Coursework Report

Sonnan Naeem
40341713@napier.ac.uk
Edinburgh Napier University - Mobile Applications Development (SET08114)

1 Introduction

Goal The goal of my coursework assignment was to implement and design a prototype of an app for the Android platform using the Android SDK. It was allowed to be any kind of app I wanted as long as it showed that I could implement basic app fundamentals such as the use of multiple activities/fragments, use of basic I/O principles, and also have a user-friendly interface.

Overview The application I chose to create would be a note-taking app that was aimed at both songwriters and poets. The app would allow users to write, save, update, and delete songs, generate a list of rhymes from a word of choice, and also generate random words to give users the tools they might need in order to complete a song or poem. I was able to successfully implement the songwriting and the rhyme generator portion of the application. However, I was not able to implement the random word generator because of the unexpected amount of time spent on understanding how web API's are used in Android Studio. Therefore, I had spent more time successfully programming a cohesive application with the intention to implement a user-friendly interface and strong functionality for the other two tools.

Inspiration The inspiration for the app came from the fact that I am a songwriter and know other songwriters in the music industry, and the features that my app demonstrate are usually separated into their own apps causing their use to be fragmented and not in one streamlined platform. This is what motivated me in trying to implement these tools all in one application.

2 Software Design

2.1 Core Features

The first thing I did to try and understand how to develop the songwriting portion of my application was downloading different note taking apps and writing down what they had in common. Most of the apps I found had three key elements in their note taking interface. 1. Notes would usually be displayed as a list on the Main Activity 2. They would preview the contents of the note in the Main Activity 3. Notes would always be accessed by simply clicking the respective item on the screen. This allowed me to notice the core elements that I needed to make sure to implement in my application.

2.2 Sketching

The second step I undertook in my software design process was sketching the look and interface of the application. I sketched the different screens of what could be my application with different forms of navigation for switching between activities. The three main forms of navigation that Android's Material design uses are the "Navigation Drawer", the "Bottom Navigation Bar", and the "Tabs" [1]. So I made sure to sketch screens with each type of navigation in order to try and see what would seem to be visually the most intuitive (Figures 1-5).

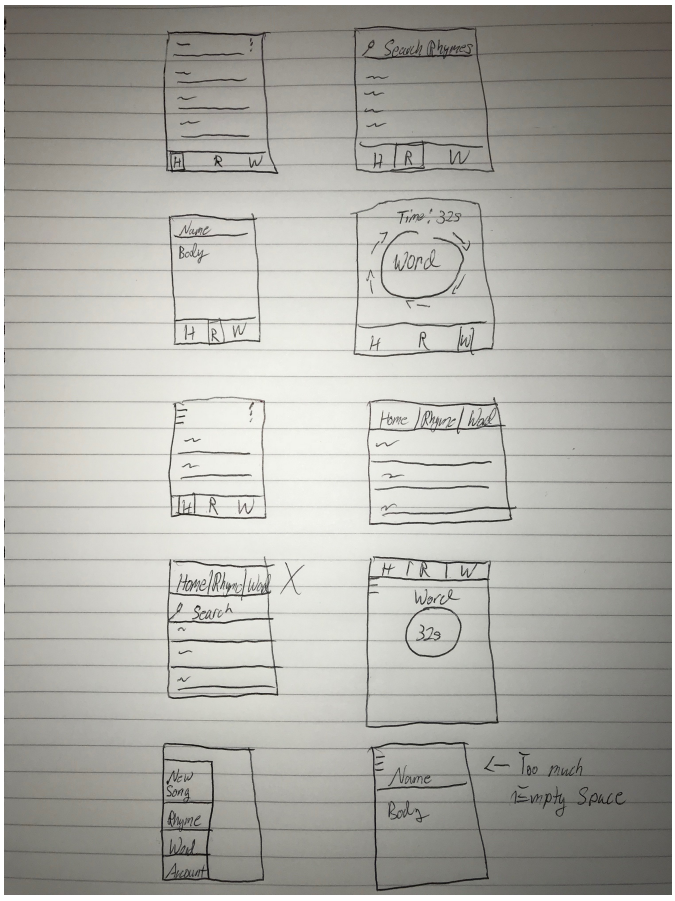

Figure 1: **Sketch 1**

2.3 Wizard-Of-Oz

Third, I used the Wizard-Of-Oz method with my sketches to see how different songwriters I knew would try and interact with them as if it was a working application. The majority of them told me that they preferred the bottom navigation bar rather than the other navigation methods in going between the different tools.

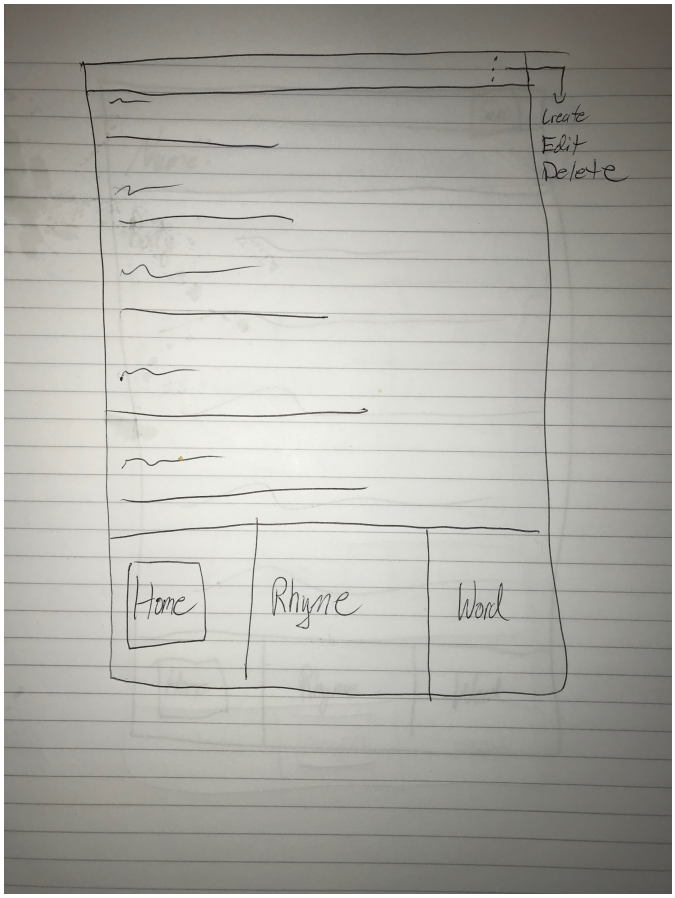

Figure 2: **Sketch 2**

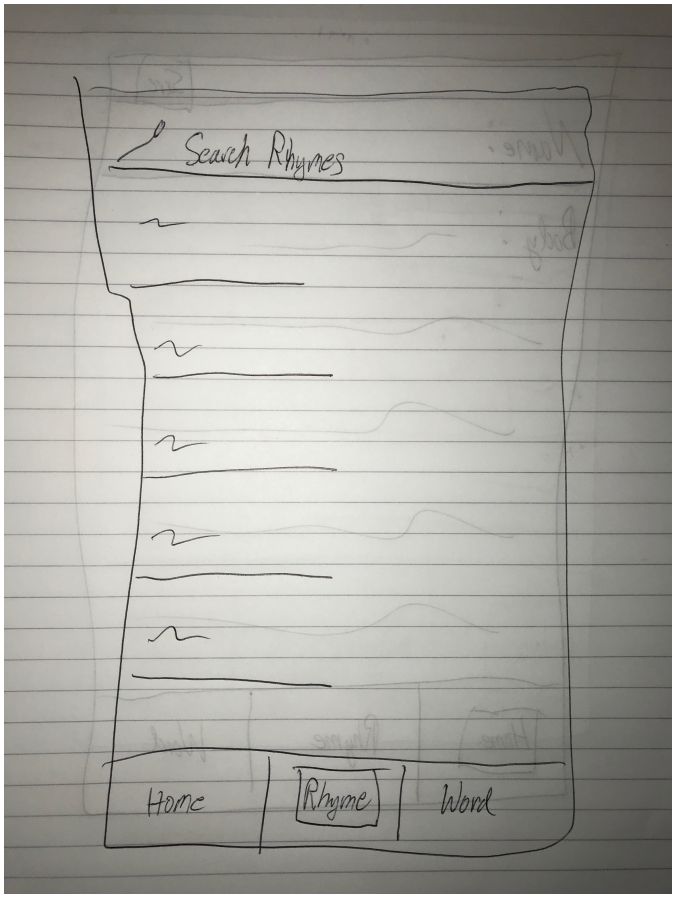

Figure 3: **Sketch 3**

2.4 Structure

My next task was looking for any video resources on creating a basic note-taking app in order to get an idea of what my class structure would look like and how they would interact with each other. I had found a helpful two-part tutorial on YouTube that taught me some basics in what I would need to implement to get the songwriting portion of my application going [2][3]. The videos showed me how to properly go about saving songs and accessing them as well as how to populate lists which would help me in both the list of written songs and as well as the list of rhymes for the rhyme finding tool.

2.5 API

Concurrently, I needed to find an API that would be able to provide the data I needed in order to provide the functionality for my rhyme finding tool. In the past, I had used different sites dedicated to this task and looked into whether they were using any publicly accessible web API. One popular website that I found called RhymeZone.com used an API called Datamuse which is focused on being a "a word-finding query engine for developers"[4]. I had found out that the data it outputs was written in JSON which was perfect since I had known about the fact that Java has its own JSON library.

2.6 Use Case Diagram

Lastly, I tried making a use case diagram to help me understand the flow of how the features of the app would work with the user and with other features (Figure 6). I wanted to make sure that I listed everything the application would have in order to keep track of any edge cases I might encounter during implementation.

3 Implementation

3.1 Features

A major element in how the application functions, though not being the first thing I implemented, is the Features class. The class handles most of the application's background tasks such as accessing, saving, updating, and deleting files and as well as parsing through data gathered from the web API.

3.2 MainActivity

The first part of my implementation was using the "Bottom Navigation Bar" template in Android studio as my MainActivity and setting it up with a FrameLayout so that the different tabs in the bar would trigger the different tools and populate the respective tool as a fragment in the FrameLayout.

3.3 Song

The next part was the Song class which acted as an object for any song created. It extends Serializable so that the songs can be saved as files in the application's directory, and it hold variables that store the name, lyrics, time and the date of when the song was created.

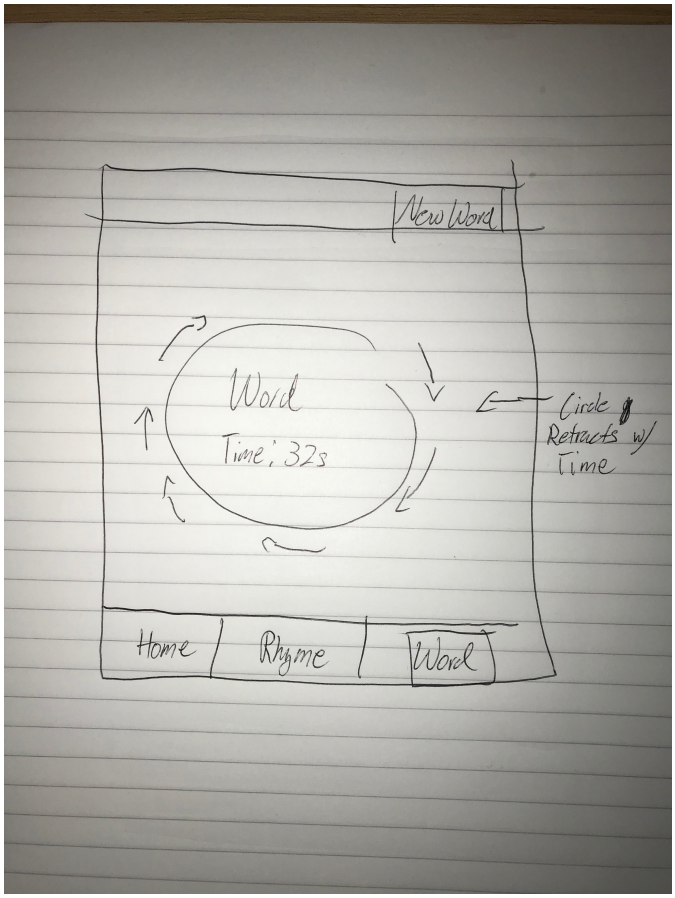

Figure 4: **Sketch 4**

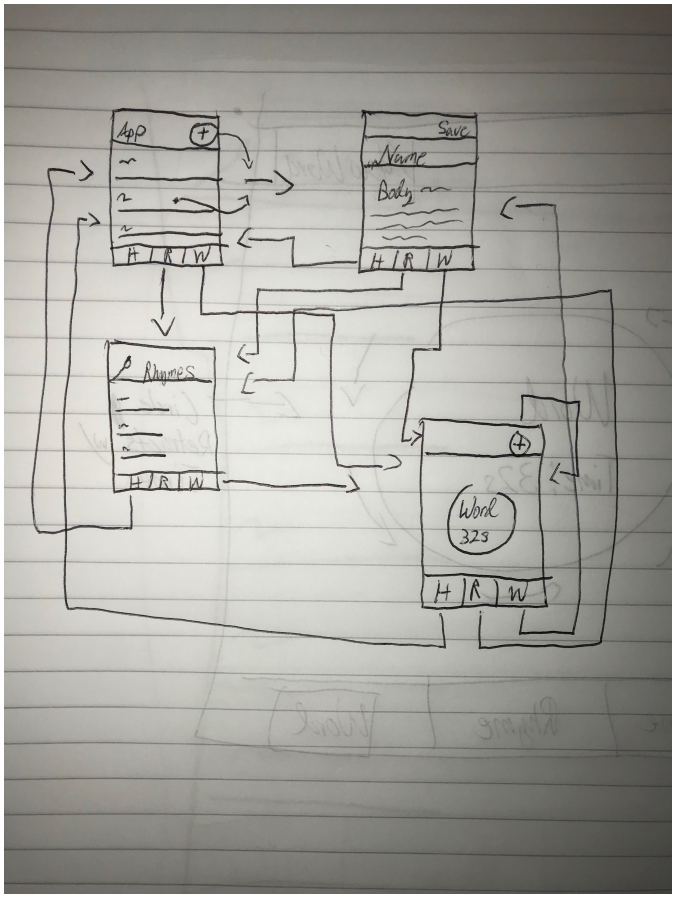

Figure 5: **Sketch 5**

3.4 SongList

The first tab became the songwriting tool which was achieved by creating the SongList fragment (Figure 7). The fragment would act as the list of any saved songs by using a ListView. This was followed by creating a menu option for the SongList that would act as the way to create a new song. The SongList then required a SongAdapter class that would populate the list and a WritingActivity for creating a new song.

SongAdapter The SongAdapter class was then implemented by extending ArrayAdapter and using a method from the Features class to grab all the saved songs in application's file directory.

3.5 WritingActivity

The WritingActivity class would serve as the activity for creating, updating, and deleting a Song (Figures 8, 9). The Song would then be able to be saved using a method in the Features class that uses the time of when the Song is created as the name of the file. This allows easy access to a pre-existing Song that the user might want to edit or delete when being selected from the SongList. I then made menu options for both saving and deleting the Song that is currently displayed in the activity.

3.6 RhymeGenerator

Next was creating my RhymeGenerator fragment for the second tab in the navigation bar (Figures 10, 11). It uses both a SearchView for letting the user search rhymes for any word and a ListView for listing any available rhymes. The fragment has an inner class called APITask which extends AsyncTask in order to read the information from the web API. Then the word that the user is searching for is passed into the URL of the API in order to grab the data for that word. The data gathered is then parsed using a method in the Features class and then returned as a list of rhymes.

RhymeAdapter This prompted me into additionally making a RhymeAdapter class for populating the list of rhymes. The RhymeAdapter class then did the same as the SongAdapter but instead uses the list of rhymes.

4 Implementation Evaluation

4.1 User Feedback

When testing the app with users there were a few points of improvement that I noted. The most common being that the user should be able to not only rearrange the order of their songs but also categorize them. A direct comparison I normally received was the functionality of Google Keep allowing users to group, color, and organize their notes in whatever way the user wants. Another point that was discussed when testing my apps with users was being able to import their own files from other applications or sites. For example, allowing the application to request access to a user's notes on Google Keep and copying them into the application's list. The last common point of feedback I had gotten from users was the

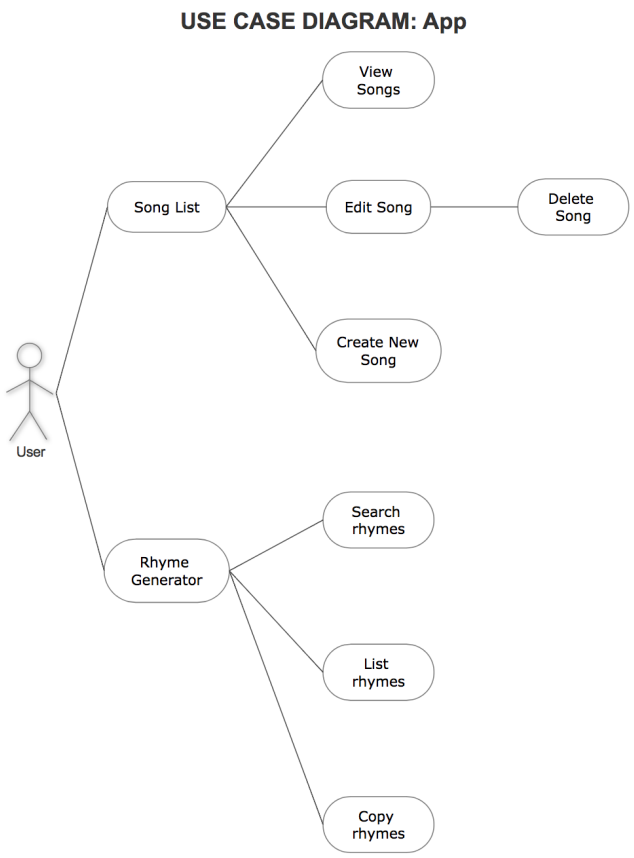

Figure 6: **Use Case Diagram**

ability to be able to store the songs they saved to some sort of cloud that would be attached to an account they could create.

4.2 Similar Apps

Since I had gotten many comparisons to Google Keep, I had decided to look into what other features it provided for users that I had not implemented myself. One element that stood out the most is the ability to search the users' own notes. Looking at other songwriting apps such as Rhymer's Block on iOS, I noticed that I could have tried listing rhymes in real time within the WritingActivity which would have allowed users a more streamlined experience. Lastly, an app on the Google Play Store called SongSpace is known for allowing users to share their songs with other users in order to collaborate. If I were to have incorporated this feature into my app, it would have granted more connectivity between users.

4.3 Improvements

There were some improvements that I wanted to make on my application but did not get the opportunity to create. First, the RhymeGenerator fragment restarts every time it is no longer in view, which impairs a smooth user experience. Second, another improvement for user experience would have been highlighting an item in the RhymeGenerator list whenever the user clicked on the item. Lastly, if I had more skill in designing, I would have wanted to create more relevant and modern icons for the different menu options rather than

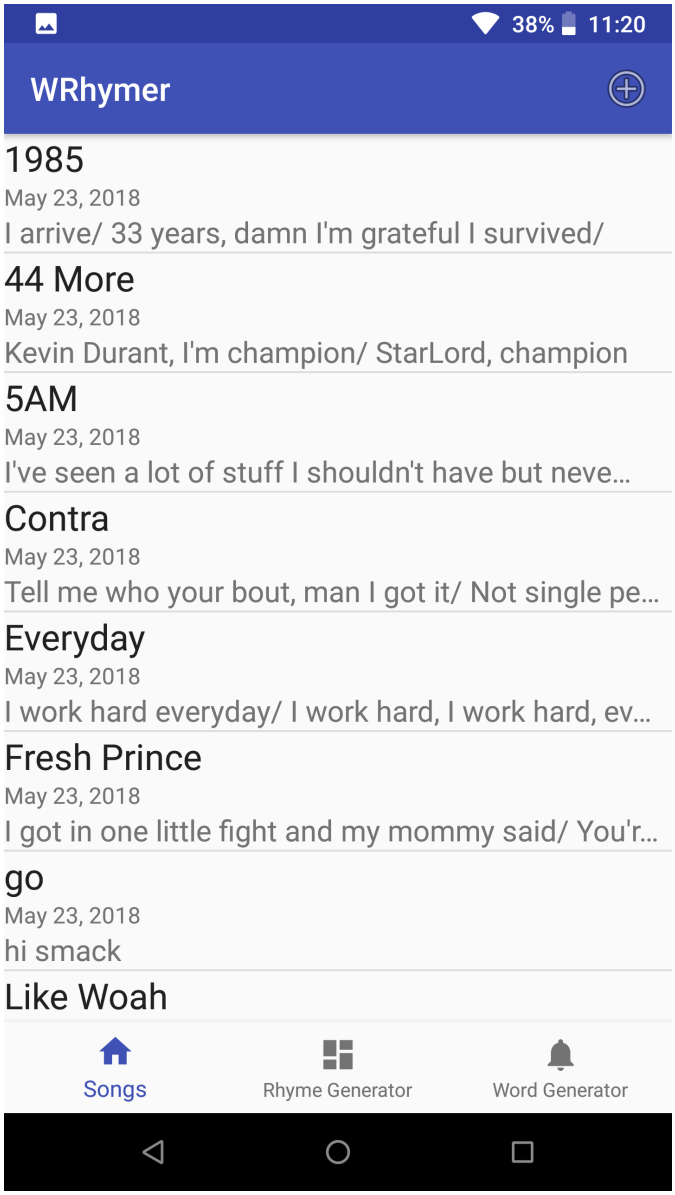

Figure 7: **Song List**

using the default icons that were available in Android Studio.

5 Personal Evaluation

5.1 Reflection

When it came to what I learned during the development process, one of the main lessons I took away was how to plan a fully functioning application rather than implementing the usual single task applications I had experienced making in the labs. This was the first time I was putting into practice and combining the different methods I learned in class into one cohesive application. Something else I realized was how the extension to my project submission affected how I approached my development process. Because of my extensions, I had the liberty of being able to take the exam before completing and submitting my application. The preparation for the exam gave me more insight into the different theories

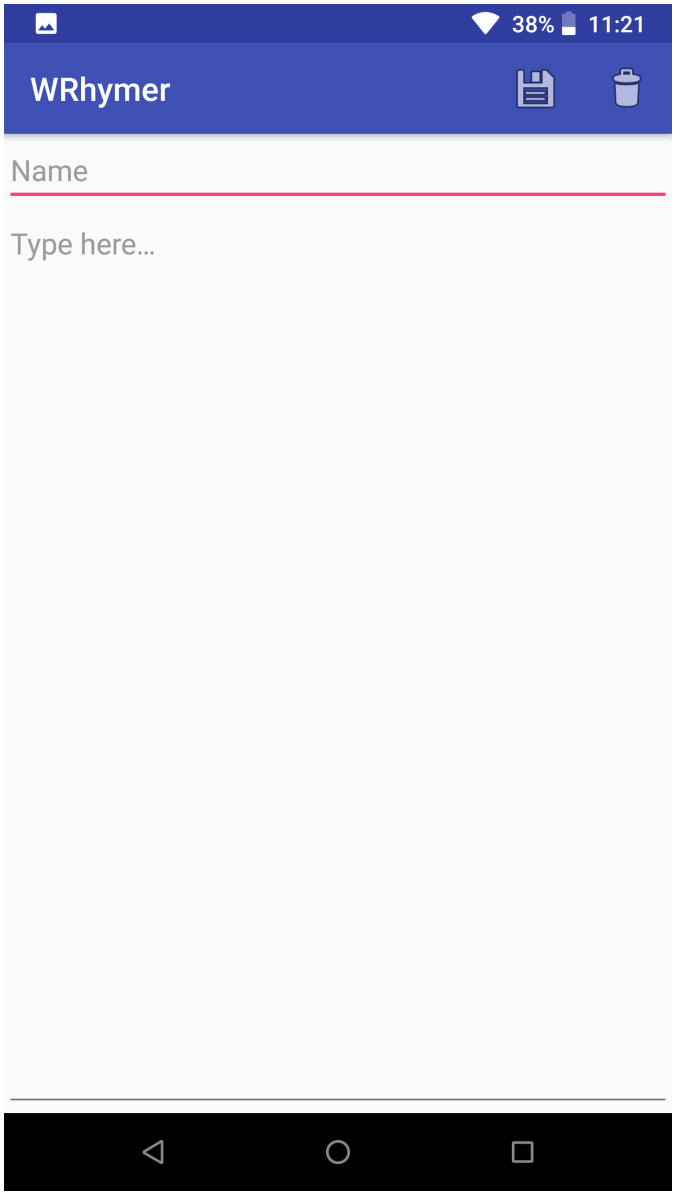

Figure 8: **Writing New Song**

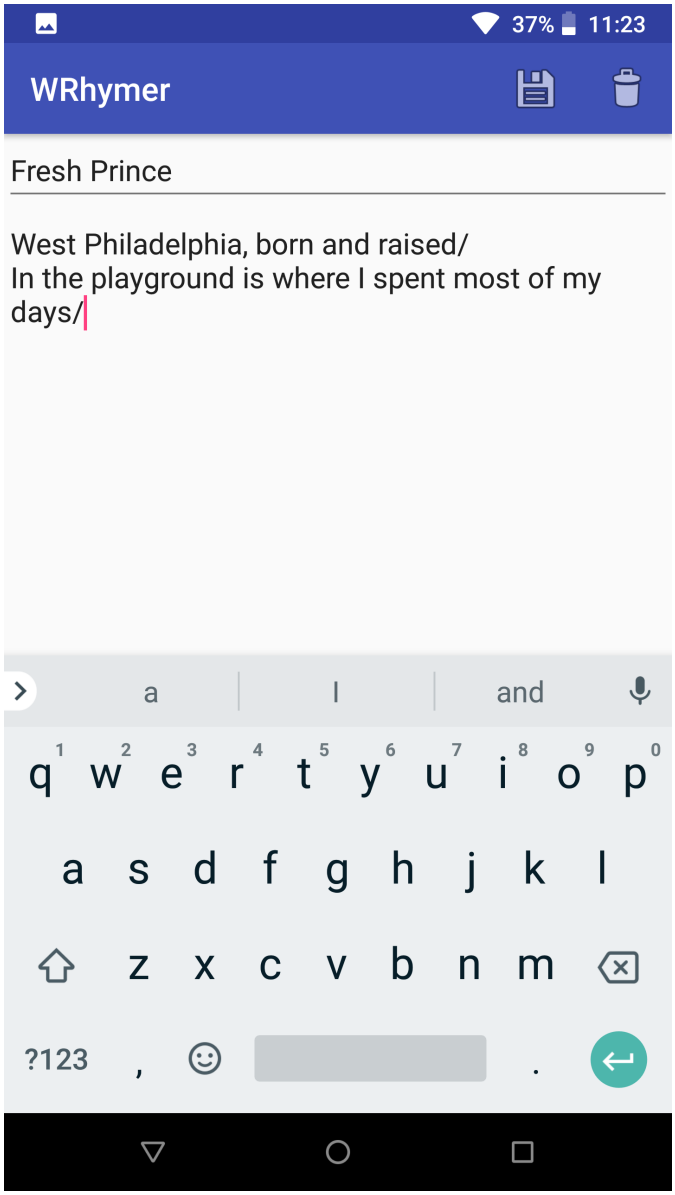

Figure 9: **Editing Song**

behind planning and properly mapping out development for applications in general. This had better equipped me when structuring and tackling the development for my own project. Therefore, when it came to my research for the app, I had a stronger idea of where to start looking. For example, if I had not studied for the exam before submitting my application, I would have not thought to look into design patterns that are listed on the Android developers website before sketching out my ideas for the design, nor would I have possibly remembered that the Wizard-Of-Oz method could be used to make sure that the interface I was planning would turn out to be user-friendly. This in turn gave me more of a solid start to my development and gave me a clearer path to what I set my goal for the application to be.

5.2 Challenges

On a more technical level, I learned how to seek and understand data given from an API and how to tailor that data for my specific use. I had always known the theory behind using APIs and what they offer but never knew how to apply it and use them for a custom goal. It was quite daunting at first grasping how to implement an API for my use but I found that through reading general articles of how to use an API and looking at example code on GitHub, I was able to utilise the API I had found for the data that I needed. I feel that that was the method I used mostly when faced with other challenges such as understanding how to use the JSON library in Java to my advantage when grabbing the data from the API. The use of APIs in general were new to me so with the help of both articles and documentation, I was able to teach myself exactly what I needed in order to fully implement the RhymeGenerator portion of my project.

5.3 Performance

My performance was something that I always felt could have been improved upon. The major drawback I had when working on the project was when it came to dealing with errors and the like. I felt that I didn't do the best job at debugging them. Usually what would end up happening in the first half of my development is that I would get lost in my code or not

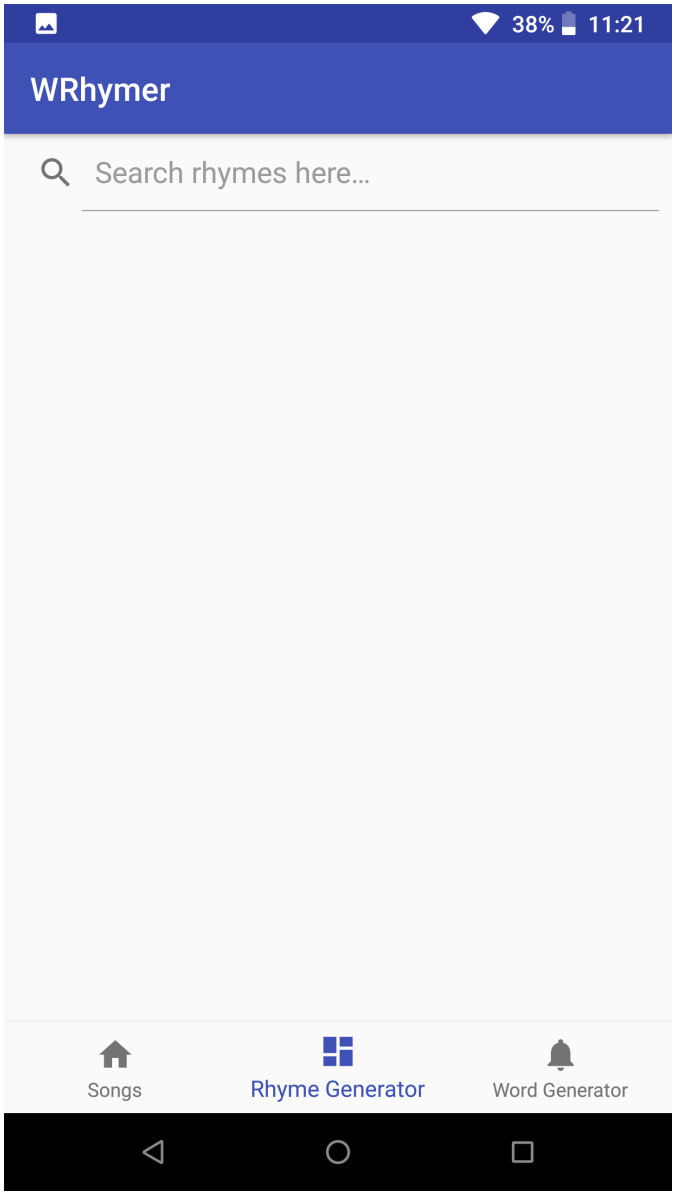

Figure 10: **Rhyme Search**

Figure 11: **Rhyme List**

know where to start when debugging certain runtime errors. This would end up with me deleting the portion of code that was causing the error and then trying to reprogram it from scratch. It was more of a cause of my anxiety that made me do that rather than actually not knowing how to debug the issue. Of course this slowed me down during my development and could have been avoided if I had taken more time out learning how to debug runtime errors properly rather than starting parts of my program all over again.

5.4 Conclusion

On the whole, with the resources I had and the fact that I had fallen behind in the semester due to my extenuating circumstances, I believe that I was able to complete this project with innovation and determination and learned about my capabilities when programming an application.

References

[1] L. Spradlin, "A primer on android navigation." https://medium.com/google-design/a-primer-on-android-navigation-75e57d9d63fe, July 2018. [Online; last accessed 22-May-2018].

[2] EMBEDONIX, "Android tutorial 2 - note taker application — part 1." https://www.youtube.com/watch?v=ysEeCph0GPA, May 2016. [Online; last accessed 22-May-2018].

[3] EMBEDONIX, "Android tutorial 2 - note taker application — part 2." https://www.youtube.com/watch?v=4SLa3EucLI0, June 2016. [Online; last accessed 22-May-2018].

[4] "Datamuse api." https://www.datamuse.com/api/. [Online; last accessed 22-May-2018].

[5] "Display the current time and date in an android application." https://

stackoverflow.com/questions/2271131/display-the-current-time-and-date-in-an-android-application. [Online; last accessed 22-May-2018].

[6] "Add a menu to an empty activity." https://stackoverflow.com/questions/36090542/add-a-menu-to-an-empty-activity. [Online; last accessed 22-May-2018].

[7] "Android add placeholder text to edittext." https://stackoverflow.com/questions/8221072/android-add-placeholder-text-to-edittext. [Online; last accessed 22-May-2018].

[8] "Go back to specific activity from stack." https://stackoverflow.com/questions/23826483/go-back-to-specific-activity-from-stack. [Online; last accessed 22-May-2018].

[9] "How to change position of toast in android?." https://stackoverflow.com/questions/2506876/how-to-change-position-of-toast-in-android. [Online; last accessed 22-May-2018].

[10] "How to show a progress spinner in android, when doinbackground() is being executed." https://stackoverflow.com/questions/11752961/how-to-show-a-progress-spinner-in-android-when-doinbackground-is-being-execut. [Online; last accessed 22-May-2018].

[11] "Copy and paste." https://developer.android.com/guide/topics/text/copy-paste. [Online; last accessed 22-May-2018].

[12] O. Ogbo, "How to use a web api from your android app." https://www.androidauthority.com/use-remote-web-api-within-android-app-617869/, June 2015. [Online; last accessed 22-May-2018].