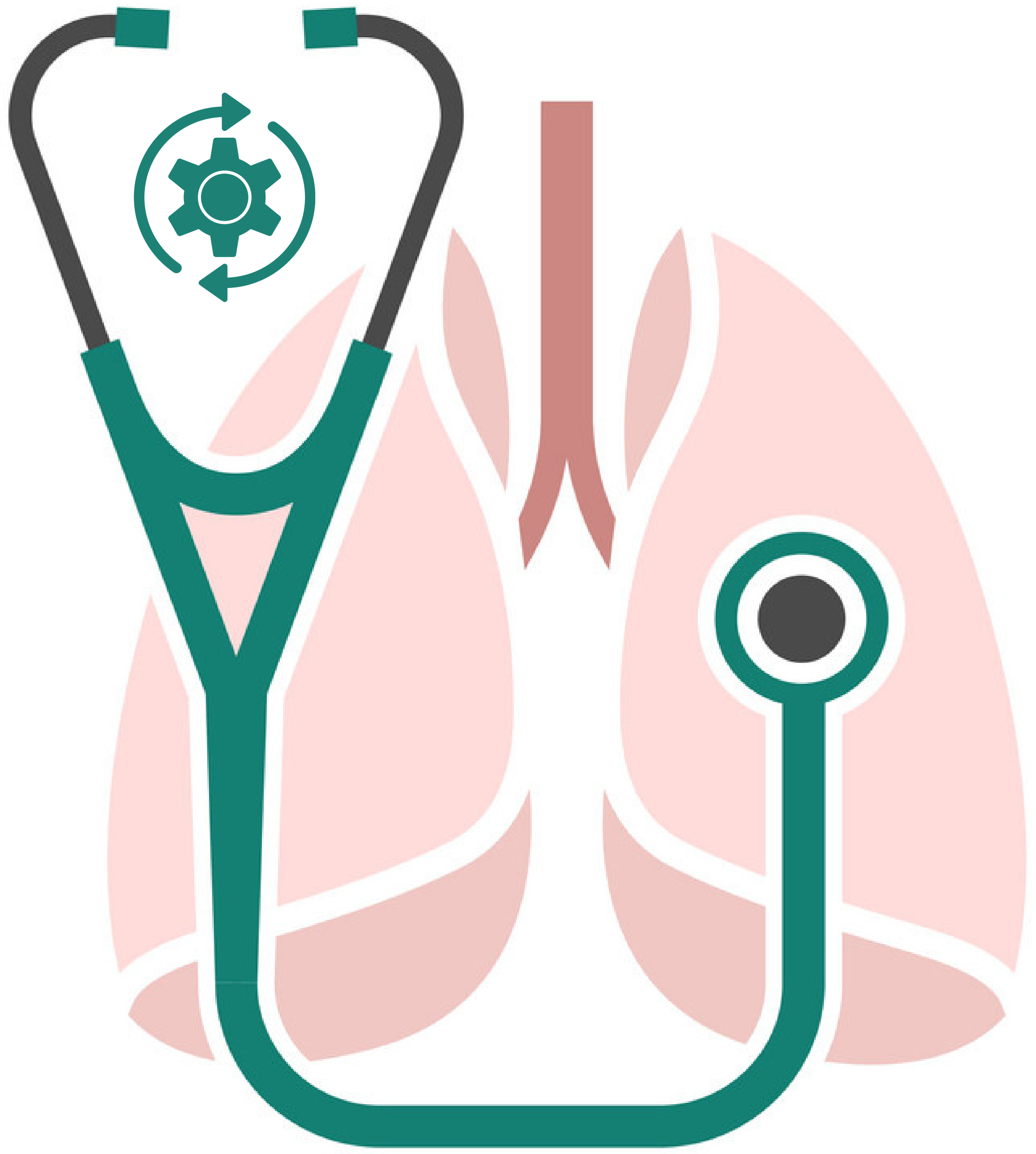


LUNG SOUND RECORDER

The Quad Chips

Team 38

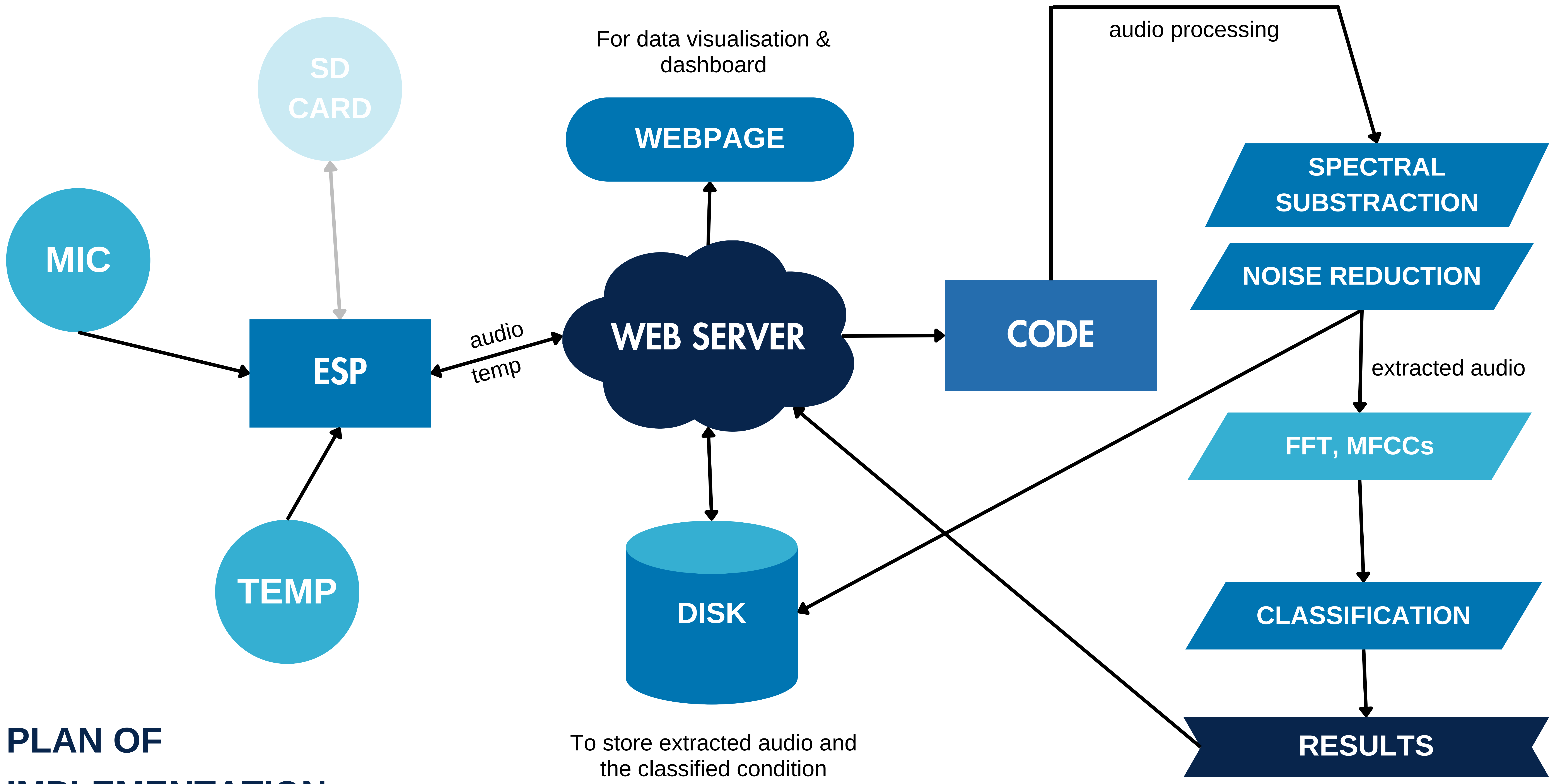
Sai Praneeth - 2022101097

Jahnavi - 2022101118

Vaishnavi Priya - 2022101108

Sai Divya - 2022101090

Associated Professor - Abhishek Srivatsava
Associated TA - Santhoshini Thota



PLAN OF IMPLEMENTATION

CHANGES IN IMPLEMENTATION

1 Removed the SD card module from the picture, since it is not working, instead we have implemented the file transfer via http to the Flask server without the involvement of intermediate storage. Also implemented sockets for seamless interactions in UI.

2 Tried new Noise reduction techniques for audio processing like bandpass filtering,moving average, weiner.

3 Spectral Subtraction. As the audio was transferred byte-by-byte to the server. There were these constant beats that occured when writing it to the audio file. This was solved by using the Spectral Subtraction where we input the audio file and beats file (which contain the constant beats). The program subtracts the beats from the audio file and refines the output file.

Switched from Pattern Matching to training ML models.

Previously, we have used Pattern Matching (with a sample audio of the lung sound), which has narrower range of variations. As this technique is less efficient, we have started to go in the path of Machine Learning.

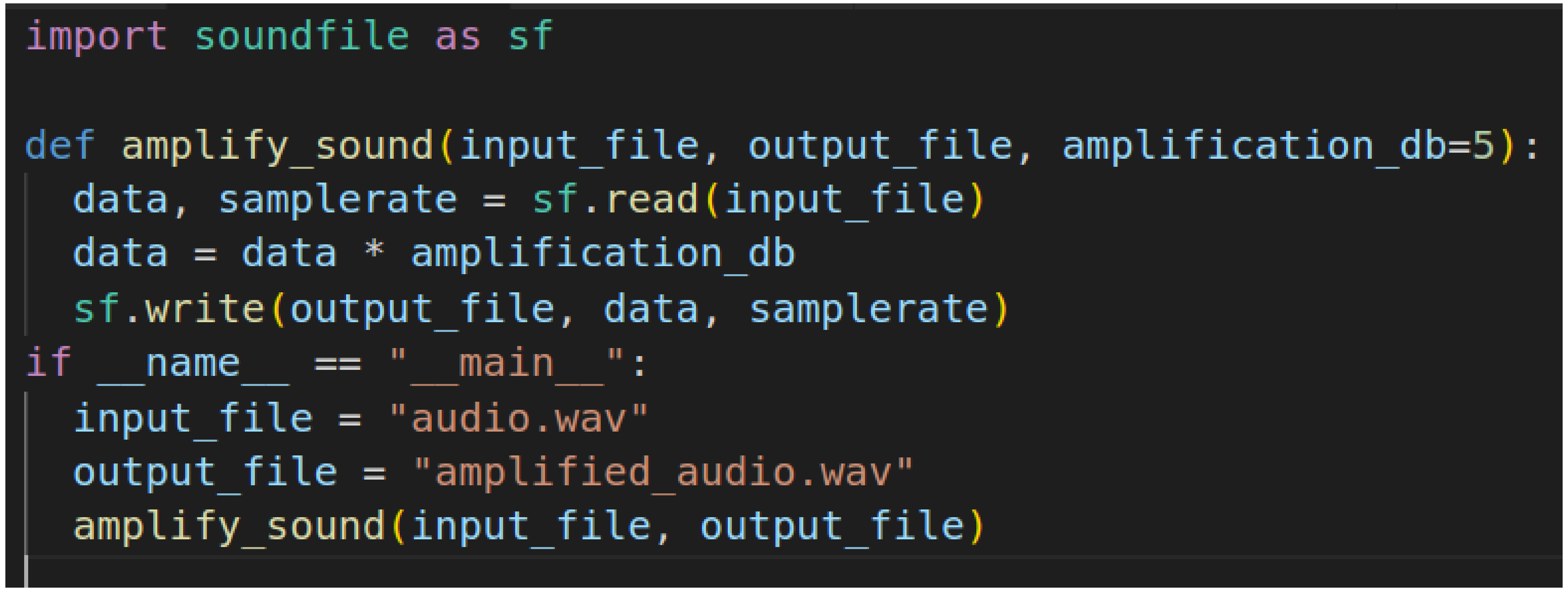
For the classification, we have trained SVM - a supervised machine learning algorithm. We have trained the model on a smaller dataset - which we have found here.

For training the model, we have divided the audio dataset into three classes, wheeze, crackle, and normal. We extract the features or (MFCCs) from the audio files, then, labeling the extracted features.

Then, Splitting the data into train & test.
We train the classifier on the train data and perform prediction by the model.

We get the accuracy score while testing the dataset.

Amplifying the audio

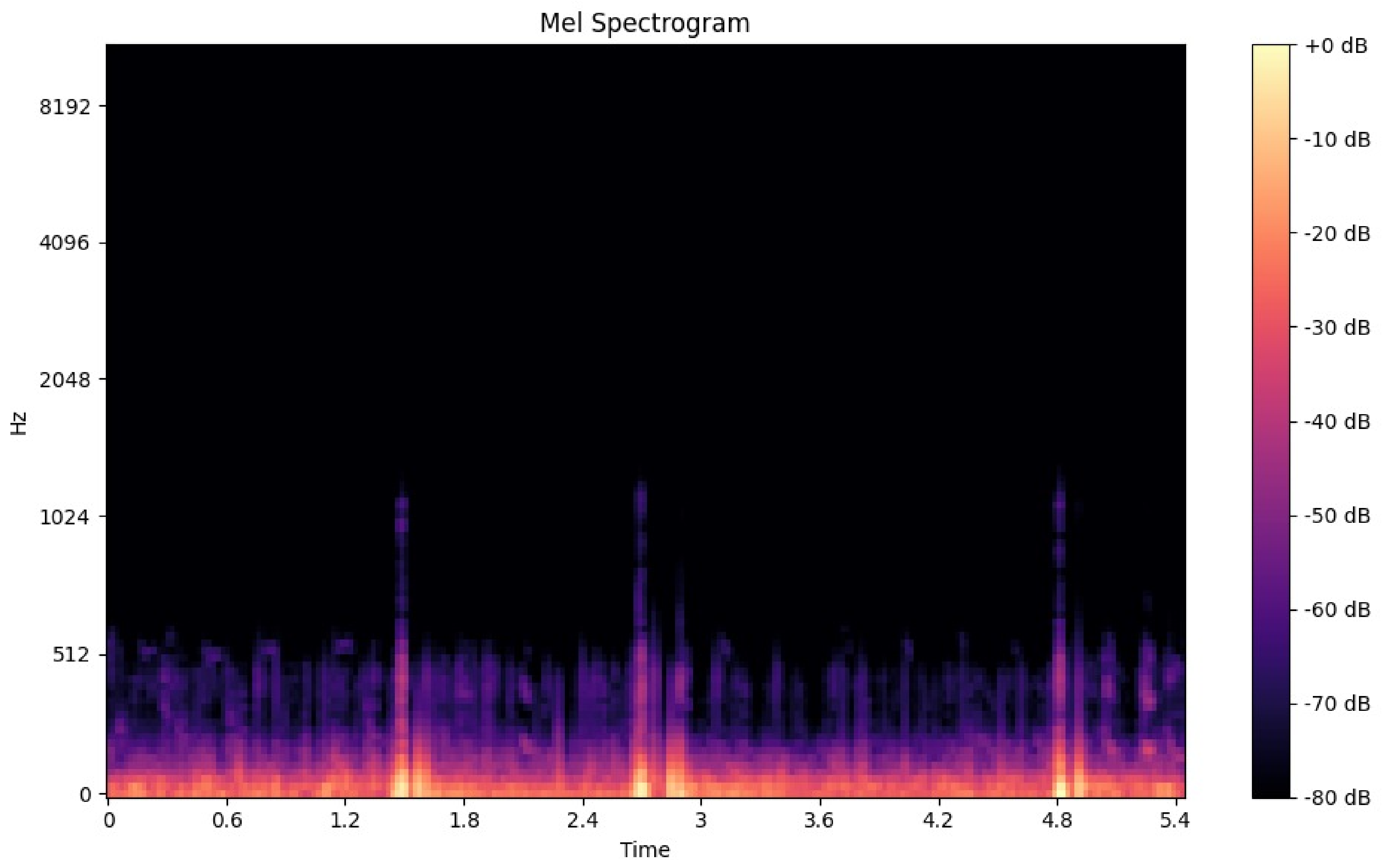

```
import soundfile as sf

def amplify_sound(input_file, output_file, amplification_db=5):
    data, samplerate = sf.read(input_file)
    data = data * amplification_db
    sf.write(output_file, data, samplerate)
if __name__ == "__main__":
    input_file = "audio.wav"
    output_file = "amplified_audio.wav"
    amplify_sound(input_file, output_file)
```

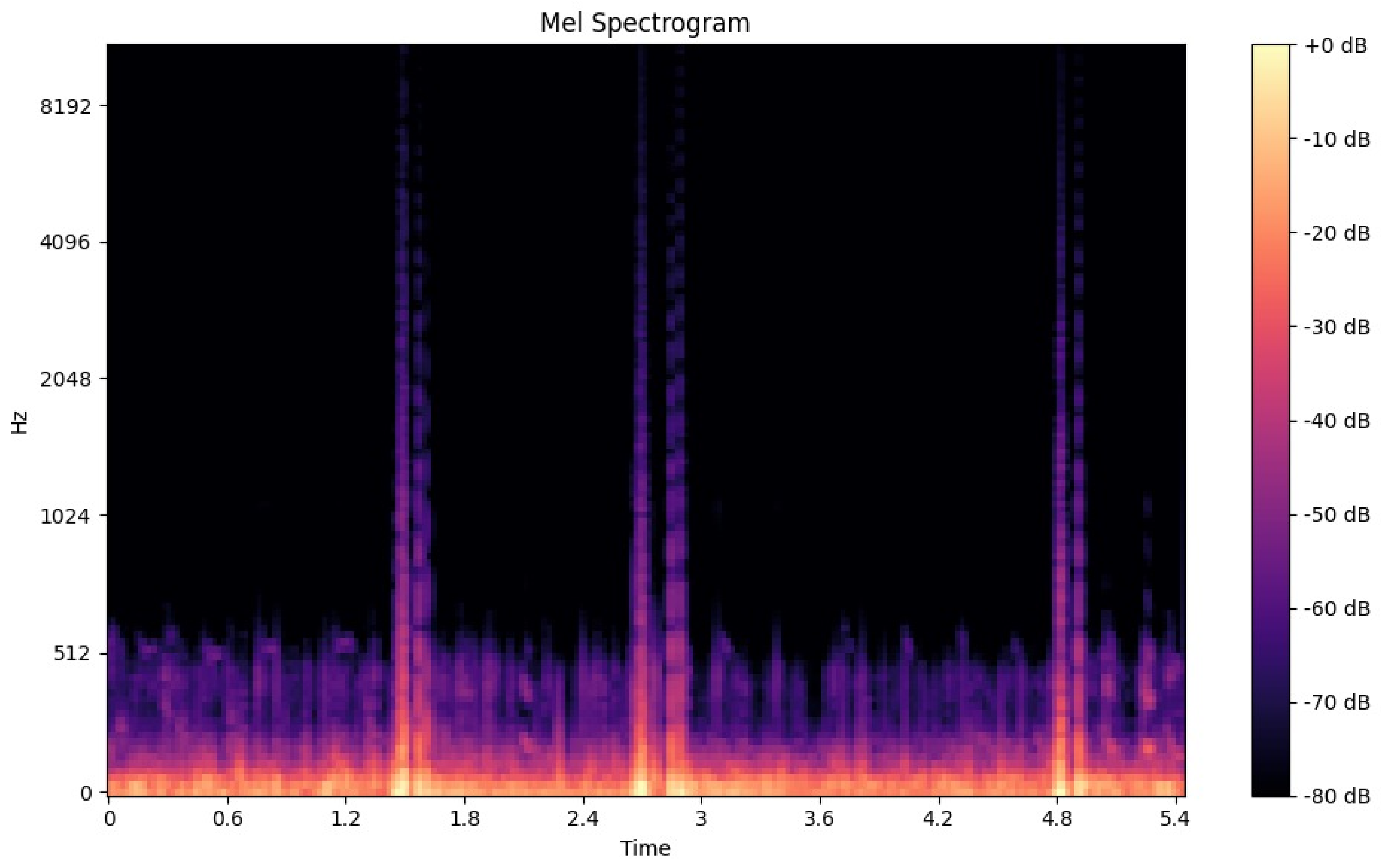
Since the recorded audio is low it is amplified for better feature extraction.

source:https://g.co/bard/share/1bd18dfcf4a1

Mel Spectrograms



Before amplification



After amplification