School of Electronic Engineering and Computer Science

Final Report

Programme of study:
BSc Computer Science

<u>Project Title:</u>

ChordTrace: a music browsing application that allows users to find music by chord.

Supervisor:
Johan Pauwels

Student Name:
Jonita Jasici

Final Year
Undergraduate Project 2023/24

Date: 29/04/2024

Queen Mary
University of London

Abstract

In current music applications, users are recommended music based on preferences that they have shown beforehand, which may lead to users listening to the same genres of music with no variation. My research aims to allow casual listeners to learn the basics of music theory whilst searching for new music, and reinforcing music theory for users who have previous knowledge. In this report, I detail the research and planning that has taken place to create the prototype of a music mobile application that will allow users to search for music by chord while explaining the music terminology for their choices. Requirements and design choices will be discussed in correlation to the aims of the project. Two forms of testing are conducted to improve the application: usability testing and unit testing. Usability testing is conducted to find out whether the prototype of the application has achieved its aims while providing high quality user experience, whilst unit testing is conducted to find errors within the code. Finally, the conclusion details what has been achieved, the challenges and limitations faced, and possible future improvements.

Contents

Chapter 1: **Introduction**

1.1 Problem statement

Modern day music streaming applications use recommendation systems to recommend music, in which recommendations are made using similarity measures, either on item data or user patterns (Van den Oord *et al.*, 2013). This can be problematic as it can cause users to listen to music in similar styles or genres, resulting in a generation of music listeners that lack variation in consumption.

Another topic addressed within this report is music theory learning, as a common difficulty found in teaching music theory is the lack of student engagement (Wang, 2021). Music theory can be complicated and is often taught through textbooks, which may make engaging with music theory difficult for people trying to learn. This is an issue as it can discourage learners from pursuing music more in the future.

This paper aims to solve these problems by combining them into one question: how do we create a new way to discover music and engage with learning music theory? The proposed solution aims to solve these problems by creating a music application that encourages users to learn music theory by searching for music by chord rather than by name or genre.

1.2 Aims

My main aim for this project is to create a music application that allows users to discover music in a new and innovative way via a chord search while encouraging them to learn the basics of music theory. For users that do not have any knowledge on music theory, this application will give information on core music terminology and allow the user to make decisions that will actively contribute to increasing their music knowledge. For the experienced musician, this application will allow them to solidify their knowledge on music theory and discover music with specific chords, either to learn how to play songs with chords from the search or just to get inspiration on how chords may be implemented when writing songs.

Upon opening the application, users can pick a set of chords, with the option to pick from a particular preset scale. This is where the educational aspect of this application is implemented, as they will be able to learn about scales and different types of chords in a scale, as well as the relationships between them. The user will then get a list of the songs including these chords. From there, users will have the opportunity to listen to the songs from their search, similar to other mundane streaming platforms.

Unlike my application, existing music listening platforms recommend music based on what a user has listened to before-hand, which may mean that the user will end up only listening to the same genres of music. My application will create an interesting and engaging way of listening to music that has not yet been seen, preventing users from being uninterested in music from the lack of variety they get from existing software. Most importantly, my application will teach the basics of music theory in a way that is proactive and not through a textbook.

1.3 Objectives

To fully complete the aims of my project, I have created a basic list of objectives as follows:

- Create a fully functional music browsing application based on a chord search.
- Create an optimized UI for iPhone with the possibility of expanding into Android.
- The application must be able to show a list of songs based on the chords from the chord search with the artist's name and album cover included.
- The application must allow the user to click on and listen to the songs from the chord search, as well as view the song's chords and other information.
- There must be a waveform that shows the progression of the song while it is playing.
- The application must have educational information for people who do not have music theory knowledge.

Chapter 2: **Literature Review**

2.1 Music recommendation

2.1.1 What are recommendation systems?

Jain *et al.* (2015) stated that, “recommendation systems are software tools and techniques whose goal is to make useful and sensible recommendations to a collection of users for items or products that might interest them”.

In the context of music recommendation, these systems will recommend new music to listeners that will match their preferences (Van den Oord *et al.*, 2013). Most music applications make use of these systems and in the following sections I will discuss the types of recommendation systems used and how they have been implemented.

2.1.2 Content-based filtering

Content-based filtering is a filtering technique that uses the data associated with a user and data associated with the items within the system (Jain *et al.*, 2015) .In the context of music recommendation, this may mean that it will recommend music based on the meta-data of music the user has already listened to, for example the artist of the song (Van den Oord *et al.*, 2013).

While content-based recommendation may recommend music based entirely on what the user has interacted with beforehand, this method has its limitations. Van den Oord, *et al.* (2013) stated that this method of recommendation leads to “predictable recommendations”, for example “recommending songs by artists the user is known to enjoy”. Given this information, we can infer that while content-based filtering is able to recommend music the user will enjoy, the recommendations may not lead to music that is new to the user.

2.1.3 Collaborative filtering

Collaborative filtering is a filtering technique that uses existing information on multiple users and their preferences and makes comparisons to generate recommendations (Jain *et al*, 2015).

To recommend similar music, this recommendation system technique is highly effective. In a study conducted by Slaney (2011), it was found that when users were given two playlists, one created by using user similarity ratings, and one created by filtering songs with the same genre, users were inclined to say that the playlist created using similarity ratings was the most similar. This study shows that when content is recommended through user ratings, the content is more likely to be similar than if the recommendation is based on metadata.

However, this recommendation system can suffer from the cold start problem, i.e., this recommendation system only works if there is available, pre-existing data in place (Van den Oord *et al.*, 2013). Therefore, this system would not be advisable for my application as there would be no user data available to recommend music.

Another issue with this recommendation system is that it can be constrictive. While the user may reliably be able to find music within the same genres that they are known to enjoy, they may find it difficult to expand their music tastes in genres the user does not typically listen to. This type of music recommendation system may be efficient enough for a casual listener but may be dull for people who have a passion for music, or aspiring artists.

2.1.4 My approach to music recommendation

Whilst both recommendation techniques listed above are good for similarity-based recommendation, my application is not concerned with what the user enjoys. Instead, my application is more concerned with encouraging casual listeners to learn the basics of music theory, helping people learn how to play new songs on their instruments with chords they already know, and recommending entirely new music then what the user has listened to beforehand.

Therefore, instead of using a traditional recommendation system, I have decided to allow for searches based only on chords, without preference-based recommendation of music.

2.2 Evaluation of existing applications and similar research

2.2.1 Comparison of generic music streaming applications to my project

To determine whether my application was unique to existing music applications, I created a list of functionalities that my application should have and then compared it to the standard music applications that are out there.

Application function	My app	Spotify (Spotify, 2023)	Apple Music (Apple, 2023)	Deezer (Deezer, 2023)	SoundCloud (SoundCloud, 2023)
Searching songs based on chords.	Yes	No	No	No	No
Searching for songs based on a chord preset.	Yes	No	No	No	No
Displaying a list of songs based on a search.	Yes	Yes	Yes	Yes	Yes
Allows users to listen to songs.	Yes	Yes	Yes	Yes	Yes

Allows users to view a songs waveform.	Yes	Partially (miniaturized waveform)	Partially (miniaturized waveform)	Partially (miniaturized waveform)	Yes
Displays a song's chords.	Yes	No	No	No	No
Gives information on music theory or terminology.	Yes	No	No	No	No

Based on the above observations, none of the standard music streaming applications allow the user to search by chord or gives information on music theory. Therefore, this shows that my application will be unique in comparison to the standard music streaming application.

2.2.2 Comparison of similar music applications to my project

To further analyse whether my application is unique, I used the same list of functionalities from the previous section, and looked at music applications that had similar functionalities, regardless of whether it was a streaming platform or not.

Application function	My app	ChordGenome (*Search songs by chord, Progression, and decade*, 2021)	Hooktheory (*Chord progression trends*, 2024)	Guitar Player Box (*Choose songs by selecting chords*, 2024)	Chordify (*Instant chords for any song*, 2024)
Searching songs based on chords.	Yes	Yes	Partially (one chord from a scale)	Yes	Yes
Searching for songs based on a chord preset.	Yes	No	Yes	No	No
Displaying a list of songs based on a search.	Yes	Yes	Yes	Yes	Yes

Allows users to listen to songs.	Yes	Partially (redirects to YouTube)	Yes	No	Yes
Allows users to view a songs waveform.	Yes	No	No	No	No
Displays a song's chords.	Yes	Yes	Yes	No	Yes
Gives information on music theory or terminology.	Yes	No	No	Partially (Shows chord finger placement)	Partially (Shows chord finger placement)

Based on the above observations, the applications that allow for chord searches are typically lacking in the ability to search based on a preset, and some cannot be used as a streaming platform. Most importantly, none of these websites can be used by a normal music listener as there is no definitions for music theory. These websites mainly seem to be used by instrument players to learn new songs with the chords they know, rather than as a music streaming application. The main purpose of my application is to be used to listen to new music while also providing knowledge on music theory. This makes my application unique to other chord search applications, providing a wider variety of uses, and catering to a larger audience through the music terminology information.

2.2.3 Similar research

The most similar study to mine would be considered Pauwels and Sandler's (2019) web application. To summarise their work, they created a web application that suggests music to learning musicians based on a chord search, for the purpose of practicing songs with chords that they are familiar with to improve the efficiency of learning an instrument (Pauwels and Sandler, 2019).

Though their motivation may be different, the overall functionality compared to my project can be considered similar, with the main function being to search music based on a chord search. While their method of creating this application works for their target audience of aspiring instrument players, it otherwise cannot be broadened to a larger audience. Only people with knowledge of music theory can understand and use this web application and it cannot be used by a normal music listener for music recommendation.

While my application can also be used by instrumentalists to learn songs, there are many key differences between my application and this research. The main aim of my application is to encourage users to learn the basics of music theory whilst searching for new music, with or without having any prior music knowledge, which is unlike the web application and research conducted by Pauwels and Sandler (2019).

2.3 Tech-stack and justification

2.3.1 React Native

React Native is a JavaScript framework used to create mobile applications on a native level (Dabit, 2019). Initially created by Facebook in 2013 to create their mobile application, this framework allows coders to create multi-platform applications (Boduch *et al.*, 2022).

The main reason that I chose to use React Native for the development of my application is that it is a JavaScript React library. Dabit (2019) stated, “many web developers have JavaScript experience, which helps ease the transition from web to mobile app development.” As someone who has experience using both React and JavaScript, I decided that developing my application with React Native was the best way to ensure that I would not be spending a copious amount of time trying to understand the language instead of working on my application.

Another important reason to take into consideration is that React Native is multi-platform. Therefore, if I decide to develop this application further in the future and make it applicable to both iOS and Android, I will be able to use the same source code with minor alterations.

2.3.2 TypeScript

TypeScript is a JavaScript extension that allows developers to catch type errors within code (Rippon, 2023). One of its main benefits is that it takes in the code and looks for type errors, making code easier to debug (Goldberg, 2022).

When I started doing my project, I realised that React Native code was difficult to debug. The errors I was given were unspecific and hard to locate within the code. TypeScript allowed me to view errors in the code before compiling, so I did not have to deal with confusing errors.

2.3.3 Expo CLI

Expo CLI is a React Native framework that not only allows developers to create mobile and web applications from a singular source code, but also allows developers to quickly build and test their applications (Boduch *et al.*, 2022). There is also React Native CLI, a similar tool that is used to build applications, however applications built using this tool would need to be built separately for Android and iPhone due to the differences in underlying native components (Boduch *et al.*, 2022). An advantage of using Expo CLI rather than React Native CLI is that developers do not need to have an understanding of native components (Zammetti, 2018).

Another advantage of using Expo CLI is Expo Go. Expo Go is a mobile application that can be used to run React Native projects without having to compile and build a native application (Boduch *et al.*, 2022). This will allow me to be able to easily run and test my application without having to go through the process of building, which will allow to complete my application faster.

2.3.4 APIs for music applications

To best determine what APIs I should use for my application, I investigated the API's available for free to use music APIs and compared them in the table below.

API	Available information (excluding user information and information unrelated to music)	Disadvantages
Soundcloud (SoundCloud, no date)	• Can lists songs in a playlist. • Can create, update, and delete playlists. • Allows streaming tracks (that are not restricted). • Allows searching tracks and playlists. • Can create, update, and delete tracks. • Returns extra information on a track or artist.	• When searching for a song, you can only search by title, id, genre, tag, bpm, duration, or time created. Cannot search with chords. • Not all songs can be played within the API call.
Deezer (Deezer, no date)	• Returns information relating to albums, tracks, artists, playlists, and radios. • Returns data of a chart of a specific genre. • Data on genres. • Allows search of tracks and artists, which can return a preview of track.	• When searching for a song, you can only search by artist, album, title, label name, duration, and bpm. Cannot search by chords. • Can only return a preview of a track, unless authenticated.
Jamendo (Jamendo, no date)	• Returns information relating to albums, artists, tracks, playlists, and radios. • Can return the entire mp3 of a track if audio download is allowed for that song. • Includes waveform data for their tracks.	• When searching for a song, you can only search using specific data such as artist, track ID, track name, type, tags, album, etc. Cannot search by chord. • Not all songs can be played within the API call.
Audio-analysis API (This is the same API used in Pauwels and Sandler's (2019) research).	• Specializes in track searches. Can search for tracks by tempo, tuning, key, or chords. Returns an ID and information related to search (for example if it was search by chord, the ID and the chords used would be returned).	• Information such as song title, artist, album etc. are not given within this API. However, this API can return the ID of a song in correlation to other API's, including Deezer and Jamendo, which may then be used to get other information.

For my project, I have decided to use the Jamendo API and the Audio Analysis API in parallel to each other. It is crucial for me to use the Audio Analysis API to be able to search for songs based on chords as the other APIs I have researched do not have the same complexity in their searches.

However, as this API does not return information such as song title or artist, I will need to use another API to get this information. The Audio Analysis API can be given a parameter to only return the IDs of a song from a given provider. From the APIs listed above, the Audio Analysis API can be filtered to show songs from Jamendo and Deezer.

The main reason why I chose to use the Jamendo API over the Deezer API includes the fact that the Jamendo API can return the complete mp3 of most audio tracks, whereas in comparison, the Deezer API can only return a preview of a song unless the user is authenticated. While it is achievable to authenticate the user, there is no guarantee that the users of my application will have a Deezer account, and I do not want to revolve my application around Deezer.

Chapter 3: **Functional requirements**

The functional requirements of a system are the functions of a system that define what the system should do (AltexSoft, 2021). This chapter details these functional requirements for my application.

3.1 Main functional requirements

Below I have detailed the main functional requirements of what every screen in my application should achieve.

3.1.1 Search Screen

The search screen of the application should allow the user to search for songs by choosing one or more chords from a selection. There must be an option available on the search screen to pick from a set of preset chord scales if the user wishes. By not choosing from a preset scale, the user will have more control of the creative aspect of this application, which in turn could lead to them discovering unique songs. For users with no music theory knowledge, the preset scales help the user choose chords if they feel overwhelmed by number of choices. Finally, the search screen should have information on music terminology to help the user understand their choices, such as preset and chord information.

The functionalities of this page can be split into these basic requirements (RQ's):

- RQ1: The application should allow the user to choose a preset chord scale.
- RQ2: The application should allow the user to select a set of 1 or more chords.
- RQ3: The user should be able to view music terminology information.
- RQ4: The user should be able to make a search with these chords.

3.1.2 Results Screen

The results screen should list the songs that have the chords selected from the search screen. The results screen should show each song with the artist title and album cover, and then should allow the user to click on the song to see a song screen.

The functionalities of this page can be split into these basic requirements:

- RQ5: The user should be able to view a list of songs that have those chords with song title, artist name, and album cover included.
- RQ6: The application should allow the user to click on a song to be redirected to the song screen.

3.1.3 Song Screen

Each song should have a song screen. This screen will show the songs information, such as the chords used, the artist title and album. Furthermore, this screen should also allow the user to listen to the song. When the user listens to the song they should be able to view a waveform that shows the progression of the song while its playing.

The functionalities of this page can be split into these basic requirements:

- RQ7: The application should display information related to the song, such as artist, album, etc.
- RQ8: The application should display the chords used in the song.

- RQ9: The application should allow the user to listen to the song and view the song's waveform.

3.2 Optional functional requirements

These optional functional requirements list out some general ideas that can be implemented as part of my application once the main objectives are achieved. While these ideas would improve the overall functionality of my application, they are less important than the main functional requirements, and should only be attempted if all the main requirements are completed.

3.2.1 Search Screen

Once the core functionalities are set in place, some extra functionalities for those who have less theory knowledge can be put in place. For example, the presets can include common chord combinations to make choosing chords easier for the user. Potentially, the user should be able to hear each chord when clicked to show what the chord sound like, which gives the user a better understanding of what the chord is.

These extra functionalities can be listed as such:

- RQ10: Given that they have selected a preset, the user should choose if they want to have the most common chord combination from that preset or not.
- RQ11: The application should allow the user to listen to each selected chord when clicked on.

3.2.2 Song Screen

Once the core functionalities are set in place, some extra functionalities can be added to increase the quality of the user experience. This may include extra information such as links related to the song, or being able to see when each chord is used while the song is playing.

These extra functionalities can be listed as such:

- RQ12: The user should be able to follow links to other streaming platforms.
- RQ13: The user should be able to see when each chord is used in the song.

Chapter 4: **Design and implementation**

This chapter includes a UX flow design and a wireframe that is made in correlation to the functional requirements outlined in Chapter 3.1. These designs were made prior to developing the application as to speed up the process of development. I finally address how the design is implemented within the code.

4.1 UX flow diagram

User flow diagrams define how a user should be able to interact with a given system from start to finish (Browne, 2023). Designing how the user should interact with the system is beneficial in the process of development to avoid confusion, making sure that all the main functions of the system are implemented and flow well together.

Below is a UX flow diagram which details how the user should interact with each page of the application without including the extra requirements in the Chapter 3.2.

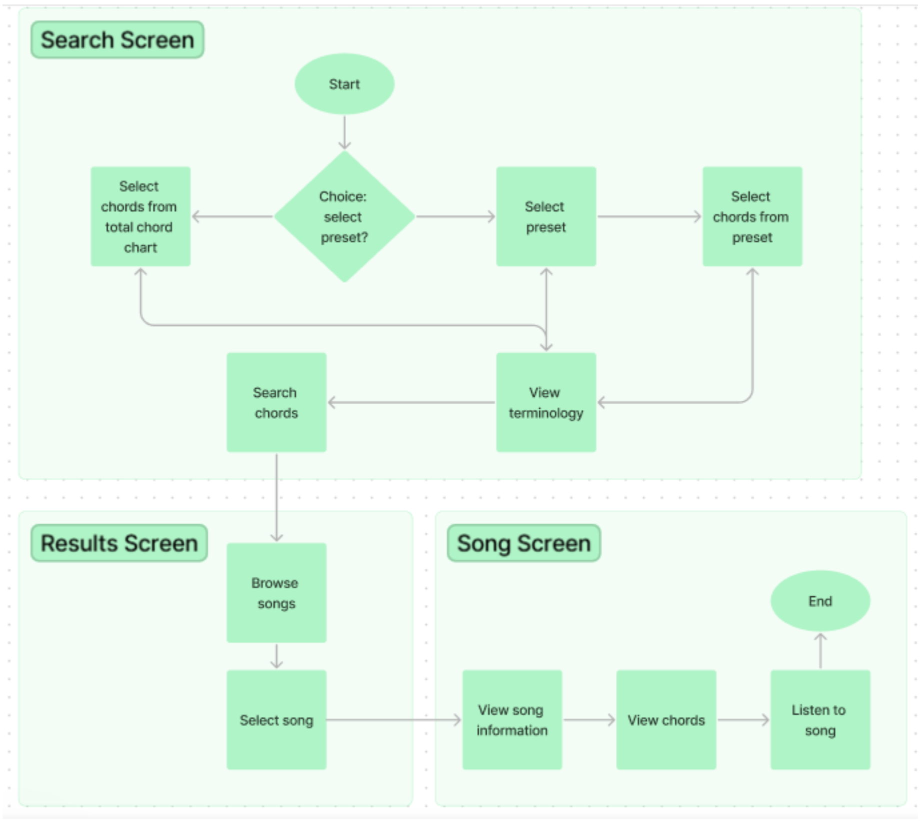


Figure 1. UX flow made using Figma.

In the Figure 1, there are two ovals that represent the start and end states of the diagram. The start state points to the first action that a potential user would have to complete within the application. The end state is positioned after the final task that the user would have to complete to have used all the functions of the application. In between these states are square states which detail an action, and diamond states which indicate a decision. These states are grouped by their respective screens, and placed in the order of how the user is expected to interact with the application.

In the home screen the user first has an option of whether they want to select a preset or not which correlates to RQ1. If the user selects a preset, the user will be able to view what the terminology of the preset means and continue to make selections, correlating to RQ3. The user should then be able to choose a set if chords, as well as view their corresponding terminology, which correlates to RQ2 and RQ3. After they have finished selecting their chords, the user should then be able to search for songs with these chords, complying with RQ4.

Following that, the user should be redirected to the results screen. Within the results screen the user should be able to view all songs with the chords searched and browse through them, corresponding to RQ5. After the user has finished browsing and decided on what song to view, the user should be able to select the song to view a song screen, which correlates to RQ6.

Once on the song screen, the user should be able to view the songs information as well the chords that are used, corresponding to RQ7 and RQ8. Then the user should be able to listen to the song and view its waveform, complying with RQ9. This completes all the main functionalities of the application from start to finish.

4.2 Wireframe

A wireframe details how an application should be structured in terms of content and functionality (Hannah, 2023). Wireframes are important to design as knowing the structure of the content within each screen speed up time when it comes to development, however it does not necessarily show the final structure of the complete outcome.

Figure 2 shows a wireframe that correlates to the main functional requirements, detailing each page of my application.

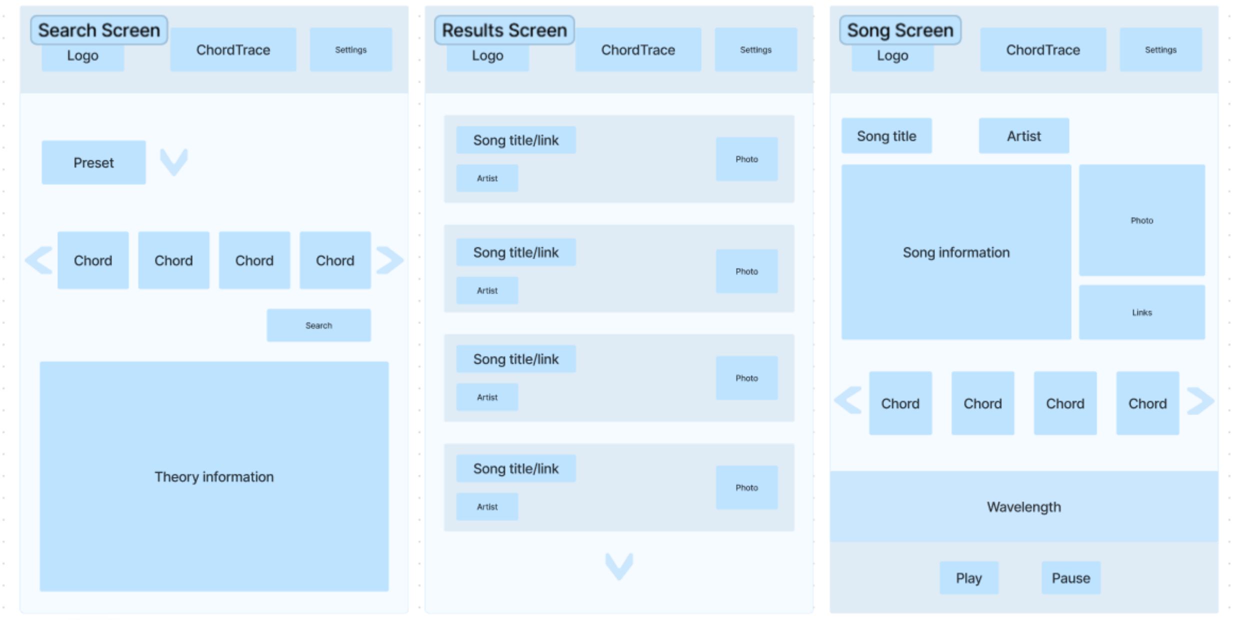


Figure 2. Wireframe made using Figma.

In Figure 2, there are three screens: the search screen, the results screen, and the song screen. In each screen, there are squares that represent an area of the screen, with each square labelled by that area's expected content. Furthermore, there are arrows which indicate that the area could be scrolled though or opened.

The search screen has three main sections: the presets dropdown, the chord list, and the theory information section. The preset dropdown will allow people to select a preset if they want to, correlating to RQ1. The chord buttons will allow users to select their desired chords as well as make a search which correlates to RQ2 and RQ4. The theory information section will show theory information based on the previous choices the user has make, for example preset or chord information, corresponding to RQ3.

The results screen consists of multiple cards that show songs with the chords that have been searched. The list can be browsed, and a song can be clicked on to view a song page, which correlates with RQ5 and RQ6.

The song screen shows the song's information at the top, with the chords below it, corresponding to RQ7 and RQ8. At the bottom of the screen there is a play and pause button, as well as a waveform, which will allow the user to listen to the song, correlating to RQ9.

4.3 Implementation

Within my source code I have a folder labelled 'Screens'. This folder contains three screens which correspond to the screens discussed in the requirements and shown within Figures 1 and 2.

'SearchScreen.tsx' contains the code for the search screen. The code from this file allows the user to select a preset, select a set of chords, view the corresponding terminology, and make a search. 'ResultsScreen.tsx' corresponds to the results screen shown within Figure 1 and Figure 2, which allows the user to view a list of songs and make a selection. Finally, 'SongScreen.tsx' corresponds to the song screen, which allows a user to view a song's information, including the chords and the waveform, and play the songs.

These screens follow all the main requirements listed in Chapter 3.1, and follow the designs suggested within this chapter. The UX flow diagram shown in Figure 1 accurately portrays the flow of the application, and the wireframe shown in Figure 2 was used to structure each screen.

Chapter 5: **User testing and usability**

This chapter highlights the importance of user testing within a software engineering project and details the methodology and findings of said testing.

Disclaimer: this user research is exempt from needing ethical approval as it is a service evaluation, which is exemption 3 on the Ethics Committee website (*EECS – Devolved School Research Ethics*, no date). It follows exemption 3 as:

- The aim of the research is to improve my application.
- The conclusions derived from this research is solely used to improve the application and is not generalisable.
- The conclusions will not be externally published.

5.1 What is user testing and usability?

User testing is a form of testing that provides product feedback based on the perspective of potential users (Rosenbaum *et al*., 2010). I will be using this form of testing to test usability within my application.

Usability is the extent to which a given software fulfils its purpose, whilst also being easy to learn and use by a potential user (Niranjanamurthy *et al*., 2014). Abbasi *et al*. (2023) stated that “a faulty interface design may prevent users from performing tasks correctly”. To ensure that a system can be used to fulfil its purpose, usability needs to be considered within its development.

Usability can be measured by considering 5 key questions (Niranjanamurthy *et al*., 2014):

- To what extent is the system error free? This refers to how well a system prevents and recovers from errors.
- To what extent is the system efficient? This refers to how quickly a user can use a system to complete a task.
- To what extent is the system learnable? This refers to how well a user can learn how to use a system to complete a given task.
- To what extent is the system satisfactory? This refers to how much a user enjoys using the system.
- To what extend is the system memorable? This refers to how well a user can remember how to use a system to complete a given task.

5.2 Methodology

To conduct my research, I have decided to do a questionnaire. This is due to the fact that questionnaires can be easy to carry out, while still providing a high level of knowledge and insight. The questionnaire will be taken by a group of five people, regardless of whether they know anything about music theory. This will be done to see whether non-musicians can understand how to use the application without any confusion.

To make sure that all the questions can be answered by the participants, I made a list of tasks that the participants will have to accomplish whilst testing my application. These tasks lead the participants through the main functions of the application, getting a good overview of how my application works. These tasks can be listed as such:

- Task 1: Select a chord preset from the dropdown menu.

- Task 2: Select a set of chords.
- Task 3: Make a search.
- Task 4: Browse the song results.
- Task 5: Select a song.
- Task 6: Play the song.
- Task 7: Interact with the waveform.

Once the tasks have been completed by the participant they can answer the questionnaire. I will be using a mixture of qualitative and quantitative questions to get the most out of the questionnaire.

I decided to ask these quantitative questions:

- On a scale of 1-5, where 1 is highly disagree and 5 is highly agree, give a rating for each of these statements:
    - The application is efficient and completes tasks quickly. Each task was able to be completed in an acceptable time frame.
    - The application is easy to learn how to use. Each function of the application was easy to understand and find upon first use. There was no confusion with how to complete tasks.
    - The application is easy to remember how to use. The user can easily remember how to complete each task within the application.
    - The application is satisfying to use. The application looks aesthetically pleasing and feels enjoyable to the user.
    - The application is error free or deals with errors if they occur. No errors occurred whilst completing the tasks given or the application deals well with errors by informing the user of the problem and how to avoid it.

Each of these statements corresponds to the usability components: efficiency, learnability, memorability, satisfiability, and error prevention. The mean rating for each of the statements will be calculated, to find out what is thought about each usability component in correlation to my application on average. By comparing these averages, I can find out which areas need to be improved, and which needs the most attention.

After each of these quantitative questions is a qualitative question:

- If you gave a score below 5, please explain your answer in the text box below.

This question helps shed some insight into why the participant does not fully agree with this statement. This will pinpoint what areas of the application need to be improved so that I can maximise the functionality and user experience of my application.

5.3 Findings and analysis

5.3.1 Quantitative results on the average scores of each usability component

Figure 3 shows a bar chart with the average score for each usability metric. An average of 5 suggesting that no further improvements can be made, and an average of 0 suggesting that the application needs significant improvement for that area of usability.

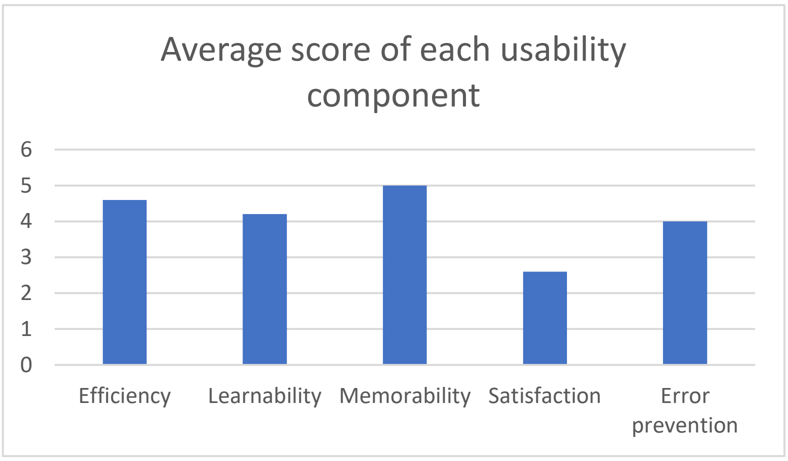


Figure 3: Average score of each usability component.

As you can see from Figure 3, only one usability metric received an average of 5: memorability. This shows that while my application does not need further improvement in this field, the other usability metrics need more attention. This suggests that my application needs to be improved in tangent to the severity of these scores, with the satisfaction metric needing the most attention.

5.3.2 Memorability

As discussed in Section 5.3.1, memorability was the only metric that received a maximum score. As such, there was no feedback given within the qualitative question for this metric.

5.3.3 Efficiency

For efficiency, the participants expressed concern about the number of loading screens within the application. However, each main screen of my application only has one loading sub-screen. Upon opening the application, there is a loading screen that ensures that all the chord sounds are loaded and ready to use. This is to guarantee that when the chords are clicked within the search screen, each chord sound is played without a delay, as it takes time for each sound to be loaded. When a search is made, and the user is redirected to the results screen, there is another loading screen. Within this screen, two API calls are made, and the song results cannot be shown without the data from these API calls. Similarly, when a result is clicked and the user is redirected to the song screen, there is another loading screen in which an API call is made, and the song is loaded before the song screen can be shown.

Both the loading of sounds and the API calls can be difficult to speed up, and as each screen cannot be shown without this data, the loading screens are necessary for the application to function. Otherwise, the score for this component is high, and does not need as much attention as the other metrics.

5.3.4 Learnability

When asked whether the application was easy to learn, users gave feedback on three areas: the dropdown menu, the chord selection area, and the song page redirection area. Figure 4 and Figure 5 show how the dropdown menu looks when it is closed and when it is open.

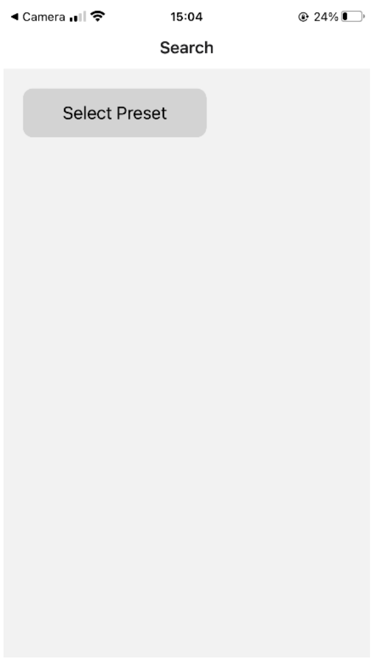


Figure 4: Closed dropdown menu.

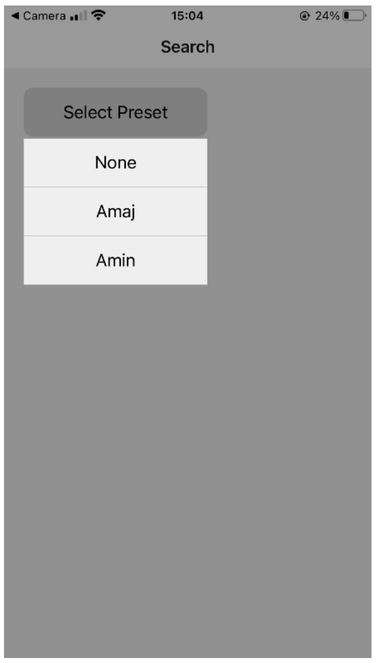
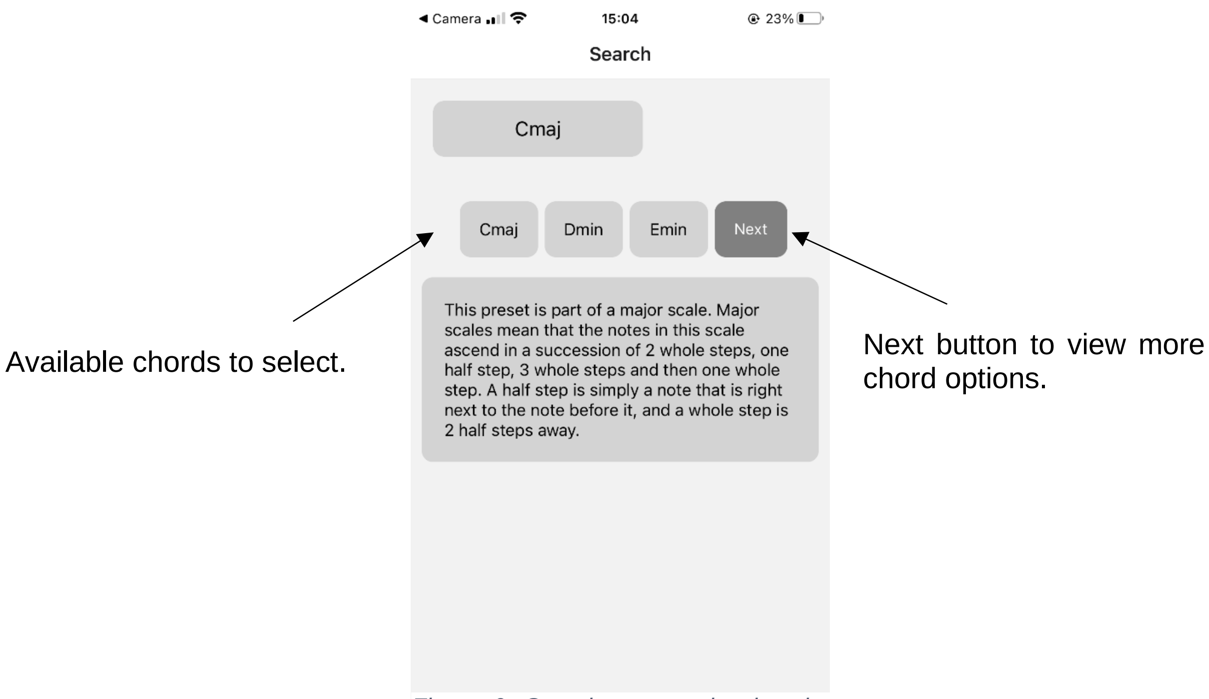


Figure 5: Open dropdown menu.

One participant explained that they could not recognise the dropdown menu as a dropdown menu, making the task of using it difficult to understand. Another participant stated that while they easily recognised the dropdown menu, at first, they did not realise there were other options apart from the three above, as the scroll bar was only visible once scrolling started. An easy way to rectify this mistake is to change the styling of the dropdown menu, adding a dropdown symbol, as well as making the scrolling more apparent.

Moving on to the chord selection area, Figure 6 depicts the available chords to select as well as the next button within the search screen.


◂ Camera 15:04 23%
Search
Cmaj
Cmaj
Dmin
Emin
Next
Available chords to select.
Next button to view more chord options.

This preset is part of a major scale. Major scales mean that the notes in this scale ascend in a succession of 2 whole steps, one half step, 3 whole steps and then one whole step. A half step is simply a note that is right next to the note before it, and a whole step is 2 half steps away.

Figure 6: Search screen showing the chord selection area.

One participant mentioned that when trying to complete a chord search, they were not immediately aware that there were more chords that they could choose from. This is due to the appearance of the next button, which as shown above, is a button with the word 'Next' on it. Typically, within applications, a next button is shown as an arrow rather than text. As I did not adhere to this standard, the participant was confused, and perhaps mistook the button as another chord. To address this mistake, I will change the buttons to arrows to fit expectations and make its use more known.

Regarding the song redirection area, Figure 7 shows where the user must click to be redirected to the song screen.

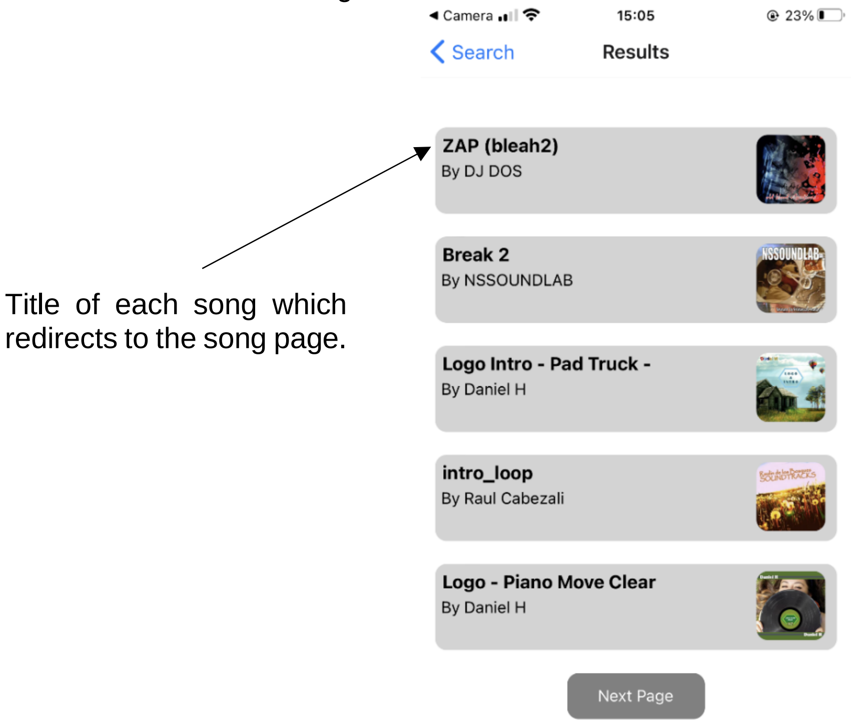
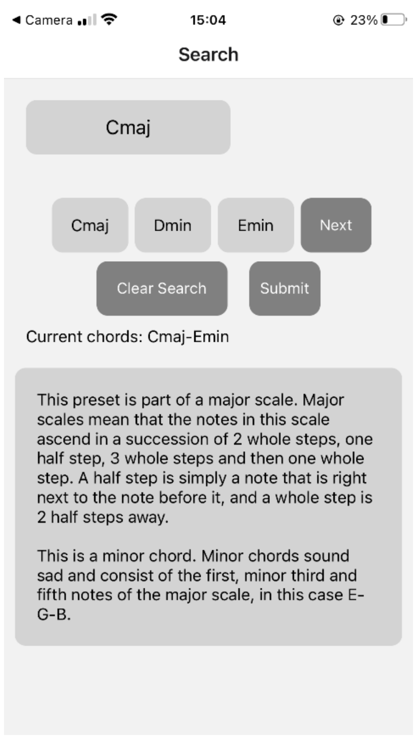


Figure 7: Results screen labelled with the redirection area.

Participants expressed confusion when trying to find out how to access the song page. To redirect to the song page, the users must click on the title of the song, however a couple of participants instead tried to click on the album cover image. To fix this, I will either make it possible for the user to be redirected to the song page through the album cover image or through a different method.

5.3.5 Satisfaction

For satisfaction, participants expressed disappointment in the design and layout of the application. Figures 8, 9, and 10 show the current layout of the application.

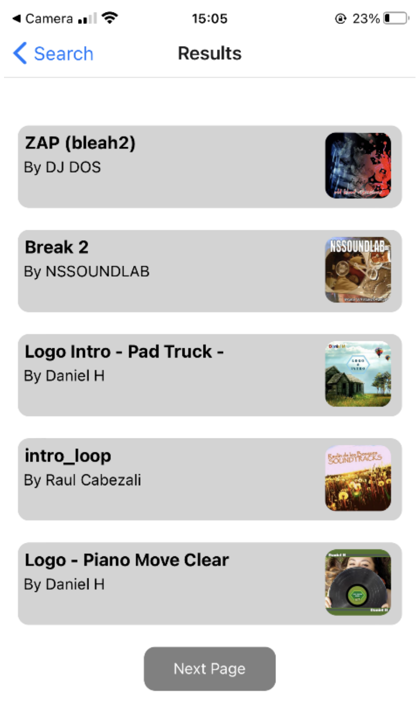


Current chords: Cmaj-Emin

This preset is part of a major scale. Major scales mean that the notes in this scale ascend in a succession of 2 whole steps, one half step, 3 whole steps and then one whole step. A half step is simply a note that is right next to the note before it, and a whole step is 2 half steps away.

This is a minor chord. Minor chords sound sad and consist of the first, minor third and fifth notes of the major scale, in this case E-G-B.

Figure 9: Search Screen



ZAP (bleah2)
By DJ DOS

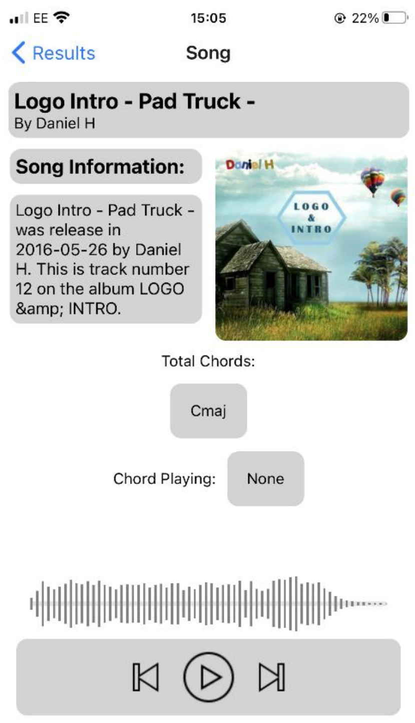
Break 2
By NSSOUNDLAB

Logo Intro - Pad Truck -
By Daniel H

intro_loop
By Raul Cabezali

Logo - Piano Move Clear
By Daniel H

Figure 10: Results screen.



Logo Intro - Pad Truck -
By Daniel H

Song Information:

Logo Intro - Pad Truck - was release in 2016-05-26 by Daniel H. This is track number 12 on the album LOGO &amp; INTRO.

Figure 8: Song screen.

The participants stated that while the styling was consistent in each page, the colour scheme was dull and overall, the application was not aesthetically pleasing. This metric had the lowest rating, and as such I will take the most time to fix the appearance of the applications so that potential users can enjoy using it.

5.3.6 Error prevention

For error prevention, participants had two main concerns: the chord buttons and the song page. Some of the participants expressed that they were not able to click all the chord buttons, or that they would occasionally have to double click for the button to work. This is due to the React Native Pressable component that I used to create the chord button. The Pressable component wraps around the React Native Text component that hold the name of the chord, however that also means that the button only works when the text is clicked, rather than the whole button. To fix this, I must wrap the Text component in another component that is the same size as how the button looks. Another concern that participants had was that when they were redirected to the song page from the results page, the song page occasionally got stuck on the loading view. This is due to the fact that some songs are not downloadable, and the code within the song page did not allow the rest of the screen to load without the audio content being downloadable. To rectify this, I can load the song content and audio content separately, and not show the player but still show the rest of the content when the song is not downloadable.

Chapter 6: **Unit testing**

This chapter underlines the importance of unit testing within a software engineering project. Additionally, it details how unit testing has been carried out to benefit my project and reveals my findings.

6.1 What is unit testing and why is it important?

Unit testing is a form of testing in which the code is broken down into small sections, i.e. units, and tested (RuneSon, 2006). Unit testing is highly beneficial as not only does it test whether the code is completing its purpose, it also highlights which parts of the code are low-quality throughout the process of coding, allowing the project to be built efficiently (Khorikov, 2020).

The main reason I am conducting this test is that occasionally while writing code, mistakes within functions can easily go unnoticed if the function runs with no errors. To ensure that each section of my code is completing its given task correctly, I need to test my functions with pre-written expected outcomes for each test. If the outcome is not as expected, flaws within the code can be found, and the functions can be amended.

6.2 Methodology

To accurately test my code, I first had to make it easier to access each of my separate functions. To do this I extracted the necessary functions out of their respective files and placed them all into one file, labelled ‘utils.ts’. This file exports the functions so that other files can use it, making it easier to do testing as I could import them all from one file. Then I deleted the functions from their original files and imported them from the utils.ts file to ensure that there would be no duplicate code. I extracted as many functions as possible, however I had to leave those that were deemed too simple to test, or those that were too difficult to extract, within their original files.

After this was completed, I was then able to write tests for each of my functions. For my testing I was not able to use a framework, as the testing framework available for React Native test UI rather than functions. Therefore, I had to write out each test and comparison manually.

Within each test, I had a set of subtests. Multiple subtests allowed me to test each of my functions with different parameters, which increased the likelihood of finding defects within my code. This is due to the fact that some of my functions had conditions within them, meaning that not all the code could be validated with one test. Multiple subtests could however be used to test the entirety of the function.

For each subtest, I had to create a set of variables for the parameters needed to call that function, a variable to store the outcome of the function call, and a variable to store the expected outcome of the function call. Then the outcome of the function call and the expected outcome would be compared to discover whether the function has completed its given task correctly.

This is an example of a function that was tested:

```
export function generateTheoryString(chord: Chord, presetSelected) {
    let theory = "";
    if(presetSelected){
      theory+="\n";
    }
    theory += info.terminology[chord.type];
    theory += chord.notes + ".";
    if (chord.flat) {
        theory += "\n\n" + info.terminology.flat;
    }
    return theory;
}
```

The code above is a function that returns a string about the music theory of a chord chosen by the user. Depending on if a preset has been selected, what type of chord is selected (the modality of the chord, i.e. major/minor), and if the chord is flat, the returned string may be different. Therefore, to correctly test this function, subtests with inputs testing each outcome was created.

This is the test function created for the function above:

```
function generateTheoryStringTest(){
      let testOutcomes="\n";
      let chord = {"name": "Bmaj", "type" : "major", "flat" :
      false,  "notes": "B-D#-F#"} as Chord;
      let presetSelected = true;
      let theoryString = utils.generateTheoryString(chord,
      presetSelected);
      let expectedOutcome = "\nThis is a major chord. Major chords
      sound happy and consist of the first, third and fifth notes of
      the major scale, in this case B-D#-F#.";
      if (theoryString!=expectedOutcome){
          testOutcomes+= "Subtest 1 : Failed.";
      }
      else{
           testOutcomes+= "Subtest 1 : Passed.";
      }

      chord = {"name": "Gbmin", "type" : "minor", "flat": true,
      "notes": "Gb-A-Db"} as Chord;
      presetSelected = false;
      theoryString = utils.generateTheoryString(chord,
      presetSelected);
      expectedOutcome = "This is a minor chord. Minor chords sound
      sad and consist of the first, minor third and fifth notes of the
      major scale, in this case Gb-A-Db.\n\nThis is also a flat chord.
      This simply means that the root of this chord is flat, i.e. a
      half-step below.";
      if (theoryString!=expectedOutcome){
            testOutcomes+= "\nSubtest 2 : Failed.";
      }
      else{
            testOutcomes+= "\nSubtest 2 : Passed.";
      }

      return testOutcomes;
}
```

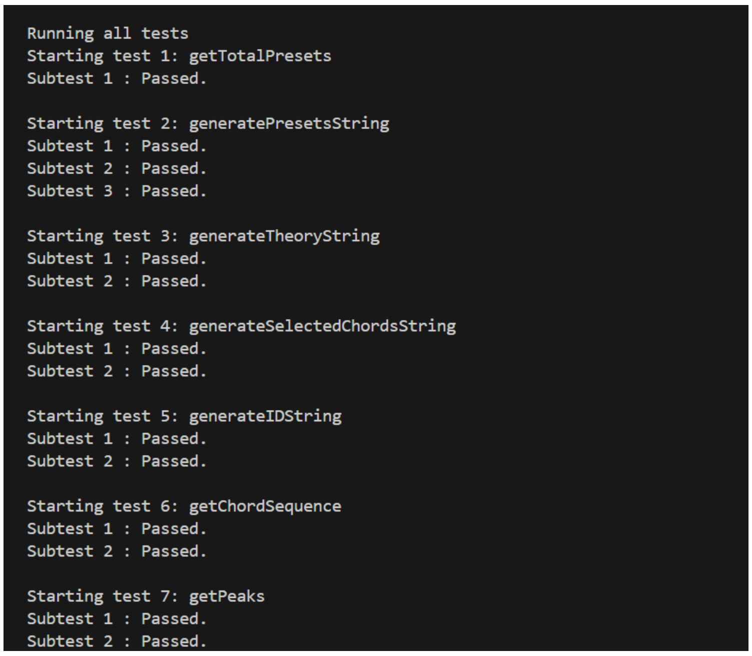
In the test shown above there are two subtests. The first subtest has a preset selected, the type of the chord is major, and the chord is not flat. The second subtest does not have a preset selected, the type of the chord is minor, and the chord is flat. This enables the test function to test each line of the original function as well as testing all the chord values that could change the outcome of the original function. After all subtests have been executed, the function returns whether each subtest has failed or not, which allows me to better understand what area of the code needs adjustment if one of the subtests fail.

Noticeably, there are no control tests in the code above, meaning there are no tests with undefined or incorrect parameters. This is because the functions being tested do not get called in the actual source code unless the parameters needed for them are correct and not undefined.

Once all the test functions and their subtests have been defined, I call all the tests in one function. This function logs the name of the function being tested, and calls the function, which will then carry out the subtests and log whether they have failed or not.

6.3 Findings

In total, seven units were tested with a total of fourteen subtests. This is the outcome of the tests as shown in the terminal:



```
Running all tests
Starting test 1: getTotalPresets
Subtest 1 : Passed.

Starting test 2: generatePresetsString
Subtest 1 : Passed.
Subtest 2 : Passed.
Subtest 3 : Passed.

Starting test 3: generateTheoryString
Subtest 1 : Passed.
Subtest 2 : Passed.

Starting test 4: generateSelectedChordsString
Subtest 1 : Passed.
Subtest 2 : Passed.

Starting test 5: generateIDString
Subtest 1 : Passed.
Subtest 2 : Passed.

Starting test 6: getChordSequence
Subtest 1 : Passed.
Subtest 2 : Passed.

Starting test 7: getPeaks
Subtest 1 : Passed.
Subtest 2 : Passed.
```

Figure 11: Unit test outcome.

Despite thorough testing, none of the fourteen subtests failed, and all outcomes were as predicted. This means that all the unit tests showed no visible errors, and that all functions were successful in completing their desired tasks.

Chapter 7: **Conclusion**

This chapter gives a final overview of my project by listing my achievements, challenges, and limitations. Finally, I suggest ideas for future improvements, and how these improvements could enhance my application.

7.1 Achievements

In the introductory chapter, I listed a set of objectives that I wanted my project to accomplish. Given my dedication to the project, each of the objectives were completed and expanded upon. To summarise my objectives, I aimed to make an iPhone music application that allowed the user to search for songs by chord, which could double as a music education application and a streaming application.

Initially, I had aimed to only make basic observations on music terminology in the application. In the end, I managed to include music theory on chords, scales, and modality, as well as some basic ear training by including the sounds of each chord used in this application. In terms of the music streaming aspect of this application, I aimed to have a chord search that redirected to a list of results which when clicked on would reveal a song page with the song information, a waveform, and a way to listen to the song. I achieved this, going the extra step to make my own customized player, and making the waveform fully interactive by adding a slider.

Alongside the functionalities of this application, I have acquired new knowledge about making mobile applications and the process of development within a software engineering project. Prior to the development of this application, I had no knowledge of how to create a mobile application, or the languages involved. To undertake this project, I had to learn React Native from scratch, which is an accomplishment.

In terms of software development, I learned how to effectively design my application in accordance with the functional requirements of the application, creating a UX flow diagram and wireframe. This was beneficial in helping me effectively plan out how the functions of the application will link together, whilst also structuring each screen. This made the process of creating my application easier, speeding up the process of development.

Moreover, I learned how to test my application using two different methods: usability testing and unit testing. Prior to the development of this application, I was unaware of how these forms of testing were beneficial, or how to approach carrying them out. In the end, these testing methods were advantageous in finding flaws within the user interface of application and the underlying functions used to make the application.

In conclusion, I have succeeded to accomplish my aims and objectives for this project. Furthermore, the knowledge that I have acquired in the completion of this application will be useful when undertaking similar tasks in industry.

7.2 Challenges and limitations

Within the development of this project, I came across challenges and limitations that I had to work around. The main challenges that I came across was the limitations of the tech-stack that I was working with, and the difficulties associated with the relying on APIs to develop my application.

This includes the limitations of using Expo CLI rather than React Native CLI. Midway through the development of my application, I came to the realisation that in comparison to React Native CLI, there were a limited number of packages available to use. This is due to the fact that React Native CLI is used to write code natively, having packages that link to native modules, whereas Expo CLI does not allow users to interact with the native parts of their application (Boduch *et al.*, 2022). This caused me to use packages that were occasionally not the best equipped. For example, for the sound in my application, I had to use the expo-av package, as this was the only audio package available for Expo CLI. Due to the limitations of this package, I had to create a song player from scratch. Moreover, one of the main objectives for my application was to display a waveform for each song, and I had aimed to do this by extracting waveform data from the audio. Unfortunately, expo-av could not accomplish this task. Initially I used the Deezer API for the song data, but in order to have access to waveform data, I had to switch to the Jamendo API. I was two-thirds of the way through completing my application when I made this decision, meaning I had to rewrite a section of my code, losing valuable time. Nevertheless, using Expo CLI rather than React Native CLI was still a wise decision, as it allowed for me to easily test my application.

Another challenge I had to face was relying on multiple APIs to get song data. Throughout development, there were occasions in which the APIs were down or had issues, meaning that I could not use them at those given times. To combat this challenge, I often had to work on other parts of the code that did not require API data or mock data when it was absolutely necessary.

Despite all the challenges and limitations, I was able to work around them to the best of my abilities to create my application and fulfil all my aims. Most importantly, these challenges taught me the importance of researching prior starting a project, as some of the challenges could have been avoided with a different tech-stack.

7.3 Future improvements

There are a list of improvements and additional features that could be added to my project in the future. Some of the improvements include making changes to my tech-stack, whilst others are more concerned with the overall functionality of my application.

One of the limitations listed in Section 2 of this chapter was that Expo CLI has less packages available to use compared to React Native CLI, which complicated my project and forced me to use the Jamendo API rather than the Deezer API. The Jamendo API has less songs than the Deezer API, and the songs from this API are not as popular. If this application were to be released as is, people might not want to use the application from the lack of available songs, and because the songs are not as high quality as those from other music streaming platforms. Consequently, in the future I would like to change from Expo CLI to React Native CLI, simplifying my code by adding more complex packages and changing from the Jamendo API to the Deezer API.

In terms of functionality, more features could be added to improve both the music theory and the music streaming aspects of the application. For music theory, I could add features that teaches the user how to play songs and chords. This would include creating chord charts that teaches users the finger placement of chords on an instrument chosen by the user. Then within the song page, the chord charts could be shown in time with the lyrics of the song. This feature would not be for the casual users of this application but would instead just cater to the musicians using the application. In addition to this feature, I could also add different methods of searching that includes different theory aspects. For example, the user could search songs by instruments or melody rather than by chord. Adding these type of search features would expand the ways the application could be used and enable me to add more music theory information on instruments in general, as well as theory on notes and scales.

To improve the music streaming aspect of this application, it would be beneficial to add a backend, which could keep track of the users and their data. With a backend in place, users could log in to see their previous searches and liked songs, also giving them the ability to create playlists and view what other users are listening to. This would make the application more like other music streaming platforms, enticing the users to use it more consistently.

Overall, the features and improvements listed above would highly enhance my application. Due to the time constraints for this project, these features were not included in the plan, however these could all be considered for future improvement.

References

- Van den Oord, A., Dieleman, S. and Schrauwen, B. (2013) ‘Deep content-based music recommendation’, in *Advances in neural information processing systems 26: 27th Annual Conference on neural information processing systems 2013; December 5-10, Lake Tahoe, Nevada, USA; proceedings*. Curran Associates, Inc. Available at: https://proceedings.neurips.cc/paper_files/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf (Accessed: 11 November 2023).
- Wang, A., 2021. Models of student engagement in music education classroom in higher education. *Frontiers in psychology*, *12*, p.738207.
- Jain, S., Grover, A., Thakur, P.S. and Choudhary, S.K. (2015). ‘Trends, problems and solutions of recommender system’, *International Conference on Computing, Communication & Automation*. Greater Noida, India, 15-16 May, pp. 955-958. doi:https://doi.org/10.1109/CCAA.2015.7148534.
- Slaney, M. (2011). Web-Scale Multimedia Analysis: Does Content Matter? *IEEE Multimedia*, 18(2), pp.12–15. doi:https://doi.org/10.1109/mmul.2011.34.
- Spotify (2023) *Web player: Music for everyone*, *Spotify*. Available at: https://open.spotify.com/ (Accessed: 24 November 2023).
- Apple (2023) *Get Millions of songs. all ad-free.*, *Apple Music - Web Player*. Available at: https://music.apple.com/us/browse (Accessed: 24 November 2023).
- Deezer (2023) *Listen to music: Online music streaming platform*, *Deezer*. Available at: https://www.deezer.com/en/ (Accessed: 24 November 2023).
- SoundCloud (2023) *Stream and listen to music online for free with SoundCloud*, *SoundCloud*. Available at: https://soundcloud.com/ (Accessed: 24 November 2023).
- *Search songs by chord, Progression, and decade* (2021) *The Chord Genome Project*. Available at: https://www.chordgenome.com/ (Accessed: 08 March 2024).
- *Chord progression trends* (2024) *Songs With The Same Chords*. Available at: https://www.hooktheory.com/trends (Accessed: 08 March 2024).
- *Choose songs by selecting chords* (2024) *GuitarPlayerBox*. Available at: https://www.guitarplayerbox.com/select/chords/find/songs/ (Accessed: 08 March 2024).
- *Instant chords for any song* (2024) *Chordify*. Available at: https://chordify.net/ (Accessed: 08 March 2024).
- Pauwels, J. and Sandler, M. (2019). A Web-Based System For Suggesting New Practice Material To Music Learners Based On Chord Content. *IUI Workshops*.
- Dabit, N. (2019). *React native in action : developing iOS and Android apps with JavaScript*. Shelter Island, NY: Manning Publications.
- Boduch, A., Derks, R. and Sakhniuk, M. (2022) *React and react native: Build cross-platform JavaScript applications with native power for the web, desktop, and Mobile*. Birmingham, UK: Packt Publishing Ltd.
- Rippon, C. (2023). *Learn React with TypeScript*. Packt Publishing Ltd.
- Goldberg, J. (2022) *Learning typescript: Enhance your web development skills using type-safe JavaScript*. Sebastopol: O’Reilly Media.
- Zammetti, F. (2018) *Practical react native: Build two full projects and one full game using react native*. Berkeley, CA: Apress.
- Deezer (no date) *Deezer for developers*. Available at: https://developers.deezer.com/api (Accessed: 22 November 2023).
- SoundCloud (no date) *API - Guide - SoundCloud Developers*. Available at: https://developers.soundcloud.com/docs/api/ (Accessed: 22 November 2023).

- Jamendo (no date) *Jamendo API documentation*. Available at: https://developer.jamendo.com/v3.0/docs (Accessed: 01 March 2024).
- AltexSoft (2021) *Functional and Nonfunctional Requirements: Specification and Types*, *AltexSoft*. Available at: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/ (Accessed: 20 November 2023).
- Browne, C. (2023) *What are user flows in UX design? [full beginner's guide]*, *CareerFoundry*. Available at: https://careerfoundry.com/en/blog/ux-design/what-are-user-flows/#:~:text=User%20flows%2C%20UX%20flows%2C%20or,through%20to%20the%20final%20interaction. (Accessed: 22 November 2023).
- Hannah, J. (2023) *What is a wireframe? your best guide*, *CareerFoundry*. Available at: https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/ (Accessed: 22 November 2023).
- *EECS – Devolved School Research Ethics* (no date). Available at: https://qmulprod.sharepoint.com/sites/EECS-DevolvedSchoolResearchEthicsCommittee?xsdata=MDV8MDJ8fGU5MmJkNWNjNTJhOTRjZmM5YjY2MDhkYzM5M2RiNDQ0fDU2OWRmMDkxYjAxMzQwZTM4NmVlYmQ5Y2I5ZTI1ODE0fDB8MHw2Mzg0NDgxODQzNTk5NTEyMDZ8VW5rbm93bnxWR1ZoYlhOVFpXTjFjbWwwZVZObGNuWnBZMlY4ZXlKV0Iqb2lNQzR3TGpBd01EQWlMQ0pRSWpvaVYybHVNeklpTENKQlRpSTZJazkwYUdWeUlpd2lWMVFpT2pFeGZRPT18MXxMM1JsWVcxekx6RTVPbGMzTjJWUGEydHVNVGxpZE1Razg1UVd0a1VGWlBTMFpXYmxVeU1sVk5ZbWQzZW5wblQyc3hRSFJvY21WaFpDNTBZV04yTWk5amFHRnVibVZzY3k4eE9UcFhOemRsVDJ0cmJqRTVZbGxqVFdKblRFSlBPVUZyWkZCV1QwdEdWbTVWTWpKVlRXSm5kM0A2WjA5ck1VQjBhSEpsWVdRdWRHRmpkakl2YldWemMyRm5aWE12TVRjd09USXlNVFI6TkRnNU13PT18NTlhNzk3ZDUyYjA2NGE0ODM5MmYwOGRjMzkzZGI0NDJ8MWU4ODY1Mzk1NzAwNGQ2MTgyYTVlMzVhYTg4MDRkYzI%3D&sdata=ZVh5UlZwQjR0MUZRY2JjYWpCSEE3OXVvV3dWcG9nWFNyMTgvQnZjZmRlcz0%3D (Accessed: 16 March 2024).
- Rosenbaum, S.E., Glenton, C., Nylund, H.K. and Oxman, A.D. (2010). 'User testing and stakeholder feedback contributed to the development of understandable and useful Summary of Findings tables for Cochrane reviews', *Journal of Clinical Epidemiology*, [online] 63(6), pp.607–619. doi:https://doi.org/10.1016/j.jclinepi.2009.12.013.
- Niranjanamurthy, M., Nagaraj, A., Gattu, H. and Shetty, P.K., (2014). 'Research study on importance of usability testing/User Experience (UX) testing', *International Journal of Computer Science and Mobile Computing*, *3*(10), pp.78-85.
- Abbasi, M., Sarmento, G., Mota, M., Martins, M., Sa, F. and Cardoso, F. (2023). 'Comparing User Experience: An Analysis of Usability Testing Methods for Mobile Applications', *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, Aveiro, Portugal, 20-23 June. Aveiro: IEEE, pp. 1-10.
- Runeson, P., 2006. A survey of unit testing practices. *IEEE software*, *23*(4), pp.22-29.
- Khorikov, V. (2020) *Unit testing principles, practices, and patterns*. New York, NY: Manning Publications.