



GMIT

INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

BSc in Software Development – Year 3
Professional Practice in IT

Sound Hub

Enda Goggin – G00369903

Owen Kelly – G00366614

Supervisor – Martin Kenirons

GitHub Link - https://github.com/OwenKe11y/PPIT_Project

Abstract

This document serves as the **documentation** for the Professional Practice in IT project. In this document, details of the application are explained as well as the design mentality and approach that underwent the development of this project. Headers outlining each aspect of the project are detailed below. The team behind the development of this project had decided that a **mobile application, used as a tool for musicians** was the idea both developers were interested in making.

This document is written in a formal and non-personalized format unless stated otherwise.

Table of Contents

Abstract.....	i
Table of Figures.....	v
Introduction (Owen)	1
Project Aims	1
Specification (Owen).....	2
Recording	2
Saving Clips.....	2
Tuner	2
Quick Navigation	2
User Specific Data	2
Technology Implemented (Owen)	3
React Native (Owen)	3
Expo CLI (Owen).....	3
Recording (Enda).....	4
Firebase (Owen & Enda)	4
Authentication (Enda)	5
Storage (Enda).....	5
Architecture of the Solution (Owen).....	6
Sound Hub Home Page	6
Main Page	6
Tab Bar	6
Clips.....	6
Recording Pages.....	7
Tuner	8
Welcome Pages.....	9
Welcome Front Page.....	9

Sign Up	9
Sign In.....	9
Software Development Life Cycle (Owen & Enda)	10
Project Start (Owen)	10
Initial Tester App (Owen).....	10
Project Structure (Owen).....	11
Design (Owen).....	11
Firebase Implementation (Enda)	11
Clip storage (Enda)	12
Audio Implementation (Enda).....	13
Limitations and Known Bugs (Enda)	14
The Clip Data	14
Media Player	14
Recorder.....	14
Sign-in bug.....	15
Performance	16
Keyboard	16
Testing Plans (Enda).....	17
Tests for Current Build	17
Sign in/Register	17
Soundhub	17
Recording	17
Tuner	17
Future Feature Test Plans	18
Music Recognition.....	18
Better Tuner	18
Ear Trainer.....	18
Recommendations for Future Development (Enda).....	19

Possible solutions for bugs.....	19
Clip Data.....	19
Sign In.....	19
Performance	19
Future Development.....	19
Other Users Clips.....	19
Music Recognition.....	19
Better Tuner	20
Ear Trainer.....	20
Conclusion (Owen).....	21
References	22
Resources.....	23
Aggie.io	23
Discord	23
LucidChart	23
Expo Documentation	23
Firebase Documentation.....	23
Reaper	23
Boogex	23
Shazam.....	23

Table of Figures

Figure 1; The Firebase model.....	4
Figure 2; Firebase Console	5
Figure 3; Sound Hub Home page and menu	6
Figure 4; Delete a clip	7
Figure 5; Recording Process	7
Figure 6; Tuner page	8
Figure 7; Sign Up, Welcome and Sign In pages	9
Figure 8; Original ideas page	10
Figure 9; Project Hierarchy	11
Figure 10; Firebase User accounts	12
Figure 11; Firebase storage.....	12
Figure 12; Reaper Workstation with Boogex	13
Figure 13; The user gets sent from the login page (left) to the sign in page (right) and back again	15
Figure 14; Keyboard bug	16

Introduction (Owen)

For the Professional Practice in IT project, a piece of software must be designed, developed, and deployed to demonstrate the best practices that the student has learned in the previous semesters of the course. The student or pair of students must deliver a piece of software in a timely and driven manner.

The documentation is intended to provide different perspectives on the project. It is intended to provide both user and developer with background information for the system.

These perspectives can be broken down to the following:

1. The **user is interested in what the project does**, how they interact with it and what the system looks like. The requirements form part of this section of documentation and will provide detail on the functional components of the system.
2. The **developer is concerned with how the application is implemented**. This concentrates on the technical aspects of the user interactions rather than what the system looks like. As part of the development process, the documentation allows the development team to scope out the different technologies that are available to provide a solution to different aspects of system implementation. These technologies can be compared, and decisions can be made on the merits of each, as to whether it is appropriate to use in the development of the new system.

Project Aims

For the documentation, the following aims must be met and substantially discussed and documented:

- To provide a functional breakdown of the system to be developed and to showcase the Architectural Overview of the system.
- To identify and compare the possible technologies, justify the choice of technology and to explain how it will be used in the implementation.
- To give a detailed description of any database that will be used with the application.
- To clearly show the screen layout providing a blueprint for how the application will look to the end user.

Beside each header of this document, the name of the developer will display to show who had worked on that section.

Specification (Owen)

The developers were very liberal in the specification of this project. Some features for the app were cut early in development due to either limitations in technology or knowledge however, there were always a few key aspects the developers made sure that the user could achieve. Each of these aspects are detailed below.

Recording

The app must include some form of recording functionality where the user can record themselves using the microphone from their phone. Recording must be quick and seamless, where the user can start recording at any time from the touch of a button. It should be clear to the user that the device is clearly recording, and it should equally be just as obvious when the recording has stopped.

Saving Clips

The user will have the option to save their recordings and play them back. They should also be able to label them and add a description to differentiate one recording to the other. The clips should be displayed on the main page for easy access and shown in a list format. Tapping on an item plays the recording for the user. If a user doesn't like a recording, they will be able to delete it.

Tuner

The user will have access to a pitch pipe style tuner [\[1\]](#) for their instruments. This tuner will play the notes of E standard tuning on a guitar (EADGBE) and will repeatedly play a note until the user presses stop. This pitch pipe is used so the user can tune their instrument, such as a guitar, to the sound of a note playing repeatedly until eventually their instrument plays the note that can be heard from the app. The user should be able to press a button corresponding to the note they want, and listen to that note repeatedly until a "stop" button is pressed.

Quick Navigation

The user should have seamless navigation between all aspects of the app. This will include a tab bar that supports swipe functionality which is expected of most modern-day applications. Buttons should be responsive; they should be recognizable, and headers should be displayed to clearly show where the user is on the app. Options to log out and sign in should be easily accessible and navigable.

User Specific Data

The user's recordings should be tied to an account the user creates at the first launch of the app. Their recordings should be specific to their account and is only accessible by their account. If the user were to sign into another account, the recordings from the first account will not appear.

Technology Implemented (Owen)

To reach the user specifications outlined for the app, in-depth research was conducted to find the most fitting technology to reach those requirements. Outlined below is the developer's thoughts on what technology they picked for each requirement, why they chose that technology and how it affects the application and user specifications.

React Native (Owen)

From an early stage, we had decided that we wanted to make a mobile application that was cross platform and could be ran on any device. We considered the prospect of using Xamarin but ultimately, we decided it was the best choice for what we wanted to create.

React Native is very much supported in this time and has extensive documentation, video tutorials for seemingly anything, and libraries and dependencies created by its community. We also wanted to test ourselves and try something new. During our second semester of this course, we had previously experienced what Xamarin had to offer and were ultimately disappointed with the framework. Not only this, but in May of 2020, Microsoft had announced Xamarin would be deprecated.

This told us that React Native was the best choice for what we wanted to create. On top of extensive support and a large community, both of us had prior experience using JavaScript. We felt a lot more comfortable writing in a language that we both enjoyed.

Expo CLI (Owen)

Expo is a free and open source project which provides developers with a toolchain built around React Native, that helps building native iOS and Android apps faster using just JavaScript and React.[\[2\]](#)

As developers completely new to React Native, we decided that using Expo and its tools would be best fitted for running, creating and testing our application. Expo has a reputation of being the starting point for a lot of developers who wish to begin their journey on React Native because of its easy and quick creation of mobile apps.

Expo is, however, limited on the number of third-party libraries and extended app functionality. If we wanted access to more complex libraries, we would have to implement our other option, React Native CLI. The main downside of this is setting up independent build chains for both platforms on our systems. For Android we would need Android Studio and for iOS we would need to setup Xcode so that we could build and test on different devices. As novices to React Native, we decided to go with the quick and easy deployment of apps using Expo despite being limited to the Expo ecosystem.

Recording (Enda)

The recording and playback functions are handled by the Audio package from Expo. The documentation for expo was very useful and had functions for recording, stopping the recording and playing the recording. These were easily implemented into the project and worked with the front end straight away. The recorder also returned a URL to the temporarily stored file that could be used to send straight to firebase.

Firebase (Owen & Enda)

Firebase is Google's mobile application development platform that helps you build, improve, and grow your app.[\[3\]](#)

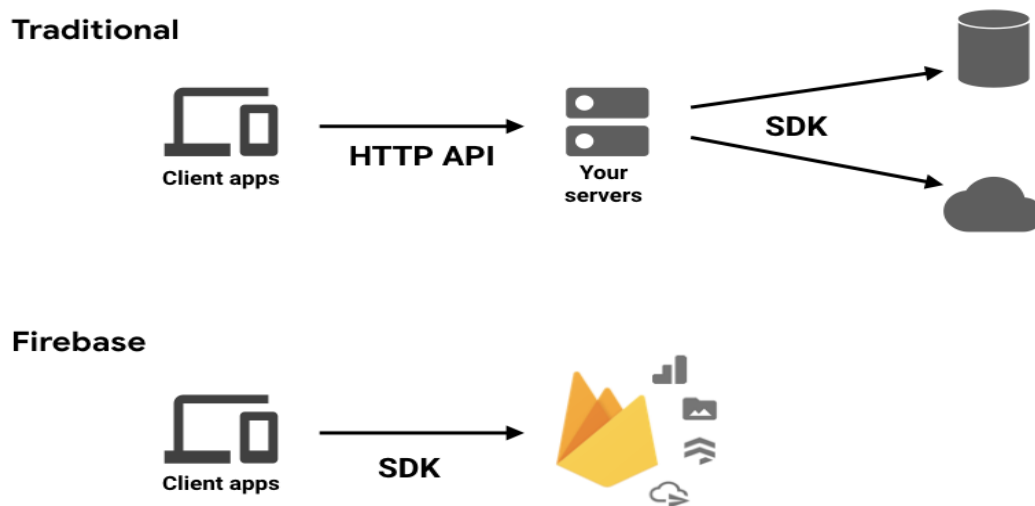


Figure 1; The Firebase model

Originally for this project, we had decided to use MongoDB as our backend as we thought the implementation for recording storage and sign in authentication could be done easily. However, our lecturer recommended Google's Firebase as it provided a lot of those features straight out of the box without us building a database from scratch. This was the logical choice because of Firebases' strong and dedicated community.

Using Firebase meant that we could spend a lot of time and effort into the front end of our app and our attention wasn't divided between two work environments. Firebase is also very compatible with React Native and the Expo CLI [\[4\]](#) as compatibility issues were a big deciding factor in our decision to choose Firebase.

There were a few limitations with Expo that made some Firebase functionality incompatible but the sections of firebase that we needed like Storage and Authentication worked, so it was an easy integration.

Authentication (Enda)

The authentication for firebase let you use a variety of different sign in methods like social medias and normal email and passwords, but we decided to just stick with a simple email and password. When the app launches it prompts you to log in or register, these are both done with the `firebase.auth()` methods and all that's passed in is the email and password. The logout function also uses `auth()` but it doesn't need and parameters.

Storage (Enda)

We use the storage for holding the users sound clips and the clips of the tuning notes. The recorded clips URI is first turned into a Blob and then pushed to the Clips folder on the Firebase storage using the `put()` method that Firebase uses. This also takes in a metadata file which contains the users name and the description they entered before uploading the clip.

The clips are then pulled back from the storage using a `forEach` loop on each item in the clips folder. Firebase has a `getDownloadURL` and a `getMetadata` function that we used to get the clip info and push it to an array if the name of the user matches the name in the metadata of the file. These are then displayed to the user. The delete function sends the name of the clip into the `delete()` Firebase method and the clip with the matching name is then removed from the storage.

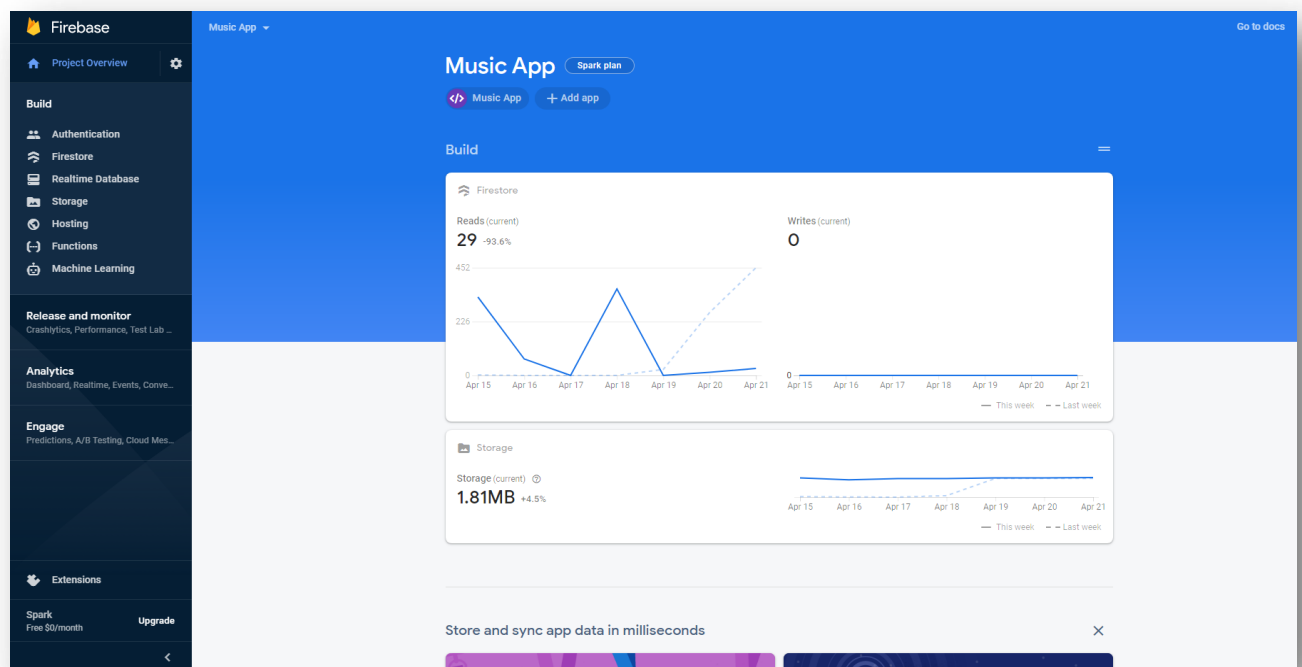


Figure 2; Firebase Console

Architecture of the Solution (Owen)

The following showcases the different elements of the mobile app and how they were implemented according to the user specifications. An accompanying screenshot will feature beside each component and will be described in detail.

Sound Hub Home Page

Main Page

This is the main page of the Sound Hub application. Here, users will be able to see any clips that they have recorded. The user can log out of their account by tapping the log out button on the top right-hand corner of the header. A floating menu is displayed at the bottom right where the user can create a new clip, or refresh the page to check for new clips

Tab Bar

As per the user specification, the user should be able to seamlessly navigate the application quickly and efficiently. The tab bar features on the bottom of the screen showcasing the 3 available tabs that can be navigated. Tabs can be swiped to and the displayed screen will be highlighted in orange.

Clips

The recorded clips can all be found on the main page. The clips can be tapped to play the selected clip, doing so will playback the recording the user has made. Holding the tap on a clip displays a pop up asking the user if they would like to delete the selected clip. The clips feature a title, a description and- the name of the user's account.

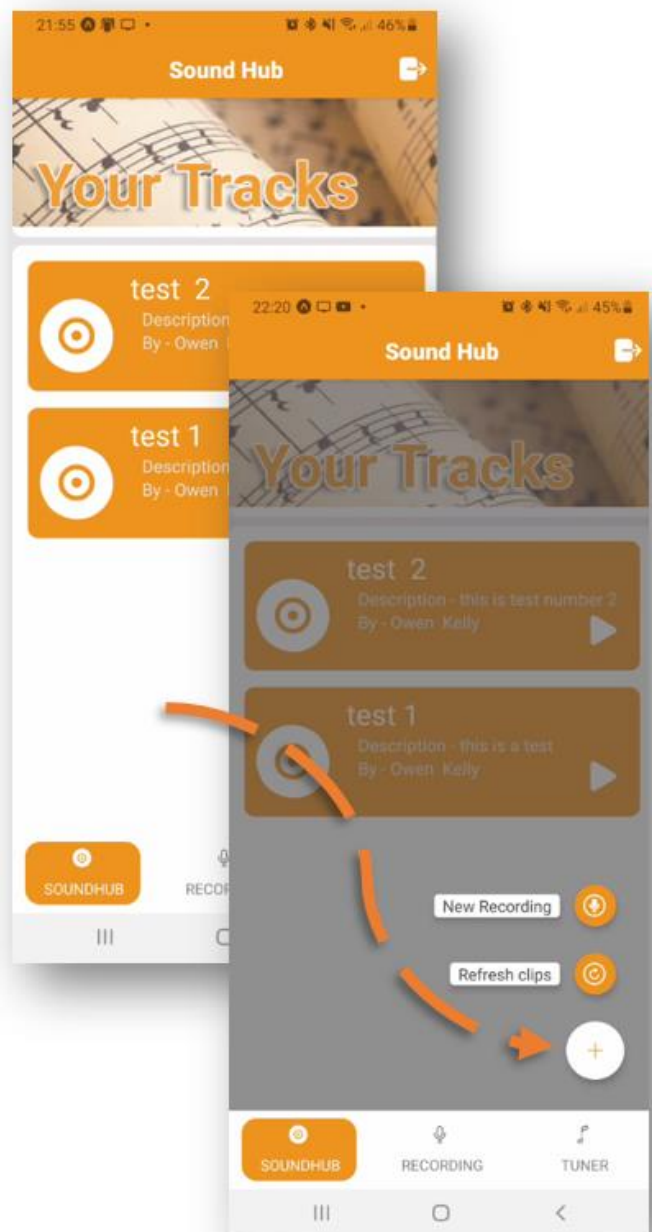


Figure 3; Sound Hub Home page and menu

All clips are displayed in alphabetical format and appear in large tappable boxes so that users don't find it difficult to play a clip.

The clips are account based, meaning if a user created a new account only the clips they had created on that account will display.

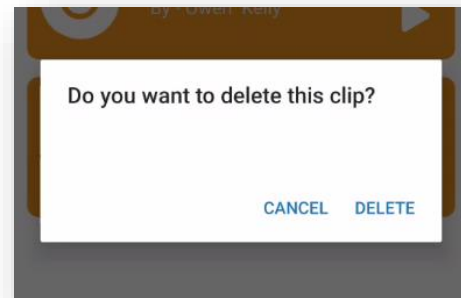


Figure 4; Delete a clip

Recording Pages

Starting a new recording can be done at the touch of a button. The main recording page features just one play icon button which the user can tap to begin the recording process. Once pressed the icon will change to a stop icon as well as a moving waveform at the top of the screen. Once the stop button is pressed, a screen will appear prompting the user to enter a name and a description for the new recording. If the user doesn't like their recording they can simply cancel and begin the process once again.

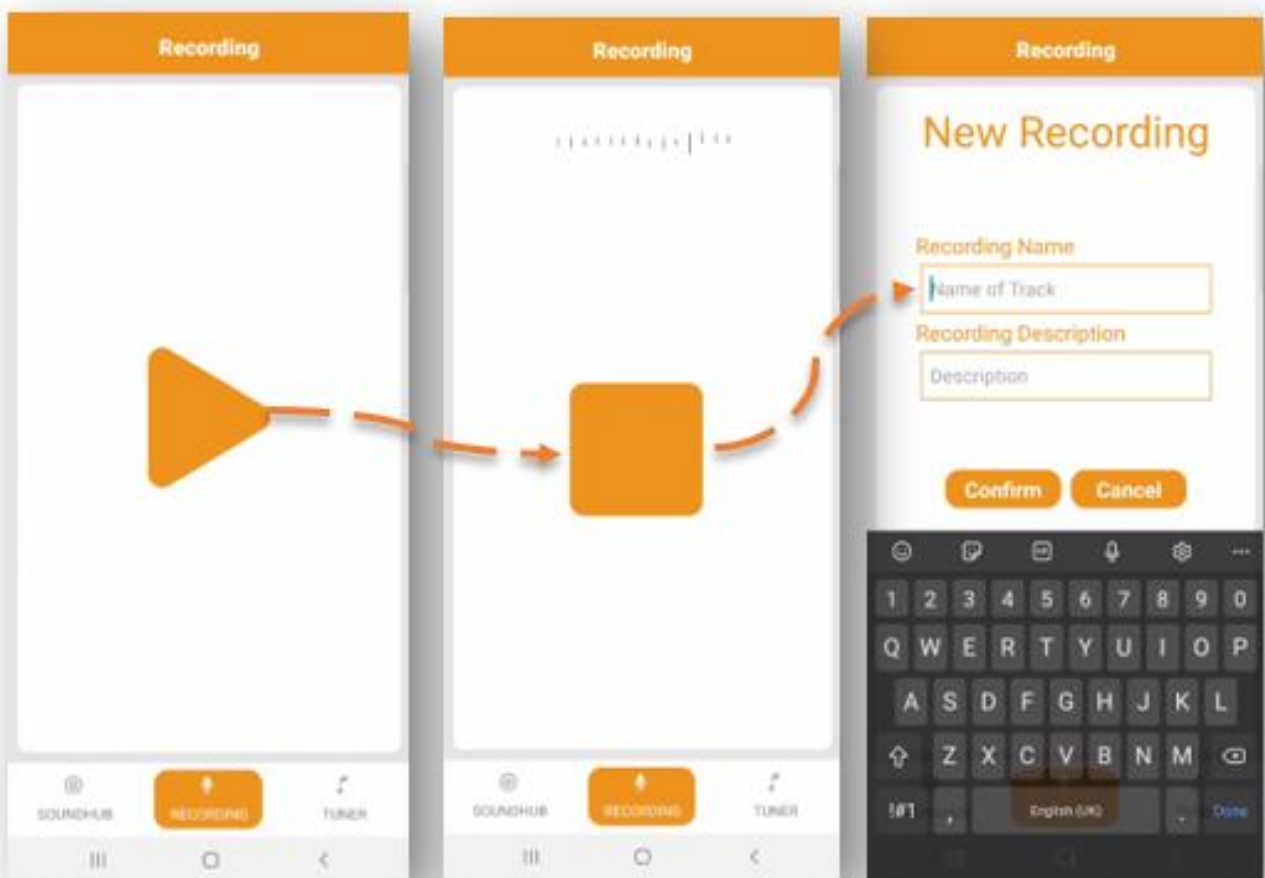


Figure 5; Recording Process

Tuner

The tuner features the silhouette of a Fender guitar head. Beside each tuning peg features a button with the corresponding note for that string.

This tuner will play the notes of E standard tuning on a guitar (EADGBE) and will repeatedly play a note until the user presses stop. Logging out is also possible from this screen by means of the log out button at the top right-hand side of the header.

The musician can listen for the note they've selected and play that note on their own instrument to ensure that their instrument is tuned.

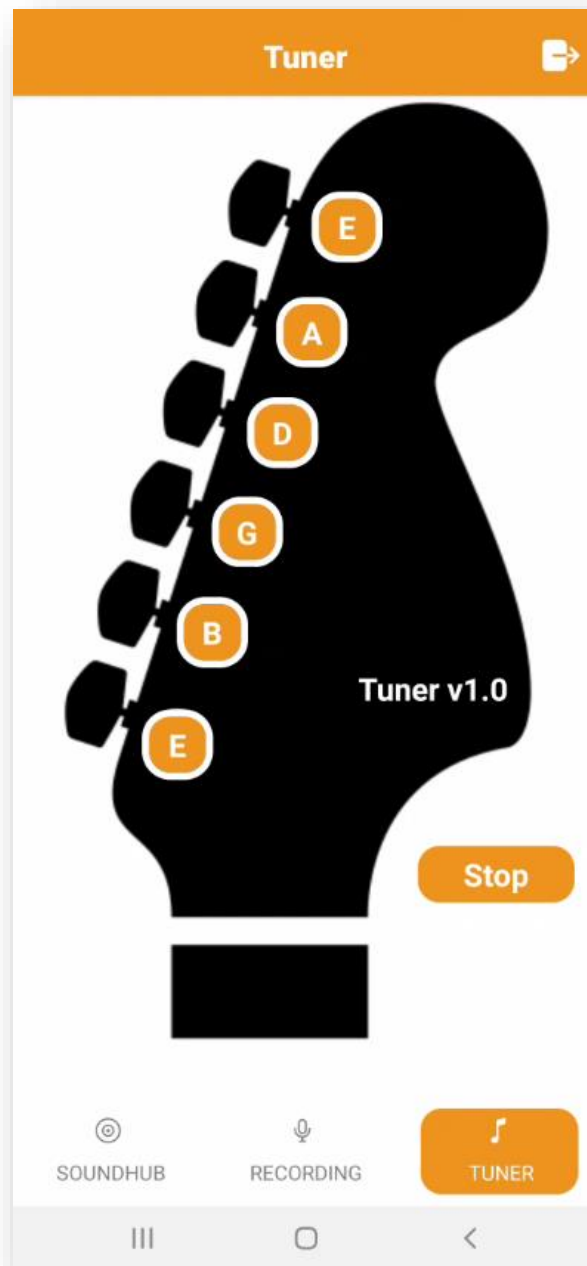


Figure 6; Tuner page

Welcome Pages

Welcome Front Page

The welcome page is the first thing the user sees upon opening the application. From here, users will be able to choose to sign up to the application or to sign in if they already have an account. This can be accomplished by tapping of the two buttons shown on the screen.

Sign Up

Here, users will be asked to enter their first name, last name, email, password and to confirm their password. The input boxes have checks tied to them to ensure the user enters the correct information. The user must enter the proper format for an email e.g. "someone@something.com" and they must enter a password that has more than 6 characters. Upon inputting the right information and pressing sign up, the user will be redirected to the sign-in page.

Sign In

Once the user has created an account, the user can now sign in by inputting their email and password. Once this has been completed the user will then be redirected to the sound hub main screen.

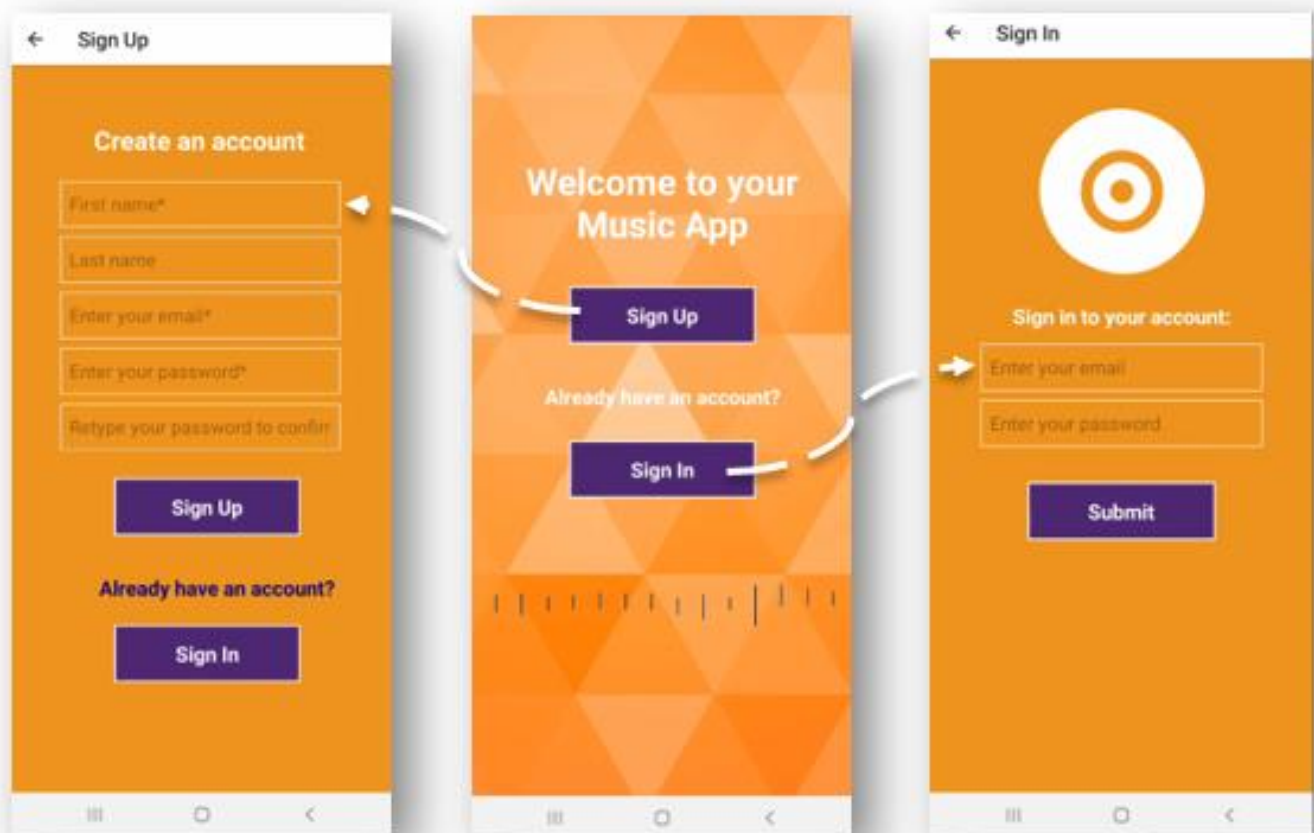


Figure 7; Sign Up, Welcome and Sign In pages

Software Development Life Cycle (Owen & Enda)

In this section, the developers will discuss the process of completing each aspect of the project, providing a functional breakdown of each component. Their thoughts are outlined below accompanied by screenshots to illustrate their ideas. The developers both decided to play to their strengths when approaching this project, with Owen handling the design of the application and Enda working with server-side functions and functionality.

Project Start (Owen)

In the beginning of the project, we began with a simple brain storming session to figure out what kind of technologies we would utilize. We used [a free, online collaborative work space](#) to draw out our ideas like a traditional whiteboard. Using [a free calling application](#), we called while we threw out any ideas that would stick.

We were torn between making a music or a workout application as they are two topics we both thoroughly enjoy but we decided that we could add more interesting and creative features to a music application.

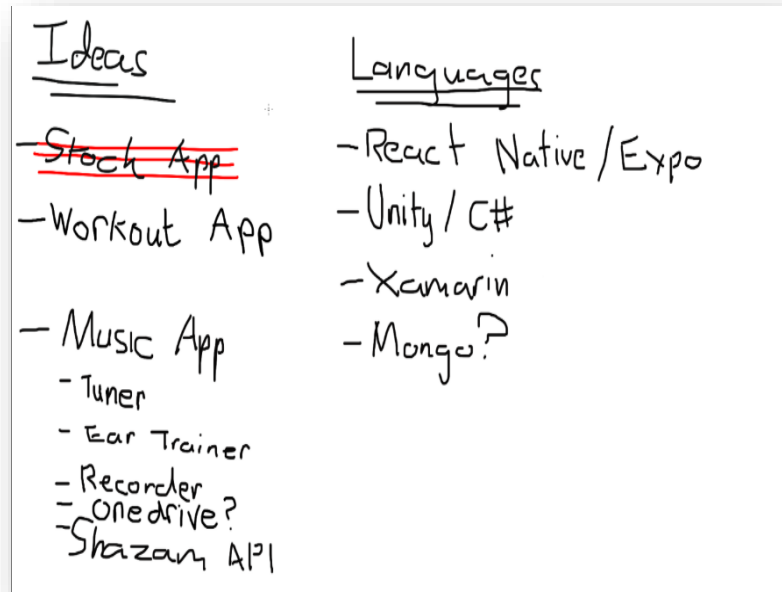


Figure 8; Original ideas page

Initial Tester App (Owen)

After deciding that we would use React Native and Expo for our project, our priority was to get familiar with the language as well as the command line interface for Expo. We created a separate repository to store a sample project to. This would act as our test ground for getting to know the framework and to make sure we don't push anything that was unfit for our project repository. The test repository can be found [here](#). A wiki was constructed with a step by step guide on how to create any amount of test projects if we needed to.

Project Structure (Owen)

Once we were familiar with the tools React Native provided us, we decided to get started on what the project would look like and how it would operate. Using a [free online flowchart maker](#), we created a project hierarchy for the application.

This gave us a very rough idea on how to set up the project and what we would need to achieve our vision. Eventually more intricacies and details would be added to this model, but never strayed too far from the core structure. With this model in mind we finally began to code our project.

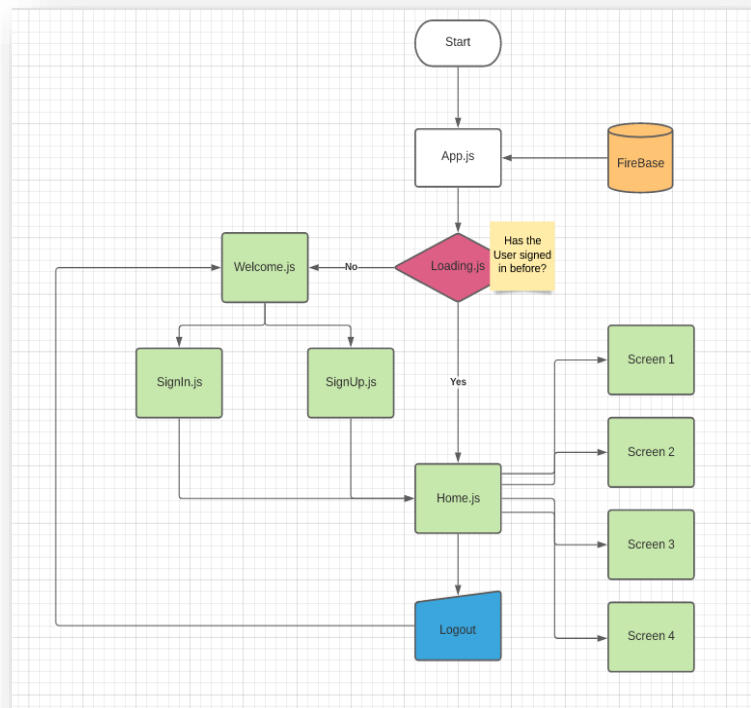


Figure 9; Project Hierarchy

Design (Owen)

Before any features were implemented into the application, we discussed about a general design and theme of the overall project. We wanted the design to be simple and clean, having only enough on screen to fulfill its intended purpose. A white and orange theme was chosen to keep with the simple design methodology. Having a simple color scheme would accompany the design well as we didn't want to overwhelm the user with multiple loud colors.

We also had a strong focus on reducing the amount of text on screen, opting to use icons and simple buttons to show functionality.

Firebase Implementation (Enda)

Once we had a basic display for the user to enter details for logging in/registering, the main focus was then on linking up Firebase authorization so that all of the user data and account handling would be done on the server side. Expo had great [documentation](#) for use with Firebase so this is where we started. The initialization of Firebase is done on launch of the application and this establishes a connection with the database that was created on the Firebase console.

For the user login and register, we used the `firebase.auth` method from the previously found documentation and once the user details were sending and receiving from the server we were content to move on to the next step, which was using Firebase for storage. This would need to be able to push data to storage and then pull it back to have the required functionality that we needed for the sound clips

Users accounts stored on the
Firebase console

Clip storage (Enda)

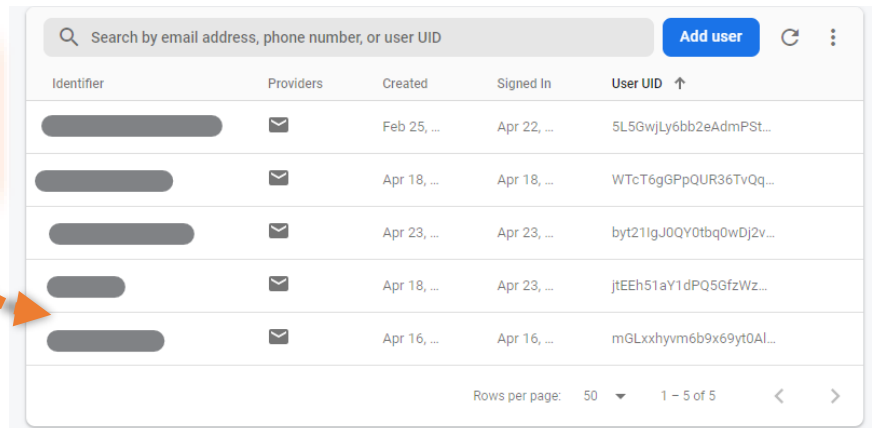
For the clips storage, a method of pushing to the database was needed.

The Firebase [documentation](#) had a section on uploading a file. After

adapting the code on the documentation, the clips were successfully uploaded. This was done by turning the temporarily stored recording URL into a blob and using the firebase put() method, and after seeing that the same method could take metadata, we added the name of the user and description for the clip into the upload as well. This meant that we could display specific clips for different users and have a more detailed display of the clips.

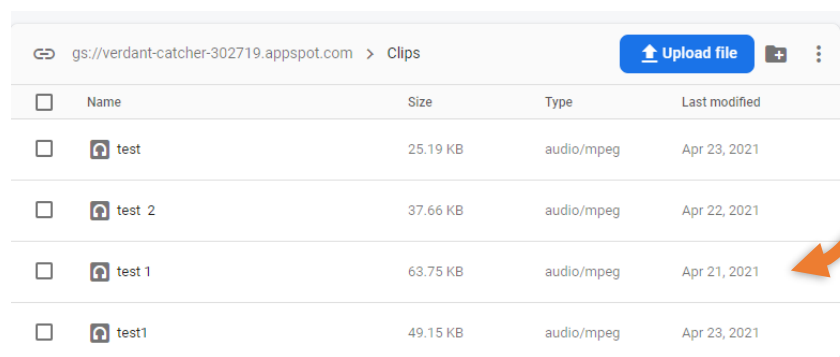
Fetching the data back down was a bit more complicated than uploading. Using the getDownloadUrl method we got the link to the clip with the desired name, but the URL of all clips was needed. After searching for people with similar problems we found an [answer](#) on stack overflow that solved the problem. This was used to loop through each item and grab the download URL and metadata. These clips could then be displayed with all the data on the server and could be manipulated to only add the clip to the display if it had the same name associated to it as the current user.

The tuner notes are only stored on the server in a separate folder, but these were added manually on the firebase console. They are pulled from the server the same way as the clips and are ready to use as soon as the main screen is entered



Identifier	Providers	Created	Signed In	User UID
[redacted]	[email icon]	Feb 25, ...	Apr 22, ...	5L5GwjLy6bb2eAdmPSt...
[redacted]	[email icon]	Apr 18, ...	Apr 18, ...	WTcT6gGPpQUR36TvQq...
[redacted]	[email icon]	Apr 23, ...	Apr 23, ...	byt21lgJ0QY0tbq0wDj2v...
[redacted]	[email icon]	Apr 18, ...	Apr 23, ...	jTEeh51aY1dPQ5GfzWz...
[redacted]	[email icon]	Apr 16, ...	Apr 16, ...	mGLxxhyvm6b9x69yt0Al...

Figure 10; Firebase User accounts



Name	Size	Type	Last modified
test	25.19 KB	audio/mpeg	Apr 23, 2021
test 2	37.66 KB	audio/mpeg	Apr 22, 2021
test 1	63.75 KB	audio/mpeg	Apr 21, 2021
test1	49.15 KB	audio/mpeg	Apr 23, 2021

The Firebase storage clips
folder with testing clips

Figure 11; Firebase storage

Audio Implementation (Enda)

The recording and playback of the audio was done using Expos Audio package. The [documentation](#) for this has functions for starting and stopping recording and playing sounds. These functions were implemented and after linking them to the front end, the recording was working as intended. After calling the stop recording method it returned a URL to the temporarily stored file. This was perfect because the URL could be used in the setup for firebase that was already in place.

Playing the audio using the playSound function on the same documentation worked at first but it would only play if the sound file was in the local directory on startup of the app. This was used for sounds that were stored locally but for playing the clips, we knew this wouldn't work with the functionality of our Firebase methods. So after reading through the [documentation](#) again we found a more basic version of the play sound function that didn't require a clip to be loaded on startup. The clips could then be played using the link pulled straight from firebase

The startup sound/load clips sound that plays was recorded by one of the developers on a guitar using software called [Reaper](#), that lets you record and mix audio. A VST plugin called [Boogex](#) was then added to Reaper to add simulation of an amplifier to the guitar. This also allowed editing of the guitar sound like modulation effects.

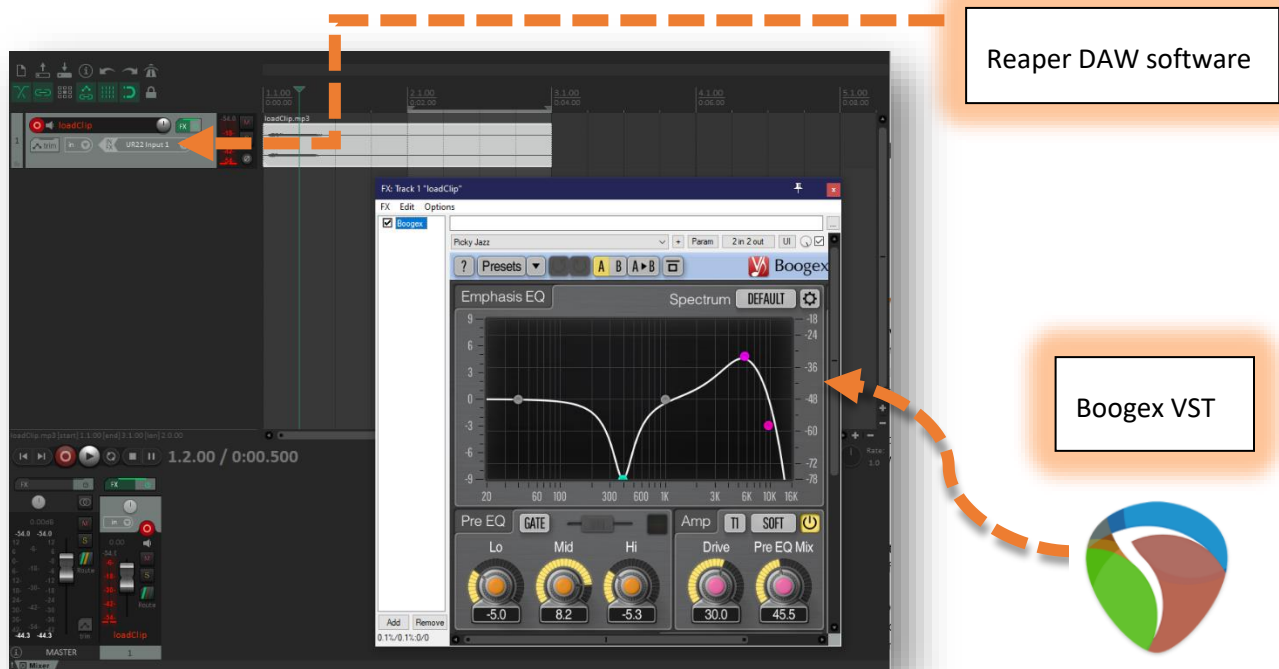


Figure 12; Reaper Workstation with Boogex

Limitations and Known Bugs (Enda)

One of the main limitations that was encountered on this project was the use of default functions instead of classes, this led to a lot of problems when looking for solutions during development. Since Expo forced us to structure the app this way most of the documentation online used some form of classes so we had to adapt them to use in functions.

The current Implementation of updating the screen isn't very efficient and led to a lot of unexpected outcomes in the output on the screen vs. what was intended. The use of `setState` allowed updates to the screen but combined with code written before this was implemented, made it difficult to refresh the screen to show new clips.

The Clip Data

The clip data is acquired by calling the async function `loadClips()`. This looks through all of the files in the clips folder on the firebase and if the username of the current user matches the name in the clip metadata then it adds it to the clip array. Since this takes a few seconds, a loading screen had to be added when you log into the app. The clips must be refreshed manually using a button that changes the state, currently there is no way to update the clips automatically without changing the implementation of the clips array. This is a bit clunky, as automatic refresh was the intended way of using the clips data.

Media Player

The clips on the main screen were intended to be loaded into a media player and let the user pause and skip to different times in the clip. Due to the limitations with Expo this turned out to be harder than expected and didn't end up being added. With the way the clip data was stored it is difficult to use the Expo media player libraries and having different clips be loaded into the player required the page to be refreshed every time to make the player appear and disappear.

Recorder

There is a bug on the submit screen for the recorder when you want to name your clip and upload it. When you go to press confirm or cancel the button takes two presses to do the intended action. This could be due of the way `setState` is used on the page, as this is how swapping from the recording button, stop button and back to the submit screen is done. Sending the user to a separate stack.

Sign-in bug

On starting the app for the first time the user is prompted to register or sign in, the register works as intended but when you sign in, it brings you back to the register or sign in page again. It checks if the user is authorized before the async function finishes. This can be bypassed by restarting the app after signing in. When it launches up again it will authorize the user since they had already sent the request to authorize. Logging out of the app will cause this bug to happen again when you try to sign in after.

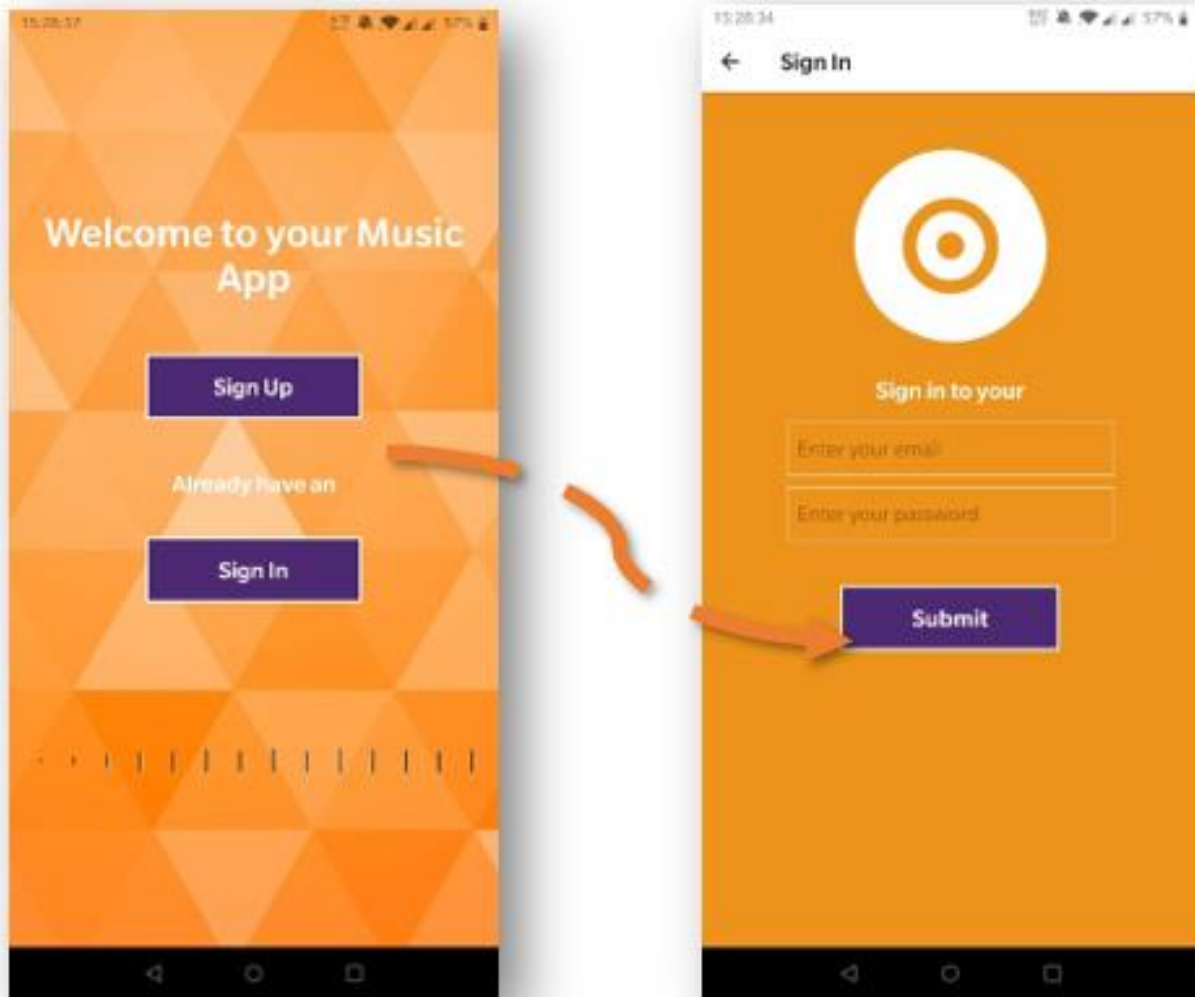


Figure 13; The user gets sent from the login page (left) to the sign in page (right) and back again

Performance

The apps performance gets worse when the user signs out then back in again. This is a big problem and can stop the user from enjoying the app. The connections to firebase could be the cause of the problem if there are calls that aren't reset when the user logs out. Calling the clips too many times could also be the cause.

Keyboard

When the keyboard pops up for the user to type in the details for their recording, the navigation bar at the bottom pops up with the keyboard. While this doesn't break the app, it can reduce the feeling of quality for the user and cause accidental navigation to other screens.

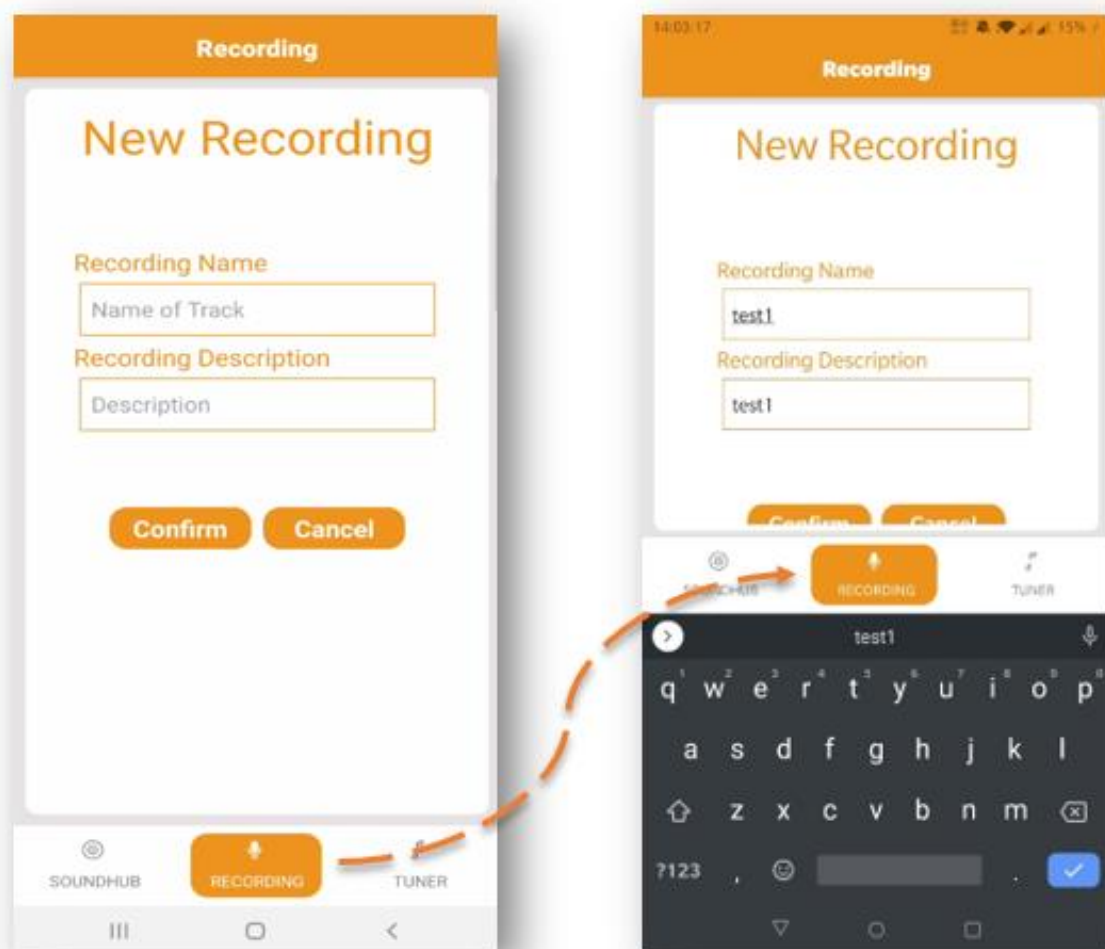


Figure 14; Keyboard bug

Testing Plans (Enda)

The app that we have built would greatly benefit from having tests written for each feature so that we can assure that each feature will operate without fault for the user. Currently there hasn't been any tests run for the application, but these are a few that we could run for the current build and a few that we could use for future features.

Tests for Current Build

Sign in/Register

- Input validation for both register and sign in submissions
- Navigate over and back from sign in to register
- Press accept buttons with no input and cancel buttons
- Try register already registered email
- Try to sign in again when you reach the verify user screen
- Logout works on every screen

Soundhub

- Press each clip to make sure they play
- Delete a clip to make sure it is removed
- Compare firebase data to displayed clips so every clip that should be displayed is on screen
- Press the floating action button to pop it open and close
- Press record in the floating action button
- Swipe on the screen to change page

Recording

- Press recording button
- Press stop button
- Cancel the recording and check if it was saved
- Swipe out on every recording screen/ Mid recording or before save
- Input validation
- Check for new clip in SoundHub

Tuner

- Press each note to check sound is playing
- Correct note is playing on press
- Stop button works as intended

Future Feature Test Plans

Music Recognition

- Song that is being listened to is the correct one displayed
- Press any button on the page to make sure they work
- Swipe during listen to make sure it cancels

Better Tuner

- Make sure note displayed is the correct note
- Check if frequency is correct

Ear Trainer

- Validation on correct note pressed
- Validation on incorrect note pressed
- Press each answer button
- Correct note/chord is being played
- Correct note/chord is displayed

Recommendations for Future Development

(Enda)

Possible solutions for bugs

Clip Data

The clips could be stored into the array using the `setState` function so that every time it is updated the page will refresh. This would make it so the user wouldn't have to manually press the reload button to view new clips that they have recorded.

Sign In

Add a delay to the authorization so that it has a chance to pull the user data before it checks what the users name is.

Performance

Make it so that getting the clips is only called when needed and all connections are reset after you sign out. This will stop unnecessary functions running in the background so the app will run a lot smoother.

Future Development

For future development the first focus would be on fixing the above bugs and making sure our current app works as it was intended to. After these problems are ironed out then adding more tools and features will be the focus. The ideas that were dropped early in development will slowly be implemented in.

Other Users Clips

Possible feature for seeing clips of other people that have uploaded and selected an option to share with everyone when they save the clip. This would give the app a link between users and allow new ideas and influence from hearing other users clips.

Music Recognition

This will be another page that can listen to a song and retrieve the song data using music fingerprinting. This can be done using things like the Shazam API. This feature isn't high on the priority list, but it would add to the user experience and the idea of the app to be an all in one hub for music.

Better Tuner

The tuner was originally supposed to record the users instrument and display the frequency and note on the screen, but this wasn't possible to include with all the other features in the given timeframe. However, in future development this would be a high priority so that you can tune to any note and not just the preset ones.

Ear Trainer

A section where a note is played and you must select the pitch of the note being played, this is to improve the users pitch recognition which is very useful to a musician. This could also play different chord variations to test the users pitch on these as well.

Conclusion (Owen)

In conclusion, the purpose of this module was to test the individual in creating, managing, and refining a piece of software in a professional and timely manner. The developers were tasked with creating a piece of software that must be designed, developed, and deployed to demonstrate the best practices that the student has learned in the previous semesters of the course.

The developers feel that they have tackled this project in a professional and timely manner by attending weekly meetings with their supervisor, attending the multiple web seminars throughout the duration of the course, and keeping their commit history on GitHub consistent.

The developers feel that this document aims were sufficiently met by doing the following:

- Providing a [functional breakdown of the system](#) that was developed and showcasing the architectural overview of the system.
- Identifying and comparing [the possible technologies](#), justifying the choice of technology and sufficiently explaining how it was used in the implementation.
- Giving a detailed description of [any database that was used with the application](#).
- Clearly showing the [screen layout](#) providing a blueprint for how the application looks to the end user.

Many skills were learned during the development of this project and the developers feel that they were tested sufficiently in the context of software development.

Due to the COVID-19 Coronavirus disease, communication was vital for the developers to progress through the project. Frequent planning in remote calls to discuss code, design and the project scope was imperative to the success of this project. Learning how to use video calling services to a high standard was a priority.

[Stated earlier in this document](#), the developers planned on challenging themselves in using React Native and learning how to utilize the tools the framework provided. Learning a brand-new framework for a project was in fact challenging but the developers feel they've learned a lot about the fundamentals of JavaScript as well as the trials and tribulations of mobile application development.

References

[1] En.wikipedia.org. 2021. *Pitch pipe - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Pitch_pipe> [Accessed 20 April 2021].

[2] Wijdan, S., 2018. *React Native - Expo vs React Native CLI?*. [online] Xencov. Available at: <<https://xencov.com/blog/react-native-expo-vs-react-native-cli#:~:text=Expo%20is%20a%20free%20and,using%20just%20JavaScript%20and%20React%20.&text=Thi s%20is%20because%20Expo%20projects,that%20react%2Dnative%20cli%20does.>>> [Accessed 21 April 2021].

[3] Stevenson, D., 2018. *What is Firebase? The complete story, abridged.*. [online] Medium. Available at: <<https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>> [Accessed 21 April 2021].

[4] OS-System. 2020. *Best Databases to Use for React Native Mobile App Development*. [online] Available at: <<https://os-system.com/blog/best-databases-to-use-for-react-native-mobile-app-development/>> [Accessed 21 April 2021].

Resources

Aggie.io

<https://aggie.io/>

This was used to create a collaborative whiteboard where we came up with our ideas.

Discord

<https://discord.com/>

Using this application was imperative in the completion of this application. This was our main source of communication

LucidChart

<https://www.lucidchart.com/pages>

Used to create a flowchart that showcased the hierarchy of our application

Expo Documentation

<https://docs.expo.io/guides/using-firebase/>

<https://docs.expo.io/versions/latest/sdk/audio/>

<https://docs.expo.io/versions/latest/sdk/audio/#playing-sounds-1>

Utilizing this documentation helped us understand firebase in conjunction with Expo.

Firebase Documentation

<https://firebase.google.com/docs/storage/web/upload-files>

Utilizing this documentation helped us understand firebase.

Reaper

<https://www.reaper.fm/>

Used for making the sounds for the application.

Boogex

<https://www.voxengo.com/product/boogex/>

Used in conjunction with Reaper in making the application sounds.

Shazam

<https://www.shazam.com/gb>

Application similar to what we wanted to develop into our app