Logo, company name

Description automatically generated

BSc in Software Development – Year 3  
Professional Practice in IT

Enda Goggin – G00369903

Owen Kelly – G00366614

# 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 the 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](#_Toc69985845)

[Table of Figures iv](#_Toc69985846)

[Introduction 1](#_Toc69985847)

[Documentation 1](#_Toc69985848)

[Specification 2](#_Toc69985849)

[Recording 2](#_Toc69985850)

[Saving Clips 2](#_Toc69985851)

[Tuner 2](#_Toc69985852)

[Quick Navigation 3](#_Toc69985853)

[User Specific Data 3](#_Toc69985854)

[Technology Implemented 4](#_Toc69985855)

[React Native 4](#_Toc69985856)

[Expo CLI 4](#_Toc69985857)

[Recording 5](#_Toc69985858)

[Firebase 5](#_Toc69985859)

[Authentication 6](#_Toc69985860)

[Storage 6](#_Toc69985861)

[Architecture of the Solution 7](#_Toc69985862)

[Sound Hub Home Page 7](#_Toc69985863)

[Tab Bar 7](#_Toc69985864)

[Recording Pages 7](#_Toc69985865)

[Tuner 7](#_Toc69985866)

[Welcome 7](#_Toc69985867)

[Sign In 7](#_Toc69985868)

[Sign Up 7](#_Toc69985869)

[Loading Screen 7](#_Toc69985870)

[Software Development Life Cycle 8](#_Toc69985871)

[Limitations and Known Bugs 9](#_Toc69985872)

[The Clip Data 9](#_Toc69985873)

[Sign-in bug 9](#_Toc69985874)

[Media Player 9](#_Toc69985875)

[Recorder 10](#_Toc69985876)

[Performance 10](#_Toc69985877)

[Keyboard 10](#_Toc69985878)

[Testing Plans 11](#_Toc69985879)

[Recommendations for Future Development 12](#_Toc69985880)

[Possible solutions for bugs 12](#_Toc69985881)

[Clip Data 12](#_Toc69985882)

[Sign In 12](#_Toc69985883)

[Performance 12](#_Toc69985884)

[Future Development 12](#_Toc69985885)

[Music Recognition 12](#_Toc69985886)

[Better Tuner 13](#_Toc69985887)

[Ear Trainer 13](#_Toc69985888)

[References 14](#_Toc69985889)

# Table of Figures

[Figure 1; The Firebase Model 5](https://galwaymayoinstitute-my.sharepoint.com/personal/g00366614_gmit_ie/Documents/College/Third%20Year/Documentation.docx#_Toc69896413)

# Introduction

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 standards 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 made on the merits of each as to whether it is appropriate to use in the development of the new system.

## Documentation

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.

# Specification

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 the user 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]](#_References) 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

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

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 due to one of our prior experiences with React Native, 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

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]](#_References)  
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

The recording and playback is handles 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 straight away. The recorder also returned a URL to the temporarily stored file that could be used to send straight to firebase.

## Firebase

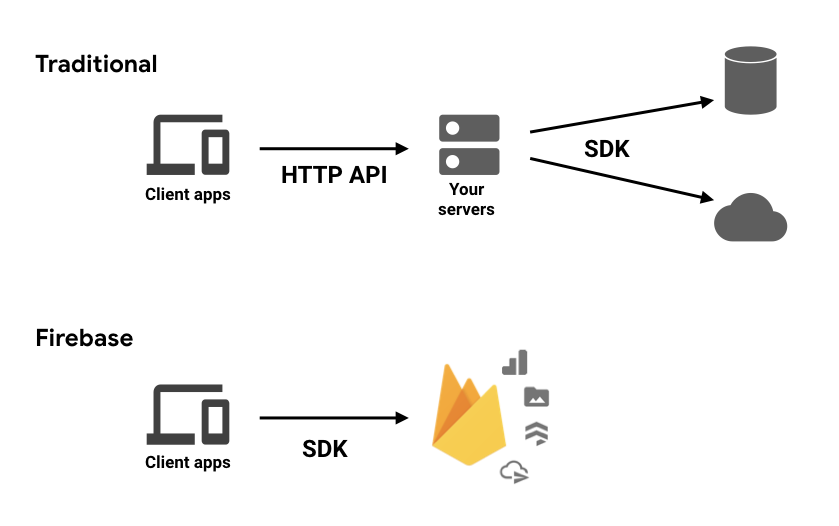
Firebase is Google’s mobile application development platform that helps you build, improve, and grow your app.[[3]](#_References)

Figure 1; The Firebase Model

Originally for this project, we had decided to us MongoDB as our backend as we thought the implementation for recording storage and sign in authentication could be done easily. However, our lecture 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]](#_References) 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

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

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

# Architecture of the Solution

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

### Tab Bar

## Recording Pages

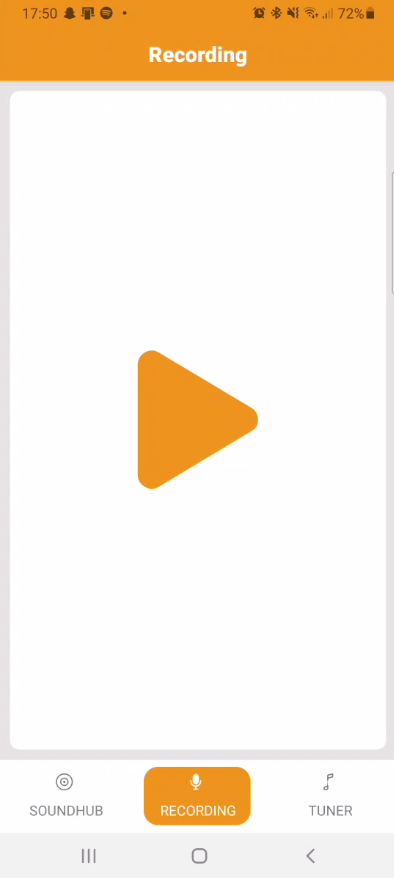
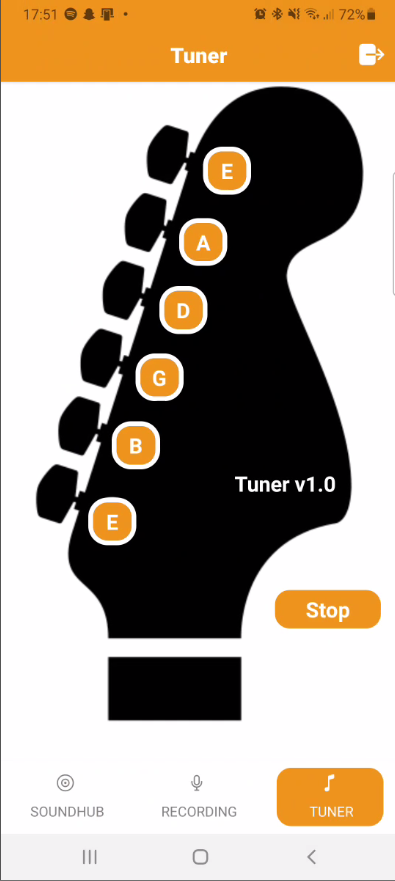
## Tuner

## Welcome

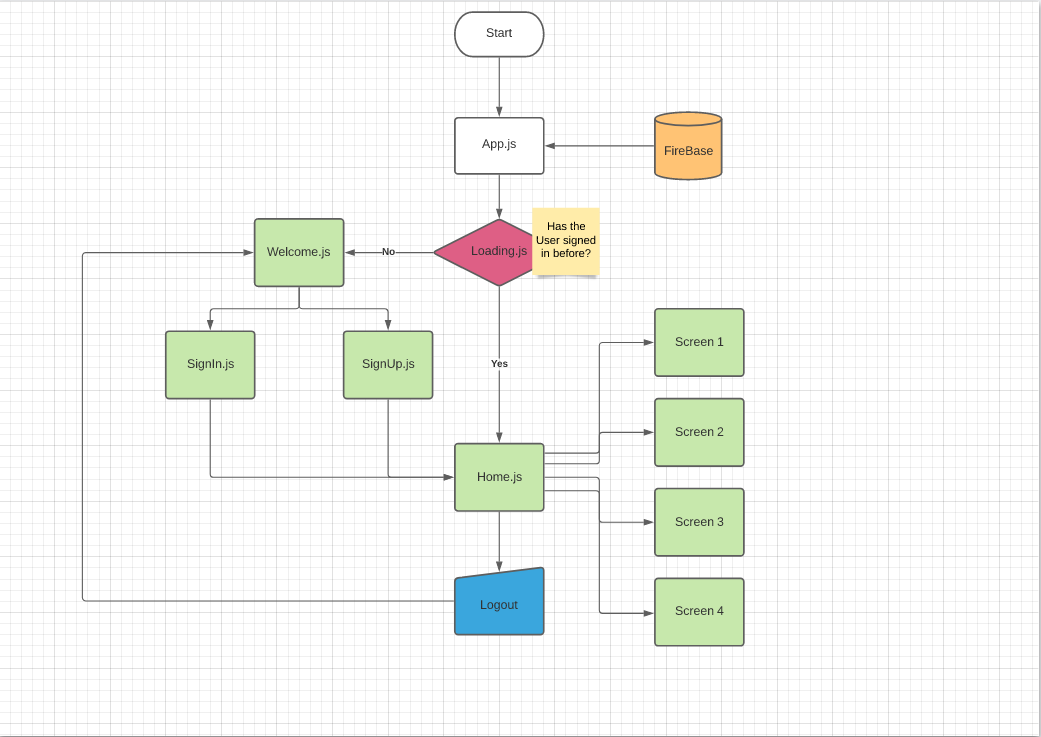
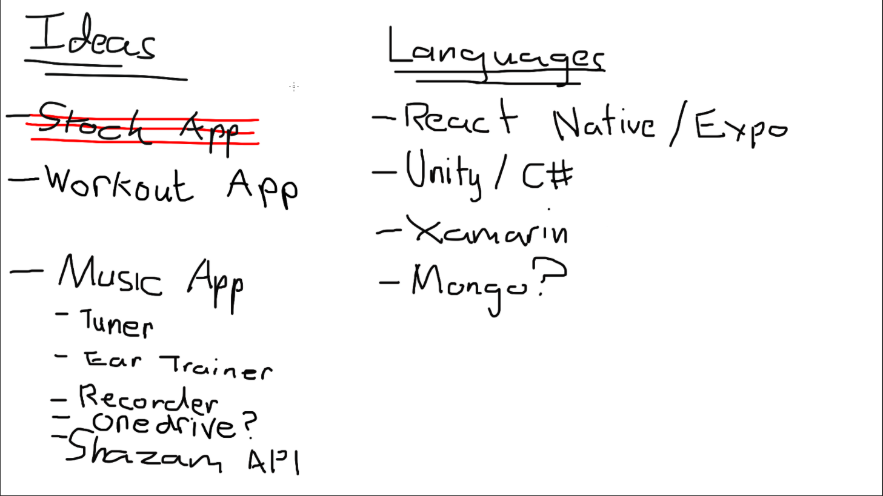
## Sign In

## Sign Up

## Loading Screen



# Software Development Life Cycle



# Limitations and Known Bugs

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 gotten 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 and automatic refresh was the intended way of using the clips data.

## Sign-in bug

On starting the app for the first time the user is prompted to register or sign in, the register works fine but when you sign in it brings you back to the register or sign in page again. This is because the it checks if the user is authorized before the async function for authorizing finishes what its doing. 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.

## Media Player

The clips on the main screen were intended to be loading 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 because of the way setState is used on the page because that is how swapping from the recording button to the stop button to the submit screen is done. Sending the user to a separate stack

## 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 and using 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 of this also.

## 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.

# Testing Plans

The app that we have built would greatly benefit from having tests written for each feature so that we can assure

# Recommendations for Future Development

## 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

## 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.

### 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 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 but 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.

# 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=This%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].