COSC345 Assignment 4 - Report

Anthony Dickson, Rory Jackson, Johnny Mann

October 1, 2018

Contents

1 Overview

For our project we made an Android application, called Aural Learner, which aims to provide a way to practise aural music skills. We have been able to implement the majority of features that we set out to implement, however some features are still buggy (namely voice recognition) and some features were dropped (e.g. rhythms exercises) due to the unexpected complexity of the app. As such, one might be able to argue that the code we are delivering for this assignment is closer to a beta version than a 'final polished' version that we were initially intending on delivering. In this report we will go into further detail about our app, some of problems we encountered and how we handled them, and how difficult a new developer may find contributing to our project.

2 Group Members

- Anthony Dickson - 3348967
- Rory Jackson - 2208377
- Johnny Mann - 3891999

3 About Aural Learner

3.1 Aims and Goals

Our app, Aural Learner, is an app that aims to provide a way for people, regardless of any visual impairments, a way to practise aural music skills. While many similar apps do already exist, none of them allow users who are visually impaired to practise this knowledge. The aural theory you can practise with this app includes major scales, intervals, and basic melodic skills. These skills are crucial for musicians who want to play music requiring good theory skills or those who wish to train their ears. Currently the only methods that exist are braille score (which is costly to produce and can turn two pages of score into twenty pages), cumbersome audio cd's, or expensive private tuition. Our app is not meant to compete with these strategies, as that is outside of the scope of the app, our app is meant to be a tool that can help with some of the memorisation and practice that is involved with these skills. Helping visually impaired people with these skills can lessen their need to rely on those other methods, which can hopefully make learning music cheaper, more accessible, and affordable.

3.2 How It Works

3.2.1 Exercises

The way our app works is that it relies on the user singing the exercises in order to memorise the information. In our interval exercise, for example, a user selects either easy, medium, or hard for the difficulty and then they get told what interval they will need to sing. The user is played the interval, the interval is repeated and they are encouraged to sing along to it. Next the user must sing the interval, this is without the audio backing, and it is also graded. They then get presented of a grade ranging from bad to perfect, based on how on many notes they managed to sing 'correctly' (close enough for a note to register as the desired note).

3.2.2 User Interface

While there is a simple GUI for those that want to use it, most of the exercise is conducted through the headphones with the user able to navigate through the menus and exercises with their voice. Everything the user should need to know is accessible by voice, and the important information of an exercise is spoken back to the user using text-to-speech. We were able to test this app on a few musicians who could not see the screen and they gradually get better at the exercise. It is hard to perform a full test of the efficacy of this app to teach people with no sight, but really, we are simply improving on the audio learning method by automating it and providing user feedback. Provided people find this app easy to navigate, then they should have little problem learning with this method.

3.3 Bugs

Voice control is still a bit buggy, and not quite where we wanted it for this assignment. It occasionally mishears what the user says to it, causing it to go the wrong place or to do the wrong thing. Unfortunately, a lot of the work on this part of the app was left to later while we worked on getting the core parts of the app done (e.g. the exercises). And a lot of the difficulties with integrating voice recognition stemmed from having to deal with a complicated setup which required the device to play notes to the user, speak to them via text-to-speech, record the user's singing and figure out their pitch with a pitch detection algorithm (Fast-Fourier Transformation), and on top of all that manage voice recognition. This configuration meant that we were constantly juggling limited system resources such as access to the device microphone and speakers, which only allowed on thing to use them at any given time.

4 Problems & Solutions

4.1 Dealing With People's Strengths & Weaknesses

4.1.1 Rigid Roles and Task Allocation

Over the course of this assignment, we encountered many problems that hindered our progress. One of them concerns group members strengths & weaknesses, and the roles and tasks assigned to various group members. We found that assigning people roles that were too strict hindered progress, since it meant relying on people to deliver their code at specific agreed times, regardless of unexpected task difficulty. Once we made our roles more flexible, code got developed much quicker, and it worked better since we were able to have more perspective when developing our parts, especially when we were writing code together.

4.1.2 Pair Programming

Another problem we found was although we had regular communication, it was still hard to build things correctly in a way that made it both inline with music theory, and robust. To address this, we tried pair programming nearer to the end of the deadline. We found that it worked surprisingly well and made the task more enjoyable since it was much harder to get stuck in a rut full of bugs or build redundant code. It was also extremely helpful being able to combine the strengths of two members, meaning that while some members may not be as musically inclined, or as skilled a programmer, as the other members, we could rely on the others to fill in the gaps in our knowledge/skills. Building and testing new features on different git branches worked well, especially when paired with the continuous integration of the javadoc pages. This made it considerably easier to figure out what the code was doing when building the individual parts.

4.2 Singing Is Hard

4.2.1 Grading Metrics

Another problem that existed was people finding it hard to sing the pitches. In order to remedy this, our app allows the user to sing any octave of a given note, as long as it is the same note, and still have it marked as correct. Also, instead of using a distance metric for the final aggregate score, we opted to use a simplified scoring system that simply looks at how many notes were close enough to register as the desired notes. This makes the exercises much more forgiving, while still retaining their usefulness.

4.2.2 Noisy or Missing Audio Data

Another problem was related to how the pitches were graded: it was hard to know when the exercise began; and it was hard to determine whether the user was actually singing the pitches correctly based on the often noisy audio data.

To address the first issue we made the app wait for the user to start singing, which helped to prevent 'null' readings (where the user's pitch would be not be a valid frequency due to the lack of audio input) which would cause the grading algorithm to give erroneous grades.

To address the second issue, we employed a bit of statistics to ignore 'noisy' pitch readings that would significantly skew the average frequency readings. This gave us more accurate readings, which improved the quality of the grading algorithm.

5 Ease of Integrating New Developers

In order to accommodate another developer, we have employed techniques to make integration easier.

5.1 Documentation

The first is the use of javadocs. We have made an effort to ensure that code is sufficiently commented with javadoc comments so that the purpose of variables, methods, and classes are clear. We have setup gradle to automatically generate javadocs, and travis-ci to publish these javadocs to GitHub pages for each successful build. This makes the javadocs highly accessible and allows developers to view the docs without having to trawl through hundreds of lines of code. We also added more general information to the readme in order to make it easier for a new developer to understand the purpose of the app and the features that it provides.

5.2 Bug Tracking & Task Management

Starting at about halfway through the project, we realised Facebook chat was not going to meet our needs for bug tracking and task management. To this end, we started utilise GitHub's mechanisms for bug tracking and task management, namely issues and projects. We set up a kanban style board to track tasks that needed to be done, bugs that needed to be fixed, and other things that were in progress or completed. Through GitHub's automation features we could make issues for our tasks/bugs, assign them to group members, and have them automatically appear in the correct column in the kanban board (e.g. To do, in progress, done). This greatly increased our ability to keep track of what needed to be done and places to discuss certain bugs/features. Long gone are the days of scrolling through days of Facebook chat to find what causes bug x, or what we wanted included in feature y.

5.3 Code Design & Clarity

We also made efforts to have readable code, this allows to someone with a decent amount of musical understanding to figure out how the code works. Carefully choosing variable names that reflect what the code is doing musically in somewhat musical terms meant that we were able to mirror the musical relationships that the code was aiming to represent. They should also be able to figure out what the functions should be doing based on how the functions are being tested in the unit tests, or through the surrounding code. The only thing that would hamper them would be our somewhat unclear structure, which was mainly due to the complexity of the tasks and having many asynchronous tasks. Although we did make efforts to have classes and activities that reflected how the app works, it would still require some reading into even with a music background. This is something we have been working on as more of the core functionality was implemented.

6 Summary

Overall, this was a quite a difficult project for us, and it tested us in many aspects. We had to tackle software engineering on a platform and scale with tools and technology that we had no experience with, and on top of that learn methods to help us work effectively as a group. We are pleased with what we were able to achieve and feel like we learned a lot through this project.