Project Proposal Report

Team #11

Team Members:

Andrew Gould
Turner Graham
Joshua Haxton
Braden Lockwood
Yuelin Xie

Project Name: JayHear

Project Synopsis:

Audio denoising web application used for removing background noise from audio files.

Project Description:

The project we will be doing this year is denoising an audio file. The simple idea is that you feed a noisy file, a file with background noise such as loud voices or loud cars, into the model and it will give you the same audio file except without the noise.

This project is being undertaken because it is still a real world problem in how to properly denoise an audio file. Even more specific, this is a real world problem in the realm of hearing aids. Hearing aids amplify every signal coming in, instead of important ones such as someone talking to you. By undertaking this project we hope to find a way to not only remove the background noise, but to enhance important signals such as people talking.

The end goal of this project can either go two ways. It can be implemented on a website so that people may upload audio files they wish to denoise or it can be implemented onto an embedded system, such as hearing aids. Later in this semester we will narrow down our choice and decide which route we will go down.

Project Milestones:

Fall 2021:

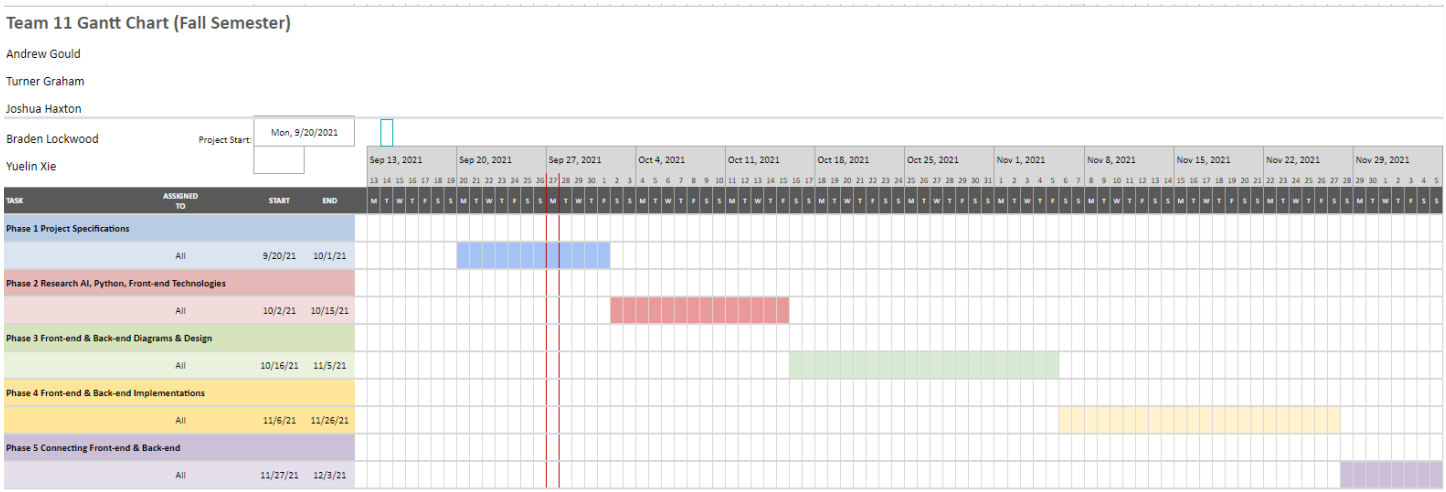
- Milestones and Project Requirements Defined (October 1)
- Research Python and Frontend Tech (October 15)
- Front-End and Back-end Diagrams and Design (November 5)
- Front-End and Back-end Implemented (November 26)
- Connecting Front-end and Back-end (December 3)

Spring 2021:

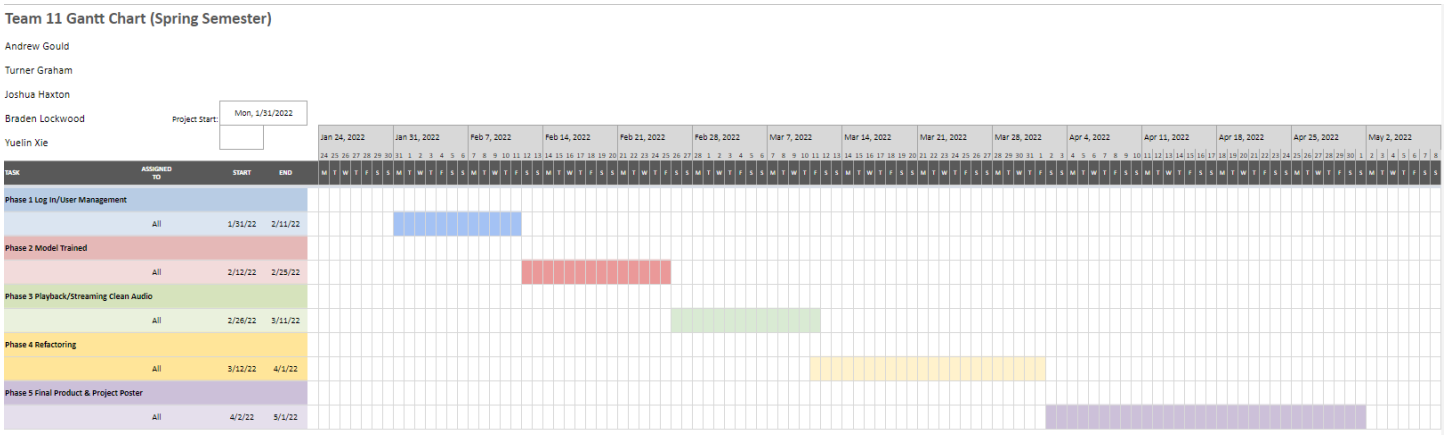
- Log In/User Management (February 11)
- Model Trained (February 25)
- Playback/Streaming Clean Audio (March 11)
- Refactoring (April 1)
- Final Product & Project Poster (May 1)

Gantt Charts

Fall Semester:



Spring Semester:



Project Budget:

Since our project will be training an AI model we will need a sufficient amount of storage to store the data. To do this all we will need an external SSD between 256GB and 512 GB. These usually range from 100-200 dollars. We can typically buy these from Amazon.

Amazon

Another thing we will need is a GPU to train on. Luckily, there are a lot of free ones online that we can use, like Google Colab. If this does not work, we can rent one that charges per minute we use it. This can range from 50-75 dollars.

Google Colab

SSD: $100-$200, Date needed by December 1
GPU: $50-$75, Date needed by December 1

Total: $150-$275

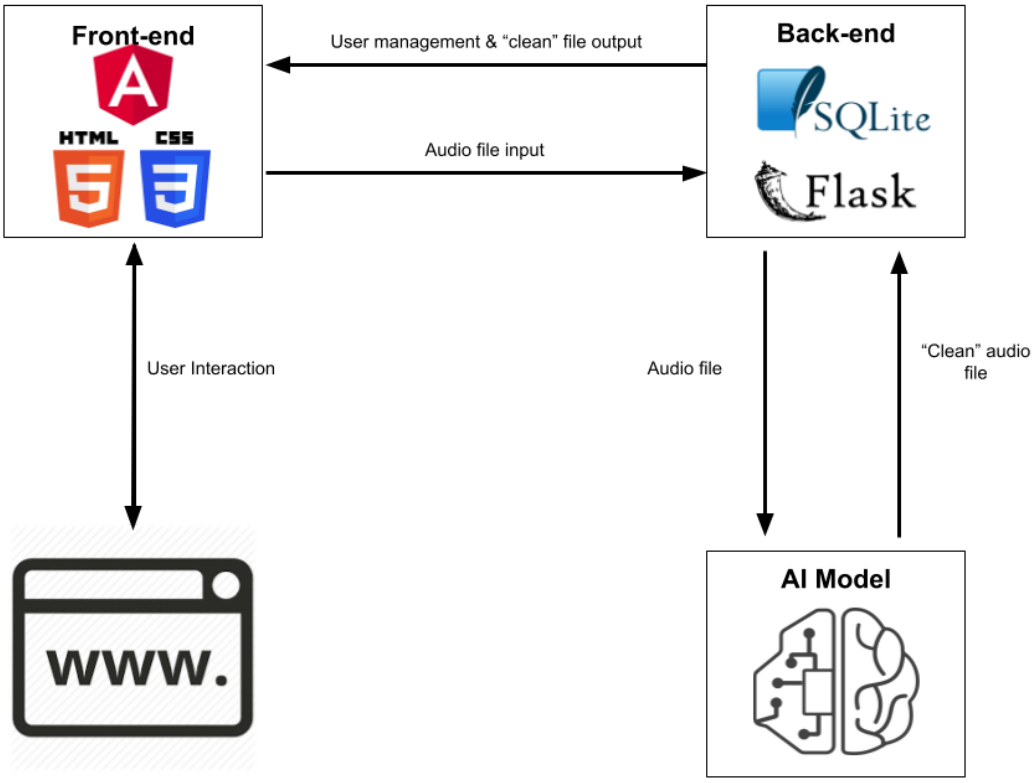
Preliminary Project Design:

JayHear is a web based application that consists of a web frontend and a backend. The application will have three main components. The first component is the frontend. This will be the main interaction between the user and our application. The frontend will utilize the user's inputted audio file to communicate with the other two components. Then once a "clean" audio file is returned from the other two components, it will allow the user to play or download the new file. Additionally, the frontend will help display and manage the user that is logged in and the saved files tied to their account.

The second component is the backend. The backend will handle the creation of user accounts and any account authorization. The other main purpose of the backend is to act as the middleman by retrieving the audio files from the front end and then passing that file to the other component. Once the last component is finished with the original audio file, it will then use the backend to send the updated file to the frontend.

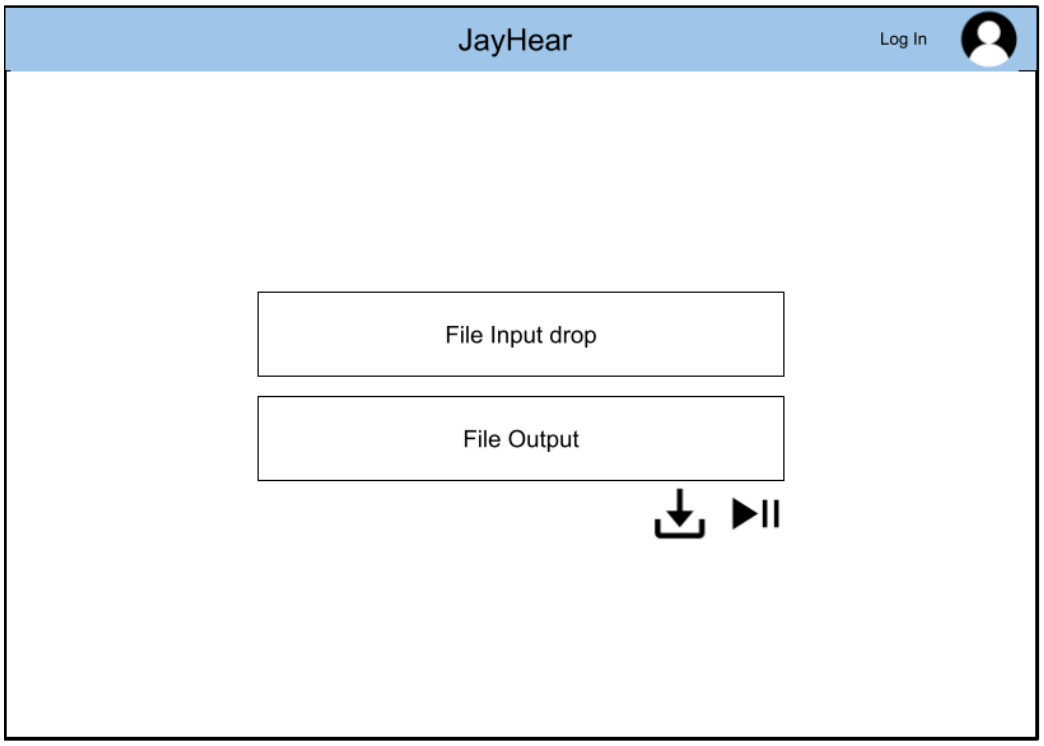
The last component is the AI model. The AI model will utilize the concept of deep learning to denoise audio files. The model has two main parts: the convolutional denoising auto-encoder and denoising auto-encoder. Both of these auto-encoders are crucial in creating "clean" versions of the audio files.

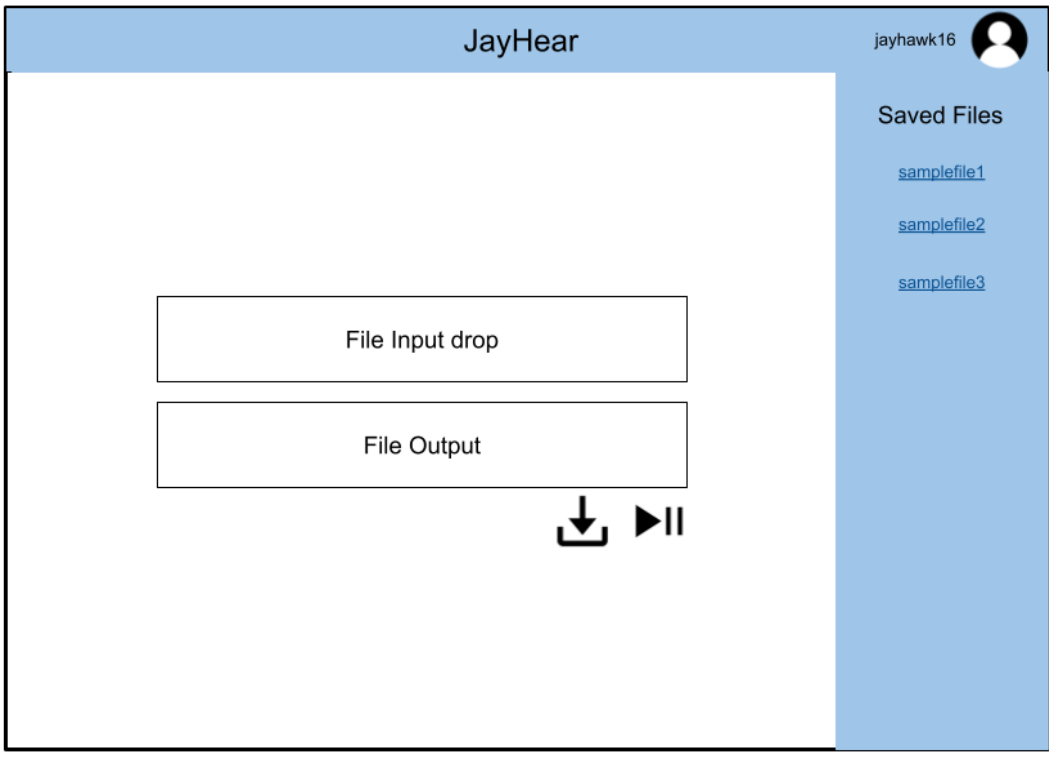
Software Stack:



Front-end/UI :

For the web application, we will implement the front end using Angular, HTML, and CSS. Since our web application mainly interacts with an audio file that the user uploads, the implemented interface is simple and minimalistic. The primary focus of the interface is on the file input and the file output functions of our application. Therefore, the interface has a section for inputting an audio file and a section for audio file output that allows a user to play or download the resulting "clean" audio file. The secondary focus of the application is on a user's account and their saved files. The interface will have a small section dedicated to this purpose by showing the logged in account and their previously saved files. The pictures below are the outlines of our plan for the user interface.





Backend:

For the back end implementation of Jayhear, we will be using the Python web framework, Flask. Flask is a barebones implementation of a standard Web Server Gateway Interface, which will handle the requests needed to provide the Jayhear service. Jayhear will require an implementation of account creation and authorization, an interface layer that accepts sound files from the client side, a database layer where user sound files can be stored, and another interface layer that allows for input of the audio files to the machine learning

model, as well as retrieval of the output from the model. Flask, as a framework, prides itself on being a slim framework, with many common features being excluded, especially in comparison to its sister framework, Django. Although in some ways this increases development time, the reduced execution overhead and complexity, alongside the availability of third-party extensions that supplant much of Django's prepackaged functionality, results in a much leaner and performant web application. Since Jayhear is tasked with processing audio files via a machine learning model, performance and delivery in a reasonable timeframe is a key factor of a successful application. In addition to the processing of audio files, Jayhear will implement storage of processed audio files for retrieval later in time. These files will only be accessible by the account holders who originally processed the audio files. To facilitate the storage of these files, a database layer that will store the metadata for these files, as well as a disk storage that will store the actual files. For the database layer, we will use SQLite, which removes the necessity of running a dedicated SQL server and embeds the database into the application itself.

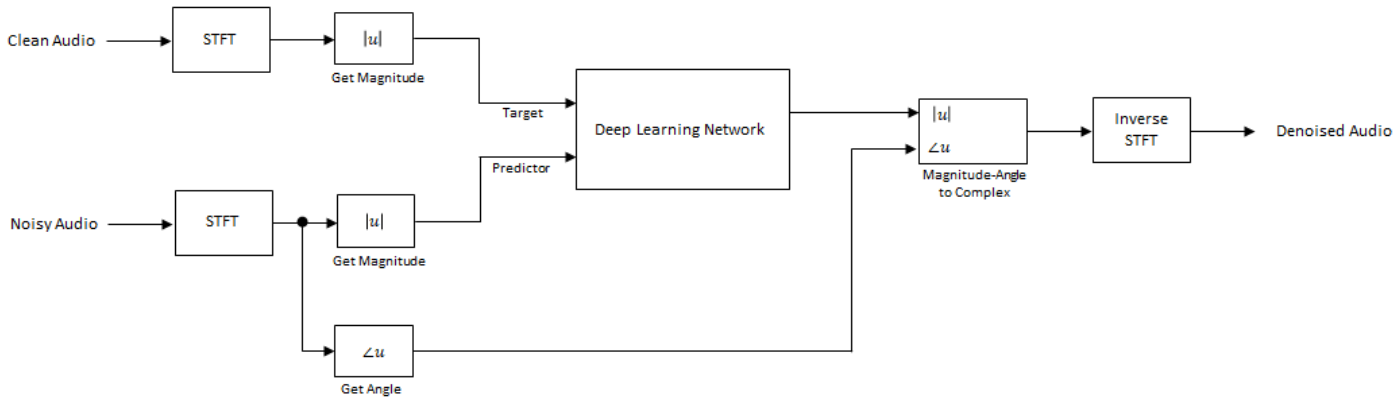
AI Model:

The process in which Jayhear will remove the unwanted audio is through machine learning, more specifically deep learning. Deep learning has been an idea for a while. It first took form back in 1961 when Frank Rosenblatt created the multilayer perceptron which looks to copy how the brain works with its interconnected neurons. Then in the mid 1980's, Geoffery Hinton and others created an algorithm called backpropagation. This algorithm computes the gradient with respect to the loss functions and changes the weights of the network with the hope that it will improve with accuracy. After this algorithm had been discovered, deep learning was quiet for a time due to hardware limitations. Then in 2012, AlexNet was created and performed extremely well in the ImageNet competition compared to other non-deep learning models. After this, the popularity of deep learning began to rise.

This is a very brief background on deep learning just to give some context to our application's capability. What Jayhear will use specifically is called a convolutional denoising auto-encoder. This uses a convolutional neural network(CNN) because it excels at recognizing images. CNN's excellence with recognizing images is a result of it using something called a kernel. The kernel scans over an image and extracts features from them. These features could be something like an edge on a door or a line on the street.

The next part of the model is the denoising auto-encoder. An auto-encoder is a generative model meaning that it will recreate what you put in it. If we fed a spectrogram into the auto-encoder, the model would encode the image into a compressed version, and then decompress it back to the original image. Then we would compare the original image and the image created from the network to see how different they were and compute the loss. After computing the loss, we would then use backpropagation, as mentioned earlier, to change the network in hopes that it would become more accurate.

This is when the denoising portion of the auto-encoder comes in. Instead of feeding the auto-encoder an image, we feed it a noisy image. Then instead of comparing the original noisy image to the one created, we compare the denoised image to the one created. We hope that through comparing the denoise image and the one created that the network can find a mapping between the two so that when the next noisy image is fed in, it can denoise it better than the previous time.



The image above gives the path the data will follow in Jayhear. After the audio file is passed into the backend through the front-end, it will go through some data preprocessing. After the data has been cleaned up, it then reaches this diagram. First, it will be transformed into a spectrogram using the STFT algorithm. This process gives us a visual representation of the frequencies of the signal as it varies with time. We then can take the absolute value of STFT to remove the complex numbers. It is then fed into the convolutional denoising auto-encoder. After the new image has been generated, we apply phase aware scaling to recover the phase, which was removed when we removed the complex number. After this recovery, we apply the inverse STFT and we will have our denoised image.

This is a system that can also be implemented in an embedded system. Since the network has ~33,000 parameters it is relatively lightweight, which makes it perfect for embedded systems. The embedded system of choice would be hearing aids. If we could get this to actively denoise audio in real time, it could be a huge leap forward for people with hearing aids. This is because hearing aids amplify everything coming in, instead of only enhancing specific audio such as people talking. This could be a possible next step of the project after the web version of the implementation is complete.

Design Constraint:

Technical Constraints

Programming language: Our application utilizes the Angular framework to implement the front-end. Angular bundles HTML, CSS, TypeScript/JavaScript to take care of its functionalities. Additionally, this framework utilizes Node.js to help build, develop, and compile the application. In regards to the back-end, we will utilize the Flask framework to interact with our front-end and our defined AI model.

Platforms supported: Our application aims to have an accessible web application for all computers. As a result, the front-end has to have a scalable interface for all sizes of monitors and screens. This means that the user interface must have code that changes the interface's width and height in reference to a screen's dimensions. We have recognized this as a manageable constraint due to the capabilities of CSS and available libraries, such as BootStrap, in controlling display properties.

Since our front-end utilizes the Angular framework, our applications rendering is determined by platforms that support our particular version of Angular. Currently Angular is supported by all of the latest versions of major browsers like Firefox, Chrome, Safari, iOS and Android. However, only Angular versions 9, 10, and 11 are supported by Internet Explorer. This constraint is minimal due to the decreasing popularity in the use of Internet Explorer.

Ethical Issues:

Information Security

There are no readily apparent ethical issues with Jayhear itself. The product is simply a tool for cleaning audio files. However, there are potential issues with the storage of data for the product. As Jayhear will allow users to make an account, the product will have the necessity to store information tied to the accounts (e.g. usernames, passwords, email addresses, and the users' audio files) on its server(s). Consequently, Jayhear will be responsible for the safe and secure storage of this data, in accordance with §2.1 of ACM Code of Ethics. As such, data must be encrypted, as defined by §1.6 and §1.7 Codes. As outlined in §1.2 of the ACM Code of Ethics, the systems ought to be backed up for disaster recovery. In the event of a data leak, Jayhear must be transparent with its users about the nature and extent of the breach, according to §3.1 of the ACM Code of Ethics. Finally, there must be a system in place to ensure that secure protocols are followed, as described in §2.2 of the ACM Code of Ethics.

Intellectual Property Issues:

Patent and Copyright

The main intellectual property issue we may encounter is patent and copyright issues. First, our code will be uploaded to a public repository in Github, which will make our code open source. As a result, we will not have to consider obtaining any intellectual property rights for our project. Moreover, our project utilizes technologies such as Angular, Flask, and SQLite. All of the technologies mentioned are open source, so there will be no intellectual property right infringements. In addition, the AI model that we have outlined was influenced by the paper "A Fully Convolutional Neural Network for Speech Enhancement" by Se Rim Park and Jinwon Lee, which is cited below. With this influence, we will make sure to accredit the authors of this paper. In the event that we use other people's technologies or methods in the future, we will make sure to ask for their consent, use their product within the authorized scope, and give the owner's proper credit.

Park, S. R., & Lee, J. W. (2017). A Fully Convolutional Neural Network for Speech Enhancement. *Interspeech 2017*. https://doi.org/10.21437/interspeech.2017-1465

Change Log:

Not that much changed since the initial project description. The SSD needed was much less expensive coming to be under $100. The GPU also will not cost any money due to the hardware that the team already owns.