Pitch Perfect

Authors: Oluwafunmbi Jaiyeola, Sai Chandana Korivi, Carlos Taveras
Carnegie Mellon University Department of Electrical and Computer Engineering

***Abstract*—**A web based application that provides novice singers with intuitive lessons and feedback in hopes of improving their pitch and rhythm control, as well as provide them with a foundation in music theory. The feedback is generated based on features derived from pitch detection and clap detection algorithms. Additionally, Pitch Perfect will provide ample resources and exercises needed to improve users' musical theory expertise, as well as access to resources on more advanced topics in vocal performance.

***Index Terms* — Vocal Coach, Web application, Rhythm Detection, Note Detection**

INTRODUCTION

Music is ubiquitous and has been a central part of what it means to be human for several millennia. In fact, the Smithsonian [1] claims that making music is a universal human trait that goes back at least 35,000 years. Since then, music has evolved and manifested itself in many shapes and forms, but, despite this, the most prevalent instrument continues to be the human voice.

When learning how to perform a task, it is common to seek instructors who can use their expertise to guide your development. This is no different in learning how to sing. However, as a novice singer, it can be discouraging and difficult to seek the help of a professional to develop your singing ability, especially considering the steep cost of personal singing lessons which tend to range between $50 - $100 per hour [2], and the dependency of available instructors nearby. These concerns are now exacerbated by the COVID-19 pandemic, which has almost certainly increased the prices and decreased the availability of in-person vocal instruction. In response, many instructors have started hosting their lessons online, but it can be difficult to justify paying full price for a singing lesson whose effectiveness is affected by the degradation of voice quality on platforms like Zoom.

To address this issue our group, The Ensemble Methods, is creating Pitch Perfect: a web application that helps users perfect aspects of their singing from the comfort of their own home. Pitch Perfect is not an automatic voice quality assessment tool, which is still an open research problem, but instead a platform to help novice singers perfect their pitch control and identification, understand and develop rhythm, and polish their music theory knowledge through tailored lessons and exercises with feedback on how to improve. To accomplish this, we will employ the use of pitch and clap detection algorithms to measure pitch and rhythm, respectively, as well as a note strain detector based on pitch contour statistics over a note.

DESIGN REQUIREMENTS

Our requirements are broken into several parts, highlighting the main areas of user interaction with our product, namely: pitch to note accuracy, clap detection accuracy, note strain analysis, user interface interaction, feedback, and latency. Since much of our application's functionality depends on the effectiveness of our note and clap detection algorithms, we must ensure they are robust and accurate. The user interaction focuses primarily on making our application intuitive and enjoyable for the user. Finally the feedback is the means by which our application will communicate to the user how well they are performing, as a function of note accuracy, timing, and intonation.

A. Pitch To Note Accuracy

Note detection is a critical component in Pitch Perfect's functionality, therefore our system must be able to reliably detect notes. Notes are extracted from the pitch detection algorithm, so our pitch detection algorithm must accurately estimate pitch. We require our note detection algorithm to achieve a test accuracy of no less than 95%. We will thoroughly test the note detection accuracy against pure tones with added white Gaussian noise. After achieving this baseline accuracy, we will test our algorithm against a labeled dataset of sung tones [3].

B. Clap Detection Accuracy

Clap detection is an integral part of our rhythm exercises, thus we require no less than 95% clap detection accuracy on a preliminary collection of self-annotated claps that simulate the behavior we expect from our users. Ideally, we will like to test this against an existing annotated clapping dataset to test the generalizability of our algorithm.

C. Note Strain Analysis

Note or intonation strain will provide us with a metric to potentially discriminate between good and bad singing. We extract this feature by taking statistics of pitch data over a note, particularly the per-note interquartile range. We will collect samples of good and bad singing and determine how well this parameter correlates with the classification. We expect low

interquartile ranges to highly correlate with good singing quality, and vice versa.

D. User Interface Interaction

The main requirements for the web application's user interface are usability and desirability. We want the users to be able to easily and intuitively navigate to different pages and singing exercises. Since this web application will be used for learning purposes, with the majority of our target users most likely being children, we want our web application to be engaging with appealing visuals to grab their attention as well. To ensure these requirements, we will be conducting focus group studies. We will begin conducting these studies remotely through zoom once our minimum viable product is ready to be used. Before these sessions, we will prepare a list of tasks that involve using the web application. These tasks will fall into 4 main categories:

1. Navigation to an exercise
2. Performing an exercise
3. Interpretation of feedback
4. Navigation to past feedback

During our user study session, we will provide all of the participants with the same list of tasks. While they perform each task, they will be sharing their screen so that we can monitor their performance on the tasks. While we monitor their interaction and completion of tasks, we will measure how long it takes them to complete each task. After they complete all the tasks, ask them to rate their experience on a scale ranging from 1 to 5. These survey question responses will provide us with qualitative feedback.

Overall, these focus group studies will provide us with the quantitative and qualitative metrics that we can utilize to improve the user interface.

E. Feedback

We will provide our users with useful and concise feedback. As we provide our users with both visual and textual feedback, we expect the visual feedback to be comprehendible. In order to meet this requirement, we will send audio recordings to be analyzed by our feedback algorithms. Ensuring that all visual feedback is readable and the textual feedback is not too verbose. Based on this, we will be able to modify our feedback algorithms and representation.

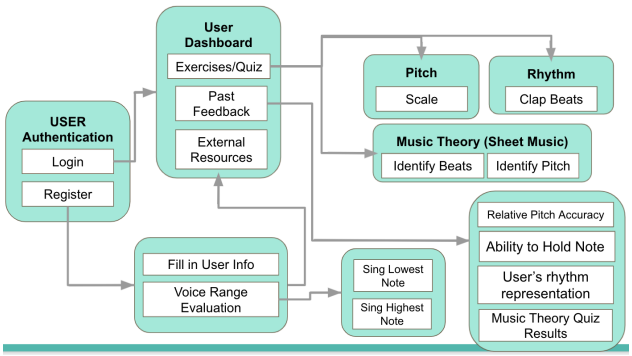
F. Latency

Long response times and high latency can adversely affect a user's experience with a web application, which can lead to low product use and a poor perception of the product itself. Thus, we want to provide the ideal response times for different types of user inputs. According to an article, "Response Times - The 3 Important Limits" [4] written by Jakob Nielsen, a web usability expert and human computer interaction researcher, there are three main response time limits for a web application. First, for user inputs that don't require special feedback from the application, 0.1 seconds is the response time limit for the user to feel that the web application is reacting instantaneously. Second, for user inputs that do require special feedback, 1.0 seconds is the response time limit for the user's flow of thought to remain uninterrupted. Third, to keep the user's attention, the response time limit for the special feedback is 10 seconds. Therefore, following the response time limits of this article, we will be limiting response times in a similar fashion based on the types of user inputs and form of feedback. For feedback on users' pitch and rhythm exercises we expect the response time of the feedback to be at most, 10 seconds. For feedback that doesn't require special processing such as navigation from one page to another, the response time will be at most, 1 second. In order to measure the response times reliably and accurately, we will be using the django-debug-toolbar, a configurable set of panels that display debug information and CPU times about HTTP requests and responses. It allows us to see what tasks the code is doing and the time is spent for each of them. Based on the insights this toolbar provides us, we can determine if we need to optimize our code in the case of any long response times.

ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

A. WebApp Implementation Diagram

All users have to go through an authentication process, as an existing user, a verification would be required at the login. If not an existing user, they would be navigated to a registration page where the user would fill out the necessary personal information as well as take a quiz on their vocal range. From this stage as well as after verification of login, the user will be redirected to the dashboard page where they have access to all the lessons, their feedback as well as external resources. The diagram below demonstrates the flow of the web application.



B. System Diagram

Figure 1: System Diagram below represents the integration of the user inputs and outputs, our pitch and clap detection algorithms, feedback generation, and hardware.

C. Django Framework (models, views, forms, templates)

The entire web application will run on the Django Framework. We have decided to use Django as it is based on Python, and already has MVC as its core architecture. This framework allows us to have large files, such as our audio files from the user as well as their music theory lessons. In addition, the framework is employed in several major projects, and is well established with a vast community to guide us through any errors we may run into. The Django Framework will also allow us to hold many of the media files and access them with low latency. All functionality of the system will be handled in the views.

D. Music Theory

Our entire product is focused on music training. For music theory, this aspect of the product is necessary for the user to understand some of the more advanced pitch and rhythm lessons. The music theory lessons are, however, provided in parallel with these lessons and can be run independently from the other lessons. We have gathered lessons from websites such as www.musictheory.net, as well as resources from classes offered at CMU. These resources will guide the lessons we provide to our users. Every lesson will have a corresponding theory exercise that can test how well the user understood the lessons. In addition there are also lessons on note duration for the rhythm exercises and listening to pitch to perfect their pitch detection for pitch exercises. After taking the quiz the user will have access to the correct answer, and all answers will be stored in the database, allowing users the opportunity to retake the lessons.

E. Pitch and Note Detection

The singing exercises will make use of this module to extract pitch from raw audio recordings. This extracted pitch will then be used in the note detector to segment the recording into musical notes. The resulting notes will then be compared against the expected notes in cents.

F. Clap Detection

Rhythm exercises consist of users clapping along to a beat. This module will detect those claps and annotate the original audio recording with their locations. To determine whether a user is clapping on beat, we will set a threshold on the allowed elapsed time between expected and detected claps. This metric will then be used to generate relevant feedback about the users performance.

G. Note Strain Analysis

Upon registration, our application will identify users' vocal range. In doing so, we must quantify their ability to sustain a note. We characterize the difficulty of hitting a note as strain which is a function of change in pitch over time. Throughout singing exercises we will also collect data on note strain as a metric to gauge user improvement. The output of this module will be fed to the feedback generation module to provide the user feedback on their performance.

H. Feedback Generation

For the 2 main aspects of the system, pitch and timing, we will be generating the feedback from the detection algorithms. For pitch feedback, after the user's pitch has been recorded, several metrics will be extracted from the user's performance and it will be evaluated relative to the exercise template. For timing feedback, the clap detection algorithm will process the user's claps and return the timestamps at which the claps are detected. Based on these timestamps, the music notes representing the user's rhythm will be shown in the visual feedback. The timestamps of the user's claps will be compared against expected times of the rendered rhythm that is in the exercise. The expected times will be following a specific tempo of 120 beats per minute. Based on how the user's timestamps and expected timestamps align, we will be showing the user if they clapped all the beats correctly, if they missed clapping a beat, or if they clapped an extra beat. For the visual feedback on timing, the user's claps will be rendered as music notes to represent their rhythm. The expected rhythm will also be rendered this way.

DESIGN TRADE STUDIES

A. Measure Intervals (Cents vs Hz)

One metric we will use to generate user feedback is the difference between the exercise and detected tones. To make this comparison, we will convert the measured pitch values to cents. The distance between one tone to the next is exponential in Hz. Converting to cents allows us to linearize frequency measurements, which is extremely useful when switching between octaves. In cents, octaves are separated by a fixed 1200 cents, whereas the difference in octaves in Hz differs by a power of 2. A semitone is the smallest interval of music within an octave and is equal to 100 cents. At this point, the musical pitch is considered to be the most dissonant when sounded harmonically. Although converting to cents will add another step to the system, its addition will greatly impact the ability of our system to provide useful feedback.

B. Pitch Detection Algorithm Selection

Since we're building a singing coach application, pitch detection algorithm selection was a critical decision in our design process. Pitch detection methods leverage properties either in time, frequency, or both in some cases, of periodic or quasi-periodic signals to estimate the fundamental frequency, or pitch, of a sound. The autocorrelation method is a common time-based approach that correlates segments of a signal in time with a shifted copy of itself over a small range of values. It estimates the periodicity of a signal by locating the shift that produces the largest peak. Another approach analyzes a signal in the Cepstral domain where periodicity is easier to analyze. Using a method similar to the Short-Time Fourier Transform (STFT), we can attain a time-cepstrum representation of the original signal which can then be used to estimate the pitch. While the cepstral method can reasonably estimate the pitch, the autocorrelation method, especially when augmented with post processing steps like in the Yin fundamental frequency estimator [5], outperforms in detecting monophonic pitch. In the end, we chose the autocorrelation pitch detection algorithm implemented by the linguistics tool Praat [11] because of its superior performance over Yin as determined in [12] and the ease of use of its python interface to Parselmouth [13].

C. Visual Feedback

a. *Pitch Exercise Feedback*: Visual feedback will be provided to users for each exercise to help them understand their pitch accuracy and timing accuracy from the scale and rhythm exercises. To represent their pitch accuracy, we initially planned to use a pie-donut chart from the Highcharts library to represent their accuracy as a percentage. Since our target users are beginners with little to no knowledge of the technical details of pitch, such as cent margins, we thought percentages would be the most intuitive way for them to get a good understanding of how accurately they sang a note. The alternative would be to show them the exact frequency that they sang a note and show that against the frequency of the note they were supposed to imitate. However, due to varying acceptable and unacceptable cent margins, this isn't an accurate representation of whether the user was singing the correct pitch. Furthermore, we later discovered that no user will have perfect pitch, so the cent margins are very high, and many factors need to be taken into account to represent a user's pitch accuracy. Therefore, a percentage representation based on the cent margin isn't a very encouraging or comprehensive feedback metric to show users, so this representation was ruled out. We later discovered that the relative pitch of each note in a scale, compared to the absolute pitch, was a better representation of how flat (lower frequency than expected) or sharp (higher frequency than expected) each note the user sings is. The magnitude of how sharp or flat each note that was sung is represented with bars in a chart generated using Chart.js. Additionally, we evaluated how well the user was able to hold each note in the scale, which is represented by the height and color of bars in a bar chart. These charts are better able to provide comprehensive and intuitive feedback representations that are encouraging for the user to understand.

b. *Rhythm Exercise Feedback:* To represent a user's timing accuracy from the rhythm exercise, we chose to represent the user's rhythm with music notes, such as whole notes, half notes, quarter notes, and eighth notes in 4/4 time. This rendering of music notes will be done through VexflowAPI [10]. We chose this form of visual representation over representing their user's rhythm symbolically with durations indicated by bars instead of notes because durations would be useful to represent when providing real-time feedback to the user on their claps. However, since we are representing their rhythm post-processing, it will not be as intuitive to understand how to clap for durations. Therefore, we will assume the user understands the number of beats each music note represents and render the rhythm for the timing exercise and the user's claps with music notes and rather than durations for the feedback.

D. Django Framework vs Flask

Django Framework was compared to Flask, as these are the two top web development frameworks in the industry both boasting of having a vast community. One of the first considerations was the database. As we would be handling the storage of a lot of files, a relational database is considered, and as Django has an inbuilt ORM we can be able to manage testing on a small scale before integrating our Amazon S3 database. With Flask, although there is a flexibility to the database, we considered issues of compatibility as well as the possible learning curve in correlation with the amount of time allotted to building the application. There is an upside of Flask when it comes to routing and views, as in this case unlike Django which requires explicit statements of response handling it's request objects are always readily available. Finally, the big decision for Django over Flask is the security. Django comes with its in-built protection against common attack vectors with injections like CSRF, however with Flask so reliant on third party extensions, there is more pressure to maintain security by monitoring these third party extensions

E. Real-time vs. Post-processing

Initially, we considered detecting and displaying users' pitch in real-time to gauge their performance at any point in time. However, this task proved to be much more difficult than we had expected. For our application to truly be considered real-time, we would need to display users' pitch within no more than 100 ms, which severely constrains the amount of processing we can perform. Under these constraints, pitch post processing would have been minimal, thus increasing the opportunity for pitch, and note detection error. Furthermore, we reasoned that the inclusion of real-time pitch feedback will distract the singer from singing leading them to perform worse than usual.

F. Singing Quality Feature Extraction

In our ideation process, we considered developing a system that would extract rich features from a user's performance based on objective metrics of good singing. In our literature review, we found a paper [6], that attempts to solve this problem. In their paper, they identify 12 generally accepted criteria for good singing, including appropriate vibrato, resonance/ring, intensity, and dynamic range, some of which the authors tried to quantify as features. In order to evaluate a user's performance, though, we would need to provide the user with a song to perform and a reference by which to compare it to. While we considered allowing users to perform a song and compare it with a reference, like in [6], we figured that the problem was too unconstrained to solve over the course of a few weeks, so we constrained the problem to one that was feasible to solve within the allotted time. Instead of focusing on a breadth of features, we decided to focus on two key aspects of a singer's performance: pitch and timing. Instead of providing users feedback on their rendition of a song, we will develop lessons and exercises aimed to improve their pitch control and recall, rhythm, and music theory foundation.

G. Hardware

To have users provide an audio input when recording their voice and their claps, as well as allow them to listen to their recordings, we will be utilizing a noise isolating, wired headset, which includes headphones and a microphone. More specifically, we will be using the Shure BRH440M Broadcast Headset. We chose to use an external microphone as opposed to utilizing a standard laptop's built-in microphone because external microphones aid audio processing. There are many factors of built-in microphones that can obscure the audio input. Laptop fans can add noise to your audio recordings, built-in microphones can pick up background noise since it picks up noises as far as three feet away and the microphone is more omni-directional. Overall laptop built-in microphones are ideal for facetime or virtual meetings rather than vocal recording. Furthermore, a user's mouth needs to be very close to the microphone when recording their audio so as to ensure that the audio quality of the recording is optimized. The microphone in the SHURE headset is a dynamic cardioid microphone with a boom microphone mounting type, which is ideal for clear vocal reproduction. We also chose to use external headphones as opposed to a standard laptop's built-in speakers because a user will still be able to hear external noises from whatever environment they are in. However, the SHURE headphones, which are noise-isolating, block out any surrounding and background noise. Additionally, since it can be inconvenient to purchase and use headphones and a microphone separately, we thought it would be ideal to use a headset so as to combine the two. Lastly, we chose the SHURE headset over other headsets because this headset is ideal for media production applications and is compatible with many audio processing softwares.

H. APIs

a. AudioSynth.js vs *WebAudioAPI*: In order to generate sounds of notes for users to listen to in order to identify and imitate pitches, we originally planned on using WebAudioAPI, but ultimately ended up using a JavaScript library called AudioSynth.js [8], developed by Keith Whor. More specifically, we planned to use WebAudioAPI to generate piano notes to represent pitches and scales. WebAudioAPI is a very commonly used API for any developers looking to record, add effects to, or generate audio in different forms. Furthermore, many examples of different implementations of the API exist online. However, after trying to implement the piano note generation using Web Audio API, we realized that it would take a lot of code and time to figure out how to get the audio output from the API to sound less robotic and more like a realistic piano. That's when we discovered the AudioSynth.js library, which had already done all the work to filter the audio to make notes sound like real-life piano notes and required us to write only two lines of code to generate a piano note. This library's main purpose was to generate realistic sounding piano and guitar notes, and that's the only scope we needed for our project, whereas the Web Audio API had a wider scope of applications which we did not need to implement. Overall, AudioSynth.js was easier to use and produced better sounds than Web Audio API.

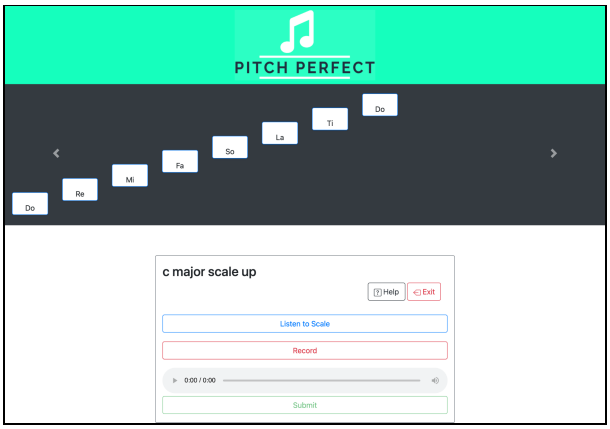
b. *MediaStream Recording API:* For the recording and playback feature of our pitch and rhythm exercises, we chose to use MediaStream Recording API. This API makes it possible to capture audio data for analysis, processing, and saving to disk. Its major interface is MediaRecorder. Mediastream represents audio tracks, and the MediaRecorder object takes the data from MediaStream and delivers it to us. We could've used WebAudioAPI to record the user's audio as well, however, there are more recording functions that MediaStream Recording provides compared to WebAudioAPI, such as pause() or resume() to pause recording your audio and resume whenever in addition to start and stop. Furthermore, it's easier and more straightforward to work with as the whole API's focus is on recording and playing back audio.

c. *Charts:* To provide visual feedback to the user on their exercises, we plan on using pie-donut charts with percentages representing the user's pitch accuracy and voice strain amount. Many APIs for rendering visual charts from data exist, however we chose to use Chart.js [9]. Chart.js has a wide variety of data representation tools that we could use, and was very simple and easy to incorporate into our code. It works very well and is compatible with most web browsers, as well as compatible with other services that we would be incorporating into our project, such as Web services and our MySQL database..

d. *VexFlow API:* To provide visual feedback on a user's rhythm exercise, we will be rendering the rhythm of the user's claps with music notes. We will match up the music note duration with their clap duration. VexFlow API is an open source online music rendering API. Since this is the only API we could find that has a library to assist in rendering music notes based on input data, there wasn't much deliberation on choosing this API amongst other libraries.

SYSTEM DESCRIPTION

Pitch - Scale Exercise



A. User Interface

The UI can be divided into four main categories. Each of these categories will have main features that distinguish them.

1. *Pitch and Rhythm Exercises*: Each exercise will have a rendering of the exercise, a pop-up instructions panel, a listen button, a record button, playback widget, and a submit button. The instructions panel will pop up when the user first enters the exercise, and the user can close it and open it whenever they need to by clicking the "Help" button until they're done with the exercise. The listen button, when clicked, allows the user to listen to either the note, scale, or rhythm they are supposed to imitate, as many times as they need to. When the user is ready, they can click the record button to record their audio input. The record button will be shaded red while the user is recording and the label will change to stop, so that the user can click it when they are done recording. The playback widget allows the user to listen to their recording. Clicking the submit button submits the audio recording for processing and analysis.

2. *Feedback:* The feedback page will be provided to the user after they submit their recording for each exercise. The feedback can also be accessed at any time after the exercises from the feedback dashboard in the main dashboard drop down. The feedback dashboard will be divided into the categories, pitch feedback, rhythm feedback, and music theory feedback. For the pitch feedback, the feedback is organized by exercise and date. Opening the feedback for a specific exercise and date will navigate to a separate page with the pitch feedback charts and recording. For the rhythm and music theory feedback, the feedback is organized in the same way, however, opening the feedback will show a dropdown with the feedback instead of navigating to a separate page.

3. *Voice Range Evaluation:* When a user is registering for an account with Pitch Perfect, they will be required to complete a voice range evaluation test. This voice range evaluation will consist of an instructions panel, and a record button. The user will record and submit the highest note and lowest note they can sing. The Range Detection algorithm will detect these notes and return their range. This range will be used to categorize the user as either Soprano, Mezzo-Soprano, Alto, Tenor, Baritone, or Bass. Furthermore, the vocal range they're categorized into will be used to customize the pitch exercises to play notes in that range, and the pitch exercise feedback to evaluate users based on their range. The vocal range test will be available for the user to take whenever they want to be evaluated again in the dashboard dropdown. The exercises and feedback will be updated every time the voice range evaluation is re-done. This vocal range evaluation feedback will be available for the user to access at any time in their dashboard dropdown as well.

4. *Dashboards:* Each exercise will be listed in the user dashboard with its description and purpose. They will be categorized and put into tabbed lists based on if it's a pitch, rhythm, or music theory exercise. The user can access their feedback by clicking on their profile icon which would take them to a page that allows them to specify which feedback (pitch, theory or rhythm) they would like to get more information on.

B. APIs

a. AudioSynth.js: The AudioSynth.js library will be used for piano note generation. For the piano note generation, we will create an instance of the Synth class provided in the library, passing in the "piano" parameter to specify that we need to generate piano notes. With this instance, we can use its "play" method to play a note, by passing in which note, which octave, and the duration of the note we want to play.

b. MediaStream Recording API: MediaStream Recording API will provide the widgets for a user to record and playback their audio inputs. To record an audio input, MediaStream and MediaRecorder objects need to be initialized. The MediaStream will capture the user's microphone stream and the MediaRecorder object will emit the recorded data as events while recording. We can use start(), pause(), and stop() commands on the MediaRecorder object. We accumulate the events into an array until a stop command is detected, and gather it all into a Blob form of data. In order to display the playback widget, we will render the Blob into an <audio> element.

c. Charts: Charts will allow us to display a user's theory feedback in the form of percentages doughnut charts, as well as the pitch feedback as bar charts to represent the users' strain analysis and horizontal bar charts to depict the users' pitch accuracy, indicating whether they were sharp, flat or on pitch.. This code would be attached to a element in the HTML. The calculated percentage would be provided to the data field of the Javascript object. Other visual customizations can be made to the object such as size, color, and labels.

d. Vexflow API: VexFlow API, which is written entirely in Javascript, will be used to render the music notes representing a user's rhythm. Each note is represented as a VF.StaveNote object, for which we can specify the clef, exact note, and duration. Since the clef will always be Treble Clef and we aren't concerned about the pitch for all the rhythm exercises, we mostly care about setting the duration for each instance of the object. For a whole note, the duration will be set as "w", for a half note, as "h", for a quarter note, as "q", and for an eighth note, as "8". To render more notes, more instances of VF.StaveNote can be added to a notes array so that they can all be rendered at once.

C. Note Detection Algorithms

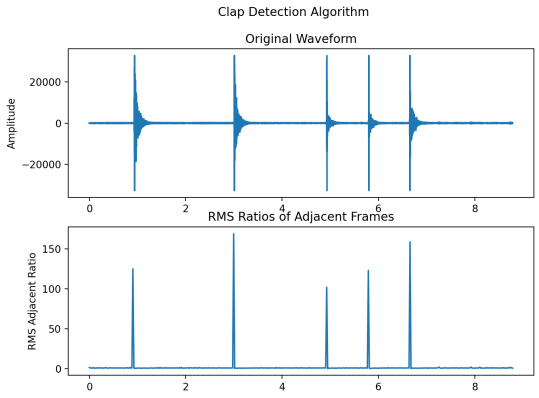
As aforementioned, we will be using the Praat fundamental frequency estimator as our pitch detection algorithm. This algorithm will be used to extract pitch from a recording of a sung performance. To extract notes from pitch, we will first convert the detected frequency to cents, take the average cent difference over an interval, and map the interval to the nearest semitone. To convert pitch measurements from hertz to cents, we use the following equation:

$$Cents = 1200\ log_2(\frac{f}{f_{ref}})$$

where $f_{desired} = 440\text{Hz}$.

D. Clap Detection Algorithm

For the rhythm exercises of our application, we must reliably detect claps to determine whether the user is keeping the rhythm specified by the program. Claps have a distinct envelope, resembling decaying exponentials. The figures below serve examples of the shape of a clap in time and in frequency as detected by a 44.1 kHz microphone.



To detect the location of the claps, we take the root-mean-square (RMS) of the original waveform in 125 ms frames, and calculate the ratio between adjacent frames. For recordings with little added noise, quiet sections are expected to have little change in RMS over adjacent frames until there is a clap, which we exploit to detect the presence of claps. We experimentally determined that RMS ratios greater than 20 will be classified as claps given a spacing of more than 125 ms, and that our algorithm is robust for SNRs greater than 14 dB. This algorithm is susceptible to false positives as a result of speech onsets, though so the only acceptable background noise is that which can be approximated as additive white gaussian noise.

E. Note Strain Detection Algorithm

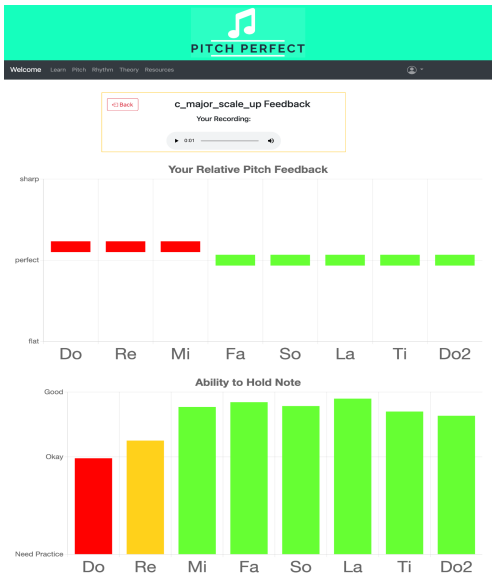
To detect the difficulty users have maintaining a note, we will analyze the statistics of their pitch over a specified range. Vibrato, which is a slight regular variation in tone, is to be expected, but large and irregular tone variations indicate poor pitch control. To capture this variability, we will compute the interquartile range per note that a user sings. An interquartile range less than or equal to 30 cents or less per note is classified as professional-like, and the greater this value, the worse the singer is able to hold the note.

F. Feedback Generation

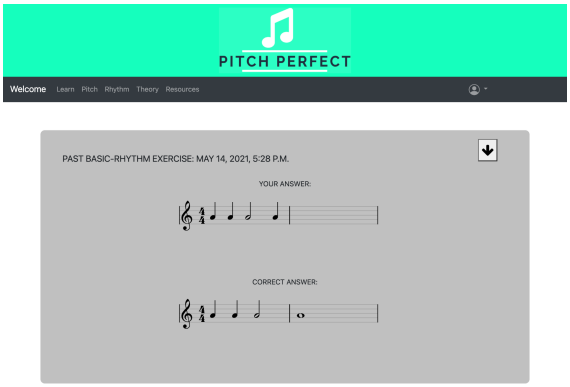
Feedback would be provided on four main metrics. Namely, relative pitch accuracy, timing accuracy, intonation and transition strain and the results of the music theory quizzes. When representing the pitch feedback, the relative error is calculated which is dependent on the users' relative pitch accuracy. As most users' especially beginners do not have perfect pitch, their relative pitch is calculated and the deviation from that relative scale is calculated with respect to the number of half steps off they are from the desired note. This is then represented in a bar chart that shows if they are too sharp or too flat. Then we have the strain in terms of intonation.The note strain detection is going to be a function of the frequency and their ability to not deviate from the desired pitch over a certain period of time. Taking into consideration the user's natural vibrato, the cent deviation would be calculated over a period of time, and then the user's ability to stay within a certain range without too much deviation would then be collated and compared to our metric of strong, good or weak strain. When generating the visual feedback to represent this strain, the deviation would be divided by 2 and subtracted from 100. This calculated value will be represented in a bar chart for each note. The value will fall into either a "Good", "Okay", or "Need More Practice" category to show the user how well they can hold a note. Then, the timing accuracy. The claps of the users are sent through our clap detector algorithm which would return back the time stamps of all claps produced by the user. The timing of these claps as well as the number of claps would be compared with the expected timing and frequency of the claps and these comparisons would be reproduced as feedback for the user in a form of notes (i.e quarter note, half note, etc).
These notes will be rendered on a staff for the user to see. The expected rhythm will also be rendered with music notes on the staff.
Finally, we have feedback on music theory. After the user goes through the lessons on music theory they would be provided quizzes which would test their understanding of these lessons. Once completed the user would then be provided a percentage based on how well they could answer the questions, as well as the solution, for some exercises, represented by VexFlow, to what they had gotten wrong.

G. Visual Feedback

Pitch Feedback:



Timing Feedback:



TEST AND VALIDATION

A. Results for Clap Detection Algorithm

In order to make sure that our test algorithm meets the expectations of getting 95% of the claps produced by the user, we tested our algorithm on a series of claps, and the times that the algorithm detected the claps were then compared to the times the claps actually occurred, The occurrence of the claps were observed manually using Audacity to document the exact time of the claps. The table below shows the time of the claps observed by the user and that obtained from the clap algorithm.

Clap #	Expected	Actual
1	1.01	1.00
2	1.87	1.86
3	2.73	2.72
4	3.57	3.56
5	4.37	4.36
6	5.19	5.18
7	6.01	6.00
8	6.43	6.42
9	6.85	6.84
10	9.07	9.06
11	9.51	9.50
12	9.94	9.92
13	10.39	10.38
14	10.82	10.80
15	11.26	11.24
16	11.69	11.68
17	12.12	12.10
18	12.57	12.56
Total	18	18

From the table you can deduce that the algorithm is very close to accurate with a margin of error of about 0.01, which in turn does not affect the feedback representation as we allow a slack of about 0.2s for the user's reaction time. In addition, the algorithm detects the total number of claps produced by the user and as such meets the design requirement we had set for the algorithm at the start of the application creation.

B. Results for User Interface Interaction Survey

To test the usability and desirability of our web application, we conducted user experience surveys with 5 people. Four of these surveys were conducted in person while one of them was conducted remotely. We originally planned on conducting these surveys after our web application had been deployed and fully integrated with the feedback detection, however, due to time constraints, we had to conduct the survey with only the exercises and quizzes and with the application being run locally. So, we were only able to use tasks from the first two categories we originally planned on using:

1. Navigation to an exercise
2. Performing an exercise

Furthermore, since we didn't get to deploy the application before the surveys, we also couldn't measure the latency for these tasks either, since we planned on using the EC2 instance monitoring logs to track the response times. The user experience survey results showed that the web application had the desirability quality, but was lacking in usability. Many of the users struggled to understand how to do all of the exercises, and quizzes and wanted more navigation features. Since the test users were able to provide very specific and helpful suggestions on what we could improve, we were able to easily improve the experience, such as adding more instructions for the exercises, adding "exit" and "back" buttons, and creating a main home page that provides more guidance on using the website. For quantitative feedback, we had users rate navigating to different pages, the exercises, and their overall experience with the application on a scale from 1 to 5, with 1 being impossible and confusing to use, and 5 being very easy and intuitive to use. The table below summarizes the survey responses with the percentage of users who gave each number rating in the columns for the tasks and features shown.

Feature	**1**	**2**	**3**	**4**	**5**
Registration	0%	0%	0%	0%	**100%**
Voice Range Evaluation	0%	0%	0%	**40%**	**60%**
Dashboard	0%	0%	**40%**	**20%**	**40%**
Pitch Exercise	0%	0%	**40%**	**20%**	**20%**

Navigation					
Pitch Exercise Experience	0%	0%	**20%**	**20%**	**60%**
Music Theory Quiz Navigation	0%	0%	0%	**60%**	**40%**
Music Theory Quiz Experience	20%	0%	0%	**20%**	**60%**
Overall Experience	0%	0%	0%	**100%**	0%

If we had more time, we would also focus on evaluating the user's experience with understanding and interpreting the visual feedback from their exercises.

C. Latency Results

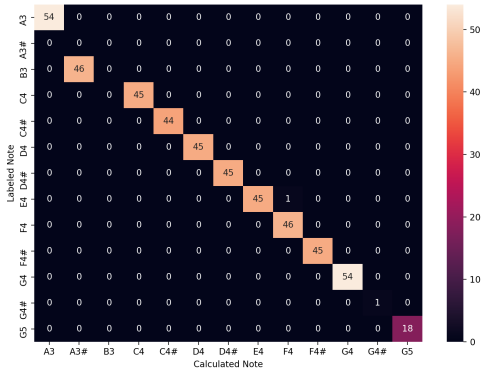
Action	Latency (in seconds)
Navigating between dashboard tabs(pitch, rhythm, theory)	< 1
Load Dashboard	3
Pitch: Load Exercise	9
Pitch: Feedback Generation	12
Rhythm: Load Exercise	10
Rhythm: Feedback Generation	6
Theory: Load Quiz	2
Theory: Feedback Generation	3

Calculating latency, we calculated the time it took for our application to load the pages and display the pages to the user, as well as process the feedback and return the visual feedback to the user. Most of the time delays were due to the time it takes to process the HTTP requests in addition to loading the APIs used in the static files. Based on the results we can say that our page is relatively close to our ideal expectation of 1 second for navigation to 10s for processing data with 7 out of our 8 tests meeting the requirements we had set for the application.

D. Note Segmentation Evaluation

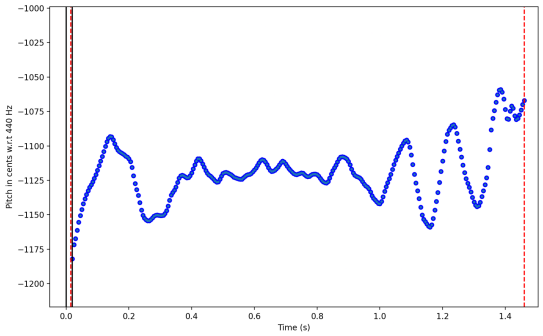
Pitch Perfect's pitch exercises depend critically on our system's ability to accurately detect pitch and map segments to their corresponding musical tones. Note segmentation proved to be more difficult than we had initially imagined because of the non-trivialities of real speech like pitch drifting, intonation, and non-syllabic sounds. The first set of tests that we ran on the note segmentation algorithm were on randomly generated pure tones with added modulation and noise. The pitch detection algorithm was able to accurately classify 100% of the pure tones that we tested it on for SNRs greater than or equal to 14 dB and for modulation frequencies less than 25 Hz. For users using the app in a quiet place, we expect SNRs much greater than 14 dB, and modulation frequencies from the singer much less than 25 Hz, so the pitch detector is robust enough.

Next, we tested our algorithm on [3], a labeled dataset of tones sung by a professional singer in different phonation modes. Below is a confusion matrix of the classification of notes of the dataset, which ranges from A3 to G5.



Note Segmentation Confusion Matrix

Our note segmenter achieved note segmentation and classification results of 95+% for all notes in the set except B3, which resulted in an accuracy of 0% for the 46 B3 occurrences in the dataset, all of which were classified as A3#. Since there is no functional difference in how B3 was calculated with respect to the other notes, we figured that there may have been a labelling error in the dataset.

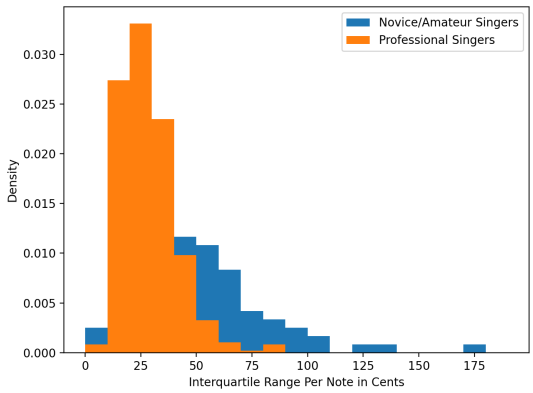


B3/A3# Tone Example

The pitch contour of one of the B3 samples which is displayed above indicates that the singer is singing approximately -1100 cents with respect to A4. This implies that the singer is singing about 11 semitones below A4, which is in fact A3#. Therefore, our algorithm was able to classify 443 out of 444 singing samples, resulting in a classification accuracy of 99.8%.

E. Singer Quality Metric Evaluation

To provide feedback to users about their performance, we calculated several metrics that were indicative of good singing. In particular, we calculated scale transposition, per note time difference, and per note interquartile range (IQR) in cents. We chose Per note IQR as a measure of how well singers can control their pitch about the median pitch value. We hypothesized that the better a singer is, the lower their IQR generally is. To test this hypothesis, we calculated this metric for the singing samples in [3] and for a compilation of samples provided by novice and amateur singers. As can be seen in the histograms below, novice/amateur singers tended to have higher IQRs than professional singers.



IQR Histograms (Professional vs. Amateur)

Thus, we decided a note with an IQR of 30 cents or less was professional like, and the higher IQR the less professional. It must be noted that there were many more professional singing samples than amateur singing samples, and the professional singing was produced by one singer whereas the amateur singing was produced by three different singers. Ideally, we would have tested this hypothesis on a much larger dataset of several professional and amateur singers, but given the resources, the evidence seems to support our claim.

PROJECT MANAGEMENT

A. Phase System

Our schedule has been broken down into 3 phases: Alpha Phase, where we would be handling most of the functionality of our product. Then we have the Beta Phase, the expectation is that this phase coincides with our interim demo, so we hope to have a semi-working application with a few bugs and some functionality that may need to be improved on, and then finally our deployment phase, our product would be deployed on the cloud, free of bugs and ready for the final presentation.

B. Team Member's Responsibilities

We have divided our project work such that Funmbi and Sai are developing the website's many pages and functions, as well as creating the rhythm and scales exercises, respectively. Carlos will be primarily responsible for the processing of users' clapping and singing, analyzing these detected actions, and interfacing with the website's frontend.

C. Risk Management

There are three main risk factors that we've considered in the design of our product. One of the risks is the possibility of external noise affecting the audio input recorded from the user. In order to mitigate this risk, we have decided to use an external noise-cancelling microphone instead of a built-in microphone in a laptop. Another risk is using a third-party implementation of the Yin Pitch Detection Algorithm might be an unreliable detection algorithm that might produce inaccurate results. Therefore, we will be thoroughly testing this module with the pitch accuracy tests aforementioned in the Design Requirements and tune the parameters of the algorithm accordingly. The last risk is variation in compatibility of our web application with different browsers and devices as our web application utilizes many different APIs. Some of these APIs might have poor compatibility with a few select browsers or devices. Therefore, to mitigate this risk, we will be building a user interface and Django code that is scalable to suit any browser or device.

D. Bill of Materials

Item	Cost
Shure BRH440M Broadcast Headset	$229.00
AWS Credits	~$10.00
Total:	$239.00

AWS CREDITS

For the AWS credits used for the project, we used a total of $3.25, most of the the tools we ended up using for this project were through the free tier (e.g our EC2 instance was the Free tier Ubuntu x86). The $3.25 spans across Data Transfer and Elastic Cloud Computing. We had hoped to move our database storage to AWS S3 to allow for a larger storage space for all our media files. However due to time constraints we were only able to have our database on MYSQL instead. We are incredibly grateful to Amazon for providing us with these credits.

ETHICAL ISSUES

Thinking about our application, the efficiency of the algorithms that we use can play a huge role in affecting the morale of our users, especially younger kids. If our product cannot accurately differentiate between good and bad signers, and wrongly tells a user that they are a bad singer than a good one, younger kids may stop singing entirely and be left really depressed and sad, especially if they had a strong passion for singing. It is for this reason that we ran our algorithms through robust checks with a small acceptable margin of error, so as to prevent such cases, or the possibility of users discontinuing the use of our application.

Additionally, our application must ensure that it is secure and can prevent the possibilities of hacking. Users' audio recordings are saved on our application and as such we need to ensure that our application is not susceptible to hackers who could modify their data and leave behind derogatory remarks, or links that could leave our users' susceptible to malware and other viruses.

RELATED WORK

Below, we listed several products that capture some of the

A. Live Singing Coach

First we have the live singing coach, an in person tutor which according to lessons.com has a cost ranging from $50 – $100 and in current pandemic situations most of these classes have gone virtual, diminishing the training quality. However, it does have the upside of being trained by a professional, as well as getting better personalized feedback and covering more singing paradigms.

B. *Yousician*

Another related work is Yousician, costing $9.99 monthly or $119.99 yearly, which, while not as pricey as an in-person vocal coach, is still relatively costly in comparison to our free app. Also, according to some vocal instructors on this site: https://singwell.eu/singing-apps/ the scoring system seems arbitrary and some of the lessons are a bit too complicated for a novice singer. although it does provide video feedback from vocal instructors to help guide the user.

C. Voco Vocal Coach

Another application that can be found in the AppStore is Voco Vocal Coach which provides the users with lessons as well as feedback on their musical performance, this app is rated a 4.1, however in the reviews it is stated to have poor instructions that makes it difficult for the user to easily navigate the app. In addition, the app is known to be constantly crashing for some of it's users rendering it ineffective for its users.

D. Pitchy Ninja

Pitchy Ninja is an application which allows the users to perfect their pitch accuracy, grading users on their ability to accurately reproduce a specific pitch as well as their ability to hold this note over a duration of time. However, pitchy ninja does not take into consideration the user's vocal range, producing lessons that are well out of their vocal range. It also does not produce useful and understandable feedback that can help the user improve their pitch.

E. Singing Carrots

The singing carrots application allows users to train their pitch accuracy as well as be made aware of their vocal ranges so as to practice with songs within their vocal range. However, this application also comes at a cost, rangine from $1 a month to $24 a month to get the full training experience. In addition, it doesn't provide great feedback on which pitch you couldn't accurately represent or the deviation of the user's pitch from the desired pitch.

SUMMARY

Overall, we were able to meet a majority of the requirements that we had set out to accomplish in order to create the application, Along the way, we changed a few of the software packages that we had initially set out to use, some of these changes were due to the inability to properly process the data or visualize the data to what we had hoped. Other reasons, like the change from S3 to MySQL were due to time constraints and hopes of deploying a working application. We had to make changes to these tools and given more time may still be able to revert some of the changes and further increase the functionality of the project.

A. Lessons Learned

The journey in the development of this project has taught us a lot about technical aspects of the application as well as the theory behind how music and singing can be evaluated and quantized.

a. API Compatibility: We used many different APIs and javascript libraries in building the frontend of Pitch Perfect. Different APIs and libraries have many different versions as well as varying compatibility with different browsers. We focused too much on finding an API that will most easily fulfill the features we need, but we should've been more attentive to whether they were compatible with most browsers.

b. Functionality vs Aesthetics: To ensure the desirability requirement for the web application, a lot of time did go into thinking about the look of the web application. However, too much time went into this planning as well

as fixating on small visual details on the frontend at the very beginning of the building process that we lost some valuable time in ensuring that the functionality of the web application was completely error-free. This delayed the integration, since we discovered that the audio file recording format had some issues very late.

c. Pure Tones: While to some degree, pure tones can be used to approximate human singing, they cannot capture the nuances of a real speech or singing signal.

d. Singing Datasets: There is an abundance of singing data readily available online, but it can be difficult to find a dataset that is usable for our specific application.

e. Characteristization of Singing Quality: Characterizing the quality of "good" singing is still an open research question. Judging how well someone can sing still remains a very subjective method. Therefore quantifying the quality of singing and providing actionable feedback on advanced singing features is still a very difficult task.

f. Cloud Deployment: To avoid the integration issues we ran into at the end of the project during deployment, we should have started deploying parts of the code at a time. This would have avoided the integration errors we ran into with scipy and apache integration, as well as provided enough time to have S3 as our main database instead of MySQL.

REFERENCES

[1] "What does it mean to be human?" 2020. Smithsonian National Museum of Natural History. https://humanorigins.si.edu/evidence/behavior/art-music/musical-instruments.
[2] "2021 Singing Lesson Cost." 2021. Lessons.com. https://lessons.com/costs/singing-lessons-cost#:~:text=The%20average%20cost%20for%20singing,and%20even%20by%20zip%20code.
[3] Proutskova, Polina. 2017. "Phonation Modes Dataset." OSF. https://osf.io/pa3ha/wiki/home/.
[4] World Leaders in Research-Based User Experience. (n.d.). Response time Limits: Article by Jakob Nielsen. https://www.nngroup.com/articles/response-times-3-important-limits/
[5] Cheveigne, Alain d. 2002. "YIN, a fundamental frequency estimator for speech and music." *The Journal of the Acoustical Society of America* 111 (4).
[6] Gupta, Chitralekha & Li, Haizhou & Wang, Ye. (2017). Perceptual Evaluation of Singing Quality. 10.1109/APSIPA.2017.8282110.
[7] Guyot, Patrice. 2018. "Yin." GitHub. https://github.com/patriceguyot/Yin.
[8] Whor, Keith. (2019). "JS Dynamic AudioSynth" GitHub. https://github.com/keithwhor/audiosynth.
[9] "Chart.js." Chart.js | Open source HTML5 Charts for your website. https://www.chartjs.org/.
[10] Cheppudira, Mohit. (2011). "VexFlow" https://github.com/0xfe/vexflow/wiki/Using-EasyScore.
[11] Boersma, Paul & Weenink, David (2021). Praat: doing phonetics by computer [Computer program]. Version 6.1.43, retrieved 13 May 2021 from http://www.praat.org/
[12] Strombergsson, Sofia. 2016. "Today's most frequently used F0 estimation methods, and their accuracy in estimating male and female pitch in clean speech." *INTERSPEECH 2016*.
[13] Jadoul, Yannick. n.d. "Praat Parselmouth." Github. https://github.com/YannickJadoul/Parselmouth.

Figure 1: System Diagram

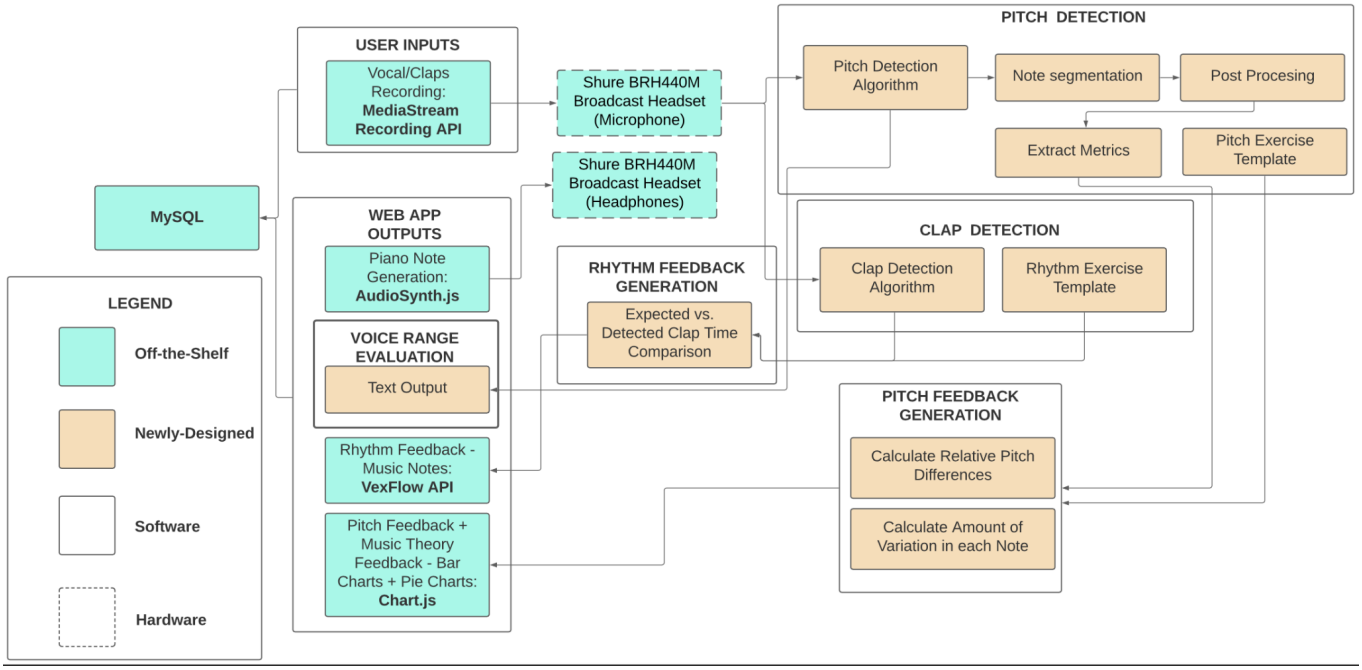


Figure 2: Project Schedule

Key	
Funmbi	
Carlos	
Sai	
General	

A. Phase Alpha

Tasks	Phase 1	
	3/8 - 3/14	3/15 - 3/21
Pitch Detection and Testing		
Pitch to tone detection		
Clap Detection		
Strain Detection		
Feedback Generation		
Pitch Matching (single notes)		
Scale lesson (includes flexibility) piano		
Login and registration		
Home page (user account)		
Lessons Page - Category 1		
Feedback Page - UI		
Recording Functionality		
Piano Note Generation		
Voice Range Evaluation		
Clap Detection Exercise Record		
User Interface Refinement		
Database relationship		
Music theory lesson		
Interval Lesson		
Metronome clap along rhythm generation		
Resources Page (Information)		
Resources Page (UI)		
Lessons Page - Theory & part of clap		
Feedback Page - Information		
Cloud Deployment		
Design Report		
Final System Integration(Feedback & survey)		
Testing		

B. Phase Beta

Tasks	Phase 2		
	3/29 - 4/3	4/4 - 4/10	4/11 - 4/17
Pitch Detection and Testing			
Pitch to tone detection			
Clap Detection			
Strain Detection			
Feedback Generation			
Pitch Matching (single notes)			
Scale lesson (includes flexibility) piano			
Login and registration			
Home page (user account)			
Lessons Page - Category 1			
Feedback Page - UI			
Recording Functionality			
Piano Note Generation			
Voice Range Evaluation			
Clap Detection Exercise Record			
User Interface Refinement			
Database relationship			
Music theory lesson			
Interval Lesson			
Metronome clap along rhythm generation			
Resources Page (Information)			
Resources Page (UI)			
Lessons Page - Theory & part of clap			
Feedback Page - Information			
Cloud Deployment			
Design Report			
Final System Integration(Feedback & survey)			
Testing			

C. Final Phase