a mobile application to refine music suggestions based upon user preferences and feedback

[date]

**Abstract**

**Declaration**

*I certify that all material in this dissertation which is not my own work has been identified.*

**Contents**

[**1 Introduction** 3](#_Toc36938963)

[**1.1** **Motivation** 3](#_Toc36938964)

[**1.2** **Broad Objectives** 3](#_Toc36938965)

[**2** **Literature & Specification** 3](#_Toc36938966)

[**2.1** **Literature Review Summary** 3](#_Toc36938967)

[**2.2** **Project Specification** 3](#_Toc36938968)

[**3** **Design** 3](#_Toc36938969)

[**3.1** **Pathways & UI** 3](#_Toc36938970)

[**3.2** **Recommendation Engine** 3](#_Toc36938971)

[**4** **Development Lifecycle** 3](#_Toc36938972)

[**5** **Implementation** 3](#_Toc36938973)

[**6** **Testing & Evaluation** 3](#_Toc36938974)

[**7** **Critical Discussion & Conclusion** 3](#_Toc36938975)

# **1 Introduction**

### **Motivation**

Music is a central part of a huge number of people’s lives. People listen to music when working, exercising, partying and during so many other daily activities. Searching for new music has evolved in the past 50 years away from vinyl record shops to digital formats such as CD’s, MP3 players and now streaming services. An (IFPI, 2018) study found that global average music consumption per week sits at almost 18 hours a week – or 2.5 hours a day. It also forms a huge economic industry; Spotify disclosed an annual revenue of almost $6 Billion in 2018, with total users standing at over 200 million in the fourth quarter of that year (Spotify Technology, S.A, 2018).

The way in which users can be exposed to new music content has changed along with the evolution in consumption. During the days of vinyl records, shopkeepers would recommend fresh music they have recently taken into stock. Now, modern music platforms such as Spotify or Deezer provide suggestions or recommendations to users on new music to listen to in the form of a recommendation engine. Platform-curated recommendations often form a significant chunk of consumption; (Economic Times, 2018) established a figure of 15% for the Gaana music platform. These recommendations are often based upon user feedback – known as collaborative filtering.

Whilst this technique is likely to result in a majority of listeners receiving desired results, making it reliable, it is often limited outside of the generic listeners use-case. Music tracks have attributes which aren’t created in artificial big-data situations; for example, signal frequencies can be analysed. The Spotify API is a fantastic case study of this; it stores data on danceability, valence and energy to give some examples, as well as storing more traditional, collaborative attributes such as popularity (Spotify, 2019). These content-based attributes can be combined with collaborative ones in order to provide more interesting and demand-specific suggestions.

Considering use cases, the initial motivation for this project came from a personal interest in discovering music based upon similarities in the content, rather than music that was popular in the genre I was listening to or the region I was listening in. As a DJ, I have found that traditional collaborative recommendations typically show me music I am already aware of. In designing a platform to discover music based upon its content, I hope to be able to enhance my ability to discover more unknown music which would fit into a specific vibe with which I am attempting to play.

### **Aims & Objectives**

[REWRITE -> 1x MAIN AIM, 4-5x OBJECTIVES]

In the broadest terms, the aim of this project is to design and implement a platform to create refined music suggestions using a recommendation engine biased towards a content-based filtering technique, against the industry-preferred collaborative filtering approach. The (IFPI, 2018) study referenced earlier also finds that 27% of total music listening time is on mobile devices; accounting for the continued use of non on-demand services such as radios, that suggests that a mobile application would be the most commercially sensible development platform. In order to provide ease-of-use to users of the application, and to take advantage of the huge amount of resources available for developing such an application through music platforms, tying the application to (at least) one platform through an API should be another of the broad objectives.

# **Literature & Specification**

### **Literature Review Summary**

Recommendations engines for music are inherently a big data problem; (Sagiroglu & Sinanc, 2013) define big data as ‘data sets large, more varied and complex structure with the difficulties of storing, analysing and visualizing for further processes’. Using earlier figures, users listen to just 22.5 minutes of recommended music a day, showing the challenge in creating good concise recommendations. Considering the four key areas of big data, volume clearly presents the biggest challenge; (Spotify, 2019) has over 50 million individual tracks stored. Veracity is interesting in the context of generated attributes being subjective.

Data characteristics in music can be split into ‘content-based’ and ‘collaborative’ (Lin, et al., 2014). Content-based attributes, such as ‘danceability’, refer to the content of music, whereas collaborative attributes, such as ‘popularity’, denote user preferences. Content-based attributes can be split into factual (musical key) and subjective (energy). Subjective attributes are the result of complex analysis to generate comparative values; the above examples are results from the Spotify API (Spotify, 2019).

The ‘cold start problem’ is a common issue facing recommendation engines, described as the ‘need to accumulate personal information in advance’; industry leaders such as Spotify & Apple Music face this problem (Wang, et al., 2014) (Jenkins & Yang, 2016). Existing literature shows the requirement of engines to use a combination of both techniques. One example would be to store users’ listening history to prevent repeat recommendations - (O'Bryant, 2017) proposes use of such data in making informed recommendations.

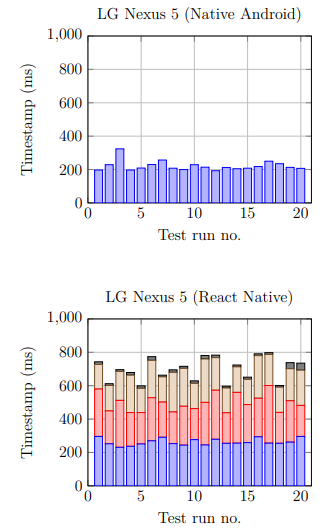
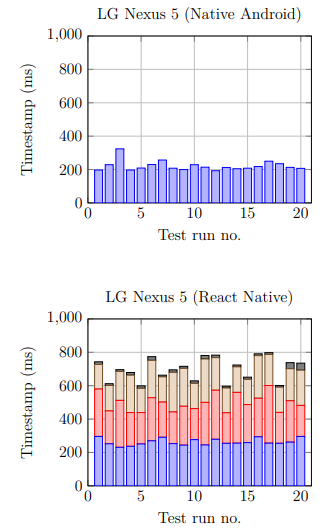
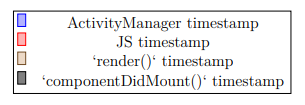
Existing literature on designing music recommendation engines spans from complex mathematical designs to linear, iterative ones. (Kodama, et al., 2005) discuss a system defining a ‘mood’ of a song by extracting values including tonality, dispersion of chords and signal levels; they then proceed to ask a user to select from ‘reference words’ to describe how the song makes them feel. (Kaji, et al., 2005) discuss a system utilising a three-stage system to generate and refine playlists through generation, transcoding and data analysis.

Often discussed in literature is the tendency of collaborative-biased engines to favour ‘popular’ music. (Celma, 2008) uses Amazon’s website as an example of this. (Soleymani, et al., 2015) propose a heavily mathematical, content-based solution to the cold-start problem, using ‘modulation analysis to extract timbral features’. They argue that use of attributes more complicated than genre provide better results for users.

Ensuring the application can reach the maximum number of potential users is paramount to success as a commercial project. Android & iOS devices make up >99.9% of all smartphone sales internationally (Gartner, 2018). Further, Android units make up approximately 86% of this (IDC, 2019). (Majchrzak, et al., 2017)’s analysis of contemporary cross-development platforms concludes that React Native would be the most suitable development platform for an application targeting both Android & iOS; specifically, its development community is strong in comparison to Fuse & Ionic.

(Danielsson, 2016) highlights the strengths of the React Native platform in comparison to native Android development in efficiency; stating that, from an example ‘less then half of the amount of code was used [in writing the React Native Application]. Despite this, he notes that ‘there are some faults in React Native’, as it is a relatively newer platform, although Danielsson’s paper is 4 years old at the time of writing, making React Native just 1 year old at the time (Papp, 2017).

Performance differences between native Android and React Native development lends towards single platform development; (Eskola, 2018) highlights launch times as an area where React Native suffers, with applications ‘load[ing] significantly slower than a native Android application’ [1]. Eskola suggests the use of ‘splash screens’ in alleviating performance deficits, as well as stressing the increasing negligence of such effects as device performance increases.



[1]

### **Project Specification**

# **Design**

As a project heavily leaning towards software engineering, as opposed to research, design decisions should be made with a commercial viewpoint at the forefront of the process.

### **Pathways & UI**

### **Recommendation Engine**

# **Development Lifecycle**

Delivering a software project efficiently, wholly and timely requires good development practices throughout. Two common lifecycle methods are used to achieve this: Waterfall and Agile. Waterfall is a traditional lifecycle method; linear in nature, it is a ‘sequential […] process […] flowing increasingly downwards […] through a list of phases that must be executed in order […]’, where ‘requirements [should be] defined and analysed prior to any design or development’ (Bassil, 2012) (Ruparelia, 2010). On the other hand, Agile is ‘based on the idea of incremental and iterative development’, dividing the lifecycle into ‘smaller parts, called increments’ (Leau, et al., 2012) (Beck, et al., 2001). This allows for changing requirements and creates deliverables more regularly than Waterfall. Often accused of being documentation-light, (Vijayasarthy & Turk, 2008) claim code often acts as documentation.

# **Implementation**

# **Testing & Evaluation**

### **Performance Evaluation & Iterations**

# **Critical Discussion & Conclusion**

Bassil, Y., 2012. A Simulation Model for the Waterfall Software Development Life Cycle. *International Journel of Engineering and Technology,* 2(5), pp. 742-749.

Beck, K. et al., 2001. *Manifesto for Agile Software Development.* s.l.:s.n.

Celma, O., 2008. *Music recommendation and discovery in the long tail,* Barcelona: UPF.

Danielsson, W., 2016. *React Native application development - A comparison between native Android and React Native,* Linköping: Linköpings universitet.

Economic Times, 2018. *Music app Gaana introduces new features to improve user experience.* [Online]   
Available at: https://economictimes.indiatimes.com/industry/media/entertainment/music-app-gaana-introduces-new-features-to-improve-user-experience/articleshow/66813088.cms?from=mdr  
[Accessed 13 November 2019].

Eskola, R., 2018. *React Native Performance,* Aalto: Aalto University School of Science.

Gartner, 2018. *Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018.* [Online]   
Available at: https://www.gartner.com/en/newsroom/press-releases/2018-05-29-gartner-says-worldwide-sales-of-smartphones-returned-to-growth-in-first-quarter-of-2018  
[Accessed 11 November 2019].

IDC, 2019. *Smartphone Market Share.* [Online]   
Available at: https://www.idc.com/promo/smartphone-market-share/os  
[Accessed 11 November 2019].

IFPI, 2018. *Music Consumer Industry Report 2018.* [Online]   
Available at: https://www.ifpi.org/downloads/music-consumer-insight-report-2018.pdf  
[Accessed 7 November 2019].

Jenkins, E. & Yang, Y., 2016. *Creating a Music Recommendation and Streaming.* Porto, Springer.

Kaji, K., Hirata, K. & Nagao, K., 2005. *A music recommendation system based on annotations about listeners' preferences and situations.* Florence, IEEE.

Kodama, Y. et al., 2005. *A music recommendation system.* Las Vegas, NV, IEEE.

Leau, B. Y., Loo, W. K., Yip, T. W. & Tan, S. F., 2012. *Software Development Life Cycle AGILE vs Traditional Approaches.* Singapore, IACSIT Press.

Lin, N., Tsai, P.-C., Chen, Y.-A. & Chen, H. H., 2014. *Music recommendation based on artist novelty and similarity.* Jakarta, IEEE.

Majchrzak, T. A., Biørn-Hansen, A. & Grønli, T.-M., 2017. *Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks.* Hawaii, 50th Hawaii International Conference on System Sciences.

O'Bryant, J., 2017. *A survey of music recommendation and possible.* [Online]   
Available at: https://jacobobryant.com/about/mrs.pdf  
[Accessed 10 November 2019].

Papp, A., 2017. *The History of React.js on a Timeline.* [Online]   
Available at: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/  
[Accessed 16 November 2019].

Ruparelia, N. B., 2010. Software Development Lifecycle Models. *Software Engineering Notes,* 35(3), pp. 8-13.

Sagiroglu, S. & Sinanc, D., 2013. *Big Data: A review.* San Diego, IEEE.

Soleymani, M., Aljanaki, A., Wiering, F. & Veltkamp, R. C., 2015. *Content-based music recommendation using underlying music preference structure.* Turin, IEEE.

Spotify Technology, S.A, 2018. *Shareholder Letter Q4 2018.* [Online]   
Available at: https://s22.q4cdn.com/540910603/files/doc\_financials/quarterly/2018/q4/Shareholder-Letter-Q4-2018.pdf  
[Accessed 9 October 2019].

Spotify, 2019. [Online]   
Available at: https://newsroom.spotify.com/company-info/  
[Accessed 9 November 2019].

Spotify, 2019. *Get Audio Features for Several Tracks.* [Online]   
Available at: https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-audio-features/  
[Accessed 10 October 2019].

Vijayasarthy, L. R. & Turk, D., 2008. Agile Software Development; A Survey of Early Adopters. *Journel of Information Technology Management,* 19(2), pp. 1-8.

Wang, M. et al., 2014. Context-Aware Music Recommendation with Serendipity Using Semantic Relations. *Lecture Notes in Computer Science*, 21 May, pp. 17-32.