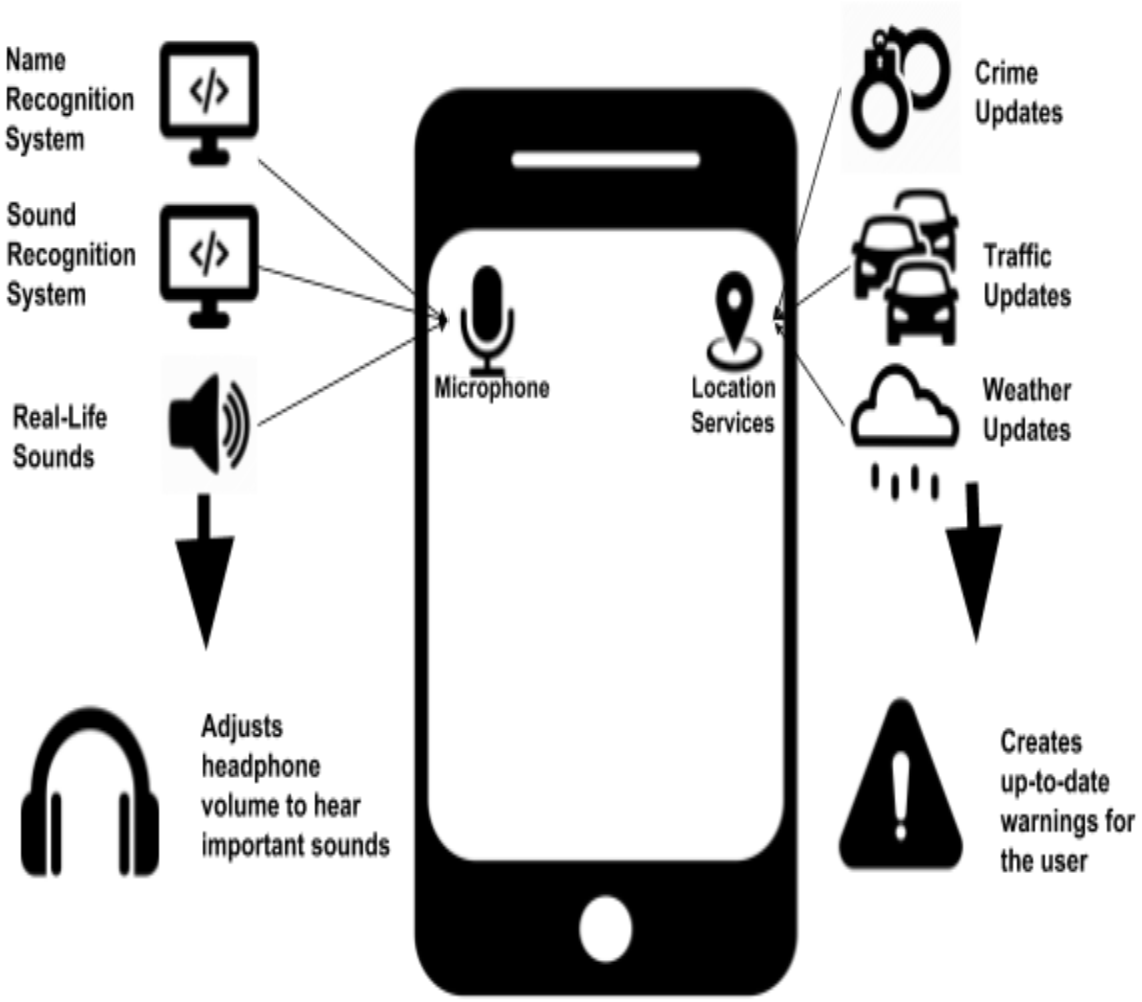
Interface Design Doc

Connor Kutz, Cole Weinhauer



The figure above shows a very general design of our system

System Overview

The desired outcome of our system will be an android application that can be used as an interface between users wearing headphones and incoming audio stimuli. Specific goals in terms of functionality include alerting users about upcoming heavy traffic and crime using GPS location services, changing the volume levels of your device based both on general and specific audio, like your name or screaming, honking, and other things of the nature, and finally creating customized.

Name Recognition System(Cole)

The Name Recognition System is a Python-based machine learning algorithm that is intended to allow users to say their name and then develop a sound profile around it to later be recognized when it is said out loud. This will also be backed up by a speech to text library that will be used if the model encounters a word that it recognizes as the name but with low confidence. These sound profiles will be developed using the AudioRecord class that comes built in and the SciChart libraries that help further analyze the sounds. For documentation, look below.

AudioRecord: https://developer.android.com/reference/android/media/AudioRecord
SciChart: https://github.com/ABTSoftware/SciChart.Android.Examples

Sound Recognition System(Cole)

Like The name recognition system, the sound recognition system is a Python-based machine learning algorithm. The system utilizes the UrbanSounds8K Dataset as developed by J. Salamon, C. Jacoby and J. P. Bello. This Dataset supplies 3 things for each entry: a WAV file that is composed of the sound of the event, the metadata of that file, and a classification of the sound. Then features are constructed using the Librosa and Scipy.io.wavfile libraries. These features include the number of channels, the sampling rate, bit depth, mel-frequency cepstral coefficients (MFCCs), tonal centroid features, and other sound based features. Next, these features are used with a Tensorflow machine learning algorithm to develop a model to predict the classification of real-time urban sounds. This is then deployed onto the app and to be used with the microphone to listen for certain sounds, like car alarms, screams, or gunshots. This is done to limit the loading time of the algorithm. For full documentation and information, look below.

Librosa: https://librosa.github.io/librosa/
Scipy.io: https://docs.scipy.org/doc/scipy/reference/io.html
UrbanSounds Dataset: https://urbansounddataset.weebly.com/

Weather, Traffic, and Crime Updates(Cole)

The weather, traffic, and crime updates will be done through APIs through the Android Application. This is because they dont require the computational power or time like the algorithms listed above. For weather, we will use the Openweathermap API, which gives current weather forecasts and information. For crime, we will use the Spotcrime API which can find crimes within a certain radius as well as filter them by types of crimes. Finally, for the traffic, we will use the Google Maps API which will be used to record traffic by area. We intend to use a system that updates whenever the user enters a new area as well as updating over time. Documentation can be found below.

Openweathermap: https://openweathermap.org/api
Spotcrime: https://spotcrime.com/
Google Maps: https://developers.google.com/maps/documentation/android-sdk/intro

Location Helper (Connor)

This location Helper will contain any objects necessary for obtaining the user's most recent location.
An invocation of getCurrentLocation() will return with a Location object containing Latitude, Longitude, etc

Maps Activity (Connor)

This activity can be initialized with a call to startActivity(intent) from anywhere within the app
Will contain switches that toggle the visibility of the Crime data, traffic data, etc

Microphone Service (Connor)

This service will contain the worker thread that uses the microphone in the background
Provides links to microphone and sound processing libraries
It contains a service to actively measure ambient noise level and adjust media output volume
Gives the ability to add threads to perform other background tasks