COM814: Project 2015-16

Dissertation

School of Computing & Information Engineering

Niall O'Boyle – B00584699

Mobile Music Tutor - Guitar

Supervisor: Dr. Inaki Rano

Second Marker: Dr. Tom Lunney

1st September 2016

Plagiarism Statement

"When submitting your work you are agreeing with the following statement: I declare that this is all my own work and that any material I have referred to has been accurately referenced. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file."

Signed – Niall O'Boyle

Acknowledgements

Particular thanks should be directed to my project supervisor, Dr. Inaki Rano, for his excellent guidance and continual support throughout the duration of this project. Inaki was always available to provide assistance in any way that was required, and kept in regular contact with me throughout the project, which is greatly appreciated. I would also like to thank my second marker, Dr. Tom Lunney, for his guidance and feedback provided throughout the project duration.

I would like to express my gratitude to all of the respondents who completed the questionnaires for the purposes of research and evaluation. Their excellent feedback was vital to the success of the project as a whole. For them to be so generous in giving their time to assist in the project is greatly appreciated.

A final thank you should be given to my family, for their constant support and encouragement throughout the past year.

Table of Contents

Tables

Figures

Abstract

Music plays a hugely important role in the lives of people throughout the world. As a result of this, millions of people are inspired to learn their favourite instrument. Beginner guitar players traditionally find themselves faced with two choices – they can either seek education from a professional guitar teacher, or they can attempt to teach themselves through various resources, namely the internet, or relevant literature. These methods have been used by people for decades, with varying levels of success. While they undoubtedly have merit, there are several significant flaws associated with each approach. Professional education can be prohibitively expensive, while self-education can lead to bad habits being learned. Given the modern reliance on smartphones and applications, it is unsurprising that there are a large number of guitar-focused apps already available on the market. However, they are of very little use to a beginner player. The market-leading applications are aimed at much more advanced players, and provide little to no guidance or assistance to beginners. Given the current options available, it is little surprise that so many aspiring players quickly become frustrated, and give up.

This dissertation details the work undertaken to develop an Android application that would provide a complete educational experience for beginner guitar players, and could also be used to complement education from a professional tutor, or a user's own educational methods. This application would provide explanations and guidance for some of the core concepts of guitar playing, including chords, scales, tuning, and troubleshooting any problems which may occur. Given the much vaunted educational benefits provided by audio-visual features, video tutorials and other multimedia files are used throughout the application, to provide clear demonstrations and the best educational experience possible for the user. A list of songs will also be provided for the user, to allow them to sharpen their skills and implement what they have learned in a practical way. If the user finds a song that they particularly enjoy playing, or one which they wish to practice, they can either create an account or sign in and favourite the song, allowing easy access to songs with which they have a particular affinity.

The application was designed to be used on an Android tablet with a screen size of 7 inches. Prior to any development, questionnaires were issued to a range of individuals, including experienced guitar players, professional guitar teachers, and those that have never played the instrument. Based on these responses, a prototype of the application was designed and created. This prototype was then tested, and any issues that were discovered were quickly resolved. After further development and testing, the application was given to various users with a wide range of musical experience, in order to evaluate the success of the app, both in terms of its operation and its use to beginner players. Any issues that were encountered were considered and, along with any feedback provided by the respondents, recommendations for the application's future development were made.

Keywords: *guitar, musical instrument, music education, Android application.*

Chapter 1: Introduction

Music is an intrinsic part of people's lives; therefore it is unsurprising that so many people wish to learn to play a musical instrument. Playing an instrument is one of the most popular leisure activities in the UK, with 10.4% of the population playing an instrument of some kind (Department for Culture, Media and Sport, 2015, p. 3). Alongside this, there are numerous health and social benefits associated with music, and it can even "lower blood pressure, decrease heart rate, reduce stress, and lessen anxiety and depression" (Cicetti, 2013).

However, the actual process of learning to play an instrument can be particularly troublesome. A beginner can either seek professional education, or can attempt to educate themselves. In seeking tuition from a professional music teacher, the player will undoubtedly receive excellent guidance, but there are significant problems associated with this approach. Lessons can be quite expensive - the UK average for a one hour lesson with a qualified teacher is £30 (Incorporated Society of Musicians, 2015). It is certainly conceivable that a large number of lessons may be required; therefore the overall outlay of money could prove to be extraordinary. The social implications of this are hugely significant – there are few families who could afford to pay these sums for guitar lessons alongside other household bills. Therefore, those from a lower socio-economic background are either discouraged from taking lessons, or simply do not have the means to access them, and are thus being deprived of a higher quality of music education. It can also be quite tedious, as the player is forced to learn quite dry and complex information at an early stage, and this could leave the player frustrated, confused, and, frankly, bored.

Regarding the possibility of self-education, the player is free to plan their own lessons and learn whatever they wish. While this may be more enjoyable, and provides additional flexibility, it will inevitably harm the player's development. There is a monumental amount of information available to beginners, mainly through books and the Internet. Often, this information can be difficult to understand, and may even directly contradict information elsewhere. The user will inevitably pick up bad habits or incorrect techniques, which may negatively affect the player's playing style and technique in the future, often making it harder for the player to play at a more advanced level. With these potential hurdles facing a beginner guitar player, it is unsurprising that so many quickly become disheartened and frustrated, with 90% of beginner guitar players quitting inside their first year of learning (Ultimate Guitar, 2015).

Technology assists us in virtually every aspect of our lives, from banking to fitness to commerce. It has become ubiquitous in modern life. Given the proliferation of technology and applications throughout society as a whole, one would naturally assume that an application exists that is aimed at aspiring guitar players, to provide a clear and thorough educational experience. However, this is not the case. The market leading guitar-focused application, Ultimate Guitar, is quite obviously aimed at much more advanced players, providing a huge database of songs to learn, but providing little assistance or guidance to beginner players. Other popular applications pose a similar problem, in that they are of very little use to a beginner guitar player.

This rather startling fact presented a clear opportunity for the developer – to create an Android application specifically aimed at, but not limited to, beginner guitar players. This application would provide clear explanations and assistance regarding the core fundamentals of playing the guitar, especially in relation to playing chords and scales, how to tune the guitar, and how to play songs, through the chord sheets provided in the app. Video lessons will also be provided in the "Lessons" section. This follows the current trend in education in transitioning towards a more Technology Enhanced Learning (TEL) approach, which provides a much "more active learning experience through interacting with various technologies and multi-media resources" (The University of Sheffield, 2016). The application will provide crucial knowledge and information to beginners, and could be used to complement either professional education or self-education, providing resources to allow the user to practice and sharpen their skills, while also providing them with the flexibility to practice whenever they desire. This flexibility is imperative, especially given the busy nature of modern life.

In order to create this application, careful planning was initially carried out in relation to the features that should be included, as well as the general design of the app itself. Questionnaires were designed and given to individuals who either had experience of learning the instrument, ranging from new players to experienced teachers, or those who had never played but wanted to learn. These responses, particularly in relation to the features that should be included and the various concepts that must be addressed, were critical when designing the application. Subsequent to this, a prototype of the application was developed, and tested. Any issues or bugs that were encountered at this point were resolved, and the app was retested. Further development and testing was undertaken, and once the final stage of testing was complete, the app and a questionnaire were given to a group of individuals for the purpose of evaluation. This evaluation is crucial in two senses – the operation of the application will be assessed, as will the application's appeal and suitability for beginner guitar players. Any potential improvements suggested by the respondents will be considered for the future development of the application.

1.1 Problem Statement

"Learning to play the guitar can be a demanding and time-consuming task, and beginner guitar players can quickly become frustrated. The aim of this project is to provide technology-supported ways to assist a beginner guitar player, either through self-study or to complement professional education."

1.2 Aim

The aim of this project was to develop an Android application that would assist beginner guitar players in attempting to learn the guitar. This app would operate on a tablet, and would be able to be used by a beginner player to complement either professional education or self-education.

1.3 Objectives

In order to accomplish the aim specified above, the following objectives were formulated:

1. Research guitar-focused apps currently available on the market, especially the market leaders, and analyse the strengths and weaknesses present within each one.

2. Create and dispense a questionnaire to individuals of various musical abilities, ranging from those that have never played the instrument to experienced guitar teachers. The responses from these questionnaires will assist in the design, development and implementation of the application.
3. Ensure that the features developed for the application use a variety of resources, in order to provide the best possible educational experience for the user.
4. To develop a user-friendly, aesthetically-pleasing user interface that is easy to navigate, and provides clear and concise information and guidance to the user.
5. Create another questionnaire for the purposes of evaluating the application, both in terms of its operation and its suitability for beginner guitar players. This questionnaire will be dispensed to individuals who have agreed to test and evaluate the application.

1.4 Requirements for Development

The application was designed and created using currently existing hardware and software that is available and easily accessible. The following tools in particular were necessary in order to develop the application, and to address the aims and objectives listed above:

- An Android tablet with a screen size of 7 inches running the Android 4.4 (KitKat) operating system.
- Android Studio, an integrated development environment that will allow the app to be designed, as well as allow for the creation and utilisation of databases through the SQLite Database feature.
- A laptop or desktop computer capable of running the Android Studio development platform in order to create the Android application.
- Programming languages: Java, SQL.

1.5 Dissertation Outline

Chapter 2 will detail the problems inherent with the traditional methods of music education, and why these problems pose a significant issue for beginner players. The results of an investigation into the most popular apps on the market will be analysed, as well as the strengths and weaknesses associated with each app. Information will be provided regarding the importance of Technology Enhanced Learning (TEL), and the benefits it provides within education.

Chapter 3 will identify the end-user and system requirements, as well as the process through which these requirements were created – using questionnaires, and the feedback from respondents of various musical abilities.

Chapter 4 provides an overview of the design options selected for the app. This will also feature a detailed explanation of the platform used, the database and Java class design, and the features and design elements of the user interface and system architecture.

Chapter 5 discusses the implementation of the design created, the process through which this implementation occurred, and the various steps taken and prototypes developed when creating the

final version of the application. This will include a discussion of, for example, the implementation of the database, the main Java classes that were used, and the manner in which the various sections of the application were developed.

Chapter 6 details the procedure under which testing of the application will occur, and the various methods of testing utilised by the developer. The app will be evaluated using in-house testing and respondent questionnaires, and an analysis of this evaluation will be undertaken in this chapter.

Chapter 7 concludes the dissertation. In this chapter, an overall analysis of the final application and an examination of the work carried out will be performed. The success of the application will be judged based on whether or not it satisfied the aims and objectives specified in the Introduction chapter. Further recommendations for future development of the application will also be considered in this chapter.

Chapter 2: Analysis

2.1 Introduction

This chapter provides an overview of the background research undertaken in relation to this project and the creation of the guitarGURU application. This application aims to assist beginner guitar players in learning to play the acoustic guitar by providing a myriad of educational resources, thus allowing the user to learn at their own pace. As well as this, the app can be used in conjunction with the traditional educational methods of seeking professional tutelage and self-education. The information garnered from the background research was collated from a variety of resources, including the internet, journals, newspapers, and the developer's personal experiences with playing the guitar. An analysis will be carried out regarding the various benefits that playing the guitar can have for a person, as well as the methods through which a beginner guitar player would traditionally learn to play the instrument. The importance of technology within education will be discussed, as well as the reasons why this has become an ever-growing trend, before the strengths and weaknesses of the guitar-focused applications currently on the market are dissected. The chapter will conclude with a summary of the key findings from this analysis, as well as the recommendations for the developer's application that have been garnered from these key findings.

2.2 The Benefits of Playing Music

There are millions of people worldwide who play a musical instrument, and there is no doubt that they all decided to learn to play for a myriad of different reasons. There are several significant benefits associated with playing an instrument, which make learning to play even more attractive to potential learners. These benefits, which will be discussed below, relate to both health and social factors.

2.2.1 Health Benefits

Playing a musical instrument has been medically proven to have a hugely positive effect on both the mental and physical health of all players. In terms of the impact upon physical health, increased brain function and sharper response times have been shown to be positively linked to the amount of musical practice carried out by an individual, and playing music has even been described as "*an effective intervention to slow, stop or even reverse age- or illness-related decline in mental functioning*" (Paddock, 2013). Alongside this, the parts of the brain that control motor skills and memory actually increase in size and become more active (Matthews, 2011). One of the most notable health benefits was highlighted by Harvard Medical School, in relation to the positive impact on cardiac function. Playing an instrument, or even simply listening to music, can assist in the recovery from a heart attack, stroke, or other cardiac procedure, and can even lower blood pressure (Harvard Medical School, 2009).

Stress is something that everyone will experience in their lives, and it can have a devastating impact on an individual's health. People who have high levels of both stress and depression are 48% more likely to die or have a heart attack than those who do not suffer from these conditions (Geggel, 2015). Playing a musical instrument has been shown to temporarily switch off the stress responses that are triggered when a person's senses detect a possible threat in their environment (Kuchinskas, 2010), thus leading to a significant reduction in stress levels. Alongside this, playing an instrument also fosters a sense of pride

and self-confidence, decreasing levels of anxiety and depression. In fact, music therapy has even been used in treating children and teenagers with depression, anxiety, and autism (Matthews, 2011).

2.2.2 Social Benefits

Playing a musical instrument can have numerous advantageous outcomes for a player's social life. As has been previously mentioned, learning any new skill, but particularly an instrument, is a tremendous confidence boost, and generates a real sense of achievement, contributing to increased self-esteem and self-worth (Stone, 2014). The process of learning an instrument is frustrating, and there may come a point when the player wishes to give up. However, the attitude to continue to learn builds a player's discipline and perseverance; two characteristics that will serve them well in later life. The player may also find that they now have a much wider social circle of friends, and may even join a band, or meet up with some friends to jam on a regular basis, thus improving their ability to work as part of a team.

However, the most important benefit of playing a musical instrument is perhaps to most basic one – it is fun. Musicians learn an instrument, and continue to play, because they love doing it. It provides a creative outlet for them, allowing them to express themselves in a way that few others can. To be able to do something that you love, that also provides all of the benefits listed above, is perhaps one of the core reasons why there are so many millions of people across the world that play a musical instrument.

2.3 The Traditional Methods of Guitar Education

The guitar is one of the most commonly played instruments in the world, and its universal appeal cannot be overstated. For every individual that can play the guitar, there are many more who have tried but have been ultimately unsuccessful in learning to play the instrument. For decades, there have been two main methods through which an aspiring player can learn to play, and these methods will no doubt continue to remain as popular as ever. A player can either seek professional education from a qualified guitar teacher, or they can attempt to teach themselves, using a wide variety of resources available to them. While these two very different approaches have undoubted merit, and have created some of the greatest musicians the world has ever seen, there are significant problems associated with each, as shall be discussed below.

2.3.1 Lessons from a Professional Guitar Teacher

There is a school of thought that puts forward the idea that the only way to learn guitar in a truly proficient way is to get lessons from a professional; to do otherwise would mean that a player would never truly master the instrument. The rationale behind getting lessons is obvious – the player would get the best possible education and gain a clearer understanding of the instrument. The learning experience is structured and designed in order to ensure that the player is fully competent in a certain area before moving onto a new lesson, and it is the common belief that "if you really want to learn music well, you will do so much faster with a good guitar teacher" (Hess, 2016). However, despite the benefits that this may have for a player's technique, there are notable problems inherent with this approach.

The main problem regarding professional tuition is, undoubtedly, the exorbitant costs. In the UK, the average price for a one hour lesson with a qualified music teacher is £30 (Incorporated Society of

Musicians, 2015). One may not consider this to be an unusually high price, especially given the quality of service provided. This is undeniably true. However, the price becomes an issue when the player becomes dedicated to learning the instrument, and requires more lessons to progress onto the next stage of their development. Bill Evans, the legendary jazz pianist, was correct in his assertion that "you never master the instrument. You always just strive to get better" (Brannon, 2011). Based on this idea, a player may require regular lessons for years, at which point the amount of money spent may prove to be astronomical.

This is not just a problem for the individual who is paying for the lessons; the social implications of this fact are a problem for society as a whole. There are very few families that could afford to pay these sums for guitar lessons alongside other everyday financial outlays and household bills, especially in the current austerity economy. Disposable income for everyday UK citizens has plummeted over the last few years, and shows little sign of drastically improving in the immediate future. In 2015, 10% of the population, or "5 million British adults, ha(d) less than £10 a month left over once they have paid their essential bills", with the average monthly disposable income being £187, which is £38 lower than the 2013 figure (Webb, 2015). This is an astounding statistic, and is indicative of a wider problem in the United Kingdom surrounding the class divide between the rich and poor. With 10% of the population barely being able to pay their bills, it is absolutely out of the question that they could afford music tuition if they desired. Therefore, those from lower socio-economic backgrounds are either discouraged from taking lesson, or simply don't have the financial means to access them.

This is in direct contrast to those from a higher socio-economic background, who presumably have significantly higher levels of disposable income. There are various studies that do indeed prove this hypothesis to be correct. The following statistic comes from an Associated Board of the Royal Schools of Music (ABRSM) 2014 report – "The significant disparity between children from AB social backgrounds and those from social grades C1, D and E is worrying. 90% of children from AB backgrounds will have played an instrument, compared with 80% of children from other social grades. In addition, 74% of children from AB backgrounds have had instrumental lessons compared with only 55% of children from social grades C1 and DE" (ABRSM, 2014, p.17). In this study, AB backgrounds were considered to be the highest socio-economic background, with C1, C2, D, and E all forming the decreasing social statuses. 77.83% of the UK population is classified as being in social grade C1 or lower (UK Geographics, 2014). This again demonstrates how the cost of music tuition is prohibiting the vast majority of the population from seeking professional education. Despite all of the benefits that it provides, it is simply not a viable option for many people. It is unsurprising, given these facts, that "music education is often still the preserve of the rich… (while) poorer youngsters are more likely to miss out" (The Guardian, 2014).

Regarding access to professional education, there is another important consideration that must be taken into account – there appears to be a dearth of qualified guitar teachers available to students, especially in comparison to other, more popular, instruments. In the 2014 ABRSM report referenced previously, a study was carried out in which 4,491 music teachers were surveyed to determine their specialist subjects. Of this sample size, only 7% of the respondents listed "Guitar", with piano dominating the list with 67% (ABRSM, 2014, p.9). Again, this is a surprising statistic, particularly given the obvious desire for professional education. It is clear that in relation to other instruments, the guitar

is under-resourced when it comes to the availability of professional tutelage. Music education as a whole is also grossly under-resourced, especially within schools. There are thousands of students across the UK who benefit from music lessons at school through their education board, who have found that the budget provided for these lessons has been slashed over recent years, with some schools only having an allocated music budget of £2.20 per child. In line with the rest of the public sector, music education has had to cope with significant budget cuts of up to 25% in some areas (Murray, 2014). Depriving students of music lessons at school may force aspiring players to seek private education, at a significantly higher price, which, as discussed, is simply not an option for most people.

Seeking lessons from a tutor may often have the opposite effect to that which was intended. Two of the major reasons that a student chooses to quit lessons are that they either have a bad teacher, or they do not play the type of music that they want to play, perhaps due to the restrictive, and often tedious, schedule implemented by a guitar teacher (RockGuitarPower, 2016). This is a particular problem for children and teenagers, who may take music lessons while at school. Speaking from the developer's personal experience, during the early stages of music education, learning the basic concepts and techniques that are essential to an aspiring musician can be rather dry, and hard to understand. Individuals may quickly become bored and disillusioned with learning the instrument, particularly those that want to move on to the more enjoyable aspects of music. This can have a much more damaging long term effect; a student may be put off playing the instrument for good, and may be unwilling to return to it based on the bad experience that they had previously.

The final significant problem with receiving lessons is the lack of flexibility that this approach provides. Modern life is hectic, and many people may struggle to fit regular music lessons around their busy schedules and other, more important, commitments. This is an unavoidable problem unfortunately, as it is very much dependent on the availability of the student, as well as the availability of the teacher, who may have other lessons or everyday engagements to deal with. Given these problems, it is clear why so many players decide to forego professional education, and try to teach themselves to play the instrument.

2.3.2 Self-Education

The other educational approach available to beginners would be to teach yourself to play the instrument, using a wide variety of resources that would be easily accessible; namely the internet, YouTube lessons, and books and other literature. This is a fairly popular option, with "over a fifth (21%) of young musicians (saying) they had taught themselves rather than having lessons" (BBC News, 2014). There may be a slight stigma surrounding self-education of a musical instrument, in that there is a belief that you will not develop a proper playing style, or won't get a correct education. However, this is unfair, especially in light of the fact that some of the greatest musicians in history have been entirely self-taught, including B.B. King, Prince, Eric Clapton and Jimmy Page. Despite this, as with professional education, there are serious problems surrounding self-education of an instrument.

Self-education can be detrimental to a player's technique and playing style. As the player is not getting direct guidance as to the correct methods of playing, it is very easy to pick up bad habits and incorrect techniques, which may make playing at more advanced levels more difficult. This is by no means an

irreversible problem, as any incorrect techniques can be amended in the future, but it slows down the learning process, and may hinder a player's progression to a more advanced level. Beginners are often in a rush to start playing their favourite songs, often skipping the basics that are essential to a correct playing style. Again, this could make playing much more difficult, and hinder future development.

Speaking from the developer's personal experiences, the monumental amount of information available to beginners may in fact prove to be a hindrance. A wide variety of resources are likely to be utilised by a beginner player, ranging from the internet, to YouTube, to literature, all of which will provide different opinions and approaches, depending on the author. Often, this information will be confusing, and sometimes, directly contradictory to information provided somewhere else. To a beginner, who is unsure of the correct approach, this information is of limited use, especially if you have no knowledgeable person to turn to for advice, as "some things are really hard to understand unless someone demonstrates them to you" (Cudjoe, 2016). In all probability, the Internet will form the basis of the player's education. It is where most of the information can be found, and it is all easily accessible. It is because of this, and the fact that there are so many millions of people who want to learn an instrument, that websites purporting to provide the best ways to teach yourself to play an instrument are widespread. How is a beginner player, who already possesses limited knowledge of the instrument, supposed to discern which information is reliable? It is understandable why so many guitarists proclaim that "my biggest mistake was turning to the Internet to try and find answers to my own personal guitar playing problems" (Freeman, 2016).

It is almost inevitable that the problems mentioned above will lead to intense frustration with the instrument, and may cause the player to become so disheartened that they quit playing. Beginner guitar players will hit a wall at some point; it is unavoidable. This problem is exacerbated by the absence of any teacher or expert to turn to in order to solve these problems. This is the reason why so many beginner players eventually stop playing the instrument. On the issue of frustration, beginner players may be unaware of just how much time and effort it requires in order to learn an instrument. Many players want to learn in order to play their favourite songs, and have little interest in the basics of guitar playing that, while quite dry and complex, will help them in this task. Therefore, when a player doesn't progress as quickly as they'd hoped, and can't play songs immediately, they quickly become disinterested and stop playing. Alongside this, it could be argued that "procrastination is often the real problem when it comes to self-instruction" (Mansfield, 2010). Self-motivation and perseverance are the key characteristics that will determine whether or not a person continues to play an instrument. Lessons provide a structured learning system and a clear method of progression and advancement in difficulty. Self-education does not have this; players mostly just move on to the next technique they want to learn. As they are not taking lessons, they have no motivation to persevere and keep playing. In order to teach yourself to play an instrument, tremendous levels of self-motivation, perseverance and discipline are required.

When one considers the problems associated with the traditional methods of music education, it is unsurprising that so many beginner guitar players inevitably become disinterested and quit playing.

2.4 The Importance of Technology and Multimedia Resources in Education

Technology is utilised throughout the modern world for virtually every aspect of modern life, including work and leisure. People use technology, in particular smartphones and applications, to assist them for the vast majority of daily tasks that they require, from shopping, to banking, to their various hobbies. In recent years, there has been a movement within the educational world to ingratiate technology into lessons. This idea is becoming more popular, and technology is becoming a more integral part of the education experience. This is of great relevance to this project, given that the aim is to teach users a new skill through technological means. The two prevailing concepts in this area are E-Learning (Electronic Learning) and TEL (Technology Enhanced Learning). E-Learning is defined as "learning facilitated and supported through the use of information and communications technology" (JISC Digital Media, 2016), and TEL is "the application of information and communication technologies to teaching and learning" (Kirkwood and Price, 2014). Educators are becoming keenly aware of the opportunities presented by the introduction of technology into the classroom, and software developers are also beginning to reap the rewards that this presents.

The use of technology in order to educate provides numerous benefits. Perhaps the most appealing factor for students is the flexibility that this medium provides. Technology-based learning, for example through smartphones and applications, allows 24/7 access to all learning resources, and they can study anytime, anywhere. This allows users to fit in their education around a busy schedule, and thus maximises their ability to learn. This flexibility "focuses on giving students choice in the pace, place and mode of their learning, and all three aspects can be assisted and promoted through appropriate pedagogical practice, practice that can itself be supported and enhanced" (Gordon, 2014, p.3). Flexibility would clearly be incredibly desirable in relation to an application that would teach beginners how to play guitar, as it would to allow users to fit their education around their schedules. Allowing users to learn at their own pace also allows for a significantly more thorough education. Students have the ability to progress as slowly as they desire, to ensure that they fully understand all previous information before moving on to more advanced material. All material will be also be easily accessible for the student, allowing them to refer back to the information provided should they so desire.

One of the core advantages in using technology in education revolves around the use of multimedia and interactive resources, which provide a much more engaging experience for students. Various studies have clearly demonstrated the remarkable effects that the use of multimedia resources can have on information retention and understanding among students. Two such studies were undertaken by Richard Lehrer, and Cynthia Okolo and Ralph Ferretti. In the Lehrer study, Dr. Lehrer studied a class of students who were learning about the American Civil War. The conclusions of this study were that the students who had learned through multimedia resources had made "long lasting connections with the materials", while students who had learned through the traditional methods demonstrated "little to no retention of the material" one year later (Lifetime Library, 2012). The Okolo and Ferretti study in 1998 concluded that using a wide variety of multimedia resources, ranging from text, audio, and video, "increased the likelihood that students will acquire an understanding of complex information". The students that use these resources were also highly motivated to use them again in future projects, and the resources "increased their variety of expression enhanced attitudes as well" (Lifetime Library, 2012).

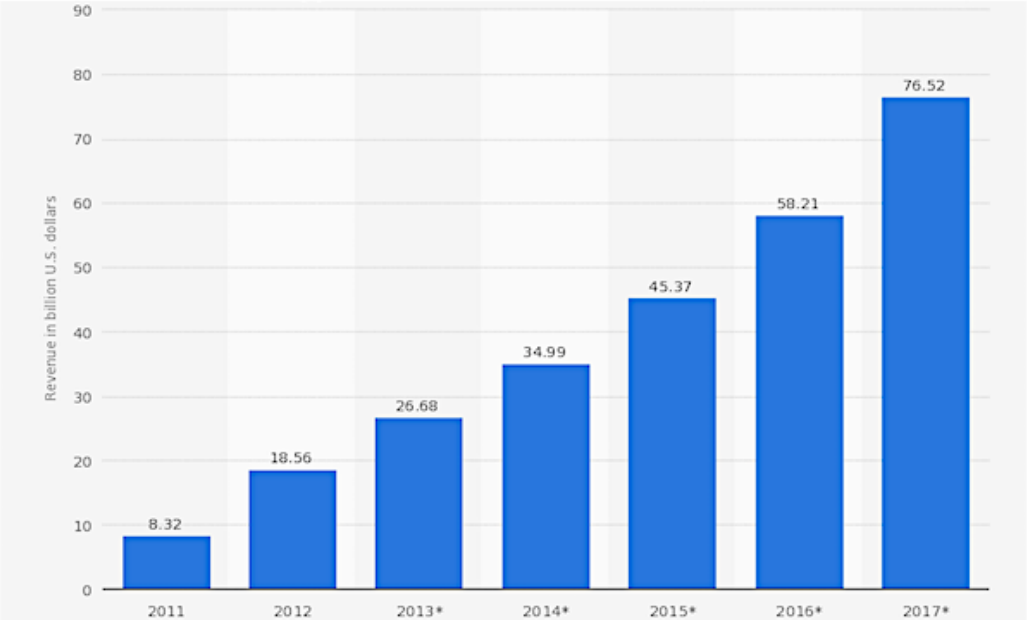
1 – Worldwide
app revenues
011 to 2017
US dollars)
:v, 2015)

When one analyses the hugely advantageous outcomes that can result from the use of technology and multimedia when attempting to assimilate new information, it is foreseeable that these techniques will become fully ubiquitous within any field of education. The analysis of these outcomes proved to be hugely influential in relation to the development of this project application. It was decided by the developer to implement multimedia into the application as much as possible, especially in relation to learning the key concepts required to play the guitar successfully. One variety of multimedia to be used frequently is video files, particularly regarding the "Lessons" section of the application. The video files will provide a visual demonstration of the various techniques that will be taught to the user, and in light of the research regarding multimedia files, this appears to be the most appropriate method to do so. Other forms of multimedia that will be utilised include image files, to provide diagrams and visual explanations of the various ideas being presented, and sound files, which will be used particularly in the "Tuner" section. In the final application, the effect of the background research into TEL and multimedia will be clearly seen.

2.5 Applications currently available on the market

In order to create a solution to the problem, an analysis must be made regarding the current applications that are available to beginner guitarists. Applications are omnipresent throughout the modern world, assisting us in so many of the tasks that we require on a daily basis. The figures surrounding this omnipresence are staggering – in 2015, there were 25 billion iOS app downloads, and 50 billion Android app downloads (Dogtiev, 2015), while the global app market is expected to be worth an estimated $77 billion by 2017 (Clifford, 2014). Figure 1 demonstrates the worldwide app revenues from 2011 to 2015, as well as the projected revenue values for 2016 and 2017.

Worldwide mobile app revenues from 2011 to 2017 (in billion U.S. dollars)



Given the astounding financial gains that could be achieved in the app market, it would naturally be assumed that the market was saturated, and that there are few emerging possibilities left. A search of the various app stores will reveal a large number of guitar-related apps that are available to a potential user and, in some cases, are extremely popular. This is unsurprising, given that, as has been discussed in the introduction, 10% of the UK population play the instrument. However, these applications are of very little use to a beginner, and are quite obviously aimed at much more experienced players. Prior to analysing the various apps available to a user, there were seven key categories that were considered to be particularly important to a beginner guitar player – the cost of the app, the use of video tutorials, chord diagrams and explanations, discussions surrounding the basics of the instrument, a list of songs or tabs to practice, a guitar tuner, and features that allow the user to interact with the app. After these seven categories are considered in relation to each app, a more detailed discussion will be undertaken regarding the specific strengths and weaknesses of each individual app.

The five most popular applications that will be analysed are Guitar+, Guitar Basics, Guitar Lessons Lite, Ultimate Guitar, and Yousician. Particular attention should be paid to Guitar+, which is the app with the highest downloads (over 10,000,000) and Ultimate Guitar, which is associated with the biggest tablature website in the world. A table summarising these findings can be found in Table 1.

- *Cost* – Ultimate Guitar was the only app that required payment to download, at a cost of £2.39. The other four apps were free to download. However, in order to access additional and often necessary content that would useful to a beginner, a payment was required. Only Guitar+ and Guitar Basics were completely free with no additional payments needed. Ultimate Guitar, Yousician, and Guitar Lessons Lite required payment in order to access, for example, explanations regarding key concepts, or additional sections within the app. In light of the research carried out for this project, the decision was made to ensure that the application would be free to download, and would require no payment for additional content.
- *Video Tutorials* – Video tutorials are hugely beneficial to a beginner faced with a variety of jargon terms and concepts that seem difficult to grasp when presented only in the written form, especially in light of the research into the benefits of TEL and multimedia resources. They are also useful when explaining basic procedures such as changing a string, which is a frequent activity for guitar players. However, only Yousician and Guitar Lessons Lite feature video tutorials. Guitar+, Ultimate Guitar, and Guitar Basics featured no video lessons at all. Guitar Basics did provide lessons, but were presented in the written form only, and were very verbose and would be hard to understand for a beginner player.
- *Chord Diagrams & Explanations* – For a guitar player to sharpen their skills and learn chords, it is important that diagrams and explanations of the various chords be included in the app. Given the importance of this basic information, it is surprising that only Guitar Basics provided this feature, while Guitar Lessons Lite and Ultimate Guitar required additional payment. Guitar+ and Yousician provided no such information.
- *Instrument Basics* – A guitar player must be aware of all of the parts of a guitar in order to get the best use and sound from their instrument. This information is imperative; a player cannot hope to progress if they do not know the basic features of the instrument. One would think that

this is the minimum level of information that a guitar-focused app should provide to its users. However, only Yousician, Guitar Lessons Lite and Guitar Basics provided this to it users, while Ultimate Guitar required additional payment for access. Guitar+ did not provide any guidance of any kind.

- *Songs or Tabs* – This would allow a beginner to put their new skills to the test, and fully explore concepts such as chord changes and tempo. Playing songs and tabs allow a user to sharpen their skills in a practical sense. All of the apps provided this facility, except for Yousician, which required additional payment for every song.
- *Tuner* – A tuner is crucial for a guitarist. It is arguably the most basic feature that is necessary, as without it playing will sound unpleasant and it may be difficult to gauge progress in terms of how a user's playing sounds. A player will simply not be able to play without access to a tuner. Surprisingly, only Yousician and Guitar Lessons Lite provided a tuner, with Ultimate Guitar requiring additional payment. Guitar Basics and Guitar+ did not provide any facility for a tuner.
- *Interactive Aspects* – An app that supports interaction with a user provides a significantly more enjoyable and educational experience, and can only improve the experience of learning an instrument. Of the five apps, three provided interactive aspects – Guitar+, Yousician, and Guitar Basics. Guitar Basics provided a feature allowing the user to pluck the strings on the screen, but special mention should be given to Guitar+, which provided a user with tablature and allowed them to play along using the strings on screen, and Yousician, which utilised the microphone in the device and allowed the user to play along to a tab on their actual guitar. Ultimate Guitar and Guitar Lessons Lite did not provide any interactive features.

Table 1 – Analysis of the 5 applications

	Video Tutorials	Free	Chords /Key /Scales	Guitar Basics	Ease Of Use	Songs To Learn	Login	Tuner	FAQ	Interact With App?	No. of Downloads
Guitar +	×	✓	×	×	✓	✓	×	×	×	✓	10,000,000+
Yousician	✓	**	×	✓	✓	£	✓	✓	×	✓	1,000,000+
Guitar Lessons Lite	✓	**	£	✓	×	✓	×	✓	×	×	1,000,000+
Guitar Basics	×	✓	✓	✓	✓	✓	×	×	✓	✓	100,000+
Ultimate Guitar	×	×	£	£	✓	✓	✓	£	×	×	1,000,000+

- ** While the basic app remains free, additional content within the app is not.
- £ = Additional content that requires payment

Alongside the features of the applications, consideration must be given to how the application will help its beginner users. One would think that it would be vitally important to appeal to beginner guitar players, as this would provide an even greater potential user base, and encourage more people to learn to play. Learning to play guitar is difficult, and the player will face many challenges in this process. Tom Fontana, a professional guitar teacher, discusses the "5 Biggest Challenges of Learning the Guitar" (Fontana, 2012). Any guitar application should surely at least attempt to confront these challenges. Table 2 clearly demonstrates that several of the applications address, or partially address, these issues, however many of them are ignored or omitted completely.

The five challenges will be described below, along with an analysis of the extent to which these challenges were confronted within each of the applications.

- *Changing between chords*: Learning chords is essential, but it can become quite difficult to change the finger positions to play a new chord. The flow between the chords should be smooth, and the strumming pattern and sound should not be affected by the change. This is a tricky skill, and advice and tips may be necessary to help beginners. As Table 2 demonstrates, only Yousician provides a lesson on changing chords. Ultimate Guitar and Guitar Lessons Lite required payment to access this lesson, while Guitar Basics and Guitar+ omitted this feature.
- *String skipping*: When attempting to pluck two or more non-adjacent strings, the player inadvertently plucks an incorrect string. This is a common problem, and can even regularly happen to experienced players. As with chord changes, only Yousician offers tutorials on the relevant technique, fingerpicking. Guitar Lessons Lite and Ultimate Guitar required payment to access this information, while Guitar Basics and Guitar+ did not provide any such guidance.
- *Barre Chords*: Barre chords can be very difficult, and rather painful, for a beginner to play. Playing a barre chord involves placing an entire finger, usually the index finger, over every string. Barre chords are indispensible, and learning to play them is vitally important. Yousician and Guitar Basics provided lessons on this, while Ultimate Guitar and Guitar Lessons Lite required payment for access. Guitar+ did not provide any information on barre chords.
- *Which Song to Practice*: Players usually just want to play the songs they love, and it can be disheartening when they can't. Beginners often do not know which songs they should attempt to learn first. All five applications have a song catalogue of some kind; however in relation to Yousician, additional payment was required to access the vast majority of songs, while Guitar Lessons Lite also required payment to access songs. Special recognition should be given to Ultimate Guitar, which allows access to 800,000 basic chords and tabs for no additional cost.
- *No Time for Practice*: Beginners often cannot find the time to practice, and may even begin to lose focus and interest. It is therefore crucial that the app contains lessons and tutorials. This provides the crucial flexibility that will allow the user to learn the skills and techniques at their own pace. Yousician and Guitar Basics provided these features, while Guitar Lessons Lite and Ultimate Guitar required payment. Guitar+ did not provide any lessons or tutorials, and thus did not provide the same level of flexibility provided by the other apps.

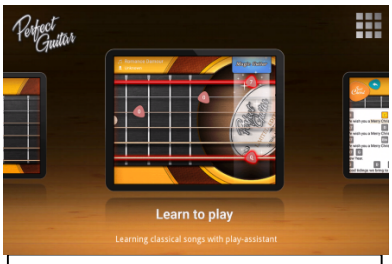
Table 2 – Analysis of the five applications regarding "The 5 Biggest Challenges"

	Changing Between Chords	String Skipping	Barre Chords	Which Songs to Practice	No Time for Practice
Guitar +	×	×	×	✓	×
Yousician	✓	✓	✓	£	✓
Guitar Lessons Lite	£	£	£	£	£
Guitar Basics	×	×	✓	✓	✓
Ultimate Guitar	£	£	£	✓	£

Figure 2 – Guitar+

2.5.1 Guitar+

Guitar+ offers a virtual guitar to its users, allowing them to simulate playing along to a song using the strings on the screen. In the most notable, and impressive, feature within the app, the six strings of the guitar will be displayed on the screen. When a note needs to be played, the appropriate string will be highlighted, informing the user as to which string they should touch. Alongside this, there is also a feature that allows a user to select a song, and within this song select the chords, and strum the strings to "play" the song. These features are very impressive, and tremendously appealing to users. The user interface is also very attractive, and the app is easy to use and navigate. The key problem with this application in this context is that it is clearly aimed at a more advanced player. Table 1 and Table 2 show the significant amount of basic information that is missing. Therefore, it is of very limited use to a beginner guitarist.



Guitar+ is designed for Android devices, and will run on both tablets and phones; however, the optimal experience will be achieved through the use of a tablet. The app was very responsive, and provided aural feedback informing a user that they have selected an option. The colour scheme used was both bright and vibrant, rendering the entire user interface aesthetically pleasing as well as being easy to use.

Pros – Very strong use of interactive features. Users can play along with the songs, and the app assists them in the reading of chords and tabs. The user interface is very attractive, and the app overall is very easy to use and navigate. Guitar+ is completely free, and requires no payment for access to additional features or resources.

Cons – No video tutorials, lessons, or basic explanations of concepts at all. The application was marketed for beginners, but is of little use to them. It was clearly designed for more advanced users.

2.5.2 Yousician

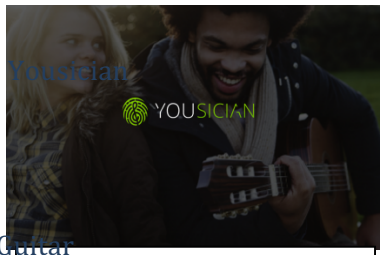


Figure 3 – Yousician

Yousician is one of the more useful apps for beginner players; however it is far from the complete educational experience. Detailed lessons and video tutorials are used in the application to great success, providing clear explanations and visual demonstrations. However, these lessons only cover a small amount of the basic knowledge that would be required by a beginner. The user interface has both an attractive and professional aesthetic, and is easy to use and navigate. There is also a facility allowing users to log in using their Facebook accounts, successfully implementing an element of personalisation to the app. The most impressive feature of the app utilises the microphone in the device, and will allow a user to play along to both tabs and songs using their own guitar. This is a hugely appealing and enjoyable feature, and may well be useful to beginner players when they have advanced to a stage where they can read tabs.

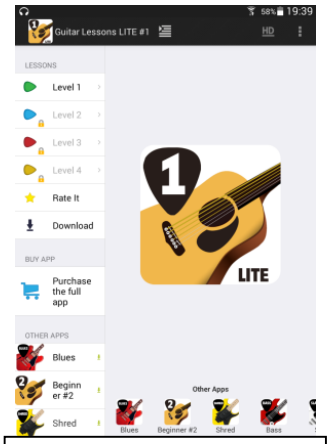
Figure 4 – Guitar Lessons Lite

Yousician is available on both iOS and Android phones and tablets. The app was both responsive and easy to use, and visually appealing. Some of the notable features include lessons, songs and tabs, and the ability to create an account and log in.

Pros – The layout is very appealing, and a personalisation feature has been implemented allowing the user to log in using Facebook. Lessons and video tutorials are included, but do not cover all of the necessary topics. The user can play along using their own guitar, utilising the device's microphone.

Cons – An important consideration for users should be that while the basic application is free, access to the majority of content requires payment, at a cost of £2.34 per item. This price is approaching extortionate, given the amount of additional content that requires purchase. For example, every song to be played must be purchased. This app could be useful for beginners, but it may be too advanced. For example, chord diagrams, or explanations about scales, keys, etc, are omitted. This information is essential to a beginner; therefore Yousician would be of limited use.

2.5.3 Guitar Lessons Lite



Guitar Lessons Lite seeks to provide step-by-step lessons specifically aimed at beginner guitar players. The app focuses on the basics, ranging from a description of the guitar parts, how to read chords and tabs, and how to hold a pick. Video tutorials are also implemented into the app to provide visual demonstrations of the concepts, and sound files are used to show the user how to play the various melodies. In theory, this application should be perfect for beginner guitar players. However, there are many notable flaws. There are four separate levels, each containing new lessons. Only the first level is free, with the other levels being available for £2.34. The lessons themselves are poor – in almost all of them, there is only a diagram provided, with no descriptions to provide any explanations.

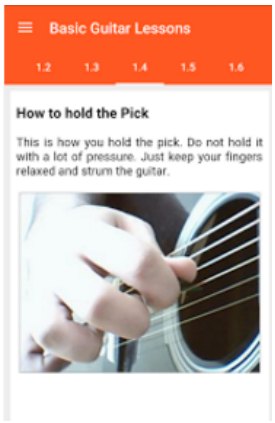
The application is available on Android phones and tablets. The app was not especially easy to use but was responsive; however the design of the user interface is very basic, and rather plain. There are relatively few notable features in comparison with the other apps, with the exception of the video tutorials and sound files for the melodies.

Pros – Guitar Lessons Lite provides explanations for the key elements of the instrument, as well as chords, tabs, and tuning. Video tutorials and sound files are also used, providing visual and aural demonstrations. A selection of melodies is also included, allowing the user to practice their skills.

igure 5 – Guitar Basics

Cons – While this app may be marketed towards beginner players, it provides little actual information, especially considering the fact that most of the lessons comprise a diagram with no explanation. There are no interactive features, or login feature, whatsoever. The layout is very basic, and is visually rather dull. Only the first level is free, with the remaining levels costing £2.34.

2.5.4 Guitar Basics



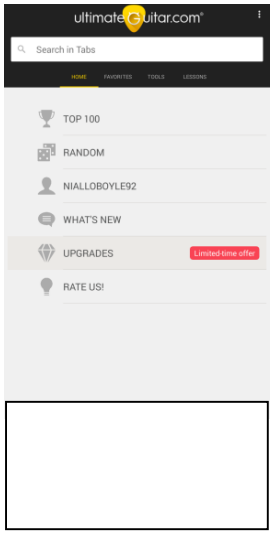
Of the five applications that were analysed, Guitar Basics is arguably the most useful to beginner players. There are a large number of useful features, many of which are critical to a beginner – explanations on the parts of the guitar, how to read tabs and chords, a tuner, finger exercises, chord diagrams, and songs to play. Guitar Basics provides detailed and clear explanations of the concepts being discussed. However, several problems must be noted. All of the lessons are presented in the form of blocks of text, with the occasional diagram. While this may provide detailed explanations, the lack of a visual demonstration could prove to be problematic. When learning a new technique on the guitar, it is often remarkably difficult to master that skill simply from reading a description of how to play it. A visual demonstration is necessary. The user interface is very easy to use and navigate, however the design of the UI is rather plain, and is not particularly aesthetically appealing. However, the lessons and guidance provided by the app are very impressive, and cover almost all of the basic skills that are required.

Guitar Basics is available for Android phones and tablets. The app is responsive, and easy to use, with a rather basic user interface design. Some of the features are impressive, notably a tuner, which allows a user to press a string and hear the sound of the string, allowing the user to tune their own guitar in order to match this sound.

Pros – Guitar Basics provides very thorough explanations of the instrument basics, as well as chords, keys, and scales, and the features provided are impressive. The application is also very easy to use and navigate.

Cons – The lessons are mainly comprised of blocks of text, with no visual demonstrations. This can be quite dry and difficult to follow and understand; this is far from the ideal manner in which to learn new information.

2.5.5 Ultimate Guitar



The Ultimate Guitar app is one of the most popular guitar apps on the market. From the author's personal experience, it is a tremendously useful app, providing access to a database of over 800,000 songs and tabs. Users can also create their own accounts, and sign in, allowing them to "like" certain songs that they wish to learn. However, this app is again far from ideal for use by beginners. The basic app, which costs £2.39, only provides access to the database of songs and the login feature. There are instructional lessons, such as instrument basics, a tuner, and a metronome; however all of these features can only be accessed if purchased at the cost of £14.99. This is a frankly extortionate price to pay for information that would be available for free elsewhere.

Ultimate Guitar is available on all Android and iOS devices. The app is very responsive to all requests, very easy to use and navigate, and has an attractive UI design. The database of 800,000 songs and tabs is impressive, and will no doubt come to form a part of the beginner's arsenal at a later point. However, at such an early stage, it is of little use to such users.

Pros – Ultimate Guitar is one of the most well-known guitar apps, with a reputation for providing a high quality of service. The huge database of songs and tabs could assist the user in practicing their skills, but it is virtually redundant if the user cannot read chords or tabs. The login feature allowing songs to be saved is also very appealing to potential users, adding an element of personalisation to the app.

Cons – Key features that are essential to a beginner are not included in the application, including a tuner, instrument basics, and reading chords and tabs. This information can only be accessed for the exorbitant price of £14.99.

2.6 Summary of Key Findings

Analysing the results of the research undertaken in the development of the application provided some very interesting conclusions. When investigating the reasons regarding why people play musical instruments, and how one would proceed in encouraging new users to learn to play, the numerous health and social benefits are integral. Playing an instrument can lead to a marked decrease in stress levels, depression, and anxiety, as well as an increase in brain function, and can even aid in the recovery from various cardiac conditions. For the health and social welfare of the population, actions should be taken to encourage as many individuals as possible to try and learn a musical instrument. In this technology-oriented society, a music tutor app, providing essential flexibility in allowing the user to learn at their desired pace, appears to be an effective way to do this.

When considered in light of the fact that 90% of beginner guitar players quit within their first year of learning to play, it is clear that the traditional methods of musical education are far from suitable for everyone. The cost of lessons from a professional teacher in the UK is prohibitive, and alienates the vast majority of the population. Even the few that can afford these lessons may find them to be ineffective, perhaps due to the regimented structure, which may remove all enjoyment from learning the

instrument, and the player may quickly lose interest in playing. Self-education is far from the ideal solution to these problems, given the detrimental impact that it can have on a player's technique. In the rush to begin playing their favourite songs, a beginner player may simply skip over the basics of the instrument that do not seem necessary at the time. However, these basics are vital to the player's development, and having to go back and learn these skills at a later point may slow down their progression. As there is no direct guidance provided by an expert, it is virtually inevitable that self-educated beginners will develop bad habits. It is clear that other, newer forms of education are required. In modern society, it is clear that a guitar tutor application is the perfect solution.

In order to ascertain the effectiveness of utilising technology in relation to musical education, one should look no further than the research that has been carried out in the arena of Technology Enhanced Learning (TEL). The implementation of technology, as well as a greater use of multimedia, is an ever-growing trend within schools, such is its efficacy as an educational tool. The studies carried out by Lehrer, and Okolo and Ferretti, confirm this idea – TEL and multimedia provide a significantly enhanced learning experience. The students are able to make a greater connection with the materials, and experience a greater understanding and retention of the information. Technology is clearly the way forward in relation to education, as it also provides a level of flexibility that is simply not possible with traditional educational methods. Students can have access to the resources at any time, and are able to learn at their own pace. This is very promising for the creation of a guitar application. The beginner guitar players will be able to learn at their own pace, and fit their education in around their busy schedules. Alongside this, the use of multimedia in the app, through, for example video tutorials, will allow the beginners to retain and understand the information to a much greater extent, thus improving their overall educational experience.

Perhaps the predominant factor that advocates the creation of a new guitar application is the sheer dearth of currently available applications that are useful to beginner players. Almost all of the applications on the market are clearly aimed at more advanced players, omitting the basic information and guidance that would be required by a beginner. The Guitar Basics app is one of the more useful applications for beginners, but despite this it still has many significant flaws. This application contains many lessons, and provides concise and detailed guidance; however the information is displayed in blocks of text. This may prove to be ineffective in educating the user – blocks of text can be very dry and difficult to understand without a visual representation. Ultimate Guitar and Guitar+, two of the most popular apps available, are again of very limited use. Guitar+ contains virtually no beginner-specific features, and Ultimate Guitar requires an exorbitant fee for access to this information. Guitar Lessons Lite also poses a similar problem. This application is specifically aimed at beginners, and contains four levels of lessons. However, only the first level is free, with the other levels requiring payment. Judging from the first level of lessons, this payment would be completely unjustified, as the lessons were of a very poor quality, comprising almost solely of diagrams, with no textual explanation whatsoever.

2.7 Recommendations for the Application

Based on the research carried out and presented in this chapter, the following recommendations shall be considered for the development of the application:

- A “Lessons” section shall be included, allowing the user to learn by fitting their education around their schedule, providing a flexible learning plan.
- A “FAQ” section is necessary, especially for self-educated beginners. With no direct guidance from an expert, a player who encounters a problem may have no-one to turn to for help. This section will provide assistance surrounding any issues the player may face.
- Based on the importance of TEL and multimedia in education, video tutorials and sound files will be used frequently throughout the app, to ensure that the user will be provided with the best education possible.
- Information essential to a beginner player, such as tuning, chords, scales, etc. must be included.
- Song sheets are required to allow a user to practice their newly learned skills in a practical way.
- The users must be allowed to log in to the app and “favourite” certain songs that they like for future practice.
- The application must be completely free, with no additional content to be purchased.
- The user interface must be aesthetically appealing, and the application itself must be both easy to use, and navigable.

Alongside these recommendations, there were certain ideas that were investigated but were ultimately discarded. An early idea formulated by the developer involved the design of the tuner. In its original conception, the tuner would utilise the device microphone in order to compare the sound of the string being played to the sound the string should make, and then inform the user whether they needed to tighten or loosen the string. However, after extensive research, this idea was abandoned, as it would require a significant amount of both complex and time-consuming work, as the signals, frequencies and pitches of each string would have to be ascertained, and a system in which these frequencies could be compared must be devised. To undertake all of this work would have been wholly unnecessary, especially given the fact that the tuner design decided upon, where the user presses a button representing a string and the sound of that string plays, is an excellent alternative, and a very useful and user-friendly way to tune a guitar.

As well as this, the developer initially intended to develop a forum, in which all users of the app could discuss any problems they may be having, and provide helpful tips and advice to all other users. However, this idea was discarded in favour of other, more important features for the app that should be investigated. Perhaps this idea may be resurrected for future development of the application, which shall be discussed in Chapter 7.

2.8 Summary

This chapter has analysed the background research undertaken regarding the various challenges posed to a beginner guitar player by the traditional methods of music education. Alongside this, an investigation into the strengths and weaknesses associated with the most popular guitar-focused applications available on the market highlighted the lack of assistance that these apps provide to a beginner player. This background research, alongside the research conducted into the importance and effectiveness of Technology Enhanced Learning (TEL) in education, has allowed the developer to design and create an application that would assist a beginner guitar player in learning to play the instrument.

Chapter 3: Requirements Analysis

3.1 Introduction

This chapter specifies the end user requirements of the application, some of which were obtained through the use of a questionnaire that was distributed to a select group of individuals with a range of musical experience. An analysis of the responses from the questionnaires will be undertaken, as well as an interpretation regarding how the suggestions should be implemented into the application. This chapter will conclude with a discussion of both the functional and non-functional requirements that must be incorporated into the app.

3.2 Requirements

Prior to development, the original proposition was to develop an Android application that could be used on both smartphones and tablets. However, based on the research undertaken regarding currently available applications, there are several problems inherent with the use of smartphones for such an application, mainly revolving around the significantly smaller screen size. The majority of the features that are to be included in the application require a larger screen than that provided by a smartphone in order to provide the optimal experience when using the app. For example, the song sheets that will be included are quite long, and this would cause the user to have to scroll down regularly. This is less of a concern when the application is used on a tablet. Therefore, in order to ensure that the user has the best possible experience when using the app, it is recommended that a tablet with a screen size of 7 inches is used. Developing the application as an Android app, and specifically using the Android Studio IDE, allows for the SQLite database library to be adopted into the application, permitting the user to create an account, log in, and "favourite" certain songs.

3.2.1 Questionnaire Responses

The questionnaire itself consisted of nine questions, and featured questions regarding the method through which the respondent learned, or would learn, to play the instrument, what features they feel would be appropriate to include in a guitar tutor app, and the extent to which they thought an application would assist a beginner guitar player. Of the ten respondents, one had never played the instrument but was keen to learn, three had been playing for five to seven years, and six had been playing for over seven years. Two of the respondents were guitar teachers, while one was a Professor within the Music field. It was decided at an early stage that the opinions of experienced players were slightly more important than those of their rather more inexperienced counterparts. While it was undoubtedly important to seek the opinions of those with less experience, experienced players would have the ability to reflect on their own education; therefore, they would have a much clearer idea of the key aspects and concepts that would be crucial to a beginner player.

The purpose behind the distribution of these questionnaires was mainly twofold – to determine to what extent experienced players believe a guitar tutor app would be useful to beginner players, and to specify what features, and key concepts, must be included and addressed within such an application. Of the fifteen questionnaires that were distributed, ten respondents provided feedback. As shall be seen, the

Figure 7 – A graph of the most popular learning resources.

feedback provided in these responses was pivotal in the development of the application, and shaped not only the overall design of the application, but also the features that were eventually included within it.

3.2.2 Summary of the Responses

As has been stated above, the purpose of the questionnaire was to determine the features that should be included in the application, and the key concepts that must be addressed. The results of the some of the questions shall be summarised below:

- *Q2 – Methods of Teaching*: Given that the purpose of the app is to assist in the education of beginner players, the methods through which the respondents were educated must be ascertained. Given the prominence of the two traditional methods of education, it is unsurprising that the results were almost exactly equal – seven respondents had had lessons from a teacher, with eight respondents being self-taught. It is important to note that several respondents had ticked both boxes, as they have had experience of both methods. This is a significant result, as it was important to have both traditional methods represented within the feedback, thus providing differing opinions regarding the two approaches.
- *Q4 – Educational Resources:* This question sought to ascertain the key resources used by the respondents to learn a new technique. Eight respondents each listed YouTube and websites as a key resource, seven respondents listed learning by ear, five listed songbooks, and three listed lessons and applications (Figure 7). These results are hugely significant. The popularity of websites and YouTube perhaps indicates a desire for technology-based learning, and for interactive and audio-visual resources. Only three respondents listed applications as a key resource, however this is unsurprising, given the narrow scope of the currently available apps.

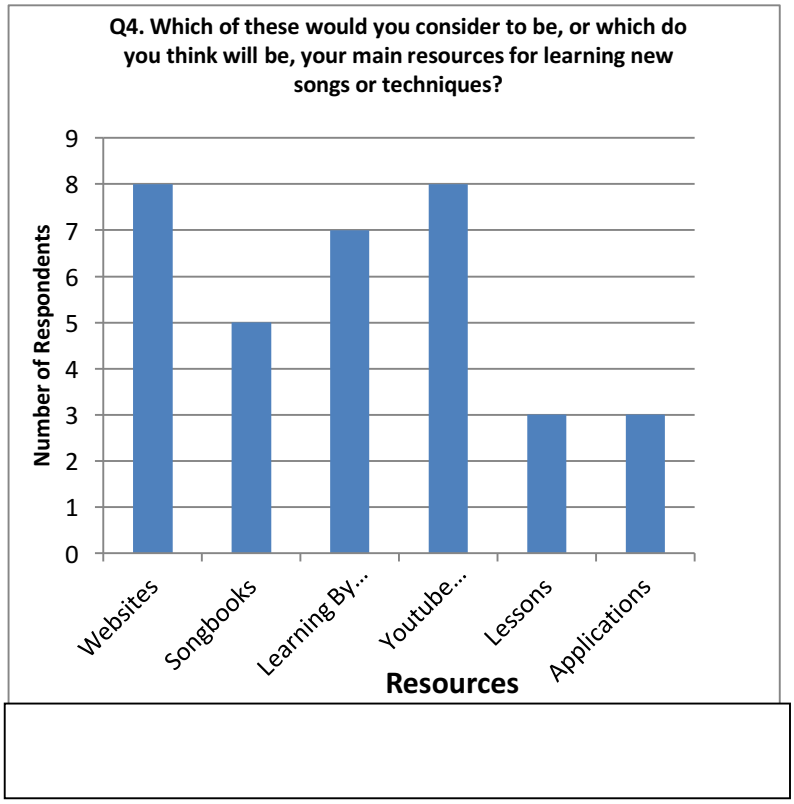
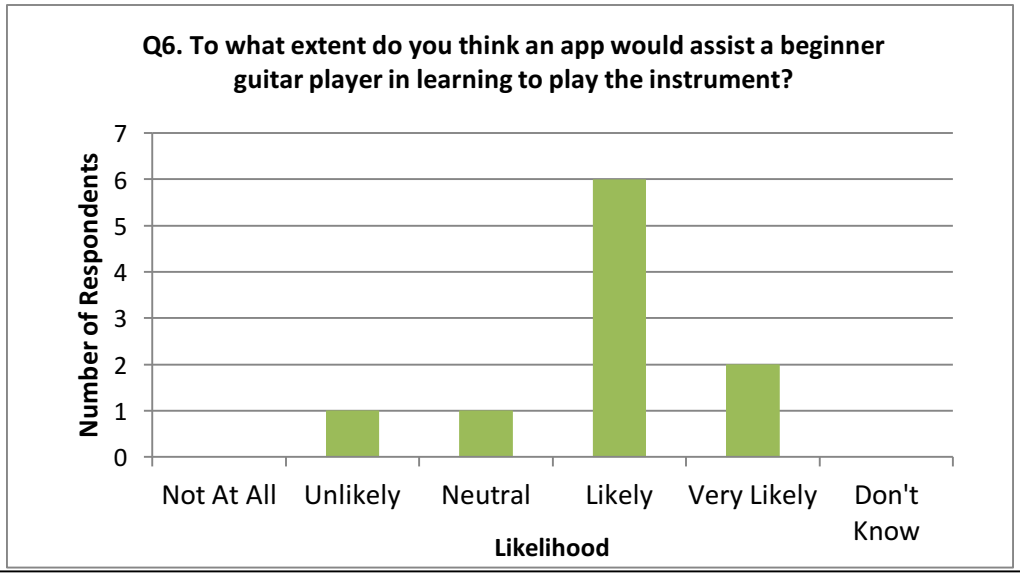


Figure 8 – Graph representing the opinions of the respondents regarding the usefulness of a guitar tutor app.

- *Q5 – Payment for an app*: Respondents were asked how likely they would be to pay for an app, or content, that would improve their ability. There appeared to be no consensus on this issue, as the results were fairly mixed. Three respondents said they were likely to pay, two respondents each answered neutral or not at all, and one respondent each selected don't know, unlikely, or very likely. It is clear, though, that there are a significant proportion of respondents who would not be prepared to pay for an app or content.
- *Q6 – Usefulness of an app:* The respondents were asked for their opinion on the extent to which an application would assist a beginner guitar player in learning to play the instrument. The results can be seen in Figure 8. Two of the respondents selected very likely, six selected likely, and one user each selected neutral and unlikely. It is clear that the general consensus is overwhelmingly in favour of the idea that it would be helpful to a beginner player, with 80% of the respondents believing it would be at least likely to be of assistance.



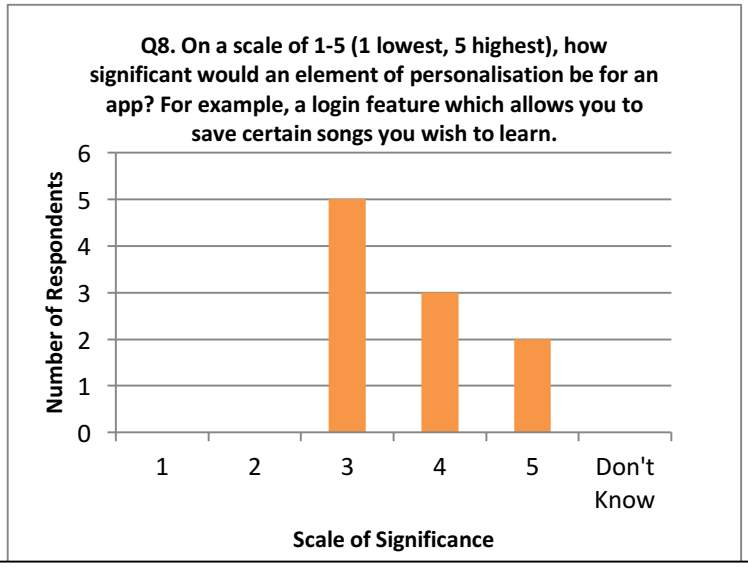
- *Q7 – Important Features:* The users were provided with a list of features that could be included, and were asked to judge their importance to a beginner player. The results can be seen in Table 3, with the red numbers indicating the number of respondents who selected each option. This question was pivotal in deciding what features should be included in the application.

Table 3 – Analysis of responses from Question 7

	Not at all	*Not Really Important*	*Neutral*	*Quite Important*	*Very Important*	*Don't Know*
Advice on guitar models, strings, cases etc.		4	3	2		
Chords (diagrams, types, variations)				1	9	
Guitar tuner			1	3	6	
Database of beginner songs & riffs	1			6	3	
Scales diagrams & tutorial			2	3	4	
Login feature allowing songs to be saved		2	3	4	1	
Video tutorials – changing strings, song playing, instrument basics etc.	1			2	7	
Facility for user to record themselves strumming a chord and see if it matches chord	2	2	1	4	1	
Buttons which when pressed play the sound of a chord/string	1	1		5	3	
FAQ section for beginners		2	2	3	3	
Explanation of keys, and a facility to change the key within a song transcript			2	5	3	
Fingerpicking demonstration	1	1	1	2	5	
Finger technique exercises		1	1	5	3	
Strumming pattern exercises	1			3	6	
Metronome		1	2	3	3	1
Video demonstration of chords that sound good together (eg 4 chord song – G, D, Em, C) & video tutorial on changing between chords	1		1	7	1	

Figure 9 – A graph representing the results of Q8, regarding the significance of a login feature.

- *Q8 – Importance of Login Feature:* Several of the currently available applications, including Yousician and Ultimate Guitar, made effective use of a login feature, allowing a user to create an account. Question 8 asked the respondents how important such personalisation would be within an app, particularly in relation to "favouriting" certain songs a user wishes to play. The responses were gauged on a scale of 1 to 5. Two respondents selected five, three respondents selected 4, and the remaining five respondents chose 3. This clearly highlights the desire for this feature, especially considering that no respondent selected below 3/5 in terms of importance.



3.2.3 User Stories

Based on the responses from the questionnaires regarding the various requirements and features that should be included in the app, a series of user stories have been created, and can be seen in Table 4.

User Story	*Criteria To Meet Requirement*
As a user, I want guidance surrounding key concepts, and instructions as to how to proceed through the app.	• A guitar plectrum button will, when pressed, provide explanations and guidance surrounding key concepts and definitions, and will instruct the user as to how to proceed through the app.
As a user, I want the ability to test and practice my skills by playing songs.	• A selection of song sheets will be available, showing the user how to play the songs.
As a user, I want to gauge my progress and select songs to play based on my skill level.	• The Songs & Tabs and Lessons sections will be divided into three categories based on ability – Beginner, Intermediate and Advanced. • A lesson tracking system will also be implemented into the app, allowing

Table 4 – User stories

	users to monitor their progress.
As a user, I want the app to be personalised to me.	• The user will be enabled to either create an account, or login. • This allows a user to "favourite" songs to be displayed on their Account page.
As a user, I want to select lessons and songs based on my requirements and needs.	• Lessons will be separated into Beginner, Intermediate and Advanced categories. • Users select lessons which will provide them with the information they require.
As a user, I want to save certain songs in one place, so I don't have to look through the different difficulty levels to find the songs I like.	• When the user logs in, they will have the opportunity to favourite certain songs from the three difficulty levels, with these songs being stored within their Account section.
As a user, I want to be assured that all personal information I provide is secure.	• All personal information will be stored on a private database, with only the administrator having access to the data.
As a user, I want easy navigation, and for it to be straightforward to access all the app's features.	• Navigation through the app will be clearly signposted. • The layout and general design of the app will remain consistent throughout. • Home and back buttons will be included on virtually every screen. • A drop-down menu will also be utilised, allowing easy access to the other features within the app.
As a user, I want an application that is reliable.	• The information in the app is correct. • The app does not continually crash, and is fully functional and responsive.
As a user, I want the best possible educational experience, using a variety of methods.	• A wide variety of multimedia and educational resources will be provided. • Video tutorials will be used throughout, including for Lessons and Chords sections. • Sound files will be used for the tuners. • Image files will be used to provide visual representations of concepts. • Diagrams for chords and scales will be displayed.
As a user, I want an application that is completely free of charge, with no additional content to be purchased.	• The app is completely free to download, and there will no additional payments required.

3.3 Functional and Non-Functional Requirements

The feedback provided by the respondents through the questionnaires, alongside the research into the currently available applications, was crucial in the development of both the functional and non-functional requirements of the application.

3.3.1 Functional Requirements

Functional requirements traditionally "relate to specific functions" (Aybuke and Wohlin, 2006, p. 84) of an application. The questionnaire responses were, again, vital when determining the functional requirements of the application. The following functional requirements were established:

- *The application must provide an assortment of educational tools, ranging from video tutorials to sound files, and other multimedia.* – It is clear from the opinions of the respondents that a wide variety of methods have been used in their education, ranging from YouTube to songbooks. It is therefore important to include a range of educational tools, especially video and sound files, given the benefits provided by TEL and multimedia use.
- *The application must provide the user with instructions and guidance when requested.* – When the user presses the guitar plectrum button, an alert dialog will be displayed providing explanations of the options available to the user, and how they can progress through the app. Also, error messages will inform the user of the nature of their error, and how it can be avoided.
- *The application must provide songs and tabs for the user to play.* – A variety of song sheets and tabs will be provided to the user, showing them how to play songs and allow them to test their newly-acquired skills in a practical way.
- *The application must allow the user to login and create an account.* – A database will be used to store the details of all individuals who have created an account, allowing them to log in. Creating an account is optional, allowing anonymous users to still use the app freely.
- *The application must allow the user to "favourite" certain songs*. – If a user finds a song that they like, or wish to practice more, they can "favourite" it. All of the user's favourited songs can be accessed through the Account section. However, only a logged in user can favourite songs. If a user has not logged in and tries to favourite a song, they will be told that they must log in.
- *The application must store all personal information securely*. – Once the information has been added to the database when a new account is created, only the administrator (the developer) can access the database to view the database tables, and to search for and delete users.
- *The application must be easy to use, and provide easy navigation throughout the app.* – The progression through the app will be straightforward, with all options available to the user clearly signposted. Back or home buttons will be displayed on every page, as will a dropdown menu that allows easy access to every section of the app.

3.3.2 Non-Functional Requirements

Non-functional requirements, on the other hand, refer to "properties that the functions or system must have" (Aybuke and Wohlin, 2006, p. 84). They refer to the characteristics or qualities that a system should possess. The non-functional requirements that must be incorporated into the application are as follows:

- *Usability* – The application must be user-friendly for all people, regardless of age, ability, or technological literacy, and must encourage the users to continue to use the app. This includes, for example, an aesthetically appealing user interface, featuring clear and legible fonts.
- *Reusability* – The application must be able to be reused by the user when required. Code should also attempt to be reused where possible, ensuring consistency throughout the application, allowing the user to quickly become familiar with the operation of the app.
- *Ethical* – The application must ensure that all personal information belonging to users is secure.
- *Portability* – The application must be capable of running on a variety of Android devices, with no negative impact on the operation or performance.
- *Reliability* – The application must be reliable, and operate smoothly without the risk of crashing.
- *Robustness* – The application must be capable of coping with any errors, and must deal with them effectively. For example, if the user attempts to favourite a song when they are not logged in, an error message will appear informing them to log in.
- *Efficiency* – The application must be responsive, and respond to all queries or selections quickly. Waiting times must be minimised, especially when obtaining data from the database.
- *Modifiable* – The software must be modifiable, in case changes are required to elements of the application.

3.4 Impact of the Questionnaire Feedback on the App Design

The responses to the various questions provided on the questionnaire were critical to the design of the application and, in particular, to the features that were to be included within it. An analysis will now be undertaken regarding the aspects of the app that were directly affected, and in what manner.

- *Price of the application* - The responses to Question 5, regarding the likelihood of the respondent paying for an app or content that would improve their ability, were fairly mixed. Three of the respondents selected likely, two each selected neutral or not at all, while one respondent each chose don't know, very likely, or unlikely. However, the majority of the respondents would be unwilling to pay. 60% selected either neutral, unlikely, not at all, or don't know. This unwillingness informed the decision by the developer to ensure that the application would be free to download, and would require no additional content to be purchased.
- *Login feature* - In relation to Question 8, which sought to deduce the importance of a login feature that would allow songs to be saved, the responses were very much in favour of implementing this feature. On a scale of 1 to 5, with 1 being lowest and 5 being highest, five users selected 3, three users selected 4, and two users selected 5. There were no responses lower than 3/5. This clearly indicated a strong feeling in favour of such a feature. It was therefore decided to introduce a database to the application, using the Android SQLite database library. This would allow the user to create an account, log in, and "favourite" songs with which they have a particular affinity. The user's details, the details of each song, and the details of all favourited songs will then be stored in three separate database tables. All of the user's favourited songs can then be accessed through the Account section of the application.
- *Chords* - Question 7 was arguably the one which had the greatest overall effect on the application. This question asked the respondents to rate the importance of various features that

could be included in the app, and how useful they would be to a beginner player. The responses for each feature were as follows – not at all, not really important, neutral, quite important, very important, and don't know. These responses, which can be seen in Table 3, were of great assistance to the developer in relation to the features that should be implemented into the application. One of the features mentioned in Question 7 was a Chords section, specifically regarding diagrams, the different types, and the variations. One respondent selected quite important, while the remaining nine selected very important. Chords are an essential part of guitar playing, and therefore it is imperative that they are included in the app. Therefore, a specific Chords section shall be included in the app, which provides diagrams on how to play each chord, as well as videos showing the user how each chord should sound.

- *Tuner* - A guitar tuner was rated as indispensible, with six respondents selecting very important, three selecting quite important, and one choosing neutral. As a result, a tuner will be designed that will allow a user to press a button representing each string, and when pressed, the sound of that string will play, allowing the user to tune their string until the sounds match. This feature also accommodates another desirable feature, according to Question 7. The importance of buttons that, when pressed, will play the sound of a string was ranked as very important by three respondents, quite important by five, not really important by one, and not at all by one.
- *Songs & Tabs* - The inclusion of songs and tabs that would allow a user to test their ability in a practical environment again was ranked as vital by the respondents. Three respondents listed this feature as very important, six believed it to be quite important, while one selected not at all. Due to the obvious demand for songs and tabs to be included, a Songs & Tabs section shall be implemented into the application. This will show the user how to play each song, allowing them to practice skills that are best learned in this manner; for example, tempo and chords changes. Users will also be able to favourite certain songs, for the purposes of future practice.
- *Scales* - Scales can demonstrate the manner in which chords are formed, allow players to create their own melodies, and are also very useful exercises to improve finger movement and placement. Four respondents believed scales to be very important, three selected quite important, and two remained neutral. A specific Scales section will therefore be designed, providing the various types of scale for each key, as well as variations of the specific scale chosen. Alongside this, a dedicated scales lesson will be included in the Lessons section.
- *Video Tutorials* - Given the much-lauded impact that Technology Enhanced Learning (TEL) and the use of multimedia resources can have on education, it is little wonder that the use of video tutorials was considered to be hugely significant. Seven respondents considered this to be very important, two selected quite important, while one respondent was of the opinion that it was not at all important. Attempting to teach someone a new technique or skill for the guitar without the use of a visual demonstration is a fool's errand; the information would be too difficult to comprehend on its own, without the aid of directly observing the skill being practiced. It was therefore decided to use video tutorials throughout the application, but particularly in relation to the Lessons and Chords sections. Sound files will also be used within the Tuner and Songs & Tabs sections, along with image files utilised in virtually every section.

3.5 Summary

This chapter has undertaken an analysis of the results obtained from the questionnaires that were distributed to various individuals. These responses were examined, before being summarised in various user stories, while this information was also used to establish the various functional and non-functional requirements. The chapter concluded with a discussion of the specific ways in which the questionnaire responses, and functional and non-functional requirements, impacted upon the design of the application, and the features that would be implemented within it.

Chapter 4: Design

This chapter provides an analysis of the design choices that were made for the application. This will also feature a detailed overview regarding the selected platform, as well as the design and structure of the database, the user interface, the various Java classes that were used, and the overall application itself.

4.1 Creation of a Successful Application

Applications are used by millions of people throughout the world, with more than one billion Android users, and over 50 billion app downloads in 2015 alone (Dredge, 2015). These applications are available on a wide range of devices, with major differences in screen size, hardware capabilities, and processing power. Naturally, this has a significant impact on the development of an application; particularly regarding the layout of the user interface, the features to be included, and the overall size of the application. Therefore, it was decided that in order for the application to be utilised to its full potential, while providing the most positive user experience possible, the app should only be used on Android tablets with a screen size of at least 7 inches. This will allow the features to be used both efficiently and effectively, providing the maximum user, and learning, experience.

In order to create a successful application, there are certain key considerations that must be taken into account. The American software company VenturePact compiled a list of the eight best practices that must be observed prior to the development of any application (VenturePact, 2015). An analysis of these considerations will be made below, along with the methods through which they were implemented into the application.

1. **Know Your Users** – The application is being designed to be used by the widest possible audience, and thus must be very user-friendly to people of all ages and abilities. Therefore, the design of the user interface and features was undertaken with usability as the key concept. All buttons would clearly indicate their function, either through use of text or symbols, while progression through the app would be straightforward and easy to comprehend. Alongside this, the use of a tablet also allows for greater text size, increasing the text legibility.
2. **Make the App Comprehensible** – As mentioned above, the buttons will provide clear explanations of their functions, through both text and symbols. Progression is also straightforward, and drop-down menus, home and back buttons on virtually every screen allow easy navigation. Images are also used to provide demonstrations and further explanations for certain concepts, and are used in tandem with text to provide concise information.
3. **Determine the right design methodology** – In order to design the application in the most efficient manner, it was decided at an early stage to adopt the Agile model of software development. However, before any development occurs, research and planning was required. User stories were developed from the questionnaire feedback, and storyboards were created, as shall be seen later in this chapter. Agile focuses on incremental development, and continuous testing and reassessment of future progression after the end of each increment. Certain aspects of the application can also be prioritised over others. This is essential, naturally, as certain important or time-consuming features can be afforded a greater period of time. For example, successfully implementing the database into the application was prioritised over several other

less significant features, ensuring that core functionality was given precedence. Regular testing throughout development also allows for any bugs or issues to be resolved at an early stage.

4. **Focus on Developing the Core** – It was vitally important to ensure that all functionality was in place and fully operational before any purely aesthetical development occurred. This ensures that the functionality is prioritised, and can be given the greatest amount of development time required. Users simply will not use an application that is not fully functional. The development of a prototype of the application ensured that core functionality was prioritised. After all of the core requirements had been fulfilled, other aspects of the application could be dealt with.
5. **Security should be your top priority** – Security is, naturally, critical to the success of any application, especially considering the fact that user details are stored in the database of the application. Therefore, it was decided that no user would have access to any of the tables or data stored in the database. Only the administrator account can view these tables, with the username and password of this account known only to the developer.
6. **Testing is the key** – In order for an application to be successful, regular testing is essential to ensure that any bugs or issues are remedied at any early stage, as well as checking that the application functions as intended. The adoption of the Agile model ensures that testing happens on a regular basis. Alongside this, when a new feature was implemented into the application, it was immediately tested to ensure that it was fully operational with no bugs.
7. & 8. **Incorporate application analytics, and have a feedback mechanism** – User feedback is indispensible to software development, in particular to applications that will be used by the general public. Prior to the application's release, it will be given to a select group of individuals for testing and evaluation. These individuals will have a wide range of both technological and musical ability. The feedback garnered from this activity will be deeply informative, especially in relation to future development and potential areas of improvement.

Human-Computer Interaction (HCI) is a concept that is ever-increasing in terms of its importance to developers and users alike, especially considering the ever-expanding application market. Users have the right to be highly selective when using applications; if they find an application that is poorly designed or is difficult to use, they will simply go elsewhere and download a similar app. There are very few gaps in the already-saturated application market; therefore it is crucial that developers ensure that users are satisfied. HCI is thus imperative to the success of an application.

In order to ensure that an application with excellent HCI is developed, Jakob Nielsen created what were titled the "10 Heuristics" (Stanton et al, 2013, p.438). Several of these considerations were implemented into the final application. For example, consistency throughout the application is important. The use of consistent features, such as the use of the same image for all home buttons, allows the user to become more familiar with the app, and to use it more effectively, thus becoming more comfortable when doing so. As well as this, all messages and text displayed to the user must be clear, concise, and easy to understand. For example, every button in the application contains either a textual description or a symbol that is easily recognisable that indicates that button's function. Error messages also provide clear descriptions of the problems that have occurred. For example, if a user who has not signed in attempts to favourite a song, they will be faced with the following message – "Oh dear! You must be logged in to

favourite a song". This message clearly informs the user of the error, and provides guidance as to the way in which this error can be amended. Jakob's 10 Heuristics were of particular assistance regarding the HCI characteristics of the application.

4.2 Proposed Application

Modern society is evolving at an ever-increasing rate. This is particularly true in relation to technology, where both the capabilities of the products themselves, and their implementation into all facets of our daily lives, have advanced at an astonishing rate. Nowadays, people have access to a virtually endless supply of information at their fingertips. Mobile devices, particularly smartphones and tablets, have now overtaken desktop PCs in terms of internet usage, with this trend set to continue. The application will be designed for a tablet, and with good reason. More than 1 billion people worldwide used a tablet in 2015, with this figure set to rise to as many as 1.43 billion in 2018 (eMarketer, 2015). In the UK, 54% of households own a tablet, and of the 46% who do not own a tablet, 21% said they were likely to purchase one within the next 12 months (Ofcom, 2015). This pervasiveness of tablets in society is only increased by the continually falling prices, which have "been in a freefall for some time" (Hughes, 2014).

Designing the application for use on a tablet is undeniably the correct choice. The large screen allows for the application features to be used to their maximum efficiency and effectiveness, while the widespread use of tablets, and their relatively low price, indicates that there is an established, and potentially expanding, audience. The touch screen allows for the features and sections within the app to be easily accessed, while the in-built standard keyboard allows for the user to input their details to create accounts. This permits the user to make full use of the features offered by an account – including favouriting songs, tracking lesson progress, and accessing the quizzes.

The application was designed to include a wide variety of features and functions that would ensure an effective learning experience, as well as an excellent user experience. These features are as follows:

- Provide the user with detailed lessons and tutorials, providing concise and easy-to-understand information and guidance.
- Allow the user to test their knowledge using a quiz, which can only be accessed once all lessons in a section have been completed.
- Provide the user with diagrams and instructional videos, showing them how to play certain chords and scales.
- Allow the user to practice their skills by providing them with songs and tabs.
- Provide the user with useful information and tips in the FAQ section.
- Allow the user to create an account to save their lesson progress and favourite songs.

4.3 User Interface Design

Regardless of the functionality, or the success with which an app may operate, if an application has a poorly designed or confusing user interface, users will be very reluctant to continue to use the app. As has been mentioned previously, users have the right to be highly selective when using applications – with so many apps available offering similar services, users can simply go elsewhere should their demands not be satisfied. Therefore, it is imperative that an application has an aesthetically-pleasing,

simple, and user-friendly user interface (UI). The design of the UI has become ever-more crucial due to the rise of touch screen technology – appropriate spacing between buttons, for example, is now a major concern, to ensure that users do not inadvertently select more than one option.

Due to the importance of the UI design, one must give careful consideration to the best practices that should be followed. The "8 Golden Rules of Interface Design" (Baber and Noyes, 2003, p.3) suggested by Ben Shneiderman have been widely regarded for decades as the leading authority on the topic of interface design. It is important to note that a few of these rules mirror several of the heuristics proposed by Jakob Nielsen (Stanton et al, 2013, p.438). These guidelines, and their similarities to the heuristics, will be analysed below.

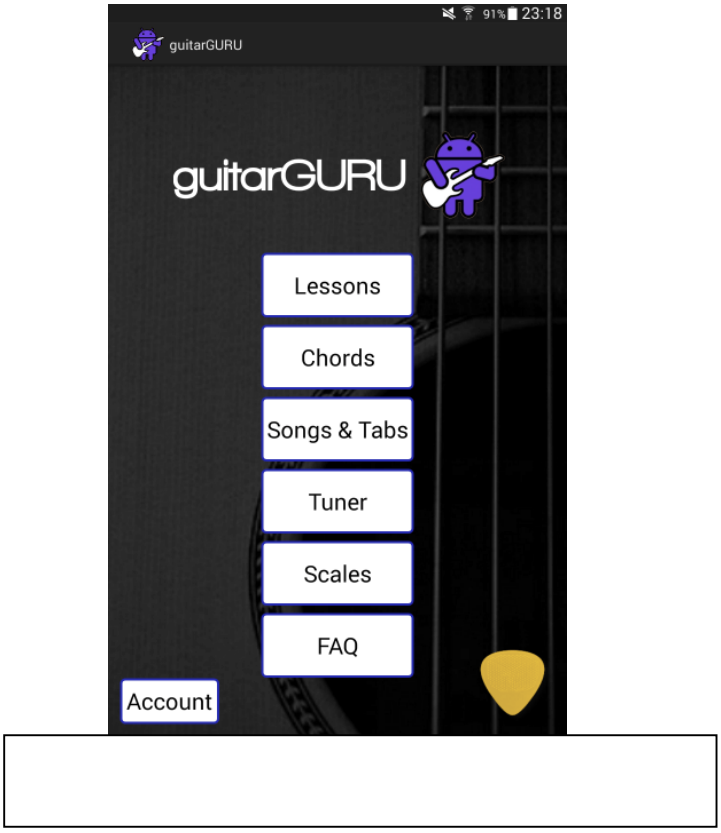
1. **Strive for consistency** – This closely resembles the "consistency and standards" heuristic, in that conventions must be followed throughout the app, allowing the user to become more familiar with the layout and the features of the app itself. As shall be demonstrated, all page layouts, colour schemes, and fonts will remain very similar, allowing the user to become familiarised with the app through use.
2. **Enable frequent users to use shortcuts** – As the user becomes more familiar with the app, the most efficient manner in which to progress through the app will become apparent, whether through the various links provided on the screens, or the drop-down menus on every screen.
3. **Offer informative feedback** – For example, when the user favourites a song, they will either be presented with a "Success" message, or an error message, informing them of any problem.
4. **Design dialog to yield closure** – The sections can clearly be differentiated into beginning, middle, and end, and progression through the app is straightforward and understandable.
5. **Offer simple error handling** – Nielsen was also a proponent of this concept, with his "help users recognise, diagnose and recover from errors" heuristic. All error messages must be expressed in plain language, and provide a concise solution to any error. Therefore, guitarGURU will be designed in such a way as to ensure that any errors that occur are dealt with, and messages that are displayed will detail the exact nature of the error, along with potential remedies.
6. **Permit easy reversal of actions** – All errors are easily reversible. For example, if incorrect details are entered to create an account, an "Edit Details" section is available. Back buttons are also present on virtually every screen, allowing the user to return to the previous screen.
7. **Support internal locus of control** – The "user control and freedom" heuristic emphasises that the user should be in full control, and should they make a mistake, for example selecting the wrong option, the user should have the power to amend this error. The application is designed to respond to the actions of the user alone. Progression is impossible without user actions, ensuring that the user is in control. As well as this, back buttons are placed on virtually every screen, allowing the user to return to the previous screen and make a new selection.
8. **Reduce short-term memory load** – "Recognition rather than recall" is one of the most important heuristics proposed by Nielsen, and establishes that users should not have to remember large amounts of information as they progress through the app. This is not a major concern for guitarGURU. For example, once a user is logged in, they remain so until they log out, therefore they don't have to regularly enter their username and password.

Figure 10 – Example buttons used in the application

As can clearly be seen, Shneiderman's golden rules and Nielsen's heuristics were imperative to the development of the UI.

4.3.1 Navigation

Navigation must be intuitive, simple, and clear throughout the application. The user must be aware at all times of where they are, what options are available to them, and how they can progress through the app.



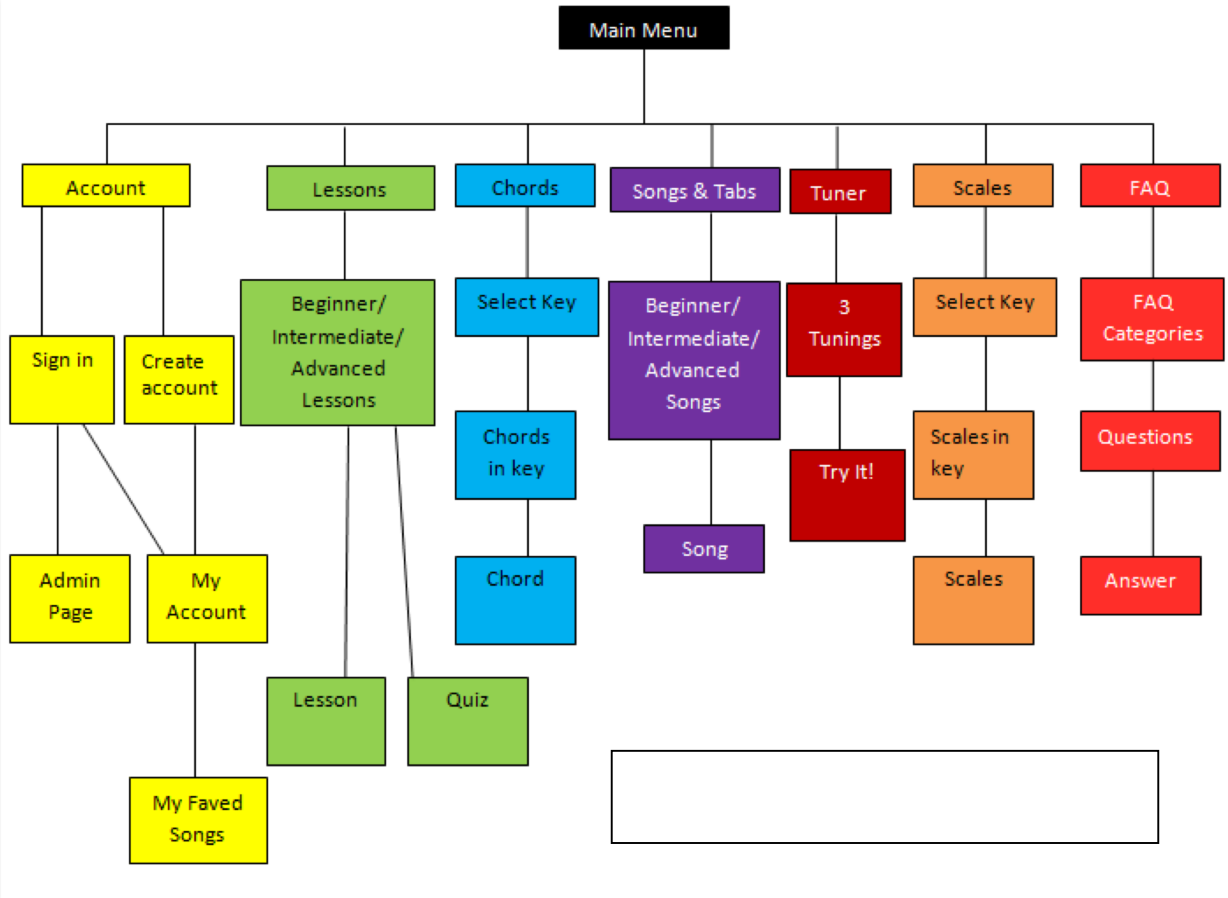
In the interests of ensuring simple and straightforward navigation through the application, the final screen of each section can be reached within three button selections. The user can always keep track of their location within the application, and they will not feel lost or overwhelmed by the wide array of options available to them within the sections. This also has the added benefit of ensuring that as the user becomes more familiar with the application, navigation will become much more efficient.

Given the fact that tablets utilise touch screen technology, the design and layout of the buttons on screen was of critical importance. Each button performs a specific task, and on most buttons there will either be a textual description of the task performed by that button, or a symbol which portrays to the user that button's function. For example, in Figure 10, all but one of the buttons displays text to indicate their own task. It was decided that the use of text was more appropriate than using symbols, which could be perhaps ambiguous, and confusing to a user unfamiliar with that symbol. Text descriptions ensure complete clarity, allowing the user to quickly and painlessly progress through the app. However, there are several different categories of button that must be discussed, as can be in Table 5.

Table 5 – The various buttons used throughout the application

Button	Purpose of this button
	This is a "Back" button, returning the user to the previous screen in the section when pressed. It was decided that this symbol was appropriate for the back button, as this is the universally recognised symbol for back or return.
	This is the "Home" button. This button is present on virtually every screen, and when pressed will return the user to the main menu. As with the back button, it was decided that an image of a stereotypical house would depict to the user that this is the home button; therefore this image was selected as the button background.
	This is the "Favourite Song" button. When a song sheet is displayed, this button will be found at the bottom of the screen. If a user wishes to save this song for practice, or has a particular affinity for that song, pressing this button will allow the user to "favourite" this song. The user can access their favourited songs through the "My Account" section. The heart image used has connotations of love and affection, therefore this will clearly portray to the user that this is the "favourite" button. If this button is pressed for several seconds, known as a long click, the My Favourited Songs section will be launched.
	A music note button is only used in the Songs & Tabs section of the application, and is displayed alongside the names of each song that is available. When pressed, this button will open the song sheet of the requested song. Again, the music note clearly represents music, and has strong connotations with sheet music in particular. Therefore, it was decided that this would appropriately represent the song sheets for the available songs.
	This is a button that will provide general information and guidance to the user, and can be seen in Figure 10. When pressed, information regarding the options available to the user and how they can progress through the app will be provided. The plectrum provides an instantly recognisable representation of assistance throughout the app, alongside the fact that the image fits in with the guitar and music theme.
	This button is only used in the FAQ section, and represents the various topics within that section. When this button is pressed, other questions within these sections will be displayed. This symbol has obvious connections with questions, or queries, and therefore was deemed to be appropriate to represent the buttons in the FAQ section.
	This button is the "Quiz" button, and is only used in three screens within the app – Beginner, Intermediate, and Advanced Lessons. This button will launch the quizzes for each difficulty level, however they can only be accessed when a user is signed in, and has completed every lesson in each section. If the user presses this button and one of these conditions has not been met, they will be met with an appropriate error message. The "QUIZ" text on the button leaves little ambiguity regarding the purpose of this button, therefore it was decided that this would be an appropriate background.
	This button allows a user to mark a lesson as being complete. When a user is signed in, the tick will be at the bottom corner of each lesson screen, and when pressed, this lesson's code will be added to the lesson progress field in the database. However, this button is only visible once a user is signed in. The tick symbol represents completion, therefore it was decided that this would be suitable for the background of this button.

The navigational aspects of the application were designed to ensure that simplicity and usability were the core concepts of the design, and that almost instantaneous access to every section within the application was possible. On every screen, there is at least either a “back” button, or a “home” button, while both are present on the vast majority of screens. This ensures quick, straightforward access to previously accessed screens, while the easy access to the main menu allows the user to return to the beginning and access an entirely new section of the app. Alongside this, drop-down menus are used throughout the application, and can be found on almost all screens. This drop-down menu features eight options, which allow the user to access each of the main sections within the app, as well as the main menu. Navigation throughout the sections of the application is straightforward, as the user can access any other section from any screen. The complete navigation through the application can be seen in the site map depicted in Figure 11, as well as the flow chart in Appendix 2 and the UML Activity diagram in Appendix 3.



The site map depicted in Figure 11 clearly demonstrates the wide range of options available to the user, and the methods through which they can access these features. This site map only provides a basic overview of the total functionality and progression through the app, as in order to provide the full progress, encompassing all potential pathways, a very large and complicated diagram would be required, and this would be entirely unnecessary. For example, in relation to the “Chords” section depicted in Figure 11, the steps are as follows: Select Key – Chords In Key – Chord. However, there are

nine keys displayed, with each key containing seven different chords. Rather than detailing the progression through this section to each chord, a more basic approach is clearly desirable.

In order to ensure that the application was designed in the most effective, and technically efficient, manner, both in terms of its user interface and functionality, it was imperative to create storyboards prior to the development of the application itself. This would also have the added benefit of allowing the developer to visualise how the user would progress through the application, as well as determine the algorithms and classes that would be required in order for the app to operate as intended. A sample storyboard for the "Tuner" section can be seen in Figure 12, with further storyboards for all other sections being included in Appendix 4.

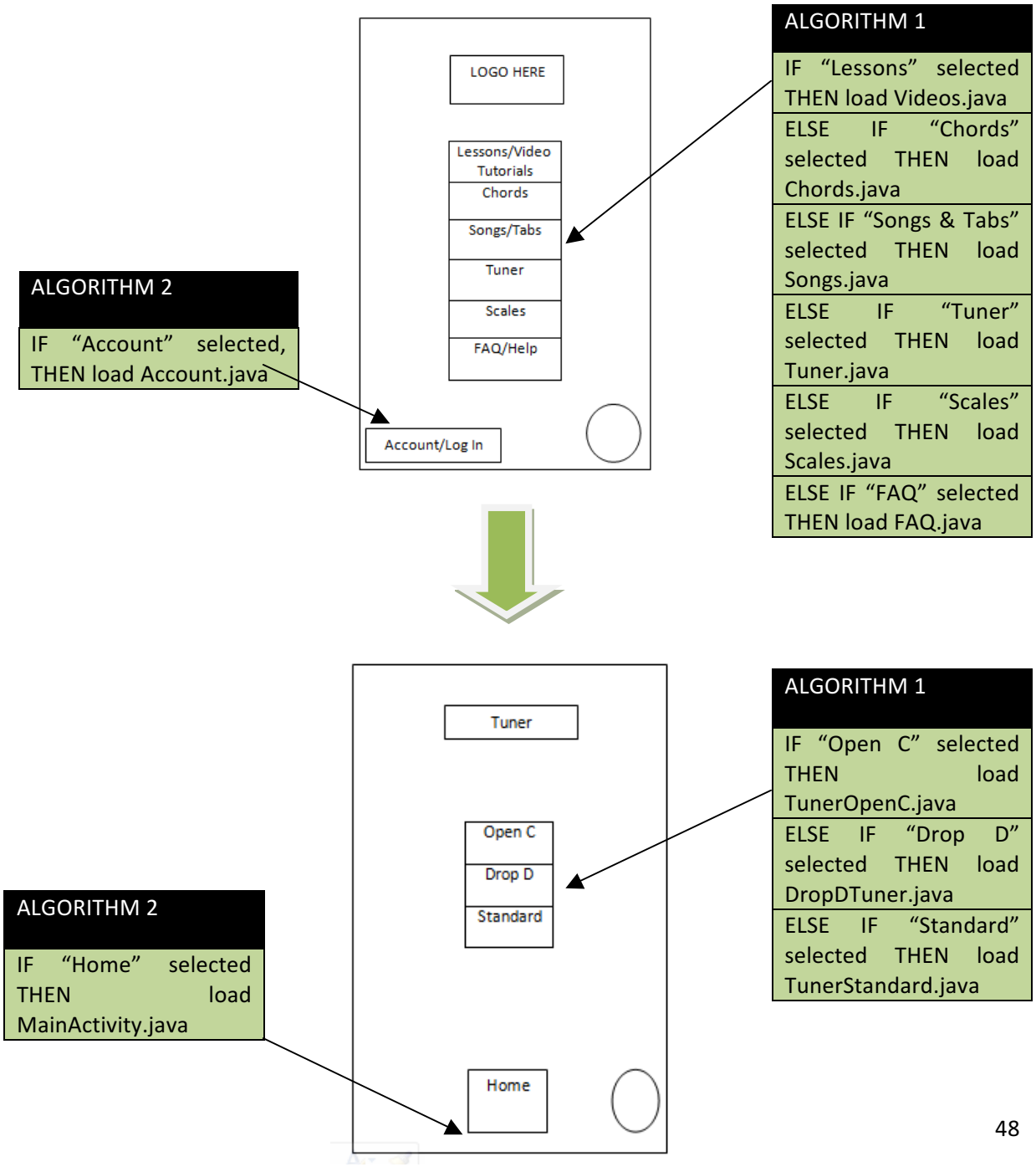
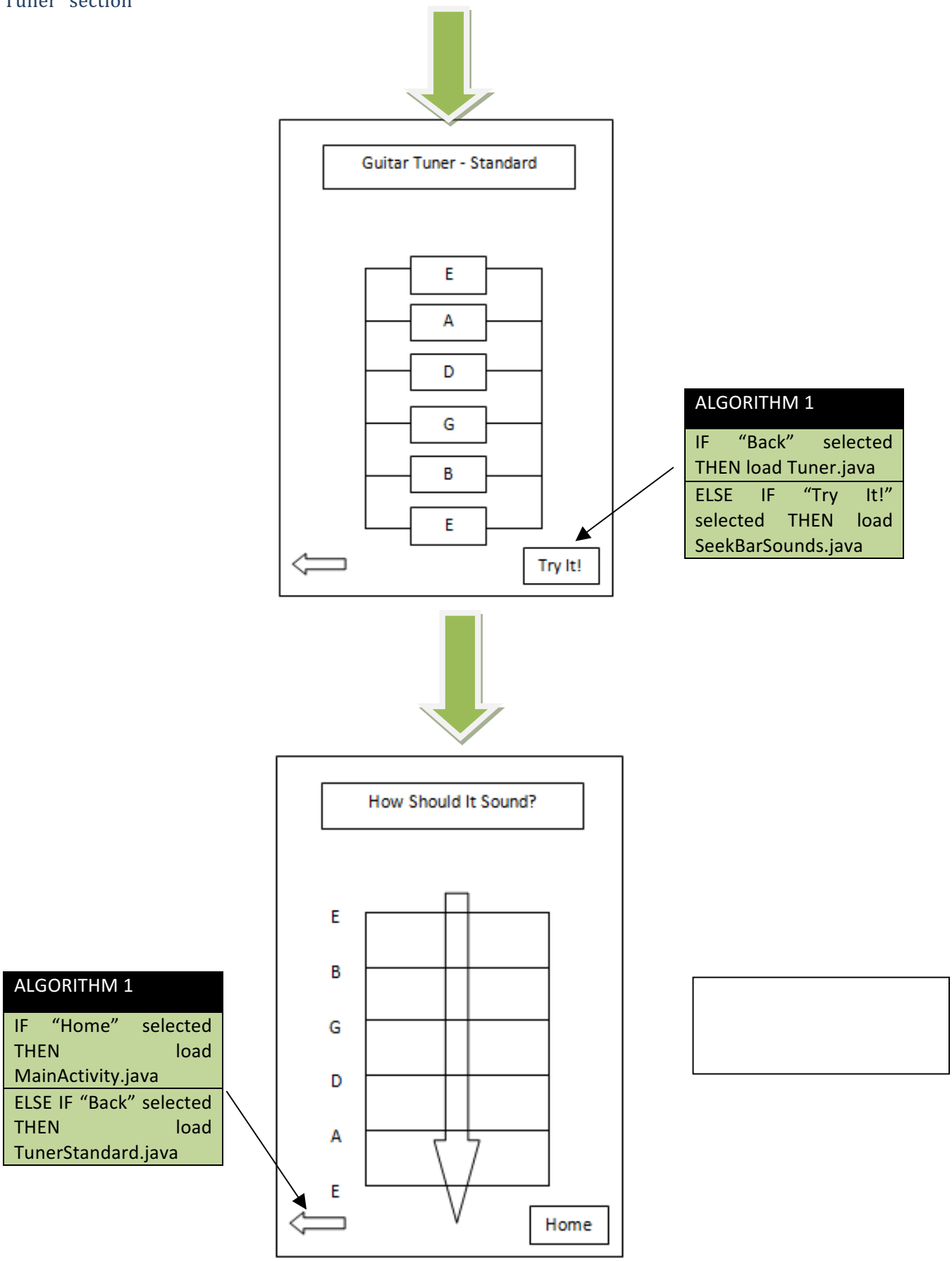
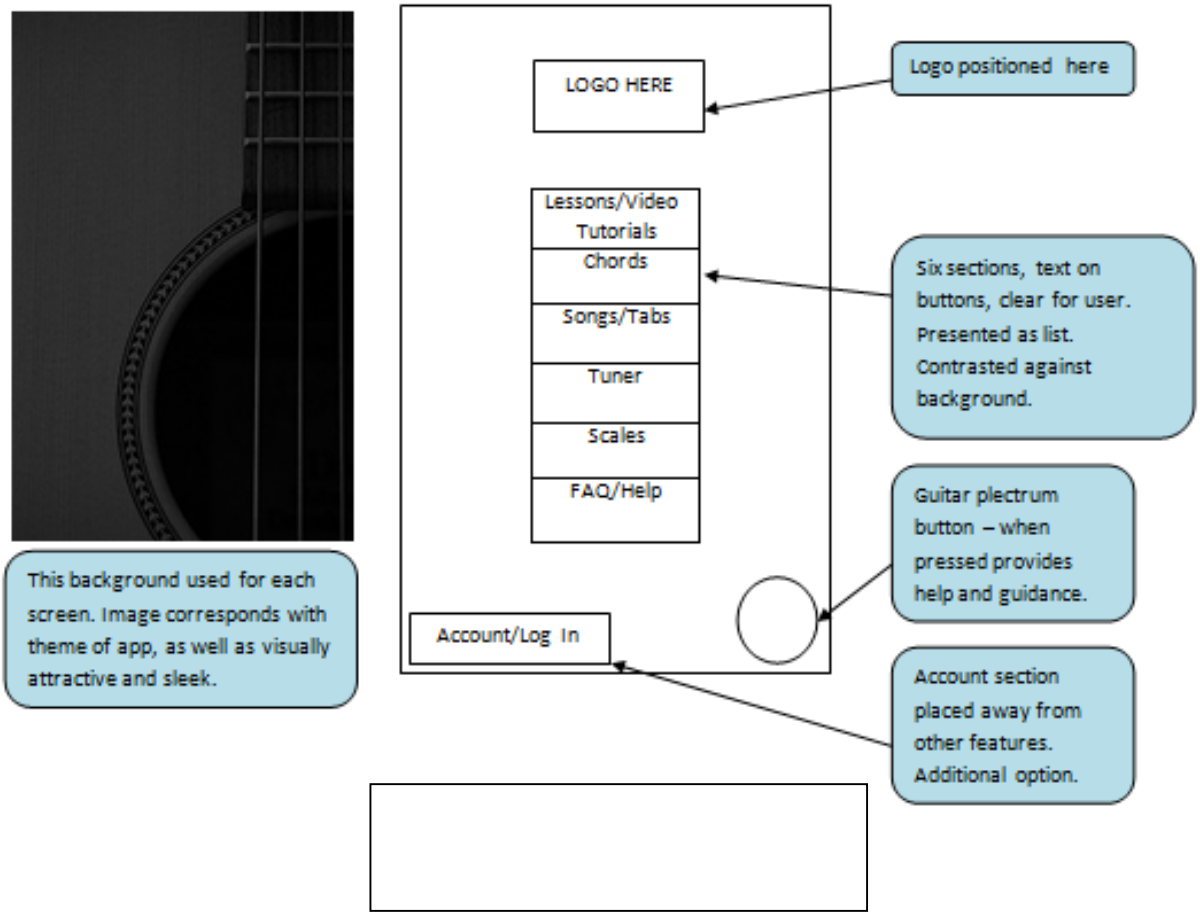


Figure 12 – Storyboards for "Tuner" section



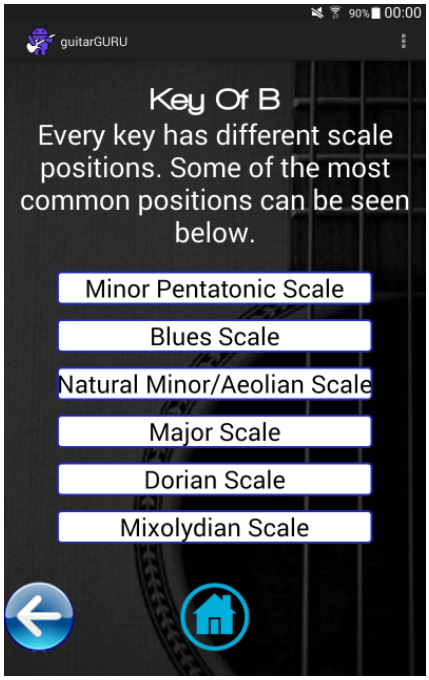
4.3.2 Page Layout and Colour Schemes

As has previously been discussed, Nielsen's 10 Heuristics and Shneiderman's eight golden rules are staunch advocates of the significance of consistency throughout a well-designed user interface. Alongside this, the foundation of an excellent user, and HCI, experience is focused on clarity, simplicity, and consistency among the screens. It is because of this that the layout and colour scheme of the application were placed at the forefront of the design process. Simply put, the UI is one of the most important factors in the success of any piece of software. If the layout is confusing and lacks clarity, for example if the content on screen is crushed together, it may deter both new users and previous users from using the app. Prior to the final design of the screen layouts, significant planning was undertaken to ensure that the layouts were simple and clear, as well as being aesthetically pleasing. A basic prototype for the page layout can be seen in Figure 13, and several wireframe designs can be found in Appendix 5.

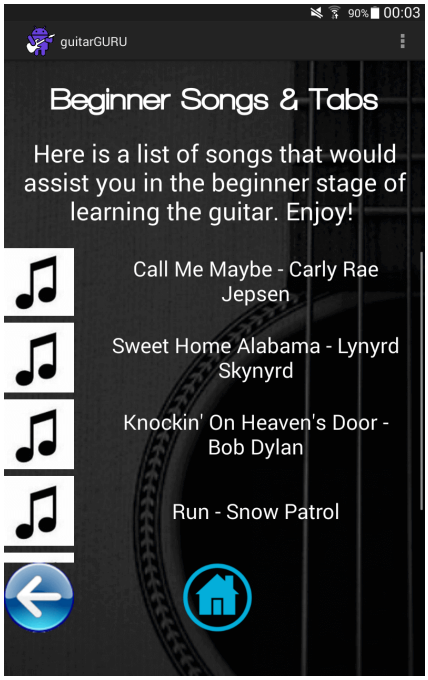


re 14 – Three screenshots of three different screens found in various sections of the application.

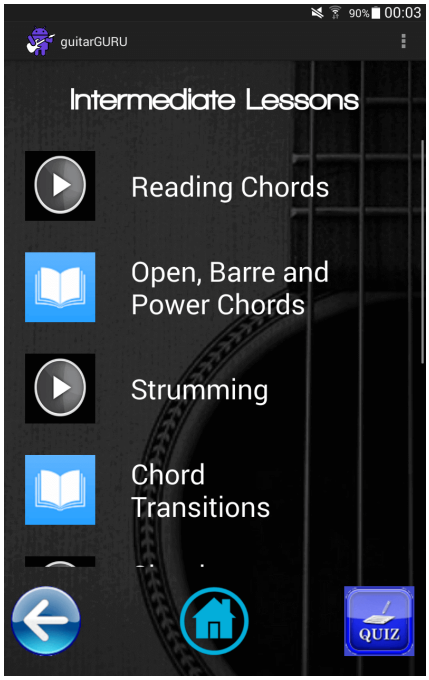
As can be seen in the three screenshots found in Figure 14, the principles of Nielsen, Shneiderman and the fundamentals of HCI were crucial in the design of the page layout.



(Figure 14.1)



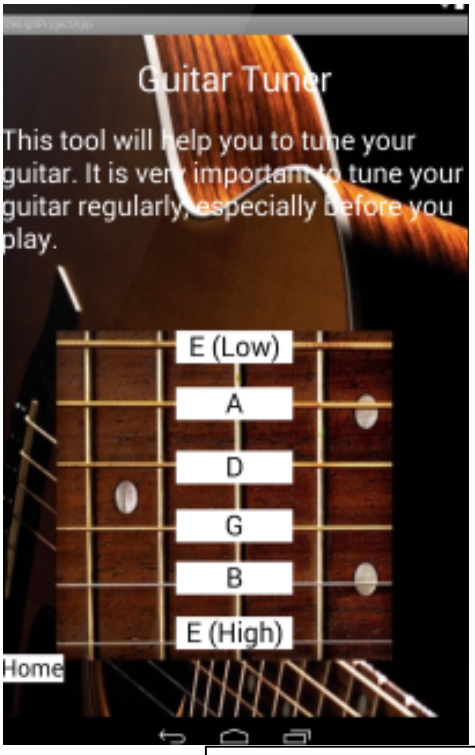
(Figure 14.2)

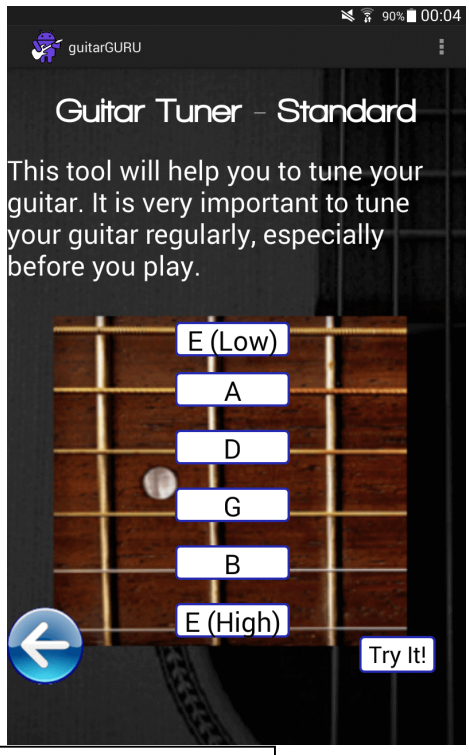


(Figure 14.3)

Figure 14.1 came from the "Scales" section, Figure 14.2 from the "Songs & Tabs" section, and Figure 14.3 from the "Lessons" section. As can be seen, consistency is apparent throughout the various sections of the application. There are various notable features and functions that remain ever-present on virtually every screen. Along the bottom of each screen is where the key navigational tools reside, with the exception of the drop-down menu, which will be subsequently discussed. In all three screenshots in Figure 14, both a "Home" and "Back" button are present along the bottom. Every screen in the application contains either a "Back" button or a "Home" button. However, in some cases, having both buttons would be simply redundant; for example, if the "back" button would return the user to the main menu, then only a "Home" button is required. In terms of consistency, all of these buttons have the same symbol as a background, with these symbols being universally regarded as a representation of the functions of the buttons. In the right screenshot of the "Intermediate Lessons" section, a button is present labelled "QUIZ" that is noticeably absent from the other screenshots. Once a user is signed in and has completed all lessons in this section, this button will allow the user to access a quiz that will test their knowledge. This button has the same appearance, and is positioned in the same place, for all three "Lessons" sections. Another notable navigational tool is the drop-down menu, which is again present on virtually every screen. At the top right corner of Figures 14.1, 14.2, and 14.3, three vertical dots can be seen. These dots represent the drop-down menu and, when pressed, will display links to every section of the app, thus providing quick and convenient access to these other sections.

Figure 14 is also representative of every screen in the application regarding the colour, size, and placement of the text, as well as the colour of the buttons and the background of each screen. The title is placed at the top of the screen, in large custom font which immediately catches the eyes of the user and informs them of their current location in the application, as well as the features and options available to them. The colour white is used for all text, as it provides a clear contrast to the dark background image. White is also used as the background colour for all buttons, again for reasons of contrast to the background, ensuring that all text on the buttons, and the buttons themselves, are clearly legible and can be seen by the user with no trouble. The current background image is a fairly recent amendment to the application, with an entirely different image having been adopted up until that point. Both the previous background and the current background images can be seen in Figure 15.

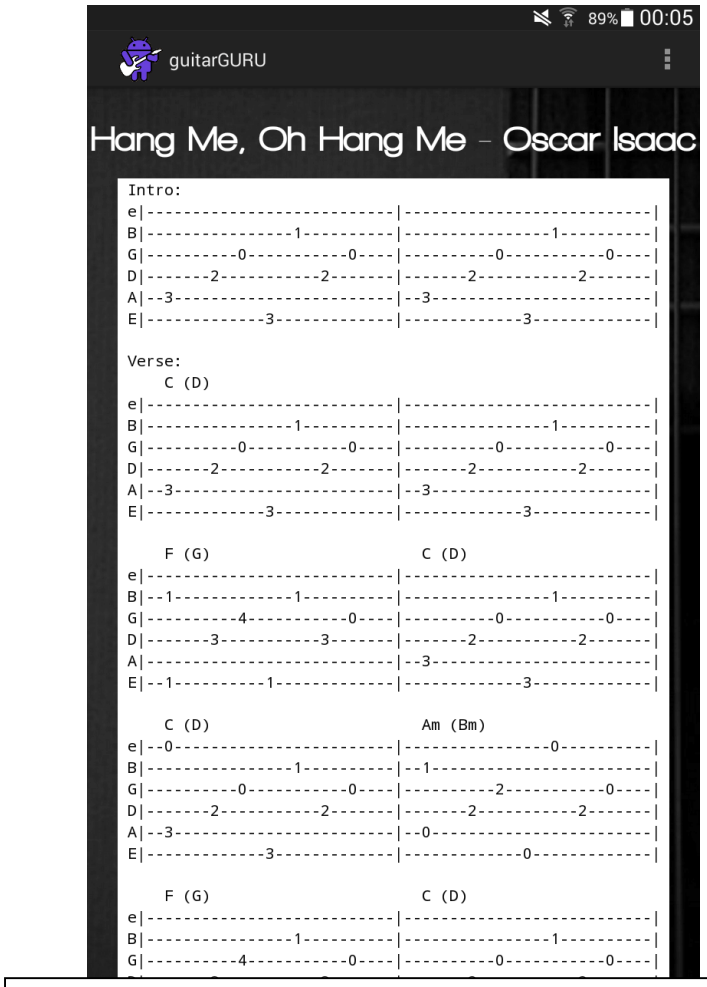




When comparing the previous background with the current background, the reasons for change become quickly apparent. Regarding the former, both the title and the description underneath are very difficult to read, particularly when compared to the current background. This proved to be a significant problem on a number of screens. Therefore, it was decided that a darker background image was required, in order to provide a greater contrast between the content on the screen and the background of the screen itself. However, once the developer came across the current background image, it was decided that this image was still insufficiently dark; therefore, the image was manually darkened. These changes have led to the desired result – the text and buttons are clearly legible, and stand out strongly against the darker background. Alongside this, the utilisation of the darker image also has an added benefit, in providing a sleek and attractive look to each screen.

Figure 16 – Monospace font used for displaying tablature.

Regarding the font choices for the text that will be displayed, it was decided that the default font provided by Android Studio, named “Normal”, was the most appropriate option for all text, with the exception of the titles, which use a custom font named “Walkway”. There is a reason as to why the “Normal” font is selected as default – it is legible and clear, with each character being easy to read, regardless of the character displayed or its case. However, when displaying tablature for the user to play, a font named “monospace” was adopted. The significant benefit provided by this font, along with all monospace fonts, is that each character occupies the exact same amount of horizontal space. This is crucial for the displaying of tablature, as each note must be played in the correct order, from left to right. In other fonts, the characters representing numbers occupied more space than “-“, therefore the order of the tabs was incorrect. Monospace font allows for the tabs to be displayed in the correct order, and with the basic timing of the notes intact, as can be seen in Figure 16.



With particular reference to the design of the user interface itself, various alternative design tools and libraries were considered in order to assist in the design and development process. The first of these tools is Bootstrap, which enables a user to create user interfaces for websites and web applications. Unfortunately, however, Bootstrap is limited to these two modes of development, and cannot be used

in relation to Android development. In order to provide visual feedback to the user when a button has been pressed, a library named "RippleEffect" was implemented into the app/lib folder, which can be found in the Project Files section of Android Studio. This custom library was downloaded from GitHub and allows the developer to design each button so that when a user presses this button, a ripple emanates from the point on the screen where the user pressed, as highlighted in Figure 17. The method through which this library was adopted into the coding of the button shall be analysed in the Implementation chapter.

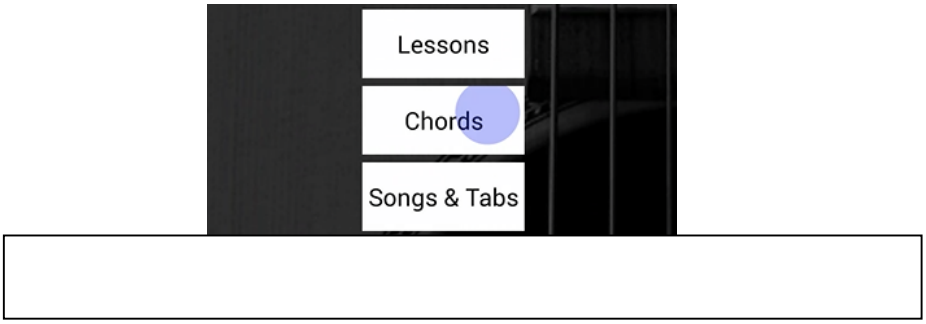


Figure 17 – The effect of the RippleEffect library on a pressed button.

4.4 Architectural Design

In order to design the application, and to ensure that the core functionality operated as intended, it was necessary to incorporate numerous different software components into the app. The implementation of these components refers in particular to the development of the database, which shall be discussed below.

4.4.1 Core Programming Languages Used – Java and SQL

During the coding of the app, two languages were essential – Java and SQL. As has been mentioned previously, the environment selected to develop guitarGURU was Android Studio, the official integrated development environment (IDE) for Android development, which allows developers to create their applications using the Java programming language. Java is one of the most popular programming languages across the world, perhaps due to its "write once, run anywhere" (WORA) configuration. However, while Android Studio has adopted Java as its language, it does not utilise the standard Java Virtual Machine (JVM). Instead, the code is converted into Java bytecode, and is then changed into Dalvik bytecode, and runs on Android's Dalvik Virtual Machine (DVM). In the developer's past coding experience, Java had been the core language used. This previous exposure to the language, and the various libraries available, was particularly helpful during the design and development stages.

In order to design the database that would be used in the application, it was necessary to utilise SQL, or Structured Query Language. SQL is the programming language that allows for data management, as well as data manipulation, within the database itself. However, the use of SQL in Android development is markedly different than with any other form of development. Android has an in-built SQL library named android.database.sqlite, or SQLite. A separate Java class must be created that extends the SQLiteOpenHelper class already present within the SQLite package. The database itself is stored on the device's internal storage, in the same disk space as the application. Therefore, all data is secure, and cannot be accessed by other applications. The separate class that is created is named DatabaseHelper,

and will control all database queries and transactions, using the SQL language. Within this class, a private static class is also present, named DBHelper, which extends the SQLiteOpenHelper class. Further analysis of the operation of this class, and the manner in which it manipulates the data held in the database, shall be undertaken in the Implementation chapter.

4.4.2 Database Design

In order for the database developed for use within guitarGURU to function as intended, it is imperative that the data held in the various tables is inter-related in some way. For example, if a user wishes to favourite a certain song, the primary key (user_id) from the Users table and the primary key (song_id) from the Songs table must be accessed and stored in the FavouritedSongs table. Therefore, it is apparent that adopting a relational database model is the most appropriate choice. This model is defined as "a collection of organised and inter-related data… This data is organised into sets (relations or tables), and each set of data may be related to another set of data" (Captain, 2013, p.2). In order to represent the relationships between the three database tables, the tables can be seen in Figure 18.

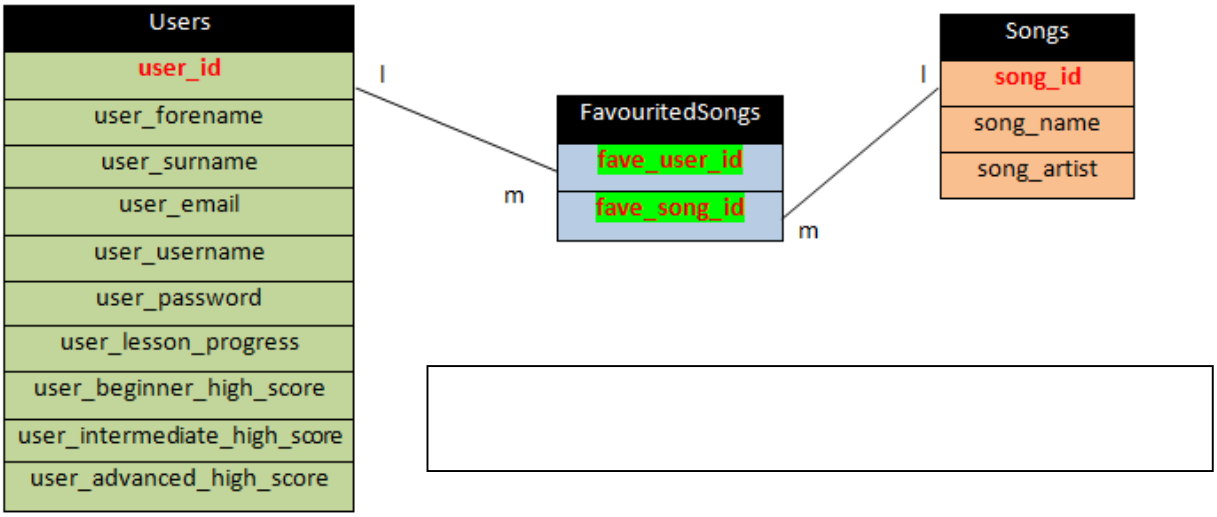
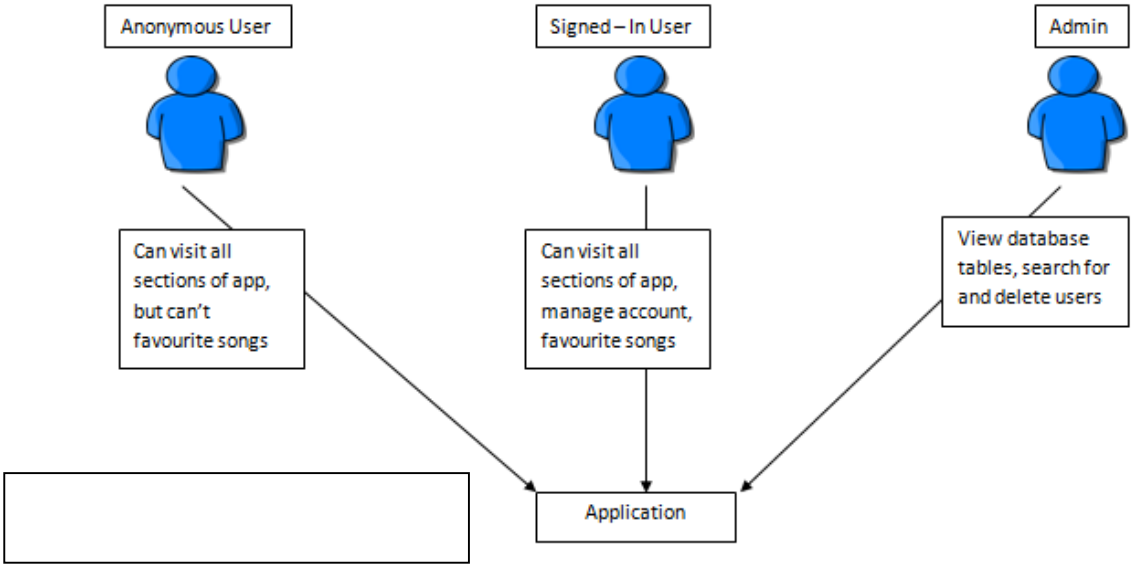


Figure 18 also displays the attributes present in each of the tables. The primary key of each table is displayed in red text, and the foreign keys are highlighted in green. There are ten attributes in the Users table, which shall be examined below.

- *user_id* - The user_id is an integer field, and is auto-incremented each time a new user creates an account. As there are therefore no duplicate entries, user_id functions as the primary key of this table, with each user being given a unique user ID.
- *user_forename* - The first name of the user, entered by the user when creating a new account.
- *user_surname* - The surname of the user, entered when the user creates a new account.
- *user_email* - The email address of the user, entered by the user when creating a new account.
- *user_username* - The username of the user, entered when creating a new account. A check is also in place to ensure that the username entered has not already been taken, ensuring that all usernames are unique.
- *user_password* – The password of the user, entered by the user when creating their account.

- *user_lesson_progress* – The data in this field is stored as text, as can be seen in Figure 21, and is automatically set to 0 when the account is created. As the user completes lessons, these lesson codes are added to this String.
- *user_beginner_high_score* – This integer field is initialised as 0 when the account is created, and will store the user's high score in the beginner quiz.
- *user_intermediate_high_score* – This integer field is initialised as 0 when the account is created, and will store the user's high score in the intermediate quiz.
- *user_advanced_high_score* – This integer field is initialised as 0 when the account is created, and will store the user's high score in the advanced quiz.

Further analysis surrounding the lesson progress and high scores will be carried out in the Implementation chapter. It is also important to note that there are three distinct groups of users that can use the app – anonymous users, signed-in users, and the Administrator, as can be seen in Figure 19. Users do not have to create an account to use the app; however users that are signed in have access to additional features, such as the quizzes and the ability to favourite songs. Alongside these users, there is also the application administrator (the developer). When the application is downloaded, and the database is created, the Admin account is created automatically within the Users table. The Admin can view all of the data in the database tables, search for users, and delete users. Only the Administrator has access to these features, through the Admin section, which can only be reached using the specific administrator username and password, known only to the developer.



Regarding the Songs table, there are three attributes – song_id, song_name, and song_artist. As with the user_id in the Users table, the song_id is an integer that is auto-incremented with each new song that is added. Therefore, each song has a unique song_id, with the song_id acting as the primary key for this table. All fifteen songs are added at the creation of the database itself. The song_name and song_artist attributes are text fields that will hold both the name of the song and the artist respectively.

Finally, regarding the FavouritedSongs table, there are only two attributes – fave_user_id, and fave_song_id. As can be seen in Figure 18, both attributes are displayed in red text, and are highlighted in green. Therefore, fave_user_id and fave_song_id are both primary and foreign keys. The fave_user_id attribute is a foreign key referencing the user_id primary key in the Users table, and the fave_song_id is a foreign key referencing the song_id primary key in the Songs table. Combined, these attributes comprise a composite primary key.

Figure 20 – Entity Relationship diagram demonstrating the relationships between the three tables.

During the initial design stages of the database, the developer was faced with two possibilities regarding the FavouritedSongs table. The first possibility was to create a unique ID for each combination of fave_song_id and fave_user_id, in the same manner as the song_id and user_id in the Songs and Users table, in that this unique ID would be auto-incremented with each new entry. However, there is one major problem with this idea – there is no straightforward way to prevent duplicate entries of fave_song_ids and fave_user_ids. If, for example, a user favourited a song twice, and then attempted to un-favourite this same song, but did so only once, their user ID and the song ID would still be listed as being favourited. This idea was quickly abandoned in favour of the current design. In the final FavouritedSongs table, the fact that the fave_user_id and fave_song_id fields form a composite primary key ensures that there can be no other entries that contain the same data. For example, if a user with the user ID of 3 favourites a song with the song ID of 6, the composite primary key fields in the FavouritedSongs table would be 3 and 6. However, if the same user tries to favourite this same song again, 3 and 6 would attempt to be entered into the table. However, this would cause an error, as this would be a duplicate primary key. An error message will then be displayed to the user, informing them that they have already favourited this song. This allows the user to favourite any song they desire, however they are unable to favourite a song when they have already done so. Once the user un-favourites said song, they are free to favourite it once again, at which point it will be added to the FavouritedSongs table.

There are clear, and very necessary, relationships between the three tables in the database. In order to visually represent this relationship, an Entity Relationship (ER) diagram has been created in Figure 20.

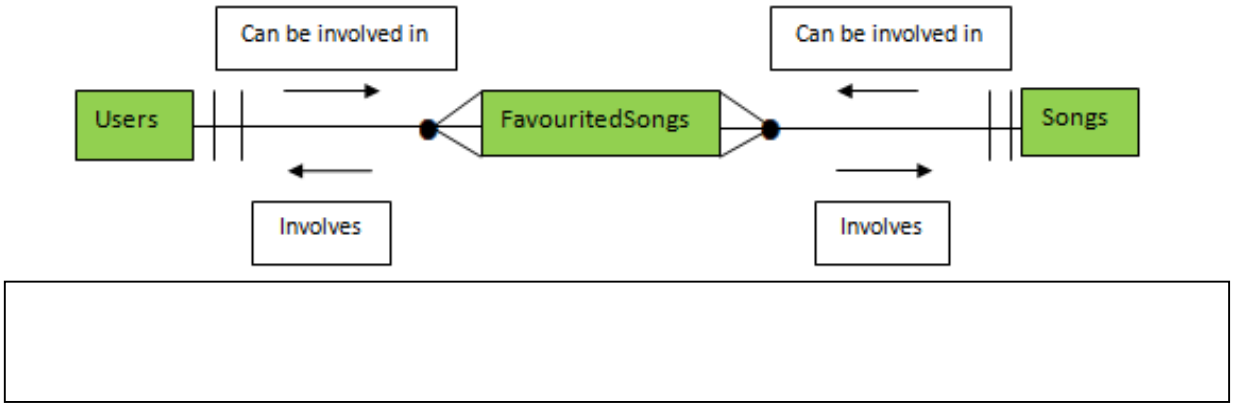
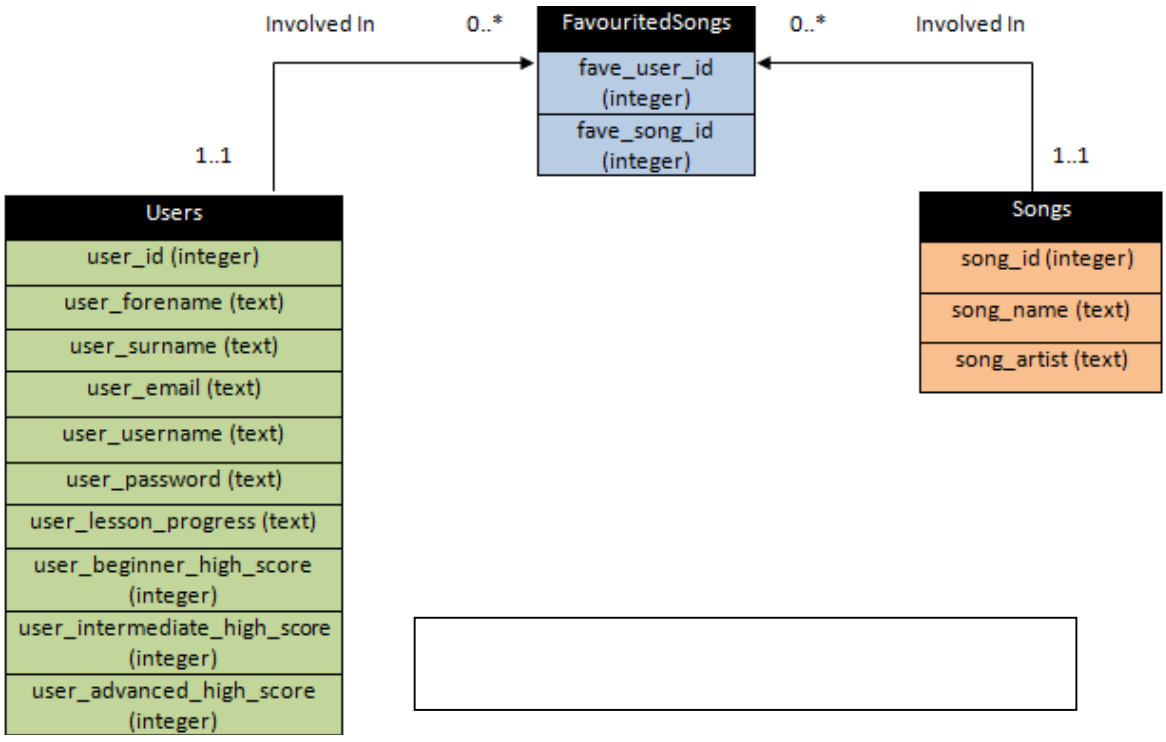


Figure 20 clearly demonstrates the relationships between the three tables. An analysis of this diagram indicates that one user can be involved in zero to many favourited songs, but one favourited song involves one and only one user. The reasons for this are fairly apparent – a user is under no obligation to favourite any songs, or they can favourite as many songs as they so desire. However, one entry in the FavouritedSongs table involves the user ID of one and only one user – the user who has favourited the song in question. A FavouritedSongs entry must have one user ID, and cannot have multiple user IDs, therefore one and only one user is involved. The other major relationship involves the FavouritedSongs table and the Songs table. As Figure 20 depicts, a favourited song involves one and only song, while one song can be involved in zero to many favourited songs. It is entirely possible that a song may not be favourited at all by any users of the app, while a song may also be favourited by numerous users. On the other hand, as has been mentioned previously, a favourited song requires only two attributes – the user ID of the user in question, and the song ID of the song that has been favourited. Therefore, for each favourited song, one and only one song is involved. A visual representation of the entire database, and the relationships between the tables, can be seen in the UML diagram found in Figure 21.

Figure 21 – UML Diagram of the database tables.



4.5 Java Classes

As well as designing both the user interface and the database, planning was undertaken regarding the various Java classes that would be used to implement the various features into the app. In the development of the entire application, a total of 68 Java classes were required. The Java classes that will be used for various sections of the app will be analysed in the UML Class diagrams found below, alongside the relationships between the classes, and how they will interact. Further demonstration of the way in which a user can access the various classes and progress through the application can also be

seen in the Appendices 2, 3 and 4. The operation and interactions of the classes will be discussed in great detail in the Implementation chapter.

4.5.1 Beginner Lessons Section

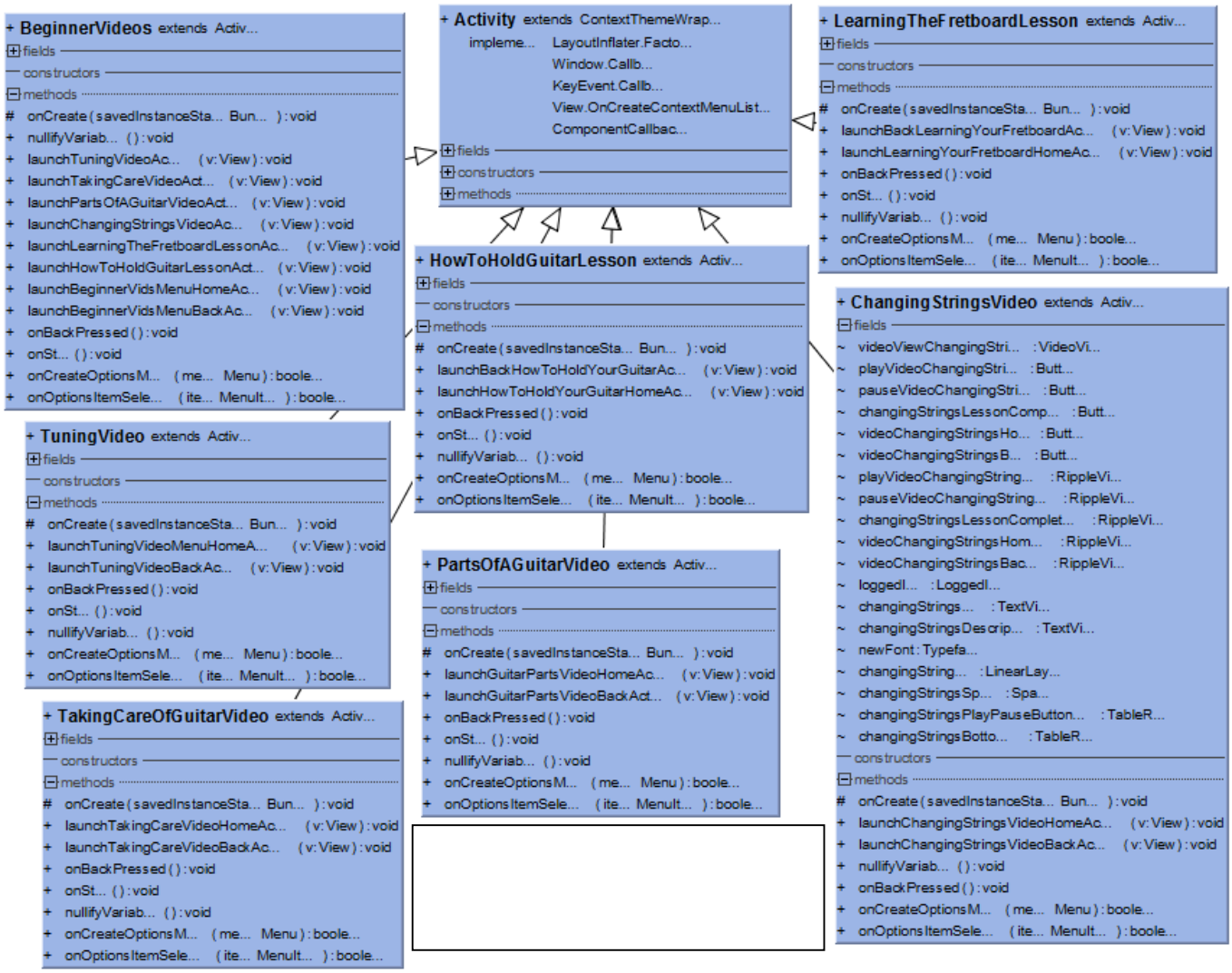
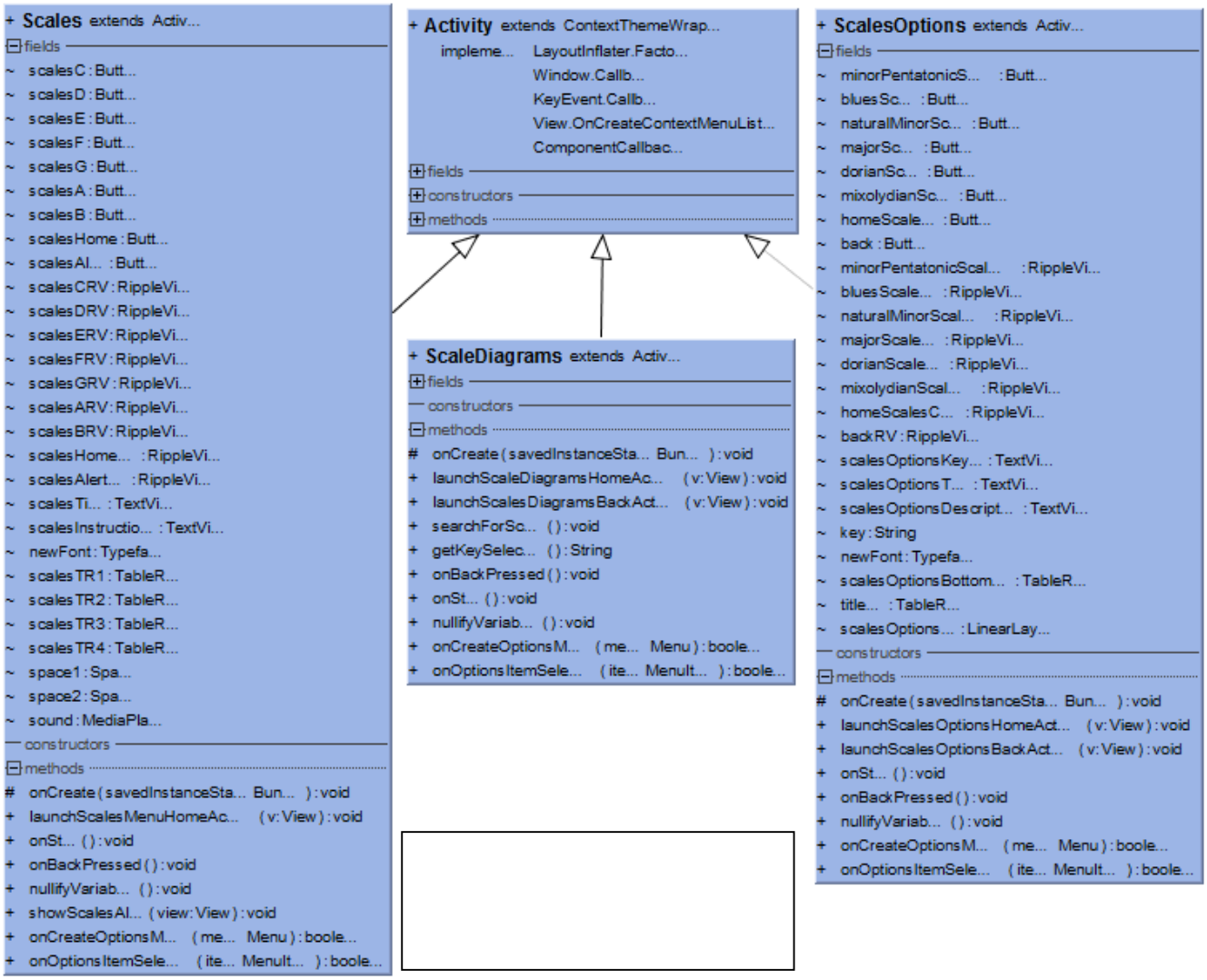


Figure 22 provides a demonstration of the various classes that will be used in the Beginner Lessons section, and their relationships. The arrows from each class pointing towards the "Activity" class clearly highlights that all of these classes extend the Android Activity class, allowing them to be launched as activities within guitarGURU. Each of the class components in the Class diagram is a Java class, displaying all of the variables and methods used within that class. The BeginnerVideos.java class component, which is itself accessed through the Videos.java class, features the methods used for launching all of the classes present within Figure 22, each of which represent a lesson within the Beginner Lessons section, with the obvious exceptions of the BeginnerLessons class itself and the Activity class. In order to create this section, eight classes in total were required, including the Activity class. The class design for the Beginner Lessons section closely resembles that of both the Intermediate and Advanced Lessons sections, in that each lesson menu, and each lesson itself, has its own Java class.

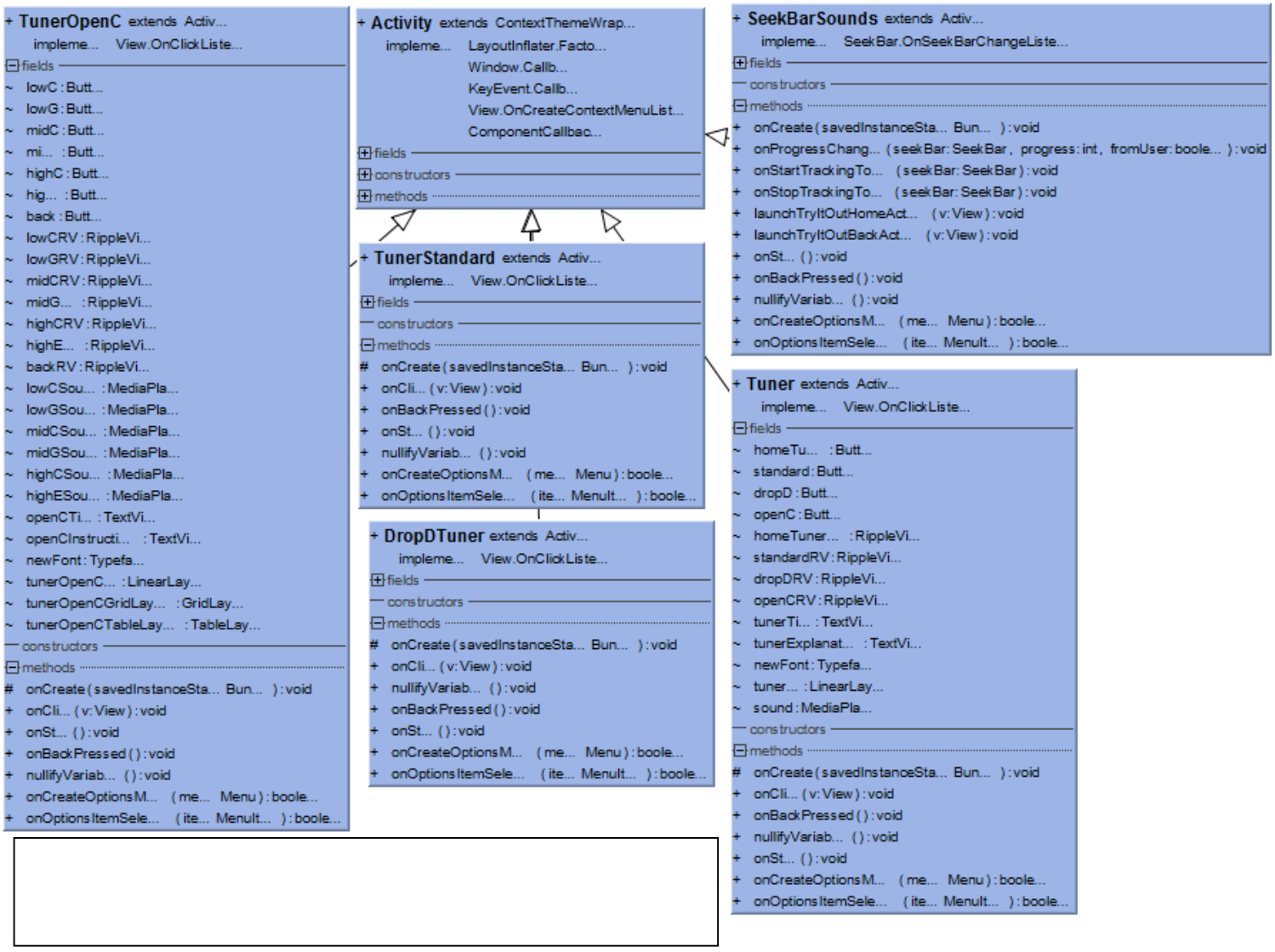
4.5.2 Scales Section



The classes needed in order to create the Scales section of the app can be found in the UML Class diagram represented in Figure 23. Again, as with the Beginner Lessons classes, each class extends the Activity class, thus allowing them to be launched as activities in the application. This extension of the Activity class is highlighted by the use of the arrows pointing from each class to the Activity component in Figure 23. When the Scales section is selected by the user, the Scales.java class is launched. The list of variables displayed within the Scales.java class component contains buttons named, for example, scalesC, scalesD, and scalesA. When the user presses one of these buttons, the ScalesOptions.java class is launched. This class will display the names of six scales available to the user, who will then select a scale, at which point the ScaleDiagrams.java class will be launched, which will show the user how to play the scale.

4.5.3 Tuner Section

Figure 24 demonstrates the various classes that are required in order to create the Tuner section. Again, as with Figures 22 and 23, all of these classes extend the Activity class, allowing them to be launched as activities. The Tuner.java class component in Figure 24 is used to create a menu within the app, which will display the options available to the user – Standard, Drop D and Open C Tuners. When the user selects an option, TunerOpenC.java, TunerStandard.java, or DropDTuner.java will be launched and the tuner will be displayed. As the variables and methods used for each tuner are similar, it was decided to only display those belonging to the TunerOpenC.java class. An additional class, SeekBarSounds.java, is used only in relation to the Standard tuner, and can only be accessed through the TunerStandard.java class. This class features methods that allow for the "progress" variable to be monitored, as Figure 24 demonstrates, thus allowing the SeekBar widget in this class to be used.



gure 25 – The LoggedInID.java class.

4.5.4 LoggedInID.java Class

In the development of guitarGURU, the vast majority of the Java classes used extend the Android Activity class, allowing them to be launched as activities within the app. These classes are used to display all of the sections and subsections that can be accessed by the user. In relation to the FAQ section, each category menu, and each question, uses a separate Java class. For example, the class FAQEquipment.java displays the "Equipment" category, with FAQWhatGuitar.java and FAQAccessories.java being the classes launched to display the answers to the questions listed in this category. However, there are two Java classes used in the app that do not extend the Activity class; they are used solely for processing purposes. The first of these, LoggedInID.java, will be examined, with the remaining class, DatabaseHelper.java, being subsequently discussed.

```
public class LoggedInID {

    //This class will store the login status of the application, an
    //logged in. When a user signs out, loginNum set to -1, loggedI
    static int loginNum;
    static boolean loggedIn = false;

    //This default constructor will allow the user ID and login sta
    public LoggedInID(){    }//Default constructor

    //This constructor takes the person ID and loggedIn status as p
    public LoggedInID(int personID, boolean loggedInNow){
        loginNum = personID;
        loggedIn = loggedInNow;
    }//LoggedInID

    //This method will return the userID of the logged in user
    public int getLoginNum() { return loginNum; }//getLoginNum

    //This method will take in the new user ID as a parameter
    public void setLoginNum(int newLogin) { loginNum = newLogin; }/

    //This method will return to logged in status boolean
    public static boolean isLoggedIn() { return loggedIn; }//isLogg

    //This method will take in the new logged in status as paramete
    public static void setLoggedIn(boolean loggedIn) {
        LoggedInID.loggedIn = loggedIn;
    }//setLoggedIn

}//class
```

The LoggedInID.java class, seen in Figure 25, is used to store the user ID of the user that is currently signed in. When a user signs in, the value of this user's user ID is retrieved from the database and is assigned to the static integer loginNum using the setLoginNum() method. Alongside this, the boolean loggedIn is set to true using the setLoggedIn() method. Should a user attempt to favourite a song or access a quiz, for example, a check is initially made regarding whether or not a user is in fact signed in, by retrieving the loggedIn boolean from this class using the isLoggedIn() method. If this method returns false, a user is not signed in, and an error message is displayed. Otherwise, a user is signed in, and the user is free to progress as they wish. The loginNum integer is used frequently throughout the app, as, for

example, when a user wishes to favourite a song, the song ID of that song is combined with the user ID and added to the FavouritedSongs table. The user ID is therefore retrieved using an instance of the LoggedInID.java class, and the associated method to return the loginNum variable. When a user wishes to sign out, the loginNum is set to -1, therefore it cannot match a user ID in the Users table, and the loggedIn boolean is set to false.

4.5.5 DatabaseHelper.java Class

This class, as with the LoggedInID class, was designed for processing purposes, as it controls the application's interactions with the database, in terms of both modifying and retrieving data. DatabaseHelper.java contains a static inner class named DBHelper, which extends the SQLiteOpenHelper.java class, allowing for the database in the app to be created, and its data to be altered and retrieved through various methods created within the DatabaseHelper class. For example, when the user wishes to mark a lesson as complete, the lesson code will be passed as a parameter to the setLessonProgress() method in DatabaseHelper, where it will be added to the previously existing USER_LESSON_PROGRESS String in this field, as can be seen in Figure 26.

```
public int setLessonProgress(long userID, String lesson) throws SQLException {
    int returnValue;
    ContentValues cvUpdate = new ContentValues();
    //If the user's current lesson progress doesn't contain the code lesson passed as a
    //parameter, this lesson is only newly completed. This code will be added to user's
    //lesson progress, and 1 is returned. Otherwise -1 is returned
    if (!getLessonProgress(userID).contains(lesson)) {
        cvUpdate.put(USER_LESSON_PROGRESS, getLessonProgress(userID) + lesson);
        ourDatabase.update(USER_TABLE_NAME, cvUpdate, USER_ID + " = " + userID, null);
        returnValue = 1; }//if
    else {
        returnValue = -1; }//else
    return returnValue;
```

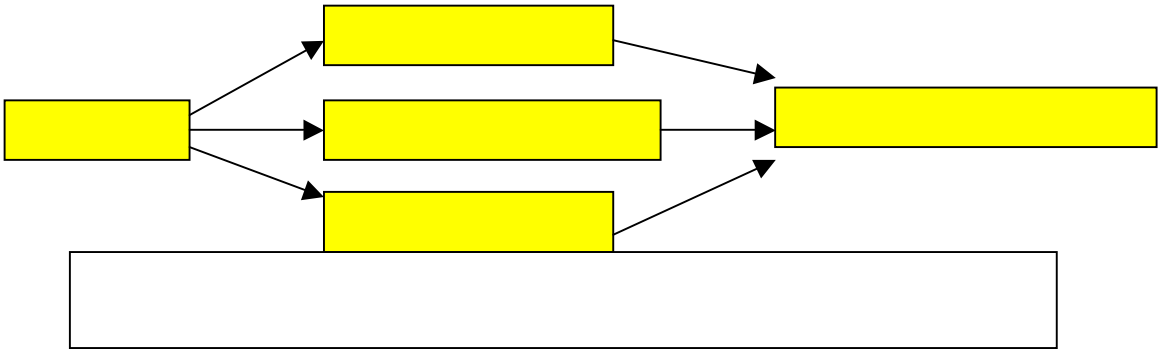
In order for other classes to access these methods, and thus interact with the database, they must create an instance of the DatabaseHelper class, and call the open() method, allowing communication with the class to be initiated. Once the transactions are completed, the close() method is called, and the communication channel to the DatabaseHelper class is terminated.

4.5.6 Other Java Classes

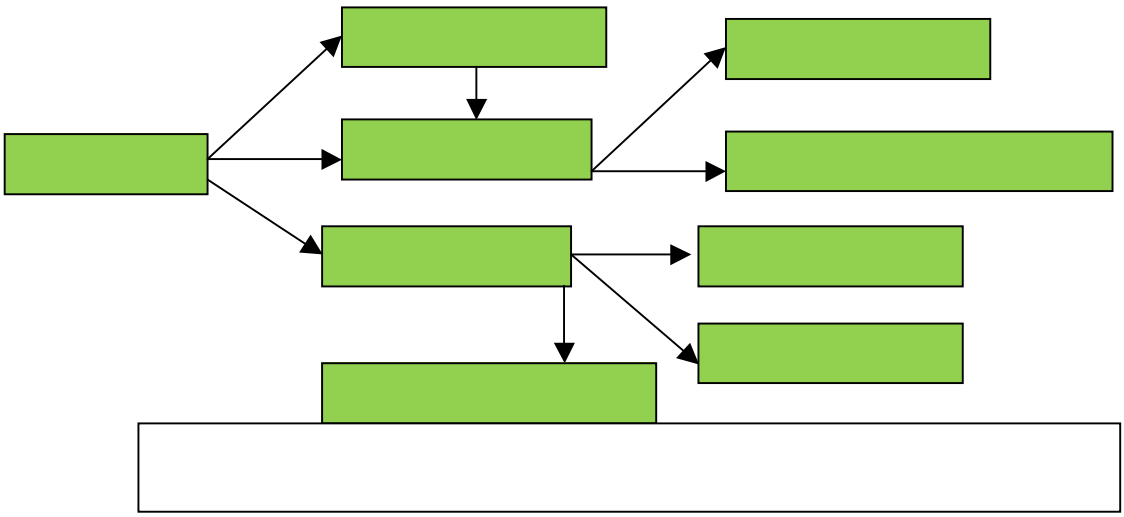
In the design of the Songs & Tabs section, a total of five Java classes were utilised. When the Songs & Tabs section is selected by the user, the Songs.java class is launched. This class will display three options to the user, in terms of song difficulty – Beginner, Intermediate, or Advanced. Depending on the option selected by the user, the BeginnerSongs.java, IntermediateSongs.java, or AdvancedSongs.java class will be launched. The five songs in the chosen section will then be displayed. When the user selects a song, the name of the song will be passed to the SongSelectedDisplayed.java class, which will use the name of the song to display the correct layout file, and initialise the appropriate variables. This process can be seen in Figure 27.

Figure 27 – The Java classes used throughout the Songs & Tabs section

Figure 28 – The Java classes used throughout the Account section.



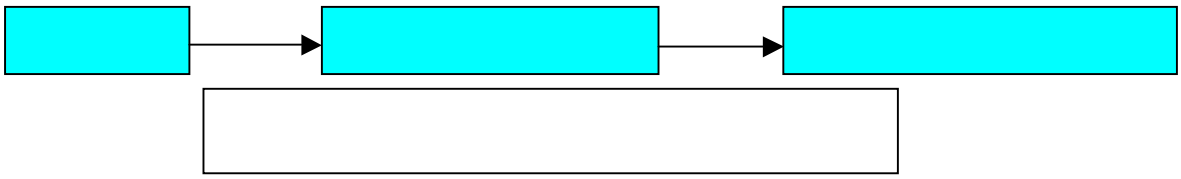
Nine classes are used throughout the Account section, four of which are available only to the administrator (the developer). When the Account section is selected by the user, the Account.java class is launched. From this class, the user can either sign in, thus launching the My Account section, or they can opt to create an account, thus accessing the NewAccount.java class. Should the user create a new account, and enter all of their details correctly, they will be taken to the My Account section, controlled by the MyAccount.java class. The user can then either access their favourited songs, launching the ViewUsersFavedSongs.java class, or edit their details, using the EditDetails.java class. However, it is possible that instead of a user signing in, the administrator may choose to sign in. If the administrator signs in using the username and password known only to the developer, the AdminPage.java class is launched. From this section, the administrator can view any of the three tables in the database, using the ViewAllSongs.java, ViewAllUsers.java, and ViewAllFavedSongs.java classes. The various classes used in the Account section can be seen in Figure 28.



Finally, there are three classes used in relation to the Chords section. When the Chords section is accessed by the user, the Chords.java class is launched. This class will display nine keys which can be selected by the user. Upon selecting a key, the name of this key is passed to the ChordsKeySelected.java class, which displays the seven chords within that key. Using the name of the key retrieved from Chords.java, the text on the buttons is changed to display the seven chord names. Once the user selects

Figure 29 – The classes used in the Chords section.

a chord, the key and the chord name are passed to the ChordsDisplaySelected.java class, which will display the appropriate diagram and a video of how to play the chord. A diagram of the various classes used in the Chords section, and the order in which they are accessed, can be seen in Figure 29.



In relation to the operation and interactions of the classes themselves, further in-depth examination will be undertaken in the Implementation chapter.

4.6 Summary

In this chapter, an analysis and a discussion was undertaken regarding the various design features that were selected for the application. In relation to the user interface, the importance of Shneiderman's golden rules and Nielsen's Heuristics were highlighted, and the layout of the screens of the user interface was analysed and explained. The architecture of the application was then examined, in particular the design of the database, and the relationships between the database tables, before the class design was considered, especially in relation to the main classes used and their purposes.

Chapter 5: Implementation

5.1 Introduction

Once the design of the application has been finalised, and all features and functions to be developed have been planned, the implementation stage of development can begin. It is at this period in the process that the ideas and designs that have been presented in the previous chapters are implemented into the final application. As shall be demonstrated in this chapter, the development of a prototype application was pivotal in this process; providing an early demonstration of the strengths and weaknesses of certain features, as well as any areas that could be significantly improved. Fortunately, meticulous planning and design during the early stages of development ensured that, for the most part, implementation was undertaken in both an efficient and relatively straightforward manner. However, as shall be discussed in this chapter, there are, perhaps unsurprisingly, several notable differences between the prototype application and the final application.

In this chapter, the implementation environment and process will be discussed in-depth. Subsequent to this, particular examination will be undertaken regarding the main Java classes that were utilised by the developer, and their operation within the app itself. The creation and implementation of the SQLite database into the app will be analysed, as will the manner through which the data within the tables can be modified and amended by the user. This will be followed by an explanation regarding the implementation of the various other sections within the app, while all additional libraries and code from other sources will be recognised, and their functions described.

5.2 Implementation Environment

There are various Integrated Development Environments (IDEs) that could be used for developing Android applications, including, for example, Eclipse and IntelliJ IDEA, which are two of the most popular Java IDEs used by developers. However, based on the research carried out by the developer regarding the suitability of these options, it is clear that Android Studio is the standout choice for developing Android applications. Android Studio is the official IDE for native Android development, and is available to download free of charge on Windows, Mac OS X, and Linux (Android Developers, 2016). The significant advantage in using Android Studio over Eclipse, for example, is that Eclipse requires considerably higher amounts of RAM space and a significantly higher CPU speed to operate. On the other hand, Android Studio's system needs are lower, and it provides a significantly more stable performance (Rajput, 2015). In order to develop the app, a Toshiba laptop (Windows 10, Intel Core i3-3110M, 2.40GHz CPU, 4GB RAM) was used. The testing, on the other hand, was carried out on a Samsung Galaxy Tab 4 7" tablet, using the Android 4.4 (API 19) operating system. As the application is specifically aimed at tablets, and in particular tablets of 7" screen size, this would allow the developer to test how the application would operate on its target device.

5.3 Implementation Process

In line with the majority of the software development industry, the Agile methodology was adopted by the developer during the development process. The advantages provided by the Agile model in terms of software development are notable. The regular increments, named sprints, allow for certain aspects of

development to be prioritised. Sprints themselves are the basic unit of development, and are "timeboxed", in that they are restricted to a specific duration. Once a sprint has been completed, the developer is able to reassess their targets, and perhaps shift their focus to other important areas of development, as can be seen in Figure 30. The adaptive nature of planning, and regular reassessment of goals, provides the most efficient development process possible. As well as this, at the termination of each sprint, testing is carried out on the software that has been developed. This frequent testing allows any bugs, issues, or potential improvements to be noted at an early stage, and dealt with appropriately.

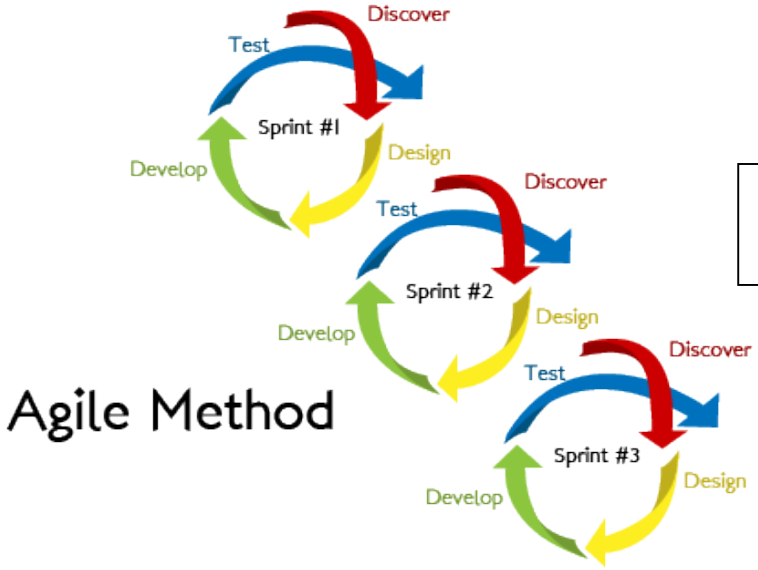
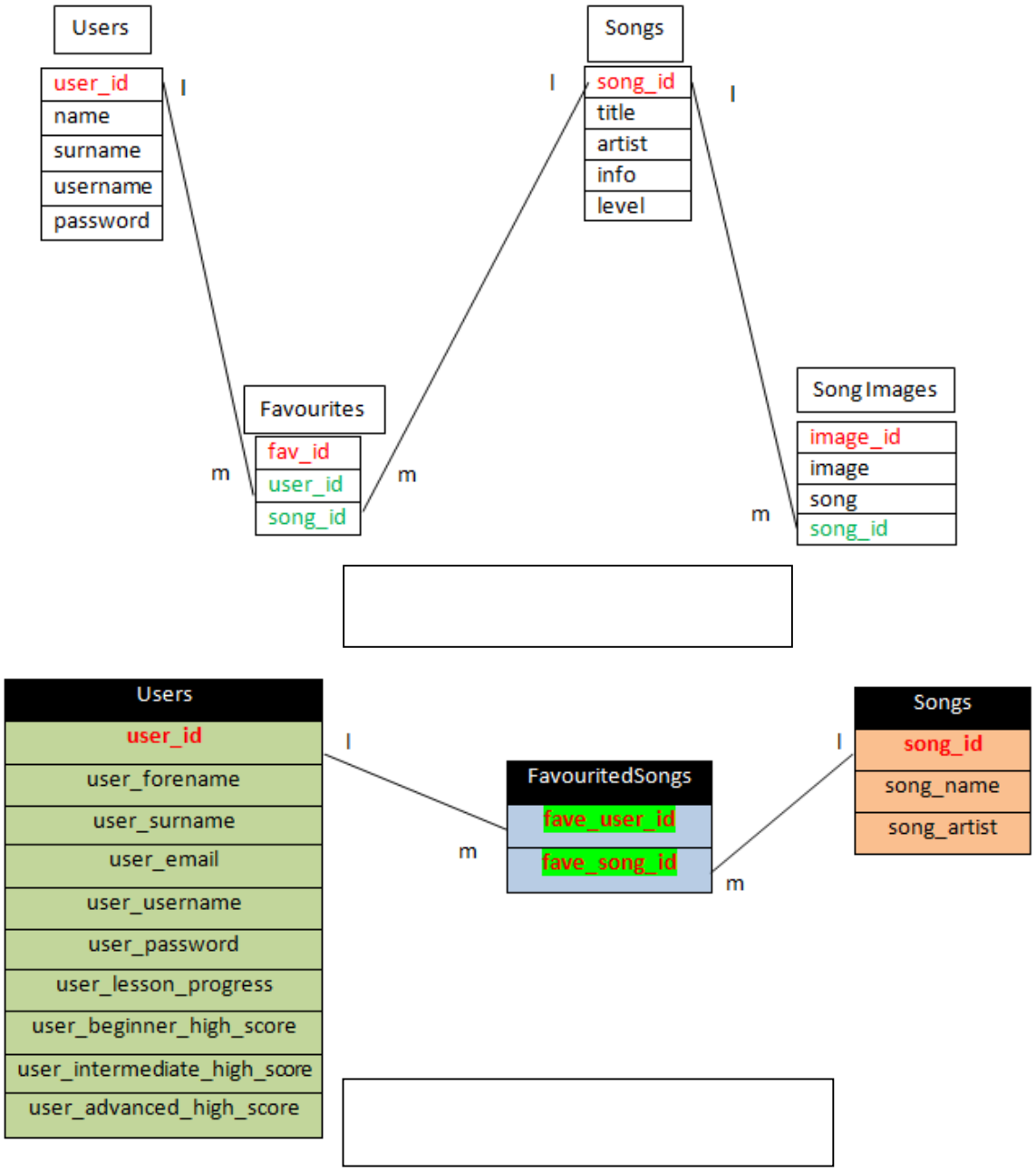


Figure 30 – The Agile methodology model

For example, the Agile model proved to be particularly beneficial to the developer during the development of the prototype application. It was decided at an early stage that the design and implementation of the database should be given precedence over other functional and aesthetic aspects of the app. The database was central to the success of the application, in that many of the key features relied almost solely on its operation – namely the ability to create an account, to track lesson progress, and to favourite songs. Once the database had been implemented and tested, and this iteration was completed, the developer then reassessed the next set of targets to be prioritised, and moved onto the next sprint.

There are several notable differences between the prototype that was designed and the final application. For example, in relation to the database tables themselves, there are numerous amendments that should be discussed. Regarding the initial database design found in Figure 31, the SongImages table was considered by the developer to be wholly unnecessary, as a layout file could just be created for each song, and called when required. The lyrics and chords are displayed using TextViews, which are much more legible than the text on an image, and the size of each TextView can be adjusted easily.

Figure 31 – Initial database design



Alongside this, there are four additional attributes in the Users table in Figure 32. The attributes user_lesson_progress, user_beginner_high_score, user_intermediate_high_score, and user_advanced_high_score have been added. In order for a user to test their knowledge once they have marked each lesson in a section as complete, it was decided to create a quiz for the user to try. The three high score attributes are integer fields, and will store the highest score the user has achieved thus far, with the score being set to 0 until the user attempts the quiz. The user_lesson_progress attribute, however, is a text field, which will store the user's lesson progress. Further explanation of the operation of the lesson tracking will be provided in the "Database Implementation" section of this chapter.

Figure 33 – LoggedInID.java class

5.4 Main Java Classes

There are several Java classes that are used in the application that may not be directly related to the features and sections that are available to the user. As well as this, there are also various folders that store resources that are essential to the application's operation. An analysis surrounding the importance of these components shall be undertaken below.

Figure 34 – Once the user signs in, their user ID is passed to LoggedInID class, which stores this integer.

5.4.1 LoggedInID.java

The LoggedInID Java class (Figure 33) was created in order to store the details of the user that is currently signed in. The class contains two static variables – an integer, loginNum, and a boolean loggedIn. When the database is created upon the installation of the app, users are able to create their own accounts and sign in, and, as has previously been established, each user is assigned a unique user ID. The user ID is auto-incremented with each new account, beginning at 2, as the administrator has the user ID of 1. Once the user has signed in, their username and password will be passed as parameters to the method findLoginPerson() in the DatabaseHelper class, which will be discussed in the "Database Implementation" section of this chapter. This method will return the user ID of the user with that specific username and password, with the value of the user ID being assigned to the loginNum variable in LoggedInID (Figure 34).

```
public class LoggedInID {

    //This class will store the login status of the appli
    //logged in. When a user signs out, loginNum set to -
    static int loginNum;
    static boolean loggedIn = false;

    //This default constructor will allow the user ID and
    public LoggedInID(){

    }//Default constructor

    //This constructor takes the person ID and loggedIn s
    public LoggedInID(int personID, boolean loggedInNow){
        loginNum = personID;
        loggedIn = loggedInNow;
    }//LoggedInID
```

```
details = entry.findLoginPerson(username, password);
loggedInID.loginNum = Integer.parseInt(details);
loggedInID.setLoggedIn(true);
```

Alongside the user ID of the signed-in user being stored in this class, the loggedIn boolean is set to true, as a user is now signed in. An example of this class in operation would be when a user attempts to access one of the quizzes in the Lessons section, as can be seen in Figure 35.

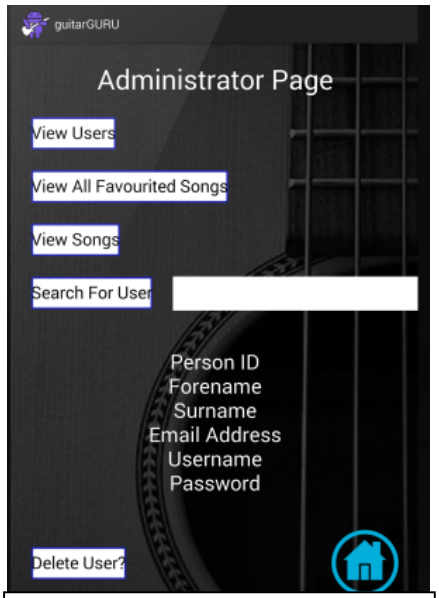
```
if (loggedInID.isLoggedIn()) {

    //Create and open a DB reference
    DatabaseHelper dbHelper = new DatabaseHelper(this);
    try {
        dbHelper.open();

        //Retrieve the user's current lesson progress as a String, pass
        //as a parameter
        lessonNumber = dbHelper.getLessonProgress(loggedInID.loginNum);
```

In order to access the quiz, a user must be signed in. Therefore, an instance of the LoggedInID class is created, and the isLoggedIn() method is called, which will return the loggedIn boolean. If the method returns true, a user is logged in, otherwise no user is logged in. If the former is correct, then the user's current lesson progress must be retrieved from the database. The getLessonProgress() method found in Figure 35 will return the String containing the lesson progress, but it requires the user ID of the user in question. This can be ascertained using the loggedInID instance, and the loginNum can be retrieved and passed as a parameter to this method.

5.4.2 Administrator Section



At an early stage in development, it was decided that the developer must be given a straightforward way to access the database tables, and the data contained within. This feature is necessary for testing purposes, to ensure that all changes made to the database had the desired effect, and to ensure that all interactions with the database functioned correctly. Therefore, upon the creation of the database, an Administrator account is created, with the username and password of this account known only to the developer. Upon entry of these details, the administrator will be directed to the Admin section seen in Figure 36. This section is completely inaccessible to regular users of the app, and they are highly unlikely to even be aware of its existence. The administrator has several options available to them once they access this section – they can view all database tables, search for users, and delete users, should they so desire.

5.4.3 RAW folder

The raw folder, found within the /res directory of the application, stores the media files that are used. Given that, during the design phase, particular emphasis was placed on

gure 37– Creating Objects, allowing the media files to be played.

the use of various forms of multimedia throughout the app, there are a large number of files present within the raw folder, including .mp4, .m4a, .3gp, and .mp3 files. This folder had to be created manually by the developer within the res directory. In order for these media files to be played, the developer had to use the file path associated with the required file, and pass it as a parameter to the object being created, be it a MediaPlayer object, or a VideoView, as in Figure 37.

```
videoViewWritingOwnSongs = (VideoView) findViewById(R.id.videoViewWritingOwnSongs);
Uri uri = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.songwritingvideo);
videoViewWritingOwnSongs.setVideoURI(uri);
```

5.4.4 AndroidManifest.xml

The Android Manifest file is arguably one of the most important for an Android application, providing information that the device needs in order to run the app. There are various components within the manifest that are particularly notable, but none more so than the <activity> elements. An activity can be viewed as a "screen" within the application, so for each new page that is displayed on the screen, an activity has been created for that page. With each new activity created, it must be added to the manifest file. Otherwise, the app will crash due to an ActivityNotFoundException.

```
<activity android:name=".SplashScreen"
    android:label = "guitarGURU"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"></action>
        <category android:name="android.intent.category.LAUNCHER"></category>
    </intent-filter>
</activity>
```

Within an activity element, an <intent-filter> can also be used to specify the tasks that can be undertaken by this activity. As can be seen in Figure 38, an intent filter has been used in relation to the SpashScreen.java class. Specifically, the commands android.intent.action.MAIN and android.intent.category.LAUNCHER ensure that the SplashScreen class is the starting point when the app is launched. Initially, it was decided that this command would be applied to the MainActivity.java class, which would display the main menu immediately when the app launched. However, it was decided that a splash screen would provide a more professional appearance to the application, with the screen displaying the app name, logo, and slogan, as well as a sound file of the developer playing an acoustic guitar riff. This screen will be displayed for five seconds, at which point the main menu will be launched, and the user is free to use the app.

Figure 39 – Fields names for the tables in DatabaseHelper.java class.

5.5 Database Implementation

As has been mentioned previously, the initial design of the database is significantly different when compared with the final design. The main difference revolves around the introduction of two features, namely the ability to track lesson progress, and the quizzes for each section of lessons, which required that the Users table have four additional attributes – a text field to store the lesson progression codes, and three integer fields to store the user's current high score in each of the three quizzes, as can be seen in Figure 32. While the developer had previous experience in working with databases, and writing with SQL, they had no knowledge of developing a SQLite database within an Android framework. Therefore, research was undertaken prior to development in order to ascertain the most effective method of implementing such a database into guitarGURU. Fortunately, a well-renowned YouTube channel contained several videos on this topic, and provided a step-by-step guide to the developer on SQLite database development for Android (thenewboston, 2015). During development, it was necessary to create a separate Java class, named DatabaseHelper.java, that would control all interactions with the database itself, and the data held within it.

5.5.1 DatabaseHelper.java Class

This class contains the methods that will allow for the creation of the database, as well the amendment and modification of the data held in it.

```
public class DatabaseHelper {

    //Initialise the DB and table names, and the DB version
    private static final String DATABASE_NAME = "AppUsers";
    private static final String USER_TABLE_NAME = "Users";
    private static final String SONG_TABLE_NAME = "Songs";
    private static final String USERS_FAVOURITE_SONGS_TABLE_NAME = "FavouritedSongs";
    private static final int DATABASE_VERSION = 1;

    //Initialise the USER_TABLE_NAME field names
    private static  final String USER_ID = "user_id";
    private static final String USER_FORENAME = "user_forename";
    private static final String USER_SURNAME = "user_surname";
    private static final String USER_EMAIL = "user_email";
    private static final String USER_USERNAME = "user_username";
    private static final String USER_PASSWORD = "user_password";
    private static final String USER_LESSON_PROGRESS = "user_lesson_progress";
    private static final String USER_BEGINNER_HIGHSCORE = "user_beginner_high_score";
    private static final String USER_INTERMEDIATE_HIGHSCORE = "user_intermediate_high_score";
    private static final String USER_ADVANCED_HIGHSCORE = "user_advanced_high_score";

    //Initialise the SONG_TABLE_NAME field names
    private static final String SONG_ID = "song_id";
    private static final String SONG_NAME = "song_name";
    private static final String SONG_ARTIST = "song_artist";

    //Initialise the USERS_FAVOURITE_SONGS_TABLE_NAME field names
    private static final String FAVOURITED_USER_ID = "fave_user_id";
    private static final String FAVOURITED_SONG_ID = "fave_song_id";

    //Declare instances of the DBHelper class, Context, and SQLiteDatabase
    private DBHelper ourHelper;
```

Figure 40 – The static inner class DBHelper, which extends SQLiteOpenHelper.

Figure 39 demonstrates that in order to name the database, its tables, and the various field names, they must be stored as static final Strings, which can then be used in SQL queries. For example, regarding the Songs table, the final String SONG_TABLE_NAME stores the table name, while the Strings SONG_ID, SONG_NAME, and SONG_ARTIST are the field names of the three fields in this table. However, in order for the database to be created, it is necessary for the Android class SQLiteOpenHelper to be implemented into the DatabaseHelper class created by the developer. Therefore, it was decided to create an inner class within the DatabaseHelper class, named DBHelper, which would extend the SQLiteOpenHelper class, thus ensuring access to the various methods contained within the extended class (Figure 40).

```
private DBHelper ourHelper;
private final Context ourContext;
private SQLiteDatabase ourDatabase;

//Create a class DBHelper, extending SQLiteOpenHelper
private static class DBHelper extends SQLiteOpenHelper {

    //Thi constructor will takes the context, DB name, and DB version as parameters, and applies
    //them to the DBHelper class
    public DBHelper(Context context) { super(context, DATABASE_NAME, null, DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //Only use when create database. Takes a SQLiteDatabase object as parameter.

        //Execute the following SQL command to create the User table
        db.execSQL("CREATE TABLE " + USER_TABLE_NAME + " (" +
                USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                USER_FORENAME + " TEXT NOT NULL, " + USER_SURNAME + " TEXT NOT NULL, " +
                USER_USERNAME + " TEXT NOT NULL, " + USER_PASSWORD + " TEXT NOT NULL, " +
                USER_EMAIL + " TEXT NOT NULL, " + USER_LESSON_PROGRESS + " TEXT, " + USER_BEGINNER_HIGHSCORE
        + " INTEGER, " + USER_INTERMEDIATE_HIGHSCORE + " INTEGER, " + USER_ADVANCED_HIGHSCORE + " INTEGER);");

        //Execute the following SQL command to create the Song table
        db.execSQL("CREATE TABLE " + SONG_TABLE_NAME + " (" +
                SONG_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                SONG_NAME + " TEXT NOT NULL, " +
                SONG_ARTIST + " TEXT NOT NULL);");

        //The following String will be used to create the Favourited Songs table
        String createFaveSongsTable = "CREATE TABLE " + USERS_FAVOURITE_SONGS_TABLE_NAME +
```

The SQLiteOpenHelper class is “used to create, open, and upgrade databases” (Harwani, 2012). Access to the methods found in this class is essential to the creation of the database. One such example is the onCreate(SQLiteDatabase db) method seen in Figure 40. When the database is created for the first time, this method is called, with the database to be created passed as a parameter. SQL commands can be executed on this database using the .execSQL() method, which accepts a String as a parameter. This String will contain the command that is to be executed. As can be seen in Figure 40, the .execSQL() method is used twice – to create both the Users table, and the Songs table, using the String variables declared and initialised in Figure 39. Further analysis should be undertaken regarding the specifics of the

commands themselves. In relation to the execSQL() String that creates the Users table, the command that ensures that the User ID field is set as the primary key can be seen – 'USER_ID + "INTEGER PRIMARY KEY AUTOINCREMENT"'. Directly stating that this field should be the primary key of this table, as well as the integer used being incremented automatically each time a new user is added, ensures that not only is this field the primary key, but that there will be no duplicate values for the user IDs.

There are several other significant transactions that are carried out within the onCreate() method in the DBHelper class. A notable decision taken by the developer is that when the database is created, the songs are automatically added to the Songs table, as can be seen in Figure 41. This ensures that all of the necessary preparations are already in place for the user to favourite a song; all that is required of them is to press a button, and a search will be made of the Songs table to find the appropriate song, which will already be present in the table. Alongside this, the Administrator account is also created at this stage, as demonstrated in Figure 41. The user will not be aware of the existence of such an account, as it is created independently of their actions, while it provides the developer with a straightforward method of accessing the database tables. The onUpgrade() method, also pictured in Figure 41, will replace an outdated version of the database with a new version, if and when the database itself is updated.

```
    String insertSweetHomeAlabama = "INSERT INTO " + SONG_TABLE_NAME + " (" + SONG_NAME + ", " + SONG_ARTIST + ") " +
            "VALUES ('Sweet Home Alabama', 'Lynyrd Skynyrd');";
    String insertWithOrWithoutYou = "INSERT INTO " + SONG_TABLE_NAME + " (" + SONG_NAME + ", " + SONG_ARTIST + ") " +
            "VALUES ('With Or Without You', 'U2');";

    //Add the song Strings to the Songs table
    db.execSQL(insertBrownEyedGirl);db.execSQL(insertCallMeMaybe);db.execSQL(insertDontStopBelieving);
    db.execSQL(insertGetRhythm);db.execSQL(insertHangMeOhHangMe);db.execSQL(insertHotelCalifornia);
    db.execSQL(insertISeeFire);db.execSQL(insertKnockingOnHeavensDoor);db.execSQL(insertRun);
    db.execSQL(insertSkinnyLove);db.execSQL(insertStuckInTheMiddleWithYou);db.execSQL(insertSunrise);
    db.execSQL(insertSweetChildOfMine);db.execSQL(insertSweetHomeAlabama);db.execSQL(insertWithOrWithoutYou);

    //The following String will be used to create the Administrator account
    String createAdminAccount = "INSERT INTO " + USER_TABLE_NAME + " (" + USER_FORENAME + ", " + USER_SURNAME + ", " +
            USER_EMAIL + ", " + USER_USERNAME + ", " + USER_PASSWORD + ") VALUES ('ADMIN', 'ADMIN', 'ADMIN', 'ADMIN',

    //The createAdminAccount String will be added to the User table
    db.execSQL(createAdminAccount);

}//onCreate

//If the table is upgraded
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + USER_TABLE_NAME);
    db.execSQL("DROP TABLE IF EXISTS " + SONG_TABLE_NAME);
    db.execSQL("DROP TABLE IF EXISTS " + USERS_FAVOURITE_SONGS_TABLE_NAME);

    onCreate(db);
}//onUpgrade
```

Various features implemented into the application make use of the database, and the various associated methods. A discussion regarding the purpose of these features, as well as the manner in which the database was integrated into them, shall be undertaken below.

5.5.2 Account

When the user accesses the Account section of the application, they will be offered the chance to either sign in, or create a new account. Regarding the former option, found in Figure 42, when a user enters their username and password and presses "Login", their details will be compared against the usernames and passwords found in the Users table of the database. The first actions taken by this class are to check whether the administrator is the user that is signing in. For the purposes of security, the administrator password has been covered in Figure 42. However, if the username and password entered match the admin username and password, the user is then taken to the AdminPage.java class. If the administrator is not signing in, further checks are required.

```
DatabaseHelper entry = new DatabaseHelper(Account.this);
entry.open();

Log.i("ACCOUNT", username + password

//If the administrator signs in, open the AdminPage class
if ((username.equals("ADMIN") && (password.equals("[redacted]"))))
    Log.i("Account", "MATCH");
    nullifyVariables();
    startActivity(new Intent(Account.this, AdminPage.class));
    finish();
}//if
```

Figure 43 – findLoginPerson() will return the user ID of the person searched for.

```
if (entry.findLoginPerson(username, password) == null) {
    didItWork = false;
    String error = "Account not found";
    Dialog d = new Dialog(Account.this);
    d.setTitle("Dang it!");
```

```
else {
    loggedInID = new LoggedInID();

    details = entry.findLoginPerson(username, password);
    loggedInID.loginNum = Integer.parseInt(details);
    loggedInID.setLoggedIn(true);

    loggedInID = null;

    nullifyVariables();
    //Open MyAccount section to view user details
    startActivity(new Intent(Account.this, MyAccount.class));
```

```
if ((username.equals("") || (password.equals("")))) {
    properPassword = false;
    properUsername = false;

    if ((properPassword = false) || (properUsername == false)) {
        didItWork = false;
        String error = "Not all fields filled out appropriately!";
```

Two boolean variables, properUsername and properPassword, have been initialised, to provide an explanatory error message should the user attempt to sign in while leaving a field blank. If the username or password fields are "", therefore they are blank, then both properUsername and properPassword are set to false. If either of these booleans is false, then a dialog is displayed, informing the user that "Not all field filled out appropriately". However, if the username or password fields are not left blank, and the data entered does not match the administrator username or password, then the class must access the DatabaseHelper class, by creating an instance of this class, named "entry" (Figure 42). The methods of the DatabaseHelper class then can be accessed; in particular, the findLoginPerson() method (Figure 43).

```
public String findLoginPerson(String username, String password){
    String [] columns = new String [] { USER_ID, USER_FORENAME, USER_SURNAME, USER_EMAIL,
            USER_USERNAME, USER_PASSWORD};
    //This query is based on the username and password passed as parameters
    Cursor c = ourDatabase.query(USER_TABLE_NAME, columns, USER_USERNAME + "= '" + username + "' AND " + USER_PASSWORD
            + " = '" + password + "'", null, null, null, null);
    //If c is not null, both username and password were found in the User table. Therefore return the userID.
    //Otherwise return null.
    if (c != null){
        c.moveToFirst();
        String details = c.getString(0);
        return details;
    }//if
    return null;
```

This method will take the username and password entered by the user as String parameters. The fields within the table that are to be searched through will be added to a String array named columns. A query will then be executed against the database, which will search all tuples within the specified table, to return all of the fields found in the columns array where the username and password of that tuple match those of the Strings passed as parameters. If the user has been found, their details will be added to a Cursor, and the String details will be assigned the value of the data at position 0, which, as specified in the columns array, is the user ID of the user, which will be returned. Otherwise, the Cursor is null, and null will be returned. If this is the case, the user will be informed that no account has been found.

However, if the user ID of the user has been found, and is returned by the findLoginPerson() method, the value returned is assigned to a String value named details. At this point, a user has now signed in, and the signed-in user's details must be stored in the LoggedInID class. By creating an instance of this class, the loginNum variable can be assigned the value of the user ID by parsing the String to an Integer, allowing the user ID to be stored in this form. The loggedIn boolean in this class is also set to true, using the setLoggedIn() method. The user is then directed to the My Account section of the app, where they can view their details and their favourited songs, as well as edit their details.

In relation to creating a new account, a similar process is followed. Five booleans are created to ensure that the forename, surname, email, username and password fields are not left blank. If one of these fields is blank, the appropriate boolean will be set to false, and a dialog will appear informing the user of this error when they attempt to create their account. However, if all fields are not blank, a check will be made to ensure that the username entered is not already taken. This is achieved, again, by creating an instance of the DatabaseHelper class, again named entry, and accessing the methods within this class. In this scenario, the checkUsername() method is required (Figure 44).

```
public String checkUsername(String username) throws SQLException{
    String [] columns = new String [] { USER_USERNAME };
    Cursor c = ourDatabase.query(USER_TABLE_NAME, columns, USER_USERNAME + " = '" + username + "'",
            null, null, null, null);
    //If c is not null, the username is found and returned. If exists, user told to enter new username
    if ((c != null) && (c.moveToFirst())){
        c.moveToFirst();
        String found = c.getString(0);
        return found;
    }//if
    //Otherwise return null.
    return null;
}//checkUsername
```

As with the findLoginPerson method, the username entered by the user is passed as a parameter, and forms part of a query, which will search the tuples of the Users table to check whether the username passed as a parameter matches a username in the table. If no matches are found, the value null is returned, otherwise the username is returned. This value is then assigned to a String variable in NewAccount.java. If this variable is null, then the username is available, and the account can be created.

Otherwise, an error message will be displayed informing the user that the username has already been taken. If the username is free, however, then the details entered are passed to the createEntry() method in the DatabaseHelper class. The forename, surname, email, username and password are passed to this method, along with four zeros, which represent the lesson progress, and the three high scores for the quizzes, which are yet to be altered by the user.

```
public long createEntry(String name, String surname, String email, String userName, String password,
                        long lesson, long beginnerHS, long intermediateHS, long advancedHS) {
    //The ContentValues will hold the information to be added to the DB
    ContentValues cv = new ContentValues();
    //Insert the parameters into the appropriate fields in the table
    cv.put(USER_FORENAME, name);
    cv.put(USER_SURNAME, surname);
    cv.put(USER_EMAIL, email);
    cv.put(USER_USERNAME, userName);
    cv.put(USER_PASSWORD, password);
    cv.put(USER_LESSON_PROGRESS, lesson);
    cv.put(USER_BEGINNER_HIGHSCORE, beginnerHS);
    cv.put(USER_INTERMEDIATE_HIGHSCORE, intermediateHS);
    cv.put(USER_ADVANCED_HIGHSCORE, advancedHS);
    //Insert the ContentValues into the User table
    return ourDatabase.insert(USER_TABLE_NAME, null, cv);
}//createE
```

.

Figure 45 demonstrates that the variables passed as the parameters will be added to a ContentValues object named cv, which is created to hold the set of values entered. Using the put() method associated with the ContentValues object, the name of the field and its associated variable can be added to the object together. Once all of the details have been added to the ContentValues object, they are inserted into the Users table, using the insert() method. It is at this point that the getUserID() method in DatabaseHelper is used to return the user ID of the new user, by passing the username as a parameter. This method will search through the Users table, and compare the username of each tuple to the parameter, and return the user ID if a match is found, or null. Once the user ID has been returned, it is parsed to an Integer, and the loginNum and loggedIn variables in LoggedInID.java are updated accordingly. The user is then directed to the My Account section of the app. At this point, should a user decide to sign out, the loginNum is set to -1, and the loggedIn boolean is set to false, thus ensuring that no matches can be found in the database, indicating that no user is currently signed in. This is particularly important, given the frequent use of drop-down menus throughout the app. When accessing the Account section through the drop-down menus, the class which will be launched differs, depending on the value returned by the isLoggedIn() method in LoggedInID.java. If the user is signed in, and the value returned by isLoggedIn() is true, they will be directed to the MyAccount.java class, otherwise they will be taken to the Account.java class, which will allow them to either sign in or create an account.

5.5.3 Favouriting Songs

During the early stages of planning and design, it was decided that one of the key functions of the app would be to enable users to "favourite" certain songs from the available list, allowing them to save them for later practice, as well as providing easy and convenient access to their favourite songs from one place. Figure 32 demonstrates how the favourited songs would be stored – the user ID of the user that is favouriting the song, and the song ID of the favourited song in question would both be foreign keys in the FavouritedSongs table, but together would form a composite primary key. By making both attributes a composite primary key, the possibility of duplicate entries is eliminated, thus no song can be favourited by the same user more than once.

Figure 46 displays the buttons on each song screen that will allow a user to both favourite and un-favourite a song. The heart represents the "Favourite" button, given its clear association with love and affection, while the "Un-favourite" button simply features a textual description, in order to prevent any measure of ambiguity. When the user presses the favourite button on a certain song, the findSongID() method in DatabaseHelper will take in the song name and artist as parameters, and will search the Songs table in the database. If there is a match, the songID will be returned by this method, otherwise null will be returned.

```
if ((entry.findSongID(songSelected, artistSelected) != null) && (personID > 0)) {
    songID = Integer.parseInt(entry.findSongID(songSelected, artistSelected));
    entry.addFaveSong(personID, songID);
}//if
else {
    if (personID <= 0) {
        didItWork = false;
        Dialog d = new Dialog(SongSelectedDisplayed.this);
        d.setTitle("Oh dear!");
```

.

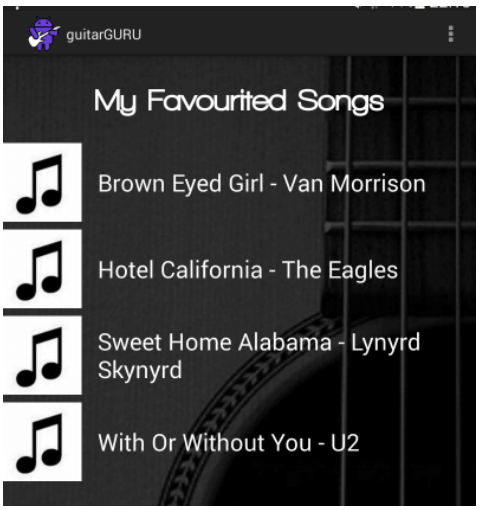
As Figure 47 demonstrates, if the value returned is not null and the person ID is greater than 0, the song ID will be parsed to an Integer and stored as the variable songID. II must be noted that the purpose of the "personID > 0" condition is that if the personID, retrieved from the LoggedInID class, is less than 0, no user is logged in. However, if it is greater than 0, a user must be logged in. As it is necessary for a user to be logged in to favourite a song, this condition is essential. Figure 47 demonstrates that if this condition is not met, an error message will inform the user that they must log in. However, if the songID is not null, and the personID is greater than 0, then the user ID and song ID are passed to the addFaveSong() method in the DatabaseHelper class.

Figure 48 – The various other dialogs that can be displayed when favouriting a song.

Using the SQL command "INSERT INTO " + USERS_FAVOURITE_SONGS_TABLE + ")" FAVOURITED_USER_ID + ", " + FAVOURITED_SONG_ID + ") VALUES (" + personID + ", " + songID + ");"; , the personID of the user and the songID of the song are added to the fields FAVOURITED_USER_ID and FAVOURITED_SONG_ID respectively in the FavouritedSongs table. However, if any exceptions are caught by this command, an error has occurred, and the didItWork boolean, which monitors whether the transaction was a success, is set to false, as can be seen in Figure 48. If any exceptions occur, it is because the song has already been favourited by this user, and duplicate primary keys are prohibited in this table. Therefore, a dialog informing the user of this fact is displayed. However, if no errors occur, the song has been favourited, and the didItWork boolean remains true. Therefore, a dialog will be displayed informing the user that this command was successful.

```
catch (Exception e) {
    didItWork = false;
    String error = e.toString();
    Dialog d = new Dialog(SongSelectedDisplayed.this);
    d.setTitle("Dang it!");
    TextView tv = new TextView(SongSelectedDisplayed.this);
    tv.setText("You have already liked this song!");
    d.setContentView(tv);
    d.show();
}//catch
finally {
    if (didItWork) {
        Dialog d = new Dialog(SongSelectedDisplayed.this);
        d.setTitle("Heck yeah!");
        TextView tv = new TextView(SongSelectedDisplayed.this);
        tv.setText("Success!");
```

The effect of favouriting a song can be seen in the FavouritedSongs table, accessed from the Admin section of the application, and the "My Favourited Songs" section, accessed through the My Account section, as can be seen in Figure 49.



In relation to the "View All Favourited Songs" screen in Figure 49, the table itself contains only the user ID and song ID, but has been inner joined with the Users and Songs tables to display the username, song title, and song artist. Each song that was favourited by the user has been added to the FavouritedSongs table, and therefore can be found in the "My Favourited Songs" section in Figure 49, which is where the user can access their favourited songs. This section can also be accessed directly from the SongSelectedDisplayed.java class, by pressing the favourite button for several seconds, known as a long click.

```
if (data.contains("Brown Eyed Girl")){
    brownEyedGirl.setVisibility(View.VISIBLE);
    brownEyedGirlTR.setVisibility(View.VISIBLE);
    brownEyedGirl.setOnClickListener((v) -> {
            song = "Brown Eyed Girl";
            artist = "Van Morrison";
            Intent i = new Intent(ViewUsersFavedSongs.this, SongSelectedDisplayed.class);
            i.putExtra("SONG", song);
            nullifyVariables();
            startActivity(i);
            finish();
    });//brownEyedGirl
}//if
```

Regarding this section, buttons that connect to every song available in the application are created, and are initially set to be "gone". The user's favourited songs will be retrieved from the FavouritedSongs table, using the getFaveSongsData() method, with the user ID being passed as a parameter. This method will return, among other things, the title of the song. Therefore, searches are made regarding whether a specific song title can be found in the returned String, and if so, the user has favourited this song. The visibility of the button connected to this song is set to View.VISIBLE, as can be seen in Figures 49 and 50.

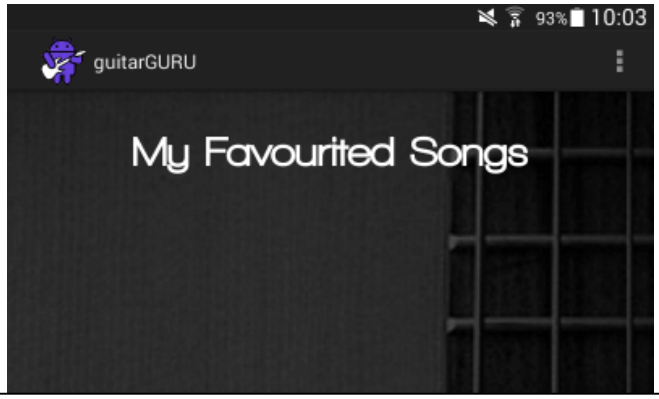
If a song is un-favourited by the user, it is removed from both of the lists seen in Figure 49. The code depicted in Figure 51 can be found in the SongSelectedDisplayed.java class, and controls the removal of songs from the user's favourites. Using the findSongID() method, the song ID can be ascertained by passing in the song title and the artist as parameters. This song ID, and the user ID, are then passed as parameters to the checkIfFaved() method in the DatabaseHelper class. The combination of the song ID and the user ID are then compared to each tuple in the FavouritedSongs table. If the user has favourited this song already, this combination will be present in the table, thus the String "FOUND" will be returned. Otherwise, this song has not been favourited by the user, and null is returned. Therefore, if "FOUND" is returned, this song can be removed from the user's favourited list, using the removeSongsFromFaves() method. This method will execute a SQL command to delete the tuple from the FavouritedSongs table where the user ID and song ID match the values passed as parameters.

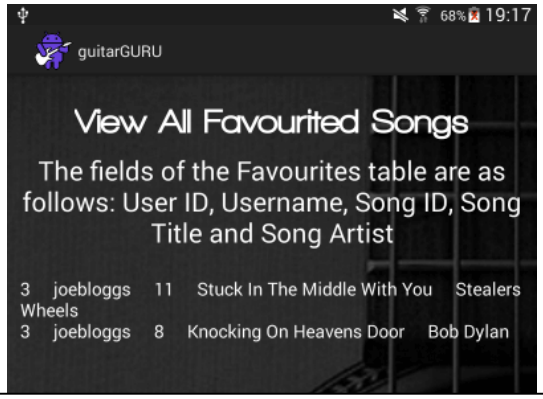
Figure 51 – Un-favouriting a song in the SongSelectedDisplayed.java class


```
songRemoveID = Integer.parseInt(entry.findSongID(songSelected, artistSelected));

//If song found, remove from table
if (entry.checkIfFaved(personID, songRemoveID) == "FOUND") {
    entry.removeSongsFromFaves(personID, songRemoveID);
    didItWorkRemove = true;
}//if
else {
    didItWorkRemove = false;
    Dialog d = new Dialog(SongSelectedDisplayed.this);
    d.setTitle("This song !");
    TextView tv = new TextView(SongSelectedDisplayed.this);
    tv.setText("This song is not in your favourites");
```

The effects of removing songs from the FavouritedSongs table can be seen in the "View All Favourited Songs" and "My Favourited Songs" screens, as seen in Figure 52. All of the songs that had been favourited in Figure 49 have been un-favourited, therefore the My Favourited Songs section is blank, and all of the tuples associated with user ID 2 have been removed. However, another user, with the user ID of 3, has favourited several songs. These songs have been added to the FavouritedSongs table, as can be seen in Figure 52.





5.5.4 Tracking Lesson Progress

In order to ensure that the user has the ability to progress through the lessons at their own pace, as well as allowing them to chart their development, a system has been implemented into the app that allows the user to track their lesson progress. This system becomes clear to the user through a system of "tick" symbols that are present beside each lesson that is marked as complete, while these ticks are absent from lessons that are yet to be completed.

It is important to point out that this feature, as well as access to the quizzes, is only available to users that have created an account, and are currently signed in when viewing the lessons. It was decided that users without an account will still be able to access the lessons, as, in the developer's opinion, it would

Figure 53 – When a user is not signed in, they cannot see the 'tick' button to complete lessons, and any tick images that have been saved are set to 'gone' unless they are displayed beside the titles of all completed lessons.

be wholly unnecessary to alienate users from the education contained in the app simply because they are unwilling to create an account, however there should be an incentive to do. The various account user-specific features, such as lesson tracking, the quizzes, and favouriting songs are incentives to create an account, and act as a reward to account holders at the same time.

Therefore, when a user is not signed in, they do not have the option to mark a lesson as complete, nor can they see the ticks that signify a lesson as being complete, as can be seen in Figures 53 and 54.

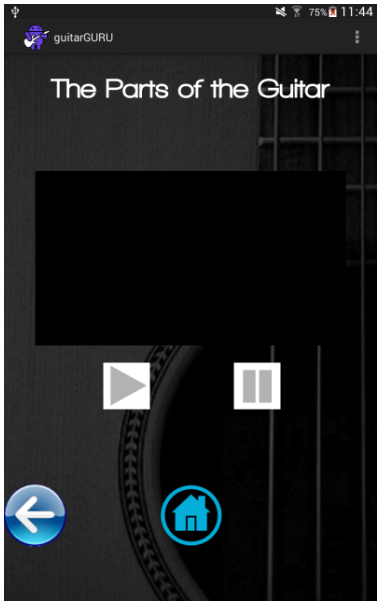


Figure 54 – When a user is signed in, they can mark a lesson as complete by pressing the tick, and an image of the same tick is displayed beside the titles of all completed lessons.

gure 55 – Code fragments used on each Lesson screen to add the lesson code to the lesson progress.

When a user that has signed in has accessed one of the lessons, and wishes to mark it as complete, they press the "tick" button. The code in Figure 55 is then used to add this lesson to the user's current lesson progress.

```
loggedInID = new LoggedInID();
if (loggedInID.isLoggedIn()) {

    //Make the complete button visible
    changingStringsLessonComplete.setVisibility(View.VISIBLE);
```

```
int added = dbHelper.setLessonProgress(loggedInID.loginNum, "4");
dbHelper.close(); dbHelper = null;

//If added is -1, the lesson is already complete
if (added == -1){
    Toast toast = Toast.makeText(getApplicationContext(), "Lesson already completed!", Toast.LENGTH_SHORT);
    toast.show();  }//if

//If added is not -1, it can only be 1, therefore the lesson is marked as complete
else{
    Toast toast = Toast.makeText(getApplicationContext(), "Lesson Complete!", Toast.LENGTH_SHORT);
    toast.show();  }//else
```

As can be seen in Figure 55, in order for the lesson to be marked as complete, the loggedIn boolean in the LoggedInID class must be set to true, indicating that a user is in fact logged in. At this point, the button to complete the lesson is changed to visible, allowing the user to press it. Should a user choose to do so, the lesson progress, stored in the Users table in the database, must be updated using an instance of the DatabaseHelper class, in particular the setLessonProgress() method. This method takes two parameters – the loginNum value from LoggedInID, which is the user's ID, and the unique lesson code associated with that specific lesson. In Figure 55, the lesson code for the lesson in question is "4". Each lesson has its own unique code, and when a lesson is completed, it is added to the String that stores all of the completed codes, as shall be demonstrated later. The setLessonProgress() method will return an integer value depending on the success or failure of the transaction, as can be seen in Figure 56. A check is initially made regarding whether the lesson has already been completed by the user using the getLessonProgress() method, which will return the String of completed lesson codes where the user ID of a tuple in the Users table matches the user ID passed as a parameter. If the String returned contains the lesson code already, the value -1 is returned. Otherwise, a ContentValues object is used to update the lesson progress field of the tuple featuring the user ID passed as a parameter by adding the lesson code to the String returned by the getLessonProgress() method. This will add the new code to the already completed codes. A value of 1 is then returned, and a Toast is displayed informing the user that the lesson is now completed.

igure 57 – If the String returned
·om the getLessonProgress()
ıethod contains a lesson code, the
ompleted tick for that lesson is set
› Visible.

```
public int setLessonProgress(long userID, String lesson) throws SQLException {
    int returnValue;
    ContentValues cvUpdate = new ContentValues();
    //If the user's current lesson progress doesn't contain the code lesson passed as a
    //parameter, this lesson is only newly completed. This code will be added to user's
    //lesson progress, and 1 is returned. Otherwise -1 is returned
    if (!getLessonProgress(userID).contains(lesson)) {
        cvUpdate.put(USER_LESSON_PROGRESS, getLessonProgress(userID) + lesson);
        ourDatabase.update(USER_TABLE_NAME, cvUpdate, USER_ID + " = " + userID, null);
        returnValue = 1; }//if
    else {
        returnValue = -1; }//else
    return returnValue;
```

Once a lesson has been completed, the user must be able to easily ascertain which lessons they have completed, and which they have yet to complete. The String of completed lesson codes returned by the getLessonProgress() method allows for these lessons to be clearly highlighted. The code pictured in Figure 57 can be found in the BeginnerVideos.java class, and is the class associated with the Beginner Lessons screenshot also found in Figure 57. Aside from the obvious differences in the lesson codes, the operation of this code in Figure 57 is the same for Intermediate and Advanced Lessons. As has already been established, the ImageViews representing the completed ticks for each lesson are initially set to be "gone". Using the getLessonProgress() method, the String of lesson codes is returned, and assigned to the String variable lessonNumber. The lessons that have been completed so far by the user are "Parts Of Your Instrument", and "Changing Your Strings", which have the lesson codes of "2" and "4" respectively. If the String of codes contain any of the codes associated with a lesson, this lesson has been completed by the user, and thus the visibility of the completed tick is set to View.VISIBLE, indicating to the user that they have completed the lesson. When every lesson in this section has been marked as complete, an ImageView of the completed tick beside the "Beginner" button in the Lessons section is set to "visible", informing the user that they have completed this section.

```
if (lessonNumber.contains("1")) {
    tuningTick.setVisibility(View.VISIBLE);
}//if
//If lessonNumber contains 2, make the completed tick
if (lessonNumber.contains("2")) {
    partsTick.setVisibility(View.VISIBLE);
}//if
//If lessonNumber contains 3, make the completed tick
if (lessonNumber.contains("3")) {
    careTick.setVisibility(View.VISIBLE);
}//if
//If lessonNumber contains 4, make the completed tick
if (lessonNumber.contains("4")) {
    changingStringsTick.setVisibility(View.VISIBLE);
}//if
//If lessonNumber contains 5, make the completed tick
if (lessonNumber.contains("5")) {
    fretboardTick.setVisibility(View.VISIBLE);
}//if
//If lessonNumber contains 6, make the completed tick
if (lessonNumber.contains("6")) {
    holdingTick.setVisibility(View.VISIBLE);
```

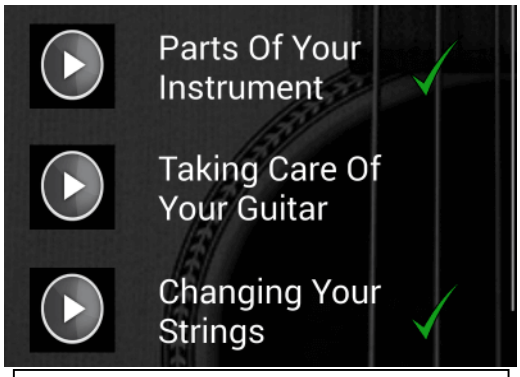
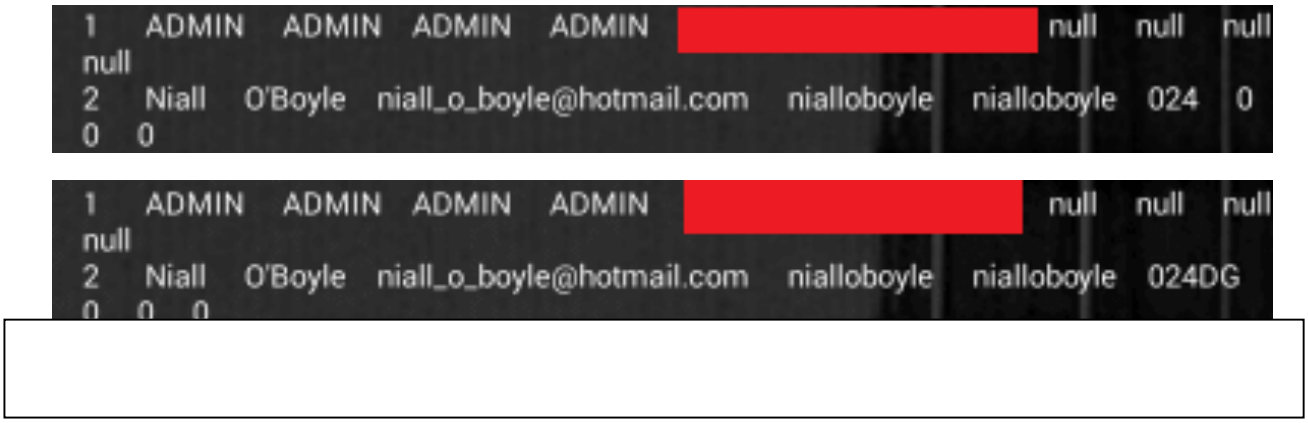


Figure 59 – The completed ticks for lessons that were completed are now visible.

As the user's lesson progress String is stored in the User's table in the database, it can be viewed by the Administrator, and the effects on the String can be seen as more lessons are marked as complete. As the lesson progress is initialised as containing only "0" at the time that the database is created, and Figure 57 makes it clear that the lessons with codes "2" and "4" have been completed, the String should only contain the characters "024". Figure 58 confirms that this is indeed the case. As the user completes more lessons, this String will contain additional codes. In order to demonstrate this, the user has completed two more lessons – Chord Transitions, from the Intermediate Lessons section, and Reading Tabs, from the Advanced Lessons section. These lessons have the unique lesson codes of "D" and "G" respectively. Therefore, when viewing the Users table data through the Admin section, Figure 58 also clearly shows that the codes "D" and "G" have been added to the lesson progress String, and Figure 59 shows that these lessons are now marked as complete.



5.5.5 Quizzes

As has been mentioned previously, the developer decided that a good way to allow the user to test their recently acquired knowledge would be to create a quiz, which would feature 10 questions. There is one quiz for each of the three sections – Beginner, Intermediate, and Advanced – with the questions appropriate for each specific difficulty level. Alongside this, three integer attributes have been created in the Users table in the database, in order for the high scores for these three quizzes to be stored. The process for accessing the quiz, and for storing the high scores, is the same for all three sections. Therefore, it would be wholly unnecessary to discuss the operation of the code of all three sections; rather, an analysis of the Beginner Lessons quiz shall be undertaken below.

```
if (loggedInID.isLoggedIn()) {
    //If the lessonNumber contains the code for every lesson in this section
    if (((lessonNumber.contains("1")) && (lessonNumber.contains("2")) && (lessonNumber.contains("3")) &&
            (lessonNumber.contains("4")) && (lessonNumber.contains("5")) && (lessonNumber.contains("6")))) {
        nullifyVariables();
        //Open the quiz
        startActivity(new Intent(BeginnerVideos.this, BeginnerLessonsQuiz.class));
        finish();
    }//if
    else {
        //Display a dialog telling the user to complete all lessons
        Dialog d = new Dialog(BeginnerVideos.this);
        d.setTitle("Sorry");
        TextView tv = new TextView(BeginnerVideos.this);
        tv.setText("You must complete all lessons before trying the quiz!");
```

Figure 60 – The code that controls access to the quiz in BeginnerVideos.java.

Figure 60 highlights the manner in which access to the quiz for each section is restricted. The quizzes are only available to users that have signed in, therefore the first check involves the loggedIn boolean found in the LoggedInID.java class. If the value returned is false, no user is signed in, and a dialog is displayed informing the user that they must sign in. Otherwise, there is a user signed in. Using the lessonNumber variable, which has been assigned the value of the lesson progress String as returned by the getLessonProgress() method, it must be ascertained whether every lesson in the section has been completed. In order to access the quiz, every lesson in the section must be marked as complete. In Figure 60, the "if" statement requires that the lessonNumber variable contains "1", "2", "3", "4", "5", and "6", which are the codes for the six lessons in this section. If the String does not contain all of these codes, a dialog is displayed, instructing the user that they must complete every lesson in the section before they can attempt the quiz. Otherwise, the quiz is launched.

Each quiz contains ten questions, and there are usually four answer options available for selection, with some exceptions, namely true or false questions. If the user selects the correct option, the background of this button is changed to green, and the integer storing the score is incremented. However, if the wrong answer is selected, the background of the selected button is changed to red, with the correct answer background being set to green (Figure 61). After any selection, the buttons are disabled until the next question is displayed. When the "Next" button is pressed, the question and answers change, and the backgrounds of the buttons are reset to white. Sometimes an image is required for certain questions, with its visibility set to "gone" when it is not needed, and "visible" when it is.

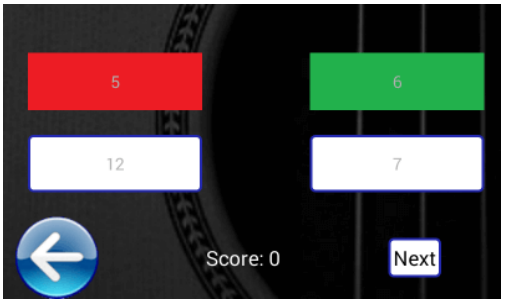
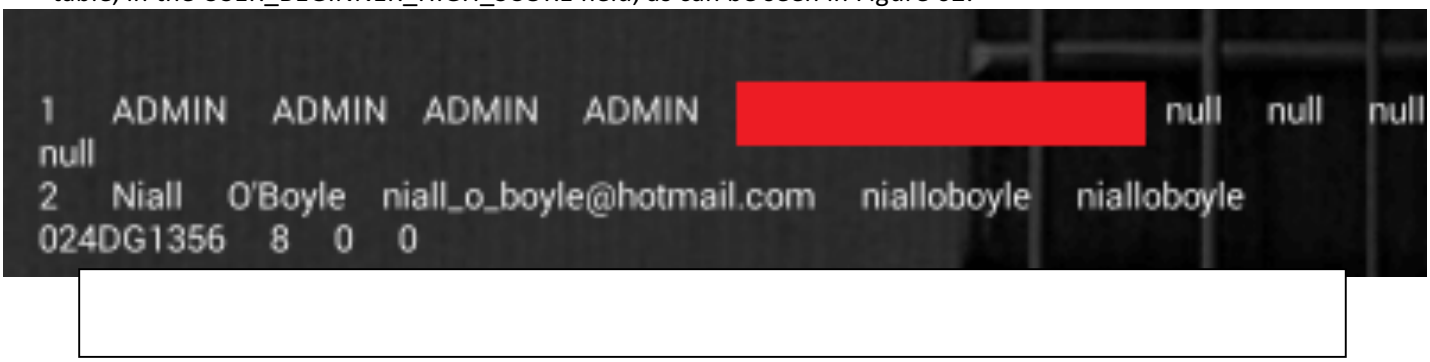


Figure 61 – When an incorrect option is selected, is it red, and the correct answer is green

When the user has completed the quiz, the current score will be compared to their previous high score. After the user's first attempt, a score of 8/10 was achieved. This score was then stored in the Users table, in the USER_BEGINNER_HIGH_SCORE field, as can be seen in Figure 62.



The user's high score for the beginner quiz is now 8, and this score will only be overwritten should a higher score be attained. On a subsequent attempt, a score of 10 was achieved. Figure 63 displays the code from the BeginnerLessonsQuiz.java class that controls the setting and retrieving of high scores.

```
if (questionNumber == -1) {
    //Create a DB reference
    DatabaseHelper dbH = new DatabaseHelper(BeginnerLessonsQuiz.this);
    try {
        dbH.open();
        //Retrieve the user's current high score from the DB
        long userHighScore = dbH.getUserBeginnerHighscore(loggedInID.loginNum);
        //If the current score is higher than the user's high score
        if (userHighScore < score) {
            //Set the current score to the user's new high score
            dbH.setBeginnerQuizHighScore(loggedInID.loginNum, score);
            //Display an alert, with the score and the previous high score
            AlertDialog.Builder builder = new AlertDialog.Builder(BeginnerLessonsQuiz.this);
            builder.setTitle("Well done!");
            builder.setMessage("Congratulations, your score was " + score + "! Your previous " +
                    "high score was " + userHighScore + "!");
```

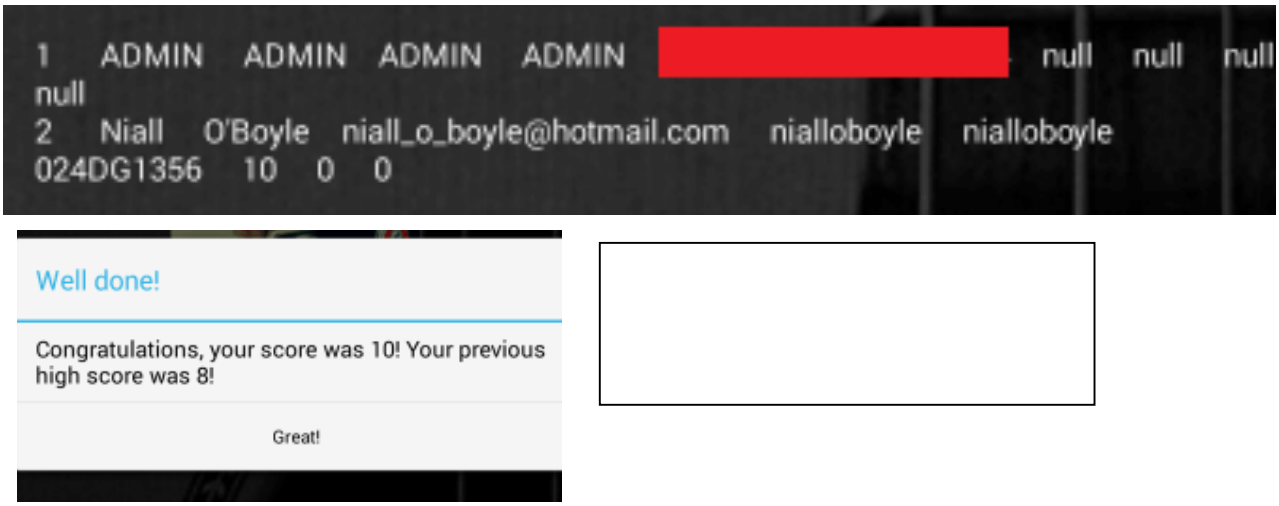
This code is only required when all ten questions have been answered. At this point, the questionNumber integer is set to -1, and this code is utilised. Using an instance of the DatabaseHelper class, the getUserBeginnerHighScore() method will return the value from the USER_BEGINNER_HIGH_SCORE field where the user ID of the tuple matches the user ID passed as a parameter. If this value is greater than the current score achieved by the user, a dialog is displayed informing the user of the current score, and their high score. However, if the new score is greater than the high score, the setBeginnerQuizHighScore() method is called (Figure 64), with the user ID and the new high score passed as parameters. Using a ContentValues object, the tuple where the USER_ID field

igure 65 – The new high score is
isplayed, and can be seen in the
sers table.

matches the user ID passed as a parameter is updated with the new high score. A dialog is then displayed, informing the user of their new high score, and their previous high score (Figure 65).

```
public long setBeginnerQuizHighScore(long userID, long highScore) throws SQLException {
    long returnValue;
    ContentValues cvUpdate = new ContentValues();
    //If the highScore passed as a parameter is higher than the user's current high score
    //update this field with the user's new high score and return 1. Otherwise return -1.
    if (getUserBeginnerHighscore(userID) < highScore) {
        cvUpdate.put(USER_BEGINNER_HIGHSCORE, highScore);
        ourDatabase.update(USER_TABLE_NAME, cvUpdate, USER_ID + " = " + userID, null);
        returnValue = 1;  }//if
    else {  returnValue = -1;  }//else
    return returnValue;
```

4 – The setBeginnerHighScore() method, which overwrites the previous high score with the new one



Also in Figure 65, the updated high score can be seen in the USER_BEGINNER_HIGH_SCORE field. As the score of 10 was achieved, this was greater than the previous high score, and thus has now been stored as the user's current high score. Regardless of the value stored in this field, it will remain this value until a higher score is achieved, at which point the higher score will overwrite the value.

5.6 Implementation of Other Features

5.6.1 Chords

Chords are an essential part of learning to play the guitar; it would be impossible to play without knowledge of how to play chords. In terms of the structure of the Chords section, three Java classes are used – Chords.java, ChordsKeySelected.java, and ChordsDisplaySelected.java. The implementation of

these three classes, and the manner in which they interact with each other, shall be analysed separately below.

The Chords class displays the main keys that a beginner guitar player will encounter. It was decided that this would be the most appropriate method through which to display the various useful chords, rather than just a list that would be rather long. Not only is separating the chords into keys more efficient and user-friendly, it also educates the user regarding the different types of keys, and the chords that are contained within each key. There are nine keys available for selection by the user. When the user selects a key, the ChordsKeySelected.java class is launched. However, this class uses only one layout file, with the various chords available for selection changing, depending on the key selected by the user. In order for the appropriate chords, and title of the screen, to be set, the ChordsKeySelected class must know what key the user has selected. This can be achieved using an Intent (Figure 66).

```
c.setOnClickListener((v) → {
        Intent i = new Intent(Chords.this, ChordsKeySelected.class);
        i.putExtra("KEY", "C");
        nullifyVariables();
        startActivity(i);
        finish();
});//c
```

In order to start a new Activity, an Intent must be declared, which specifies the current class, and the class to be launched, as can be seen in Figure 66. However, this Intent can also contain additional data, based on an instance of the Bundle class. Using the Intent method putExtra(), the key selected can be transferred to the ChordsKeySelected class. Any extra that is passed requires a key that is to be associated with the data. In this case the String “KEY”, and the value itself, which is passed as a String, are added to the Intent. As shall be discussed below, this data can be retrieved from the Intent.

Once a key has been selected, and the ChordsKeySelected class has been launched, the key selected must be ascertained by the class before any of the layout features can be displayed. This can be achieved using a Bundle object, named extras, which will be assigned the value of the extras associated with the Intent that launched the activity. Using the Bundle method getString(), the key associated with the data is used to retrieve the associated data. Therefore, using the key “KEY”, the String value of the key selected by the user is retrieved, and assigned to the String variable keySelected.

Once the key selected by the user has been ascertained, the various features of the layout file can be appropriately initialised. For example, once the keySelected variable has been assigned a value, the title at the top of the screen is set to include the key. In terms of the chords available for selection within this key, Figures 67 and 68 demonstrate that, depending on the value of keySelected, the text on the buttons, and their functions, differ. Figure 67, for example, demonstrates what would happen if the key selected by the user was the key of C. The text on the seven buttons would be set to the names of the chords in this key, and the String passed as an extra to the Intent launching the

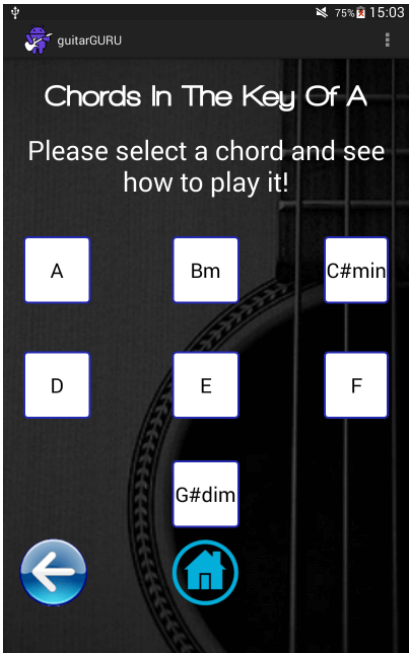
ure 68 – Despite
ng the same layout
, the title and
ttons change
pending on the key
ected by the user.

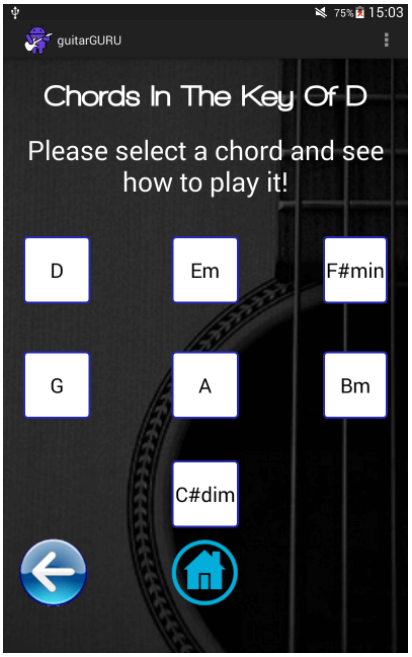
ChordsDisplaySelected.java class will include both the name of the key, and the name of the chord, with the key associated with the data being set to "KEYCHORD". For example, if the user selects the key of D, and then selects the BMinor chord, the data passed would be "DKEY BMCHORD", while selecting the C Chord from the key of G would be "GKEY CCHORD". Using the same method as the ChordsKeySelected.java class, the ChordsDisplaySelected.java class will retrieve the key and chord selected by the user, and will display the appropriate chord diagram, and video.

Figure 67 – The buttons in ChordsKeySelected.java depend on the value of keySelected.

```
cChord.setText("C"); dmChord.setText("Dm"); emChord.setText("Em"); gChord.setText("F");
fChord.setText("G"); amChord.setText("Am"); bdimChord.setText("Bdim");

//Depending on the button pressed, the chordSelected String will be assigned a different
cChord.setOnClickListener((v) -> {
        Intent i = new Intent(ChordsKeySelected.this, ChordsDisplaySelected.class);
        i.putExtra("KEYCHORD", "CKEY CCHORD");
        nullifyVariables();
        startActivity(i);
        finish();
```





The ChordsDisplaySelected.java class again uses only one layout file, chordsdisplayselected.xml, which contains a VideoView, an ImageView, a TextView title, and several TextViews representing the strings, all of which depend on the value of the key and chord selected by the user. The key and chord are retrieved by transferring the data with the Intent launching the activity. This data is assigned to the String variable chordSelected. As this String contains both the key and the chord, it is necessary to search for the key that is selected. Methods have been created to narrow the search for the chord depending on the key selected (Figure 69). This was deemed to be much more efficient than simply

ure 71 – Using the same layout
, but the VideoView and all
tViews are different.

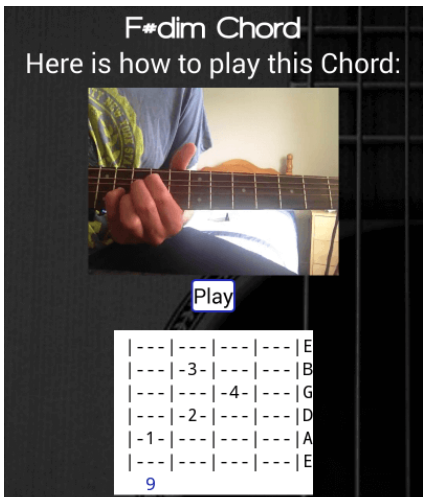
Figure 69 – There are various methods available for each key that could be selected

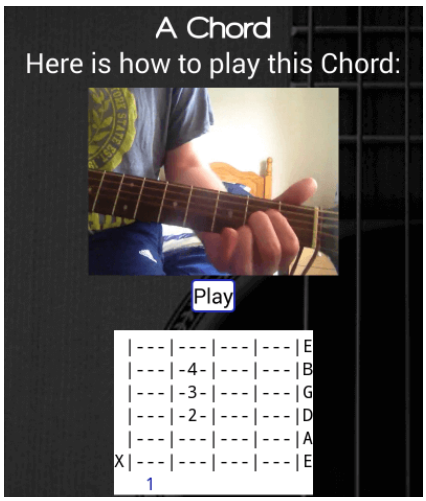
having a list of all the chords that could be selected, which could lead to a large number of options being searched through before the correct chord is found.

```
if (chordSelected.contains("CKEY")) {
    keyOfC();
    key = "C";
}//if
else {
    if (chordSelected.contains("DKEY")) {
        keyOfD();
        key = "D";
    }//if
    else {
        if (chordSelected.contains("EKEY")) {
            keyOfE();
            key = "E";
        }//if
```

Once the key has been ascertained, the appropriate method in Figure 69 is accessed. For example, Figure 70 demonstrates the keyOfC() method, which, if the key selected is C, will search the chordSelected variable for the selected chord, as the variable contains the chord as well as the key. Figure 70 highlights that, depending on the chord found in chordSelected, the TextViews representing each String change, showing the user how to play that chord, as do the title of the page, and the file-path of the VideoView. This process is the same for all of the keys, and all of the chords. Figure 71 shows that, despite the same layout file being used, the title, TextViews, and VideoViews all change depending on the chord selected.

```
public void keyOfC() {
    //Depending on the chord that was selected by the user, the title, diagram, and video will be different
    if (chordSelected.contains("CCHORD")) {
        title.setText("C Chord");
        fretNumberLine.setText("   1");lowELine.setText("X|---|---|---|---|E");aLine.setText(" |---|---|-3-|---|A");
        dLine.setText(" |---|-2-|---|---|D");gLine.setText(" |---|---|---|---|G");bLine.setText(" |-1-|---|---|---|B");
        highELine.setText(" |---|---|---|---|E");
        Uri uri = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.cmajorchordvideo);
        chordVideo.setVideoURI(uri);
    }//if
```





5.6.2 Scales

The Scales section is quite similar to the Chords section in a number of respects, especially in relation to overall structure, and the use of Intent extras to pass data. Three Java classes are used in this section of the app – Scales.java, ScalesOptions.java, and ScaleDiagrams.java – all of which use one layout file each. Like the Chords section, the options available to the user in the ScalesOptions and ScaleDiagrams classes depend very much on the previous choices made by the user. The operation of these three classes will be briefly discussed below.

Regarding the Scales.java class, the scales are divided into common keys that the user may encounter, allowing them to learn how to play the scales related directly to these keys. Depending on the key selected by the user, the String passed through the Intent is different, as can be seen in Figure 72. The Intent itself will launch the ScalesOptions class, which will retrieve the data passed through the putExtra() method.

```
scalesC.setOnClickListener((v) → {
        nullifyVariables();
        Intent i = new Intent(Scales.this, ScalesOptions.class); i.putExtra("KEY", "KEYC");
        startActivity(i);
        finish();
```

Once the ScalesOptions class is launched, a Bundle object will be used to retrieve the key selected by the user, using the key word “KEY”, in the same way as with the Chords section, with the data being assigned to the String variable “key”. Depending on the value of key, it will be assigned a new value, which contains only the name of the key, rather than remaining, for example, as “KEYB”, as can be seen in Figure 73. The title of the page will then be set using simply the name of the key selected, and six varieties of scales will be available for selection by the user. As Figure 73 demonstrates, the key and the scale selected by the user will be passed to the ScaleDiagrams class using the putExtra() method associated with an Intent, which will allow the ScaleDiagrams class to display the appropriate scale, in the appropriate key.

After the ScaleDiagrams class has retrieved the scale and key from the Intent using the Bundle object, and this has been assigned to the String variable “scale”, the correct title and TextViews representing the tablature must be displayed. Therefore, using the contains() method associated with Strings, a search must be made of this String to determine which key, and scale, have been selected by the user (Figure 74). Twelve TextViews have been used for the tablature, as there are two scale patterns, with six strings for each pattern, and their text is set depending on the scale and key selected by the user, thus allowing the same layout file to be used for this class, regardless of the numerous combinations of key and scale.

c 74 – Depending on the data found in the scale variable, the TextViews of both scale patterns will differ

Figure 73 – ScalesOptions.java, the key and scale are passed to ScaleDiagrams.java.

```
                    if (key.equals("KEYB")) {
                        key = " B";
                    }//if
                }//else
            }//else
        }//else
    }//else
}//if
scalesOptionsKeyOf.setText(key);
//Minor Pentatonic
minorPentatonicScale.setOnClickListener((v) -> {
        nullifyVariables();
        Intent i = new Intent(ScalesOptions.this, ScaleDiagrams.class);
        i.putExtra("KEYSCALE", "Minor Pentatonic Scales in the Key Of" + key);
        startActivity(i);
        finish();
});//minorPentatonicScale
```

```
if (scale.contains("Minor Pentatonic")){
    //Key Of C
    if (scale.contains("Key Of C")) {
        pattern1Line1.setText("|----3--1----------------------------------|");pattern1Line2.setText("|----------4--1----------
        pattern1Line3.setText("|----------------3--0----------------------|");pattern1Line4.setText("|----------------------3
        pattern1Line5.setText("|--------------------------------3--1------|");pattern1Line6.setText("|-----------------------
        pattern1Line7.setText("|-----------------------------------1--3---|");pattern1Line8.setText("|-----------------------
        pattern1Line9.setText("|----------------------0--3---------------|");pattern1Line10.setText("|----------------1--3---
        pattern1Line11.setText("|----------1--3----------------------------|");pattern1Line12.setText("|----1--3--------------

        pattern2Line1.setText("|----8--6----------------------------------|");pattern2Line2.setText("|----------8--6--------
        pattern2Line3.setText("|----------------8--5----------------------|");pattern2Line4.setText("|----------------------
        pattern2Line5.setText("|--------------------------8--6------------|");pattern2Line6.setText("|----------------------
        pattern2Line7.setText("|---------------------------------6--8------|");pattern2Line8.setText("|----------------------
        pattern2Line9.setText("|----------------------5--8-----------------|");pattern2Line10.setText("|----------------5--8-
        pattern2Line11.setText("|----------6--8----------------------------|");pattern2Line12.setText("|----6--8------------
    }//if
    else{
        //Key Of D
        if (scale.contains("Key Of D")) {
            pattern1Line1.setText("|----5--3--1----------------------------|");pattern1Line2.setText("|------------3--1-
            pattern1Line3.setText("|------------------2--0------------------|");pattern1Line4.setText("|------------------
            pattern1Line5.setText("|-----------------------------3--0--------|");pattern1Line6.setText("|------------------
```

5.6.3 FAQ

The FAQ section contains ten of the most common questions that are a beginner guitar player may be faced with during their education, with topics ranging from equipment, to practicalities like changing or tuning the strings. The implementation of this section was perhaps the most straightforward of all of the sections, with each question, and the topic menus, being given their own Java class and layout file.

Figure 76 – Creating a VideoView, with the video stored in the raw folder.

tion.

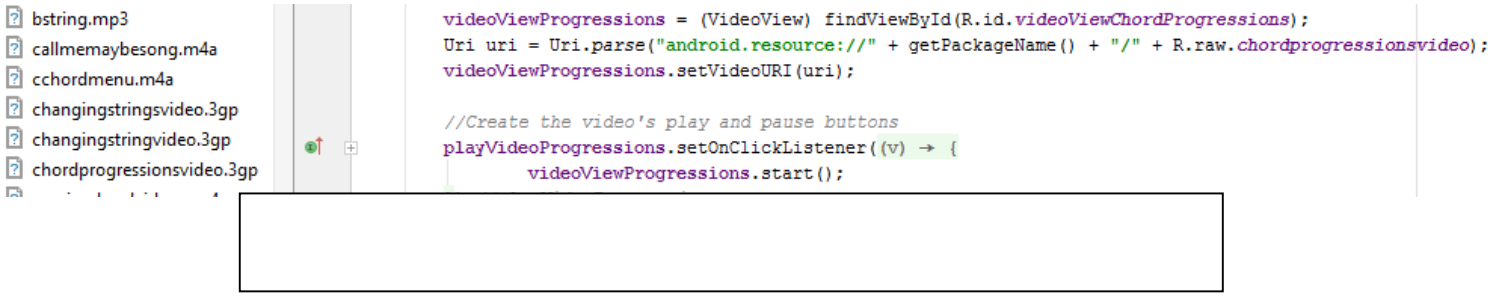
Therefore, when a selection is made by the user, the appropriate Java class and layout file will be launched. A variety of media (Figure 75) is used to answer the questions, especially text and images, with occasional links to lessons from the Lessons section that will address specific concerns.

```
setContentView(R.layout.faqaccessories);

//Initialise variables
picks = (ImageView) findViewById(R.id.accessoriesPickImage);
guitarCase = (ImageView) findViewById(R.id.accessoriesCaseImage);
strings = (ImageView) findViewById(R.id.accessoriesStringsImage);
capo = (ImageView) findViewById(R.id.accessoriesCapoImage);
back = (Button) findViewById(R.id.backButtonFAQAccessories);
home = (Button) findViewById(R.id.homeFAQAccessories);
backRV = (RippleView) findViewById(R.id.backButtonFAQAccessoriesRV);
homeRV = (RippleView) findViewById(R.id.homeFAQAccessoriesRV);
titleFAQAccessories = (TextView) findViewById(R.id.faqAccessoriesHeader);
```

5.6.4 Lessons

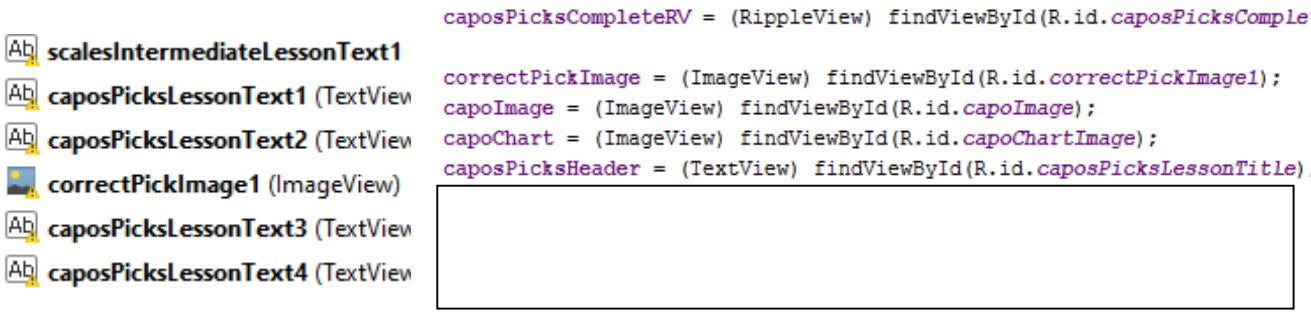
The Lessons section has already been addressed in the "Database Implementation" section of this chapter; however there are a few other issues that must be analysed with regards to the implementation of this section. In the same vein as the FAQ section, each lesson, and the lesson menus, has their own Java classes and layout files. As has been discussed at length in this report, it was the developer's intention from an early stage to ensure that video tutorials were an essential part of the Lessons section, given the myriad of noted benefits associated with this mode of education. In order to display a video in an Android framework, it must be wrapped in a VideoView container that will allow it to be played. Figure 76 demonstrates how the VideoView is created – the package name and the location of the video itself in the raw folder are required to parse the video path to a URI (Uniform Resource Identifier) object. The VideoView object's path is then set to the URI that is created. Play and pause buttons are also created, allowing the video to be played and paused at will by the user. The location of the video within the raw folder can also be seen in Figure 76, along with the other media files stored in this folder.



However, not all lessons use VideoViews. It was decided that the best educational experience for the user would involve a variety of media, including text and image. This involves the use of TextViews and ImageView widgets to display the text and images respectively. An example of one such lesson would be the "Capos & Picks" lesson, found in the Intermediate Lessons section of the app. As can be seen in

Figure 77 – Widgets in the layout file capospickslesson.xml (left) are initialised in CaposPicksLesson.java.

Figure 77, this lesson contains three ImageViews, and eight TextViews; all of which are used to provide information to the user surrounding the use of capos and picks by a guitar player.



Aside from the creation of these widgets in the layout file, in this case capospickslesson.xml, and their initialisation in the Java class, as demonstrated in Figure 77, there is little other implementation required. A large proportion of the code surrounding the Lessons section revolves around the implementation of the lesson tracking system, which has previously been discussed.

5.6.5 Songs & Tabs

As with the Lessons section, a large part of the implementation related to the Songs & Tabs section was addressed in the "Database Implementation" section of this chapter in relation to favouriting and un-favouriting songs, however there are still aspects to be analysed. The motive behind the inclusion of songs and tabs is clear – it allows the user to practice their newly acquired skills in a practical way, while allowing them to practice other skills, including keeping tempo, strumming pattern, and chord transitions. There are six Java classes directly related to the Songs & Tabs section – Songs.java, BeginnerSongs.java, IntermediateSongs.java, AdvancedSongs.java, ViewUsersFavedSongs.java, and SongSelectedDisplayed.java. The Songs.java class presents the three categories of songs available to the user - Beginner, Intermediate, and Advanced – and depending on the option selected, one of BeginnerSongs.java, IntermediateSongs.java, or AdvancedSongs.java classes will be launched. The process for selecting a song is the same for each of these sections, so for the purposes of demonstration, an analysis of the process of selecting an Intermediate song will undertaken.

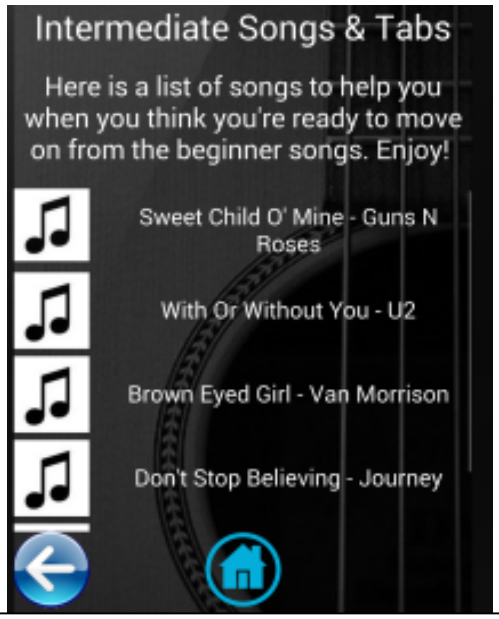
When the user selects to access the Intermediate Songs section, the IntermediateSongs.java class is launched, along with the intermediatesongs.xml layout file. In this section, five songs will be displayed, as can be seen in Figure 78. Five buttons represent each song, with each button having the background of a music note, which is clearly synonymous with music, in particular sheet music. The TextViews beside the buttons inform the user as to which song each button represents. In order to ensure that the song buttons are not too close together, and are clearly distinguishable from each other, the five buttons have been wrapped in a ScrollView container (Figure 78), allowing only the section of the screen containing the song buttons to be scrolled up and down, to access all five songs.

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="450dp"
    android:id="@+id/intermediateSongsScrollView" >

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/intermediateSongsTableLayout">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="111dp"
            android:id="@+id/intermediateSongsSweetChildTR">

            <com.andexert.library.RippleView
                android:id="@+id/sweetChildButtonRV"
                rv_centered="true"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                app:rv_color="#0317f6"
                app:rv_rippleDuration="50"
                app:rv_type="simpleRipple"
                android:layout_marginBottom="10dp"
                android:layout_gravity="center">
            <Button
                android:layout_width="100dp"
```



Once a song has been selected, the title of that song is passed to the SongSelectedDisplayed.java class as an extra to the Intent launching the activity, with the key of "SONG", as can be seen in Figure 79.

```
sweetChild.setOnClickListener((v) -> {
        nullifyVariables();
        Intent i = new Intent(IntermediateSongs.this, SongSelectedDisplayed.class);
        i.putExtra("SONG", "Sweet Child Of Mine");
        startActivity(i);
        finish();
```

SongSelectedDisplayed is the class that controls which songs are displayed. While this requires only one Java class, there is a layout file for each song, which will change depending on the song selected. The prototype design included a single Java class and a single layout file, whose features would change depending on the song selected by the user. While this system may have worked when the songs were displayed as ImageViews, as the ImageView could simply be swapped for another once a new song is selected, this was troublesome when using TextViews, as each song requires a different number of TextViews to display all lyrics and chords. Therefore, it was decided to keep the single Java class, but have a separate layout file for each song, with the layout of the class depending on the song selected. While this may have been a mild inconvenience when converting to TextViews, it was undoubtedly the correct decision to do so, given the clarity with which the songs are now displayed, as well as the ability of the developer to alter the text size and colour with ease.

The song selected by the user from the previous section is retrieved using the Bundle object, as has previously been described, and is assigned to the String variable songSelected. On each of the layout files for the songs, there are various features that must be initialised, including the layout, and the

Figure 80 - Depending on the option selected, the variables will be initialised differently.

various buttons. Using "if" statements, these features are correctly initialised depending on the value of songSelected, as can be seen in Figure 80. So if, for example, songSelected is equal to "Brown Eyed Girl", the variables declared in SongSelectedDisplayed will be initialised using the widgets created in the browneyedgirlsheetmusic.xml file. As Figure 80 highlights, the IDs of the variables in SongSelectedDisplayed.java will change depending on the song selected by the user.

```
if (songSelected.equals("Call Me Maybe")) {
    callMeMaybeVariables();
}//if
else {
    //Sweet Home Alabama
    if (songSelected.equals("Sweet Home Alabama")) {
        sweetHomeAlabamaVariables();
    }//if
    else {
```

```
public void callMeMaybeVariables(){
    artistSelected = "Carly Rae Jepsen";
    layout = R.layout.callmemaybesheetmusic;
    setContentView(layout);
    faveButtonID = R.id.addToFavesCallMeMaybe;
    unfaveButtonID = R.id.removeFromFavesCallMeMaybeSheetMusic;
    backButtonID = R.id.backCallMeMaybeSheetMusic;
    homeButtonID = R.id.homeCallMeMaybeSheetMusic;
    faveSongSelectedDisplayed = (Button) findViewById(faveButtonID);
    removeFromFavesSongSelectedDisplayed = (Button) findViewById(unfaveButtonID);
    backSongSelectedDisplayed = (Button) findViewById(backButtonID);
    homeSongSelectedDisplayed = (Button) findViewById(homeButtonID);
    playSong = (Button) findViewById(R.id.playSongCallMeMaybe);
    pauseSong = (Button) findViewById(R.id.pauseSongCallMeMaybe);
    resetSong = (Button) findViewById(R.id.resetSongCallMeMaybe);
    songMediaPlayer = MediaPlayer.create(SongSelectedDisplayed.this, R.raw.callmemaybesong);
    songSelectedDisplayedLL = (LinearLayout) findViewById(R.id.callMeMaybeLL);
```

Later in this class, the code can be found to allow for the functions of the various buttons to be implemented, depending on the layout of the file. For example, as in Figure 81, if the layout is equal to R.layout.callmemaybesheetmusic, which is an integer, then a switch statement controls the operation of the various functions of the buttons.

```
if (layout == R.layout.callmemaybesheetmusic) {
    switch (v.getId()) {
        //If add to favourites pressed
        case R.id.addToFavesCallMeMaybe:
            addToFaves();
            break;
        //Remove from favourites
        case R.id.removeFromFavesCallMeMaybeSheetMusic:
            removeFromFaves();
            break;
        case R.id.playSongCallMeMaybe:
            songMediaPlayer.start();
            break;
        case R.id.pauseSongCallMeMaybe:
            songMediaPlayer.pause();
```

For example, if the addToFaves button is called, regardless of the specific song to which it may belong, for example addToFavesCallMeMaybe or addToFavesHotelCalifornia, it will call the same methods, and perform the same activities. As there are buttons for each song available for selection, every button must be accounted for in the SongSelectedDisplayed class. Using the "if" and switch statement ensure that every button on all fifteen of the layout files performs its task as it should.

As well as providing the lyrics and chords for each song, a recording of the song being played by the developer has also been implemented into the application. The .m4a files are stored in the raw folder and, using a MediaPlayer object, are created and played by pressing the "play" button on each song screen. There are also buttons available that will both pause and reset the recording. This allows the user to hear how each song should sound, and provide them with guidance surrounding the tempo, and when the chord changes should occur.

Aside from using the Songs section of the application, there is another way to access the SongSelectedDisplayed class to display a selected song. As has already been established, when a user has favourited a song, it will added to their list of favourited songs, which is accessible through the MyAccount.java class. This list of favourited songs is controlled by the ViewUsersFavedSongs.java class, and it accesses the SongSelectedDisplayed class in the same way as the IntermediateSongs class – by passing the name of the song as an extra to the Intent launching the class (Figure 82). Depending on the button pressed by the user, the value of the String variable song will change, and this variable is passed to SongSelectedDisplayed as an extra to the Intent. This data will then be used by the SongSelectedDisplayed class to display the appropriate song.

```
if (data.contains("Brown Eyed Girl")){
    brownEyedGirl.setVisibility(View.VISIBLE);
    brownEyedGirlTR.setVisibility(View.VISIBLE);
    brownEyedGirl.setOnClickListener((v) → {
            song = "Brown Eyed Girl"; artist = "Van Morrison";
            Intent i = new Intent(ViewUsersFavedSongs.this, SongSelectedDisplayed.class);
            i.putExtra("SONG", song); nullifyVariables();
            startActivity(i);
            finish();
    });//brownEyedGirl
```

5.6.6 Tuner

The inclusion of a Tuner section into the application allows the user to ensure that their guitar is in tune prior to attempting to use any of the other features in the app. This section is divided into three different tunings – Standard, Drop D and Open C. All of these tunings have their own class and layout file, and the code for each tuner is the same, except for the sound that is played. As Standard is the most common tuning, and thus will be the tuner that is most frequently used, its implementation shall be discussed. In the tunerstandard.xml file, the background of the screen has been amended to include a fretboard, with six buttons placed over each of the strings. When one of these buttons is pressed, a

Figure 83 – The code required to play the sound file for one of the strings in the Tuner section.

recording will play the sound that that string should make, allowing the user to tune their string until the sounds match. Figure 83 highlights that a MediaPlayer object has been created for each string, however through the use of an onCompletionListener, the object can be released as soon as it is finished playing, at which point it is set to null. This allows for only one recording to be played at one time, as the "if" statements found in Figure 83 make it clear that if a MediaPlayer object is not null, it is currently playing, therefore the sound is stopped, and the object is nullified, allowing the selected MediaPlayer to be initialised and played.

```
lowC.setOnClickListener((v) → {
        if (highESound != null){
            highESound.stop();
            highESound = null;
        }
        if (highCSound != null){
            highCSound.stop();
            highCSound = null;
        }
        if (midGSound != null){
            midGSound.stop();
            midGSound = null;
        }
        if (midCSound != null){
            midCSound.stop();
            midCSound = null;
        }
        if (lowGSound != null){
            lowGSound.stop();
            lowGSound = null;
        }
        if (lowCSound != null) {
            lowCSound.stop();
            lowCSound = null;
        }

        lowCSound = MediaPlayer.create(TunerOpenC.this, R.raw.openclowestring);
        lowCSound.start();
        lowCSound.setOnCompletionListener((mp) → {
                mp.release();
                lowCSound = null;
        });
```

An additional feature specific to the Standard tuner is the "Try It Out!" section, which features a SeekBar widget that allows the user to drag the pointer up and down over the strings, allowing them to "strum" the strings. When the pointer passes over a string, the sound of that string will be played. The variable "progress" found in Figure 84 represents the position of the pointer in relation to the strings, from 0 to 100. Therefore when the progress is 4, this is over the low E string, therefore the sound of this string is played. This allows the user to simulate strumming the strings.

Figure 84 – Depending on the progress value, the sound played will change.

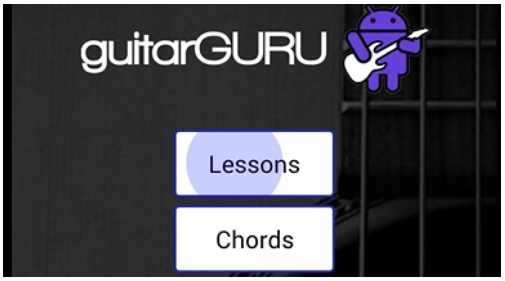
ure 85 - Implementing the Ripple
ect library.


```
if ((progress == 4)) {
    shortLowESound = MediaPlayer.create(this, R.raw.shortlowe);
    shortLowESound.start();
    shortLowESound.setOnCompletionListener((mp) → {
            shortLowESound.release();
    });
}//if
```

5.7 Libraries, and Code from Other Sources

Code from other sources was used to great effect throughout the application, being used exclusively in relation to either the design of the buttons, or the font of the section titles. In relation to the design of the buttons, it was decided that some kind of visual feedback was required when the user pressed a button. In order to add some manner of visual effect to each button, a library named “RippleEffect” was downloaded from Github (traex, 2014). When a button is pressed, a ripple spreads out from the point on the button where the user pressed, as can be seen in Figure 85. After downloading the library, the executable .jar file that was included is added to the lib folder, and added as a library. In order for this library to actually be applied to the buttons, however, each button designed in a layout file must be wrapped in a RippleView custom view, as can be seen in the code in Figure 85. The duration of the ripple, as well as the colour, and the type can all be set. By setting the width and height of the RippleView to wrap_content, it wraps around the dimensions of the button, and it appears to the user as if the ripple is actually coming from the button itself, rather than the RippleView.

```
<com.andexert.library.RippleView
    android:id="@+id/accountScreenLoginButtonRV"
    rv_centered="true"
    android:layout_width="100dp"
    android:layout_height="60dp"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:layout_marginBottom="100dp"
    android:gravity="center_vertical|center|center_horizontal"
    app:rv_color="#0317f6"
    app:rv_rippleDuration="50"
    app:rv_type="simpleRipple">
    <Button
        android:id="@+id/accountScreenLoginButton"
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:layout_gravity="center_horizontal"
        android:background="@drawable/rounded"
        android:clickable="true"
        android:text="Login"
        android:textSize="35dp" />
</com.andexert.library.RippleView>
```



As can also be seen in Figure 85, the corners of each button have been rounded, and a thin blue border added. The code to achieve this effect came from an Android developer blog (Android Code Snippets, 2015), and it recommended creating an .xml file, inside the drawable folder, which would set the

dimensions, border, and padding for the buttons. The background of each button would then be set to the .xml file, named "rounded", using the background path "@drawable/rounded", rather than having to change the dimensions of each button manually. This was a significantly time conserving practice, as to have to change the dimensions of each button would have taken a rather long period of time.

Finally, a custom font has been used for the application logo, and all screen titles, as can be seen in Figure 85. The developer had a basic idea of how to implement custom fonts into an Android application, however it was not working. The solution was found on a renowned developer website (StackOverflow, 2010), which suggested creating a new folder to store the .ttf font, which had not yet been attempted by the developer. The font, downloaded from the website dafont (dafont.com, 2016), was therefore added to a newly created folder named fonts, which remedied the problem.

5.8 Other Changes from the Prototype

The final design of the Scales and Chords sections is markedly different when compared to the initial design. When designing an early prototype of the application, each Chord that could be selected by the user had its own Java class and layout file, with the appropriate class and layout file being launched when a chord was selected. However, once the classes for every chord had been accounted for, the application had over 110 classes. It was therefore decided to combine all of the chords into a single class, with the information displayed being dependent upon the chord selected by the user. This reduced the number of classes and layout files dramatically, and the use of a single class and layout file is significantly more efficient.

Alongside this, the chord and scale diagrams were initially designed to be displayed using images, rather than the TextViews that are used in the final version. The reasons for this alteration revolve around two main factors – clarity, and the size of the application. These factors also caused a major issue for the Songs & Tabs section. Despite altering the size and definition of the images that would display the chords, scales, and songs, the clarity was not perfect, and sometimes the information was not fully legible. Alongside this, the sheer number of chord and scale diagrams that are used in this app is very high, with each image contributing to the overall size (in MBs) of the application more than virtually any other component. This was exacerbated by the size of the song images, which are unavoidably large given the dimensions required to display a song. The images also dramatically increased the size of the heap space used by the Dalvik Virtual Machine, which caused the performance of the app to suffer, and sometimes crash. Thorough research into this problem indicated that this is problem inherent with using large images. It was decided that as images for the chords, scales, and songs were not essential, as TextViews could easily be used, it would more appropriate to use TextViews instead. The added benefit of this is that there are no longer any issues with clarity, as the TextViews display the information in a crystal clear way, and the overall size of the application was reduced dramatically once all of these images had been removed.

Relating specifically to the Chords, Scales and Songs sections, the implementation of the system allowing data to be passed through Intents was only a relatively recent discovery. Previously, static variables, and get() and set() methods, had been used in all of these classes to retrieve data from the other classes. This system worked perfectly well, but on the discovery of the use of Intents to pass information, it was

decided that this would be a more appropriate, and efficient, way of retrieving data from other classes. Research into the methods through which other users pass data between classes indicated that the use of Intents appears to be one of the most common, and the use of Intents also has the advantage over static variables as there are significantly less object references that require creation. Previously, in order to access the static variables in another class, a reference to the class had to be created. The fact that this is not required with the use of Intents is significantly more efficient, and conserves time for the developer, given that the use of Intents is much more straightforward.

5.9 Summary

In this chapter, a thorough analysis was undertaken regarding the implementation process of the functions and features designed for the application. Particular focus was given to the implementation process and environment, as well as to the adoption of the Agile methodology, and the way in which this would affect the development.

Subsequently, the actual code that was created was examined, particularly in relation to the main Java classes and folders that were required, the implementation of the database into the application, and the manner in which it operated in relation to various features within the app, including favouriting songs, tracking lesson progress, and saving the high scores of the quizzes. This was followed by a discussion regarding the implementation of the other sections of the app, and the libraries and additional fragments of code that were particularly useful in development.

Chapter 6 – Testing and Evaluation

6.1 Introduction

Thorough testing of software is critical to the success of any system, particularly in terms of its quality and usability. If the software is unreliable, does not operate as intended, or its performance is sluggish, users will almost certainly be unwilling to continually use it. Testing is traditionally "performed for one of two reasons: defect detection, and reliability estimation" (Ahamed, 2009, p.119). Defect detection in particular is vital, as early detection of any bugs or issues allow for these problems to be remedied before the software is released. Testing was carried out on a regular basis during the implementation stage of development, as a consequence of the adoption of the Agile model, in which development is divided into increments named sprints. Certain aspects of development are prioritised in these sprints and, at the termination of each sprint, the implementation undertaken up until that point is tested to ensure full functionality, and an absence of defects. Therefore, continuous testing was undertaken throughout development.

Once testing of the app has been completed on two different tablets, the app will be distributed to eight individuals for evaluation. Using the application installed on the developer's tablet, the evaluator will follow the instructions provided alongside a questionnaire that has been created by the developer, with these questions then being answered by the evaluator. These questions will relate to the features of the app, its usability, and the user's overall experience in using the app. The use of a questionnaire allows the developer to gauge the extent to which the objectives, as well as the functional and non-functional requirements specified earlier in this report, have been satisfied.

6.2 Functionality

The functionality of software is arguably the most important consideration for both developers and users. For example, if an application has an excellent user interface, but regularly crashes, or some of the features themselves do not work, the application has been rendered virtually redundant in terms of its potential use to the general public. Therefore, the testing of an app's functionality must be thorough. Prior to the commencement of the testing phase, it is important to develop an initial testing plan, in order to ensure that the testing that occurs is both efficient and meticulous. The developer therefore gave particular focus to the "Seven Principles of Software Testing" (Ghahrai, 2008). Four of the principles in particular were of interest.

- *Exhaustive testing is impossible* – Given the amount of data that could be input by users, and the various data combinations that could occur, for example favourited songs and lesson progress codes for each and every user, it is simply "not possible to test all possible combinations of data and scenarios" (Ghahrai, 2008). Therefore, it is critical to prioritise various aspects of the testing. While each section will be tested thoroughly on two different devices, particular focus will be given to the features requiring the use of the database, for example favouriting songs & tabs, and lesson tracking.
- *Early Testing* – Early testing ensures that more time is available for making any amendments that may be required. As has been previously mentioned, testing was carried out at regular

intervals since the earliest stages of the implementation process. However, the final testing phase is now being conducted at an early enough stage to allow for the correction of any bugs or errors that are found during testing.

- *Defect Clustering* – The concept of defect clustering refers to the fact that most of the reported defects that are encountered during testing are contained within a small number of modules within the system. This is an example of the Pareto Principle, in which "approximately 80% of the problems are found in 20% of the modules" (Ghahrai, 2008). This is logical, particularly regarding modules that are more complex. Therefore, particularly scrutiny will be applied to lesson progression, favouriting songs, and the Tuner section, which involves the use of MediaPlayer objects and sound files.
- *Absence of errors fallacy* – It would be foolish to believe that just because no errors were encountered during testing, no errors exist. It is therefore pivotal to ensure that as many problems are dealt with as possible during this phase of the project, especially if they affect the operation of the key features of the app.

Based on the guidance provided by these principles, the developer made the decision to ensure that while each section of the application would be tested thoroughly, on more than one occasion and on two different devices, particular attention would be given to the more complex features within the application – namely the lesson tracking capability, favouriting songs, and playing their associated media files, and the Tuner section.

6.2.1 Previous Testing Throughout Development

As has already been discussed, a consequence of the use of the Agile model of development is that frequent testing occurs at the end of each sprint. This regularity in terms of testing ensures that any bugs that are unknowingly created during implementation can be spotted at an early stage, and removed before any further implementation occurs. One such example of this that became known to the developer relatively recently revolved around the use of the device's own back button, and the effect it has on MediaPlayer objects. This is particularly in reference to the Tuner section of the app, where these objects are required to play the sound of each string. In order for these MediaPlayers to play the required sound, they must be initialised, with this initialisation occurring in the onCreate() method in the class, which is called as soon as the class is launched.

To explain the issue that arose, the example of the Standard Tuner section of the app will be used. When the TunerStandard.java class is launched, the six MediaPlayer objects for each string are initialised in the onCreate() method. However, if the user then selects the "Try It!" option present on this screen, the SeekBarSounds.java class will be launched, and the MediaPlayer objects are released. In order to return back to the Standard Tuner screen, the user has two options. They can either use the device's back button, or the on-screen back button, designed by the developer. The on-screen back button will reload the TunerStandard.java class, and thus the onCreate() method is executed, initialising the MediaPlayers. However, if the device back button is pressed, the class is already present in the device memory, and thus the onCreate() method is not required. The MediaPlayers have been released, and are not re-initialised; therefore any attempt to press one of the string buttons will lead to an IllegalStateException, in which the MediaPlayer cannot play the sound file as it has been released.

Figure 86 – Overriding the onBackPressed()

When researching a solution to this problem, it was decided that perhaps utilising the device's back button was unwise in the first place, due to the possibility that the hardware could fail. If the back button stops working on any device on which the application is installed, allowing the use of the device back button would be rendered redundant. This possibility is the main factor behind the decision taken by the developer at the earliest stages of design to ensure that all navigation can be undertaken on screen, in the vein of Apple's iOS devices. The device's back button was therefore disabled in favour of the on-screen navigation functions. This was done by overriding the onBackPressed() method, which controls the device back button, and leaving it empty, as can be seen in Figure 86. When a user now presses the device back button, nothing will happen.

```
@Override
public void onBackPressed(){ }
```

6.2.2 Testing on Different Devices

Throughout the development process, all previous testing had been carried out using the developer's own Android tablet, a Samsung Galaxy Tab 4 7". It was therefore important during the testing phase to install the application on another device in order to test its operation, both in terms of the display of the screens, and the functionality of the various features. The developer would use the devised testing plan, sections of which can be found in Appendices 6 and 7, and the theoretical user scenarios found in Appendix 8, and would perform all of the tasks on each tablet, in order to ascertain whether the app would remain fully operational on different devices. This secondary device is also a Samsung Galaxy Tab 4. The results of these various tests can also be seen in Appendices 6, 7 and 8. These results will be analysed later in this report.

In terms of the layouts of the screens in particular, no problems whatsoever were encountered with either device. All features were displayed on the screen, and the user interface was structured in the exact way in which it was designed. The secondary tablet has the same screen dimensions as the primary tablet (7 inches), and this outcome might naturally be expected, given that the application was designed for tablets with 7 inch screens. However, that is no guarantee that both applications would be correctly displayed on different devices. Fortunately, this did not prove to be an issue.

6.2.3 The Functionality Testing Process

Functionality is central to the success of an application. Users have a certain expectation when it comes to using applications – they expect it to be of a high quality, to be responsive, and to deal with all of their requests both efficiently and in the expected manner. It is therefore essential that, before users get to interact with the app, all functionality must be fully tested, and be fully operational. Appendices 6, 7 and 8 detail the manner in which this testing was undertaken. As can be seen, the tests undertaken were very thorough – including a test of whether the correct screen was displayed, whether media files play and pause when prompted, as well as a test of every navigational tool present on each screen. Virtually every action that could be taken by a user was tested. An explanation should be provided

regarding the tests described as "Drop-down menu function (not signed in)" and "Drop-down menu function (signed in)". Depending on the status of the loggedIn boolean found in the LoggedInID class, the class that is launched will be different. When the user opens the drop-down menu and selects the "Account" option, if the loggedIn boolean is true, a user is signed in, and they are taken to the "My Account" section. Otherwise, the user is taken to the "Account" section. Tests were carried out to check whether, depending on the login status of the user, the correct class is launched.

In terms of the actual testing carried out on both devices, the results are summarised in Table 6 below.

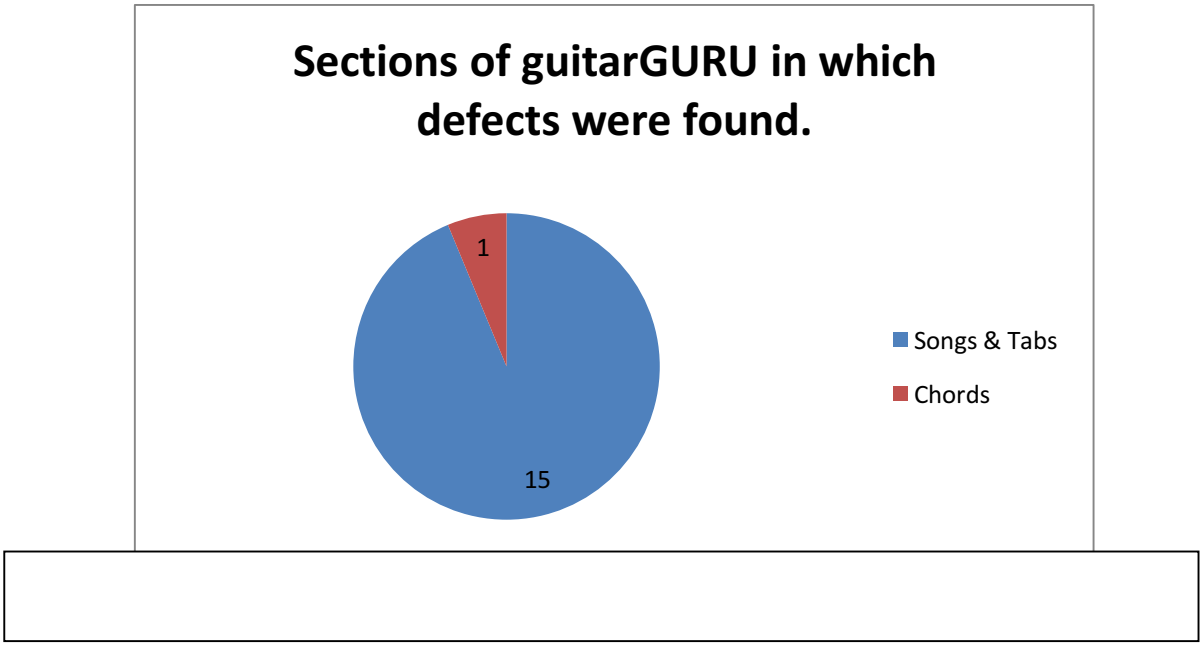
Section	*Number of Tasks*	*Tablet 1 Passes/ Failures*	*Tablet 2 Passes/ Failures*	*Reasons for failure.*
Splash Screen	3	3/0	3/0	
Main Menu	9	9/0	9/0	
Lessons	157	157/0	157/0	
Chords	616	615/1	615/1	Incorrect video recorded by developer.
Songs & Tabs	156	141/15	141/15	My Favourited Song section launched, but recording of song did not stop playing – must be amended.
Tuner	49	49/0	49/0	
Scales	376	376/0	376/0	
FAQ	102	102/0	102/0	
Account/Database	102	102/0	102/0	

.

As Table 6 demonstrates, there were 1,570 tasks carried out on each device. These tasks could realistically be undertaken by a regular user during use of the app, therefore it was essential that the testing was thorough. These tasks revolved around the testing of various features on each screen – including all navigational features, namely the back and home buttons, and the drop-down menu, as well as the playing, pausing, stopping, and resetting of various media files. As well as this, the more basic elements of the app's operation were also tested, including whether each button operated as intended, whether the features were responsive, and whether the correct screen was launched when a new activity was selected. Appendix 6 demonstrates that this testing was thorough in terms of analysing these facets of the application. Aside from the 16 failures on each device, which shall be addressed later, the developer is satisfied with the results of the testing, with 1,554 of the tasks undertaken being successful. This equates to a pass rate of 98.98% throughout the whole application. However, it must be noted that the scarcity of errors found during this final testing stage may be due to the regularity of testing that has occurred throughout the development process. Any errors that occurred during these stages may well have been encountered and addressed prior to this current testing stage, thus they are

Figure 87 – Graph showing in which sections errors were found during testing.

no longer a factor. This appears to be a logical explanation for the small number of defects present in the application.



As Figure 87 demonstrates, of the sixteen tasks that were failed, the vast majority of them occurred in the Songs & Tabs section. Fifteen of the sixteen errors were encountered in this section, with one error being found in the Chords section. In terms of the Chords section defect, this was a result of simple human error – when recording the video to be played on the screen displaying a chord, the developer had misread the chord diagram, and the finger positions for the G#dim chord were incorrect. This is a minor error, and will be solved imminently.

The fifteen other errors were encountered in the Songs & Tabs section, and are all in relation to the same aspect of the section. The fact that almost 94% of the errors found throughout the entire application are found in this section only provides further support for one of the seven principles of software testing mentioned earlier - "defect clustering", which is the concept that most of the reported defects that are encountered during testing are contained within a small number of modules. These principles, including defect clustering, were at the forefront of the developer's mind during testing, prompting the developer to be extremely thorough when testing the more complex modules within the system. This in part led to the discovery of this error.

The error itself is again fairly minor, and can be easily amended by the developer. When the device back button was disabled by the developer, if the user was accessing songs from their "My Favourited Songs" section, there was no easy way to access the "My Favourited Songs" section without returning to the Account section. It was therefore decided to provide a shortcut from the screen displaying the song directly to the "My Favourited Songs" section, using the "heart" button that allows a user to favourite a song. If the user is viewing a song, and wishes to access their favourites, they can simply press their finger down on the heart and hold it, known as a long click, until the "My Favourited Songs" section is

launched. Again, this is dependent on the signed-in status of the user, using the loggedIn boolean from the LoggedInID class.

```
faveSongSelectedDisplayed.setOnLongClickListener (((v) → {
        if (loggedInID.isLoggedIn()) {
            Intent i = new Intent(SongSelectedDisplayed.this, ViewUsersFavedSongs.class);
            i.putExtra("SONG", songSelected);
            startActivity(i);
            nullifyVariables();
            finish();
        } else {
            Dialog d = new Dialog(SongSelectedDisplayed.this);
            d.setTitle("Oh dear!");
            TextView tv = new TextView(SongSelectedDisplayed.this);
            tv.setText("You must sign in to access your favourite songs!");
```

As Figure 88 demonstrates, if the user is not signed in when the favourite button is long clicked, an error message will be displayed informing the user to sign in. Otherwise, the signed-in user is taken to their "My Favourited Songs" section. However, if the user presses play on the recording of a song, and presses the heart to access their favourited songs, this recording will not stop playing, and will continue to play while the user browses their favourited songs. As the fifteen songs available to the user use the same Java class that contains the MediaPlayer object controlling the recordings, this error affects each song, hence why there are fifteen errors. As has been mentioned, this is a minor error, and one that will be fixed imminently by the developer.

At certain stages during the implementation process, issues surrounding the ever-increasing allocation size of the Dalvik Virtual Machine (DVM) heap became particularly troublesome. The heap itself is the amount of memory that has been allocated to an application. In an ideal scenario, all memory taken up by a module within an application would be released once this module is no longer in use, allowing this memory to be used by other modules. However, a very common problem among Android developers is a "memory leak", which essentially means that the memory allocated to a module cannot be freed, as references are still held to objects within the class, or to the class itself. Therefore, as the memory cannot be freed, the heap continues to grow with each new module. The maximum heap size is strictly set, and differs depending on the specific device. However, as the heap grows, the performance of the application slows, and "if your app has reached the heap capacity and tries to allocate more memory, it will receive an OutOfMemoryError" (Android Developers Console, 2016). The app will then crash, and will have to be restarted.

During the early stages of implementation, OutOfMemory errors occurred several times. Research into the problem provided a helpful solution that made a significant difference. If the developer were to nullify all variables once an activity is no longer in use, the garbage collector would be able to recycle these variables, and release the memory. This made a notable difference to the performance of the app, and OutOfMemory errors have not occurred since the adoption of this strategy. However, after periods of prolonged use, such as during the testing phase, the heap grows to such an extent that performance

becomes quite sluggish. Although the application did not crash, an additional solution was required. Fortunately, further research after the testing phase provided a hugely helpful technique that releases much greater amounts of memory than had previously been released. By overriding the onStop() method in each activity, which is called when the activity is no longer visible to the user, the developer was able to include calls to both the finish() and onDestroy() methods, which will recycle the memory occupied by these activities when no longer in use, and release the memory to be reused by other activities. While this technique has made a hugely significant difference to the growth of the heap, it is not a final solution to this problem. A solution that will render the heap growth problem redundant could perhaps be sought during future development, as shall be discussed further in Chapter 7.

6.2.4 Testing the Database Functionality

Many of the core features of the application revolve around the successful operation of the database, for storing the details of users, and for storing their favourited songs. It was therefore decided that a separate testing section should be created to test all aspects of the database implementation, with the testing plan being found in Appendix 7. Testing was focused on signing in, creating a new account, tracking lesson progress, accessing the quizzes, and favouriting and un-favouriting songs; all of which rely on the data contained within the database. As Appendix 7 demonstrates, 102 tests were carried out on the various sections that required the use of the database. Fortunately, all of these tasks were successful, and no failures or errors occurred. This is again very pleasing, yet may be considered unsurprising, especially given the continuous testing that occurred throughout development. Any defects that may have been present were discovered at any early stage, and were remedied. Aside from any minor oversights from the developer during testing, it appears that the database is fully functional and operates as intended.

6.2.5 Testing Theoretical User Scenarios

In order to simulate how the application would operate under normal circumstances when being operated by a user, the developer created a variety of theoretical user scenarios, in which certain tasks that could be expected to be carried out by a normal user are tested. The results of these tests can be seen in Appendix 8. Fortunately, all of the tests carried out by the developer were passed, with no errors or fails being recorded. It was decided that this method of testing was necessary, in order to test both the robustness and operation of the application under realistic circumstance. Thankfully, the application succeeded in all of these tests.

6.3 Evaluation – Questionnaire Responses

Once the testing phase was complete, and all defects that were encountered were remedied, the application was distributed to eight individuals who would evaluate the application. This evaluation involved a questionnaire, which would invite the respondent to carry out various tasks, and evaluate their experiences in using the app on the developer's tablet. The questionnaire, as can be seen in Appendix 9, is targeted at specific elements of the user's experience of using the app, including the usefulness of the application as an educational tool, their opinions on how the information is presented, the usability of the app itself, and, crucially, their overall opinion of the app and whether they would be willing to use it again. The results of the eight questionnaires can be seen in Appendix 10. Prior to analysis of the results, the actual grading of the various answers should be discussed. For the purposes

Table 7 – This table features questions relating to the usability of the application.

of simplifying the results analysis, all answers will be given a number representation. All questions, aside from question 6, have five possible answers, ranging from, for example, Very Poor to Very Good, or Not At All to Definitely. These answers follow a similar scale in terms of meaning; therefore it has been decided to award each answer a number value, ranging from 1 to 5 in ascending order of positivity. This allows for average values to be calculated, and provides a greater indication of the general opinion of the respondents. Therefore, Bad/Very Poor/Very Difficult/Not At All are 1, Not Good/Not Really/Difficult/Poor are 2, Average/Somewhat are 3, Good/Easy/Quite is 4, and Very Good/Very Easy/Definitely are 5. Table 7 displays the responses to the questions relating specifically to the usability of the application. The options selected by the respondents are listed in the "Responses" column, along with the number of respondents that have selected this option.

Question	*Responses*	*Average Response (out of 5)*	*Most Common Response*
5. Were the lessons easy to access and use?	Easy – 3, Very Easy – 5.	4.625	Very Easy - 5
12. What is your overall opinion on the usability and quality of the Lessons section?	Average – 1, Good – 3, Very Good - 4	4.375	Very Good – 4
21. Was this section easy to use? (Songs & Tabs)	Definitely - 8	5	Definitely - 8
25. Was this section easy to use? (Tuner)	Quite – 1, Definitely - 7	4.875	Definitely – 7
31. Was this section easy to use? (Scales)	Definitely - 8	5	Definitely - 8
35. Was this section easy to use? (FAQ)	Quite – 1, Definitely - 7	4.875	Definitely – 7
42. Did you find it easy to navigate through the app using the Back/Home/Drop-down menus?	Somewhat – 1, Definitely - 7	4.75	Definitely - 7
44. Was the application easy to use?	Quite – 1, Definitely - 7	4.875	Definitely - 7

The results found in Table 7 are very promising for the developer. In the "Requirements Analysis" chapter of the report, one of the key non-functional requirements listed was "Usability", in which it was declared that the application must be user-friendly for people of all ages and abilities. In the interests of retaining the confidentiality of the respondents' identities, it is accurate to say that the respondents are a wide variety of ages (20 – 58), as well as a possessing a range of both technical and musical ability (no experience of playing guitar, to seven years of playing). The fact that each respondent responded so favourably in relation to the usability of the app, with the overall usability rating being 4.875 out of 5 and a rating of "Very Easy" as the most common response, indicates that the developer was indeed successful in ensuring that the app was usable and user-friendly for all potential users. Alongside this, a

key functional requirement related to ensuring that navigation throughout the app was easy, thus allowing the user to access any section of the app in an efficient and convenient way. It was pleasing, therefore, that the average response to question 42, found in Table 7, was 4.75/5, with seven respondents selecting "Definitely" easy, and one selecting "Quite" easy. This functional requirement has seemingly been satisfied.

Table 8 – Questions about the quality of the sections, and of the app itself.

Question	*Responses*	*Average Response (out of 5)*	*Most Common Response*
12. What is your overall opinion on the usability and quality of the Lessons section?	Average – 1, Good – 3, Very Good - 4	4.375	Very Good – 4
15. Overall impression of the Chords section?	Good – 4, Very Good – 4	4.5	Good - 4 / Very Good - 4
20. What is your overall impression of the Songs & Tabs section?	Good – 2, Very Good - 6	4.75	Very Good – 6
26. What is your overall impression of the Tuners section?	Good – 2, Very Good - 6	4.75	Very Good – 6
30. What are your overall impressions of the Scales section?	Good – 4, Very Good – 4	4.5	Good - 4 / Very Good - 4
34. Would this section be useful to a beginner guitar player? (FAQ)	Very Good - 8	5	Very Good – 8
40. What is your overall opinion on the usefulness of an Account feature within the app?	Quite – 1, Definitely - 7	4.875	Definitely – 7
41. What is your opinion on the use of the plectrum button for providing advice and guidance?	Quite – 1, Definitely - 7	4.875	Definitely - 7
43. What are your opinions on the overall layout/colour scheme/ legibility of the text?	Good – 4, Very Good – 4	4.5	Good - 4 / Very Good - 4
45. How would you rate your overall experience in using the app?	Good – 5, Very Good - 3	4.375	Good - 5
46. Was the app responsive to all of your requests?	Somewhat – 1, Quite – 1, Definitely - 6	4.625	Definitely - 6
47. Do you believe this application would be useful to beginner guitar players?	Definitely - 8	5	Definitely - 8
49. Would you use the app again?	Quite – 1, Definitely - 7	4.875	Definitely - 7
50. On a scale of 1 to 5, how would you rate the app overall? (1 – very poor, 2 – poor, 3 – average, 4 – good, 5 – very good).	4 – 3, 5 - 5	4.625	5 - 5

Table 8 presents the opinions of the respondents regarding the general quality of each section, as well as several general questions about their experience in using the app itself. Based on the responses, it is clear that, generally, the respondents were pleased with both the content and features provided by the app, as well as with the application as a whole. In Table 8, questions surrounding the respondent's impressions of each section within the app can be found. The responses to these questions were very pleasing. The option "Very Good", the highest option available, was selected by at least half of the respondents to each question, with the remainder of the responses being "Good", with one exception of an "Average" regarding the Lessons section, as this respondent believed that this section could be expanded with more lessons. This recommendation, along with all other recommendations provided through the feedback, will be addressed later in this chapter.

This success of the various features, based on the questionnaire responses, again appears to indicate that several of the functional and non-functional requirements listed in Chapter 3 have been satisfied. Several of these requirements shall be analysed below.

- *The application must provide an assortment of educational tools, ranging from video tutorials to sound files, and other multimedia* – A wide range of multimedia tools are used throughout the app to great effect. For example, video and text lessons in the Lessons section, sound files in the Tuner and Songs & Tabs sections, and video files in the Chords section. The overwhelmingly positive feedback regarding the quality of the features in the application surely demonstrates that this functional requirement has been satisfied.
- *The application must provide the user with instructions and guidance when requested* – Through the use of explanatory error messages, for example when trying to sign in with incorrect details, or the use of the guitar plectrum to display a helpful message, instructions are available to the user, as required by this functional requirement. The responses to question 41 in Table 8 clearly indicate the success of the guitar plectrum in providing advice and in guidance, with an average response of 4.875/5 in terms of this feature's usefulness.
- *Efficiency* – This non-functional requirement is particularly important. It is vital that the app is responsive, and deals with all requests with minimal waiting times. The responses to question 46 in Table 8 clearly indicate that efficiency and responsiveness are not causes for concern regarding guitarGURU, with one user describing the app as "Somewhat" responsive, one user selecting "Quite" responsive, and six users selecting "Definitely" responsive. This leaves the responsiveness question with an average rating of 4.625, which is indeed gratifying.

6.3.1 Additional Feedback and Comments

At the end of each set of questions in the questionnaire, a section was available for any additional comments or feedback that the respondent may wish to include. The feedback ranged from praise to useful recommendations for possible improvements or amendments, and can be seen in Table 9.

Respondent ID	Feedback
A	• The plectrum button should have an indicator informing user of what it does, so don't think it's purely for decoration. • More lessons should be present in each section. • Display length of videos • Process for displaying Chords is easy to follow • Songs & Tabs section good – clear progress in difficulty of songs • Tuner – advise people to only play one string at time, otherwise sounds overlap (since addressed by developer) • Standard Tuning section, especially "Try It!" section, very good • FAQ – good information in questions • Screen background a bit dark – but may just be personal choice • Liked plectrum and Chords section
B	• Reorganise features on Lessons screen – put lessons in different order. • Chords section – relevant and user-friendly video clips. • Tuner – for most efficient use, tap strings one at a time, otherwise overlap. • FAQ – relevant questions and user-friendly answers • Background image – good, but use different colour, maybe brown/honey colour to match a real guitar. • Shorten the length of some of the text lessons – more bullet points and images.
C	• When completing quiz, I can press next without selecting an answer. • Chords – include hand diagram to indicate which finger numbers refer to. • Scales – content difficult for beginners, maybe mark it as advanced in main menu, or only unlocked once person has completed lessons. • Make the yellow plectrum more noticeable, let user know not just for decoration, it's a feature.
D	• Intermediate lessons a little too wordy. Fast forward on videos if feasible? • Chords – better view of finger placement in videos • When pressing back on My Favourited Songs section, taken to My Account section rather than previous song – no major inconvenience, but possible fix. • FAQ – good ties into other sections of app eg Changing Strings FAQ to lesson. • App very easy to use and responsive – largely very impressive, small potential issues do not affect usability, just possible areas to improve experience. Easy to work and understand as a novice. Perhaps a little lighter background in certain areas (hard not to clash with text, I know) and maybe a little wordy in certain areas but this is mere speculation as I had a very small trial of the app.
E	• Lessons – lot of potential for future lessons and progress as you complete lessons. • Chords – as an intermediate guitar player, even I found this part of the app very useful. • Songs & Tabs – Very easy to use and excellent for a beginner to immediately start playing songs. • Scales - Again, as an intermediate guitar player, I would find this section very useful. I wish I had an app to use for scales as a beginner. • FAQ – very common questions for all guitar players are included.

Table 9 – Table summarising the feedback received from the questionnaires.

	• Excellent app for beginner guitar players but also for more practiced guitar player. I would definitely use when playing guitar.
F	• Chords – being able to see the chord being played on screen helpful. • Scales – illustrations of scales very helpful. • Easy guidance & instructions to follow. Very informative.
G	• Lessons – more pictures may be useful. I like the inclusion of the videos & quiz. • Chords – I like the videos in the Chords section showing you how the chord is meant to sound. • Songs & Tabs – more variety of songs. • Scales – very easy to use and find. You can easily read it. • FAQ – good details and easy to follow. • General feedback – Add in a few more pictures, good use of videos, very easy to follow and navigate.
H	• The plectrum message would benefit from rewording and spacing • Chords – videos effective, diagrams clear and very useful. • Songs & Tabs – very good section of the app. • Tuner – message informing user how to use the tuners. • FAQ – Equipment, add in an amp and internet resources.

As Table 9 demonstrates, the feedback provided by the respondents was both detailed, and helpful. While there are numerous instances of praise for the application, for which the developer is grateful, there are certain criticisms, or recommendations for improvement that have been suggested. Several of these are worthy of comment from the developer. Both Respondent A and B noted that in relation to the Tuner section of the app, once a user presses the button for a string, this sound will overlap with the sound of another string if pressed. This was originally the developers's intention, allowing the user to tune several strings at once. However, based on the feedback from the various questionnaires, it was decided to amend the code to ensure that when a user presses a new string, the previous sound file will stop playing, allowing the new sound file to play uninterrupted.

In relation to the feedback regarding the Lessons section, several of the comments are fairly minor amendments; therefore they will be implemented in the very near future. The re-ordering of the lessons is a justified criticism, as some easier lessons are placed after more advanced lessons in each section; therefore a minor re-ordering of the lessons is a straightforward task. The same can be said of the introduction of more images and diagrams into these lessons. When the developer was having issues with the growth of the DVM heap, and the associated OutOfMemory error, which is exacerbated by the use of images, it was decided to perhaps ration the number of images used. Now that this problem has been amended to a significant degree, the developer can return to a previous version of the app and retrieve these images for use in the final version. The comments regarding the supposed verbosity of some of the textual lessons are understandable, however this was only done as the developer wished to include as much information and assistance as possible to the user. An idea for future development may be to perhaps bullet point some of the content of these lessons, as suggested by Respondent B.

Several respondents also noted that they were unsure of the suitability of the background image used for each screen, claiming that it is too dark, and should be lightened considerably. However, the developer respectfully disagrees with these opinions. The darker background provides a clear contrast to the text and features on the screen, allowing them to stand out to all users. The legibility of the text becomes much clearer when a darker background is used, and, in the developer's opinion, the background image adds a level of professionalism, as well as a sleek aesthetic. However, if, in future development and evaluation, this issue continues to be raised, the developer would perhaps be more willing to make an amendment to the background image.

6.3.2 Future Recommendations Based On the Evaluation Feedback

The thorough feedback provided by the respondents allowed the developer to consider areas of future development and improvement in relation to the application. These recommendations shall be summarised, and analysed, below.

1. *Expansion of the Lessons section* – While the feedback for the Lessons section overall was very positive, several respondents noted the potential for further expansion of this section with additional lessons. These extra lessons would allow for more focus to be placed on specific topics. The introduction of these lessons would also drastically increase the app's appeal to beginner players, and to the general public.
2. *Reformat existing lessons* – The main issues the respondents had with the content of the lessons revolved around their presentation. The videos were being displayed on VideoViews that were too small, which is again a fair criticism, and the textual lessons were too verbose, with insufficient use of images and diagrams. The video dimensions are a straightforward issue to correct, as is the introduction of images into the app. In future development, it would be ideal for the developer to contact a professional guitar teacher in order to write additional lessons.
3. *More variety of songs in the Songs & Tabs section* – An element of the feedback obtained from the questionnaires indicated that one respondent was not satisfied with the number and variety of the songs and tabs available for selection. Given that there are only 15 songs currently available as this is an early version of guitarGURU, this feedback was to be expected. Alongside this, there is an obvious human element in the evaluation – people want songs that they know or like to be included. In future development, this section would be expanded to include significantly more songs, with a wider variety in terms of both genre and playing style.
4. *Make the fact that the plectrum is a feature more apparent to the user* – A frequent point raised by the majority of respondents was that it was not clear enough that the plectrum was in fact a button that provided guidance; they thought, particularly because of its association with the guitar theme of the app itself, that it was merely a decoration. In future development, additional effort is required to ensure that the plectrum catches the eye of the user, and to ensure that they are fully aware of its function and purpose.

6.4 Summary

In this chapter, the process through which the testing and evaluation of the application was carried out was discussed. The tests that occurred, as can be seen in Appendices 6, 7 and 8, were aimed at testing the functionality of the features, and of the SQLite database that is used throughout the app. Of the

1,570 tests that were carried out on each tablet, only 16 fails were recorded, giving a successful test rate of 98.98%. This result perhaps indicates good programming and design practices on behalf of the developer. Of the 16 fails, 15 were caused by the same problem, as when a user accessed their favourited songs, any song file that was playing did not stop playing. The remaining error occurred because the developer had recorded the video of a chord being played using incorrect finger positions. By using the command songMediaPlayer.stop() when the My Favourited Songs section is accessed, the sound file is stopped, and this problem is corrected. Regarding the incorrect video, a corrected video has been recorded and has replaced the incorrect video in the app. All of these errors were fairly straightforward to amend, and have now been corrected.

After testing was completed, and all corrections implemented, the application was evaluated by eight individuals, with a wide range of technical and musical ability. Fortunately, the results of this evaluation were very pleasing; with seven of the eight respondents stating that they would “definitely” use the app again. Very positive feedback was given for virtually every aspect of the application; however various recommendations and areas for improvement were ascertained from the opinions provided by the respondents. These recommendations have been noted, and implemented into the future development plans for the application. Overall, however, the developer is very pleased with both the process and the outcome of the testing and evaluation phases.

Chapter 7 – Conclusions

In this concluding chapter, the overall successes and failures of the application, and the development process itself, will be considered, along with the degree to which the developer satisfied the objectives that were set out in the early stages of the project. This will include, for example, any significant changes or alterations that were made to the application throughout development, and the reasons for doing so. As the developer would class themselves as a relative newcomer to software development, and the methods through which software is designed and developed, it is perhaps natural that there were missteps made, and aspects which the developer would do differently given the opportunity again. These aspects, along with any lessons learned, will be discussed, before all future development plans are examined. These plans will be a combination of new ideas, along with ideas that may have been abandoned during development, and the reasons for these decisions.

7.1 Review of the Project

7.1.2 The Project Objectives

In order to review both the application itself, and the manner in which it was created, it is important to consider the original objectives set out by the developer at the very beginning of the project. The five core objectives that were established will be analysed below, as well as the degree to which these objectives were satisfied.

1. ***Research guitar-focused apps currently available on the market, especially the market leaders, and analyse the strengths and weaknesses present within each one*.**
   The developer undertook significant levels of background research regarding the guitar-focused apps available on the market. Based on an analysis of the strengths and weaknesses of five of the most popular apps, it became clear that there were very few apps aimed at beginners, and if there were they were not free, and did not contain useful information. This research was pivotal during the design phase of guitarGURU, as it was clear that certain key features were absent from these applications; for example, video lessons, a tuner, and chord diagrams.
2. ***Create and dispense a questionnaire to individuals of various musical abilities, ranging from those that have never played the instrument to experienced guitar teachers. The responses from these questionnaires will assist in the design, development and implementation of the application.***
   In order to ascertain which features should be included in the application, it was deemed appropriate to seek the advice of qualified individuals, and of potential users. Therefore, a questionnaire was created and distributed to individuals with a range of musical ability; including a Professor in the music field, guitar teachers, and those who have never played guitar but are keen to learn. The questions focused on the features that these individuals would consider to be particularly useful to a beginner guitar player. The responses were tremendously helpful, and ultimately shaped the content of guitarGURU. For example, the use of video tutorials and chord diagrams, the inclusion of songs and tabs, and the ability to create an account all rated very highly in terms of importance. The developer, noting the emphatic support for these features, felt that their inclusion was necessary. As well as the features that

were planned at this stage, additional features were added at later stages that were not included in these initial plans, including the lesson tracking system and the quizzes.

3. ***Ensure that the features developed for the application use a variety of resources, in order to provide the best possible educational experience for the user.***
   Research into the most common educational practices employed when teaching a new skill clearly highlighted both the importance, and effectiveness, of the use of technology and multimedia. Technology Enhanced Learning, or TEL, makes the educational experience more enjoyable for the student, while also having remarkable effects regarding information retention. Therefore, the developer implemented multimedia throughout the various sections of the app; for example video files in the Lessons section and Chords section, and sound files in the Tuner and Songs & Tabs sections. The developer is satisfied with the educational qualities provided by the multimedia files, as are the respondents to the evaluation questionnaires, who were impressed with the quality of the education provided by each section, based on their feedback.
4. ***To develop a user-friendly, aesthetically-pleasing user interface that is easy to navigate, and provides clear and concise information and guidance to the user.***
   The developer is satisfied with the user interface in terms of its aesthetic qualities, especially in relation to the background image, and the contrast between this image and the features on screen. Generally, the evaluators agreed with this viewpoint, however some expressed concerns that the image was too dark. On the basis of positive feedback, and the developer's own opinion, these concerns have been noted, but not acted upon. In terms of navigation, the developer is also very satisfied with the final application. Although the choice was made to disable the device's back button due to the possibility of hardware failure, extensive navigational tools have been provided in the form of back and home buttons, and drop-down menus. The evaluators were also particularly pleased with the level of navigation provided.
5. ***Create another questionnaire for the purposes of evaluating the application, both in terms of its operation and its suitability for beginner guitar players. This questionnaire will be distributed to individuals who have agreed to test and evaluate the application.***
   Once the app was created, and tested by the developer, it was given to various respondents for evaluation purposes. While the evaluators were using the app, they were asked to answer various questions on a questionnaire presented to them. The questions related to the quality of the specific sections, the usability of the app, and their overall impressions of the quality and usefulness of the app to beginner guitar players. Thankfully, the feedback was overwhelmingly positive regarding virtually every question. However, recommendations for future development were also suggested, with these recommendations being examined later in this chapter.

7.1.3 Functional and Non-Functional Requirements

Alongside the objectives devised at the beginning of the project, a series of both functional and non-functional requirements were created after the initial questionnaire was dispensed to the various respondents. These requirements were drafted using the information provided by the respondents in conjunction with the research into the currently available applications. The extent to which the requirements have been fulfilled will be scrutinised below.

Functional Requirements

- *The application must provide an assortment of educational tools, ranging from video tutorials to sound files, and other multimedia* – Initially, the developer only intended to include video tutorials, however based on the questionnaire responses, various forms of lessons were to be included, including text and diagram lessons. This requirement has already been discussed in the previous section of the chapter, but in the developer's opinion this has clearly been satisfied, with both video and audio files being used in numerous sections throughout the application.
- *The application must provide the user with instructions and guidance when requested.* – Through the use of the plectrum button to provide information, and the explanatory error messages present throughout the app, this requirement has been met.
- *The application must provide songs and tabs for the user to play.* – The Songs & Tabs section of the app appears to be the most popular, based purely on the opinions of the evaluators. Fifteen songs were included, as well as accompanying audio files of the developer playing the songs, to allow the user to learn each song.
- *The application must allow the user to login and create an account.* – As well as the ability to both login and create an account, users can also track their lesson progress, favourite songs, and attempt to beat their high scores in each of the three quizzes.
- *The application must allow the user to "favourite" certain songs*. – This was originally one of the key features that were to be included in the app. By pressing the heart button, the user can favourite a song. This song will be added to the user's favourited songs list, which can be accessed through the My Account section, or by long clicking the heart button. An "Un-favourite" button is also present, allowing the user to remove a song from their favourites.
- *The application must store all personal information securely*. – The data stored in the database can only be accessed using the "Administrator" account, with the username and password being known only to the developer. A recommendation for future development, as shall be discussed later, would be to implement password encryption, to provide even further levels of security.
- *The application must be easy to use, and provide easy navigation throughout the app* – Again, the developer is satisfied with the usability of the app, and with the quality of the navigation permitted. The evaluators provide justification for this belief, as in relation to the questions surrounding the ease of navigation throughout the app, and the overall usability of the app, seven of the eight respondents selected the highest possible rating.

Non – Functional Requirements

- *Usability* – In relation to the overall usability of the app, the developer is very satisfied with the final application. As has previously been mentioned, seven of the eight respondents selected the highest rating available regarding the app's overall usability. Each of the sections also rated very highly in terms of their usability. Alongside this, usability also encompasses the extent to which the app can be used by people of all ages and abilities; therefore the user interface is relevant. When questioned about the legibility of the text and features on the screen, four respondents

selected "Good", while the remaining four selected "Very Good". In the developer's opinion, therefore, the app is suitable for people of all ages and abilities.

- *Reusability* – The user is free to use the application as regularly as they desire, with no foreseeable problems. Reusability also refers to the code itself, particularly in relation to consistency. Similar code is used in the design of the layouts, in order to ensure consistency, while the same code in a single Java class is used to deal with several different tasks. For example, all fifteen songs in the Songs & Tabs section can be displayed using the same code in a single Java class, as can the chords and scales in the Chords and Scales sections.
- *Ethical* – All data is secure, and can only be viewed by the developer using the Administrator account. In the future, password encryption and stronger data security should be implemented.
- *Portability* – The application has been tested on two different Android devices, and has proven to be fully operational on both. However, in the future the developer wishes to ensure that the application can be used on devices with different screen sizes, with layout files for each different screen size being designed to accommodate for this.
- *Reliability* – Due to the partial solution to the DVM heap growth problem, the application can be used for extended periods, with virtually no impact on performance, as memory is now freed once an activity is no longer in use. While the app is still very reliable because of this, a full solution to this problem is very much a priority in future development.
- *Robustness* – The app has been designed in such a way as to cope with any errors that may occur, particularly in relation to the database operation. For example, if a user tries to sign in using incorrect details, or favourite a song that they have already favourited, an error-specific message will be displayed, informing the user of the nature of the error they have committed.
- *Efficiency* – Based on the evaluator responses, and the developer's own opinion, guitarGURU is very responsive, and executes all requests with minimal waiting times. The evaluators were asked whether the app was responsive to all of their requests, and the responses were emphatic – six selected "Definitely", one selected "Quite", and one selected "Somewhat". Efficiency also refers to resource usage. The solution to the DVM heap issue ensures that the app operates efficiently, releasing memory occupied by a module when this module is no longer in use.
- *Modifiable* – The application is easily modifiable, using any Android IDE. Amendments to the source code are straightforward to undertake, as the developer has done on many occasions.

As can be seen in the analysis of the requirements and objectives listed above, the developer is satisfied that all of the requirements and objectives that were set out at various stages in the development process has been met, and in some cases, exceeded.

7.2 Critical Analysis of the Project

In this section of the Conclusions chapter, particular focus will be given to the general successes or failures that occurred throughout the development process, especially in regard to any aspects that could have been done differently.

The original concept behind the app was to develop an educational tool that would assist beginner guitar players in learning the instrument; one which could also be used in conjunction with the

traditional methods of education. Therefore, the features that were to be included within the app, as well as the device on which it would operate, were two crucial factors to be considered at an early stage. The features would be based on the background research and questionnaire responses, which shall be discussed later, while in relation to the device, it was decided that a tablet is the most appropriate device. The major advantage tablets have over smartphones relates to the screen size; the much larger screen size available with tablets allows for the features to be displayed with more clarity. This would be particularly useful for the Scales, Chords, and Songs & Tabs sections, in which the developer has to read the text present on screen. The app was specifically designed for a tablet with a screen size of 7 inches. This is due to the fact that the developer's own tablet, a Samsung Galaxy Tab 4 7", has these screen dimensions, allowing the developer to undertake thorough testing of the app on an actual device, rather than an emulator.

In general, the developer is pleased with the application that has been created, and believes that it satisfies the core aim of the project itself – to provide an educational tool that will help beginner guitar players. In order to get to the stage where the application is now completed, a tremendous amount of work was required, with this work being divided into phases. The choices made and actions taken in each of these phases shall be critically analysed below.

7.2.1 Background Research

Before any work relating to the actual app itself was undertaken, planning and research was necessary to ascertain what features and functions were to be considered essential within an app that would be aimed at beginner players. Therefore, five of the most popular guitar-focused apps available on the market were downloaded, and their strengths and weaknesses were analysed. It quickly became apparent that almost all of them were much too advanced, and the ones that were not provided little useful information. Alongside this, research regarding the traditional methods of education was undertaken, particularly in relation to the significant flaws associated with these approaches, including the cost of lessons and the associated exclusion of those from lower-income households, and the incomplete education that is synonymous with self-education. Once this research was complete, contact was made with ten individuals who possessed a wide range of musical ability. They were asked for their opinions regarding the features and concepts that they believed should be included in an application of this type. Their responses, along with the research into the currently available apps and the traditional educational methods, allowed the developer to compile a list of features that should be implemented.

Overall, the developer is very satisfied with the level of background research that was undertaken prior to designing the app. This research allowed for the weaknesses present in currently available apps to be addressed in guitarGURU, and the feedback from qualified teachers and potential users allowed the developer to ensure that the correct information and features were included, thus making the app as appealing as possible to the general public. However, if the developer was permitted the chance to undertake this phase again, a greater sample size in terms of questionnaire respondents would be sought. While the responses from the ten individuals were excellent, a greater variety of opinions would have been significantly more helpful. As well as this, different questionnaires would be developed for specific groups of people, based on their musical ability. For example, a questionnaire would be dispensed to all guitar teachers that would focus very specifically on the content that should be included

in the lessons, and what features are necessary for beginner players, while a different questionnaire focusing much more on the features that would tempt a very recent learner or non-player to use the app would be distributed to those who fall under this category. However, the developer is satisfied with the responses overall, and is grateful to the respondents for giving their time and feedback.

7.2.2 Design

After close communication with the project supervisor, the developer was ready to proceed to the design phase. At this stage, the user interface was designed for several screens using Android Studio, in order to provide a visual demonstration of how the user interface would look in the actual prototype. Alongside this, storyboards were also designed in order to demonstrate a basic layout of the user interface, as well as the functions that would be associated with each feature. The original application architecture and Java classes required were also designed, as was the initial database table structure. The UI itself, which was heavily influenced by Shneiderman's eight golden rules and Nielsen's ten heuristics, originally featured an image of a brown guitar as a background, with white text and white buttons. While the majority of screens were aesthetically very pleasing, problems became apparent regarding the legibility of the text. The white text became difficult to read against the lighter sections of the background image. After consultation with the project supervisor, it was decided that the design of the UI should be amended, with this image being substituted for the current black and white image of a guitar. This image provides a much greater contrast to the white text, and white and blue buttons. As well as this, in the developer's opinions, and those of the evaluators, the user interface is both aesthetically attractive, and allows for all text and features to be easily seen.

When analysing the design phase, the developer is again satisfied with the manner in which this phase was undertaken. The UI was well planned and researched, especially regarding the adoption of the philosophies of Shneiderman and Nielsen, while the storyboards and architecture were well designed. However, the developer learned, to their detriment, the importance of well-considered and thorough planning. When designing the initial database tables, the developer had not yet studied databases as a module, therefore, in hindsight, there are notable errors with this initial design. Perhaps greater focus should have been given to studying the operation of databases prior to designing the tables. If the design phase could be undertaken again, more thought would be put in to the features that were to be included from the beginning. This point shall be expanded when discussing the implementation phase, but there were features added during the implementation phase that were not even considered during the design phase. The potential impact of such additions on the app as a whole should have been considered during the design phase, when the planning of the overall app was undertaken. This is a vital lesson learned by the developer; one which will be carried with them into future development projects.

7.2.3 Implementation

Once all design had been completed, the various features were implemented into the application. It was during this phase that the adoption of the Agile methodology became particularly useful, as at the termination of a sprint the developer could "re-prioritise work" (CPRIME, 2016), and ensure that more complex or important aspects could be focused on. This, along with weekly meetings with the project supervisor, allowed for the developer to ensure that all targets were met before the self-imposed deadlines, and to ensure that the implementation was proceeding as planned. Initially, the developer

decided to focus their attention almost solely on the development and implementation of the SQLite database into the application. As the developer had no previous experience with Android databases, it was assumed that two to three weeks would be required in terms of a timescale for the completion of the database, including research and education. Fortunately, a YouTube tutorial that addressed the key issues that were required by the developer was discovered, allowing for these basic ideas to be adapted to suit the needs of the application, and implemented. Rather than the allocated two to three weeks, the database was implemented and operational within five days. This allowed the developer to spend this additional time constructively, deciding to add the lesson tracking system, and the quizzes. This required the database to be redesigned, with four additional fields being added to the Users table.

If the developer were to carry out this phase again, they would take much greater care regarding the timescales, and late additions of new features. The allocation period of up to three weeks was perhaps unnecessarily lengthy, especially given the fact that in reality the database only took five days to implement. The more significant problem, however, was the late additions of the lesson tracking system and quizzes. The developer is obviously proud of these features, particularly considering the very positive feedback received during evaluation, but these features should have been planned during the design phase, when the impact on the other features of the app caused by these additional features could have been considered. The developer quickly learned the importance of clear structure, both in terms of planning and the time-scale of a project. The arguably scattershot approach taken by the developer may not have had a negative impact on the app itself, but it is certainly not good practice. Once the database was implemented, the developer re-evaluated their targets, and the various other features of the app were implemented.

Regarding the Lessons section, it was decided that a variety of educational techniques should be employed, including video tutorials, and text and diagram lessons. Overall, the developer is satisfied with the quality of the Lessons section, and with the lessons themselves. The feedback from the evaluators indicates that they, largely, also agree that the lessons are very useful to a beginner player. Some criticism was made regarding the verbosity of the text lessons, with some claiming that the lessons were rather difficult to understand, and were too long. The developer is willing to admit that this is an entirely justified criticism. Perhaps the developer was using too much technical jargon, and was focusing too much on providing as much information as possible; therefore the lessons became rather too difficult to follow. However, this may be classed as an educational mistake rather than a developer error. It is perfectly conceivable that an educator in any field may provide too much information in their teaching, rather than simply providing the most important information. This is something that could easily be amended during future development.

One issue that was raised during evaluation was whether or not the use of the YouTube videos in the video lessons may contravene YouTube's copyright rules. The developer then explained to the evaluator that no copyright law has been breached, and the use of the videos in guitarGURU is covered by the "Fair Use" exemption, which allows for these videos to be used without the consent of the copyright holder, as the videos are being used for non-profit educational purposes, they are of a factual nature, they do not constitute the heart of the app, and the use of the videos does not affect the value of the videos themselves (YouTube, 2016). The developer is satisfied with the Lessons section, as are the

majority of the evaluators, but there is undoubted room for improvement in terms of both quality of information and the variety of lessons provided.

The Chords and Scales sections, in the developer's opinion, are well designed in terms of both structure, and effectiveness as an educational tool. The layout of the sections – displaying some common keys, then the chords and scales within that key – allows for the user to learn about keys and the chords and scales within each key, as well as forming an easy to follow, and visually clear, method in which to display all of the information a user may require. Originally, the chords and scales were displayed using images, however to display the finger positions more clearly, TextViews were used, allowing the user to change the size of the diagrams at will without affecting their clarity. In hindsight, the developer is particularly pleased with these sections, especially in relation to their use to all guitar players, not just beginners. One of the evaluators, a self-dubbed "intermediate" guitar player, claims that they found these sections very useful. The developer would change very little with these sections, and no major issues were highlighted by the evaluators. One minor amendment would be the recording of new videos for the Chords section. Although these videos were only recorded to demonstrate the sound and fret positions of a chord, the videos could also focus much more on the finger positions.

The most popular section of the app appears to be Songs & Tabs, and, in the developer's opinion, understandably so. The fifteen songs chosen are excellent songs for a beginner to learn, as they will test the player, without being too difficult to play. The ability to favourite songs, which will be discussed later, is also a hugely popular feature among the evaluators. As with Scales and Chords, the songs were also originally displayed as images. However, as has been mentioned earlier in the report, images exacerbated the growth of the DVM heap, and inevitably lead to an OutOfMemory error, causing the app to crash. The images to display the songs were naturally quite large, and at one stage the developer couldn't open all fifteen songs without an OutOfMemory error occurring. Very early in the implementation stage, after regular consultations with the project supervisor surrounding a solution to this problem, the developer decided to display the songs using TextViews, thus removing the images. The conversion to TextViews was rather time-consuming, however in hindsight it was undoubtedly the correct decision. Even with the OutOfMemory problem partially resolved, the use of TextViews is preferable to images. The lyrics and chords are displayed with perfect clarity, and the developer was able to change the colour of the chords to blue, thus differentiating them from the lyrics. The size of both lyrics and chords can also be amended easily by the developer, if necessary. In retrospect, again, there is little that the developer would change regarding the implementation process of this section. Perhaps the use of TextViews should have been adopted from the beginning, rather than images, thus avoiding the time spent researching a solution to the DVM heap growth caused by images. However, due to the additional time the developer had because of the quick database implementation, the conversion to TextViews did not have an impact on the overall timescale of the project. This taught the developer the crucial importance of allowing significant time before any deadline to appropriately deal with any error that may occur. However, overall, the developer is very pleased with this section.

Another section that was roundly praised by the evaluators was the Tuner section. Initially, the Tuner section contained only one tuner, standard, however the developer thought that additional tunings should be included, in case they are ever needed by a user. As standard is the most common tuning a

user will need, an additional feature can be accessed through the standard tuner, which uses a SeekBar to allow the user to "strum" the strings. This feature was particularly popular among those who used the app, and the evaluators. Overall, the developer is pleased with the Tuner section included in the final app, but if they could do it again, the Open C and Drop D tuners would have been included from the start, rather than a relatively late addition, as this involved redesigning the entire section, including a menu to display all of the tuners. This is again an indicator of the sometimes scattershot approach to the implementation of the app's features, and avoiding such an approach is desirable in future projects.

The FAQ section is arguably the most basic section in the app; displaying ten of the most common questions that a beginner may face, with these questions being split into four categories. While the section may be basic, it proved to be particularly popular with the evaluators, who considered both the questions and the answers to be very informative and very useful for beginner players. If the developer had to change anything about this section, perhaps additional questions could be added, but this is very much something that could be recommended for future development. Overall, the developer is satisfied with this section, and is pleased to hear that it could be useful to beginner players.

From the earliest conception of the project, the ability to create an account and have the details stored in a database was intended to be a core feature within the app. During the initial design phase, the database had three tables – Users, Songs, and FavouritedSongs, with the Users table being expanded once the lesson tracking and quizzes were introduced. Along with the lesson tracking and quizzes, the creation of an account allows users to favourite songs, if they have a particular affinity for that song, or wish to practice it at a later time. All things considered, the developer is extremely pleased with the manner in which the ability to create an account, and to access the associated additional features, was implemented into the app, and is also pleased with the effect that this has had on the final version. The element of personalisation that this provides, as well as the additional features afforded to account holders, makes creating an account very appealing. This idea is supported by the evaluators, who were asked to rate the likelihood that they would create an account if they were a regular user of the app. Seven of the respondents stated that they would "Definitely" create an account, with the remaining respondent claiming they would be "Quite" likely to create an account. The additional functionality in various sections of the app offered by the implementation of the database, and the creation of an account, are a source of great pride for the developer.

There are few changes the developer would make in relation to the database and Account section. Perhaps stronger data protection systems could have been put in place from the early stages of implementation, as well as some manner of password encryption. Aside from these changes, which will be recommended for future development, the developer is very pleased with both the implementation process, and the effect that the database has had on the application as a whole.

7.2.4 Testing and Evaluation

The testing carried out by the developer upon the completion of the implementation phase was thorough, and through the 1,570 tests that were carried out, virtually every section and aspect of the app was scrutinised. Few defects were encountered, perhaps due to the regularity with which testing occurred under the Agile model, ensuring that any bugs that were unknowingly created were addressed

at an early stage. The developer is satisfied with the testing that was undertaken; it was thorough, and tested every facet of each section, including navigational tools, media player buttons, and various problems that could be encountered when entering data, for example incorrect details, or blank fields. An important lesson learned, however, relates to the crucial importance of testing itself. Of the 1,570 tests carried out, only 16 defects were recorded, and they were all fairly minor defects that may not have been spotted unless thorough testing had been undertaken. Therefore, the results of the testing process, and the manner in which it was carried out, are pleasing to the developer.

The feedback provided by the evaluators was incredibly detailed, and helpful, as well as being crucial in shaping the recommendations for future development. The evaluation itself was also hugely important in providing supporting evidence for the successes or failures of certain aspects of the application. Thankfully, based on the overwhelmingly positive responses to the questionnaires, guitarGURU can be considered to be an overall success. If the developer could carry out evaluation again, a greater sample size of evaluators would be sought. Although the feedback garnered was excellent, a wider variety of opinions is always more useful. However, the developer is satisfied with the manner in which the evaluation phase was executed, and is grateful to the evaluators for their assistance.

7.2.5 Summary

Overall, the developer is very pleased with the manner in which the development of guitarGURU occurred. It was a steep learning curve at the beginning; however the developer quickly learned the importance of careful and meticulous planning, and the extent to which this can be of assistance in software development. Some of the arguably questionable decisions made by the developer, such as the late introduction of new features, have taught the developer about the crucial importance of having a clear and well-defined plan. Fortunately, the sometimes haphazard approach adopted did not negatively affect the development process; however it is not a desirable practice. Therefore, if the developer were to undertake this project again, significantly greater levels of planning would be carried out prior to development. This lesson will prove to be extremely helpful in future development projects.

7.3 Recommendations for Future Development

During the design and implementation of the app, certain features or ideas were abandoned in favour of others that were deemed to be more appropriate at the time. Alongside this, possible areas of improvement, or sections that could be expanded, were suggested during the evaluation phase. Some of these ideas may prove to be very useful in future development, as shall be discussed below.

- **Recording of own videos for lessons** – Originally, the developer wanted to record their own videos for the lessons section. However, this idea was correctly abandoned – the developer is not a professional guitar teacher, and does not possess the requisite knowledge to educate beginner players. However, in the future, the advice and guidance of a professional teacher is desirable, in order to draft the textual lessons, and perhaps record the video lessons.
- **Allow for the app to be used on devices with differing screen sizes** – As the app was designed for tablets with 7 inch screens, it will be of very little use to a smartphone owner. However, it is a priority for additional layout files to be created for each screen size, allowing the app to be used on a variety of devices with different sizes of screens.

- **Expansion of the application content** – While the developer is pleased with the content available in the application, there are clear possibilities for expansion of the various sections. For example, additional lessons, FAQs, and songs and tabs could be added, to provide an even greater variety of learning resources to assist beginner players.
- **Professional recording of all media files** - All media files, for example the recordings of the songs, and the audio files used in the tuners, were recorded using the developer's iPhone 4S. While the sound quality is perfectly fine, having these files recorded professionally would dramatically improve the sound quality, and the overall professionalism of the app itself.
- **Extend the Try It! feature to Drop D and Open C tunings** – Currently, the Try It! feature is only available in relation to the standard tuner. As it is the most common tuner that will be used, the developer did not feel that it was necessary to extend such a feature to the other tunings. However, this is a very realistic possibility for the future, and could be easily done.
- **Increased data security** – The administrator (developer) is the only individual who can see the data stored in the database at the current time, however data security can always be strengthened. Some form of password encryption appears to an ideal solution to this concern, and this will almost certainly be considered a priority in future development.
- **A final solution to the DVM heap growth issue** – This problem has largely been resolved already, to such an extent where it is not really a concern for the developer, but a final solution that will completely resolve this problem is vital, in order to ensure that the application is fully reliable, and the possibility of an OutOfMemory error occurring is virtually impossible.
- **A forum, and a way to contact a qualified guitar teacher** – The introduction of a forum will allow beginner players to discuss any problems they have, brainstorm solutions, and even provide helpful tips and techniques that were useful to them. Alongside this, the ability to contact a qualified teacher with any concerns that are not addressed in the app would be hugely helpful to beginner players. The teacher's contact details would never be released directly; rather the user would submit a question through a specific section in the app, along with their email address. The teacher would then direct their answer to the email address provided.

7.4 Summary

In retrospect, having considered the development process, along with the favourable feedback from the evaluators and the quality of the final application, the developer is satisfied with the application that has been created, and with the project as a whole. All of the objectives and requirements declared as essential early in the development process have been met. The official deadlines, as well as the self-imposed deadlines, have been met in good time. The original purpose of the project was to create an application that would help beginner guitar players to learn to play the instrument, and the developer believes that this objective has been satisfied. The developer is also pleased with the way in which the development was undertaken, and with the final result. With a view to improving the quality and content of the app, recommendations for future development have also been proposed for future versions, aimed at improving the appeal of the application to potential users.

References

AHAMED, S.S.R. (2009) Studying the Feasibility and Importance of Software Testing. *International Journal of Engineering Science and Technology.* Vol. 1(3), 2009, pp.119-128.

ANDROID COPE SNIPPETS (2015) *How to create a rounded corners button in Android*. [Online]. Available from: android—code.blogspot.co.uk/2015/01/android-rounded-corners-button.html. [Accessed 15th August 2016].

ANDROID DEVELOPERS CONSOLE (2016) *Android Studio – The Official IDE for Android.* [Online]. Available from: https://developer.android.com/studio/index.html. [Accessed 15th August 2016].

ANDROID DEVELOPERS CONSOLE (2016) *Managing Your App's Memory*. [Online]. Available from: https://developer.android.com/training/articles/memory.html. [Accessed 21st August 2016]

AYBUKE, A. and CLAES, W. (2006) *Engineering and Managing Software Requirements.* Springer Science & Business Media, 1st edn.

BABER, C. AND NOYES, J. (2003) *Interactive Speech Technology: Human factors issues in the application of speech input/output to computers*. 2nd edn. London: Taylor & Francis Ltd.

BBC NEWS. (2014) *Electric guitar overtakes violin in music lesson boom.* [Online]. Available from: http://www.bbc.co.uk/news/education-29117849. [Accessed 2nd July 2016].

BRANNON, M. (2011) *Bill Evans The Man With The Other Horn* [Online]. Available from: http://www.jazzreview.com/jazz-artist-interviews/bill-evans-the-man-with-the-other-horn.html. [Accessed 1st July 2016].

CAPTAIN, F.A. (2013) *Six-Step Relational Database Design: A Step by Step Approach to Relational Database Design and Development*. 2nd edn. Fidel A. Captain.

CICETTI, F. (2013) *Is Playing A Musical Instrument Good For Your Health?* [Online]. Available from: http://www.livescience.com/40597-playing-musical-instrument-good-health.html. [Accessed 30th June 2016].

CLIFFORD, C. (2014) *By 2017, the App Market will be a $77 Billion Industry (Infographic)* [Online]. Available from: https://www.entrepreneur.com/article/236832. [Accessed 3rd July 2016].

CPRIME (2016) *Benefits of Agile*. [Online]. Available from: https://www.cprime.com/resources/what-is-agile-what-is-scrum/. [Accessed 29th August 2016].

CUDJOE, D. (2016) *Teach Yourself To Play An Instrument*. [Online]. Available from: http://www.selfgrowth.com/articles/Teach_Yourself_To_Play_An_Instrument.html. [Accessed 2nd July 2016].

dafont (2016) *Walkway*. [Online]. Available from: www.dafont.com/walkway.font. [Accessed 15th August 2016].

DEPARTMENT FOR CULTURE, MEDIA AND SPORT (2015) *Taking Part 2013/14, Focus On: Free time activities* [Online]. Available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/413047/Yr_9_Free_time_activities_Short_Story.pdf. [Accessed 30th June 2016].

DOGTIEV, A. (2015) *App Usage Statistics: 2015 Roundup* [Online]. Available from: http://www.businessofapps.com/app-usage-statistics-2015/. [Accessed 3rd July 2016].

DREDGE, S. (2015) *Android In 2015: 1BN Users and 50BN Annual App Downloads* [Online]. Available from: http://musically.com/2015/05/29/android-2015-1bn-users-50bn-app-downloads/. [Accessed 25th July 2016].

EMARKETER (2015) *Tablet Users to Surpass 1 Billion Worldwide in 2015* [Online]. Available from: http://www.emarketer.com/Article/Tablet-Users-Surpass-1-Billion-Worldwide-2015/1011806. [Accessed 26th July 2016].

FONTANA, T. (2012) *The 5 Biggest Challenges of Learning the Guitar.* [Online]. Available from: http://www.theguitarlesson.com/guitar-lesson-blog/beginner-guitar-lessons/5-biggest-challenges-learning-guitar/. [Accessed 2nd July 2016].

FREEMAN, R. (2016) Why Is Teaching Yourself How To Play The Guitar Actually Preventing You From Getting Better?. [Online]. Available from: http://www.tunedinguitarlessons.com/whyisteaching-yourself-how-to-play-the-guitar-actually-preventing-you-from-getting-better. [Accessed 2nd July 2016].

GEGGEL, L. (2015) Severe Stress and Depression Increase Risk of Early Death [Online]. Available from: http://www.livescience.com/50101-stress-depression-early-death-risk.html. [Accessed 1st July 2016].

GHAHRAI, A. (2008) *Seven Principles of Software Development*. [Online]. Available from: http://www.testingexcellence.com/seven-principles-of-software-testing/. [Accessed 20th August 2016].

GORDON, N. (2014) *Flexible Pedagogies: technology-enhanced learning* [Online]. Available from: https://www.heacademy.ac.uk/sites/default/files/resources/tel_report_0.pdf. [Accessed 2nd July 2016].

HARVARD MEDICAL SCHOOL (2009) *Using music to tune the heart* [Online]. Available from: http://www.health.harvard.edu/newsletter_article/using-music-to-tune-the-heart. [Accessed 1st July 2016].

HARWANI, B.M. (2013) *Android Programming Unleashed*. 1st edn. Indianapolis: Pearson Education Inc.

HESS, T. (2016) *Do You Really Need a Guitar Teacher?* [Online]. Available from: https://tomhess.net/Articles/DoYouReallyNeedATeacher.aspx. [Accessed 2nd July 2016].

HUGHES, N. (2014) *Current tablet sales growth being driven by sub-$250 devices, IDC says* [Online]. Available from: http://appleinsider.com/articles/14/11/14/current-tablet-sales-growth-being-driven-by-sub-250-devices-idc-says. [Accessed 26th July 2016].

INCORPORATED SOCIETY OF MUSICIANS. (2015) *Fees for private music tuition: our survey results.* [Online]. Available from: http://www.ism.org/advice/article/private-music-teacher-fees. [Accessed 30th June 2016].

JISC DIGITAL MEDIA (2016) *Introduction To E-Learning*. [Online]. Available from: http://www.jiscdigitalmedia.ac.uk/guide/introduction-to-elearning. [Accessed 2nd July 2016].

KIRKWOOD, A. and PRICE, L. (2014) 'Technology-enhanced learning and teaching in higher education: what is 'enhanced' and how do we know? A critical literature review.'. *Learning, Media and Technology* 39(1), p. 6 – 36.

KUCHINSKAS, S. (2010) *How Making Music Reduces Stress* [Online]. Available from: http://www.webmd.com/balance/stress-management/features/how-making-music-reduces-stress. [Accessed 1st July 2016].

LIFETIME LIBRARY (2012) *The Value of Multimedia-based Learning* [Online]. Available from: http://lifetimelearning.com/the-value-of-multimedia-based-learning. [Accessed 3rd July 2016].

MANSFIELD, E. (2010) *Can You Teach Yourself To Play An Instrument?* [Online]. Available from: http://www.mansfieldmusic.com/can-you-teach-yourself-to-play-an-instrument/. [Accessed 2nd July 2016].

MATTHEWS, M. (2011) *18 Benefits of Playing a Musical Instrument* [Online]. Available from: http://www.effectivemusicteaching.com/articles/directors/18-benefits-of-playing-a-musical-instrument/. [Accessed 1st July 2016].

MURRAY, J. (2014) 'Pianist tackles 'shocking state' of school music lessons', *The Guardian,* 2nd September. [Online]. Available from: https://www.theguardian.com/education/2014/sep/02/james-rhodes-improve-music-education-schools. [Accessed 2nd July 2016].

OFCOM (2015) *Half of UK homes turn to tablets – in just five years* [Online]. Available from: http://media.ofcom.org.uk/news/2015/five-years-of-tablets/. [Accessed 26th July 2016].

PADDOCK PhD, C. (2013) *Music practice can sharpen the brain* [Online]. Available from: http://www.medicalnewstoday.com/articles/266809.php. [Accessed 1st July 2016].

RAJPUT, M. (2015) *Why Android Studio Is Better For Android Developers Instead Of Eclipse* [Online]. Available from: https://dzone.com/articles/why-android-studio-better. [Accessed 11th August 2016].

ROCKGUITARPOWER (undated) *The Top 10 Reasons Why Students Quit Taking Guitar Lessons*. [Online] Available from: https://www.rockguitarpower.com/the-top-10-reason-why-students-quit-taking-guitar-lessons/. [Accessed 1st July 2016].

STACKOVERFLOW (2010) *Android – Using Custom Font*. [Online]. Available from: https://stackoverflow.com/questions/3651086/android-using-custom-font. [Accessed 15th August 2016].

STANTON, N., SALMON, P., RAFFERTY, L., WALKER, G., BABER, C., AND JENKINS, P. (2013) *Human Factors Methods: A Practical Guide for Engineering and Design*. 2nd edn. Surrey: Ashgate Publishing Limited.

STONE, D. (2014) *10 Reasons Why You Should Learn To Play A Musical Instrument*. [Online]. Available from: http://www.normans.co.uk/blog/2014/04/10-reasons-learn-play-musical-instrument/. [Accessed 1st July 2016].

THE GUARDIAN. (2014) *Music education still preserve of the rich, UK study shows* [Online]. Available from: http://www.theguardian.com/education/2014/sep/15/music-education-instrument-rich-poor. [Accessed 1st July 2016].

thenewboston (2015) *Android App Development for Beginners – 49 – Saving Data with SQLite*. [Online]. Available from: https://www.youtube.com/watch?v=_vQaOvPsLko. [Accessed 14th August 2016].

traex (2014) *RippleEffect – Implementation of Ripple effect from Material Design for Android API 9+*. [Online]. Available from: https://github.com/traex/RippleEffect. [Accessed 15th August 2016].

UK GEOGRAPHICS. (2014) *Social Grade A, B, C1, C2, D, E* [Online]. Available from: http://www.ukgeographics.co.uk/blog/social-grade-a-b-c1-c2-d-e. [Accessed 6th July 2016].

ULTIMATE GUITAR (2015) *90% of New Guitarists Quit in First Year, and That's Why Fender Is Going Digital* [Online]. Available from: https://www.ultimate-guitar.com/news/general_music_news/90_of_new_guitarists_quit_in_first_year_and_thats_why_fender_is_going_digital.html. [Accessed 30th June 2016].

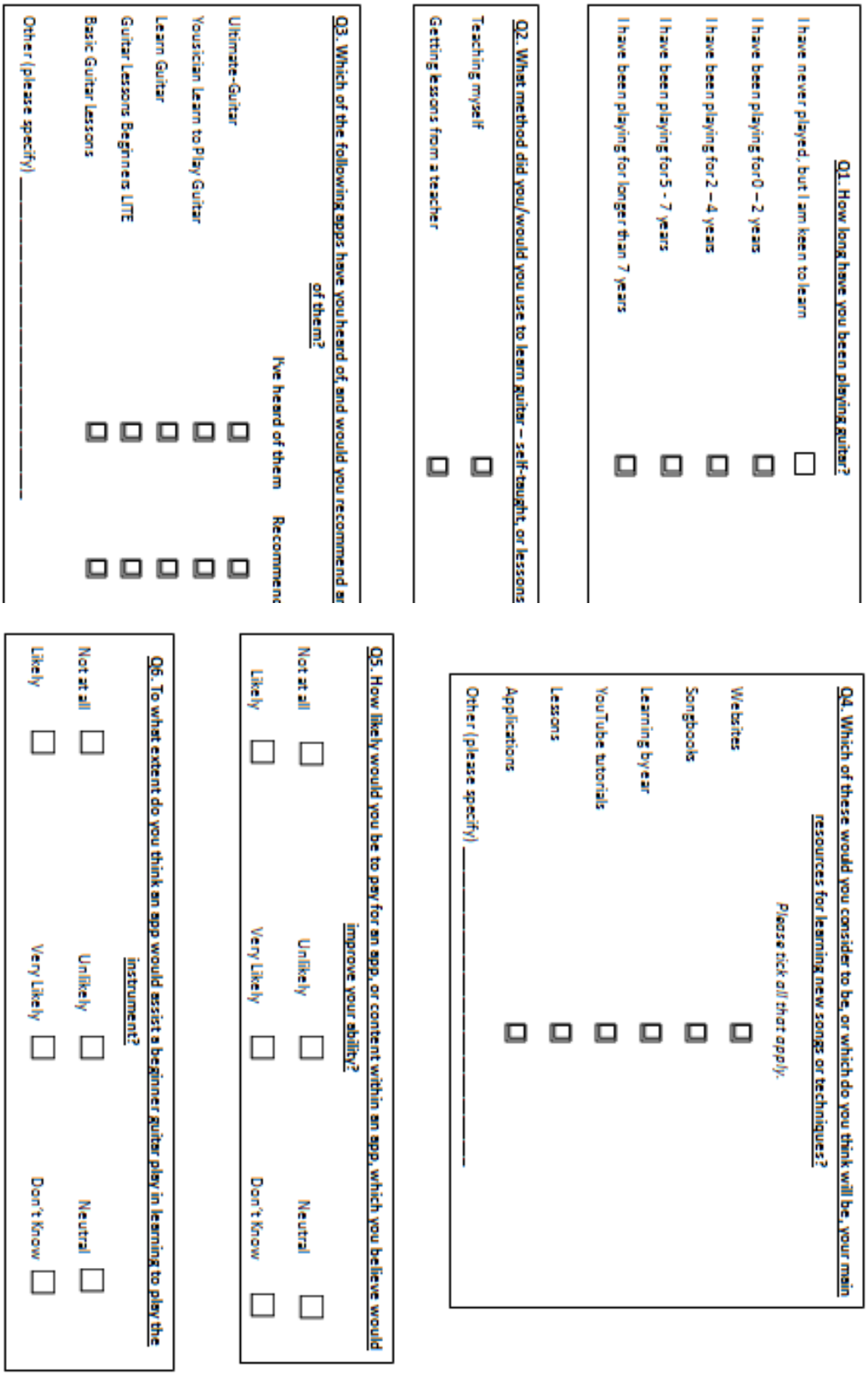
UNIVERSITY OF SHEFFIELD (2016) *E-Learning and Technology Enhanced Learning – Guidance for Academics.* [Online]. Available from: https://sheffield.ac.uk/lets/toolkit/teaching/e-learning/tel. [Accessed 30th June 2016].

VENTUREPACT (2015) *8 Tips and Best Practices for Mobile Application Development* [Online]. Available from: http://blog.venturepact.com/8-tips-and-best-practices-for-mobile-application-development/. [Accessed 25th July 2016].

WEBB, A. (2015) *Only £10 left – the disposable income of one in ten Brits* [Online]. Available from: https://www.moneyadviceservice.org.uk/blog/only-10-left-the-disposable-income-of-one-in-ten-brits. [Accessed 1st July 2016].

Appendices

Appendix 1 – Analysis Questionnaire



Questionnaire

Name: _____

Q1. How long have you been playing guitar?

- I have never played, but I am keen to learn
- I have been playing for 0 – 2 years
- I have been playing for 2 – 4 years
- I have been playing for 5 - 7 years
- I have been playing for longer than 7 years

Q2. What method did you/would you use to learn guitar – self-taught, or lessons

- Teaching myself
- Getting lessons from a teacher

Q3. Which of the following apps have you heard of, and would you recommend so of them?

	I've heard of them	Recommend
Ultimate-Guitar	☐	☐
Yousician Learn to Play Guitar	☐	☐
Learn Guitar	☐	☐
Guitar Lessons Beginners LITE	☐	☐
Basic Guitar Lessons	☐	☐

Other (please specify) _____

Q4. Which of these would you consider to be, or which do you think will be, your main resources for learning new songs or techniques?

Please tick all that apply.

- ☐ Websites
- ☐ Songbooks
- ☐ Learning by ear
- ☐ YouTube tutorials
- ☐ Lessons
- ☐ Applications

Other (please specify) _____

Q5. How likely would you be to pay for an app, or content within an app, which you believe would improve your ability?

Not at all		Unlikely		Neutral		Likely		Very Likely		Don't Know	
☐		☐		☐		☐		☐		☐	

Q6. To what extent do you think an app would assist a beginner guitar player in learning to play the instrument?

Not at all		Unlikely		Neutral		Likely		Very Likely		Don't Know	
☐		☐		☐		☐		☐		☐	

Q7. To what extent would you consider the following features to be important within a guitar tutor app?

Please tick the appropriate box.

	Not at all Important	Not Really Important	Neutral	Quite Important	Very Important	Don't Know
Advice on guitar models, strings, cases etc.						
Chords (diagrams, types, variations)						
Guitar tuner						
Database of beginner songs & riffs						
Scales diagrams & tutorial						
Login feature allowing songs to be saved						
Video tutorials – changing strings, song playing, instrument basics etc.						
Facility for user to record themselves strumming a chord and see if it matches chord						
Buttons which when pressed play the sound of a chord/string						
FAQ section for beginners						
Explanation of keys, and a facility to change the key within a song transcript						
Finger picking demonstration						
Finger technique exercises						
Strumming pattern exercises						
Metronome						
Video demonstration of chords that sound good together (eg 4 chord song – G, D, Em, C) & video tutorial on changing between chords						

Any additional features which you believe would aid in the education of a new guitar player (Please specify):

_____

_____

_____

Q8. On a scale of 1 – 5 (1 lowest, 5 highest), how significant would an element of personalisation be for an app? For example, a login feature which allows you to save certain songs you wish to learn.

1 ☐ 2 ☐ 3 ☐

4 ☐ 5 ☐ Don't Know ☐

Q9. Would you be willing to assist in the testing of the app once it is available?

☐ Yes ☐ No

If yes, please include your contact details:

_____

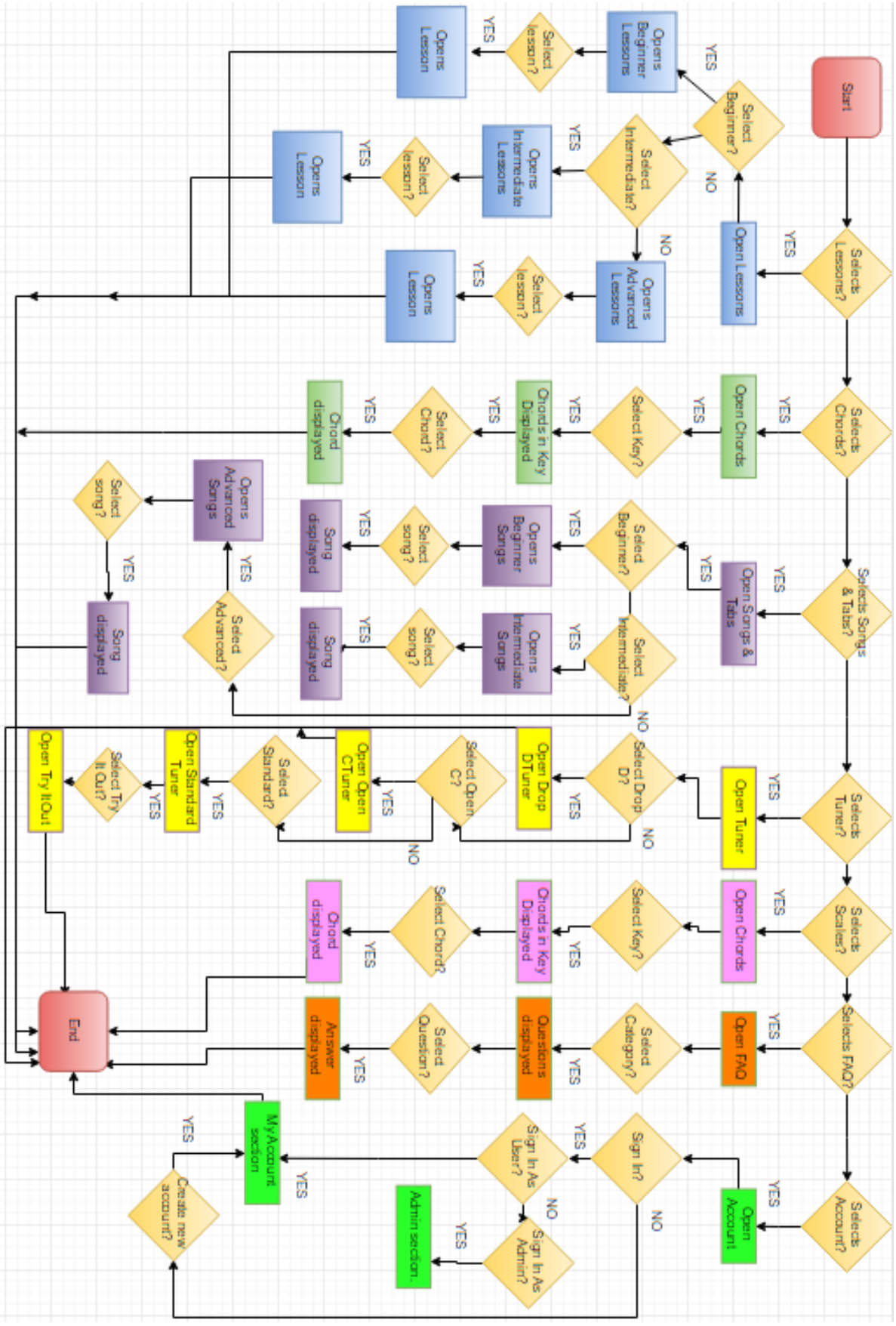
_____

_____

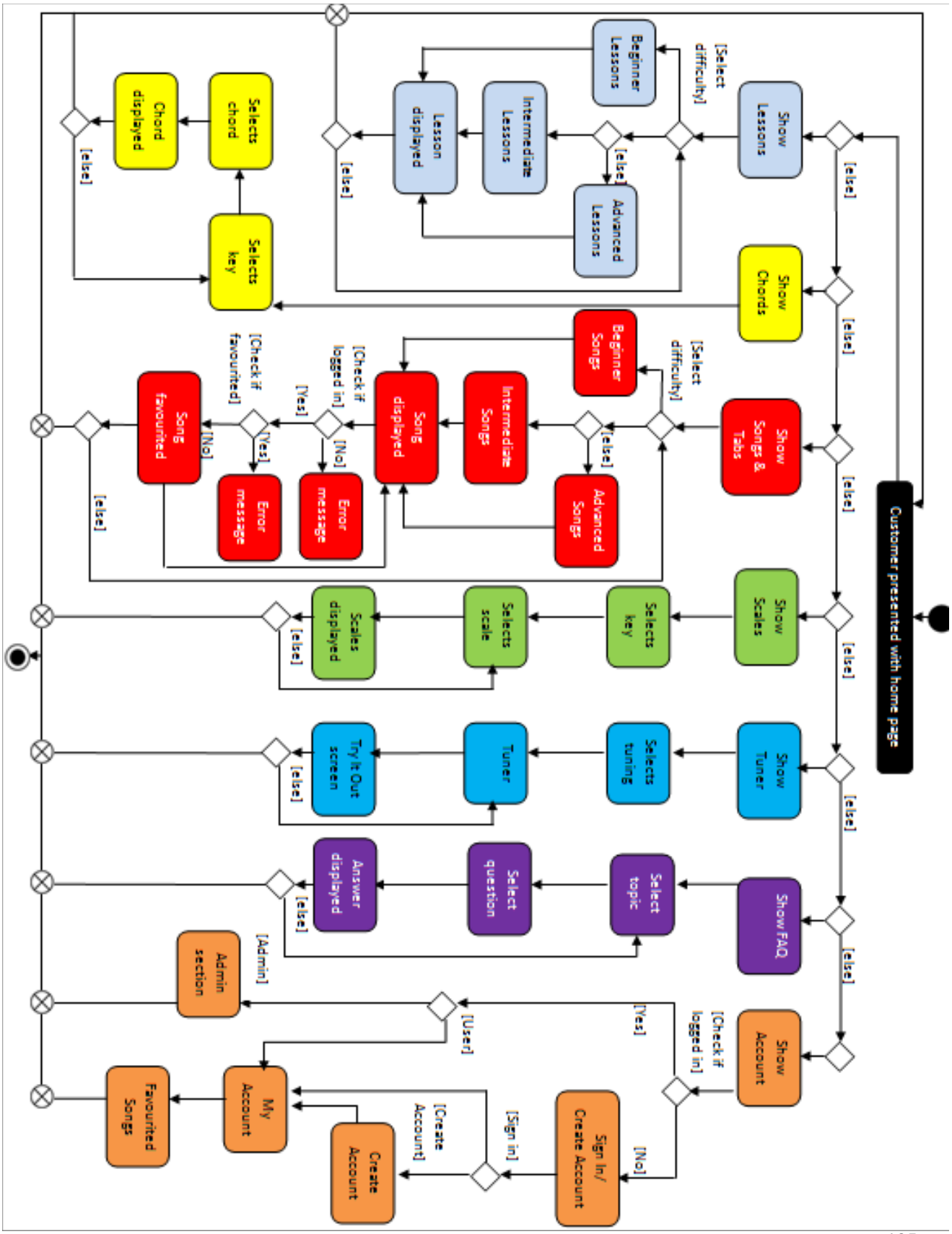
_____

Thank you for taking the time to complete the questionnaire. Your effort and co-operation are greatly appreciated.

Appendix 2 – Application Flow Chart

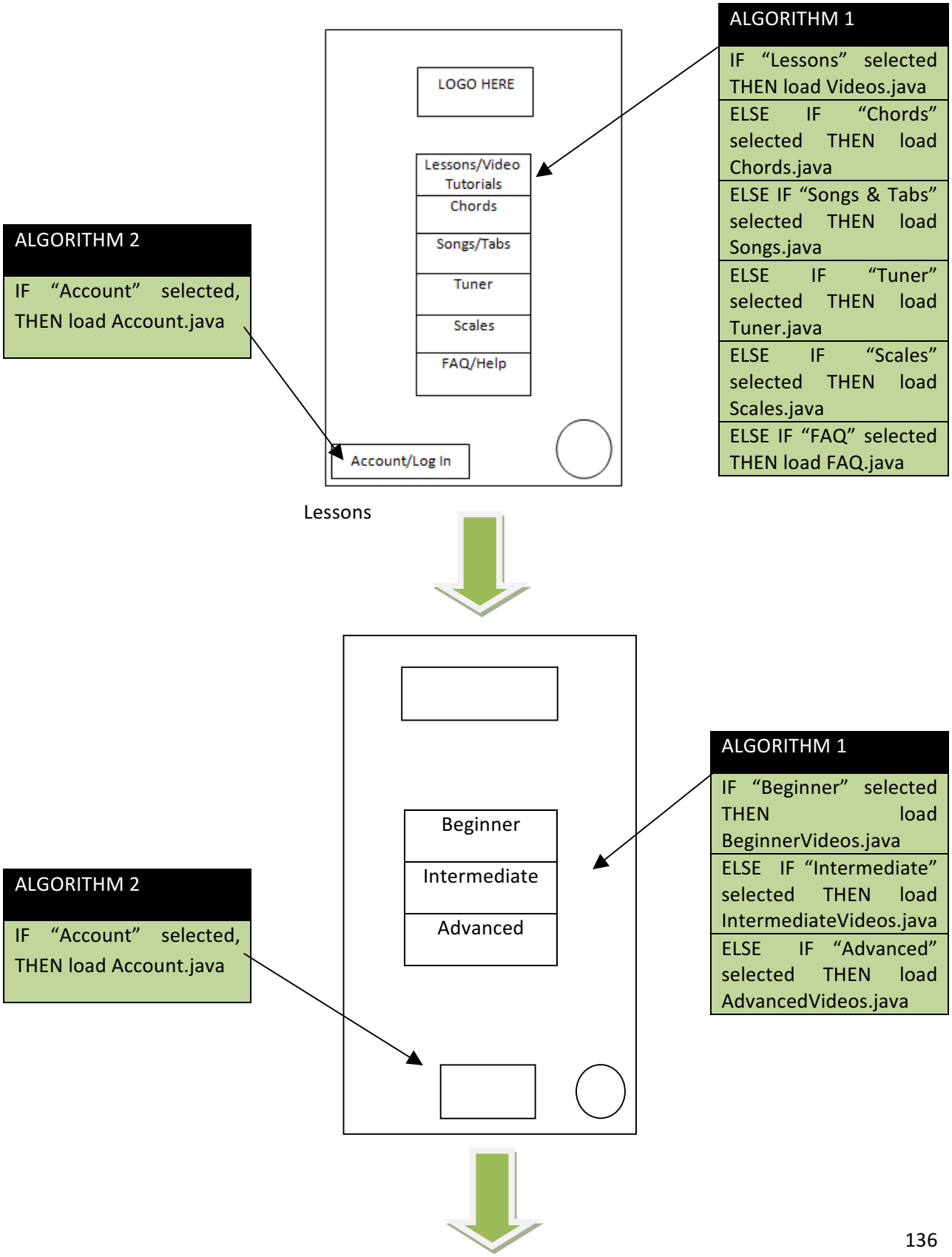


Appendix 3 – UML Activity Diagram

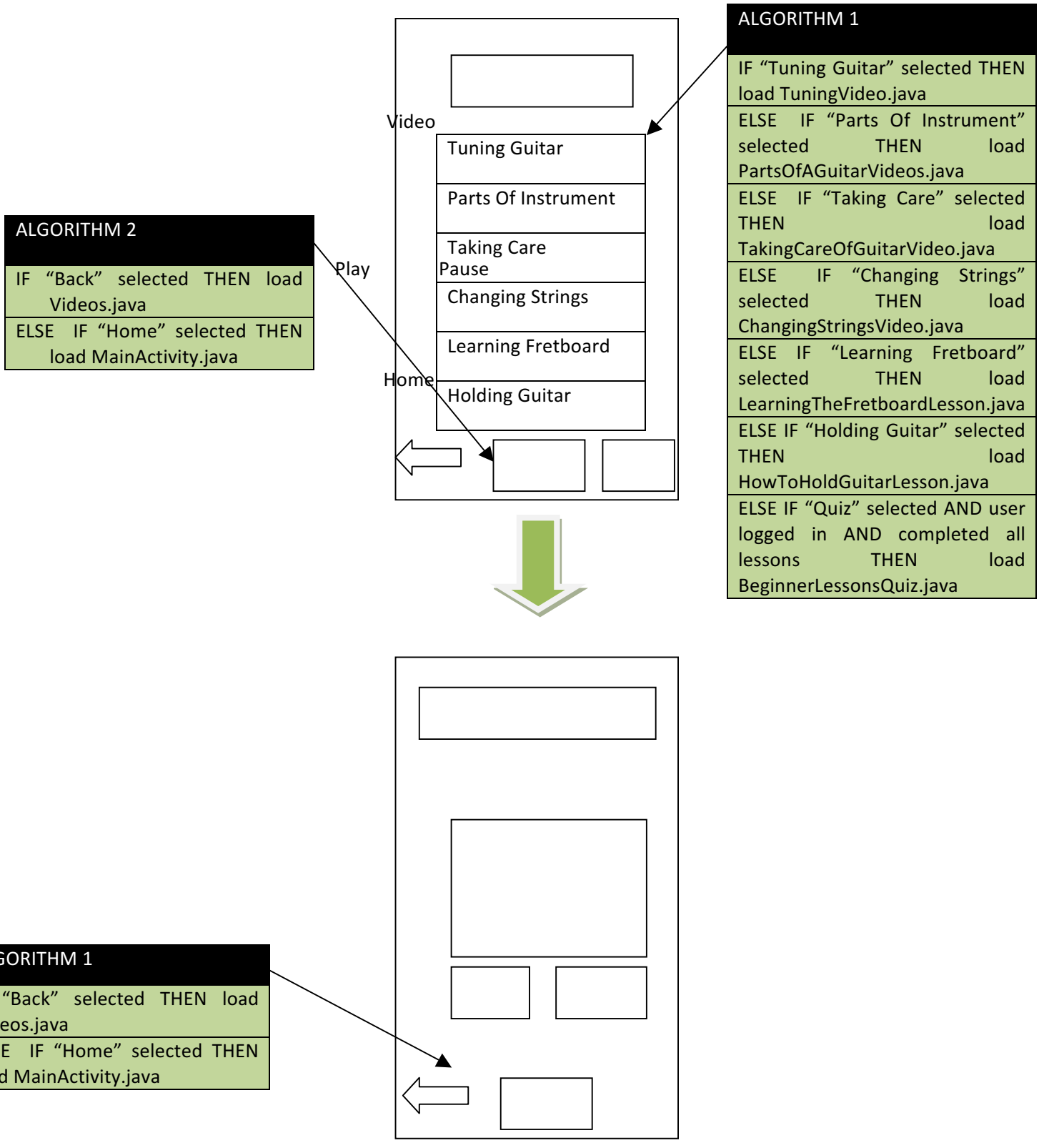


Appendix 4 – Example Section Storyboards

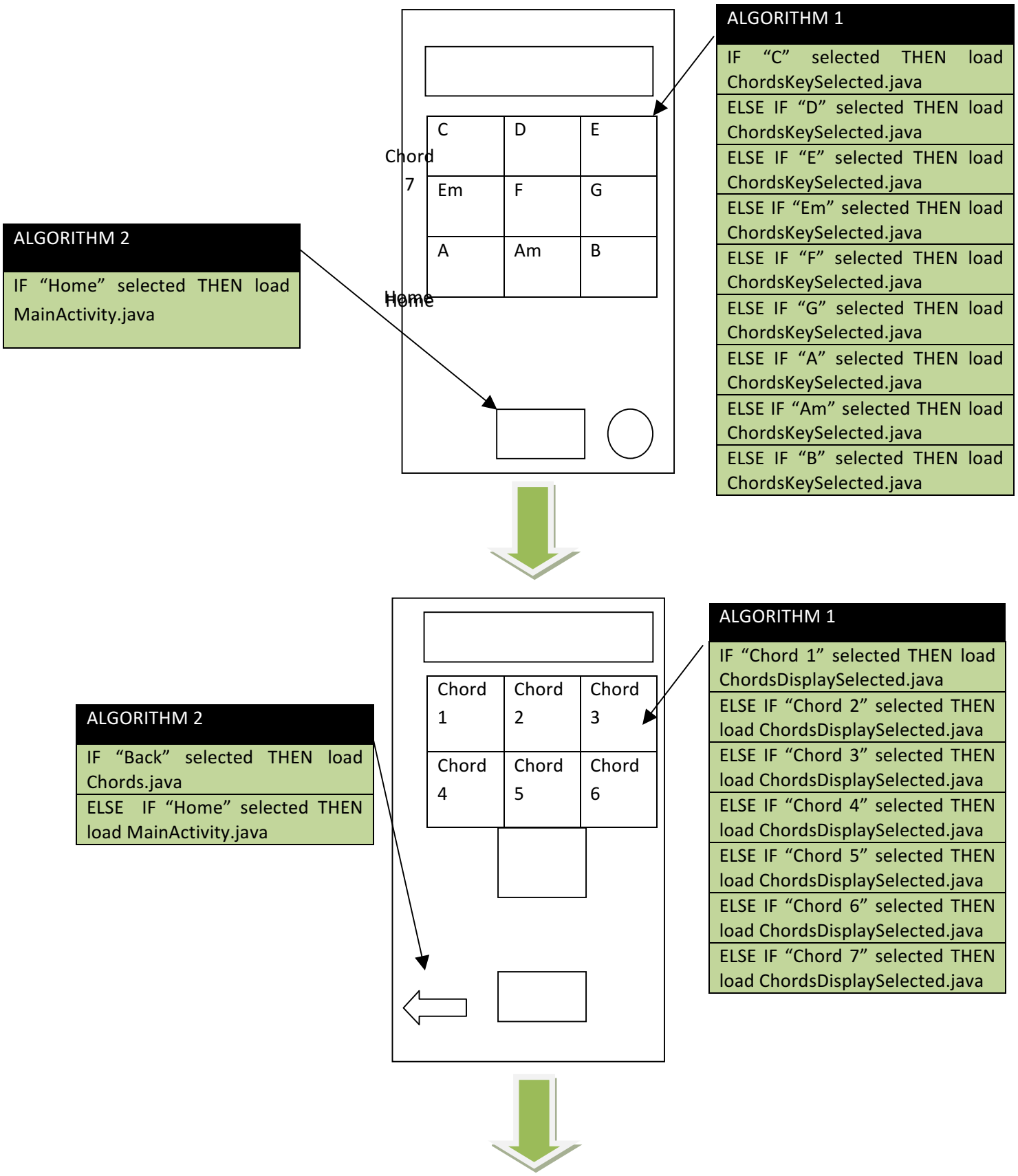
Lessons



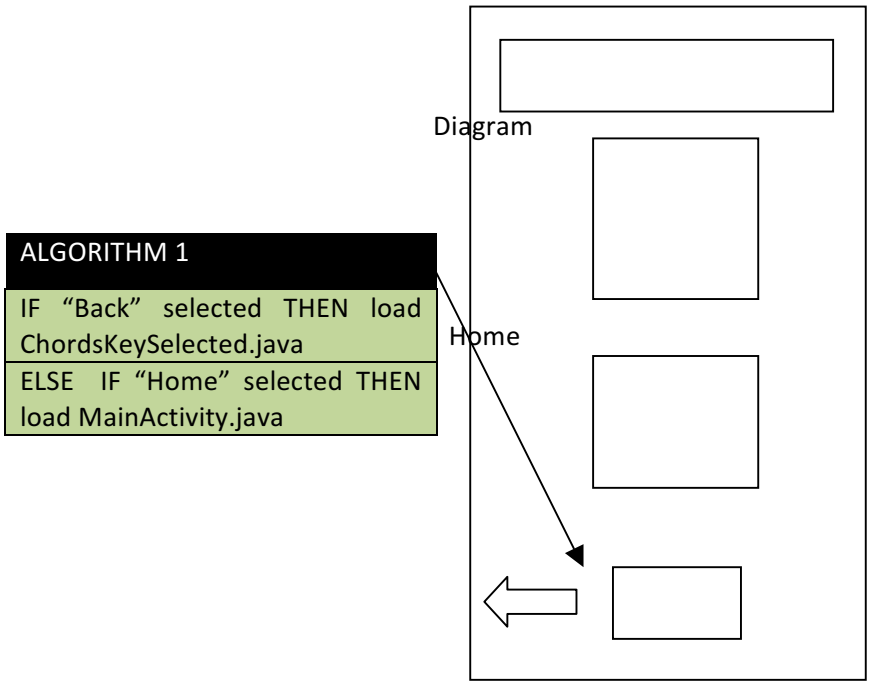
The Parts of the Guitar



Chords

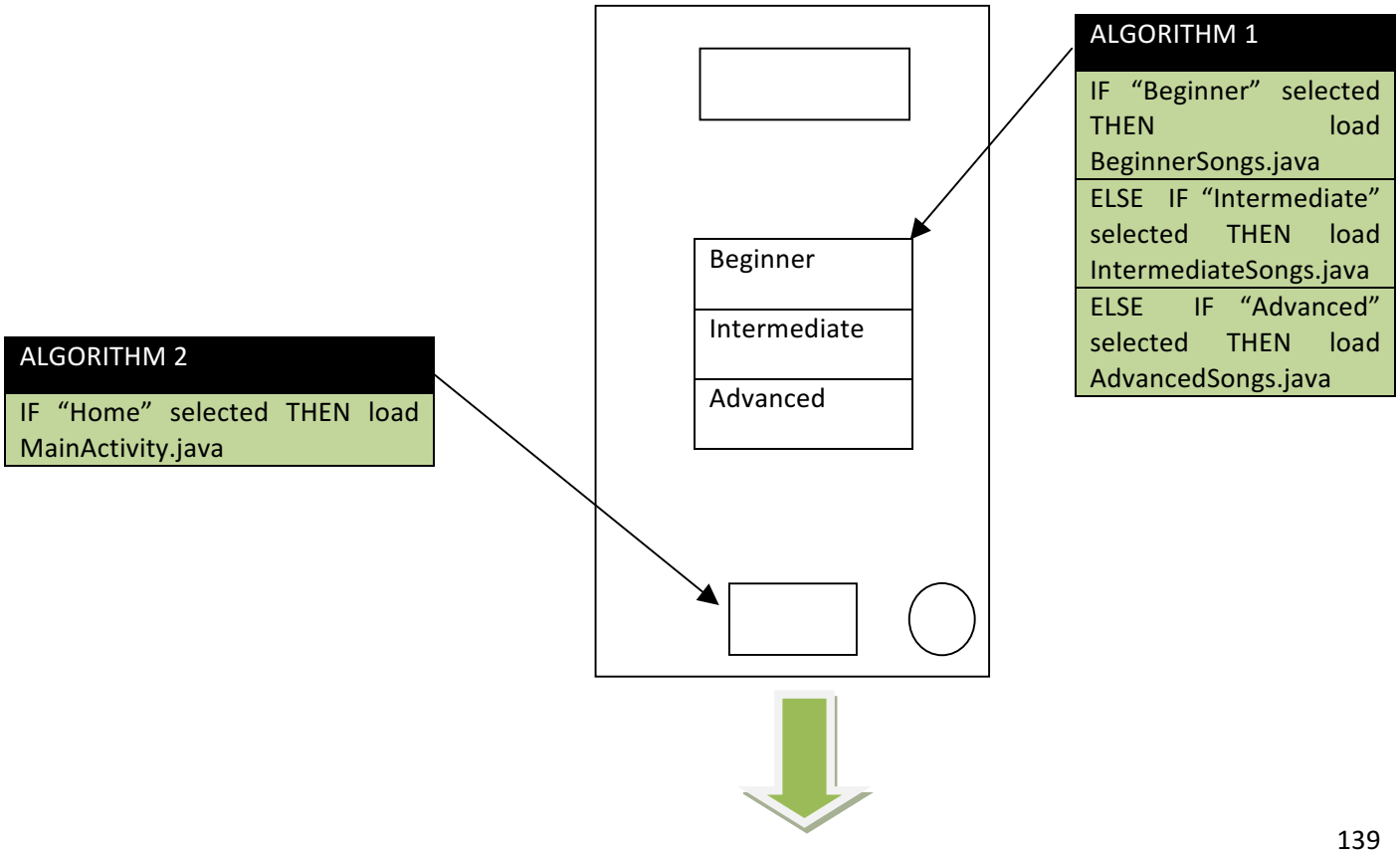

ALGORITHM 1
IF "C" selected THEN load ChordsKeySelected.java
ELSE IF "D" selected THEN load ChordsKeySelected.java
ELSE IF "E" selected THEN load ChordsKeySelected.java
ELSE IF "Em" selected THEN load ChordsKeySelected.java
ELSE IF "F" selected THEN load ChordsKeySelected.java
ELSE IF "G" selected THEN load ChordsKeySelected.java
ELSE IF "A" selected THEN load ChordsKeySelected.java
ELSE IF "Am" selected THEN load ChordsKeySelected.java
ELSE IF "B" selected THEN load ChordsKeySelected.java
C
D
E
Chord 7
Em
F
G
A
Am
B
Home
ALGORITHM 2
IF "Home" selected THEN load MainActivity.java
ALGORITHM 1
IF "Chord 1" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 2" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 3" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 4" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 5" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 6" selected THEN load ChordsDisplaySelected.java
ELSE IF "Chord 7" selected THEN load ChordsDisplaySelected.java
Chord 1
Chord 2
Chord 3
Chord 4
Chord 5
Chord 6
ALGORITHM 2
IF "Back" selected THEN load Chords.java
ELSE IF "Home" selected THEN load MainActivity.java

Songs & Tabs
Home
Video



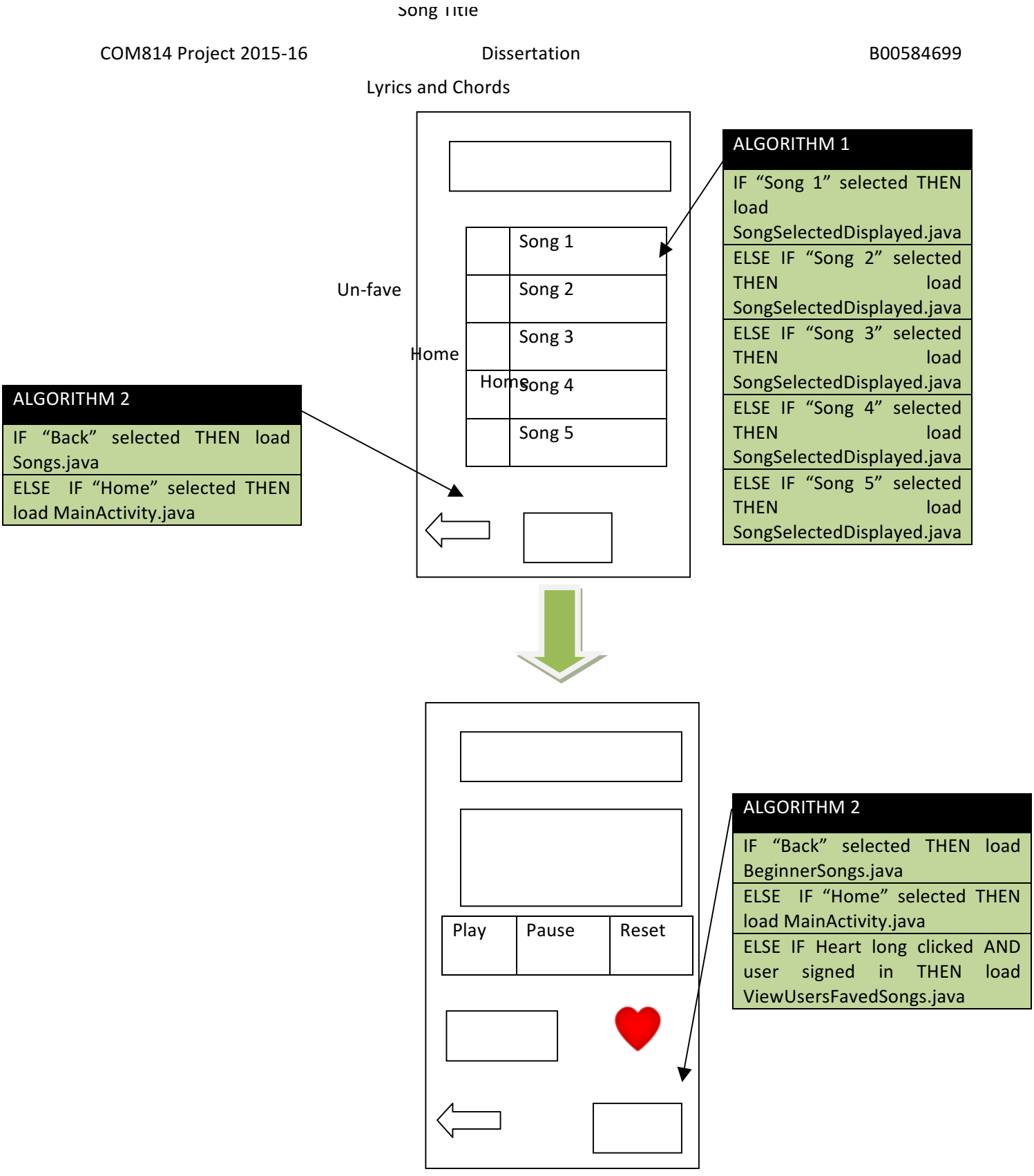
ALGORITHM 1
IF "Back" selected THEN load ChordsKeySelected.java
ELSE IF "Home" selected THEN load MainActivity.java

Songs & Tabs



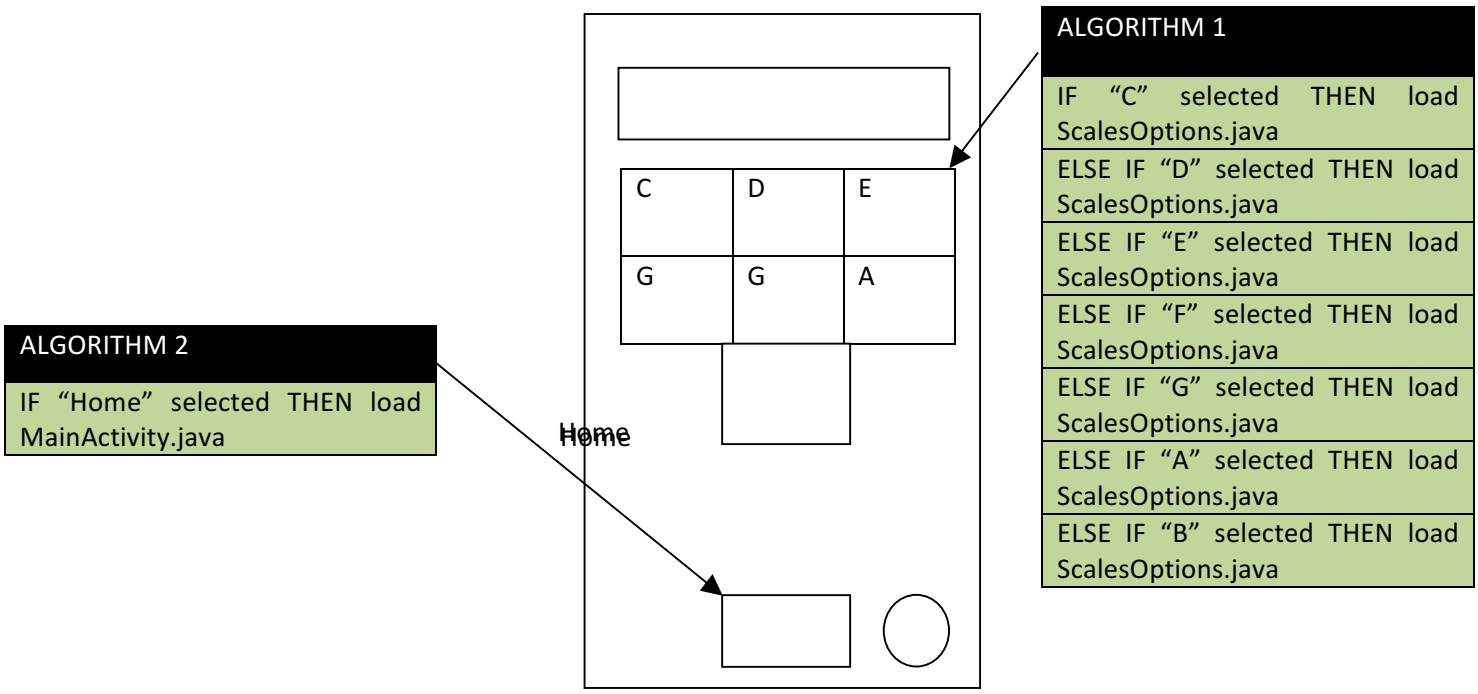
ALGORITHM 1
IF "Beginner" selected THEN load BeginnerSongs.java
ELSE IF "Intermediate" selected THEN load IntermediateSongs.java
ELSE IF "Advanced" selected THEN load AdvancedSongs.java

ALGORITHM 2
IF "Home" selected THEN load MainActivity.java


Song Title
Lyrics and Chords
Song 1
Song 2
Song 3
Song 4
Song 5
Un-fave
Home
Home
ALGORITHM 1
IF "Song 1" selected THEN load SongSelectedDisplayed.java
ELSE IF "Song 2" selected THEN load SongSelectedDisplayed.java
ELSE IF "Song 3" selected THEN load SongSelectedDisplayed.java
ELSE IF "Song 4" selected THEN load SongSelectedDisplayed.java
ELSE IF "Song 5" selected THEN load SongSelectedDisplayed.java
ALGORITHM 2
IF "Back" selected THEN load Songs.java
ELSE IF "Home" selected THEN load MainActivity.java
Play
Pause
Reset
ALGORITHM 2
IF "Back" selected THEN load BeginnerSongs.java
ELSE IF "Home" selected THEN load MainActivity.java
ELSE IF Heart long clicked AND user signed in THEN load ViewUsersFavedSongs.java

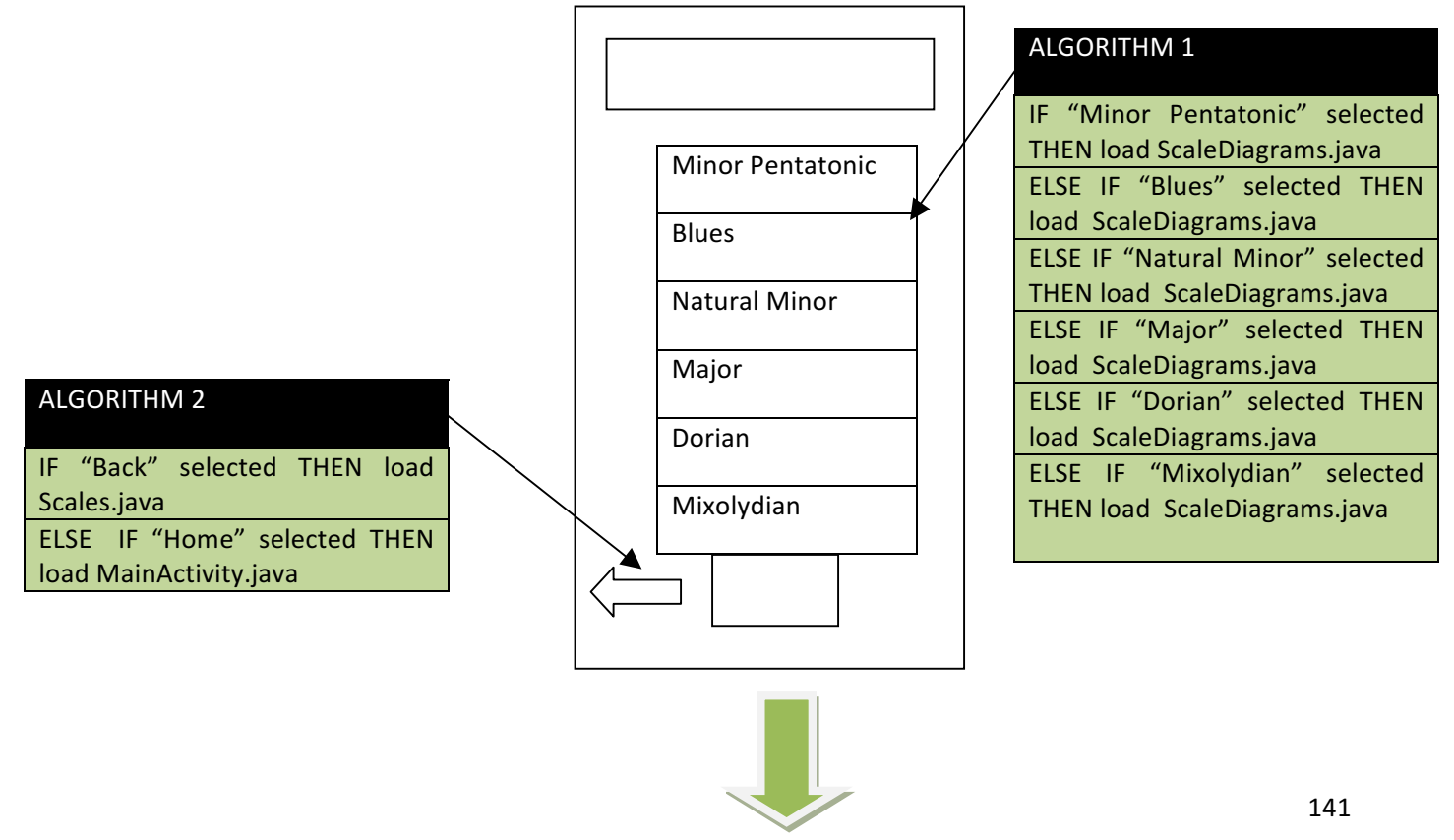
Scales In Key Of SCALE

Scales



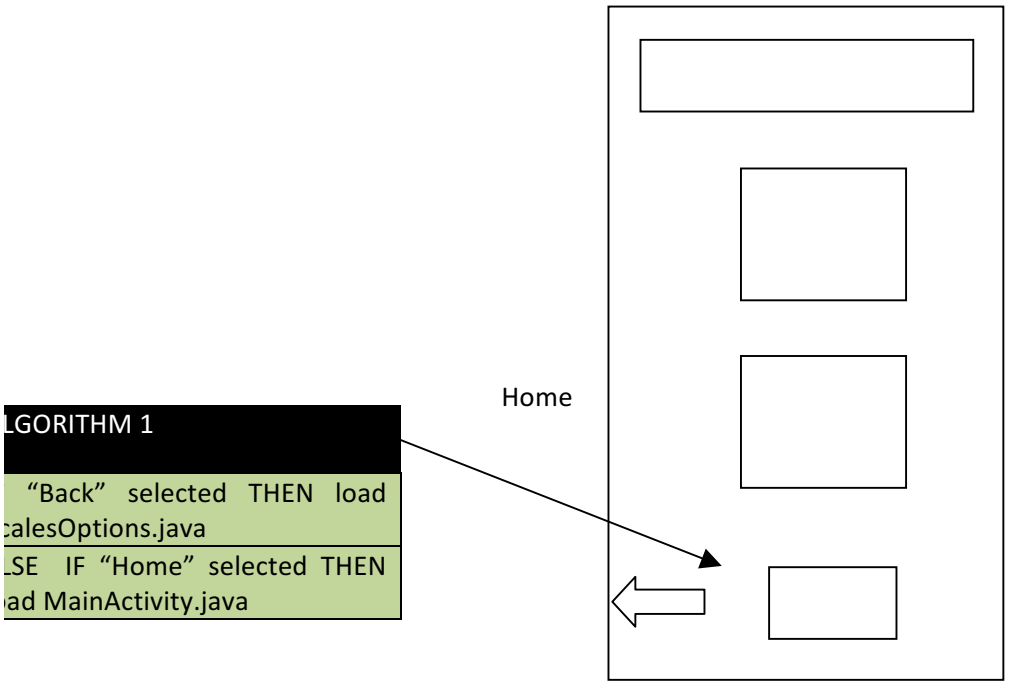
ALGORITHM 1
IF "C" selected THEN load ScalesOptions.java
ELSE IF "D" selected THEN load ScalesOptions.java
ELSE IF "E" selected THEN load ScalesOptions.java
ELSE IF "F" selected THEN load ScalesOptions.java
ELSE IF "G" selected THEN load ScalesOptions.java
ELSE IF "A" selected THEN load ScalesOptions.java
ELSE IF "B" selected THEN load ScalesOptions.java

ALGORITHM 2
IF "Home" selected THEN load MainActivity.java

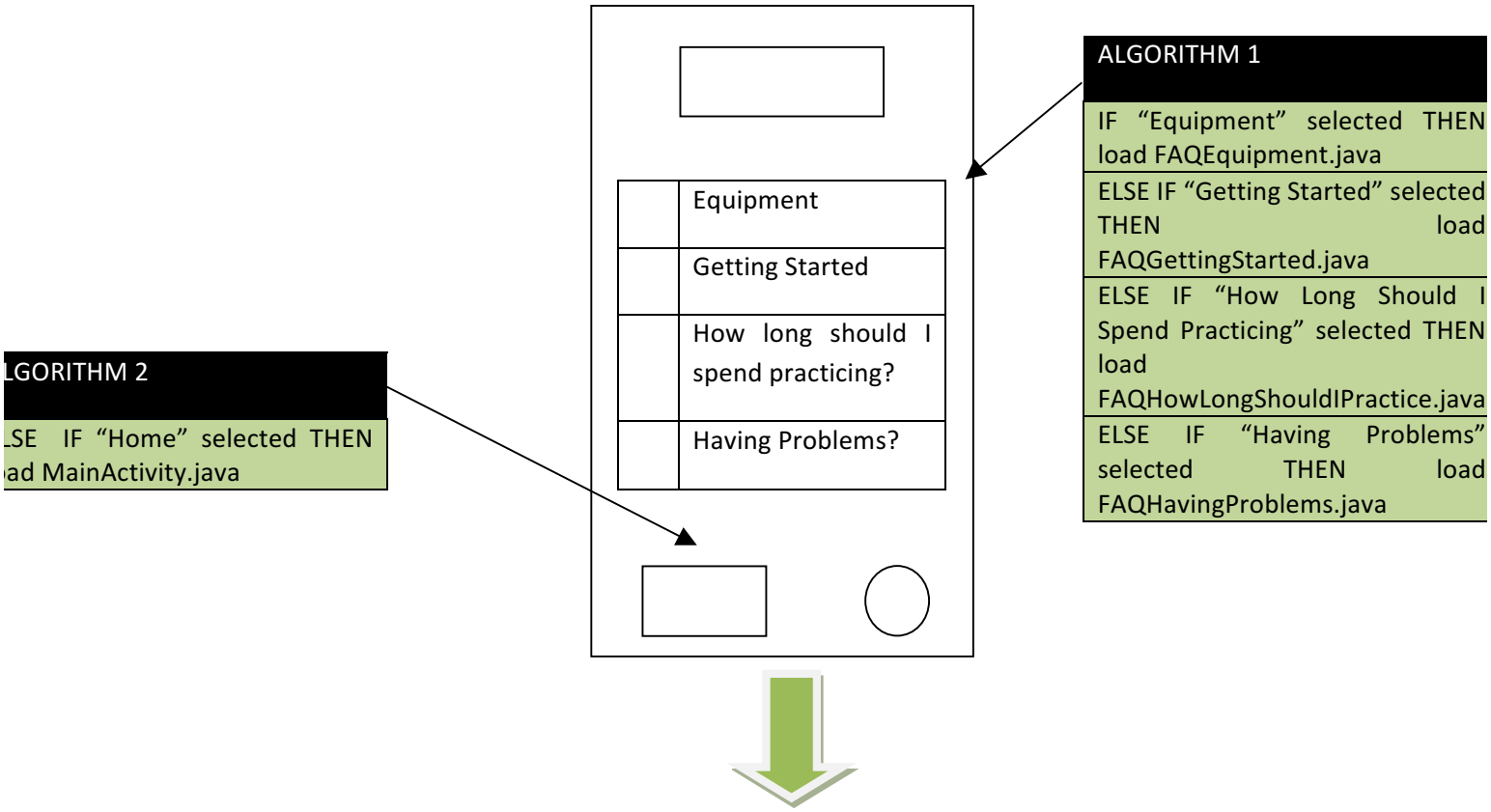


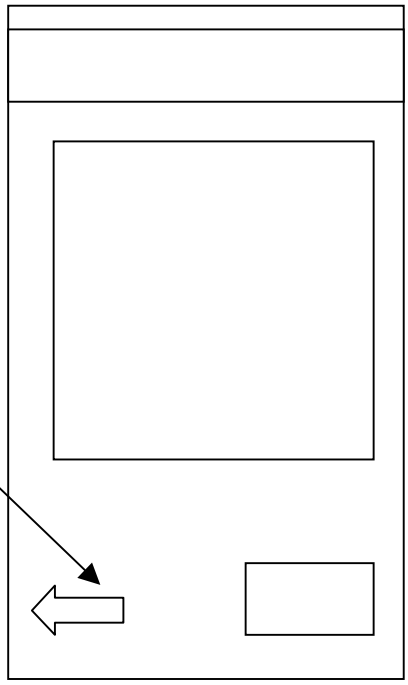
ALGORITHM 1
IF "Minor Pentatonic" selected THEN load ScaleDiagrams.java
ELSE IF "Blues" selected THEN load ScaleDiagrams.java
ELSE IF "Natural Minor" selected THEN load ScaleDiagrams.java
ELSE IF "Major" selected THEN load ScaleDiagrams.java
ELSE IF "Dorian" selected THEN load ScaleDiagrams.java
ELSE IF "Mixolydian" selected THEN load ScaleDiagrams.java

ALGORITHM 2
IF "Back" selected THEN load Scales.java
ELSE IF "Home" selected THEN load MainActivity.java


Home
LGORITHM 1
“Back” selected THEN load calesOptions.java
LSE IF “Home” selected THEN ad MainActivity.java

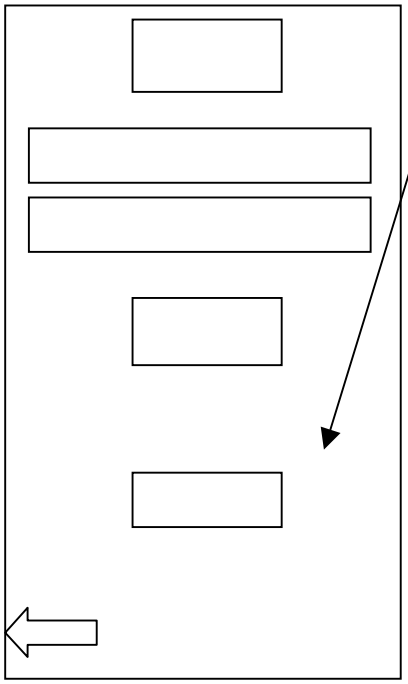
FAQ


ALGORITHM 1
IF “Equipment” selected THEN load FAQEquipment.java
ELSE IF “Getting Started” selected THEN load FAQGettingStarted.java
ELSE IF “How Long Should I Spend Practicing” selected THEN load FAQHowLongShouldIPractice.java
ELSE IF “Having Problems” selected THEN load FAQHavingProblems.java
Equipment
Getting Started
How long should I spend practicing?
Having Problems?
LGORITHM 2
LSE IF “Home” selected THEN ad MainActivity.java

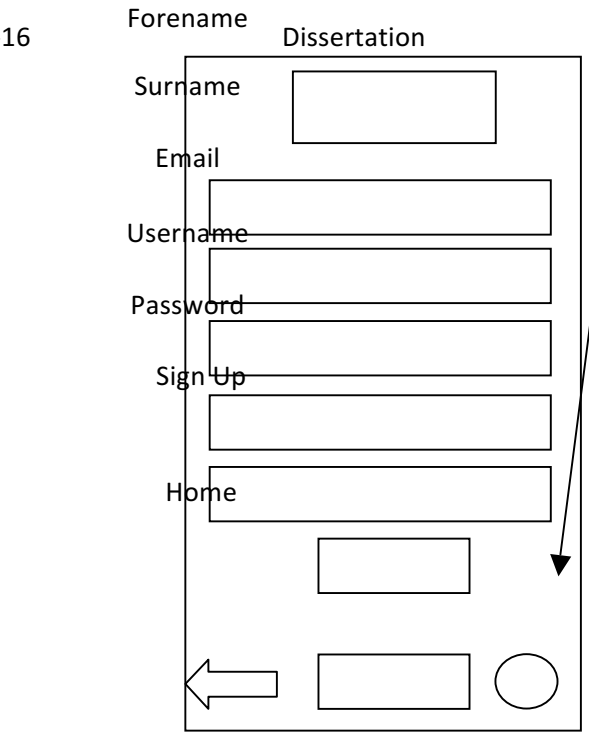


ALGORITHM 1
IF “Back” selected THEN load FAQ.java
ELSE IF “Home” selected THEN load MainActivity.java

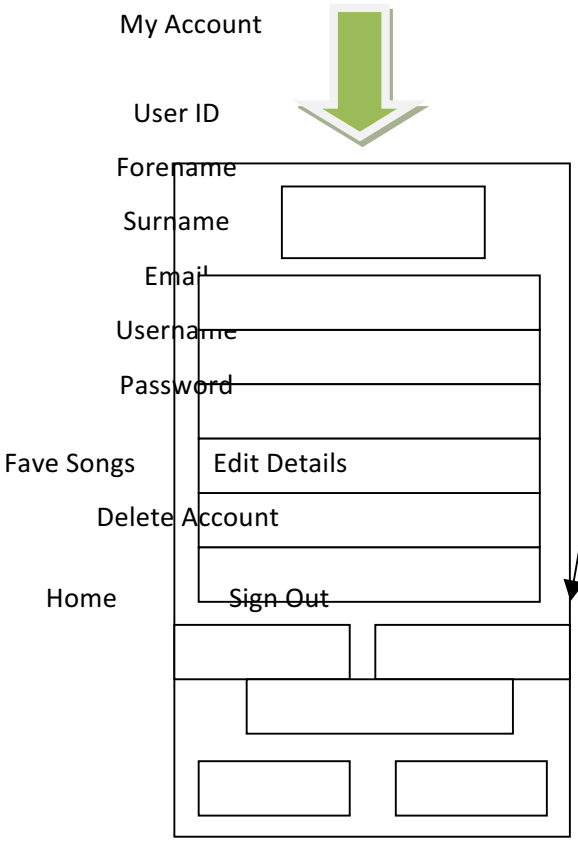
Account



ALGORITHM 1
IF “Login” selected AND user’s details correct THEN load MyAccount.java
ELSE IF “Sign Up!” selected THEN load NewAccount.java

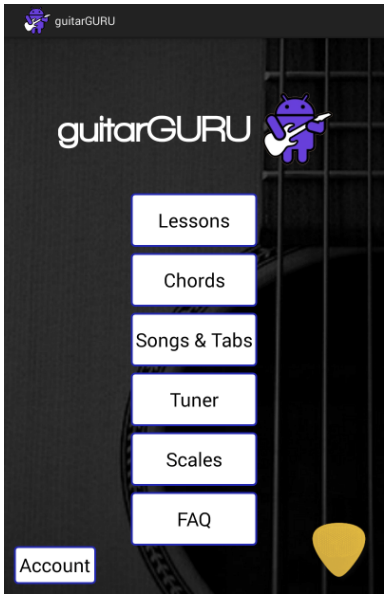


ALGORITHM 1
IF “Sign Up” selected AND all fields filled out AND username not taken THEN load MyAccount.java
ELSE IF “Back” selected THEN load Account.java
ELSE IF “Home” selected THEN load MainActivity.java

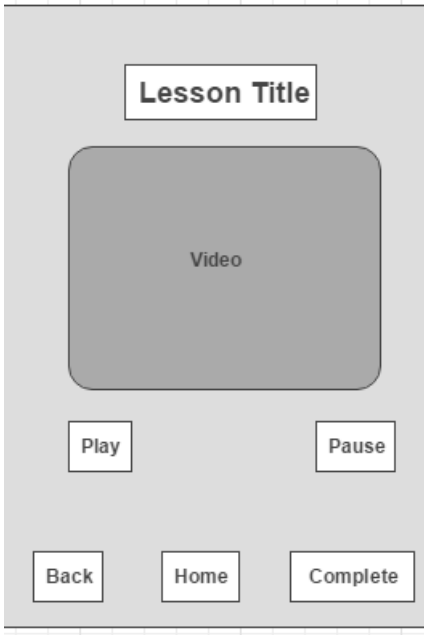


ALGORITHM 1
IF “Fave Songs” selected THEN load ViewUsersFavedSongs.java
ELSE IF “Edit Details” selected THEN load EditDetails.java
ELSE IF “Delete Account” selected THEN load NewAccount.java
ELSE IF “Home” selected THEN load MainActivity.java
ELSE IF “Sign Out” selected THEN load Account.java

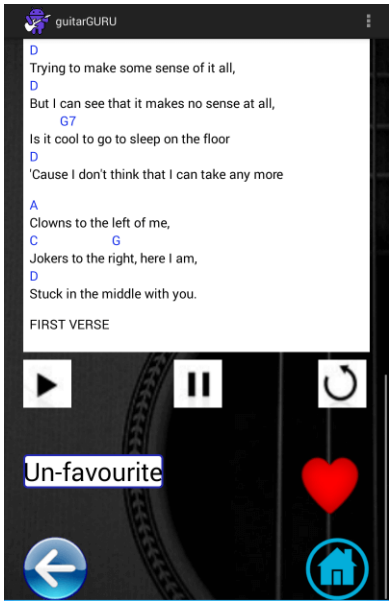
Appendix 5 – Wireframe Designs



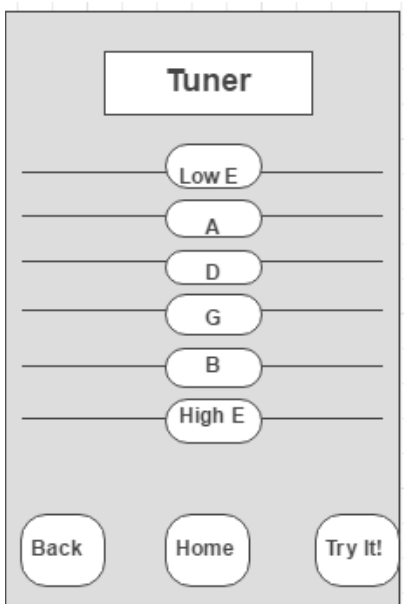
The wireframe design for the Main Menu has remained largely the same throughout development, with the six sections remaining consistent. The button design, with the rounded corners and blue border, was added, as was a dark background image of a guitar. The circle in the bottom corner of the wireframe has been replaced with the plectrum button, as it was decided that a plectrum would be more appropriate, given its association with the overall guitar theme.

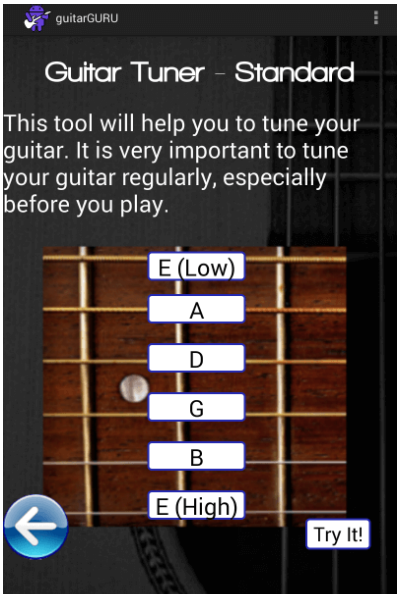


The screenshot taken of a video lesson from within the Lessons section clearly highlights that the wireframe design has remained largely consistent – for example, the play and pause buttons beneath the video, and the back and home buttons at the bottom of the screen. The wireframe also contains a "complete" button, which can also be seen in the screenshot.



As with the other wireframes that were created, the final design of this section mirrors the initial wireframe design. The only significant difference relates to the use of a ScrollView to allow the user to scroll up and down in order to display all of the lyrics and chords, and a heart button being used for the "Favourite" button. The play, pause and reset buttons controlling the MediaPlayer object that plays the song recording can be found directly below the song, and the remaining buttons can also being found in similar positions to those within the wireframe.





From the earliest stages of development, the initial design of each of the tuners is virtually exactly the same to that found in the wireframe design. Rather than just placing images of six strings on the screen, and then place the buttons on these strings, it was decided to adapt the background image used for these screens to include a fretboard, with the buttons being placed over each of the strings.

Appendix 6 – Example Testing Plans

Splash Screen

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Sound file play?	Pass	Pass	
Display for five seconds?	Pass	Pass	

Main Menu

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Lessons button function?	Pass	Pass	
Chords button function?	Pass	Pass	
Songs & Tabs button function?	Pass	Pass	
Tuner button function?	Pass	Pass	
Scales button function?	Pass	Pass	
FAQ button function?	Pass	Pass	
Account button function?	Pass	Pass	
Plectrum button function (text displayed)?	Pass	Pass	

Lessons

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Beginner button function?	Pass	Pass	
Intermediate button function?	Pass	Pass	
Advanced button function?	Pass	Pass	
Sound file on launch?	Pass	Pass	
Plectrum function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	

Beginner Lessons

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Scroll/view for lessons function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Tuning Your Guitar button function?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Video play and pause?	Pass	Pass	
Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Parts of Your Instrument button function?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Video play and pause?	Pass	Pass	
Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu	Pass	Pass	

function (not signed in)?			
Drop-down menu function (signed in)?	Pass	Pass	
Taking Care Of Your Guitar button function?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Video play and pause?	Pass	Pass	
Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Changing Your Strings button function?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Video play and pause?	Pass	Pass	
Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Learning Your Fretboard button function?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Images/text displayed correctly?	Pass	Pass	

Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
How To Hold Your Guitar button lesson?	Pass	Pass	
Lesson displayed?	Pass	Pass	
Images/text displayed correctly?	Pass	Pass	
Back button for this lesson function?	Pass	Pass	
Home button for this lesson function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	

Chords

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Plectrum button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Key Of C button function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
C Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Em Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
F Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
G Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Dm Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Am Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Bdim Chord from Key Of C?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	
Correct video displayed?	Pass	Pass	
Play button function?	Pass	Pass	
Correct TextViews?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Key Of D button function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
D Chord from Key Of D?	Pass	Pass	
Correct chord screen displayed?	Pass	Pass	

Songs & Tabs

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Plectrum button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Beginner Songs function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Call Me Maybe button function?	Pass	Pass	
Correct song displayed?	Pass	Pass	
Correct lyrics/chords?	Pass	Pass	
Play, pause, reset buttons function?	Pass	Pass	
Access Favourited Songs via long press on heart?	Fail	Fail	Favourited Song section launched, but recording of song did not stop playing – must be amended.
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Sweet Home Alabama button function?	Pass	Pass	
Correct song displayed?	Pass	Pass	
Correct lyrics/chords?	Pass	Pass	
Play, pause, reset buttons function?	Pass	Pass	
Access Favourited Songs via long press on heart?	Fail	Fail	Favourited Song section launched, but recording of song did not stop playing – must be amended.
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Run button function?	Pass	Pass	
Correct song displayed?	Pass	Pass	
Correct lyrics/chords?	Pass	Pass	
Play, pause, reset buttons function?	Pass	Pass	
Access Favourited Songs via long press on heart?	Fail	Fail	Favourited Song section launched, but recording of song did not stop playing – must be amended.
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Knocking On Heavens Door button function?	Pass	Pass	
Correct song displayed?	Pass	Pass	
Correct lyrics/chords?	Pass	Pass	
Play, pause, reset buttons function?	Pass	Pass	
Access Favourited Songs via long press on heart?	Fail	Fail	Favourited Song section launched, but recording of song did not stop playing – must be amended.
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Skinny Love button function?	Pass	Pass	
Correct song displayed?	Pass	Pass	
Correct lyrics/chords?	Pass	Pass	
Play, pause, reset buttons function?	Pass	Pass	
Access Favourited Songs via long press on heart?	Fail	Fail	Favourited Song section launched, but recording of song did not stop playing – must be amended.
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	

Tuner

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Standard Tuner button function?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Low E button sound?	Pass	Pass	Sound plays
A button sound?	Pass	Pass	Sound plays
D button sound?	Pass	Pass	Sound plays
G button sound?	Pass	Pass	Sound plays
B button sound?	Pass	Pass	Sound plays
High E button sound?	Pass	Pass	Sound plays
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Try It button function?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Low E button sound?	Pass	Pass	Sound plays
A button sound?	Pass	Pass	Sound plays
D button sound?	Pass	Pass	Sound plays
G button sound?	Pass	Pass	Sound plays
B button sound?	Pass	Pass	Sound plays
High E button sound?	Pass	Pass	Sound plays
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Drop D button function?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Low D button sound?	Pass	Pass	
Low A button sound?	Pass	Pass	
Mid D button sound?	Pass	Pass	
G button sound?	Pass	Pass	
B button sound?	Pass	Pass	
High E button sound?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Open C button function?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Low C button sound?	Pass	Pass	
Low G button sound?	Pass	Pass	
Mid C button sound?	Pass	Pass	
Mid G button sound?	Pass	Pass	
High C button sound?	Pass	Pass	
High E button sound?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	

Scales

Task	Tablet 1	Tablet 2	Detail
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Plectrum button function?	Pass	Pass	
Key Of C button function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Minor Pentatonic in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Dorian in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Blues in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Natural Minor in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Major in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Mixolydian in Key Of C?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Key Of D button function?	Pass	Pass	
Back button function?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Blues Scale in Key Of D?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Minor Pentatonic in Key Of D?	Pass	Pass	
Correct screen displayed?	Pass	Pass	
Correct title displayed?	Pass	Pass	
Correct patterns displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Major Scale in Key Of D?	Pass	Pass	

FAQ

Task	*Tablet 1*	*Tablet 2*	*Detail*
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Plectrum button function?	Pass	Pass	
Equipment button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
What Guitar button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
Other accessories button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
Getting Started button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
How Do I Tune My Guitar button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
Tuner button function	Pass	Pass	
Changing Strings button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
Changing Strings button?	Pass	Pass	
Why Do My Fingers Hurt button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
Most common pitfall button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Images displayed?	Pass	Pass	
Text displayed?	Pass	Pass	
How Long Should I Practice button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Back button function?	Pass	Pass	
Having Problems button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Home button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	

Appendix 7 – Database/Account Testing Plan

Task	*Tablet 1*	*Tablet 2*	*Detail*
Screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Login – both fields blank	Pass	Pass	Should be error, no account found
Login – username blank	Pass	Pass	Should be error, no account found
Login – password blank	Pass	Pass	Should be error, no account found
Login – incorrect details	Pass	Pass	Should be error, no account found
Login – correct details	Pass	Pass	My Account activity launched
My Account – correct details displayed	Pass	Pass	
Home button function?	Pass	Pass	
My Favourited Songs function button?	Pass	Pass	
Back button function?	Pass	Pass	
Edit Details – blank forename?	Pass	Pass	Should be error, all fields must be filled in
Blank surname?	Pass	Pass	Should be error, all fields must be filled in
Blank username?	Pass	Pass	Should be error, all fields must be filled in
Blank email?	Pass	Pass	Should be error, all fields must be filled in
Blank password?	Pass	Pass	Should be error, all fields must be filled in
Use already taken username?	Pass	Pass	Should be error, username must be available
All fields filled in, new username?	Pass	Pass	Details changed, displayed in My Account page
Delete Your Account button function?	Pass	Pass	
Delete Account – Yes, delete	Pass	Pass	Account deleted, taken to New Account screen.
Delete Account – No, do not delete	Pass	Pass	Don't delete, remain on My Account

Login As Administrator	Pass	Pass	Taken to Admin page
View Users button function?	Pass	Pass	
Back button function?	Pass	Pass	
View All Favourited Songs button function?	Pass	Pass	
Back button function?	Pass	Pass	
View Songs button function?	Pass	Pass	
Back button function?	Pass	Pass	
Search For User button function?	Pass	Pass	Enter number, details displayed
Delete User button function – no number entered	Pass	Pass	Button disabled
Delete User button function – Number entered	Pass	Pass	Button enabled.
Sign Up button function?	Pass	Pass	
Screen displayed?	Pass	Pass	
Back button function?	Pass	Pass	
Drop-down menu function (not signed in)?	Pass	Pass	
Drop-down menu function (signed in)?	Pass	Pass	
Home button function?	Pass	Pass	
Forename field blank?	Pass	Pass	Error message, not all fields filled out.
Surname field blank?	Pass	Pass	Error message, not all fields filled out.
Email field blank?	Pass	Pass	Error message, not all fields filled out.
Username field blank?	Pass	Pass	Error message, not all fields filled out.
Password field blank?	Pass	Pass	Error message, not all fields filled out.
Use already existing username?	Pass	Pass	Error message, not all fields filled out.
All details correct?	Pass	Pass	Account created, taken to My Account screen.
All details displayed correctly?	Pass	Pass	
Lesson Tracking			
Signed out – complete lessons?	Pass	Pass	Complete buttons not visible.
Signed in, not all lessons complete – access Beginner Quiz?	Pass	Pass	Error message, user must complete all lessons.
Signed in, not all lessons complete – access Intermediate Quiz?	Pass	Pass	Error message, user must complete all lessons.
Signed in, not all lessons	Pass	Pass	Error message, user

complete – access Intermediate Quiz?			must complete all lessons.
Signed out – try to access Beginner quiz?	Pass	Pass	Error message, user must sign in.
Signed out – try to access Intermediate quiz?	Pass	Pass	Error message, user must sign in.
Signed out – try to access Advanced quiz?	Pass	Pass	Error message, user must sign in.
Signed in, complete all lessons in Beginner Section.	Pass	Pass	All ticks appear, all codes added to lesson progress in database
Signed in, complete all lessons in Intermediate Section.	Pass	Pass	All ticks appear, all codes added to lesson progress in database
Signed in, complete all lessons in Advanced Section.	Pass	Pass	All ticks appear, all codes added to lesson progress in database
Beginner Quiz – signed in, all lessons complete	Pass	Pass	Quiz launched
Back button function?	Pass	Pass	
Next question button function?	Pass	Pass	New question displayed, buttons set to white, buttons enabled.
Correct screen displayed?	Pass	Pass	
Correct answer turns green?	Pass	Pass	
Incorrect answer turns red?	Pass	Pass	
Score increments with correct answer?	Pass	Pass	
Images appear and disappear when required?	Pass	Pass	
Questions and answers change?	Pass	Pass	
Previous high score displayed?	Pass.	Pass	
High score overwritten with higher score?	Pass	Pass	
High score stored in database?	Pass	Pass	
Intermediate Quiz – signed in, all lessons complete	Pass	Pass	Quiz launched
Back button function?	Pass	Pass	
Next question button function?	Pass	Pass	New question displayed, buttons set to white, buttons enabled.
Correct screen displayed?	Pass	Pass	
Correct answer turns green?	Pass	Pass	
Incorrect answer turns red?	Pass	Pass	
Score increments with correct answer?	Pass	Pass	
Images appear and disappear when required?	Pass	Pass	
Questions and answers change?	Pass	Pass	
Previous high score displayed?	Pass	Pass	
High score overwritten with higher score?	Pass	Pass	
High score stored in database?	Pass	Pass	
Advanced Quiz – signed in, all lessons complete	Pass	Pass	Quiz launched
Back button function?	Pass	Pass	
Next question button function?	Pass	Pass	New question displayed, buttons set to white, buttons enabled.
Correct screen displayed?	Pass	Pass	
Correct answer turns green?	Pass	Pass	
Incorrect answer turns red?	Pass	Pass	
Score increments with correct answer?	Pass	Pass	
Images appear and disappear when required?	Pass	Pass	
Questions and answers change?	Pass	Pass	
Previous high score displayed?	Pass	Pass	
High score overwritten with higher score?	Pass	Pass	
High score stored in database?	Pass	Pass	
Favouriting Songs			
Favourite each song, using heart button	Pass	Pass	Success messages, songs added to My Favourited Songs section
Long press heart button function?	Pass	Pass	My Favourited Songs section open, songs displayed.
Attempt to favourite each song again	Pass	Pass	Error messages, already favourited song
All songs added to FavouritedSongs table?	Pass	Pass	
Un-favourite each song, using Un-favourite button.	Pass	Pass	Success, songs removed from My Favourited Songs section.
Attempt to un-favourite each song again	Pass	Pass	Error messages, songs not in favourites list.
Entries removed from FavouritedSongs table?	Pass	Pass	
Long press heart button function?	Pass	Pass	My Favourited Songs section open, no songs displayed.
Long press heart button function? (Signed out).	Pass	Pass	Error message, user must sign in.

Appendix 8 – Example Tested Theoretical User Scenarios

Complete Learning To Fingerpick, How To Read Tabs, and Percussive Guitar

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Lessons option	Pass	Pass	
Select Advanced option	Pass	Pass	
Select Learning To Fingerpick	Pass	Pass	
Play video	Pass	Pass	
Complete Lesson	Pass	Pass	
Select back option	Pass	Pass	
Lesson marked as complete?	Pass	Pass	
Select How To Read Tabs	Pass	Pass	
Text/Images displayed?	Pass	Pass	
Complete Lesson	Pass	Pass	
Select back option	Pass	Pass	
Lesson marked as complete?	Pass	Pass	
Select Percussive Guitar	Pass	Pass	
Text/Images displayed?	Pass	Pass	
Complete Lesson	Pass	Pass	
Select back option	Pass	Pass	
Lesson marked as complete?	Pass	Pass	

Tuner

Use the Open C Tuner

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Tuner option	Pass	Pass	
Select Open C option	Pass	Pass	
Each button produces sound?	Pass	Pass	
When new string pressed, other sound files stop playing?	Pass	Pass	
When exit, all sound files stop playing?	Pass	Pass	
Back button function?	Pass	Pass	

Use the Drop D Tuner

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Tuner option	Pass	Pass	
Select Drop D option	Pass	Pass	
Each button produces sound?	Pass	Pass	
When new string pressed, other sound files stop playing?	Pass	Pass	
When exit, all sound files stop playing?	Pass	Pass	
Back button function?	Pass	Pass	

Chords

Select Am Chord from Key of C

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Chords option	Pass	Pass	
Select Key Of C option	Pass	Pass	
Select Am Chord	Pass	Pass	
Play video	Pass	Pass	
Correct diagram displayed?	Pass	Pass	

Select F#min Chord from Key of D

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Chords option	Pass	Pass	
Select Key Of D option	Pass	Pass	
Select F#min Chord	Pass	Pass	
Play video	Pass	Pass	
Correct diagram displayed?	Pass	Pass	

Select Hotel California and Get Rhythm

Task	*Tablet 1*	*Tablet 2*	*Details*
Open Main Menu	Pass	Pass	
Select Song & Tabs option	Pass	Pass	
Select Advanced option		Pass	
Select Hotel California	Pass	Pass	
Correct song displayed?	Pass	Pass	
Play recording?	Pass	Pass	
Pause recording?	Pass	Pass	
Reset recording?	Pass	Pass	
Play recording again?	Pass	Pass	
Favourite song	Pass	Pass	
Song displayed in My Favourited Songs?	Pass	Pass	
Recording stops playing?	Fail	Fail	Song displayed, My Favourited Songs section launched, but recording did not stop playing – must be amended.
Un-favourite song	Pass	Pass	
Song removed from My Favourited Songs?	Pass	Pass	
Open Main Menu	Pass	Pass	
Select Song & Tabs option	Pass	Pass	
Select Get Rhythm	Pass	Pass	
Correct song displayed?	Pass	Pass	
Play recording?	Pass	Pass	
Pause recording?	Pass	Pass	
Reset recording?	Pass	Pass	
Favourite song	Pass	Pass	
Song displayed in My Favourited Songs?	Pass	Pass	
Recording stops playing?	Fail	Fail	Song displayed, My Favourited Songs section launched, but recording did not stop playing – must be amended.
Un-favourite song	Pass	Pass	
Song removed from My Favourited Songs?	Pass	Pass	

Appendix 9 – The Evaluation Questionnaire

Name:	
Contact Details:	Date:

This questionnaire is to evaluate the application, guitarGURU. Numerous aspects will be focused on – namely, the various features, the quality of the user interface, and the overall usability. The developer will explain the tasks, and once completed, you will be asked to rate your experiences by selecting the appropriate option. You have the option to withdraw from the testing should you so desire, while you can also access your completed questionnaire at any point. Once the questionnaire has been completed, your details will be removed from the questionnaire printed in the final dissertation, and will be held confidentially until the project is completed, at which point they will be securely destroyed. Thank you for agreeing to evaluate the application, your time and feedback are greatly appreciated.

Main Menu

Open the application. You will be taken to a splash screen, and then the main menu.

1. What are your first impressions of the app? Especially regarding the colour scheme and sounds.

Very Poor	Poor	Average	Good	Very Good

Lessons

Select the Lessons option. You will be taken to the Lessons menu, displaying Beginner, Intermediate and Advanced options. The developer will direct you to two video lessons in the Beginner Section.

2. Do you think the video lessons would be useful to a beginner player?

Very Poor	Poor	Average	Good	Very Good

3. Were the videos easy to access and use? (ie playing and pausing the videos).

Not at all	Not Really	Average	Easy	Very Easy

The developer will direct you to two textual lessons in the Intermediate lessons section.

4. Do you think the information in these lessons would be useful to a beginner player?

Bad	Not Good	Average	Good	Very Good

5. Were these lessons easy to access, and use?

Not at all	Not Really	Average	Easy	Very Easy

You will be directed to an Advanced lesson.

6. In your opinion, is there a clear progression in terms of complexity from Beginner to Advanc lessons?

Not At All	Not Really	Somewhat	Definitely

Press the Home button. Create an account. Then return to the Lessons section, and complete all of the lessons in one of the sections. Attempt the quiz. Then, attempt to beat your high score.

7. What is your opinion on the usefulness of the quizzes in education?

Very Poor	Poor	Average	Good	Very Good

8. Was the quiz easy to access, and attempt?

Not at all	Not Really	Average	Easy	Very Easy

9. What are your views on the quality of the questions asked?

Very Poor	Poor	Average	Good	Very Good

10. How easy did you find it to mark a lesson as complete?

Very Difficult	Difficult	Average	Easy	Very Easy

11. What is your opinion on the usefulness of the ability to track lesson progress?

Very Poor	Poor	Average	Good	Very Good

12. What is your overall opinion on the usability and quality of the Lessons section?

Very Poor	Poor	Average	Good	Very Good

Feedback/Comments:

Chords

Use the drop-down menu to access the Chords section.

13. What are your initial opinions on the layout of the Chords section – Most common keys displayed, chords in key selected, then Chord displayed.

Very Poor	Poor	Average	Good	Very Good

14. Are the videos and diagrams of the chord useful educational tools when learning to play chords?

Not at all	No	Somewhat	Quite	Definitely

15. Overall impressions of the Chords section?

Very Poor	Poor	Average	Good	Very Good

Feedback/Comments:

Songs & Tabs

Return to the main menu using the Home button. Open the Songs & Tabs section.

16. What are your opinions on the format of the section – Beginner, Intermediate, Advanced?

Very Poor	Poor	Average	Good	Very Good

17. Would the song sheets be useful when attempting to learn a song?

Not at all	No	Somewhat	Quite	Definitely

Favourite a song. Open the Account section. View My Favourited Songs. Un-favourite the same song.

18. Were the lyrics and chords/tabs easy to read?

Not at all	No	Somewhat	Quite	Definitely

19. Is the ability to favourite songs, and access them through your account, a useful feature?

Not at all	No	Somewhat	Quite	Definitely

20. What is your overall impression of the Songs & Tabs section?

Very Poor	Poor	Average	Good	Very Good

21. Was this section easy to use?

Not at all	No	Somewhat	Quite	Definitely

Feedback/Comments:

Tuner

Return to the main menu using the Home button. Open the Tuner section.

22. What are your opinions on the layout of the tuners?

Very Poor	Poor	Average	Good	Very Good

23. Did you find the Try It Out! Section easy to use?

Not at all	No	Somewhat	Quite	Definitely

24. Would the Tuner section be useful to a beginner guitar player?

Not at all	No	Somewhat	Quite	Definitely

25. Was this section easy to use?

Not at all	No	Somewhat	Quite	Definitely

26. What is your overall impression of the Tuners section?

Very Poor	Poor	Average	Good	Very Good

Scales

Use the drop-down menu to access the Scales section.

27. What are your initial opinions on the layout of the Scales section – Most common keys displayed, scales in key selected, then scale displayed.

Very Poor	Poor	Average	Good	Very Good

28. Would the scales displayed be a useful tool to a beginner guitar player?

Not at all	No	Somewhat	Quite	Definitely

29. Are the scales easy to access, and easy to read?

Not at all	No	Somewhat	Quite	Definitely

30. What are your overall impressions of the Scales section?

Very Poor	Poor	Average	Good	Very Good

31. Was this section easy to use?

Not at all	No	Somewhat	Quite	Definitely

Feedback/Comments:

FAQ

Return to the main menu using the Home button. Open the FAQ section.

32. What are your opinions on the quality of the questions that have been included?

Very Poor	Poor	Average	Good	Very Good

33. What are your opinions on the quality of the answers provided?

Very Poor	Poor	Average	Good	Very Good

34. Would this section be useful to a beginner guitar player?

Very Poor	Poor	Average	Good	Very Good

35. Was this section easy to use?

Not at all	No	Somewhat	Quite	Definitely

Feedback/Comments:

Account

36. If you were a regular user of the app, would you be willing to create an account, given the various additional features that will be available to you?

Not at all	No	Somewhat	Quite	Definitely

37. Do you believe that the ability to track your progress through the lessons is a useful feature?

Not at all	No	Somewhat	Quite	Definitely

38. Do you believe that the ability to test your knowledge through quizzes is a useful feature?

Not at all	No	Somewhat	Quite	Definitely

39. Do you believe that the ability to favourite songs to practice is a useful feature?

Not at all	No	Somewhat	Quite	Definitely

40. What is your overall opinion on the usefulness of an Account feature within the app?

Not at all	No	Somewhat	Quite	Definitely

Miscellaneous

41. What is your opinion on the use of the plectrum button for providing advice and guidance?

Not at all	No	Somewhat	Quite	Definitely

42. Did you find it easy to navigate through the app using the Back/Home/Drop-down menus?

Not at all	No	Somewhat	Quite	Definitely

43. What are your opinions on the overall layout/colour scheme/legibility of the text?

Very Poor	Poor	Average	Good	Very Good

44. Was the application easy to use?

Not at all	No	Somewhat	Quite	Definitely

45. How would you rate your overall experience in using the app?

Very Poor	Poor	Average	Good	Very Good

46. Was the app responsive to all of your requests?

Not at all	No	Somewhat	Quite	Definitely

47. Do you believe this application would be useful to beginner guitar players?

Not at all	No	Somewhat	Quite	Definitely

48. Did you encounter any problems or errors when using the app?

Not at all	No	Somewhat	Quite	Definitely

49. Would you use the app again?

Not at all	No	Somewhat	Quite	Definitely

50. On a scale of 1 to 5, how would you rate the app overall? (1 - very poor, 2 – poor, 3 – average, 4 - good, 5 – very good).

1	2	3	4	5

51. Any additional comments or feedback you would like to make?

Feedback/Comments:

Thank you for taking the time to complete the questionnaire. Your assistance is greatly appreciated.

Appendix 10 – The Responses to the Questionnaire

This appendix contains a table summarising the responses of each evaluator to the questions asked in the questionnaire. The columns represent the individual respondents, while the rows represent the questions themselves.

	A	B	C	D	E	F	G	H
Q1	Good	Good	Very good	Very good	Very good	Good	Very good	Good
2.	Average	Very good	Very good	Good	Very good	Good	Very good	Very good
3.	Very Easy	Easy	Very easy	Easy	Very easy	Easy	Very easy	Easy
4	Good	Average	Very good	Average	Very good	Very good	Very good	Good
5	Very Easy	Easy	Very easy	Very easy	Very easy	Easy	Very easy	Easy
6	Somewhat	Somewhat	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
7	Very good	Very good	Good	Average	Very good	Good	Very good	Good
8	Very easy	Very easy	Easy	Easy	Very easy	Easy	Very easy	Very easy
9	Good	Very good	Good	Average	Very good	Good	Good	Good
10	Very easy	Very easy	Very easy	Very easy	Very easy	Easy	Very easy	Very easy
11	Very good	Good	Very good	Good	Very good	Good	Good	Very good
12	Average	Good	Very good	Good	Very good	Good	Very good	Very good
13	Very good	Very good	Very good	good	Very good	Good	Good	Good
14	Definitely	Definitely	Quite	Somewhat	Definitely	Definitely	Definitely	Definitely
15	Very good	Good	Very good	Good	Very good	Good	Good	Very good
16	Very good	Very good	Very good	Very good	Very good	Good	Very good	Very good
17	Definitely	Definitely	Definitely	Quite	Definitely	Definitely	Definitely	Definitely
18	Definitely	Definitely	Definitely	Quite	Definitely	Quite	Definitely	Definitely
19	definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
20	Good	Very good	Good	Very good	Very good	Very good	Very good	Very good
21	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
22	Very good	Good	Very good	Good	Very good	Very good	Very good	Very good
23	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
24	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
25	Definitely	Definitely	Quite	Definitely	Definitely	Definitely	Definitely	Definitely
26	Very good	Good	Very good	Very good	Very good	Very good	Very good	Good

27	Very good	Very good	Very good	Very good	Very good	Good	Very good	Good
28	Quite	Definitely	Somewhat	Definitely	Definitely	Definitely	Definitely	Definitely
29	Definitely	Definitely	Definitely	Quite	Definitely	Definitely	Definitely	Definitely
30	Good	Very good	Good	Good	Very good	Good	Very good	Very good
31	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
32	Good	Good	Very good	Good	Very good	Very good	Very good	Very good
33	Good	Good	Very good	Good	Very good	Good	Very good	Very good
34	Very good	Very good	Very good	Very good	Very good	Very good	Very good	Very good
35	Quite	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
36	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Quite
37	Definitely	Definitely	Definitely	Quite	Definitely	Definitely	definitely	Definitely
38	Definitely	Definitely	Definitely	Somewhat	Definitely	Quite	Definitely	Definitely
39	Definitely	Definitely	Definitely	Definitely	Definitely	Quite	Definitely	Definitely
40	Definitely	Definitely	Definitely	Definitely	Definitely	Quite	Definitely	Definitely
41	Definitely	Quite	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
42	Definitely	Definitely	Definitely	somewhat	Definitely	Definitely	Definitely	Definitely
43	Good	Good	Good	Good	Very good	Very good	Very good	Very good
44	Definitely	Definitely	Definitely	Definitely	Definitely	Quite	Definitely	Definitely
45	Good	Good	Good	Good	Very good	Good	Very good	Very good
46	Definitely	Definitely	Definitely	Somewhat	Definitely	Quite	Definitely	Definitely
47	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely	Definitely
48	No	No	Not at all	No	Not at all	No	Not at all	Not at all
49	Definitely	Definitely	Quite	Definitely	Definitely	Definitely	Definitely	Definitely
50	4	5	4	4	5	5	5	5

Appendix 11 – Code

The code has been submitted to the School Office at the Magee Campus on a USB Memory pen.