Personal information

Nickname: amCap1712
IRC nick: amCap1712
Email: amcap1712@gmail.com
GitHub: https://github.com/amCap1712
I am a former Google Code-In participant. I was selected as a finalist by MetaBrainz Foundation in Google Code-In 2017. At that time, my nick was kartikohri13 and email was kartikohri13@gmail.com .

Bringing back MusicBrainz for Android

Background and Overview

MusicBrainz for Android was first created in 2010-11 as a part of GSoC. Since 2015, no updates have been made to the project. The app is currently broken as the it does not adhere to android ecosystem. I feel a mobile app is a necessity for an organization like MusicBrainz. A mobile app will help increase user engagement and open up MusicBrainz to a plethora of new users.

Proposed Use Cases

1. Search MB entities and view their details.
2. View and search collections through text and barcodes.
3. Add/Edit tags, ratings and collections. (for registered users)
4. Tag music using MusicBrainz. (A basic implementation of Picard)

Technical Overview

Here is an exhaustive list of tasks, I intend to undertake as a part of GSoC. I have also mentioned an estimate of the time required to do each of them.

Step 1: Migrating the existing codebase

The existing MusicBrainz app provides the facility to search for an artist/release in the database, to view collections if a user is authenticated and donate money to MetaBrainz foundation via PayPal. Sadly, many of these features are broken as the app is outdated.The first task will be to re-work the app so that the all features currently offered work again.

- Currently, the project uses **Maven** as build automation tool. But Android has long moved to Gradle. All dev tools are only compatible with **Gradle**. So the first task is to port the project to Gradle. Android support libraries have now now migrated to *AndroidX*, our app will also do so.

The external dependencies of the app are **Novoda ImageLoader, ViewPagerIndicator, Apache HttpComponents & Paypal**. All of these need to be replaced due to one-another reason.

- *Novoda ImageLoader* is no longer in development. It will replace with *Glide*.

- *Apache HttpComponents* is obsolete and all its functionalities are offered by native OS APIs. The current networking API will be modified to use native OS support instead of this library.

- Same goes for *ViewPagerIndicator*, a GUI library which be replace by Android Arch Components.

- *Paypal Android SDK* (infact all Paypal libraries) are now deprecated. The recommended replacement is *BrainTree* but it requires to configure a custom server in conjunction with the app. For the time being, existing Paypal integration will be removed (currently it is not in working state) and the Donate page will redirect to MetaBrainz Website.

- The residual and unneeded files will be removed. The existing files will be reorganized to ensure a clean architecture.

- With Android 23, the permissions API underwent a major change. To become compatible with newer Android versions (Marshallow and ahead), the way the app handles permissions needs to be changed.

- Coming to the Login feature, I intend to add OAuth2 functionality. Users will be able to use MusicBrainz credentials to allow the app to access their data just like other MetaBrainz projects. The user will be prompted to Login in the browser and allow access. Irrespective of the user's decision, the user will be redirected to the app and greeted with an appropriate message.

With this Step 1 is over, we now have an app which on the outside resembles the existing version but possesses a remarkable different and cleaner core.

Step 2: Adding new features (Backend)

Now that our app is working, we will add new features to it except **Tagging facility**. This will be done in 2 steps. In the first one, we will add backend code for them and in the second one, the necessary GUI will be added.

- The app uses a custom java api to access MusicBrainz database. Currently, the app only searches for artists/releases and but no other entities available in the website search dialog. So the API needs to be extended to search/lookup all entities in MusicBrainz. Also, the parser will be migrated from XML to JSON. I will attempt to reduce bandwidth usage and number of requests.

- Background tasks and Network request are currently handled with *Async Tasks and Loaders* but Loaders were deprecated the last year. To become, future-proof I will attempt to replace Loaders with Android Architecture Components. Therefore, the entire way concurrency is handled by the app will be modified. This will improve performance and stability.

- Functionality to search releases through Barcodes will be added using *Zxing* Library and to allow logged in users to add/edit tags and ratings. This will be added on the top of login framework already set up earlier.

Now getting as to what will be available for end use as a result of the above modifications. The users will be able to search these entities *Artist, Release, Recording, Release Group, Instrument & Event*. The users will be able to navigate between entities. For example: - If a release is clicked on an artist's page, it will open up. There will be recent search suggestions as well. All of these entities will have individual info pages. The users will be able to search releases using barcodes as well. Logged In users will be able to view their collections and add/edit tags & ratings.

With this, I will exhaust a month of GSoC.

Step 3: Add new features (Frontend)

We improved the framework of the back. Now, it is time to repaint the app.

- The GUI of the app will be reworked to match the current MusicBrainz theme. Extra buttons and screens where ever needed will be added. I will prepare GUI mock ups beforehand for getting feedback from the community.

- I will add support for international languages in the app. As suggested I will use *Transifex* for translations. I intend to provide support for all languages currently supported by MusicBrainz. Transifex and Android provide out of the box support for i18n so the task will be not lengthy.

Visuals of the current functioning of the application.

1. Search Interface (https://www.youtube.com/watch?v=L_Mz_ZGhVZ4)
2. Donate Interface (https://www.youtube.com/watch?v=MEwOjmiZvUI)
3. Login Interface (https://www.youtube.com/watch?v=IwnSdlLDesc)
4. Artist Details (https://www.youtube.com/watch?v=9XN_4IiQEbg)

We thus come to end of another two weeks.

Step 4: Adding the tagging facility

Till now, we have done some minor upgrades to the app's usability and brought substantial changes to its interface. Now its time do a major upgrade. Finally, we implement a basic Picard for our app. (*Yay!! I'm really excited.*)
To begin with, as per my understanding the feature will have 2 dependencies JAudioTagger & FpCalc-Android. JAudioTagger will be used to manipulate metadata of the audio file ie. reading and writing. FpCalc Android provides an Android compatible Java wrapper for Chromaprint. The implementation will match desktop Picard's implementation. In case of doubts, I'll consult Picard developers. But as of now, the decided version is as follows:

- There will be 2 options to match metadata to tracks namely *LookUp* and *AcoustID Fingerprinting*. **As part of the GSoC project, the final app will facilitate to tag only one track at a time.**

- Coming to LookUp, first the metadata of the track will be read. The metadata fields, namely, *Title, Album, Artist, Track Number & Total Tracks* will be included in the search query to MusicBrainz Webservice. Each item in the search result, will be processed to calculate a *Similarity* score using the *Weighted Levenshtein Distance*. The best match will be displayed to the user and there will be an option to view all results and select an alternative as well. Then, upon clicking the save button the metadata will be written to the audio file.
- Now onto the Fingerprinting part, the AcoustID fingerprint will be calculated using the FpCalc Android wrapper through Chromaprint algorithm. The audio fingerprint will be sent to the AcoustID webservice. A lookup request will be performed on MBIDs of the recordings returned by AcoustID. The result will be displayed to the user and the user can save the metadata.

I believe this is a time intensive feature and will require around a 30-40 to be finished. I plan to dedicate **14 days to each option.** The remaining **10-12 days** will be used to build the supporting UI and act as an internal buffer period.

Note: - The reason behind the internal buffer period is that the original libraries on which the feature depends are not completely compatible with Android. So, I might end up

using unofficial forks of the libraries mentioned. I will experiment before to make sure if the selected libraries work fine but just in case, it dosen't 😉.

Step 5: Wrapping Up

It is time to wind up the work of 2 months. The app will be tested intensively by the community and bug fixes will be made. I will be documenting the code side by side but will review it during this period for clarity.
I estimate another one and half week to pass by.

Step 6: Stretch Goals (Optional)

Until now, all the listed tasks were a necessity, now we come to those ones which will be completed only if time permits.

- To improve performance of the app, a local cache database using Room will be implemented to store latest query results.
- Add functionality to search using voice.
- To create a Google assistant action for the app. With this, the app's features can be directly accessed using Google Assistant.

Current Progress

Since, I have been working on the app independently of GSoC, a part of the work is completed. The test version of the app is accessible at https://play.google.com/apps/testing/org.metabrainz.android . **A changelog for the same is available at MusicBrainz for Android Changelog .**

- The migration of the app to Gradle file system and AndroidX is complete.
- The obsolete and deprecated dependencies are completely removed. They are being replaced as the app is being refactored.
- Search results page is complete. Recent search suggestions are in place.
- The details page of Artist is complete. For other entities, it is in the works.
- Basic OAuth framework is set up. Although, a lot of work is to be done on it.

There are a few other community posts on specific topics relating to this proposal. They are linked below.

1) Android app architecture (Discuss about the architecture to follow in the Android)
2) Suggestions for MusicBrainz for Android (Advice from the community regarding the UI)

Weekly Schedule

An estimated schedule is provided alongside the detailed description of the tasks to be undertaken. Here I provide, a summary of it.

- Week 1: Complete search pages. Start work on entity details info pages.
- Week 2: Complete details pages and add robust navigation between them.
- Week 3: Complete login flow with OAuth. Start adding option to view collections and ratings.
- Week 4: Work on editing collections and ratings.
- Week 5: Add barcode search and submission.
- Week 6: Implement new GUI.
- Week 7: Add additional languages support.
- Week 7 - 9: Begin adding tagging facility. Implement tagging using lookup of existing metadata.
- Week 10-11: Implement tagging using audio fingerprinting and AcoustID.
- Week 12-13: Bug fixes, testing and completing documentation. Buffer Period.

I intend to begin coding right from start of community bonding period.

Detailed information about yourself

- Tell us about the computer(s) you have available for working on your SoC project!
  I have a HP laptop with 8GB RAM, 1TB Hard disk and 2GB graphics card. The main tool required for the task is Android Studio which runs smoothly on this system.

- When did you first start programming?
  I started programming when I was in Grade VI as a part of school curriculum.

- What type of music do you listen to? (Please list a series of MBIDs as examples.)
  I love listening to Punjabi Music. Some of my favorite artists are Amrinder Gill, Sidhu Moose Wala & Guru Randhawa.

- What aspects of the project you're applying for (e.g., MusicBrainz, AcousticBrainz, etc.) interest you the most?
  The prospects of cleaning up my entire music library on my mobile automatically sounds just great. Also I prefer mobile apps to websites, so I am making work easier for me with project.

- Have you ever used MusicBrainz to tag your files?
  Yes, I use it regularly say on a weekly basis or so.

- Have you contributed to other Open Source projects? If so, which projects and can we see some of your code?
  I have contributed to a couple of other open source projects which can be seen at my Github Profile. I have been the co-moderator of an app built to raise awareness on women safety.

- What sorts of programming projects have you done on your own time?
  I have done many programming projects. Many of them are personal ones. I usually do Android projects but I have worked on a few python projects as well. The apps I have built include a news app, a notice delivery app for my school and some fun personal projects.

- How much time do you have available, and how would you plan to use it?
  I will be working full time on this project in GSoC.

- Do you plan to have a job or study during the summer in conjunction with Summer of Code?
  No except for the last fortnight, I don't have any such plans. In the last fortnight, I will attend orientation classes at my college.