University of Brighton

Project Proposal

Recommusic

23rd January 2014

Andrei Martins Silva
Student number 13837492

CI360 - Mobile Application Development
School of Computing, Engineering and Mathematics

1. Elevator pitch

The proposed app is intended to recommend songs to the user based on what the user usually plays on its smartphone. The app will extract the most played tracks and submit it to a web service. The role of the web service is to find a similar set of songs as answer for the song requested. Then, the answer will be processed and displayed on the UI as a list of more recommended music with links to stream and buy.

2. Rationale

The app market is full of good options for people who want to play songs locally on the smartphone such as Android Music Player© and Rocket Music Player©. For people who like to stream music from the cloud SoundCloud© and Grooveshark© are good options. On one hand, the user who wants to play locally needs to put his preferred .mp3 files on the storage in order to be able to play it. On the other hand, the user has a much larger collection of songs available to stream, but requires internet connection.

Both classes of apps are good for its purposes. However, they don't fulfil an emergent need of the user: music recommendation.

The life in 21th century is much busier than have ever been. Huge amounts of data are popping up every second. The music follows this rhythm. SoundCloud states that "more than twelve hours of audio is posted on its platform every minute" in your description at PlayStore©.

Of course the user don't have enough time to keep up to date. The user knows what he wants to listen, but he doesn't know how to find it. Thus, an app that does this 'dirty work' is very welcome. Every android smartphone user who has songs in its storage and wants to have some fresh songs recommendations is able to use the app.

3. Application

The Recommusic – mix between the words recommendation and music - app consists of four main parts: the input, a web service handling, processing and the output.

In a typical app, the input is some text typed by the user, or a set of options chosen by the user. In this app, the input is a set of .mp3 files on the user smartphone's storage.

With this input in memory, the app is going to verify the most played songs on the smartphone by using the data contained in the .mp3 file. [1]

The set of data gathered from the .mp3 will then be submitted to a web service provided by Last.fm© [see resources section].

Last.fm© is a widely used platform for streaming music across the internet. Last.fm© also provides a recommendation service which is going to be used here to recommend music based on the songs the user usually listens.

The output from the web service is a set of recommended songs. This set of songs is organized in a list and displayed to the user.

It is not in the scope of the app to provide a music player. So, links to stream the recommended song is provided for the user. Also, for the users that like to buy songs in the .mp3 format, the app provide links to online stores.

Music discovery is an emergent subject in the digital industry. Much research has been done around this subject [2, 3, 4]. On the last decade a lot of web sites have also emerged such as Last.fm©, Pandora© and Spotify©.

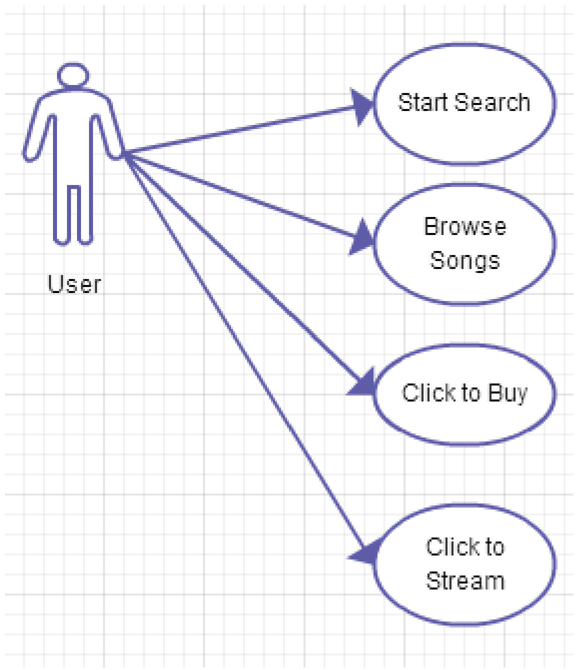
The last one, Spotify© is a good example of music discovery app for the Android platform. It has a huge library that allows the user to find and stream its favourite songs. It is also possible to share playlists with other users. Despite of present these features, we are more concerned with the music discovery feature provided. Spotify© is capable of recommend music based on what the user is listening. However the data about what the user plays is hold on the Spotify's database. In other words, the recommendation is only performed based on the music played using Spotify player.

Recommusic differentiate itself because it takes the input for recommendation directly from the source .mp3 files. This means that any music player that modifies the .mp3 files will contribute to a better recommendation.

4. Use case

The Recommusic music discovery app is as simple as it seems to be. Thus, the app is suitable for situations where the user just need to kill some time, e.g. waiting for the bus, waiting in a queue or lunch time.

Basically, the user has four actions available. The first one is the 'Start Search'. In order to perform this action the user needs to click the button 'Start' in the app home screen. Then, the app is going to retrieve a list of recommended songs in relevance order. At this stage, the user can 'Browse Songs' and 'Click to Buy' from an online store or 'Click to Stream' from an online radio.

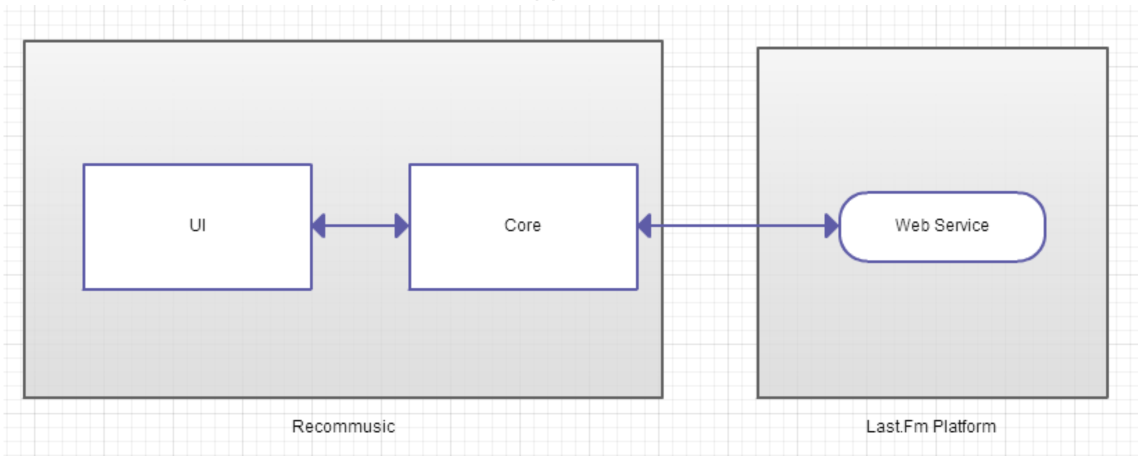


As an example, let's say John is a user. He has a tiny selection of songs stored in its SD storage and wants to discover new songs using the app. First, John clicks on the Start Button. The app verifies the most played song among its mp3 files and then returns a list of the most recommended songs. In this case, the most played song was

‘Jailhouse Rock’ from ‘Elvis Presley’. The returned list would contain the following songs: ‘Hound Dog’ – ‘Elvis Presley’; ‘All Shook Up’ – ‘Elvis Presley’; ‘Great Balls of Fire’ – ‘Jerry Lee Lewis’ and ‘Johnny B. Goode’ – ‘Chuck Berry’. Finally, John decides if he wants to stream, buy or just keep the recommended song in mind.

5. Technical overview

The Recommusic is an app strongly based in the web service provided by Last.fm api. Thus, the application is designed as an interface between the UI and the web service. The diagram below illustrates the application architecture.



The UI is the block responsible for displaying the components such as buttons, lists, text views and links. The commands from the user are intercepted in this block and sent to the application core. The data to be displayed as output is also handled by the user interface. The core acts as a bridge between the UI and the web service. The basic flow of the core is to get an input, transform, send to the web service, get the answer, transform again if needed and send to the UI. The internal actions of the web service is not clear for the app, because is provided by an external agent.

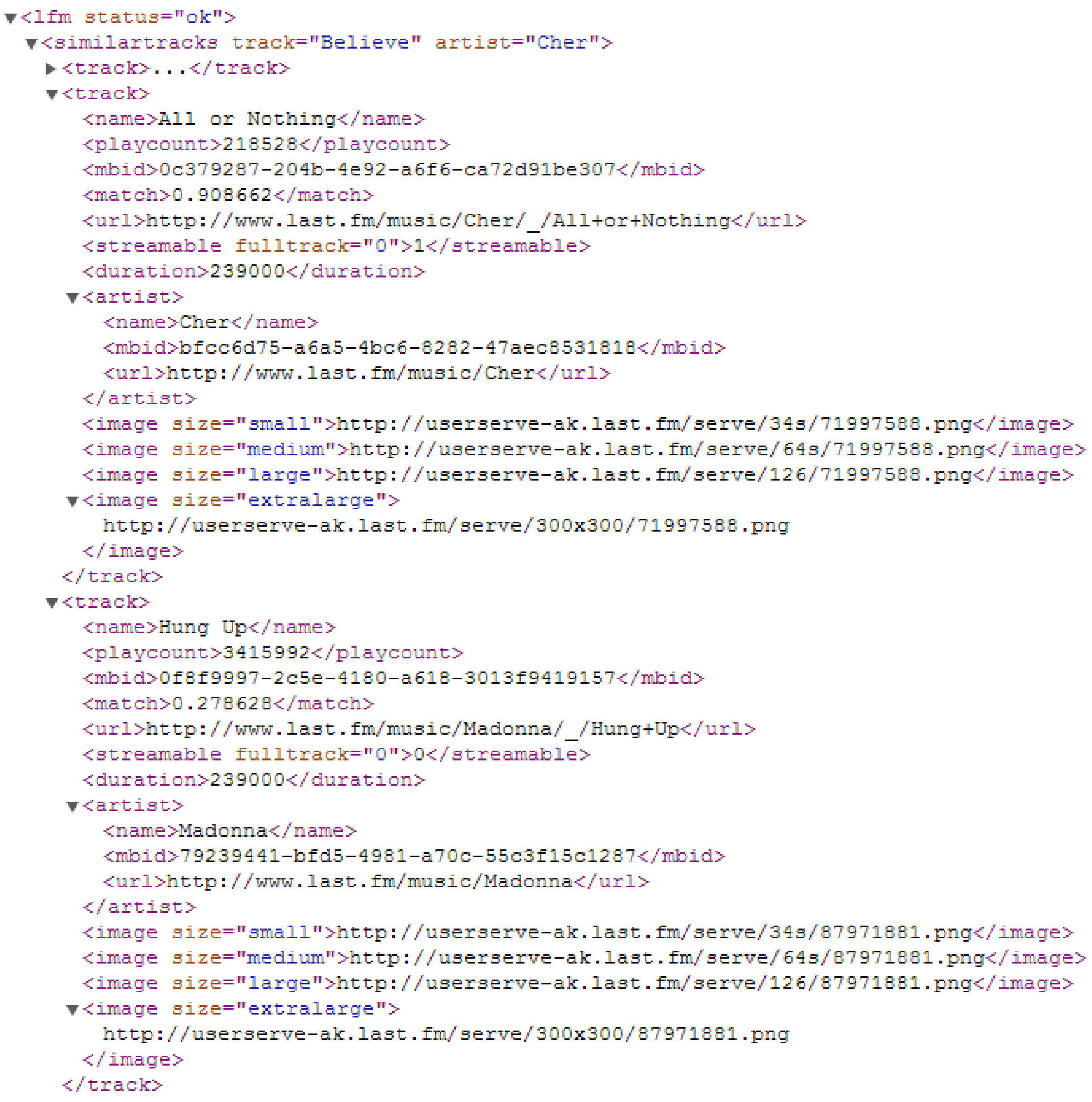
Persistance will not be required since all the information will be read from the .mp3 files.

As the input comes from the user .mp3 files, it makes necessary to know how to access the mp3 files through the file system and read the metadata held in the file.

An mp3 file can contain an ID3 tag that contains attributes like **song title**, **artist**, genre, album and **playcount**.

There are several JAVA libraries that provide classes to get this information [see resources section]. The library will be chosen later on the beginning of the development phase of the project. Once we get the playcount for every single song in the mp3, it is easy to determine what the most played song is.

The next step is to submit the song title and the artist to the method getSimilar of the Last.Fm api. This REST service will return an xml with an ordered list to the application Core. The list is returned by relevance order. Below is an example of the returned XML.



```
▼<lfm status="ok">
  ▼<similartracks track="Believe" artist="Cher">
    ▶<track>...</track>
    ▼<track>
        <name>All or Nothing</name>
        <playcount>218528</playcount>
        <mbid>0c379287-204b-4e92-a6f6-ca72d91be307</mbid>
        <match>0.908662</match>
        <url>http://www.last.fm/music/Cher/_/All+or+Nothing</url>
        <streamable fulltrack="0">1</streamable>
        <duration>239000</duration>
      ▼<artist>
          <name>Cher</name>
          <mbid>bfcc6d75-a6a5-4bc6-8282-47aec8531818</mbid>
          <url>http://www.last.fm/music/Cher</url>
        </artist>
        <image size="small">http://userserve-ak.last.fm/serve/34s/71997588.png</image>
        <image size="medium">http://userserve-ak.last.fm/serve/64s/71997588.png</image>
        <image size="large">http://userserve-ak.last.fm/serve/126/71997588.png</image>
      ▼<image size="extralarge">
          http://userserve-ak.last.fm/serve/300x300/71997588.png
        </image>
      </track>
    ▼<track>
        <name>Hung Up</name>
        <playcount>3415992</playcount>
        <mbid>0f8f9997-2c5e-4180-a618-3013f9419157</mbid>
        <match>0.278628</match>
        <url>http://www.last.fm/music/Madonna/_/Hung+Up</url>
        <streamable fulltrack="0">0</streamable>
        <duration>239000</duration>
      ▼<artist>
          <name>Madonna</name>
          <mbid>79239441-bfd5-4981-a70c-55c3f15c1287</mbid>
          <url>http://www.last.fm/music/Madonna</url>
        </artist>
        <image size="small">http://userserve-ak.last.fm/serve/34s/87971881.png</image>
        <image size="medium">http://userserve-ak.last.fm/serve/64s/87971881.png</image>
        <image size="large">http://userserve-ak.last.fm/serve/126/87971881.png</image>
      ▼<image size="extralarge">
          http://userserve-ak.last.fm/serve/300x300/87971881.png
        </image>
      </track>
```

In order to be able to call the Last.fm web service an http client for android is required. According to the recommendation from [5], HTTPURLConnection will be used to connect to the service and retrieve the xml. There are two options being considered to parse the xml. They are: Simple API for XML (SAX) or Document Object Model (DOM). At the moment, SAX seems to be better since its memory consumption is much fewer than DOM's.

Finally, the core is going to arrange the top tracks in a CustomListView showing to the user data about the recommended song such as song title, artist, album, and links to stream and buy.

6. Technical challenges

There are possible technical challenges that may compromise the success of this project along with mitigating actions:

Networking: How to use the web service efficiently considering user's package data constraints. If a user has data usage constraints, may be a good strategy to limit the answer from the xml, so the kilobytes transferred reduces dramatically.

XML Parsing: How to parse the answer from the web service? There are Java libraries available for this task, but it demands a certain learning curve.

ID3 tag consistency: The app needs a consistent data from the playcount contained in the ID3 tag. If the main music player of the user doesn't update the playcount, the recommendation may not be so good.

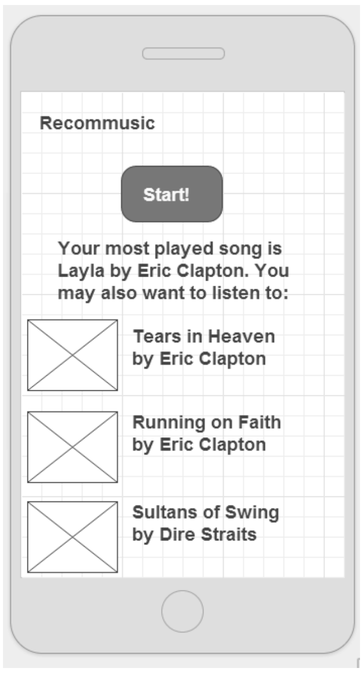
Concurrency: How to entertain the user while waiting for response? How to call the webservice without crashing the app? AsyncTask, Handler, Spinner and ProgressBar components may help with this task.

UI: It will be a challenge to organize the information about songs in list rows without compromise the user experience

Input: How to deal with a user that has no music files prior to the first app use?

7. UI design draft

The app is simple. Most of the work is done under the curtains, so one main screen may be enough to fulfil the user needs for this application. A sketch of this main screen is shown below:


Recommusic
Start!
Your most played song is Layla by Eric Clapton. You may also want to listen to:
Tears in Heaven by Eric Clapton
Running on Faith by Eric Clapton
Sultans of Swing by Dire Straits

8. Work plan

The project itself is divided in several phases. Each phase has a different impact on the project and all are sequentially connected and some tasks are dependant of the others.

The requirements analysis is the phase where we are concerned with gathering user requisites for the app. All aspects are considered, from user interface requirements to functional requirements going through non-functional requirements such as performance.

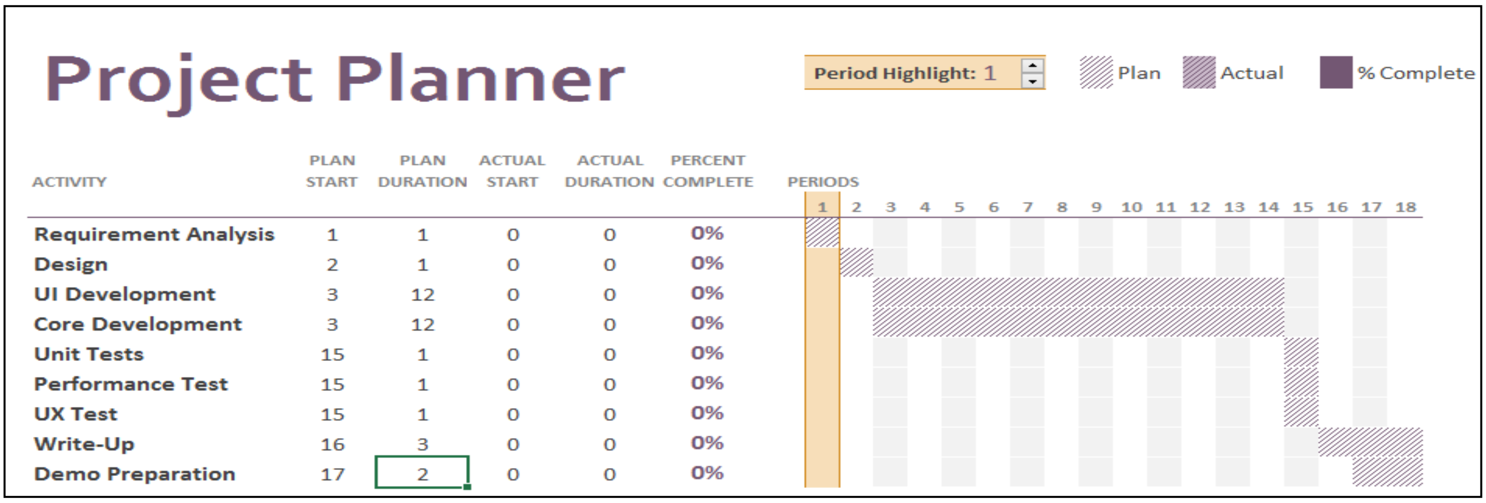
Once the requisites are gathered, it is time to design the application. In this phase, the app architecture will be designed, ensuring that all the requirements are properly supported. The design phase provide essential model for the development phase.

In the development phase, two tasks are done simultaneously: UI development, Core Development. This is because the app is a relatively simple program with high interaction between these two tasks. Although treat these tasks sequentially would benefit the organization and better project management, it would take too long to have functional working versions of the program.

After the development phase, starts the testing phase. At this stage, several tests will be carried out to ensure the software physically supports all the requisites gathered in the beginning of the project. Java Unit Tests are carried out to ensure that the logic and runtime code are in good form. Performance tests are carried out to verify parameters such as memory usage, battery usage, processing rates and mobile data required. The UX test is a set of tests intended to verify if the app complains with user experience guidelines adopted by the development community [6, 7]. These tests are performed simultaneously in terms of schedule.

At last, a final report will be written in order to register the software conceptual model, the development issues, tools used, tests carried out and a reflective analysis of what this project meant to the people involved after its execution. A demo preparation is also planned to show the capabilities and constraints of the final product.

Below a Gantt chart illustrates the project timeline from the beginning to the end of the project. The idea is to use this chart to control the project execution.



Project Planner

Period Highlight: 1 ▨ Plan ▨ Actual ■ % Complete

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Requirement Analysis	1	1	0	0	0%
Design	2	1	0	0	0%
UI Development	3	12	0	0	0%
Core Development	3	12	0	0	0%
Unit Tests	15	1	0	0	0%
Performance Test	15	1	0	0	0%
UX Test	15	1	0	0	0%
Write-Up	16	3	0	0	0%
Demo Preparation	17	2	0	0	0%

9. Ethical considerations

Privacy and Data Protection

The most played song is accessed by the app in order to provide a good recommendation for the user. This is the only data that the app holds about the user. The use of this data is strictly used to recommend music.

Other trademarks

Last.fm free web service provides links to buy songs from several brands, including Amazon.com, Ebay and iTunes. There is no commercial relation between the Recommusic app and these brands. Amazon, Ebay, iTunes and Last.fm are trademarks registered in your respective countries.

10. Resources

Last.fm API

The last.fm is essential for the success of this project. It provides ways of accessing the last.fm recommendation service. This resource is accessed through a free valid key and according with the instructions to do so. (http://www.last.fm/api/mobileauth).

The api is available at http://www.last.fm/api and provides data interchange in JSON and XML.

The terms of service must be applied (http://www.last.fm/api/tos).

Creately.com

This is a free online tool that provides a rich platform for diagramming and collaboration across the web. The Use case on the section 4 and the overview diagram on section 5 were designed using the free version of this tool. More information about it can be found at https://creately.com/.

Wireframe

Wireframe is a tool for sketching prototypes of user interfaces. It provides a quick and straightforward way of sketch some app screens. The sketch presented on the section 7 was designed using the wireframe web utility. The suite is available at https://wireframe.cc/#

Id3 Java Library

Id3 is a standard for holding information about songs in .mp3 files. There are several libraries in the market available for manipulate this type of specification. A great selection of implementations can be found at http://id3.org/Implementations.

As we work with Java platform, we may decide between one of the implementations available.

11. References

[1] Java ID3 Tag Library. [online]. Available at: http://javamusictag.sourceforge.net/. [Accessed: 21 Jan 2014]

[2] Logan, B. (2004, October). Music Recommendation from Song Sets. In ISMIR.

[3] Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2006, October). Hybrid Collaborative and Content-based Music Recommendation Using Probabilistic Model with Latent User Preferences. In ISMIR (Vol. 6, p. 7th).

[4] Chen, H. C., & Chen, A. L. (2001, October). A music recommendation system based on music data grouping and user interests. In CIKM (Vol. 1, pp. 231-238).

[5] Android Developers Blog. Android's HTTP Clients. [online]. Available at: http://android-developers.blogspot.co.uk/2011/09/androids-http-clients.html. [Accessed: 22 Jan 2014]

[6] Apple. IOS Design Resources. [online]. Available at: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html#//apple_ref/doc/uid/TP40006556. [Accessed: 23rd Jan 2014]

[7] AndroidPatterns. [online]. Available at: http://www.androidpatterns.com/. [Accessed: 23rd Jan 2014]