iOS-Based Voice Analysis Application for Speech Therapy

A senior design report submitted to the
Faculty of the Electrical & Computer Engineering Technology Program
of the University of Cincinnati

This document is a partial fulfillment of the
requirements for the degree of

Bachelor of Science

in Electrical or Computer Engineering Technology
at the College of Engineering & Applied Science

by

ROBERT PFISTER
KYLE LIPPERT

Bachelor of Science University of Cincinnati

May 2013

Faculty Advisor: Xuefu Zhou, Ph.D., EE
iOS-based Voice Analysis Application for Speech Therapy

Final Report

Prepared for: Professor Xuefu Zhou
Professor James Everly

Prepared by: Robert Pfister
Kyle Lippert

April 26, 2013
# Table of Contents

Abstract .................................................................................................................................................. 1  
Introduction.......................................................................................................................................... 2  
Summary of Functionality...................................................................................................................... 2  
Problem................................................................................................................................................ 2  
Solution............................................................................................................................................... 3  
  Hardware........................................................................................................................................... 3  
  Software........................................................................................................................................... 3  
Credibility............................................................................................................................................ 4  
  Robert Pfister...................................................................................................................................... 4  
  Kyle Lippert....................................................................................................................................... 4  
Goals and Methodology.......................................................................................................................... 5  
Overview.............................................................................................................................................. 5  
Discussion........................................................................................................................................... 6  
  Project Concept.................................................................................................................................. 6  
Design Objectives................................................................................................................................ 7  
  Voice Analysis................................................................................................................................... 7  
    Local Jitter........................................................................................................................................ 7  
    Cepstral Peak Prominence.................................................................................................................. 7  
    Fundamental Frequency.................................................................................................................... 7  
  Local Notifications.............................................................................................................................. 8  
  Dynamic Exercise Files....................................................................................................................... 8  
  Easy to Use GUI.................................................................................................................................. 8  
  Later Review of Exercises and Tests................................................................................................... 8  
  Expandable........................................................................................................................................ 8  
Technical Approach............................................................................................................................... 9  
  Voice Analysis................................................................................................................................... 9
Abstract

The iOS-based Voice Analysis Application for Speech Therapy consists of a software application built to run on the popular Apple mobile devices: iPhone, iPod Touch, and iPad. These devices are significantly cheaper in comparison to the clinical devices, and many patients already own one of these devices for one of its numerous other uses, so the cost of running this application is a fraction of its counterparts. The software allows the patient to perform his or her exercises and get immediate feedback at any time or in virtually any place, in the form of readouts for jitter, frequency, and cepstral peak prominence. By providing this feedback, the patient can use his or her practice time optimally, knowing that he or she is doing the exercises precisely as prescribed by the therapist.

All of the problems which were encountered were overcome in the end, including but not limited to: developing and tweaking algorithms to give values similar to gold standard applications, conversion of algorithms from MatLab to C and further corrections on that platform, isolation of the multiple variables which may affect a usable test score.

Many improvements are planned for the future of this application, including the addition of more measures, user accounts, and linking the therapist to the patient through a network database to allow the therapist to see a patient’s progress and listen to audio samples without plugging a device into a computer.
Introduction

The primary focus of this document will be the past nine months of development and testing on this project, but will briefly touch on the work on the project which has occurred over the past two years. We will touch on the design, brainstorming, development, testing and distribution of the application.

Summary of Functionality

The iOS-based Voice Analysis Application for Speech Therapy is a mobile speech laboratory written for Apple’s iOS devices. It allows speech disorder patients to stay focused and in touch with the goals set by their clinician. The user is able to both get immediate feedback and perform his or her exercises precisely as prescribed. The immediate feedback portion is in the form of Local Jitter, Cepstral Peak Prominence (CPP), and Fundamental Frequency readings. These feedback items are crucial in the detection of dysphonia in the human voice. This dysphonia may manifest itself in the form of hoarse, breathy, pressed or nasal sounds in the voice.

Problem

Current patients of speech therapists and speech pathologists are only obtaining feedback on their progress once per week, or even less, depending on how often they are visiting their clinician. Patients are also assigned weekly exercises, which they should be performing multiple times a day, to aid in improving the health of their voice. These exercises often leave the individual wondering if he or she is performing the exercise correctly. Because the person feels that he or she may be doing damage to his or her voice, practicing tends to slip. When the patient returns to his or her therapist and is told that the exercises have been performed wrong the entire week, this discourages the patient from practicing, as well as damaging his or her voice. When the patient is at the clinic, if that clinic has invested in the very expensive apparatuses and machines, he or she can get excellent feedback on the quality of his or her voice, but visits to the clinic may be few and far between due to time or monetary constraints on the patient. The exercises are assigned in a packet form, giving the user no way to hear the example again. He or she must remember the example given while he or she was at the clinic. Further, these paper packets are not always with the patient, and the therapist cannot review any work done by the patient on these exercises.
Solution

Our proposed solution to these problems was the development of a software application that runs on the very common, very widely used iOS platform. Devices that run this platform are significantly cheaper, in comparison to the clinical devices, and many patients find themselves owning one of these devices for one of its numerous other uses, so the cost of running this application is a fraction of its counterparts. The software allows the patient to perform his or her exercises and get immediate feedback at any time or in virtually any place, in the form of readouts for jitter, pitch, and cepstral peak prominence. These feedback items are crucial in the detection of dysphonia in the human voice. By providing this feedback, the patient can use his or her practice time optimally, knowing that he or she is doing the exercises precisely as prescribed by the therapist. We also found that users tend to practice more when they were using the application. The program added a sort of gamification to the prescribed exercises, by allowing the user to “compete” with his- or herself to always be improving on his or her previous score.

Hardware

The software was developed for iOS capable devices, which means that it requires an Apple iPhone, iPad or iPod Touch for the hardware, to run the application. These devices are significantly smaller than a traditional personal computer, but contain a much higher computing capacity than computers for which benchmark software such as PRAAT or MDVP was originally written. This allows us to perform many of the same calculations performed in these programs on a device that is a fraction of the size.

Software

This project was highly focused on software. The application was written in Cocoa, Objective-C, C, C++, and included some data persistence via PHP and MySQL. The algorithms for each calculation were developed in MatLab mathematics laboratory software, and converted to C and C++, which is compatible with the iOS platform. Although the algorithms are a very important portion of this application, the use of the iOS platform allows for numerous other features to increase the user’s likelihood to use the application. The program also allows the user to review past exercises, make notes about them so that he or she can discuss it with their clinician, and even extract the audio file from the application via iTunes so that it can be sent to the clinician. A major benefit of an iOS application is the easy-to-use graphical user interface, which allows the application to assist the user in performing a task, rather than being a hinderance or layer between the user and the task needing to be completed.
Credibility

This project is a great fit for our group, as it deals with things with which we are already familiar, but pushes farther, so that it also poses a challenge to us. Mobile development is something we have done before (specifically iOS), but the addition of the voice analysis tools push for us to learn how to use many new parts of the device, as well as further our knowledge of voice analysis. The basic ideas of voice analysis are also used in other forms of digital signal processing, and we have taken classes which have given us a basic knowledge of these topics. This allows us to build upon ideas with which we are already familiar to complete the voice analysis task. In addition to the knowledge we possess, Professor Zhou acts as our senior design advisor. We believe he is a great resource in voice analysis and digital signal processing tasks, as well as the basic application development task.

The idea for this application arose directly from Doctor Eva van Leer, a doctor working in the area of otolaryngology and speech pathology. Individuals in need of voice therapy need to know where they stand between meetings with a clinician. A device currently exists that is used in many clinics, but it costs upwards of two-thousand dollars, takes up a lot of space, and requires a professional trained in the use of the device to perform the tests. Our proposed software is geared toward a simpler solution, for use by laypersons, with a focus on encouraging practice.

Robert Pfister

Rob has been developing using on the iOS platform for over four years now, and has written both iPhone and iPad applications to solve multiple different problems. He also has an extensive background in C/C++, C#, Objective-C, Cocoa, CocoaTouch, SQL, HTML, PHP, JavaScript, Matlab, and a multitude of other skills that helped immensely in the development and testing of this application. His experience with iOS development was indispensable in the development of many of the features within the application, as well as the layout of a user-friendly graphical user interface. His knowledge of C/C++ was used in the conversion of algorithms from MatLab mathematics laboratory code to C code, which could be implemented within the iOS application.

Kyle Lippert

Kyle’s primary focus was the testing and tweaking of the functions and algorithms ensuring that there are no inconsistencies in the results and accurate measurements. Kyle has been interested in software development since early high school, mostly involving Visual Basic and Java. While pursuing a bachelors degree in Computer Engineering Technology Kyle has also been introduced to many other programming languages including Visual Basic .NET, C, C++, C#, SQL and MATLAB. Although Kyle’s primary focus was on the accuracy and functionality of the algorithms, he was also helpful resource in the layout and usability of the user interface of the app.
Goals and Methodology

This project consists of two major portions: the Voice Test and the Exercises. The two primary objectives of this project are to develop a system for providing immediate feedback to the user, and provide a way for the user to perform the exercises assigned to him or her at any given time. The first objective is accomplished through the development of multiple digital signal processing tools which analyze the user’s voice and provide scores. The two measures we have chosen to incorporate are Jitter and Cepstral Peak Prominence. The second objective is accomplished through a system that allows the recording of exercise audio by the clinician, so the user knows what to do. The system sends the user a notification at set intervals, reminding him or her to perform the exercises.

Overview

The remainder of this report describes in detail how the project was completed. We will discuss design objectives, technical approach, budget, timeline, problems encountered, and future recommendations pertaining to the project in its entirety.
Project Concept

This project came about through a collaboration with the University of Cincinnati Medical Center. Dr. Eva van Leer came to us with the original idea, and it has slowly evolved into a very robust application. The original idea was a basic jitter readout, allowing the user to record a sample as long or short as he or she wished. The first version of the application is shown in Figure 1. This application did not even specify what the vocal score was, or how to use the application. It was a one use application, and even the save function (seen in the upper right hand corner of Figure 1) was not implemented until later.
As time went on, we had more ideas for useful tools to add to the application, and soon we added more analysis of the sound file, and ways for the user to keep track of his or her scores. Another item which was added later was the saving of the audio files for later review with the clinician. This arose when we realized that storing the score did not do much for us if we could not review how it sounded when that score was achieved. This spurred the idea of wanting to export the audio, as well, so the audio was later capable of being exported through the iTunes desktop application. The Exercises portion of the application was originally planned for an entirely separate app, but we later decided that it fit much better as an extension on the already existing application. This made it easier to distribute, download, and use.

**Design Objectives**

The objectives and design criteria for the iOS-based Voice Analysis Application for Speech Therapy are shown below. These are the major objectives and in no way represent the application in its entirety.

**Voice Analysis**

This application’s core is based around digital signal processing of audio recorded when the user performs a test. The voice analysis is integral to giving immediate feedback to the user at any time.

**Local Jitter**

Jitter was the initial voice analysis measure implemented in our application. We chose jitter because it is widely used for detection of voice disorders and has been used for many years. This measure can only be used on a sustained vowel sound, but it gives excellent detection of change in the fundamental frequency in someone’s voice.

**Cepstral Peak Prominence**

Cepstral Peak Prominence was decided early on as a great goal for our application. It has gained a lot of notoriety lately for being a great way to analyze for dysphonia. It correlates well with perceptual aspects of voice, and can be used for both connected speech and sustained vowel. These two things make it a slightly more desirable measure than Jitter for detection of dysphonia.

**Fundamental Frequency**

Fundamental Frequency was added when we came to the realization that not many applications on the iTunes store provided this functionality, despite the usefulness of such
data. We also could calculate it fairly easily in our Cepstral Peak Prominence algorithm, and happened to already be calculating it in our Jitter algorithm.

**Local Notifications**

The idea for of sending the user Local Notifications when it is time to perform an exercise initially came about from Dr. van Leer's idea for the user to receive a “fake phone call”, and carry on a fake conversation with the clinician's recorded audio samples. Although the iOS software does not allow a phone-call-like notification, the local notification sends the user directly into the *Exercises* portion of the app when the notification is acknowledged. The user also configures when and how often these notifications occur.

**Dynamic Exercise Files**

The ability for the clinician to record specific exercise instruction files per user, as well as allowing as many exercise instructions as the clinician wishes, was highly important for the *Exercises* portion of the application. By allowing this, the application can be customized to each user, giving optimal results.

**Easy to Use GUI**

The idea of a highly easy to use Graphical User Interface (GUI) was also very important, to make the user want to use the application. We wanted the user to feel the application was a help, rather than a hindrance, in completing the work assigned by the clinician. Also, being an iOS application, a certain user-friendly feel is expected by the owners and users of these devices.

**Later Review of Exercises and Tests**

Saving scores and audio files, as well as allowing the user to make notes on the saved tests made sense because the clinician would be able to hear the user’s progress over the time he or she was unable to come in for an appointment. Also, the user can compare exercises much more easily when he or she can hear the past exercises.

**Expandable**

The opportunity for expansion of this application was immediately realized when looking at other voice analysis software from the past. These applications had numerous voice analysis measures, so we wanted to be certain that these measures could be easily added later, as we developed the code. Making the application in general expandable, as well, was important because we hope to one day make the application connect to a website, allowing the clinician and user to connect through web-based logins.
Technical Approach

The research, design, development and testing of the iOS-based Voice Analysis Application for Speech Therapy was an in-depth process. The information contained within this section details the multiple different functions built into the application. The Appendix at the end of this document details the code involved, with descriptions of each object and the functionality of each method.

Voice Analysis

During the Voice Test portion of the application, audio is recorded in the WAV file format. The audio file is then processed, breaking the file into “chunks” and removing the headers. The “FMT” and “RIFF” headers store useful information, such as sample rate, number of channels, “chunk” data size, and number of bytes read. The “data” portions extracted from the audio file are then passed off to an algorithm to calculate Local Jitter and Cepstral Peak Prominence. These are handled in two separate algorithms, to allow each to be calculated side-by-side via multithreading. Fundamental Frequency is calculated in both of these algorithms, using different methods.

Local Jitter

Local Jitter is the cycle-to-cycle variation in the fundamental frequency, i.e. the average absolute difference between consecutive periods. This is a useful analysis technique because it detects, hoarse, nasal, pressed, breathy, and a multitude of other characteristics in one’s voice. Jitter is measured on a percent scale, with a good score being lower, meaning that the fundamental frequency changes very little between cycles. Although this measure is only proven to work for sustained vowel sounds, we had significant luck during clinical testing with using it as an indicator for connected speech using a monotone voice. As a result of this, we analyze Jitter in both the sustained vowel and connected speech portions of the Voice Test. The variation in Jitter tends to be between one tenth and four percent for a sustained vowel sample, and between four and twenty for a connected speech sound using monotone voice.

This algorithm works by finding the period of the sound and breaking the sound file down by each period. Each period is then analyzed for its fundamental frequency, and these values are then compared to the previous period. The change is recorded, and the average is displayed to the screen. the fundamental frequency is found via autocorrelation techniques, and although the average fundamental frequency is calculated in this algorithm, this is not the fundamental frequency that is sent to the user. This is due to this technique not working well for connected speech analysis.

During the testing of this algorithm, we found that we had a very close correlation to the current gold standard software, MDVP, as shown in Figure 2. As in all of the comparison diagrams that will follow, much of the difference in values comes from the fact
that our algorithms clip out the middle of the audio file, to prevent possible analysis of noise before the user begins the test sound, or if he or she ends early. This also avoids any noise which may occur as a result of the microphone turning on or off.

![Graph showing comparison of Local Jitter Values for Different Audio Files](image)

**Figure 2: Comparison of Local Jitter Values for Different Audio Files**

Jitter has been used for quite some time as an analysis technique for vocal dysphonia, but it does not show a close correlation to what humans perceive as a voice disorder. Because of this, we were in search of another analysis technique to combine with Jitter, to provide adequate analysis of multiple aspects of the voice.

**Cepstral Peak Prominence**

Cepstral Peak Prominence shows a much closer correlation to the human perception of dysphonia. It is a cepstrum-based measure which compares the fundamental frequency of one’s voice to subsequent formants. Cepstral Peak Prominence, or CPP, is measured in decibels, and is the difference between the amplitude of a linear least squares regression line drawn across the cepstrum and the fundamental frequency. To the human ear, this is most evident in how well someone’s voice carries in a large room. In this case, a higher score is better, and scores usually range between ten and thirty decibels. Figure 3 demonstrates the process of analyzing a single frame of the audio. Note that the fundamental frequency in this example occurs around 138 Hertz at an amplitude of 88 decibels, which puts the regression line at 60 decibels at this point. With this information, we come to the conclusion that the Cepstral Peak Prominence is 28 decibels, a very good score.
In this case, the algorithm works by breaking the audio sample into 40 millisecond windows and finding the Cepstral Peak Prominence of each of these. The average is what is displayed to the user. Because this process is fully capable of analyzing connected speech samples, we do not use the period to break the sample, but rather use windows of a consistent size. By analyzing connected speech samples, we avoid the issue of patients becoming fluent when performing sustained vowel sounds, but still using his or her “bad voice” when doing every day activities.

When comparing our algorithm to that of the VoiceSauce software, which is based off of the Hillenbrand algorithm, we again show a very close correlation across numerous different sound files (shown in Figure 4). These sound files consisted of clinically diagnosed patients with speech disorders, as well as voices without dysphonia.
**Fundamental Frequency**

The average fundamental frequency is also displayed to the user after a test is taken. Although we measure the average fundamental frequency of both sustained vowel and connected speech samples, this value is inherently more accurate for sustained vowel sounds, because a person’s voice consistently changes frequency while speaking. The value is another indicator to help the patient get in his or her “good” voice. This value is calculated both within the Local Jitter calculation, using auto-correlation, and in the Cepstral Peak Prominence calculation, using the cepstral peak to find the fundamental frequency. Currently, only the value calculated in the Cepstral Peak Prominence calculation is being displayed to the user.

Figure 5 shows a comparison of the fundamental frequency calculated within the application alongside both the *VoiceSauce Snack* and *Straight* algorithms. Again, there was a very close correlation across a wide range of sound files. Areas where there was the largest difference were sentence-style tests. The iOS application showed higher values for these, again due to the clipping of the audio file’s beginning and end. By cutting the beginning and end off of a sentence, we consistently saw higher values, when compared to the sentence in its entirety.

![Figure 5: Comparison of Fundamental Frequencies](image)

**Local Notifications**

The iOS Notification Center was used in the *Exercises* portion of the application, allowing the user to setup reminders for when he or she should be performing an exercise. The user receives a notification with an alarm sound, as shown in Figure 6. These notifications act in the form of a message within the iOS *Messages* application in that they are only dismissed from the Notification Center when they have been opened. By swiping...
the application icon across the screen within the notification window, the user is taken directly to the Exercises portion of the application. The user then can begin his or her exercises, which are a series of audio files recorded by the clinician and played back to the user, then repeated by the user. These phrases are meant to sound like the user is in a phone call, but can be anything the user is comfortable saying in public. The audio is played through the phone’s earpiece to further mimic a phone call, allowing the user to feel less self-conscious when performing his or her exercises in public.

Figure 6: A Local Notification For an Exercise

Exercises Configuration

The exercise configuration screen of the application took a significant amount of time to program, due to its many feature rich elements. This screen, shown in Figure 7 allows the user and therapist to turn exercise notifications on and off, select between what times of day a notification should occur, set the number of exercises which should be performed, and record, re-order, and delete exercise audio selections.
By allowing the user to select between what times of day exercise reminders occur, he or she can set quiet hours while asleep or at work. This also makes the user more likely to perform the exercise immediately when the reminder appears, rather than waiting and possibly forgetting.

The exercise recurrence interval sets the number of exercises to perform each day, week, or month, allowing the intensity of the program to be scaled. This recurrence interval is taken into account when registering the local notifications with the Notification Center. This involved a lot of code to equally space the number of recurrences across a given time, and only allow them to occur between the user selected times.

Adding, deleting, and re-ordering exercise audio is also a very useful feature, as it saves time by allowing some audio to be re-used when changing an exercise plan. By changing the order of the audio, the patient is also kept more alert when performing his or her exercises, keeping him or her focused on the quality of voice being used.
General GUI Layout and Programming

A significant amount of time was spent making sure that the application felt very
natural in its flow and did not hinder the user in performing tests or exercises. The drawers
used for walking a user through a Voice Test or Exercise were all created and coded from
scratch, along with the test result screen which appears after a Voice Test. The animations
of these items were created using the UIAnimation layer and were written to closely
resemble other native iOS applications as closely as possible. It was very important to
make the user feel comfortable with the application from the first time using it because if
he or she struggled with it the first time, he or she was less likely to continue using the
application. Many items within the application are showy. This was done intentionally, to
keep the user interested. A very still, static application does not keep the user engaged,
and the return value diminishes significantly.

Budget

Because our project is mainly software-based, our budget is kept very low. Any
larger purchases needed for development have either been provided through research
funding or pre-existing pieces of hardware. IntelliDesign Software has provided the Apple
iOS Developer License needed to test this application on devices, as well as the computer
to write the software. To verify our algorithms and benchmark our application as it
compares to “Gold Standard” applications currently in use, we used PRAAT and
VoiceSauce freeware and MDVP software, in combination with a clinical-quality machine.
We were granted access to this machine by our medical advisor, Doctor Eva van Leer, at
University Hospital.

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
<th>Obtained By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Developer License</td>
<td>$100</td>
<td>Provided by IntelliDesign Software</td>
</tr>
<tr>
<td>Mac OSX based Computer</td>
<td>$2000</td>
<td>Provided by IntelliDesign Software</td>
</tr>
<tr>
<td>iPhone / iPad</td>
<td>$600</td>
<td>Pre-owned by Kyle and Rob</td>
</tr>
<tr>
<td>TOTAL</td>
<td>$2700</td>
<td></td>
</tr>
<tr>
<td>OUT OF POCKET</td>
<td>$0</td>
<td></td>
</tr>
</tbody>
</table>

This budget can be misleading, because it appears as though this project will be
very costly. However, all of the items required for this project are one-time purchase items.
This is by no means a “parts list” showing what must be paid for each item we want to make. Because it is software, the total cost is in overhead. There is no cost for making multiple of the final product. Furthermore, all of the items needed for this project are items that we already possessed from some other project or application.

Timeline

Our project was originally scheduled to be presented to the College of Engineering Computer Engineering Technologies faculty before March 26, 2013, which was the date that it was to be presented to the entire university. We later came across another deadline, which was for a grant for which Professor Eva van Leer wished to apply. This pushed our deadline forward to February 15, putting a rush on the project, and moving all of the development items forward significantly. This left us with a significant amount of time, however, for testing from the February deadline to the Technology Exposition in March. The Chart shown in Figure 8 shows the allotment of time for each item in the project. There is a bit of flexibility built in to the end of the project, to allow for testing and repair of minor issues as they are discovered.

Many items are contingent upon the completion of a previous item to allow it to begin, such as testing always relies on completion of development, which relies upon completion of research. These correlations are shown by an arrow connecting the end of one task to the beginning of another. This Gantt chart shows the time spent on the four major portions of the application: pitch, jitter, cepstral peak prominence, and the “Exercises” portion of the application. It also includes ample testing and debugging time.
Problems Encountered

Throughout the development of this application, many problems were encountered. All of the problems were overcome, and the major problems are detailed below.

iOS Low-Frequency Cutoff

In the early versions of our application, we were experiencing a problem with low and deep voices. The scores these voices would receive were not consistent, and seemed to be way off of what the voice actually scored. This was due to a high pass filter built into a software filter within iOS devices. This filter began attenuating the signal as high as 200 Hertz, causing the fundamental frequency to be cut from the audio sample. With the release of the sixth version of the iOS software, Apple made it possible to turn this filter off and receive raw data from the microphone. This was not done automatically, however, so a significant amount of research and coding trial and error was performed to get the filter removed. Once this filter was removed, the application showed consistently accurate results for any range of voice types.

Algorithm Conversion

The algorithms we used in this application were initially developed using MatLab mathematics laboratory software. This allowed us to troubleshoot the code far easier. We were able to graph the data and make adjustments to the code to get it close to the gold standard software we were using for comparison. Once the MatLab code was completed, however, we needed to convert this code into C/C++ code readable by the iOS framework. This was difficult because many simple functions which are mathematical operations built into MatLab needed to be explicitly coded when converting to C/C++. Furthermore, once the code was successfully translated, we needed to adjust values again to return the results to a match of the gold standard software. This proved to be a time consuming task, as when we believed we had the code finished and working, we would analyze a file which would result in outliers. After numerous iterations back and forth, we have come to a point where the algorithms are stable for a very large number and wide range of sound files.

Testing and Comparisons

Testing the algorithms within the application also proved to be a difficult task when comparing to the gold standard software. We initially analyzed the sound files bypassing the microphone on the iOS device. This was done to keep as many variables equal as possible, allowing us to edit the algorithm to match in comparisons. After this was finished, we needed a way to test the iOS application using the microphone, and comparing to the same audio put through on the gold standard application. For this, we used audio files played through high quality, flat response speakers, as well as using live voices. This
proved to be very difficult, because the recordings had to be started and stopped at the same time to avoid adding another variable element into the comparison. This was the initial use of a set recording time in the Voice Test portion of the application. With the application stopping the recording after a certain amount of time and telling the tester, the tester could focus on stopping the gold standard application. This is what later sparked the conversion from a completely user-reliant start and stop of voice recordings to the drawer which walks the user through the entire testing process.

**Clinical Testing**

Due to the collaboration with University of Cincinnati Medical Campus, and the grant application deadline of mid-February, we were able to test this application in a clinical trial with speech therapy patients. Many of the changes which were made to the graphical user interface and the basic functionality of the application were done due to feedback from patients who borrowed iOS devices from the clinic to use for a trial period. For these tests, patients were also asked to fill out a System Usability Scale (SUS) form. The resulting data from approximately twenty patients who used the most recent version of the application (v1.0) is summed up in Figure 9. Note that this percentage is strictly meant to be compared to the percentage score of other applications. It is not an objective score. The current average shown below is the average of all similar applications whose results have been submitted for inclusion in the trial.

![Figure 9: System Usability Scale Comparison](image)

During the use of the application, we also collected some comments from the users. We were consistently told that the application was easy to use, would encourage them to practice more, made them more interested in improving their voice, and many other encouraging comments.
Future Recommendations

While this application will be released in the iOS App Store in the coming months, and has been presented at the Technology Exposition, it is by no means a finished product. We will continue to develop this application and release new updates regularly. Some suggestions that we have come up with for additional features are listed below.

Addition of More Measures

This application was written with the idea of adding more voice measures in mind. The Voice Test portion is highly modular, analyzing the WAV header once and collecting the data, then sending the data off to different methods to calculate the measures. As new measures are developed and converted to C/C++, we will very easily be able to add them to the application for use.

Cloud-Based Sharing

Ultimately, we would like users to setup credentials and accounts, allowing them to tie their account to their clinician. By allowing this, the clinician will no longer need access to the user’s device to listen to tests or exercises and give feedback. This would make clinical feedback significantly more available and could significantly increase the effectiveness of clinical speech therapy. The user would wait a much shorter time for results from the clinician, getting feedback before he or she has had time to create a bad habit with his or her voice.

Improved Microphone Calibration

While this application was developed for iOS devices, which all have a similar, high quality microphone, we believe there may be room for marginal error when testing between two separate devices. We would like to improve our calibration method so that the microphone has less impact on the results of the Voice Test. This may also open our application to be ported to other devices in the future, allowing virtually anyone with a smartphone or similar device access to the application. Tests were performed analyzing the same audio on two different devices, and results were very similar for the currently released iOS devices, but the future devices may have significantly different microphones, causing different results between devices.

Variable Voice Test Length

Currently, a Voice Test is recorded for only a few seconds, which shows excellent results for both sustained vowel and connected speech. The problem is that the user may need to take a breath during the connected speech, while he or she can probably hold the sustained vowel much longer. Taking a breath introduces significant white noise into the
analysis, and can occasionally skew scores. By allowing the user to select how long a Voice Test recording should be, he or she could optimized the results. It is best to have the longest recording possible before the user runs out of air to continue the sound efficiently.

**Score Graphing and Comparison**

Currently, the user selects a single instance of a Voice Test to look at the scores obtained. These scores are color coded, but that only gives the user an objective goal. We would like to graph the user’s test scores so that he or she knows whether or not it is improving. This would also encourage the user much more to compete with him- or herself, because the previous scores would be shown directly.
Conclusion

The final outcome of this project is an iOS application which has been tested extensively and is ready for release and sale to the public. While there are always areas to improve on a product of this caliber, we believe that we currently have a highly usable tool in aiding speech therapy clinicians and their patients in getting the most out of their time and recovery. The items listed above, in the Future Recommendations section will begin development in the following months, and will be rolled out to the public in updates via the iOS App Store. We believe this product is robust enough to hold a place in the market, and we are excited to see where this project will lead us in the following months and years.

Various Application Screenshots

Figure 10: Various Voice Test Screenshots
Figure 11: Various Exercise Screenshots


Contents

1 Hierarchical Index ............................................ 1
  1.1 Class Hierarchy ............................................ 1

2 Class Index ............................................... 3
  2.1 Class List ................................................ 3

3 File Index ..................................................... 5
  3.1 File List ..................................................... 5

4 Class Documentation ...................................... 7
  4.1 CoreDataTableViewController Class Reference ............... 7
    4.1.1 Method Documentation .................................... 7
      4.1.1.1 performFetch .................................... 7
    4.1.2 Property Documentation ................................ 7
      4.1.2.1 debug ............................................. 7
      4.1.2.2 fetchedResultsController ......................... 7
      4.1.2.3 suspendAutomaticTrackingOfChangesInManagedObjectContext . . . 7
  4.2 CoreDataTableViewController() Category Reference ........... 8
    4.2.1 Property Documentation ................................ 8
      4.2.1.1 beganUpdates ................................... 8
  4.3 Exercises Class Reference .................................. 8
    4.3.1 Detailed Description ..................................... 8
    4.3.2 Property Documentation .................................. 9
      4.3.2.1 dateTaken ........................................ 9
      4.3.2.2 name ............................................. 9
      4.3.2.3 note ............................................. 9
      4.3.2.4 soundFilePath ..................................... 9
  4.4 IDAAddExerciseCell Class Reference ......................... 9
    4.4.1 Detailed Description ..................................... 9
    4.4.2 Property Documentation .................................. 10
      4.4.2.1 addExerciseButton ................................ 10
      4.4.2.2 addExerciseLabel ................................... 10
4.5 IDAAddExerciseViewController Class Reference ........................................ 10
  4.5.1 Detailed Description ................................................................. 11
  4.5.2 Method Documentation .............................................................. 11
    4.5.2.1 cancelPressed: ....................................................................... 11
    4.5.2.2 playPausePressed: ................................................................. 11
    4.5.2.3 recordPressed: ....................................................................... 11
    4.5.2.4 savePressed: ......................................................................... 11
    4.5.2.5 sliderChanged: ....................................................................... 11
  4.5.3 Member Data Documentation ............................................................ 11
    4.5.3.1 audioProgressTimer .............................................................. 11
    4.5.3.2 recordedTmpFile .................................................................. 11
    4.5.3.3 recordLightTimer ................................................................. 11
  4.5.4 Property Documentation ............................................................... 11
    4.5.4.1 audioSession ........................................................................ 11
    4.5.4.2 cancelBarButtonItem .............................................................. 12
    4.5.4.3 exerciseNameTextField ......................................................... 12
    4.5.4.4 exerciseNoteTextField ......................................................... 12
    4.5.4.5 levelMeter .......................................................................... 12
    4.5.4.6 lightUpImageView ................................................................ 12
    4.5.4.7 playbackSlider ..................................................................... 12
    4.5.4.8 player .................................................................................. 12
    4.5.4.9 playPauseButton ................................................................. 12
    4.5.4.10 recordButton ..................................................................... 12
    4.5.4.11 recorder ............................................................................ 12
    4.5.4.12 saveButton ........................................................................ 13

4.6 IDAAppDelegate Class Reference ............................................................. 13
  4.6.1 Detailed Description ................................................................. 13
  4.6.2 Method Documentation ............................................................... 13
    4.6.2.1 applicationDocumentsDirectory ........................................... 13
    4.6.2.2 saveContext ....................................................................... 13
  4.6.3 Property Documentation .............................................................. 14
    4.6.3.1 fetchedResultsController ...................................................... 14
    4.6.3.2 managedObjectContext ......................................................... 14
    4.6.3.3 managedObjectModel ............................................................ 14
    4.6.3.4 persistentStoreCoordinator ............................................... 14
    4.6.3.5 window .............................................................................. 14

4.7 IDAExerciseConfigTVC Class Reference .................................................. 14
  4.7.1 Detailed Description ................................................................. 15
  4.7.2 Method Documentation ............................................................... 15
    4.7.2.1 endTimePickerChanged: ...................................................... 15
4.7.2.2 notificationSwitchChanged: .......................... 15
4.7.2.3 saveWasPressed: ........................................ 15
4.7.2.4 startTimePickerChanged: .............................. 15

4.7.3 Member Data Documentation ................................ 16
4.7.3.1 endOccurrenceTime ...................................... 16
4.7.3.2 exercisesArray ........................................... 16
4.7.3.3 interval .................................................... 16
4.7.3.4 intervalNumber ............................................ 16
4.7.3.5 notificationSwitchSetting ................................. 16
4.7.3.6 numberOfTimesArray ..................................... 16
4.7.3.7 recurrenceArray ........................................... 16
4.7.3.8 startOccurrenceTime .................................... 16

4.7.4 Property Documentation..................................... 16
4.7.4.1 addExerciseCell .......................................... 16
4.7.4.2 audioSession ............................................. 16
4.7.4.3 endOccurrenceCell ....................................... 16
4.7.4.4 endTimeDatePicker ....................................... 16
4.7.4.5 intervalCell ............................................... 16
4.7.4.6 notificationSwitchCell ................................. 16
4.7.4.7 player ..................................................... 17
4.7.4.8 recurrencePickerView .................................. 17
4.7.4.9 startOccurrenceCell .................................... 17
4.7.4.10 startTimeDatePicker ................................... 17

4.8 IDAExerciseConfigTVC() Category Reference ...................... 17
4.9 IDAExerciseHistoryTVC Class Reference .......................... 17
4.9.1 Method Documentation ...................................... 18
4.9.1.1 playWasPressed: ....................................... 18
4.9.1.2 sliderChanged: ......................................... 18

4.9.2 Property Documentation ...................................... 18
4.9.2.1 audioSession ............................................ 18
4.9.2.2 delegate .................................................. 18
4.9.2.3 fetchedResultsController .............................. 18
4.9.2.4 managedObjectContext .................................. 18
4.9.2.5 playbackSlider .......................................... 18
4.9.2.6 player ..................................................... 19
4.9.2.7 playPauseButton ....................................... 19
4.9.2.8 selectedExercise ....................................... 19
4.9.2.9 sliderTimer ............................................... 19

4.10 IDAExerciseHistoryTVC() Category Reference ...................... 19
4.11 <IDAExerciseHistoryTVCDelegate> Protocol Reference ............... 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12</td>
<td>IDAExercisesViewController Class Reference</td>
<td>20</td>
</tr>
<tr>
<td>4.12.1</td>
<td>Detailed Description</td>
<td>20</td>
</tr>
<tr>
<td>4.12.2</td>
<td>Method Documentation</td>
<td>21</td>
</tr>
<tr>
<td>4.12.2.1</td>
<td>beginExercisePressed:</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3</td>
<td>Member Data Documentation</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.1</td>
<td>animationTimer</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.2</td>
<td>exerciseNumber</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.3</td>
<td>exercisesArray</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.4</td>
<td>progressBarsFlipped</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.5</td>
<td>progressTimer</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.6</td>
<td>recordedTmpFile</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.7</td>
<td>secondsTimer</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.8</td>
<td>timeRecorded</td>
<td>21</td>
</tr>
<tr>
<td>4.12.3.9</td>
<td>timeToRecord</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4</td>
<td>Property Documentation</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4.1</td>
<td>audioSession</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4.2</td>
<td>beginExerciseButton</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4.3</td>
<td>configAlertView</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4.4</td>
<td>configureBarButtonItem</td>
<td>21</td>
</tr>
<tr>
<td>4.12.4.5</td>
<td>exerciseCountLabel</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.6</td>
<td>exerciseLabel1</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.7</td>
<td>exerciseLabel2</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.8</td>
<td>exercisesTabBarItem</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.9</td>
<td>exerciseViewDrawer</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.10</td>
<td>historyBarButtonItem</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.11</td>
<td>levelMeter</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.12</td>
<td>managedObjectContext</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.13</td>
<td>player</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.14</td>
<td>progressView</td>
<td>22</td>
</tr>
<tr>
<td>4.12.4.15</td>
<td>recorder</td>
<td>23</td>
</tr>
<tr>
<td>4.12.4.16</td>
<td>tabBar</td>
<td>23</td>
</tr>
<tr>
<td>4.12.4.17</td>
<td>vocalTestTabBarItem</td>
<td>23</td>
</tr>
<tr>
<td>4.13</td>
<td>IDAExercisesViewController() Category Reference</td>
<td>23</td>
</tr>
<tr>
<td>4.14</td>
<td>IDAGraphViewController Class Reference</td>
<td>23</td>
</tr>
<tr>
<td>4.14.1</td>
<td>Detailed Description</td>
<td>23</td>
</tr>
<tr>
<td>4.15</td>
<td>IDAGraphViewController() Category Reference</td>
<td>24</td>
</tr>
<tr>
<td>4.16</td>
<td>IDAInfoViewController Class Reference</td>
<td>24</td>
</tr>
<tr>
<td>4.16.1</td>
<td>Detailed Description</td>
<td>24</td>
</tr>
<tr>
<td>4.17</td>
<td>IDAInfoViewController() Category Reference</td>
<td>24</td>
</tr>
<tr>
<td>4.18</td>
<td>IDAIntervalCell Class Reference</td>
<td>24</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.22.1</td>
<td>Method Documentation</td>
<td>30</td>
</tr>
<tr>
<td>4.22.1.1</td>
<td>editWasPressed:</td>
<td>30</td>
</tr>
<tr>
<td>4.22.1.2</td>
<td>playWasPressed:</td>
<td>30</td>
</tr>
<tr>
<td>4.22.1.3</td>
<td>sliderChanged:</td>
<td>30</td>
</tr>
<tr>
<td>4.22.2</td>
<td>Property Documentation</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.1</td>
<td>audioSession</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.2</td>
<td>buttonEdit</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.3</td>
<td>buttonPlay</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.4</td>
<td>delegate</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.5</td>
<td>labelCPPScore</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.6</td>
<td>labelDateTaken</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.7</td>
<td>labelFundFreq</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.8</td>
<td>labelJitterScore</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.9</td>
<td>labelName</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.10</td>
<td>managedObjectContext</td>
<td>31</td>
</tr>
<tr>
<td>4.22.2.11</td>
<td>notesEditWindow</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.12</td>
<td>originalName</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.13</td>
<td>originalNotes</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.14</td>
<td>player</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.15</td>
<td>sliderAudioProgress</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.16</td>
<td>sliderTimer</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.17</td>
<td>test</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.18</td>
<td>textFieldName</td>
<td>32</td>
</tr>
<tr>
<td>4.22.2.19</td>
<td>textViewNotes</td>
<td>32</td>
</tr>
<tr>
<td>4.23</td>
<td>IDASavedTestDetailViewController() Category Reference</td>
<td>33</td>
</tr>
<tr>
<td>4.24</td>
<td>IDASavedTestsDetailViewControllerDelegate &gt; Protocol Reference</td>
<td>33</td>
</tr>
<tr>
<td>4.25</td>
<td>IDASavedTestsTVC Class Reference</td>
<td>33</td>
</tr>
<tr>
<td>4.25.1</td>
<td>Property Documentation</td>
<td>33</td>
</tr>
<tr>
<td>4.25.1.1</td>
<td>delegate</td>
<td>33</td>
</tr>
<tr>
<td>4.25.1.2</td>
<td>fetchedResultsController</td>
<td>34</td>
</tr>
<tr>
<td>4.25.1.3</td>
<td>managedObjectContext</td>
<td>34</td>
</tr>
<tr>
<td>4.25.1.4</td>
<td>selectedTest</td>
<td>34</td>
</tr>
<tr>
<td>4.26</td>
<td>IDASavedTestsTVC() Category Reference</td>
<td>34</td>
</tr>
<tr>
<td>4.27</td>
<td>IDASavedTestsTVCDelegate &gt; Protocol Reference</td>
<td>34</td>
</tr>
<tr>
<td>4.28</td>
<td>IDAVocalTestViewController Class Reference</td>
<td>34</td>
</tr>
<tr>
<td>4.28.1</td>
<td>Detailed Description</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2</td>
<td>Method Documentation</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.1</td>
<td>beginVoiceTestPressed:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.2</td>
<td>dismissResultsViewPressed:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.3</td>
<td>playWasPressed:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.4</td>
<td>saveWasPressed:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.5</td>
<td>sliderChanged:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.6</td>
<td>typeCancelSelected:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.7</td>
<td>typeSpeechSelected:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.2.8</td>
<td>typeVowelSelected:</td>
<td>36</td>
</tr>
<tr>
<td>4.28.3</td>
<td>Member Data Documentation</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.1</td>
<td>audioProgressTimer</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.2</td>
<td>cppStore</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.3</td>
<td>error</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.4</td>
<td>fundamentalFrequency</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.5</td>
<td>jitterStore</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.6</td>
<td>progressTimer</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.7</td>
<td>recordedTmpFile</td>
<td>37</td>
</tr>
<tr>
<td>4.28.3.8</td>
<td>TESTURL</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4</td>
<td>Property Documentation</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.1</td>
<td>audioSession</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.2</td>
<td>beginVoiceTest</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.3</td>
<td>buttonInfoDisclosure</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.4</td>
<td>buttonPlay</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.5</td>
<td>buttonSave</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.6</td>
<td>cancelButton</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.7</td>
<td>exercisesTabBarItem</td>
<td>37</td>
</tr>
<tr>
<td>4.28.4.8</td>
<td>isCalledFromNotification</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.9</td>
<td>labelStatusLine1</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.10</td>
<td>labelStatusLine2</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.11</td>
<td>labelStatusLine3</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.12</td>
<td>labelVoiceTest1</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.13</td>
<td>labelVoiceTest2</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.14</td>
<td>levelMeter</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.15</td>
<td>managedObjectContext</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.16</td>
<td>player</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.17</td>
<td>progressVoiceTest</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.18</td>
<td>recorder</td>
<td>38</td>
</tr>
<tr>
<td>4.28.4.19</td>
<td>savedTestsNavButton</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.20</td>
<td>sliderAudioProgress</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.21</td>
<td>speechButton</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.22</td>
<td>tabBar</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.23</td>
<td>testResultsView</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.24</td>
<td>vocalTestTabBarItem</td>
<td>39</td>
</tr>
<tr>
<td>4.28.4.25</td>
<td>voiceTestView</td>
<td>39</td>
</tr>
</tbody>
</table>
4.28.4.26 vowelButton ........................................ 39
4.29 IDAVocalTestViewController() Category Reference ........................................ 39
4.30 UIView(FindFirstResponder) Category Reference ........................................ 40
  4.30.1 Detailed Description ........................................ 40
  4.30.2 Method Documentation ........................................ 40
    4.30.2.1 findFirstResponder ........................................ 40
4.31 Usage Class Reference ........................................ 40
  4.31.1 Detailed Description ........................................ 40
  4.31.2 Property Documentation ........................................ 41
    4.31.2.1 function ........................................ 41
    4.31.2.2 timestamp ........................................ 41
4.32 VocalTests Class Reference ........................................ 41
  4.32.1 Detailed Description ........................................ 41
  4.32.2 Property Documentation ........................................ 41
    4.32.2.1 cppScore ........................................ 41
    4.32.2.2 dateTaken ........................................ 42
    4.32.2.3 fundFreq ........................................ 42
    4.32.2.4 jitterScore ........................................ 42
    4.32.2.5 name ........................................ 42
    4.32.2.6 notes ........................................ 42
    4.32.2.7 soundFilePath ........................................ 42
    4.32.2.8 testType ........................................ 42

5 File Documentation ........................................ 43
  5.1 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/New-VocalTest/NewVocalTest/CoreDataTableViewController.h File Reference ........................................ 43
  5.2 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/New-VocalTest/NewVocalTest/CoreDataTableViewController.m File Reference ........................................ 43
  5.3 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/New-VocalTest/NewVocalTest/Exercises.h File Reference ........................................ 43
  5.4 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/New-VocalTest/NewVocalTest/Exercises.m File Reference ........................................ 44
  5.5 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/New-VocalTest/NewVocalTest/ftt.hpp File Reference ........................................ 44
  5.5.1 Macro Definition Documentation ........................................ 44
    5.5.1.1 TWO_PI ........................................ 44
  5.5.2 Function Documentation ........................................ 44
    5.5.2.1 fft ........................................ 44
    5.5.2.2 fft_rec ........................................ 44
    5.5.2.3 ifft ........................................ 44
  5.5.3 Variable Documentation ........................................ 44
    5.5.3.1 N_master ........................................ 44
<table>
<thead>
<tr>
<th>Version</th>
<th>File Name</th>
<th>Reference Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.h</td>
<td>44</td>
</tr>
<tr>
<td>5.7</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.m</td>
<td>45</td>
</tr>
<tr>
<td>5.8</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.h</td>
<td>45</td>
</tr>
<tr>
<td>5.9</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.m</td>
<td>45</td>
</tr>
<tr>
<td>5.10</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAppDelegate.h</td>
<td>45</td>
</tr>
<tr>
<td>5.11</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAppDelegate.m</td>
<td>46</td>
</tr>
<tr>
<td>5.12</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExerciseConfigTVC.h</td>
<td>46</td>
</tr>
<tr>
<td>5.13</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExerciseConfigTVC.m</td>
<td>46</td>
</tr>
<tr>
<td>5.14</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExerciseHistoryTVC.h</td>
<td>46</td>
</tr>
<tr>
<td>5.15</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExerciseHistoryTVC.m</td>
<td>47</td>
</tr>
<tr>
<td>5.16</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExercisesViewController.h</td>
<td>47</td>
</tr>
<tr>
<td>5.17</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAEExercisesViewController.m</td>
<td>47</td>
</tr>
<tr>
<td>5.18</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAGraphViewController.h</td>
<td>47</td>
</tr>
<tr>
<td>5.19</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAGraphViewController.m</td>
<td>47</td>
</tr>
<tr>
<td>5.20</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.h</td>
<td>48</td>
</tr>
<tr>
<td>5.21</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.m</td>
<td>48</td>
</tr>
<tr>
<td>5.22</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAIntervalCell.h</td>
<td>48</td>
</tr>
<tr>
<td>5.23</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAIntervalCell.m</td>
<td>48</td>
</tr>
<tr>
<td>5.24</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDALevelMeterView.h</td>
<td>48</td>
</tr>
<tr>
<td>5.25</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDALevelMeterView.m</td>
<td>48</td>
</tr>
<tr>
<td>5.26</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDANotificationSwitchCell.h</td>
<td>49</td>
</tr>
<tr>
<td>5.27</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDANotificationSwitchCell.m</td>
<td>49</td>
</tr>
<tr>
<td>5.28</td>
<td>/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAOccurrenceCell.h</td>
<td>49</td>
</tr>
</tbody>
</table>
CONTENTS

5.29/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDAOccurrenceCell.m File Reference .......................................................... 49

5.30/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDASavedTestDetailViewController.h File Reference .................................................. 50

5.31/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDASavedTestDetailViewController.m File Reference .................................................. 50

5.32/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDASavedTestsTVC.h File Reference .................................................. 50

5.33/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDASavedTestsTVC.m File Reference .................................................. 50

5.34/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDAVocalTestViewController.h File Reference .................................................. 51

5.35/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/IDAVocalTestViewController.m File Reference .................................................. 51

5.35.1 Variable Documentation ........................................................................... 51

5.35.1.1 testCount ......................................................................................... 51

5.35.1.2 typeOfTest ....................................................................................... 51

5.36/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/JitterCalc_mod.mm File Reference .................................................. 51

5.36.1 Macro Definition Documentation ............................................................. 52

5.36.1.1 JITTERCALC_MOD_H ................................................................. 52

5.36.2 Function Documentation ......................................................................... 52

5.36.2.1 AnalyzeSoundFile .......................................................................... 52

5.36.2.2 CPPCalc ......................................................................................... 52

5.36.2.3 JitterCalc ......................................................................................... 52

5.36.2.4 LinLeastSquares ............................................................................. 52

5.37/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/main.m File Reference .................................................. 52

5.37.1 Function Documentation ......................................................................... 53

5.37.1.1 main ............................................................................................... 53


5.39/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/Usage.m File Reference .......................................................... 53


5.41/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/NewVocalTest/VocalTests.m File Reference .......................................................... 53

5.42/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/UIView+FindFirstResponder.h File Reference .......................................................... 54

5.43/Users/robertpfister/Dropbox/Documents/Xcode_Projects/Eva's_Projects/VocalTestVersions/NewVocalTest/UIView+FindFirstResponder.m File Reference .......................................................... 54

Index 54

Generated on Mon Apr 8 2013 21:53:11 for iOS-based Voice Analysis Application for Speech Therapy by Doxygen
Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<AVAudioPlayerDelegate>
  IDAAddExerciseViewController ........................................ 10
  IDAEExerciseConfigTVC ........................................... 14
  IDAEExerciseHistoryTVC ........................................... 17
  IDAEExercisesViewController ..................................... 20
  IDASavedTestDetailViewController .................................. 29
  IDAVocalTestViewController ....................................... 34

<AVAudioRecorderDelegate>
  IDAAddExerciseViewController ........................................ 10
  IDAEExercisesViewController ..................................... 20
  IDALevelMeterView ............................................... 25
  IDAVocalTestViewController ....................................... 34

CoreDataTableViewController() ...................................... 8
IDAEExerciseConfigTVC() ........................................... 17
IDAEExerciseHistoryTVC() ........................................... 19
IDAEExercisesViewController() ..................................... 23
IDAGraphViewController() .......................................... 24
IDAINfoViewController() ........................................... 24
IDASavedTestDetailViewController() .................................. 33
IDASavedTestsTVC() .................................................. 34
IDAVocalTestViewController() ....................................... 39

<NSThreadDelegate>
  IDAAddExerciseViewController ........................................ 10
  IDAEExerciseConfigTVC() ........................................... 17
  IDAEExerciseHistoryTVC() ........................................... 19
  IDAEExercisesViewController() ..................................... 23
  IDAGraphViewController() .......................................... 24
  IDAINfoViewController() ........................................... 24
  IDASavedTestDetailViewController() .................................. 33
  IDASavedTestsTVC() .................................................. 34
  IDAVocalTestViewController() ....................................... 39

<NSFetchedResultsControllerDelegate>
  CoreDataTableViewController() ...................................... 7
  IDAEExerciseConfigTVC() ........................................... 17
  IDAEExerciseHistoryTVC ........................................... 17
  IDASavedTestsTVC .................................................. 33
  IDAAppDelegate ..................................................... 13

<NSObject>

NSManagedObject
  Exercises ............................................................ 8
  Usage ............................................................... 40
  VocalTests .......................................................... 41

<NSFetchedResultsControllerDelegate>
  IDAEExerciseConfigTVC ........................................... 17
  IDAEExerciseHistoryTVC ........................................... 17
  IDASavedTestsTVC .................................................. 33
  IDASavedTestsTVC .................................................. 33

<NSFetchedResultsControllerDelegate>
  IDASavedTestsDetailViewControllerDelegate ................................ 33
  IDASavedTestsTVC .................................................. 33

<NSFetchedResultsControllerDelegate>
  IDASavedTestsTVC .................................................. 34
  IDASavedTestsTVC .................................................. 34
  IDAVocalTestViewController ....................................... 34
Hierarchical Index

UIApplicationDelegate

IDAAppDelegate .................................................. 13

UIResponder

IDAAppDelegate .................................................. 13

UITabBarDelegate

IDAViewPagerController ............................................ 34

UITableViewCell

IDAAddExerciseCell ............................................. 9
IDAINtervalCell .................................................. 24
IDANotificationSwitchCell ...................................... 28
IDAOccurrenceCell ............................................... 29

UITableViewController

CoreDataTableViewController ...................................... 7
IDAExerciseConfigTVC .......................................... 14

UITableViewDataSource

IDAExerciseConfigTVC .......................................... 14

UITableViewDelegate

IDAExerciseConfigTVC .......................................... 14

UITextFieldDelegate

IDAAddExerciseViewController .................................... 10
IDAInfoViewController .......................................... 24

UITextViewDelegate

IDAInfoViewController .......................................... 24

UIView

IDALevelMeterView ............................................. 25

UIView(FindFirstResponder) ..................................... 40

UIViewController

IDAAddExerciseViewController .................................... 10
IDAExercisesViewController .................................... 20
IDAGraphViewController ......................................... 23
IDAInfoViewController .......................................... 24
IDASavedTestDetailViewController ............................. 29
IDAViewPagerController ......................................... 34
Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<table>
<thead>
<tr>
<th>Class Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreDataTableViewController</td>
<td>7</td>
</tr>
<tr>
<td>CoreDataTableViewController()</td>
<td>8</td>
</tr>
<tr>
<td>Exercises</td>
<td>8</td>
</tr>
<tr>
<td>IDAAddExerciseCell</td>
<td>9</td>
</tr>
<tr>
<td>IDAAddExerciseViewController</td>
<td>10</td>
</tr>
<tr>
<td>IDAAppDelegate</td>
<td>13</td>
</tr>
<tr>
<td>IDAExerciseConfigTVC</td>
<td>14</td>
</tr>
<tr>
<td>IDAExerciseConfigTVC()</td>
<td>17</td>
</tr>
<tr>
<td>IDAExerciseHistoryTVC</td>
<td>17</td>
</tr>
<tr>
<td>IDAExerciseHistoryTVC()</td>
<td>19</td>
</tr>
<tr>
<td>&lt;IDAExerciseHistoryTVCDelegate&gt;</td>
<td>19</td>
</tr>
<tr>
<td>IDAExercisesViewController</td>
<td>20</td>
</tr>
<tr>
<td>IDAExercisesViewController()</td>
<td>23</td>
</tr>
<tr>
<td>IDAGraphViewController</td>
<td>23</td>
</tr>
<tr>
<td>IDAGraphViewController()</td>
<td>24</td>
</tr>
<tr>
<td>IDAInfoViewController</td>
<td>24</td>
</tr>
<tr>
<td>IDAInfoViewController()</td>
<td>24</td>
</tr>
<tr>
<td>IDAIntervalCell</td>
<td>24</td>
</tr>
<tr>
<td>IDALevelMeterView</td>
<td>25</td>
</tr>
<tr>
<td>IDANotificationSwitchCell</td>
<td>28</td>
</tr>
<tr>
<td>IDAOccurrenceCell</td>
<td>29</td>
</tr>
<tr>
<td>IDASavedTestDetailViewController</td>
<td>29</td>
</tr>
<tr>
<td>IDASavedTestDetailViewController()</td>
<td>33</td>
</tr>
<tr>
<td>&lt;IDASavedTestsDetailViewControllerDelegate&gt;</td>
<td>33</td>
</tr>
<tr>
<td>IDASavedTestsTVC</td>
<td>33</td>
</tr>
<tr>
<td>IDASavedTestsTVC()</td>
<td>34</td>
</tr>
<tr>
<td>&lt;IDASavedTestsTVCDelegate&gt;</td>
<td>34</td>
</tr>
<tr>
<td>IDAVocalTestViewController</td>
<td>34</td>
</tr>
<tr>
<td>IDAVocalTestViewController()</td>
<td>39</td>
</tr>
<tr>
<td>UIView(FindFirstResponder)</td>
<td>40</td>
</tr>
<tr>
<td>Usage</td>
<td>40</td>
</tr>
<tr>
<td>VocalTests</td>
<td>41</td>
</tr>
</tbody>
</table>
Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/UIKit+FirstResponder.h ................................................... 54
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/UIKit+FirstResponder.m ................................................ 54
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/CoreDataTableViewController.h .................................. 43
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/CoreDataTableViewController.m .................. 43
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Exercises.h .................................................. 43
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Exercises.m ................................................. 44
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/ftf.hpp ...................................................... 44
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.h ......................................... 44
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.m ...................................... 45
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.h .................. 45
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.m ............ 45
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExercisesViewController.h ......................... 45
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExercisesViewController.m ................... 46
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExerciseConfigTVC.h .................................. 46
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExerciseConfigTVC.m ....................... 46
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExerciseHistoryTVC.h .................................. 46
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExerciseHistoryTVC.m ......................... 47
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExercisesViewController.h ..................... 47
/Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAExercisesViewController.m ....................... 47
Chapter 4

Class Documentation

4.1 CoreDataTableViewController Class Reference

#import <CoreDataTableViewController.h>

Inheritance diagram for CoreDataTableViewController:

```
UITableViewController <NSFetchedResultsControllerDelegate> 
CoreDataTableViewController 
IDAEExerciseHistoryTVC IDASavedTestsTVC
```

Instance Methods

- (void) performFetch

Properties

- NSFetchedResultsController * fetchedResultsController
- BOOL suspendAutomaticTrackingOfChangesInManagedObjectContext
- BOOL debug

4.1.1 Method Documentation

4.1.1.1 - (void) performFetch

4.1.2 Property Documentation

4.1.2.1 - (BOOL) debug [read],[write],[atomic]

4.1.2.2 - (NSFetchedResultsController *) fetchedResultsController [read],[write],[nonatomic],[strong]

4.1.2.3 - (BOOL) suspendAutomaticTrackingOfChangesInManagedObjectContext [read],[write],[nonatomic],[assign]

The documentation for this class was generated from the following files:
4.2 CoreDataTableViewController() Category Reference

Properties

- BOOL beganUpdates

4.2.1 Property Documentation

4.2.1.1 -(BOOL) beganUpdates [read],[write],[nonatomic],[assign]

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/CoreDataTableViewController.m

4.3 Exercises Class Reference

#import <Exercises.h>

Inheritance diagram for Exercises:

```
NSManagedObject

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exercises</td>
</tr>
</tbody>
</table>
```

Properties

- NSDate * dateTaken
- NSString * name
- NSString * note
- NSString * soundFilePath

4.3.1 Detailed Description

This object is used to store an exercise into the core data model. We store the name and note that the user gives the exercise (this function is not currently operational, but will work very similar to the VocalTests counterparts of the same name in a future release. They are included here to ensure that the user does not lose his or her data due to versioning of the xcdatamodeld object and the sqlite store. A path to the sound file is also included to allow playback. Every exercise performed is saved to the Core Data store, although they may be deleted from the IDAExerciseHistoryTVC.
4.3.2 Property Documentation

4.3.2.1 - (NSDate) dateTaken  [read],[write],[nonatomic],[retain]
A date/time stamp which represents the when the Exercise was created and performed.

4.3.2.2 - (NSString) name  [read],[write],[nonatomic],[retain]
The name given to the exercise by the user. NOTE: this function is currently not implemented. See explanation in the Brief Description of Exercises.

4.3.2.3 - (NSString) note  [read],[write],[nonatomic],[retain]
The notes taken about the exercise by the user. It is recommended the user use this section to mark the conditions the exercise was performed in, if extenuating circumstances exist. NOTE: this function is currently not implemented. See explanation in the Brief Description of Exercises.

4.3.2.4 - (NSString) soundFilePath  [read],[write],[nonatomic],[retain]
This NSString is the path to the VocalTest soundfile which was analyzed by the application. It can be replayed by the user at any time from the IDASavedTestDetailViewController. The file is deleted when the VocalTest object is deleted from the Core Data store.

The documentation for this class was generated from the following file:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/Exercises.h

4.4 IDAAddExerciseCell Class Reference

#import <IDAAddExerciseCell.h>

Inheritance diagram for IDAAddExerciseCell:

```
UITableViewCell

| IDAAddExerciseCell
```

Properties

• IBOutlet UIButton * addExerciseButton
• IBOutlet UILabel * addExerciseLabel

4.4.1 Detailed Description

This subclass of UITableViewCell is presented at the bottom of the list of exercise instruction items in the IDA-ExerciseConfigTVC, and contains an "add item" button, along with a label.
4.4.2 Property Documentation

4.4.2.1 - (IBOutlet UIButton) addExerciseButton [read],[write],[nonatomic],[weak]
This UIButton is used to signify to the user that clicking this cell will enable him or her to add another exercise instruction to the list. The button reacts the same as clicking the cell itself, and is only used so that the universal add item identifier for iOS appears and makes the user more confident in what he or she is doing.

4.4.2.2 - (IBOutlet UILabel) addExerciseLabel [read],[write],[nonatomic],[weak]
This label contains text telling the user that by selecting this cell, he or she will be able to add an exercise instruction. The documentation for this class was generated from the following file:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAAddExerciseCell.h

4.5 IDAAddExerciseViewController Class Reference

#import <IDAAddExerciseViewController.h>

Instance Methods

• (IBAction) - playPausePressed:
• (IBAction) - recordPressed:
• (IBAction) - savePressed:
• (IBAction) - sliderChanged:
• (IBAction) - cancelPressed:

Protected Attributes

• NSURL * recordedTmpFile
• NSTimer * audioProgressTimer
• NSTimer * recordLightTimer

Properties

• IBOutlet IDALevelMeterView * levelMeter
• IBOutlet UITextField * exerciseNameTextField
• IBOutlet UITextField * exerciseNoteTextField
• IBOutlet UIButton * playPauseButton
• IBOutlet UIButton * recordButton
• IBOutlet UIButton * saveButton
• IBOutlet UISlider * playbackSlider
• IBOutlet UIBarButtonItem * cancelButtonTitle
• IBOutlet UIImageView * lightUpImageView
4.5 IDAAddExerciseViewController Class Reference

- AVAudioSession * audioSession
- AVAudioPlayer * player
- AVAudioRecorder * recorder

4.5.1 Detailed Description

This subclass of UIViewController handles the addition of exercise instruction objects into the IDAEExerciseConfigTVC. This UIView contains an IDALevelMeterView for visualization of recordings, as well as UITextField objects for naming and noting an exercise. The play, record and save buttons handle basic playback, recording, and adding the items to the NSUserDefaults.

4.5.2 Method Documentation

4.5.2.1 - (IBAction) cancelPressed: (id) sender

This method is called from the cancelBarButtonItem and dismisses the UIViewController without adding the exercise instruction to the NSUserDefaults.

4.5.2.2 - (IBAction) playPausePressed: (id) sender

This method is called from the playPauseButton object and handles the play and pause of the current audio file.

4.5.2.3 - (IBAction) recordPressed: (id) sender

This method is called from the recordButton object and handles start and stop of audio recording.

4.5.2.4 - (IBAction) savePressed: (id) sender

This method is called by the saveButton object and adds the new exercise instruction to the NSUserDefaults so that it shows in the list within IDAExerciseConfigTVC.

4.5.2.5 - (IBAction) sliderChanged: (id) sender

This method is fired whenever the user changes the sliderAudioProgress UISlider. It sets the playback time for the audio file to the position selected in the UISlider.

4.5.3 Member Data Documentation

4.5.3.1 - (NSTimer *) audioProgressTimer [protected]
4.5.3.2 - (NSURL *) recordedTmpFile [protected]
4.5.3.3 - (NSTimer *) recordLightTimer [protected]

4.5.4 Property Documentation

4.5.4.1 - (AVAudioSession *) audioSession [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback of the exercise instruction file from within this UIViewController. It sets the audio to be played through the external speakers rather than the earpiece.
4.5.4.2 - (IBOutlet UIBarButtonItem) cancelBarButtonItem [read],[write],[nonatomic],[weak]

This UIBarButtonItem is located in the UINavigationController for this UIViewController, and cancels the creation of a new exercise instruction. It also dismisses the UIViewController.

4.5.4.3 - (IBOutlet UITextField) exerciseNameTextField [read],[write],[nonatomic],[weak]

This UITextField allows the user to name the exercise, to make it distinguishable later in a list of other exercises. This text also shows to the user during performance of an exercise.

4.5.4.4 - (IBOutlet UITextField) exerciseNoteTextField [read],[write],[nonatomic],[weak]

This UITextField allows the user to make notes on the exercise file. These notes show at subtitles in the IDA-ExerciseConfigTVC.

4.5.4.5 - (IBOutlet IDALevelMeterView) levelMeter [read],[write],[nonatomic],[weak]

IDALevelMeterView is a custom class for the visual representation of the volume of a recording. This is used during the recording of the voice test to let the user know if he or she is being loud enough or too loud, both of which can cause problems during the analysis of the sound file.

4.5.4.6 - (IBOutlet UIImageView) lightUpImageView [read],[write],[nonatomic],[weak]

This UIImageView overlays the recordButton object. When that button is pressed, this image's alpha value changes repeatedly to give the effect that the recordButton is lighting up.

4.5.4.7 - (IBOutlet UISlider) playbackSlider [read],[write],[nonatomic],[weak]

The UISlider shows how much of the recorded file has been played back. It can be scrubbed to change the current playback location within the current sound file.

4.5.4.8 - (AVAudioPlayer) player [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of exercise instruction audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.5.4.9 - (IBOutlet UIButton) playPauseButton [read],[write],[nonatomic],[weak]

This UIBarButtonItem handles playing and pausing of the most recently recorded audio file. It fires the method playPausePressed.

4.5.4.10 - (IBOutlet UIButton) recordButton [read],[write],[nonatomic],[weak]

This UIButton calls the method recordPressed and handles the ability to begin and end a recording. The recordButton flashes while recording is in progress.

4.5.4.11 - (AVAudioRecorder) recorder [read],[write],[nonatomic],[strong]

This AVAudioRecorder handles recording of exercise instruction audio, and makes calls to methods pertaining to the recording of audio, such as audioRecorderDidFinishRecording.
4.6 IDAAppDelegate Class Reference

#import <IDAAppDelegate.h>

Inheritance diagram for IDAAppDelegate:

```
UIResponder <UIApplicationDelegate> <NSFetchedResultsControllerDelegate>
```

Instance Methods

- (void) saveContext
- (NSURL *) applicationDocumentsDirectory

Properties

- UIWindow * window
- NSManagedObjectContext * managedObjectContext
- NSManagedObjectModel * managedObjectModel
- NSPersistentStoreCoordinator * persistentStoreCoordinator
- NSFetchedResultsController * fetchedResultsController

4.6.1 Detailed Description

This file handles all backgrounding, return from the background, launching, interruptions, and changing of state for the application. It also handles the Exercises Local Notifications, directing the user to the Exercises screen if he or she enters the app from the Exercise reminder notification object.

4.6.2 Method Documentation

4.6.2.1 - (NSURL *) applicationDocumentsDirectory

Creates a direct reference to the documents directory for the application, shortening codeblocks which access this directory.

4.6.2.2 - (void) saveContext

Saves the current state of the managedObjectContext out to the data store (In this case, an SQLite data store).
4.6.3  Property Documentation

4.6.3.1 - (NSFetchedResultsController *) fetchedResultsController [read],[write],[nonatomic],[strong]
Handles the querying of the managedObjectContext for data. Sorting, filtering, etc are done here.

4.6.3.2 - (NSManagedObjectContext *) managedObjectContext [read],[nonatomic],[strong]
Manages a collection of managed objects loaded from the data store (SQLite).

4.6.3.3 - (NSManagedObjectModel *) managedObjectModel [read],[nonatomic],[strong]
A collection of entity descriptions.

4.6.3.4 - (NSPersistentStoreCoordinator *) persistentStoreCoordinator [read],[nonatomic],[strong]
References the managedObjectModel. This is the central object in the CoreData stack

4.6.3.5 - (UIWindow *) window [read],[write],[nonatomic],[strong]
The main window the app is contained in.

The documentation for this class was generated from the following files:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAAppDelegate.h
• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAAppDelegate.m

4.7  IDAExerciseConfigTVC Class Reference

#import <IDAExerciseConfigTVC.h>

Inheritance diagram for IDAExerciseConfigTVC:

```
UITableViewController <-> UITableViewDataSource <-> UITableViewDelegate <-> AVAudioPlayerDelegate

| IDAExerciseConfigTVC |
```

Instance Methods

• (IBAction) - startTimePickerChanged:
• (IBAction) - endTimePickerChanged:
• (IBAction) - notificationSwitchChanged:
• (IBAction) - saveWasPressed:

Protected Attributes

• NS MUTABLE ARRAY * numberOfTimesArray
• NS MUTABLE ARRAY * recurrenceArray
4.7 IDAExerciseConfigTVC Class Reference

- NSMutableArray * exercisesArray
- NSNumber * notificationSwitchSetting
- NSDate * startOccurrenceTime
- NSDate * endOccurrenceTime
- NSNumber * intervalNumber
- NSString * interval

Properties

- IBOutlet UIPickerView * recurrencePickerView
- IBOutlet UIDatePicker * startTimeDatePicker
- IBOutlet UIDatePicker * endTimeDatePicker
- IDANotificationSwitchCell * notificationSwitchCell
- IDAOccurrenceCell * startOccurrenceCell
- IDAOccurrenceCell * endOccurrenceCell
- IDAIntervalCell * intervalCell
- IDAAddExerciseCell * addExerciseCell
- AVAudioSession * audioSession
- AVAudioPlayer * player

4.7.1 Detailed Description

This subclass of UITableViewController shows a configuration screen for the Exercise portion of the application. It allows the user to turn on and off local notifications, set what times the notifications should occur between and set the number of notifications he or she would like to receive.

4.7.2 Method Documentation

4.7.2.1 - (IBAction) endTimePickerChanged: (id) sender

This method is called when the value in the endTimeDatePicker changes. It changes both the variable to be stored for this value and the value shown in the endOccurrenceCell.

4.7.2.2 - (IBAction) notificationSwitchChanged: (id) sender

This method fires whenever the switch in the notificationSwitchCell is changed. It simply updates the boolean value representing this object, so that it may be stored if the user selects the save button.

4.7.2.3 - (IBAction) saveWasPressed: (id) sender

This method is called by the button in the upper right hand corner of the UINavigationController for this UITableViewController. It saves all values off to the NSUserDefaults and schedules all of the local notifications.

4.7.2.4 - (IBAction) startTimePickerChanged: (id) sender

This method is called when the value in the startTimeDatePicker changes. It changes both the variable to be stored for this value and the value shown in the startOccurrenceCell.
4.7.3 Member Data Documentation

4.7.3.1 - (NSDate) endOccurrenceTime  [protected]
4.7.3.2 - (NSMutableArray) exercisesArray  [protected]
4.7.3.3 - (NSString) interval  [protected]
4.7.3.4 - (NSNumber) intervalNumber  [protected]
4.7.3.5 - (NSNumber) notificationSwitchSetting  [protected]
4.7.3.6 - (NSMutableArray) numberOfTimesArray  [protected]
4.7.3.7 - (NSMutableArray) recurrenceArray  [protected]
4.7.3.8 - (NSDate) startOccurrenceTime  [protected]

4.7.4 Property Documentation

4.7.4.1 - (IDAAddExerciseCell) addExerciseCell  [read],[write],[nonatomic],[strong]

This custom UICell shows at the bottom of the list of recorded exercise instructions and allows the addition of new exercises. It causes the segue to IDAAddExerciseViewController, allowing the naming and recording of a new exercise instruction.

4.7.4.2 - (AVAudioSession) audioSession  [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback of exercise instruction files from within this UITableView-Controller. It sets the audio to be played through the external speakers rather than the earpiece. The audio is played by simply tapping one of the cells containing an exercise instruction.

4.7.4.3 - (IDAIntervalCell) intervalCell  [read],[write],[nonatomic],[strong]

This custom UICell shows the values chosen in the recurrencePickerView. It is how often notifications will occur.

4.7.4.4 - (IBOutlet UIDatePicker) endTimeDatePicker  [read],[write],[nonatomic],[strong]

This UIDatePicker is the input view for the endOccurrenceCell. It is used to set the time of day no reminders should occur after. There is no date associated with the time.

4.7.4.5 - (IDANotificationSwitchCell) notificationSwitchCell  [read],[write],[nonatomic],[strong]

This custom UICell contains a switch for turning on or off local notifications pertaining to the performance of exercises.
4.7.4.7 - (AVAudioPlayer*) player  [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of exercise instruction audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.7.4.8 - (IBOutlet UIPickerView*) recurrencePickerView  [read],[write],[nonatomic],[weak]

This UIPickerView is the input view for the intervalCell. It is a two part picker with the number of times in the left hand column and the period in the right hand column.

4.7.4.9 - (IDAOccurrenceCell*) startOccurrenceCell  [read],[write],[nonatomic],[strong]

This custom UICell shows the value chosen in the startTimeDatePicker.

4.7.4.10 - (IBOutlet UIDatePicker*) startTimeDatePicker  [read],[write],[nonatomic],[strong]

This UIDatePicker is the input view for the startOccurrenceCell. It is used to set the time of day no reminders should occur before. There is no date associated with the time.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseConfigTVC.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseConfigTVC.m

4.8 IDAExerciseConfigTVC() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseConfigTVC.m

4.9 IDAExerciseHistoryTVC Class Reference

#import <IDAExerciseHistoryTVC.h>

Inheritance diagram for IDAExerciseHistoryTVC:

![Inheritance Diagram](image)

Instance Methods

- (IBAction) - playWasPressed:
- (IBAction) - sliderChanged:
Properties

- `id< IDAExerciseHistoryTVCDelegate > delegate`
- `NSManagedObjectContext * managedObjectContext`
- `NSFetchedResultsController * fetchedResultsController`
- `UISlider * playbackSlider`
- `UIBarButtonItem * playPauseButton`
- `AVAudioSession * audioSession`
- `AVAudioPlayer * player`
- `NSTimer * sliderTimer`
- `Exercises * selectedExercise`

### 4.9.1 Method Documentation

#### 4.9.1.1 - (IBAction) playWasPressed: (id) sender

This method handles the pause and playback of exercise audio from the `IDAExerciseHistoryTVC`. It uses audioSession and player to playback the WAV files. NOTE: because this method is called from a UIBarButtonItem in a toolbar not defined in the Interface Builder file, this method is linked to the button programmatically.

#### 4.9.1.2 - (IBAction) sliderChanged: (UISlider *) sender

This method is fired whenever the user changes the playbackSlider UISlider. It sets the playback time for the audio file to the position selected in the UISlider. NOTE: because this method is called from a UISlider within a UIBarButtonItem in a toolbar not defined in the Interface Builder file, this method is linked to the button programmatically.

### 4.9.2 Property Documentation

#### 4.9.2.1 - (AVAudioSession *) audioSession [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback of exercise files from within this UITableViewController. It sets the audio to be played through the external speakers rather than the earpiece.

#### 4.9.2.2 - (id< IDAExerciseHistoryTVCDelegate >) delegate [read],[write],[nonatomic],[weak]

This object represents the delegate for the `IDAExerciseHistoryTVC`.

#### 4.9.2.3 - (NSFetchedResultsController *) fetchedResultsController [read],[write],[nonatomic],[strong]

This fetched results controller handles fetching and sorting the `VocalTests` objects to be displayed in the `UITableView`.

#### 4.9.2.4 - (NSManagedObjectContext *) managedObjectContext [read],[write],[nonatomic],[strong]

Manages a collection of managed objects loaded from the data store (SQLite).

#### 4.9.2.5 - (UISlider *) playbackSlider [read],[write],[nonatomic],[strong]

This UISlider shows how much of the selected `Exercises` object's audio file has been played back. It can be scrubbed to change the current playback location within the current sound file. This object is presented within the toolbar, which is a part of the UINavigationController for this `UITableViewController`. It calls the method sliderChanged when the user moves the knob of the UISlider to a new position.
4.9.2.6 - (AVAudioPlayer) player  [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of exercise audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.9.2.7 - (UIButton) playPauseButton  [read],[write],[nonatomic],[strong]

This UIButton handles playing and pausing of the currently selected audio file. It is presented within the toolbar, which is part of the UINavigationController for this UITableViewController, and fires the method playWasPressed.

4.9.2.8 - (Exercises) selectedExercise  [read],[write],[nonatomic],[strong]

This Exercises object represents the currently selected object, denoted by a blue cell in the UITableView.

4.9.2.9 - (NSTimer) sliderTimer  [read],[write],[nonatomic],[strong]

This NSTimer updates the current position of the playbackSlider UISlider object to match the current playback point of the audio, giving a visual representation of audio playback.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseHistoryTVC.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseHistoryTVC.m

4.10 IDAExerciseHistoryTVC() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseHistoryTVC.m

4.11 <IDAExerciseHistoryTVCDelegate> Protocol Reference

#import <IDAExerciseHistoryTVC.h>

Inheritance diagram for <IDAExerciseHistoryTVCDelegate>:

```
<NSObject>
<IDAExerciseHistoryTVCDelegate>
IDAExerciseHistoryTVC
```

The documentation for this protocol was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExerciseHistoryTVC.h
4.12 IDAExercisesViewController Class Reference

#import <IDAExercisesViewController.h>

Inheritance diagram for IDAExercisesViewController:

```
                                   IDAExercisesViewController
                                     │                        │
                                     │                        ▼
                                  <AVAudioPlayerDelegate>  <AVAudioRecorderDelegate>
                                     │                        │
                                  UIViewController               ₪
                                     │                        │
                                    ₪                          ₪
                                  IDAExercisesViewController
```

Instance Methods

- (IBAction) beginExercisePressed:

Protected Attributes

- NSURL *recordedTmpFile
- NSMutableArray *exercisesArray
- NSTimer *progressTimer
- NSTimer *secondsTimer
- NSTimer *animationTimer
- NSNumber *progressBarsFlipped
- NSNumber *exerciseNumber
- NSNumber *timeToRecord
- NSNumber *timeRecorded

Properties

- NSManagedObjectContext *managedObjectContext
- IBOutlet UITabBar *tabBar
- IBOutlet UITabBarItem *vocalTestTabBarItem
- IBOutlet UITabBarItem *exercisesTabBarItem
- IBOutlet IDALevelMeterView *levelMeter
- IBOutlet UIButton *beginExerciseButton
- IBOutlet UIProgressView *progressView
- IBOutlet UIView *exerciseViewDrawer
- IBOutlet UILabel *exerciseLabel1
- IBOutlet UILabel *exerciseLabel2
- IBOutlet UILabel *exerciseCountLabel
- IBOutlet UIView *configAlertView
- IBOutlet UIBarButtonItem *configureBarButtonItem
- IBOutlet UIBarButtonItem *historyBarButtonItem
- AVAudioSession *audioSession
- AVAudioPlayer *player
- AVAudioRecorder *recorder

4.12.1 Detailed Description

This subclass of UIViewController contains all of the functionality for completion of voice exercises. The audio files which are saved via the IDAExerciseConfigTVC are accessed from this view controller for playback to the user. This view also handles saving the performed exercise off to the core data store for review in the IDAExerciseHistoryTVC. It is a delegate for AVAudioRecorder and AVAudioPlayer.
4.12 IDAExercisesViewController Class Reference

4.12.2 Method Documentation

4.12.2.1 -(IBAction) beginExercisePressed: (id) sender

This method presents the exerciseViewDrawer and begins the playback of exercise instructional audio.

4.12.3 Member Data Documentation

4.12.3.1 -(NSTimer *) animationTimer [protected]
4.12.3.2 -(NSNumber *) exerciseNumber [protected]
4.12.3.3 -(NSMutableArray *) exercisesArray [protected]
4.12.3.4 -(NSNumber *) progressBarsFlipped [protected]
4.12.3.5 -(NSTimer *) progressTimer [protected]
4.12.3.6 -(NSURL *) recordedTmpFile [protected]
4.12.3.7 -(NSTimer *) secondsTimer [protected]
4.12.3.8 -(NSNumber *) timeRecorded [protected]
4.12.3.9 -(NSNumber *) timeToRecord [protected]

4.12.4 Property Documentation

4.12.4.1 -(AVAudioSession *) audioSession [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback of exercise instructional audio and recording the user performing the exercise from within this UIViewController. It sets the audio to be played through the earpiece rather than the external speakers. This is done to make the performance of exercises more private and more like a phone call.

4.12.4.2 -(IBOutlet UIButton *) beginExerciseButton [read],[write],[nonatomic],[weak]

This UIButton presents the exerciseViewDrawer, beginning the process of an exercise for the user. This UIButton fires the method beginExercisePressed.

4.12.4.3 -(IBOutlet UIView *) configAlertView [read],[write],[nonatomic],[weak]

This UIView points the user to the configureBarButtonItem when no exercise instruction audio exists. This is done because the user may be confused as to why the beginExerciseButton is disabled when first downloading the application, or after all of the exercise instruction audio has been deleted via the IDAExerciseConfigTVC.

4.12.4.4 -(IBOutlet UIBarButtonItem *) configureBarButtonItem [read],[write],[nonatomic],[weak]

This UIBarButtonItem performs the segue to the IDAExerciseConfigTVC, allowing the user to configure notifications, and allowing the clinician to record exercise instructional audio.
4.12.4.5 - (IBOutlet UILabel) exerciseCountLabel  [read],[write],[nonatomic],[weak]

This UILabel is within the exerciseViewDrawer and shows the user what exercise out of the total number of exercises he or she is currently performing.

4.12.4.6 - (IBOutlet UILabel) exerciseLabel1  [read],[write],[nonatomic],[weak]

This UILabel is within the exerciseViewDrawer and shows the top line of text while walking a user through an exercise. It lets the user know whether to listen to the exercise audio or repeat what they had just heard.

4.12.4.7 - (IBOutlet UILabel) exerciseLabel2  [read],[write],[nonatomic],[weak]

This UILabel is within the exerciseViewDrawer and shows the bottom line of text while walking a user through an exercise. It shows the user the name given to the exercise audio by the clinician.

4.12.4.8 - (IBOutlet UITabBarItem) exercisesTabBarItem  [read],[write],[nonatomic],[weak]

This UITabBarItem, shown in the right position of the tabBar UITabBar object, performs the segue to present the IDAVocalTestViewController.

4.12.4.9 - (IBOutlet UIView) exerciseViewDrawer  [read],[write],[nonatomic],[weak]

This UIView contains all of the visual elements needed to perform an exercise. The view presents itself from the bottom in a drawer-like fashion, and walks the user through performing an exercise, then exits off the bottom, as if the drawer is closing.

4.12.4.10 - (IBOutlet UIBarButtonItem) historyBarButtonItem  [read],[write],[nonatomic],[weak]

This UIBarButtonItem performs the segue to the IDAExerciseHistoryTVC, revealing previously performed exercises.

4.12.4.11 - (IBOutlet IDALevelMeterView) levelMeter  [read],[write],[nonatomic],[weak]

IDALevelMeterView is a custom class for the visual representation of the volume of a recording. This is used during the recording of the exercise to let the user know if he or she is being loud enough or too loud, both of which can leave the saved audio file in a sub-optimal state.

4.12.4.12 - (NSManagedObjectContext) managedObjectContext  [read],[write],[nonatomic],[strong]

Manages a collection of managed objects loaded from the data store (SQLite).

4.12.4.13 - (AVAudioPlayer) player  [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of exercise instructional audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.12.4.14 - (IBOutlet UIProgressView) progressView  [read],[write],[nonatomic],[weak]

This UIProgressView is displayed at the top of the exerciseViewDrawer, and shows the user how much time is left in the playback of an exercise sample or the recording of the user repeating the sample. These two types are delineated by flipping the UIProgressView (unseen by the user). When audio is being played, the bar fills left to right, when audio is being recorded, the bar empties from left to right.
4.12.4.15 - (AVAudioRecorder*) recorder [read],[write],[nonatomic],[strong]

This AVAudioRecorder handles recording of exercise audio, and makes calls to methods pertaining to the recording of audio, such as audioRecorderDidFinishRecording.

4.12.4.16 - (IBOutlet UITabBar*) tabBar [read],[write],[nonatomic],[weak]

The UITabBar shown at the bottom of the UIViewController which allows the user to switch between the Voice Test and Exercise portion of the application.

4.12.4.17 - (IBOutlet UITabBarItem*) vocalTestTabBarItem [read],[write],[nonatomic],[weak]

This UITabBarItem, shown in the left position of the tabBar UITabBar object, performs the segue to present the IDAVocalTestViewController.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExercisesViewController.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExercisesViewController.m

4.13 IDAExercisesViewController() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAExercisesViewController.m

4.14 IDAGraphViewController Class Reference

#import <IDAGraphViewController.h>

Inheritance diagram for IDAGraphViewController:

```
+--------------------------+
| UIViewController        |
| - IDAGraphViewController |
```

4.14.1 Detailed Description

This file is not implemented in the current release.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAGraphViewController.h
4.15  IDAGraphViewController() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAGraphViewController.m

4.16  IDAInfoViewController Class Reference

#import <IDAInfoViewController.h>

Inheritance diagram for IDAInfoViewController:

```
    UIViewController  <UITextViewDelegate>  <UITextFieldDelegate>
                     |                           |
                     IDAInfoViewController
```

4.16.1  Detailed Description

This subclass of UIViewController is accessed by clicking the buttonInfoDisclosure UIButton in the bottom right corner of the IDAVocalTestViewController. This screen displays information about uses of the application, definitions of the different measures, technical information about the application, and disclaimers notifying the user that this application is to be used as a supplement to seeing a clinician, not a replacement.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.h

4.17  IDAInfoViewController() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.m

4.18  IDAIntervalCell Class Reference

#import <IDAIntervalCell.h>

Inheritance diagram for IDAIntervalCell:

```
    UITableViewCell
       |
       IDAIntervalCell
```

Generated on Mon Apr 8 2013 21:53:11 for iOS-based Voice Analysis Application for Speech Therapy by Doxygen
Properties

- IBOutlet UILabel * intervalLabel
- IBOutlet UITextField * intervalTextField

4.18.1 Detailed Description

This subclass of UITableViewCell is used to allow the user or clinician to select how often a test should be performed.

4.18.2 Property Documentation

4.18.2.1 - (IBOutlet UILabel *) intervalLabel [read], [write], [nonatomic], [weak]

This label shows the value selected by the user in the UIPickerView.

4.18.2.2 - (IBOutlet UITextField *) intervalTextField [read], [write], [nonatomic], [weak]

This label is given first responder status when the user selects the cell, presenting the UIPickerView. This object is only used as a workaround because UILabel cannot be given first responder status or have an input view associated with it.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAIntervalCell.h

4.19 IDALevelMeterView Class Reference

#import <IDALevelMeterView.h>

Inheritance diagram for IDALevelMeterView:

```
[UIView <AVAudioRecorderDelegate> IDALevelMeterView]
```

Instance Methods

- (void) - levelMeterStart:
- (void) - levelTimerCallback:
- (void) - levelMeterStop

Protected Attributes

- NSTimer * levelTimer
Properties

- IBOutlet UIImageView * levelMeter1
- IBOutlet UIImageView * levelMeter2
- IBOutlet UIImageView * levelMeter3
- IBOutlet UIImageView * levelMeter4
- IBOutlet UIImageView * levelMeter5
- IBOutlet UIImageView * levelMeter6
- IBOutlet UIImageView * levelMeter7
- IBOutlet UIImageView * levelMeter8
- IBOutlet UIImageView * levelMeter9
- IBOutlet UIImageView * levelMeter10
- IBOutlet UIImageView * levelMeter11
- IBOutlet UIImageView * levelMeter12
- IBOutlet UIImageView * levelMeter13
- IBOutletUIImageView * levelMeter14
- IBOutlet UIImageView * levelMeter15
- AVAudioRecorder * recorder

4.19.1 Detailed Description

This custom object is a subclass of UIView. It contains numerous images which change color to signify amplitude levels of the microphone input. This object is set to listen on a specific AVAudioRecorder which has had metering enabled, and will show the current levels while the microphone is recording.

4.19.2 Method Documentation

4.19.2.1 - (void) levelMeterStart: (AVAudioRecorder *) recorderToUse

This method is called during the setup of the AVAudioRecorder, and begins the metering of the AVAudioRecorder which is passed in.

4.19.2.2 - (void) levelMeterStop

This method returns all of the UIImageView objects back to the default state and stops the metering of the AVAudioRecorder.

4.19.2.3 - (void) levelTimerCallback: (NSTimer *) timer

This method is called by the levelTimer NSTimer. The method handles the clearing of all UIImageView objects back to their default state, and sets the new level depending on the amplitude of the signal.

4.19.3 Member Data Documentation

4.19.3.1 - (NSTimer *) levelTimer [protected]

This NSTimer causes the display to update, so that the lights are always showing the most current level read in.

4.19.4 Property Documentation

4.19.4.1 - (IBOutlet UIImageView *) levelMeter1 [read],[write],[nonatomic],[strong]

Green.
4.19.4.2 - (IBOutlet UIImageView) levelMeter10 [read],[write],[nonatomic],[strong] Green.

4.19.4.3 - (IBOutlet UIImageView) levelMeter11 [read],[write],[nonatomic],[strong] Yellow.

4.19.4.4 - (IBOutlet UIImageView) levelMeter12 [read],[write],[nonatomic],[strong] Yellow.

4.19.4.5 - (IBOutlet UIImageView) levelMeter13 [read],[write],[nonatomic],[strong] Yellow.

4.19.4.6 - (IBOutlet UIImageView) levelMeter14 [read],[write],[nonatomic],[strong] Red.

4.19.4.7 - (IBOutlet UIImageView) levelMeter15 [read],[write],[nonatomic],[strong] Red.

4.19.4.8 - (IBOutlet UIImageView) levelMeter2 [read],[write],[nonatomic],[strong] Green.

4.19.4.9 - (IBOutlet UIImageView) levelMeter3 [read],[write],[nonatomic],[strong] Green.

4.19.4.10 - (IBOutlet UIImageView) levelMeter4 [read],[write],[nonatomic],[strong] Green.

4.19.4.11 - (IBOutlet UIImageView) levelMeter5 [read],[write],[nonatomic],[strong] Green.

4.19.4.12 - (IBOutlet UIImageView) levelMeter6 [read],[write],[nonatomic],[strong] Green.

4.19.4.13 - (IBOutlet UIImageView) levelMeter7 [read],[write],[nonatomic],[strong] Green.
4.19.4.14 - (IBOutlet UIImageView) levelMeter8 [read],[write],[nonatomic],[strong]
Green.

4.19.4.15 - (IBOutlet UIImageView) levelMeter9 [read],[write],[nonatomic],[strong]
Green.

4.19.4.16 - (AVAudioRecorder) recorder [read],[write],[nonatomic],[strong]
This AVAudioRecorder is the recorder which the object will take readings from and change the UIImageViews accordingly, to give the effect of a level meter.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDALevelMeterView.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDALevelMeterView.m

4.20 IDANotificationSwitchCell Class Reference

#import <IDANotificationSwitchCell.h>

Inheritance diagram for IDANotificationSwitchCell:

```
UITableViewCell
    IDANotificationSwitchCell
```

Properties

- IBOutlet UISwitch * notificationsSwitch

4.20.1 Detailed Description

This subclass of UITableViewCell contains a label and a switch. It was specifically designed for use in enabling and disabling local notifications for exercise reminders.

4.20.2 Property Documentation

4.20.2.1 - (IBOutlet UISwitch) notificationsSwitch [read],[write],[nonatomic],[weak]
This UISwitch controls the enabling and disabling of local notifications for exercises.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDANotificationSwitchCell.h
4.21 IDAOccurrenceCell Class Reference

#import <IDAOccurrenceCell.h>

Inheritance diagram for IDAOccurrenceCell:

![Inheritance diagram](image)

Properties

- IBOulet UITextField * occurrenceTextField
- IBOulet UILabel * occurrenceLabel
- IBOulet UILabel * titleLabel

4.21.1 Detailed Description

This subclass of UITableViewCell is used to display to the user the start and end times chosen for the exercise window. These cells allow the user to choose a window of time each day where he or she will complete the exercises assigned. By doing this, the user can prevent an alarm from occurring while he or she is sleeping or having family time, for example.

4.21.2 Property Documentation

4.21.2.1 - (IBOutlet UILabel *) occurrenceLabel [read],[write],[nonatomic],[weak]

This UILabel shows the value which was selected in the UIDatePicker by the user.

4.21.2.2 - (IBOutlet UITextField *) occurrenceTextField [read],[write],[nonatomic],[weak]

This UITextField is set as the first responder, presenting the UIDatePicker. This is object is strictly used as a work around because UILabels cannot be given first responder status or have an input view.

4.21.2.3 - (IBOutlet UILabel *) titleLabel [read],[write],[nonatomic],[weak]

This UILabel tells the user whether he or she is setting the start time or the end time.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAOccurrenceCell.h

4.22 IDASavedTestDetailViewController Class Reference

#import <IDASavedTestDetailViewController.h>

Inheritance diagram for IDASavedTestDetailViewController:

![Inheritance diagram](image)
Instance Methods

- (IBAction) playWasPressed: (id) sender
- (IBAction) sliderChanged: (id) sender
- (IBAction) editWasPressed: (id) sender

Properties

- id < IDASavedTestsDetailViewControllerDelegate > delegate
- NSManagedObjectContext * managedObjectContext
- VocalTests * test
- IBOutlet UILabel * labelDateTaken
- IBOutlet UILabel * labelJitterScore
- IBOutlet UILabel * labelCPPScore
- IBOutlet UILabel * labelFundFreq
- IBOutlet UIButton * buttonPlay
- IBOutlet UISlider * sliderAudioProgress
- IBOutlet UIBarButtonItem * buttonEdit
- IBOutlet UIBarButtonItem * notesEditWindow
- IBOutlet UITextView * textViewNotes
- IBOutlet UITextField * textFieldName
- IBOutlet UILabel * labelName
- AVAudioSession * audioSession
- AVAudioPlayer * player
- NSTimer * sliderTimer
- NSString * originalName
- NSString * originalNotes

4.22.1 Method Documentation

4.22.1.1 - (IBAction) editWasPressed: (id) sender

This method is fired when the user selects the buttonEdit UIBarButtonItem. It allows editing of the name and notes attributes of the VocalTests object.

4.22.1.2 - (IBAction) playWasPressed: (id) sender

This method handles the pause and playback of the VocalTests object's audio. It uses audioSession and player to playback the WAV files.

4.22.1.3 - (IBAction) sliderChanged: (id) sender

This method is fired whenever the user changes the sliderAudioProgress UISlider. It sets the playback time for the audio file to the position selected in the UISlider.
4.22.2 Property Documentation

4.22.2.1 - (AVAudioSession*) audioSession [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback of the test file from within this UIViewController. It sets the audio to be played through the external speakers rather than the earpiece.

4.22.2.2 - (IBOutlet UIBarButtonItem*) buttonEdit [read],[write],[nonatomic],[strong]

This UIButton enables and disables changing of the name and notes attributes of the VocalTests object currently being reviewed.

4.22.2.3 - (IBOutlet UIButton*) buttonPlay [read],[write],[nonatomic],[strong]

This button handles the playback and pausing of the audio file while reviewing.

4.22.2.4 - (id<IDASavedTestsDetailViewControllerDelegate>) delegate [read],[write],[nonatomic],[weak]

This delegate object allows IDASavedTestsTVC to pass the selectedTest object from that UITableViewController across to this UIView.

4.22.2.5 - (IBOutlet UILabel*) labelCPPScore [read],[write],[nonatomic],[strong]

This UILabel displays the Cepstral Peak Prominence score for the displayed VocalTests object (stored in the cpp-Score attribute), measured in decibels.

4.22.2.6 - (IBOutlet UILabel*) labelDateTaken [read],[write],[nonatomic],[strong]

This UILabel displays the date that the test was originally performed.

4.22.2.7 - (IBOutlet UILabel*) labelFundFreq [read],[write],[nonatomic],[strong]

This UILabel displays the Fundamental Frequency for the displayed VocalTests object (stored in the fundFreq attribute), measured in Hertz.

4.22.2.8 - (IBOutlet UILabel*) labelJitterScore [read],[write],[nonatomic],[strong]

This UILabel displays the Local Jitter score for the displayed VocalTests object (stored in the jitterScore attribute), measured in the form of a percentage.

4.22.2.9 - (IBOutlet UILabel*) labelName [read],[write],[nonatomic],[strong]

This UILabel is shown when the user clicks the buttonEdit UIBarButtonItem, and notifies the user what the textField-Name object is used for.

4.22.2.10 - (NSManagedObjectContext*) managedObjectContext [read],[write],[nonatomic],[strong]

This managed object context allows access to the NSApplication object for saving items off to the Core Data store. In this ViewController, it is used to save changes to the name and notes objects of the VocalTests object being displayed.
4.22.2.11 - (IBOutlet UIButton) notesEditWindow  [read],[write],[nonatomic],[strong]

This UIButton is NOT clickable. It strictly visually shows the user that the notes window is now clickable and editable.

4.22.2.12 - (NSString) originalName  [read],[write],[nonatomic],[strong]

This NSString object is set when the user begins editing, and allows the user to cancel the edit and be shown what used to be the name of the VocalTests object.

4.22.2.13 - (NSString) originalNotes  [read],[write],[nonatomic],[strong]

This NSString object is set when the user begins editing, and allows the user to cancel the edit and be shown what used to be the note of the VocalTests object.

4.22.2.14 - (AVAudioPlayer) player  [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of test audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.22.2.15 - (IBOutlet UISlider) sliderAudioProgress  [read],[write],[nonatomic],[strong]

This UISlider shows how much of the recorded file has been played back. It can be scrubbed to change the current playback location within the current sound file.

4.22.2.16 - (NSTimer) sliderTimer  [read],[write],[nonatomic],[strong]

This timer fires to update the position of the sliderAudioProgress UISlider so that it represents the current playback position of the audio.

4.22.2.17 - (VocalTests) test  [read],[write],[atomic]

This object represents the currently selected VocalTests object. By keeping this reference to the object, changes can be made quickly and easily.

4.22.2.18 - (IBOutlet UITextField) textFieldName  [read],[write],[nonatomic],[strong]

This UITextField is shown when the user clicks the buttonEdit UIBarButtonItem, and allows the user to change the name attribute of the VocalTests object.

4.22.2.19 - (IBOutlet UITextView) textViewNotes  [read],[write],[nonatomic],[strong]

This UITextView becomes first responder when the notes are clicked to be edited. It handles presentation and dismissal of the keyboard, and changes the notes attribute of the VocalTests object.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDASavedTestDetailViewController.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDASavedTestDetailViewController.m
4.23 IDASavedTestDetailViewController() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDASavedTestDetailViewController.m

4.24 <IDASavedTestsDetailViewControllerDelegate> Protocol Reference

#import <IDASavedTestDetailViewController.h>

Inheritance diagram for <IDASavedTestsDetailViewControllerDelegate>:

```
<NSObject>
    <IDASavedTestsDetailViewControllerDelegate>
      IDASavedTestsTVC
```

The documentation for this protocol was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDASavedTestDetailViewController.h

4.25 IDASavedTestsTVC Class Reference

#import <IDASavedTestsTVC.h>

Inheritance diagram for IDASavedTestsTVC:

```
UITableViewController
    <NSFetchedResultsControllerDelegate>
      <NSObject>
        <NSObject>
          IDASavedTestsTVC
```

Properties

- id< IDASavedTestsTVCDelegate > delegate
- NSManagedObjectContext * managedObjectContext
- NSFetchedResultsController * fetchedResultsController
- VocalTests * selectedTest

Additional Inherited Members

4.25.1 Property Documentation

4.25.1.1 -(id<IDASavedTestsTVCDelegate>) delegate [read],[write],[nonatomic],[weak]

This object represents the delegate for the IDASavedTestsTVC.
4.25.1.2 - (NSFetchedResultsController+) fetchedResultsController [read],[write],[nonatomic],[strong] 
This fetched results controller handles fetching and sorting the VocalTests objects to be displayed in the UITableView.

4.25.1.3 - (NSManagedObjectContext+) managedObjectContext [read],[write],[nonatomic],[strong] 
This managed object context allows access to the NSApplication object for saving items off to the Core Data store. In this ViewController, it is used to delete unwanted VocalTests objects.

4.25.1.4 - (VocalTests+) selectedTest [read],[write],[nonatomic],[strong] 
This object is set when the user selects a UICell from within the UITableView. It is used to pass test data on to the IDASavedTestDetailViewController.

The documentation for this class was generated from the following file:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/- NewVocalTest/IDASavedTestsTVC.h

4.26 IDASavedTestsTVC() Category Reference

The documentation for this category was generated from the following file:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/- NewVocalTest/IDASavedTestsTVC.m

4.27 <IDASavedTestsTVCDelegate> Protocol Reference

#import <IDASavedTestsTVC.h>

Inheritance diagram for <IDASavedTestsTVCDelegate>:

```
<NSObject>
  <NSObject>
    <IDASavedTestsTVCDelegate>
      IDASavedTestsTVC
      IDAVocalTestViewController
  
```

The documentation for this protocol was generated from the following file:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/- NewVocalTest/IDASavedTestsTVC.h

4.28 IDAVocalTestViewController Class Reference

#import <IDAVocalTestViewController.h>

Inheritance diagram for IDAVocalTestViewController:

```
```
Instance Methods

- (IBAction) playWasPressed:
- (IBAction) saveWasPressed:
- (IBAction) sliderChanged:
- (IBAction) beginVoiceTestPressed:
- (IBAction) dismissResultsViewPressed:
- (IBAction) typeSpeechSelected:
- (IBAction) typeVowelSelected:
- (IBAction) typeCancelSelected:

Protected Attributes

- NSURL * recordedTmpFile
- NSURL * TESTURL
- NSTimer * audioProgressTimer
- NSTimer * progressTimer
- NSError * error
- NSNumber * jitterStore
- NSNumber * cppStore
- NSNumber * fundamentalFrequency

Properties

- IBOutlet UIBarButtonItem * savedTestsNavButton
- IBOutlet UIButton * buttonPlay
- IBOutlet UIButton * buttonSave
- IBOutlet UIButton * buttonInfoDisclosure
- IBOutlet UILabel * labelStatusLine1
- IBOutlet UILabel * labelStatusLine2
- IBOutlet UILabel * labelStatusLine3
- IBOutlet UISlider * sliderAudioProgress
- NSManagedObjectContext * managedObjectContext
- IBOutlet UITabBar * tabBar
- IBOutlet UITabBarItem * vocalTestTabBarItem
- IBOutlet UITabBarItem * exercisesTabBarItem
- IBOutlet IDALevelMeterView * levelMeter
- IBOutlet UIButton * beginVoiceTest
- IBOutlet UIView * voiceTestView
- IBOutlet UIView * testResultsView
- IBOutlet UILabel * labelVoiceTest1
- IBOutlet UILabel * labelVoiceTest2
- IBOutlet UIView * progressVoiceTest
- IBOutlet UIButton * speechButton
- IBOutlet UIButton * vowelButton
- IBOutlet UIButton * cancelButton
- AVAudioSession * audioSession
- AVAudioPlayer * player
- AVAudioRecorder * recorder
- NSNumber * isCalledFromNotification
4.28.1 Detailed Description

This subclass of UIViewController contains all of the functionality for completion of voice tests. It makes calls to JitterCalc_mod.mm for analysis of sound files. The scores are then returned to this view, processed, and displayed to the user. This view also handles playing back the most recent test and saving that test for later review. It is a delegate for AVAudioRecorder, AVAudioPlayer, IDASavedTestsTVC, and UITabBar.

4.28.2 Method Documentation

4.28.2.1 -(IBAction) beginVoiceTestPressed: (id) sender

This method partially presents the voiceTestView, and presents the vowelButton, speechButton, and cancelButton.

4.28.2.2 -(IBAction) dismissResultsViewPressed: (id) sender

This method is fired when the user selects the ‘x’ button in the upper right hand corner of the testResultsView. It animates the testResultsView out from the user interface.

4.28.2.3 -(IBAction) playWasPressed: (id) sender

This method handles the pause and playback of test audio from the testResultsView. It uses audioSession and player to playback the WAV files.

4.28.2.4 -(IBAction) saveWasPressed: (id) sender

This method creates a VocalTests object and stores it into the Core Data store. The test audio is also saved in the Documents directory from this method.

4.28.2.5 -(IBAction) sliderChanged: (id) sender

This method is fired whenever the user changes the sliderAudioProgress UISlider. It sets the playback time for the audio file to the position selected in the UISlider.

4.28.2.6 -(IBAction) typeCancelSelected: (id) sender

This method is fired from the cancelButton and dismisses the vocalTestView, cancelling the voice test which the user began by pressing the beginVoiceTest button.

4.28.2.7 -(IBAction) typeSpeechSelected: (id) sender

This method is fired from the speechButton and sets the testType attribute for the VocalTests object referencing the test which is about to be taken to "speech". It also begins the voice test, presenting the vocalTestView completely.

4.28.2.8 -(IBAction) typeVowelSelected: (id) sender

This method is fired from the vowelButton and sets the testType attribute for the VocalTests object referencing the test which is about to be taken to "vowel". It also begins the voice test, presenting the vocalTestView completely.
4.28.3 Member Data Documentation

4.28.3.1 - (NSTimer *) audioProgressTimer [protected]

4.28.3.2 - (NSNumber *) cppStore [protected]

4.28.3.3 - (NSError *) error [protected]

4.28.3.4 - (NSNumber *) fundamentalFrequency [protected]

4.28.3.5 - (NSNumber *) jitterStore [protected]

4.28.3.6 - (NSTimer *) progressTimer [protected]

4.28.3.7 - (NSURL *) recordedTmpFile [protected]

4.28.3.8 - (NSURL *) TESTURL [protected]

4.28.4 Property Documentation

4.28.4.1 - (AVAudioSession *) audioSession [read],[write],[nonatomic],[strong]

This instance of AVAudioSession handles the playback and recording of test files from within this UIViewController. It sets the audio to be played through the external speakers rather than the earpiece.

4.28.4.2 - (IBOutlet UIButton *) beginVoiceTest [read],[write],[nonatomic],[weak]

This UIButton presents the voiceTestView, beginning the process of a voice test for the user. The test is not begun until the speechButton or vowelButton are selected. This button calls the method beginVoiceTestPressed.

4.28.4.3 - (IBOutlet UIButton *) buttonInfoDisclosure [read],[write],[nonatomic],[strong]

The button in the lower right of the main UIView for this UIViewController. It takes the user to the IDAInfoViewController.

4.28.4.4 - (IBOutlet UIButton *) buttonPlay [read],[write],[nonatomic],[strong]

The UIButton with a playback icon located on the testResultsView. Allows immediate playback of the test which was just completed.

4.28.4.5 - (IBOutlet UIButton *) buttonSave [read],[write],[nonatomic],[strong]

The save button located on the testResultsView. Saves to a VocalTest object and stores in the core data store.

4.28.4.6 - (IBOutlet UIButton *) cancelButton [read],[write],[nonatomic],[weak]

This UIButton allows the user to exit without performing a test after he or she has selected beginVoiceTest, but before he or she has selected the speechButton or vowelButton.

4.28.4.7 - (IBOutlet UITabBarItem *) exercisesTabBarItem [read],[write],[nonatomic],[weak]

The right button within the tabBar which allows the user to switch to the IDAExercisesViewController.
Class Documentation

4.28.4.8 - (NSNumber) isCalledFromNotification  [read],[write],[nonatomic],[strong]

This is a boolean value which is set to TRUE if the user is entering the application from a local notification for an exercise. It is then used to set the active UIViewController to IDAExercisesViewController. If this boolean value is set to FALSE, the application opens in whatever UIViewController it was left in during the last use.

4.28.4.9 - (IBOutlet UILabel) labelStatusLine1  [read],[write],[nonatomic],[strong]

The UILabel which displays the Fundamental Frequency in the TestResultsView.

4.28.4.10 - (IBOutlet UILabel) labelStatusLine2  [read],[write],[nonatomic],[strong]

The UILabel which displays the Local Jitter score in the TestResultsView.

4.28.4.11 - (IBOutlet UILabel) labelStatusLine3  [read],[write],[nonatomic],[strong]

The UILabel which displays the Cepstral Peak Prominence score in the TestResultsView.

4.28.4.12 - (IBOutlet UILabel) labelVoiceTest1  [read],[write],[nonatomic],[weak]

This UILabel is the top line of text presented in the voiceTestView.

4.28.4.13 - (IBOutlet UILabel) labelVoiceTest2  [read],[write],[nonatomic],[weak]

This UILabel is the bottom line of text presented in the voiceTestView.

4.28.4.14 - (IBOutlet IDALevelMeterView) levelMeter  [read],[write],[nonatomic],[weak]

IDALevelMeterView is a custom class for the visual representation of the volume of a recording. This is used during the recording of the voice test to let the user know if he or she is being loud enough or too loud, both of which can cause problems during the analysis of the sound file.

4.28.4.15 - (NSManagedObjectContext) managedObjectContext  [read],[write],[nonatomic],[strong]

Manages a collection of managed objects loaded from the data store (SQLite).

4.28.4.16 - (AVAudioPlayer) player  [read],[write],[nonatomic],[strong]

This AVAudioPlayer handles playback of test audio, and makes calls to methods pertaining to the playback of the audio, such as audioPlayerDidFinishPlaying.

4.28.4.17 - (IBOutlet UIProgressView) progressVoiceTest  [read],[write],[nonatomic],[weak]

This UIProgressView gives a visual representation of how much longer the user must continue making the test sound.

4.28.4.18 - (AVAudioRecorder) recorder  [read],[write],[nonatomic],[strong]

This AVAudioRecorder handles recording of test audio, and makes calls to methods pertaining to the recording of audio, such as audioRecorderDidFinishRecording.
4.28.4.19 - (IBOutlet UIBarButtonItem) savedTestsNavButton [read],[write],[nonatomic],[weak]

The button in the UINavigationBar which takes the user to the IDASavedTestsTVC.

4.28.4.20 - (IBOutlet UISlider) sliderAudioProgress [read],[write],[nonatomic],[strong]

The UISlider shown in the TestResultsView which shows how much of the recorded file has been played back. It can be scrubbed to change the current playback location within the current sound file.

4.28.4.21 - (IBOutlet UIButton) speechButton [read],[write],[nonatomic],[weak]

This UIButton sets the testType attribute on the VocalTests object. It is used for a connected speech test sample.

4.28.4.22 - (IBOutlet UITabBar) tabBar [read],[write],[nonatomic],[weak]

The UITabBar shown at the bottom of the UIViewController which allows the user to switch between the Voice Test and Exercise portion of the application.

4.28.4.23 - (IBOutlet UIView) testResultsView [read],[write],[nonatomic],[weak]

This UIView is presented as an overlay, displaying the scores and fundamental frequency for the sound file. The user can also play back the test audio and optionally save the test for later review. If saving is not desired, there is an exit button in the upper right corner of the view.

4.28.4.24 - (IBOutlet UITabBarItem) vocalTestTabBarItem [read],[write],[nonatomic],[weak]

The left button within the tabBar which shows the user that he or she is currently in the Voice Test portion of the application.

4.28.4.25 - (IBOutlet UIView) voiceTestView [read],[write],[nonatomic],[weak]

This UIView is a drawer-style modal view, walking the user through a voice test. Instructions are contained within, as well as a visual representation of how long the test has remaining. This view is presented by the method beginVoiceTestPressed.

4.28.4.26 - (IBOutlet UIButton) vowelButton [read],[write],[nonatomic],[weak]

This UIButton sets the testType attribute on the VocalTests object. It is used for a sustained vowel test sample.

The documentation for this class was generated from the following files:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAVocalTestViewController.h
- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAVocalTestViewController.m

4.29 IDAVocalTestViewController() Category Reference

The documentation for this category was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/IDAVocalTestViewController.m
4.30 UIView(FindFirstResponder) Category Reference

#import <UIView+FindFirstResponder.h>

Instance Methods

• (UIView *) - findFirstResponder

4.30.1 Detailed Description

This is a category extension for the UIView class. It will return the current UIView which holds the first responder status. It is a quick way to resign the current first responder from any point, which has the side effect of dismissing the current input view (i.e. keyboard or UIPickerView).

4.30.2 Method Documentation

4.30.2.1 - (UIView *) findFirstResponder

The documentation for this category was generated from the following files:

• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/UIView+FindFirstResponder.h
• /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/UIView+FindFirstResponder.m

4.31 Usage Class Reference

#import <Usage.h>

Inheritance diagram for Usage:

```
NSManagedObject
  Usage
```

Properties

• NSString * function
• NSDate * timestamp

4.31.1 Detailed Description

This object stores usage data for the application and is transmitted to a MySQL server for review. This is strictly used during the clinical trial, and does not directly track a specific user. An object of this type is created and stored in the data model each time a test or exercise is begun, regardless of if the user saves the exercise or test. We store the function (exercise or test) and the time and date it was performed.
4.31.2 Property Documentation

4.31.2.1 - (NSString*) function  [read],[write],[nonatomic],[retain]

Contains either "Exercise" or "VocalTest", depending upon what the user is currently using the application for.

4.31.2.2 - (NSDate*) timestamp  [read],[write],[nonatomic],[retain]

A time/date stamp indicating the moment the exercise or test was begun.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/-
  NewVocalTest/Usage.h

4.32 VocalTests Class Reference

#import <VocalTests.h>

Inheritance diagram for VocalTests:

```
NSManagedObject

VocalTests
```

Properties

- NSNumber * cppScore
- NSDate * dateTaken
- NSNumber * fundFreq
- NSNumber * jitterScore
- NSString * name
- NSString * notes
- NSString * soundFilePath
- NSString * testType

4.32.1 Detailed Description

This object represents a single voice test. It includes the CPP and Jitter scores, the date the test was taken, the fundamental frequency, the name and notes as given by the user, a path to the location of the sound file, and what type of test was taken (i.e. sustained vowel or connected speech). Only selected tests are saved, and may be deleted from the IDASavedTestsTVC.

4.32.2 Property Documentation

4.32.2.1 - (NSNumber*) cppScore  [read],[write],[nonatomic],[retain]

An NSNumber containing the Cepstral Peak Prominence score in decibels for the VocalTest.
4.32.2.2 - (NSDate) dateTaken  [read],[write],[nonatomic],[retain]

A date/time stamp which represents the when the VocalTest was created and performed.

4.32.2.3 - (NSNumber) fundFreq  [read],[write],[nonatomic],[retain]

An NSNumber containing the Fundamental Frequency in Hertz for the VocalTest.

4.32.2.4 - (NSNumber) jitterScore  [read],[write],[nonatomic],[retain]

An NSNumber containing the Local Jitter score, in the form of a percentage, for the VocalTest.

4.32.2.5 - (NSString) name  [read],[write],[nonatomic],[retain]

This name is automatically set to the value of dateTaken in a human-readable language. It can be changed in IDASavedTestDetailViewController to whatever the user would like to name the VocalTest.

4.32.2.6 - (NSString) notes  [read],[write],[nonatomic],[retain]

This note is originally set with placeholder text, and can be set by the user in IDASavedTestDetailViewController. The user is encouraged to keep notes about the conditions the test were performed in so the therapist can gain a good understanding of the scores achieved.

4.32.2.7 - (NSString) soundFilePath  [read],[write],[nonatomic],[retain]

This NSString is the path to the VocalTest soundfile which was analyzed by the application. It can be replayed by the user at any time from the IDASavedTestDetailViewController. The file is deleted when the VocalTest object is deleted from the Core Data store.

4.32.2.8 - (NSString) testType  [read],[write],[nonatomic],[retain]

This NSString contains the value of either "Speech" for a connected speech test sample, or "Vowel" for a sustained vowel test sample. This is important for the color coding and interpretation of scores.

The documentation for this class was generated from the following file:

- /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/-NewVocalTest/VocalTests.h
Chapter 5

File Documentation

5.1 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/CoreDataTableViewController.h File Reference

#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>

Classes

• class CoreDataTableViewController

5.2 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/CoreDataTableViewController.m File Reference

#import "CoreDataTableViewController.h"

5.3 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Exercises.h File Reference

#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>

Classes

• class Exercises
5.4 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Exercises.m File Reference

#import "Exercises.h"

5.5 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/fft.hpp File Reference

#include <stdlib.h>
#include <math.h>

Macros

• #define TWO_PI (6.2831853071795864769252867665590057683943L)

Functions

• void fft (int N, double(*)[2], double(*)[2])
• void fft_rec (int N, int offset, int delta, double(*)[2], double(*)[2], double(*)[2])
• void ifft (int N, double(*)[2], double(*)[2])

Variables

• int N_master

5.5.1 Macro Definition Documentation

5.5.1.1 #define TWO_PI (6.2831853071795864769252867665590057683943L)

5.5.2 Function Documentation

5.5.2.1 void fft ( int N, double() x[2], double() X[2] )

5.5.2.2 void fft_rec ( int N, int offset, int delta, double() x[2], double() X[2], double() XX[2] )

5.5.2.3 void ifft ( int N, double() x[2], double() X[2] )

5.5.3 Variable Documentation

5.5.3.1 int N_master

5.6 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.h File Reference

#import <UIKit/UIKit.h>
Classes

• class IDAAddExerciseCell

5.7 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseCell.m File Reference

#import "IDAAddExerciseCell.h"

5.8 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.h File Reference

#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreAudio/CoreAudioTypes.h>
#import <AudioToolbox/AudioServices.h>
#import "IDALevelMeterView.h"

Classes

• class IDAAddExerciseViewController

5.9 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAddExerciseViewController.m File Reference

#import "IDAAddExerciseViewController.h"

5.10 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAAppDelegate.h File Reference

#import <UIKit/UIKit.h>
#import "IDAVocalTestViewController.h"

Classes

• class IDAAppDelegate
Classes

- class IDAExerciseConfigTVC

Classes

- protocol <IDAExerciseHistoryTVCDelegate>
- class IDAExerciseHistoryTVC
#import "IDAExerciseHistoryTVC.h"
#import "IDAAppDelegate.h"
#import "Exercises.h"

Classes

• class IDAExercisesViewController

#import "IDAExercisesViewController.h"
#import "Exercises.h"
#import "Usage.h"
#import "IDAAppDelegate.h"

Classes

• class IDAGraphViewController

#import "IDAGraphViewController.h"
5.20  /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.h File Reference

#import <UIKit/UIKit.h>

Classes

- class IDAInfoViewController

5.21  /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAInfoViewController.m File Reference

#import "IDAInfoViewController.h"

5.22  /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAIntervalCell.h File Reference

#import <UIKit/UIKit.h>

Classes

- class IDAIntervalCell

5.23  /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDAIntervalCell.m File Reference

#import "IDAIntervalCell.h"

5.24  /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva's Projects/VocalTest Versions/NewVocalTest/NewVocalTest/IDALevelMeterView.h File Reference

#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreAudio/CoreAudioTypes.h>
#import <AudioToolbox/AudioServices.h>
Classes

- class IDALevelMeterView

```objective-c
#import "IDALevelMeterView.h"
```

5.25

5.26

5.27

5.28

5.29

Classes

- class IDANotificationSwitchCell

```objective-c
#import <UIKit/UIKit.h>
```

```objective-c
#import "IDANotificationSwitchCell.h"
```

```objective-c
#import <UIKit/UIKit.h>
```

```objective-c
#import "IDADoccurrenceCell.h"
```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreAudio/CoreAudioTypes.h>
#import <AudioToolbox/AudioServices.h>
#import "VocalTests.h"

Classes

- protocol <IDASavedTestsDetailViewControllerDelegate>
- class IDASavedTestDetailViewController

#import "IDASavedTestDetailViewController.h"

Classes

- protocol <IDASavedTestsTVCDelegate>
- class IDASavedTestsTVC

#import "IDASavedTestsTVC.h"
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreAudio/CoreAudioTypes.h>
#import <AudioToolbox/AudioServices.h>
#import <CoreData/CoreData.h>
#import "VocalTests.h"
#import "IDASavedTestsTVC.h"
#import "IDALevelMeterView.h"

Classes

- class IDAVocalTestViewController

Variables

- NSInteger testCount = 0
- NSString *typeOfTest = @"

5.35.1 Variable Documentation

5.35.1.1 NSInteger testCount = 0

5.35.1.2 NSString *typeOfTest = @"

5.36 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/JitterCalc_mod.mm File Reference

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "fft.hpp"

Macros

- #define JITTERCALC_MOD_H
Functions

- `float JitterCalc (SInt16 *wavdata, int *MyPitch, int numSamples, int sr)`
- `void LinLeastSquares (double *x, double *y, int n, double *Alpha, double *Beta)`
- `float CPPCalc (SInt16 *wavdata, int *MyPitch, int numSamples, int sr)`
- `void AnalyzeSoundFile (const char *soundfile, int *MyPitch, float *cpp, float *jitter)`

5.36.1 Macro Definition Documentation

5.36.1.1 `#define JITTERCALC_MOD_H`

This file contains three methods: JitterCalc, CPPCalc, and AnalyzeSoundFile. The only method that should be called is the AnalyzeSoundFile method, which calls the calculation methods from within it. The AnalyzeSoundFiles method takes four arguments: a const char pointer containing the path to the sound file to be analyzed, an integer pointer storing the pitch, and a float for each score: cpp and jitter. MyPitch, cpp, and jitter will be overwritten in this method; they do not need to contain anything when passed in.

5.36.2 Function Documentation

5.36.2.1 `void AnalyzeSoundFile (const char *soundfile, int *MyPitch, float *cpp, float *jitter)`

This method accepts a path to a WAV sound file to be analyzed. It strips off headers and analyzes values in the header, then passes this data to the JitterCalc and CPPCalc methods for further analysis. By only analyzing the headers once, we can run multiple analysis algorithms using a minimal number of CPU cycles.

5.36.2.2 `float CPPCalc (SInt16 *wavdata, int *MyPitch, int numSamples, int sr)`

This method is called from the AnalyzeSoundFile method and can only perform CPP analysis on pre-processed WAV data. It must be passed the sample rate and number of samples of the WAV data it is passed. A float value of the Cepstral Peak Prominence value, measured in decibels, is returned, and the MyPitch value is set to the Fundamental Frequency, found by analyzing the cepstral peak found.

5.36.2.3 `float JitterCalc (SInt16 *wavdata, int *MyPitch, int numSamples, int sr)`

This method is called from AnalyzeSoundFile. It can only work on pre-processed WAV data. It requires the sample rate and number of samples for the WAV data being passed into the method. It returns a float with the Local Jitter score in percentile form, and sets the MyPitch integer with the Fundamental Frequency, found by using autocorrelation techniques.

5.36.2.4 `void LinLeastSquares (double *x, double *y, int n, double *Alpha, double *Beta)`

This method is used by the CPPCalc method to find the Linear Least Squares fit regression line across the cepstrum of the audio file. It sets the alpha and beta values for the fit, which correspond to the y-intercept and slope of the line.

5.37 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/main.m File Reference

```cpp
#import <UIKit/UIKit.h>
#import "IDAAppDelegate.h"
```

Generated on Mon Apr 8 2013 21:53:11 for iOS-based Voice Analysis Application for Speech Therapy by Doxygen
Functions

- int main (int argc, char *argv[])

5.37.1 Function Documentation

5.37.1.1 int main ( int argc, char * argv[] )

5.38 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Usage.h File Reference

#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>

Classes

- class Usage

5.39 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/Usage.m File Reference

#import "Usage.h"

5.40 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/VocalTests.h File Reference

#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>

Classes

- class VocalTests

5.41 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/NewVocalTest/VocalTests.m File Reference

#import "VocalTests.h"
5.42 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/UIView+FindFirstResponder.h File Reference

#import <UIKit/UIKit.h>

Classes

• category UIView(FindFirstResponder)

5.43 /Users/robertpfister/Dropbox/Documents/Xcode Projects/Eva’s Projects/VocalTest Versions/NewVocalTest/UIView+FindFirstResponder.m File Reference

#import "UIView+FindFirstResponder.h"