### Introducing SpokenText Reader

The SpokenText Reader prototype is a hybrid iOS native application. The user interface was written in JavaScript and HTML which was then bundled together using Apache Cordova[[1]](#footnote-1) and deployed to an iOS device.

SpokenText Reader uses JavaScript Application Program Interfaces (API) provided by Cordova to talk to the phones native API’s to allow for recordings to be downloaded to the device along with supporting the capturing of audio, images and videos, which can be used as notes within SpokenText Reader

SpokenText Reader allows users to listen to recorded audio, take notes, bookmark key points in the recording, if you leave a recording and reenter the recording it starts from where you last left off and it provides a form of annotation to “mark-up” audio files in a similar way. Figure 1, shows how people who study with print, highlight, underline and write notes in the margins.

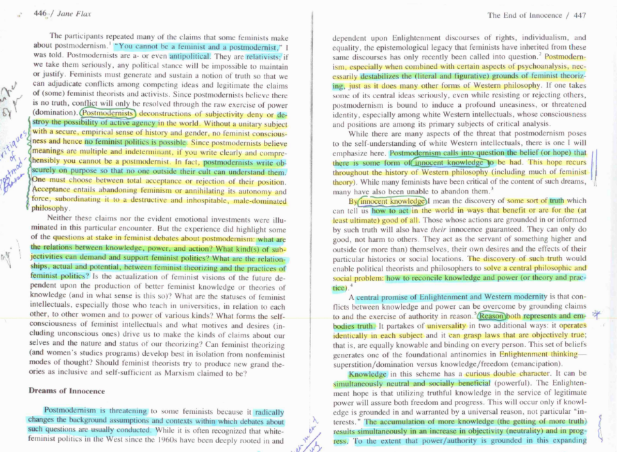


Figure : Example of text being highlighted on a page while studying

Based on the example page shown in Figure 1, how would people who need to study using the audio version of this text highlight, take notes and annotate the audio file? This thesis project involved evaluating a potential solution to this problem.

### Some challenges faced when designing SpokenText Reader

When you are listening to an audio stream you have many challenges when it comes to navigation and note taking. If you need to leave the application you are utilizing to listen to recordings to take a note this can break your sense of flow (Csikszentmihalyi, 1975) and in addition to this you have the added complexity of developing a means to link the note back to the point in the recording to which it relates. Highlighting parts of a recording to use for citations at a later time can be difficult when you only have an audio stream.

Since one of the target groups of SpokenText Reader is users with low or no vision, its interface and the design patterns used had to be fully accessible to both people who can see the interface with a moderate level of clarity and it needed to work for people who could not see the interface and relationships between controls at all. This was a huge challenge when it came to designing the user experience for SpokenText Reader.

With SpokenText Reader, users can take notes and annotate sections of audio to sight later when talking in class, writing papers or giving presentations. And hopefully in so doing, help in a meaningful way, to reduce the gap between print disabled students and their non-disabled peers.

### SpokenText Reader prototype history

Starting in the summer of 2012, in preparation for my first master’s class I started working on a very early prototype of SpokenText Reader. This early prototype then evolved during this first course. Upon using this initial prototype for a period of time and reflecting on its strengths and weaknesses, I modified and made improvements to how the prototype functioned. I then used the revised prototype for my own coursework, which I used on a daily basis to accommodate myself and make it possible for me to fully participate in my Masters courses.

From time to time the refactoring I had to do to add new features or refactoring required by Apple to enable the application to be installed on a newer version of Apple’s mobile operating system iOS caused the application to break. Thus making it not available to me as a tool to accommodate my needs. During these times I really missed having access to SpokenText Reader, which provided me with encouragement that I might be creating something that others might find of value.

In 2014, I took a directed study where I explored and expanded the capabilities of the prototype, adding the ability to take audio, image and video notes in addition to the text notes which had been first developed.

It was at this point that I had the idea to merge all of the note types into one note screen, whereas originally I thought that it would be best to take a text note separately from an image, sound or video note. Seeing that there was a lot of similarity between the note types and that you would probably want to name or describe a clip, image, audio or video note, I concluded it would be best to have one note type. This change was implemented in my next round of iteration on the prototype and it greatly simplified the user interface.

#### Initial prototype screenshots

The following screen shots are from the initial prototype I created. At this point you could only take a text based note synced to a point in the recording and the annotation feature had not yet been added.

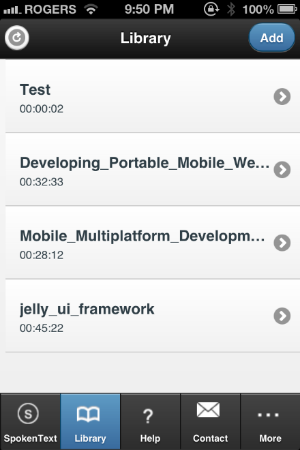
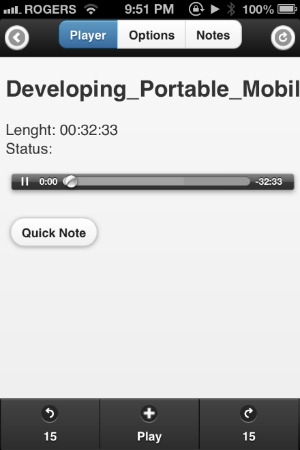
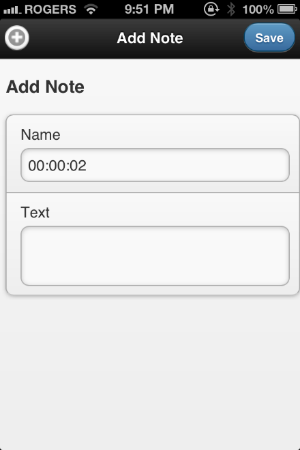
  

Figure : Initial prototype screenshots

#### Second major revision prototype screenshots

This revision saw some major changes to the user interface. SpokenText Reader now contained the ability to annotate audio files, the “Swipe me” controller was added which lets you control the key features of SpokenText Reader using gestures and the menu system was updated to be more in keeping with modern mobile phone applications.

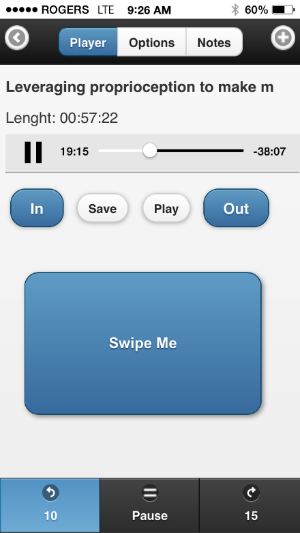


Figure : Second major revision prototype screenshots

#### Screenshots of final revision of prototype before changes made for usability testing

This revision had only minor changes. The play button at the bottom of the screen was removed. And the rewind 10 seconds and fast forward 15 seconds buttons were enlarged to fill the space to create a well a balanced aesthetical look.

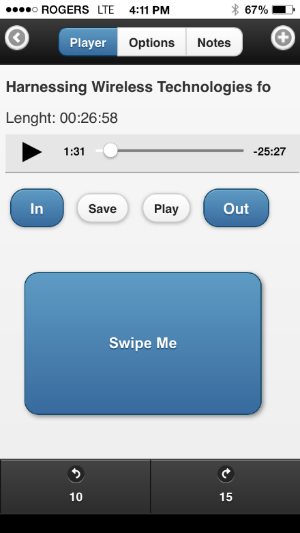


Figure : Final revision of prototype before changes made for usability testing

#### Technology used to create the prototype

SpokenText Reader is a hybrid application. It uses Apaches Cordova framework which allows developers to create cross platform HTML and JavaScript based applications which can be installed onto a user’s mobile device or placed into an App Store and downloaded to a user’s device. Applications built this way can look very similar to a native application built for a given platform. Cordova supports many mobile platforms, but for this project just the functionality relevant to Apple iOS was used.

JQuery and JQuery mobile JavaScript libraries were used to give the interface a more native look and feel. Numerus jQuery based helper libraries were used to ease and speed up the development of the prototype.

Once the HTML and JavaScript code was ready to be tested on an iOS device it was placed into a Cordova project and compiled using Xcode running on an Apple Mac Book Air. Xcode took care of installing the compiled application on to the iPhone used for testing during development.

REST API’s were developed for SpokenText.net. These were specifically designed for and to be used by SpokenText Reader. These API’s allowed SpokenText Reader to access a user’s account and determine which recordings were ready to be downloaded to SpokenText Reader via the download functionality built into SpokenText Reader.

Source code for the project was kept in a GIT repository during development to allow for easy role back and code development.

#### Why I used these technologies to create the prototype

When I started this prototype in 2012, Objective C was the language Apple officially supported for iOS developers to use, but it is very complex, time consuming to learn and it does not lend itself to quickly trying out ideas and making changes.

With a background in web development technologies such as HTML, JavaScript and PHP, it was much easier to work with these technologies. It is for the aforementioned reasons that I chose to build the prototype using programing languages, which I was already familiar with.

#### Prototype development process

One main benefit of the approach taken to develop this prototype was that much of the development could be done just using a text editor and the Google Chrome web browser. There was no need to compile the application every time you wanted to determine if the last piece of code you added was functioning as intended. This saved a lot of time during the initial stages of developing the prototype.

When it came time to developing the code to download recordings or to take pictures, video or audio to be added to a note the process changed and the code had to be compiled every time using Xcode. This was a very slow and time consuming process, which was, made worse by the lack of integrated support for HTML and JavaScript based Cordova applications with in Xcode.

#### Strengths of this prototyping technology

This prototyping approach based around a hybrid application had a lower initial learning curve than learning Objective C Apples language for native application development.

In many ways it was very fast to try out new ideas since I could find libraries which could be easily integrated and tested in a web browser without having to compile the application over and over again.

#### Weakness of this prototyping technology

The development tools used to develop and build the prototype were not mature. This led to a very slow development process since all of the various tools required to develop and debug the application were not integrated, as is the case for a true native iOS application written in Objective C.

Debugging issues and fixing bugs was very time consuming and sometimes very difficult since it was not always easy to determine which part of the code was not functioning as intended. There was limited to no support for JavaScript debugging in Xcode resulting in the code containing a lot of log messages. These log messages were then used for debugging of the code.

Occasionally when all else failed Safari was used to debug the JavaScript code. But this could only be done after the application was first compiled using Xcode and running in the iOS simulator. All of this took time and made the code, test and fix process time consuming.

#### Prototype changes made before testing began

Before I could conduct the usability testing sessions using the prototype I needed to make some updates to it.

Over the last four years, while working on the prototype, each time I changed phones I had to update the prototype to compile and deploy onto the new phone with a new version of iOS. This required upgrading it to work with and compile under Cordova 3.5 and iOS 9.

Since the iPhone 6 Plus to be used during the testing session had a larger screen size than the iPhone 4s and 5s used for previous versions of the prototype, I customized the interface to look and function well on a larger sized phone.

Once I had SpokenText Reader running again on the iPhone 6s Plus, which would be used during the usability testing sessions, a few weeks of trial and error testing was necessary to ensure features required for the usability testing sessions still functioned. Upon successful completion of this phase, I was ready for usability testing to begin.

Library

Player

Notes

Add Note

Primary SpokenText Reader Screen flow

View Note

Edit Note

Figure : Primary SpokenText Reader screen flow diagram

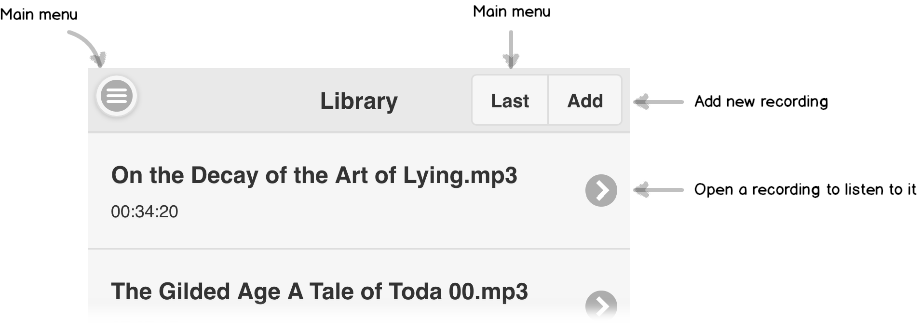


Figure : Annotated Library screen

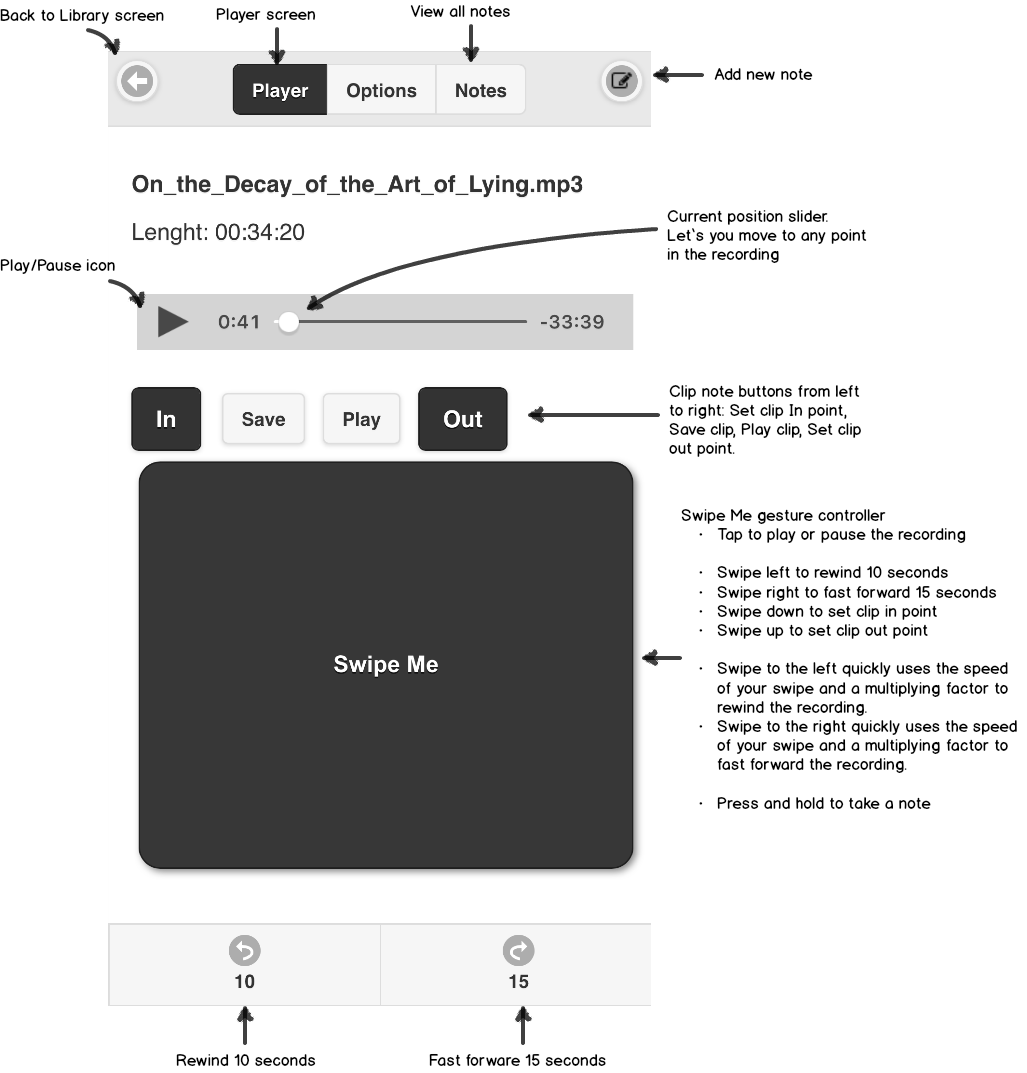


Figure : Annotated Player screen

#### Task walkthrough

Task 1, involved users using the screen shown in Figure 8. This screen lets a user switch between recordings or select a recording to listen to. During the test session the participant was placed on this screen then asked the task. The goal was to have them tap on one of the recordings shown in the list. Once they did this the player interface would appear and the recording would start to play automatically.

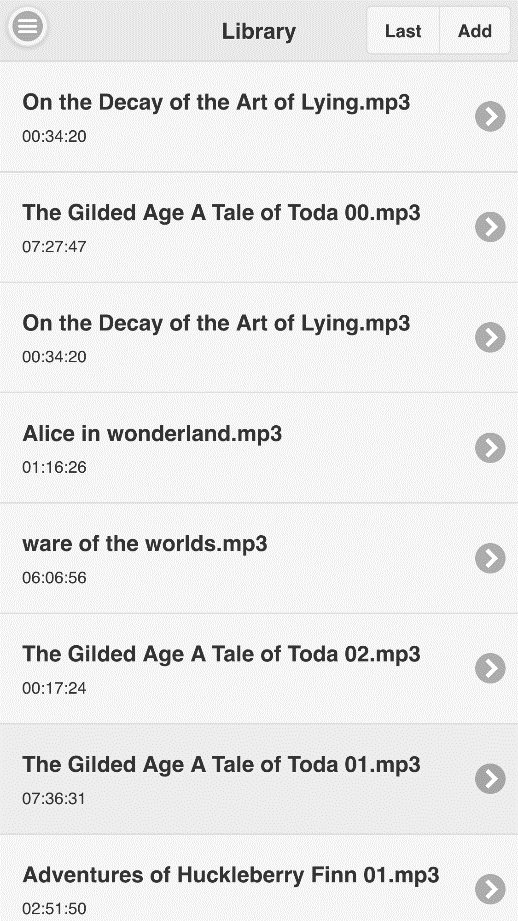


Figure : Library screen of SpokenText Reader

Task 2, starts on the screen shown in Figure 9. This screen is the main screen used to listen to audio recordings and initiate other features of SpokenText Reader. The version shown in Figure 9 demonstrates how SpokenText Reader would appear when a recording is currently being played. The goal of this task was to have them press the pause icon or tap with one finger on the Swipe Me gesture controller.

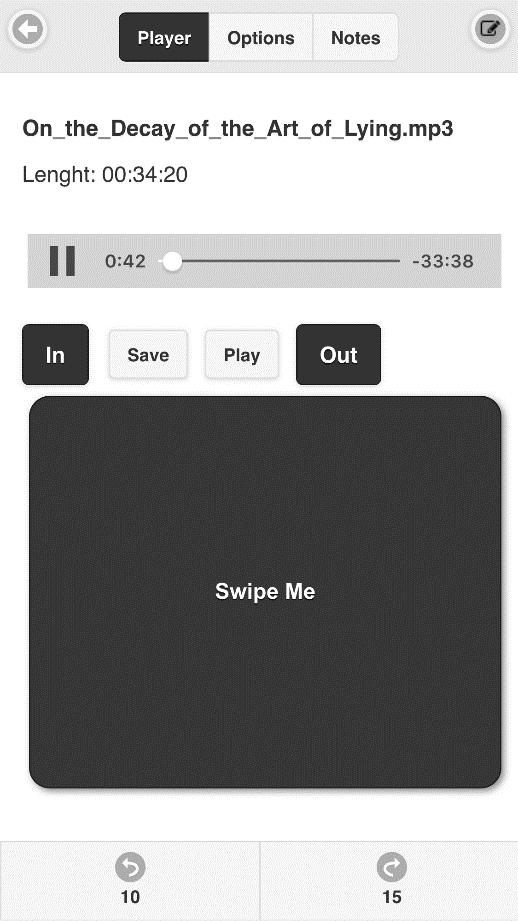


Figure : Player screen with a recording being listen to.

Task 3, involve testing how easy participants found it to play a recording once it had been paused. The participant was asked to complete this task based on the recording they used for task 2. The goal was to have them press the play icon or tap with one finger on the Swipe Me gesture controller, both of which can be seen in Figure 10.

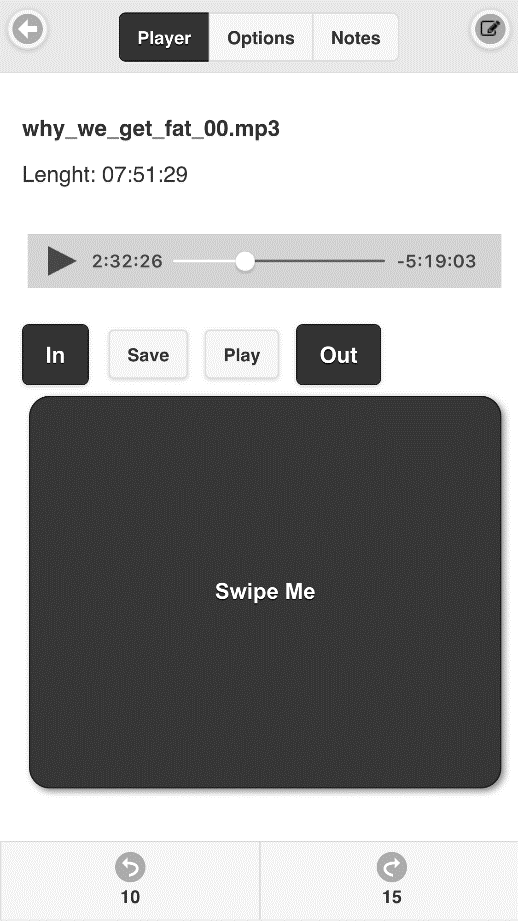


Figure : Player screen showing a paused recording

Task 4, involved asking participants how they would rewind a recording which was currently playing. There were many ways they could have done this: press the rewind 10 seconds button, swipe left on the Swipe Me gesture controller or slide the current position slider to the left. All of these controls can be seen in Figure 9.

Task 5, involved asking participants how they would fast forward a recording which was currently playing. There were many ways they could have done this: press the fast forward 15 seconds button, swipe right on the Swipe Me gesture controller or slide the current position slider to the right. All of these controls can be seen in Figure 9.

Task 6, involved setting an in point for a clip note. The goal was to have users click the Set clip in point button. By setting a clip in point, users can set the point they want the highlight or audio annotation to start from. This button can be seen in Figure 9.

Task 7, involved setting an out point for a clip note. The goal was to have users click the Set clip out point button. By setting a clip out point, users can set the point they want the highlight or audio annotation to end. This button can be seen in Figure 9.

Task 8, asked participants how they would take a note when listening to a recording. The goal was to have participants tap the Add new note icon in the top right corner of the screen, enter a note and press the save button in the top right corner of the Add note screen. The following figure shows both the Player and Add Note screens as they appeared during the testing sessions.

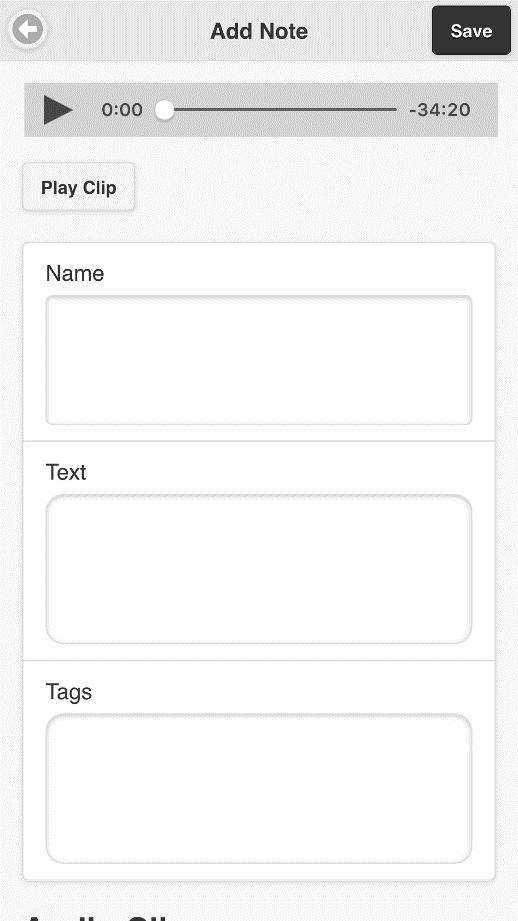
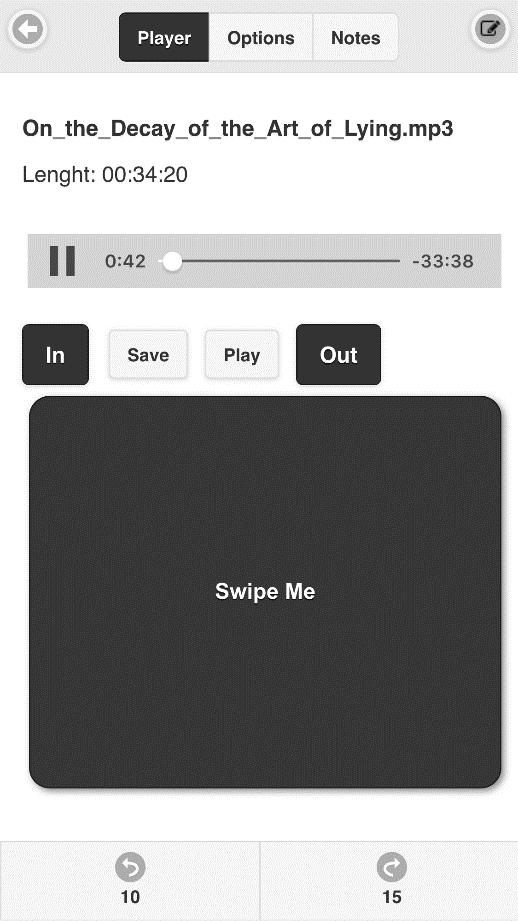


Figure : Shows the main player screen (left) and the Add new note screen (right) which is shown after the Add new note icon is pressed on the player screen.

Task 9, asked participants to review the note they had taken in task 8. The goal was to have participants tap the Notes button at the top of the screen shown in Figure 12 , which displays a list of all notes that have been taken for a recording. The participant could then tap on a note in the list to view it.

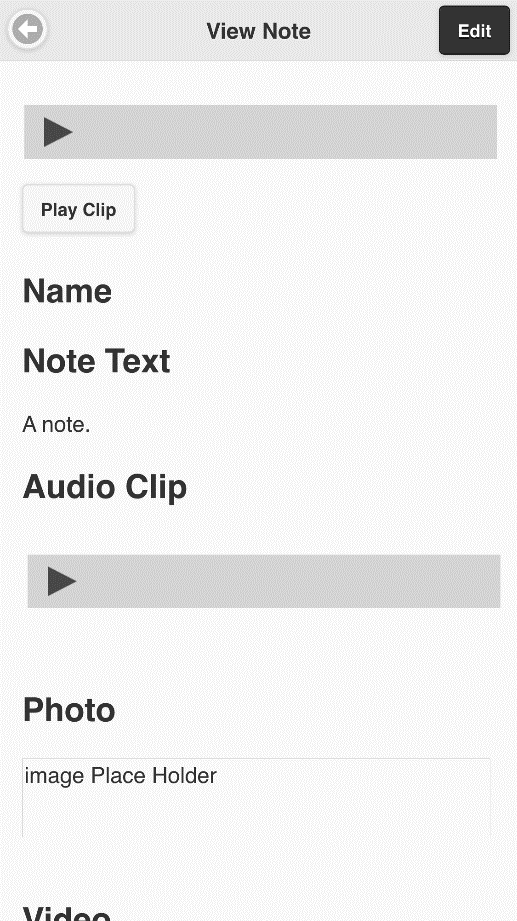
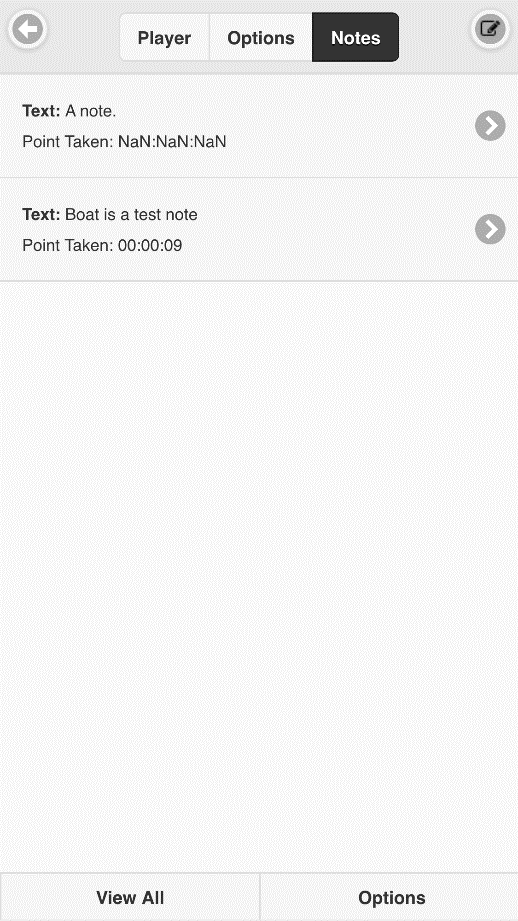


Figure : View all notes screen (left) View note screen (right)

Task 10, builds on task 9 but instead of viewing a note they are asked how to edit the note they viewed in task 9. The goal was to have them click the Edit button in the top right corner of the second screenshot in Figure 12. After tapping the Edit button, they would see Figure 13.

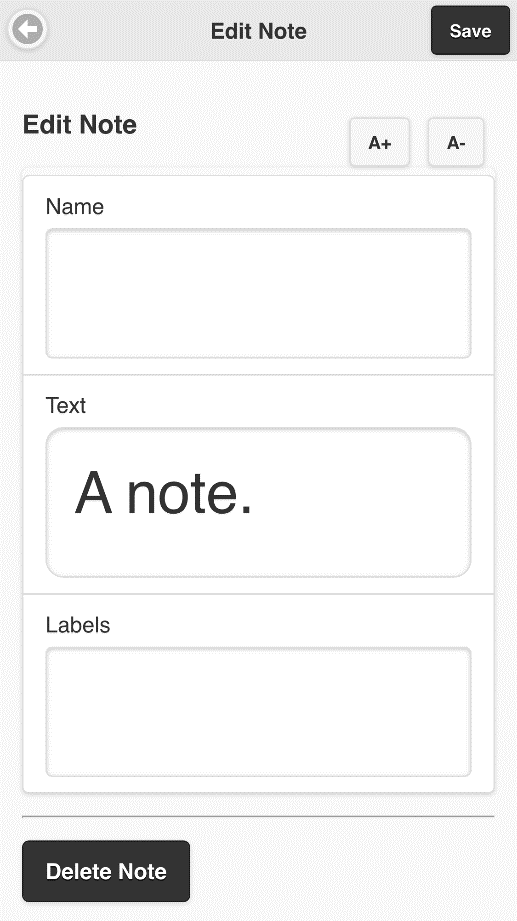


Figure : Edit note screen

### Close out message said orally to all participants

Thank you for taking part in this usability test. If you have any questions you want to ask me, please ask them now. And thanks for your participation.

### Device used for testing

An iPhone 6 Plus running iOS 9 was used by all participants during the test. SpokenText Reader was installed on the device. I launched it and navigated the application to the Library screen.

This device was used for the testing since it was the one I was most familiar with and it was the platform used by the majority of participants who owned a mobile phone. This is important as screen readers used by people with visual impairments vary between the various platforms, so it was important to use a device which could accommodate these participants and one for which they would also be quite familiar with.

1. Open source framework to develop mobile apps with HTML, CSS & JS, which supports multiple platforms with one code base. https://cordova.apache.org/ [↑](#footnote-ref-1)