From The Top

by

Joe Stainke

In partial fulfilment of the requirements for the degree of

MSc in

Software Development



Department of

Computer and Information Sciences

August, 2022

Abstract

With resources for self-learning musical instruments at an all-time high since the rise of the Internet, musicians are able to shape their own learning journey like never before. But with all the benefits that go along with this, there is a trend of musicians having a negative view of theory and leaving it out of their self-chosen syllabus.

The aim of this project is to provide musicians with a resource that encourages and allows them to learn the basics of music theory in an accessible and practical way.

**Declaration**

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc in Software Development of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself.

Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive.

(please tick) Yes [ X ] No [ ]

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is 7280.

I confirm that I wish this to be assessed as a Type 1 2 3 4 5 Dissertation (please circle).

Signature:

Date: 14/08/2022

Acknowledgements

I would like to thank my supervisor, Konstantinos Liaskos, for his incredibly helpful guidance and feedback throughout the duration of the project and the entire course.

Contents

[1 Introduction 1](#_Toc111414420)

[1.1 Project Overview 1](#_Toc111414421)

[1.2 Project Motivations 1](#_Toc111414422)

[1.3 Project Brief 2](#_Toc111414423)

[1.4 Background 2](#_Toc111414424)

[2 Requirements 4](#_Toc111414425)

[2.1 Requirements Background 4](#_Toc111414426)

[2.2 Functional Requirements 5](#_Toc111414427)

[2.3 Non-Functional Requirements 6](#_Toc111414428)

[2.4 User Stories 7](#_Toc111414429)

[3 Design 8](#_Toc111414430)

[3.1 User Interface Design 8](#_Toc111414431)

[3.2 Data Design 10](#_Toc111414432)

[3.3 System Architecture 10](#_Toc111414433)

[3.4 Technology Choices 11](#_Toc111414434)

[4 Implementation 13](#_Toc111414435)

[4.1 Lifecycle and Project Management 13](#_Toc111414436)

[4.2 Coding Standards 13](#_Toc111414437)

[4.3 Sprints 13](#_Toc111414438)

[5 Documentation 16](#_Toc111414439)

[5.1 User Guide 16](#_Toc111414440)

[5.2 Java Documentation 16](#_Toc111414441)

[6 Testing and Evaluation 17](#_Toc111414442)

[6.1 Manual Testing 17](#_Toc111414443)

[6.2 User Acceptance Testing 19](#_Toc111414444)

[6.3 Usability Evaluation 20](#_Toc111414445)

[6.4 Evaluation and Limitations 22](#_Toc111414446)

[7 Further Work 24](#_Toc111414447)

[8 Conclusion 25](#_Toc111414448)

[A Requirements Survey 28](#_Toc111414449)

[B User Stories 32](#_Toc111414450)

[C Class Diagram 34](#_Toc111414451)

[D User Acceptance Tests 35](#_Toc111414452)

Tables

[Table 1: Initial Must Have Requirements 5](#_Toc111413988)

[Table 2: Initial Should Have Requirements 6](#_Toc111413989)

[Table 3: Initial Non-Functional Requirements 6](#_Toc111413990)

[Table 4: Manual Test Case - Set Name 18](#_Toc111413991)

[Table 5: UAT Results 21](#_Toc111413992)

[Table 6: User Stories 33](#_Toc111413993)

[Table 7: User Acceptance Test 1 35](#_Toc111413994)

[Table 8: User Acceptance Test 2 36](#_Toc111413995)

[Table 9: User Acceptance Test 3 37](#_Toc111413996)

[Table 10: User Acceptance Test 4 38](#_Toc111413997)

Figures

[Figure 3.1: Set Name Screen 8](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414288)

[Figure 3.2: Set Goals Screen A 8](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414289)

[Figure 3.3: Set Goals Screen B 8](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414290)

[Figure 3.4: Quiz Screen A 9](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414291)

[Figure 3.5: Quiz Screen B 9](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414292)

[Figure 3.6: Quiz Results Screen 9](file:///E:\Dropbox\MSc\Project\REPORT.docx#_Toc111414293)

[Figure 8.1: Class Diagram 34](#_Toc111414294)

# Introduction

## Project Overview

Music theory is a useful tool for musicians of all levels. It is often likened to learning a language, where it is possible to speak the words without knowing what they mean or why they were chosen, but without that knowledge there are considerable limitations. For instance, it would be difficult to adapt phrases to different situations without a deeper general understanding of the language and culture. The other main similarity is that it allows musicians to communicate much more conveniently and effectively. [1]

## Project Motivations

Many musicians believe theory to be of no use with some even going as far to suggest it stifles creativity, and there are, on the surface, reasonable arguments to back up the first claim at least. A number of the most highly regarded musicians and composers of all time never even learned how to read music. [2] However, theory goes far beyond just reading music, and whether they choose to see it this way or not, it can be used to explain the choices they have made such as why they used this scale there, or finished on that chord. Musical theory can be seen as a set of guidelines built from all music that has been made until now, that all music made from now onwards can refer to. There are reasons why certain things sound satisfying, but there is nothing to say that they are right or that not doing them is wrong. They are not rules. [3] In order to think outside the box, it helps to know as much as possible about what is inside the box. [4]

The view that theory is unimportant or that it limits creativity seem to be held more by musicians who are on the beginner end of the scale, or who do not fully understand what is meant by “theory”. Because of this, it feels important to stress early on that this is not the case, and to describe in simple terms what people mean when they bring up the subject. The results from a public user survey to gather requirements as well as background information suggest that the more theory a person knows, the more useful they believe it to be – two thirds (66.67%) of those surveyed claimed to have intermediate knowledge or higher, with almost the exact same number (67.02%) believing that a well-rounded knowledge of theory was at least “very important” for musicians.

## Project Brief

The high-level brief for the project was as follows:

“The aim of this project is to develop a tool that helps learners to familiarise with concepts within the discipline/field of music theory as part of their independent/self-directed learning.

The end-product must include functionality on the following aspects:

- Assess readiness to learn;

- Set learning goals;

- Engage in the learning process; and

- Evaluate learning.

The target users will be learners and instructors of music theory. Requirements gathering and evaluation will involve users from this target audience.”

From this brief, a number of must have requirements (which are discussed in detail below) were translated from the four essential functionality aspects, along with GUI implementation. The target was to complete a minimum viable product (MVP). That, along with the time-sensitive nature of the project, meant that the number of initial features would be limited. However, this afforded a greater focus to be placed on making each feature work well, as well as designing them in such a way that they could be reused and expanded upon in future updates.

From the user survey, 62 out of the 84 people who answered “yes” to playing an instrument were also instructors, many of whom were enthusiastic about the idea of using an app to supplement their teaching materials. A few responses mentioned the idea of linking the learning materials to official exam grades, an idea that was already heavily considered. However, after considering the project motivations and its MVP nature, it was decided that the first iteration of the app would be more of an introduction to the subject aimed at beginners.

## Background

Since it was mentioned in the high-level brief and as it was the #1 top free education app on Google Play at the time [5], Duolingo was used as the primary reference point for initial features and GUI design. Screenshots were taken to refer back to while working through the early stages of the Spanish course. It was noted that users had the choice to begin the course from the very start, or to begin with a review from which it could work out the user’s current level.

The feature that stood out most was the considerable variety of question types such as multiple choice, translating and forming a sentence from a selection of words, filling in gaps, and matching pairs of Spanish words to their English counterparts, all complete with spoken examples. These could all provide potential equivalents when it come to learning music theory.

Also noteworthy was the practical-based nature. Most of the teaching is presented with a “learn by doing” approach, with explanations provided occasionally in a complementary manner. This is good for teaching the user in a way that they are constantly engaged (one of the requirements for this project). Translations of basic words are repeated heavily at the start, but thanks the variety of ways in which the user is quizzed, it manages to not feel repetitive.

The key motivator for creating a music theory learning app in this style was the developer’s own experience of performing with other musicians. The ease by which an idea can be conveyed if everyone knows the same terms and concepts can be beneficial in so many ways. Mainly, time is saved and momentum is maintained by not having to explain everything in simplified terms, leading to more focused and productive practice and writing sessions. There are already plenty of resources catering to people who already want to and have already started learning music theory, but more could be done to persuade and encourage the demographic of musicians who do not currently study theory or do not want to into starting.

# Requirements

## Requirements Background

Requirements were gathered from a combination of the brief, surveying learners and instructors in the relevant field, researching similar technologies, and the developer’s own personal experience. From there they were prioritised using the MoSCoW technique and divided into must haves, should haves, could haves, and won’t haves. The first and most highly prioritised of the must have requirements were adapted from the high-level brief, and the rest were essential aspects that related more specifically to a music theory app, most of which were worked into the project brief. Should haves were prioritised based on a number of factors such as how many users requested them, the amount of work and time that would need to be devoted, and how important it was that they appeared in the first release. Could haves were generally features that seemed like good ideas that could be added in future updates or if the scope of the project increases (discussed further below), and won’t haves were mostly user suggestions that did not meet the developer’s vision for the app. Won’t haves also include any features that were considered at one stage but then discarded.

## Functional Requirements

The following tables detail the initial list of requirements.

### Must have requirements:

|  |  |
| --- | --- |
| Requirement | Description |
| Users must be able to assess their own readiness to learn. | When opening the app for the first time, users need to be given enough information so they can decide for themselves if their environment is right for it and whether or not it will aid them in their learning. |
| Users must be able to set a learning goal. | Many apps, including Duolingo, have daily streaks that count how many consecutive days the user has logged in or completed a certain action. A similar feature where the app counts how many days in a week the user has completed a quiz will be utilised here. When setting up an account, users will be asked how many days they want to aim for. The app will then keep the user up to date with their progress and let them know when they have achieved their goal. |
| Users must be able to engage in the learning process. | Much like in Duolingo, it is important that learners are able to engage with the app instead of reading long passages which can lead to a very limited amount of information being taken in. To meet this requirement, the app will include audio examples as part of the learning materials since musical notation on its own would not mean much to a beginner. |
| Users must be able to evaluate their learning. | Quizzes, sometimes referred to as drills or exercises, were a common element to most apps that were used for background research. |
| Users must be able to view their past quiz results and learning progress. | This is almost a sub-requirement of evaluating learning. Being able to view past results and general progress all in one place is very useful in terms of the user being able to gain an overall sense of how well they are doing. |
| Users must be able to learn at different difficulty levels. | It is essential for learners to be able to work at the level that is appropriate for them, so, for the MVP, at least one of the topics will include learning materials at more than one difficulty level and different levels of quizzes to reflect these. |

Table : Initial Must Have Requirements

### Should have requirements:

|  |  |
| --- | --- |
| Requirement | Description |
| Users should be able to receive instant feedback on quiz answers. | It is important for users to know straight away whether they have answered correctly or incorrectly, as leaving this until the end of the quiz can lead to the user being focussed only on their score and skimming over other important information. It also benefits the ability to self-evaluate. |

Table : Initial Should Have Requirements

## Non-Functional Requirements

Non-functional requirements were also gathered during the planning stage. They are harder to define as they describe more about how the app should function rather than exactly what it must do. This means they can also be harder to evaluate as they are less objective than functional requirements. They are not mandatory in the same way that functional requirements are, but they are essential in making the system operate at a higher efficiency. [6]

|  |  |
| --- | --- |
| Requirement | Description |
| The app should be divided into small “bite-sized” sections. | This was suggested by a number of users and is in line with how the researched apps were presented. The majority of users opted for the lower answers when asked how long they would likely use such an app each day, so it needs to reflect this with short learning materials and quizzes. |
| The app should also act as a portable reference guide. | A secondary purpose of the app could be to double up as a kind of glossary or dictionary of terms. |
| The app should be beginner friendly. | As the target audience for the MVP will be learners who are new to music theory, it needs to be geared towards this type of user. |
| The app should be intuitive in terms of usability. | The app needs to follow certain UI conventions so that users are able to navigate the system easily and work out what each feature does before they use it. |

Table : Initial Non-Functional Requirements

## User Stories

User stories are used to convert functional requirements into what a user might want from the system. They are useful in allowing the developer to view the system’s requirements from a user perspective and they all follow the structure:

“As a <Role> I want <goal> so that <benefit/expected outcome>.” [7]

Filling in the structure this way for each requirement requires the developer to give a reason for implementing the given requirement. This is helpful towards reprioritisation, as it gives a clearer idea of how beneficial each piece of functionality is. It is also effective in justifying the requirements – if no benefits to the user can be attached to the story, it can be determined that including the feature might not be worthwhile.

Each story is given a number point value from the Fibonacci sequence (1, 2, 3, 5, 8 …) [8] based on its estimated difficulty and timeframe. This is so they can be directly compared when planning which stories will be implemented during each sprint, ideally spreading the workload evenly over the course of the project. Otherwise, it can be tempting to tackle all of the easier tasks at the start, resulting in a slump midway through until the more difficult tasks are completed near the end.

Prioritisation was an ongoing process – as more was learned about the project, certain requirements became more or less important, and new ones became needed – and the user stories table was regularly updated to reflect this. The aim for the MVP, and the way in which the project’s overall success was measured, was to include all must haves and as many should haves as possible.

The final user stories table can be found in Appendix B.

# Design

One of the first steps after completing the research and thinking about the initial requirements and user stories was to work on the design of the app. The research provided a couple of templates to take inspiration from, so there was already a clear idea of what the UI design should be. Since the non-functional requirements mentioned for the system to be “bite-sized” and to provide a portable reference guide that can supplement practical learning, it was decided that the system would be implemented as a mobile app. As a note, the terms “screen” and “page” are used interchangeably.

## User Interface Design

Medium fidelity wireframe prototypes were created to visualise the requirements. All of the must have and should have requirements were covered, with each screen containing information on the side about which user story or stories they are referring to. Names of the corresponding XML layout files and Java classes were later added to make it easier to keep track of where to locate each screen in the IDE. The ability in PowerPoint to move from one screen to another via button navigation was invaluable in planning how the user would move around the app. Having a visual representation also massively helped when creating the layout files.

A picture containing graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Figure .: Set Goals Screen B

Figure .: Set Goals Screen A

Figure .: Set Name Screen

Figures 3.1 to 3.3 show how the user will set up their profile and goals (Story 2) after starting up the app for the first time. These screens were modified slightly to force the user into entering a value in the fields and confirming them before the continue buttons would appear. This prevented null values from being passed through and gave the user the chance to double check that they were happy with what they had entered. Also, a number picker was used in favour of the drop-down menu. The screen at Figure 3.3 led to a page that would cover Story 1 as the final part of the intro, but it was unknown at the time what the details would be, so it was only a placeholder at this stage.

Graphical user interface, text, application, chat or text message

Description automatically generatedA picture containing diagram

Description automatically generatedDiagram

Description automatically generated

Figure .: Quiz Screen A

Figure .: Quiz Screen B

Figure .: Quiz Results Screen

Stories 4, 5 and 7 are represented by figures 3.4 to 3.6 where the user can self-evaluate by taking a quiz, see straight away whether or not their answer is correct, and view their score at the end. Radio buttons were initially used but they were more difficult to modify than what they were replaced with – for instance, there was no option to change the default colours which were hard to make out against the background. The timer was coded in but then later removed as it was decided after user testing that it added no benefit to the experience. The other changes were mostly cosmetic.

The full prototype can be found in the supporting documents folder.

## Data Design

All user data was handled by the SharedPreferences API which stores the data directly in the app. This allows for simple and fast reading and writing of key-value pairs and is best used when there is a relatively small amount of data, which is the case here. The code is also fairly straightforward (after learning the basics) and reusable, so there should be no issues when scaling up to add more learning topics.

The information stored in this way includes user profile data, quiz high scores and number of attempts, progress towards the weekly goal, and whether or not the introduction / initial setup has been completed.

An early version of the app used SQLite for the quiz questions and answers (which are now stored in their own Java class) as well as user data, but this was felt to be of no benefit as the code was overly complicated and prone to runtime errors which were hard to catch. Room, which makes use of SQLite in a more user-friendly way, was also considered before discovering that SharedPreferences could take care of everything that was required.

## System Architecture

As Android Studio was used for the IDE, the system was programmed in an object-oriented way. This is portrayed in the class diagram (Appendix B).

In terms of architectural patterns, Android Studio lends itself favourably towards Model View Controller (MVC). ‘Model’ represents the backend, which is comprised of SharedPreferences data and data classes, where the app logic is stored. Normally, the backend would also communicate with the database, but since here it is built-in, that step is skipped. The ‘View’ is the UI which is built in Android Studio using XML layout files and visualises the Model for the user to see on the screen. The ‘Controller’ is how the Model and View communicate with one another. When the user makes changes to the View, the Controller updates the Model. When the Model data is changed, the Controller updates the View to reflect this. Separating the View from the Model has the advantages of being able to quickly generate new pages that are similar to existing screens (such as the learning materials) and borrowing certain logical elements from one class to another. [9] [10]

## Technology Choices

### Android Studio

As it was decided that the app would be built for Android, Android Studio, the official IDE for developing Android apps [11], was chosen as the IDE for the project. Flutter was briefly considered for cross-platform purposes to increase the potential user base, but prior development had already been worked on in Android Studio which made it more likely to yield a better product in the time frame. Android also leads the market share in mobile operating systems by a long way [12], so the trade-off in quality for a slightly higher user base did not feel worth it for the first release. During the development phase it became evident that all of the API’s and modules that were required for completion of the app were available within Android Studio, so there was no need to connect to any other frameworks or SDK’s.

### Canva

The artistic direction of the app in its current state is very minimal. Aside from the drawable resource files created within Android Studio, the only pieces of artwork created were the logo and the play and pause buttons. Canva was chosen for its wide range of available elements and ability to export images, as well as selected elements from a larger image, as transparent PNG files. The transparency was essential to not only look professional, but to allow the logo to be used as the app icon, and for the buttons to have rounded instead of squared edges.

### PowerPoint

Prototyping was carried out in Microsoft PowerPoint as it has been the developer’s usual tool in the past. This meant that templates for the Android UI and buttons had already been created, so time could be spent more efficiently.

### Guitar Pro

All audio files were created using Guitar Pro as it can export MIDI and mp3 files and is generally better than alternative options (Logic, Pro Tools) when it comes to tasks such as transcribing notation. It was thought that MIDI files would benefit the app due to their much smaller size, but it was not known for certain at the time that SoundPool would be used and is incompatible with MIDI. However, it was convenient to have both options in case either was required.

### Audacity

One drawback of SoundPool is its very short file size and length limit. The size limit was not a problem as the files did not need to be of tremendous audio quality, so a low bitrate was already being used. The length limit, however, caused some of the files to be cut off prematurely when played in the app. It was discovered that this limit could be increased by converting the files from stereo to mono which was achieved in Audacity.

### Qualtrics

Qualtrics was used for the user requirements gathering survey as it is the university’s recommended technology and access to all premium features was available. A huge variety of question types were available as well as extremely detailed results of users’ responses, which were used for analysis purposes.

# Implementation

## Lifecycle and Project Management

Since this was a solo project with only one developer, the Agile methodology in its true state could not really be applied, but the parts of it that translated well to a non-team environment were adapted and followed. This meant the sole developer also assumed the role of project manager which mainly involved planning and keeping the work up to standard.

## Coding Standards

The main objectives relating to coding standards were consistency and readability which in turn promotes reusability and maintainability. This was a challenge as many pieces of code were reworked from different sources where varying practices and naming conventions were used. The best way to address this was to first get the code working. Then everything that was added was reviewed and modified to match the previously existing code. Finally, the manual testing steps were repeated to make sure no bugs were introduced by doing that, and that everything still worked.

## Sprints

In line with Agile, sprints were used as a way of dividing the workload into a timeframe. As it was such a short project with only one developer, there needed to be flexibility around how long each sprint would last, and which tasks were completed in each one. Equally, it was useful to adhere to the plan wherever possible to avoid running out of time. The project lasted 12 weeks and was divided into four sprints, each lasting between two and four weeks with slight overlaps.

### Sprint 0

Sprint 0 can be thought of as a pre-sprint – work that needed to be completed before actual development could start – and took place over the first two weeks.

Background research (discussed above) was the first task to take place. This was important in developing a clearer idea of what the project would become and what it needed to achieve.

A series of small apps were developed containing several features that were intended to be implemented in the main project. These included a soundboard app that used the SoundPool class to play short audio clips on button presses, and two quiz apps that used very different approaches. This was a worthwhile process as code and logic from all of them was later refactored into the final version of the app.

Initial requirements were listed, and a user survey was drafted to refine, prioritise, and expand on them.

### Sprint 1

Sprint 1 is where development properly began, and the initial requirements started to be met.

Wireframes (discussed above) were designed at the beginning of this stage to address how the user stories would be implemented, and to visualise the layout of the app.

The foundations of the app were then laid down to match the wireframes. This mainly involved designing the layout pages, naming widgets, and programming buttons to navigate the system. Essentially, a high-fidelity prototype was created as a foundation from which to build the MVP.

Participants were recruited for the user survey and responses were used to help build on the system requirements and turn them into user stories.

Next, the quiz was coded into the system using refactored code from the two apps that were built in Sprint 0, merging the best aspects of each after making a table of their respective pros and cons. The quiz was required for many of the other functions to work, so it was given a high priority and taken care of in the first sprint. A number of user stories were covered in the quiz and quiz results activities, and a large volume of programming was required which resulted in it taking up most of the sprint.

### Sprint 2

Sprint 2 started with learning materials being added to the to the Rhythm and Scales screens[[1]](#footnote-1). The combination of text and images had to be formatted in a manner that was pleasantly readable within a scroll view, which took longer than expected. These alone were inadequate to sufficiently engage the user in the learning process, so audio files were created and added, encouraging the user to listen and play along.

SharedPreferences were introduced at this point to keep track of user data in a variety of ways (mentioned above).

User Acceptance Testing and Usability Evaluation were prepared towards the end of this sprint so they could commence as early as possible in the final sprint.

### Sprint 3

It was acknowledged that the last few weeks of the project would be taken up mostly by writing the report, so not an awful lot of development was planned. Anything left to do was prioritised and it had to be accepted that not everything could be covered by the end of the 12 weeks.

The final sprint started with filling in some details that were essential but relied on other being in place before they could be added. This included populating the quiz question data and completing the About page to meet the criteria for assessing readiness to learn (Story 1).

UAT and evaluation took place next as it was a suitable point in between the app being close to finished while still having enough time to make changes if any were needed.

Heuristic evaluation was loosely arranged to take place after UAT, but it was left too close to the end of the project. Experience with previous heuristic evaluation, however, meant that the general principles were kept in mind throughout and hopefully adhered to.

# Documentation

## User Guide

A detailed user guide is included in the supporting documents folder. It is aimed at any user running the app for the first time, although it is hoped that the app is intuitive enough on its own.

## Java Documentation

Every class, method, and block of code that contained logic was commented. This was to support maintainability in case the project is ever returned to by the original developer or anyone else, as otherwise, it might take a long time to remember, or work out, what the code does.

The JavaDoc folder can be found with the supporting documents.

# Testing and Evaluation

## Manual Testing

Manual testing was used to identify bugs in favour of unit testing and system testing as it was more appropriate considering the system’s design. All the functionality related to something happening on the front end, so it was generally quite a straightforward, process to test whether or not a feature was working as intended by installing the app on the developer’s Android device and running through the steps.

One downside to this form of testing is that it requires the developer to think of every possible way in which an error could be incurred as they are not the kind of errors the IDE’s debugger will pick up on. They are usually discrepancies in what is expected to happen rather than errors in the code.

Another is that it can be time consuming for two main reasons. The first reason is that, once an error is noticed, it is not always clear what needs to be changed. The other is that sometimes a large number of steps need to be taken to test a feature, and that can be multiplied if there are different ways in which a feature can be used.

It is therefore in the developer’s interest to create comprehensive test case scenarios to cover as many possible outcomes as they can think of in order to prevent any undesired events from being able to occur. Below is an example of one of the test cases, the general template for which was followed for every feature in the system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Scenario ID:** | | Set-Name-1 | **Test Case ID:** | Set-Name-1A |
| **Test Case Description:** | | Set Name – All positive and negative test cases | | |
| **Pre-Requisite:** | | Clear app data | | |
| **Step** | **Action** | **Inputs** | **Expected Output** | |
| 1 | Launch application. | - | Welcome screen opens. | |
| 2 | Press ‘Continue’. | - | Set Name screen opens. | |
| 3 | Press ‘Set Name’. | - | Toast message ‘Please set a username’ appears. | |
| 4 | Enter name longer than limit. | Username: ‘ABCDEFGHIJK’ | Field stops accepting characters after ‘J’ (10 characters). | |
| 5 | Enter name with whitespace at the start and end. | Username: ‘ Piotr ‘ | Whitespace is stripped. Toast message ‘Username set. Hello, Piotr!’ appears. Name is set. | |
| 6 | Re-set username as a blank field. | - | Toast message ‘Please set a username’ appears again. | |
| 7 | Press ‘Continue’. | - | Toast message ‘Please set a username (nice try!)’ appears. | |
| 8 | Repeat step 5. | Username: ‘ Piotr ‘ | Whitespace is stripped. Toast message ‘Username set. Hello, Piotr!’ appears. Name is set again. | |
| 9 | Press ‘Continue’. | - | Set Goal screen opens. | |
| 10 | Continue instructions until the Home screen is reached. | - | Home screen opens. “Hello, Piotr” appears under the “Home” text. | |
| 11 | Press ‘Stats’. | - | Stats screen opens. “Piotr’s Stats” appears at the top. | |

Table : Manual Test Case - Set Name

The method used was highly effective as it forced the developer to view the app as a user and think about some of the ways in which it could be misused or manipulated. For instance, when a username was entered and completed via autofill, a space was added to the end. This upset the text formatting in other areas of the app, so the Java ‘trim’ method was added. It was then easy to test whether it had worked by continuing to the parts of the app it affected. Then it was discovered that the user could make the continue button appear by setting their name, then re-setting their name as a blank field and continuing with an empty name. This was undesired, so the developer had to come up with a way of preventing that from happening. Evidently, this process involved a great deal of trial and error, so it was important to keep track of the steps that were being performed so they could be repeated in case fixing a bug in one place created or reintroduced a bug elsewhere.

The feature that required most testing was the weekly goal tracker. Firstly, it required a huge number of steps each time – multiple quizzes had to be taken, the device date had to be manually changed, and app data had to be cleared before starting the test in order to reset everything. Secondly, the data was being mostly updated in once place (the Quiz Results page) and displayed in another (the Home page), so it was not always clear where to find any bugs that were noticed. For a long time, a bug existed where, after the system worked out that a new week had started, not all of the progress was reset straight away. Some of it was, but the rest did not clear until the second time the Home page was loaded. This was eventually solved by moving some of the code from the Home class to the Welcome class to make sure it was all definitely being executed on launch.

## User Acceptance Testing

User Acceptance Testing (UAT) is usually carried out by the shareholder to assess how successfully the system meets their requirements. However, seeing as the “shareholders” for this project are the markers, members of the public were invited to take part and assume the role. The same group of people who took part in the requirements gathering survey were invited in order to achieve continuity. A much smaller user pool was required this time. Willing participants were asked to email the developer along with information regarding whether or not they would describe themselves as “technically minded”. This was so that, even if only the bare minimum of two users took part, at least one “technical” and one “non-technical” user was involved, as it was thought that they were likely to view the app in contrasting ways and pick up on different issues.

Under Agile, UAT usually takes play at the end of each sprint. However, it was decided that one round at the start of the final sprint would be enough. The system and all of its must have and should have functional requirements and user stories can be divided into four main pieces of functionality, so a test was designed for each one. The tests, along with the steps that the users were required to take, and each step’s expected results, can be found in Appendix C. Due to the extensive manual testing that was performed, it was expected that each test would pass (and they did). It was still worthwhile to have this confirmed by users, but to make it more informative, users were only given a single instruction for each test, which was essentially the name of the test, and observed while they attempted to work it out by themselves. This technique ventured more into usability evaluation, which is covered next.

## Usability Evaluation

Usability evaluation was tied in with UAT and completed concurrently. Users were asked to think out loud while going through the testing steps which gave the developer a view into the thought process of someone with no knowledge of how the app works in the background. Then, upon completion of the UAT steps, they were asked to freely explore the app and give feedback on anything they noticed. A total of 7 issues were raised between the two testing participants. These ranged from mild suggestions to moderate usability problems and were thusly prioritised into a table based on severity and time needed to fix.

|  |  |  |  |
| --- | --- | --- | --- |
| Priority | Issue | Solution | Fixed |
| 1 | Upon pressing the back button repeatedly, the user was able retrace their entire screen history, all the way back to the welcome screen via the setup pages, which should not have been accessible after completion. | Create a method on the home screen to prevent users from being able to go back any further. Also have the method allow the user to exit the app by pressing back twice in quick succession. | Partly |
| 2 | Users are unable to change their name and goal (except by exploiting the first issue) | Add a button to the home page allowing users to revisit the setup pages. | Yes |
| 3 | The quiz select screens are incompatible with dark mode. Difficulty text is the same colour as the button background. | Redesign the button background colours on the affected screens to work in dark and light mode. Alternate fix would have been to assign the text a colour so it stayed that colour in dark mode. | Yes |
| 4 | Question text is changed to say which answer is correct after submitting an answer. User is unable to properly review as they might not remember the question. | Leave the question text unaltered and move the text to say which answer is correct elsewhere on the screen. | Yes |
| 5 | Users are only told if they have reached their goal after returning to the home page. | Tell users on the quiz results screen if they have reached their goal. | No |
| 6 | No navigation buttons. Users sometimes have to press back multiple times to arrive at the screen they are looking for. | Add navigation buttons for the home, learn, quiz, and stats pages to the bottom of the screen on certain pages to allow users to navigate more easily. | No |
| 7 | No reminder feature. Users might like to receive a notification if they have not completed a quiz towards their weekly goal at a certain time each day. | Implement a notification feature to remind users if they have not completed a quiz yet that day. | No |

Table : UAT Results

The issues that managed to be fixed were fairly straightforward as they were mostly small amendments in terms of the code required. Dark mode was never really considered at any point, but the fact that one of the screens was rendered unusable under that setting meant that it needed to be fixed.

The three suggestions at the bottom of the table were not implemented in time for the MVP and moved to the backlog. Navigation buttons were considered but not deemed essential for the time being, due to the current, relatively small size of the system. Reminder notifications were on a similar level of importance. Incidentally, both features were already included in the could have requirements, especially as the latter had been suggested in the user survey, so these results have moved them further up the priority list of features to be included in the next iteration of the app.

It was thought that issue 1 had been fixed, but subsequent manual testing discovered that it had only been partially fixed. If it is the user’s first time running the app (i.e., it they have not fully closed it yet), pressing the back button twice in quick succession on the home page will exit the app as desired, but on reopening they will be taken back to the setup pages. If the app has been shut down after the setup pages have been completed, closing and reopening it in the same manner will lead back to the Welcome screen, which is fine. It is not as large of a bug as it originally was, but it still needs fully fixed.

## Evaluation and Limitations

Overall, the app does almost everything required of it. All of the must have requirements and almost all of the should have requirements were met, which was the target for the MVP. One of the strong points is the readability and reusability of the code. The more that was added, the more it was realised that large portions of the system could be reused, leading to an increasingly faster work pace and plenty of opportunity for future expansion. Only a small number of minor bugs were leftover from testing, which is a fairly good result. However, there are a number of areas that could have been improved upon.

The learning materials in their current state are acceptable, but they fail to meet the non-functional requirement of being “bite-sized”. This became a problem on the construction side as well when building the layout for the medium scales page. Android Studio reacts strangely when designing layouts in ScrollView, which gets worse the longer the page. Using fragments to divide the materials was already being considered after the first scales page felt slightly too long, but the size of the medium scales materials confirmed their necessity moving forward. It was felt that this needed to be addressed before adding any new learning pages, but it was too close to the end of the project by the time that was decided.

Although it was never mentioned in any of the requirements or the brief, it was a desire of the developer to include more varied types of quiz questions than just text-based. However, it was always known how much of a challenge it would be to create a quiz with image and audio-based questions, let alone combining them all into one, and it was important to keep the scope of the project realistic. If there was a longer timeframe available, this would have been a much higher priority.

A flaw on the design side was the lack of explicitly object-oriented programming. It was difficult to design the code this way inside a new environment and adapt it correctly without first dedicating time to properly study the IDE. Similarly, and maybe consequentially, the absence of a strong design pattern throughout led to many situations where the code could have been more elegant and included less duplication. Dependency injection might have been a good pattern to follow, as it would have dealt with the SharedPreferences data much more concisely. [10] Too much of the logical code takes place inside the onCreate method, which is not recommended best practice, but this relates again to lack of experience within the environment.

# Further Work

One future aspect that was considered from the very start of the project was the addition of further topics, as the two included only offer an introduction into the discipline of music theory. A benefit of using the MVC architecture is that the layout files and classes belonging to the existing learning pages are highly reusable. The first step here though would be to redesign these pages in the manner discussed above, then using the new versions as templates. This would also cover one of the missing non-functional requirements by making the learning materials more “bite-sized”.

As mentioned previously, image and audio-based questions would be added to the quizzes as this would make for a much-improved way to evaluate learning.

The one should have functional requirement missing from the MVP is Story 6, concerning difficulty levels. Along with the extra learning topics to be added, the proposed redesign would make it easier to split the materials into difficulty levels. Adding the extra quizzes after they are also redesigned would also be fairly straightforward, as the code to access them already exists. It is just currently made unavailable to users until the questions are added.

While it was previously mentioned that iOS devices make up a relatively small percentage of the smartphone market, Apple products in general are more popular among creative types. Flutter can be added to existing Android apps [13], so that is an option to maximise the potential user base at some point.

Aside from these aspects, an attempt would be made at fixing all known minor bugs. Then the backlog of could have and won’t have requirements would be revisited, reprioritised, and gradually implemented into the system.

# Conclusion

The objective of this project was to create system that could aid musicians in their self-directed study of music theory. This was accomplished by creating a self-contained mobile application for Android operating systems. An adaptation of the Agile methodology was followed to manage the project lifecycle and to allow for flexibility as project goals shifted and developer skill improved. Thorough testing and analysis were performed on all aspects of the system using a number of industry proven techniques to ensure the final product was of the highest possible quality. The initial project aims were modified in places, but through effective project management, it was made sure that the key objectives were still met. It has always been possible to become a self-taught musician, the growing number of online resources is making it more and more accessible, but encouraging this demographic to teach themselves theory with the same enthusiasm is more of a challenge. With the successful creation of an introductory, beginner-friendly music theory learning app, this issue can hopefully be addressed and improved.

References [1]

|  |  |
| --- | --- |
| [1] | D. J. Clifft, "Music Theory for Music Majors," Majoring in Music, [Online]. Available: https://majoringinmusic.com/music-theory-for-music-majors-why/. [Accessed 22 July 2022]. |
| [2] | S. A. Hall, "10 legendary musicians who never learned to read music," Classic FM, 8 October 2021. [Online]. Available: https://www.classicfm.com/discover-music/famous-musicians-who-cant-read-music/. [Accessed 22 July 2022]. |
| [3] | G. Ewer, "Does Music Theory Stifle Creativity?," Easy Music Theory, 18 October 2013. [Online]. Available: https://easymusictheory.wordpress.com/2013/10/18/does-music-theory-stifle-creativity/. [Accessed 22 July 2022]. |
| [4] | P. M. Almeida, "How Music Theory is (not) Killing Your Creativity," Beyond Music Theory, [Online]. Available: https://www.beyondmusictheory.org/how-music-is-not-killing-your-creativity/. [Accessed 22 July 2022]. |
| [5] | Google, "Top free education apps," Google Play, 25 July 2022. [Online]. Available: https://play.google.com/store/apps/category/EDUCATION?hl=en\_GB&gl=US. [Accessed 25 July 2022]. |
| [6] | Indeed Editorial Team, "9 Nonfunctional Requirements Examples," Indeed, 4 May 2021. [Online]. Available: https://www.indeed.com/career-advice/career-development/non-functional-requirements-examples. [Accessed 13 August 2022]. |
| [7] | S. Paikaray, "The Structure of a User Story," DLT Labs, 30 March 2020. [Online]. Available: https://medium.com/dlt-labs-publication/the-structure-of-a-user-story-3a0dc3917d23. [Accessed 13 August 2022]. |
| [8] | Visual Paradigm, "What is Story Point in Agile? How to Estimate a User Story?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/. [Accessed 27 July 2022]. |
| [9] | GeeksforGeeks, "Difference Between MVC and MVP Architecture Pattern in Android," GeeksforGeeks, 10 November 2020. [Online]. Available: https://www.geeksforgeeks.org/difference-between-mvc-and-mvp-architecture-pattern-in-android/. [Accessed 13 August 2022]. |
| [10] | A. Hussain, "Common Design Patterns and App Architectures for Android," raywenderlich, 29 March 2021. [Online]. Available: https://www.raywenderlich.com/18409174-common-design-patterns-and-app-architectures-for-android#toc-anchor-020. [Accessed 13 August 2022]. |
| [11] | R. Barcia, "Android Studio: Build apps on the official IDE for Android," IBM, [Online]. Available: https://www.ibm.com/garage/method/practices/code/tool\_android\_studio/. [Accessed 26 July 2022]. |
| [12] | StatCounter, "Mobile Operating System Market Share Worldwide," GlobalStats, April 2022. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202204-202204-bar. [Accessed 26 July 2022]. |
| [13] | Flutter, "Add Flutter to existing app," Flutter, [Online]. Available: https://docs.flutter.dev/development/add-to-app. [Accessed 2022 July 26]. |

1. Requirements Survey

**Music Theory App**

**Start of Block: Consent**

**Consent Form for ‘From The Top’ User Questionnaire**   
**Name of department: Computer & Information Science** **Title of the study: From The Top** I confirm that I have read and understood the Participant Information Sheet for the above project and the researcher has answered any queries to my satisfaction. I confirm that I have read and understood the Privacy Notice for Participants in Research Projects and understand how my personal information will be used and what will happen to it (i.e. how it will be stored and for how long). I understand that my participation is voluntary and that I am free to withdraw from the project at any time, up to the point of completion, without having to give a reason and without any consequences. I understand that anonymised data (i.e. data that do not identify me personally) cannot be withdrawn once they have been included in the study. I understand that any information recorded in the research will remain confidential and no information that identifies me will be made publicly available. I consent to being a participant in the project.

* I agree to all of the above (1)

**End of Block: Consent**

**Start of Block: Survey**

Q1 How old are you?

* Under 18 (1)
* 18-24 years old (2)
* 25-34 years old (3)
* 35-44 years old (4)
* 45-54 years old (5)
* 55-64 years old (6)
* 65+ years old (7)

Q2 Do you teach or have you ever taught music in any capacity?

* Yes (1)
* No (2)

Q3 Do you play any musical instruments at any level?

* Yes (1)
* No (2)

Q4 If yes, what instrument(s) do you play?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q5 If you play an instrument, are you self-taught or do/did you get lessons from an instructor?

* Self-taught (1)
* Lessons (2)
* Mixture (i.e., had lessons for one, self-taught for another) (3)

Q6 How would you describe your current music theory knowledge?

* 1 - I know a couple of scales or chords at most (1)
* 2 - I know a decent amount / enough to get by (2)
* 3 - I am intermediate or higher (3)

Q7 Would you be interested in an app to help you learn music theory?

* Yes (1)
* No (2)

Q8 What would be your main reason for wanting to learn music theory?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q9 How important do you think it is for musicians to have a well-rounded knowledge of theory?

* Not at all important (1)
* Slightly important (2)
* Moderately important (3)
* Very important (4)
* Extremely important (5)

Q10 How long would you aim to use a music theory learning app each day?

* Less than 5 minutes (1)
* 5-15 minutes (2)
* More than 15 minutes (3)

Q11 Are there any particular features you would want from such an app?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**End of Block: Survey**

1. User Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_NO | Story | Tag | Points | Sprint Number |
| 1 | As a user I want to find out early on what the app is about and how it works so that I know whether or not it is right for me. | Must Have | 2 | 3 |
| 2 | As a user I want to set a goal of how many days a week I want to take a quiz so that I have something to encourage me to learn regularly. | Must Have | 2 | 2 |
| 3 | As a user I want to be given audio examples of learning materials so that I can hear what the concepts sound like, helping me to engage in the learning process. | Must Have | 5 | 2 |
| 4 | As a user I want to test my knowledge by taking quizzes so that I know how much I have really learned. | Must Have | 8 | 1 |
| 5 | As a user I want to see straight away whether my answer is correct or not so that I am instantly made aware of what I might be struggling with. | Should Have | 1 | 1 |
| 6 | As a user I want to be able to learn and evaluate at different difficulty levels so that I am working at the level that is appropriate for me. | Should Have | 2 | 1 |
| 7 | As a user I want to view my quiz results at the end of each quiz so that I know how well I have done. | Must Have | 2 | 1 |
| 8 | As a user I want to be able to take a quiz at the end of learning a topic so that I can instantly review my knowledge of the topic. | Should Have | 1 | 1 |
| 9 | As a user I want to view my progress towards my ‘quiz days per week’ goal so that I know what my current progress is. | Must Have | 5 | 2 |
| 10 | As a user I want to be told when I have reached my ‘quiz days per week’ goal so that I can feel a sense of achievement. | Must Have | 1 | 2 |
| 11 | As a user I want to view stats about my past quizzes so that I can gain an overall sense of how well I am doing. | Must Have | 3 | 2 |
| 12 | As a user I want to be able to change my learning goal at any point so that I can adjust the pace of my learning if I need to. | Could Have | 1 | 3 |
| 13 | As a user I want to set up daily notifications so I can be reminded if I have not made progress towards my weekly goal. | Could Have | 8 | 3 |
| 14 | As a user I want to learn in line with ABRSM and/or SQA grades so I can learn and revise towards exams. | Won’t Have | 8 | - |
| 15 | As a user I want to learn about world music so that I can understand more about how people from other cultures think about music. | Won’t Have | 5 | - |
| 16 | As a user I want to listen to musical excerpts and identify concepts so that I can increase my ear and knowledge of concepts. | Could Have | 3 | - |
| 17 | As a user I want to take challenges where I have to fill in the blanks but with musical notes so that I can train my ears to recognise intervals and pitch. | Could Have | 3 | - |
| 18 | As a user I want to share my results so I can compare scores and progress with friends. | Won’t Have | 2 | - |
| 19 | As a user I want to hear a cheer-like sound effect when I get something right so that receive a small dopamine boost. | Could Have | 1 | - |
| 20 | As a user, I want to continue to be tested on lower-level content as I progress so that I don’t forget the more basic concepts. | Could Have | 1 |  |

Table : User Stories

1. Class Diagram



Figure .: Class Diagram

1. User Acceptance Tests

|  |  |
| --- | --- |
| **Test name:** Complete the setup and introduction. | |
| **Test number:** 1 | |
| **User story/stories:** 1, 2 | |
| **Test step** | **Expected result** |
| Press the continue button. | The set name screen opens. |
| Type a username into the field and press the set name button. | Text appears to welcome the user and the continue button appears. |
| Press the continue button. | The set goal screen opens. |
| Choose a number (1) by scrolling and press the confirm button. | Text appears to confirm the goal has been set and the continue button appears. |
| Press the continue button. | The about screen opens. |
| Press the sounds great button. | The home screen opens. Text appears relating to the name and goal they have set. |
| Go back to the welcome screen by pressing the back button then press the continue button again. | The welcome screen opens, then leads to the home screen this time the continue button is pressed. |

Table : User Acceptance Test 1

|  |  |
| --- | --- |
| **Test name:** Complete the learning materials for the scales topic. | |
| **Test number:** 2 | |
| **User story/stories:** 3, 8 | |
| **Test step** | **Expected result** |
| Press the learn button. | The topic select screen opens. |
| Press the scales button. | The scales learning materials open. |
| Play a sound and then stop it. | The sound starts playing and then stops. |
| Play a sound and then play another sound before the first sound has finished. | The first sound plays, then stops when the second sound starts playing. |
| Press the test my knowledge button. | The scales quiz select screen opens. |

Table : User Acceptance Test 2

|  |  |
| --- | --- |
| **Test name:** Complete an easy rhythm quiz. | |
| **Test number:** 3 | |
| **User story/stories:** 4, 5, 6, 7 | |
| **Test step** | **Expected result** |
| Press the quiz button. | The quiz topic select screen opens. |
| Press the rhythm button. | The difficulty select screen opens. |
| Press the easy button, then press the start quiz button. | A border appears around the easy button, then the quiz activity starts. |
| Select an answer. | The selected answer changes colour. |
| Press the confirm button. | The correct answer changes colour, the question text changes to say which option is correct, and the confirm button changes to say “next”. |
| Press the next button. | The next question appears, the question number increases, and the confirm button says “confirm” again. |
| Repeat steps 4, 5 and 6 until the finish quiz button appears. | The confirm button changes to say “finish quiz”. |
| Press the finish quiz button. | The quiz results screen opens. |
| View the quiz results screen. | The user is able to evaluate their learning by seeing what score they achieved. |

Table : User Acceptance Test 3

|  |  |
| --- | --- |
| **Test name:** View stats. | |
| **Test number:** 4 | |
| **User story/stories:** 9, 10, 11 | |
| **Test step** | **Expected result** |
| View the quiz days per week goal progress on the home screen. | A green circle is shown above the current day and a red circle is shown above any earlier days of the week. The text tells the user they have taken a quiz on 1 day this week, and the text below tells them how many days left to reach their goal. If their goal was 1 day (which they were asked to select in test number 1), it tells the user they have reached it. |
| Press the finish stats button. | The stats screen opens. |
| View the stats. | The easy rhythm quiz and scales course completed information are updated. |

Table : User Acceptance Test 4

1. Resources licensed from <https://github.com/openmusictheory> under the Creative Commons license. [↑](#footnote-ref-1)