Audio Ember

ROBERT GONZALEZ, KEVIN WILLIAMS AND MICHAEL WOODHAM

08/02/2017

Application Details

The goal of the Application was to provide a service which Billboard is not currently providing: data amalgamation and illustration of songs as they progress through charts, as well as a definitive answer to which songs have been the best in certain categories. This was accomplished by assigning each song a point value. Points are the accumulation of ranks brought about by our formula which determines the total points that the song has generated across all weeks up to the current week. Our formula is the following, 101 - rank of song on chart. For example, a song which was ranked 22 on the Hot-100 chart will have the value of 79 in the hot 100 chart. As of current our app implements the Dates from January 1, 2016 to July 22, 2017. Our target audience is probably going to be millennials given that our target data is such a small chunk of years, but in future updates our target audience will be member of all age brackets who love music.

Related Apps vs Our Application

As far as how to format this section we considered a few different implementations. Either to go over a few different apps on the Google-Play store and go over each of their pit falls, or to sum them all up into one category. In an attempt to void redundancies as all for their pit-falls are the same, we have opted to go for a summary. We do concede that as of current standing other applications do look better than ours (Billboard Radio, Spotify, Mueso) all of which provide stream and listening options, but as far as data goes we can not find another application which succeeds in this area. As it currently stands, chart apps appear to care about this weeks charts and standings in the music industry, but for previous charts, they all fail to notice them. We believe that the reason for this is that there is no perceived demand for a publicly accessible database reporting chart progression of music. We beg to differ and contract this notion as we find this field fascinating and do believe that others feel the same. We on the other hand do not believe that billboard would wish for the public to see how songs do fair on charts overtime as it does reveal some intricacies and

inconsistencies within the industry, but that is not for us to decide, but rather for the viewer. The data does tend to speak for itself.

Development Time line

Original Concept Design

As can be seen in the two figures below, our original design was to have the user search for Songs or Artists in the Database or have a guided menu by genre, month, or year. This was scraped for the reasons of time and simplicity. We felt that this page would overwhelm users with too much information accessible at the start.

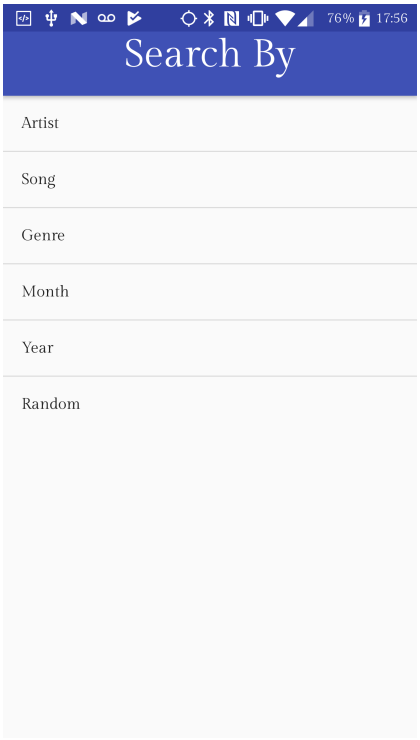

Figure 1: Month Selector

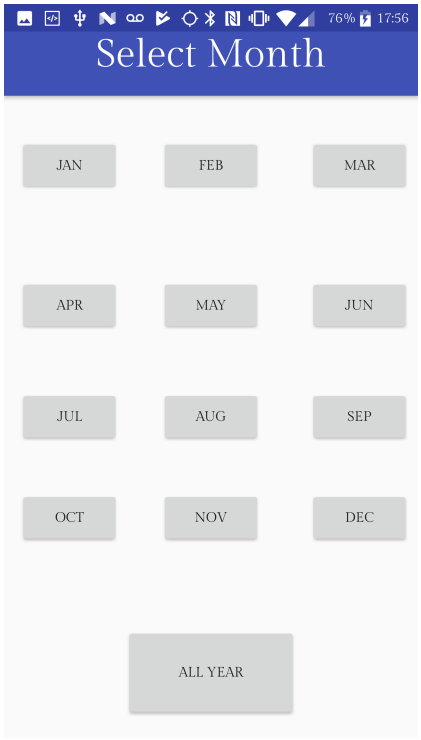

Figure 2: Search Bar

Here you can see our original artist page. This was later revised to match the color format of our project, but the general idea did remain the same.

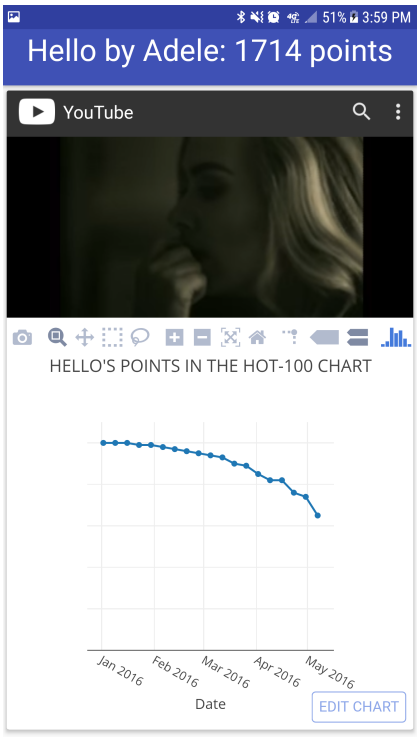

Figure 3: Original Song Page

Not shown is the second revision before we settled on our final design. In this revision There was a bottom bar similar to our final design. This bar was dynamic and would change given the user's position in the application either having 3, 4, or just two options. This design was scrapped for two reasons. Firstly, our artist integration was unable to be finalized in the project so that eliminated the need for a lot of our options. Secondly, with artist integration gone, most of our implementation was deemed to be redundant, resulting in our final design being built.

The third and final revision is shown below. We opted for a simple design of three buttons. Gone is the search function, gone is the date function, and what is left is a simple list from the top songs in a specific genre as determined by the user in the second option of the bottom bar. The third option is the video and graph generation for the specific song. While it is a downgrade from our original concept, we are happy with the results, and will be implementing the final product in future generations.

Implementation Details

Billboard API

The Billboard API was rather frustrating to use at times. Originally we wanted to have a Rap genre within our database, but for some odd reason date incrementation would not work within the Rap genre, so that was scrapped. Dates in general were rather difficult to work with. Because the API takes in dates as strings rather than as DateTime objects there was frequent conversions that had to take place between the API and our database itself. Moreover, the API came packaged with a .next function which was designed to fetch the next week's chart each time it was iterated over. Unfortunately this functionality proved to not be useful as breaking was consistent. We opted for timedelta objects for week incrementation in a substitute. Moreover, if the chart did not exist on the current date, the API would return a null list. However, the API does not fetch a chart.date from a null list, which lead to frequent breaking and patching within our dbinit file.

SQLAlchemy

In order to effectively use the Billboard API we opted to back end our initial database with python and an init script. SQLAlchemy is a great database to use for storing information when using python because it allows for SQL queries to be returned in a Python-like structure which make for an easier implementation and integration into our project overall. SQLAlchemy's documentation, while extensive, did not quite translate well to our project. Because of the environment that they placed their documentation in, we were only able to gain a basic understanding of how the system works. Often, we found ourself reverting to stack overflow posts whenever we found an issue that we needed to solve. The Flask Mega Tutorial by Miguel Grinberg proved to be vital in getting the setup of our database established, and allowed for us to get the ball rolling. While it was nice dealing with Python based objects, we have had to recompile our database on multiple occasions due to errors within the base models, and SQLAlchemy's refusal to support migration out of the box. Additionally, rather than directing to a Firebase within our python script for the initialization, we needed a way to store information to later query. The SQLAlchemy structure allowed us to have this delay ability.

FireBase

Rob Can Put Stuff Here

Plotly

Plotly was used for the graphing interface of our website. When we first acquired this service in 2016 we were promised by the company that we would

be allowed to make 10,000 API calls a day. As it currently stands, our queries have been cut to only one-tenth of our original allotment. While this did hinder our progress, it did not stop us all-together. That being said, in future updates we do doubt that Plotly will continue to be the graphing option of choice. The reason for its use is to provide off application data storage and image options to not only increase efficiency but also to not clog the app with necessary storage.

JSoup

JSoup was used for YouTube URL scraping. As the YouTube API proved to be cumbersome to use, we opted for which we neglected in choosing the Billboard API as mentioned above. JSoup allowed for our songs to be dynamically linked upon song-request from our Application. This was not placed into our database due to the volatility of YouTube videos as well as the lack of need, as nothing needs to be rendered unlike plotly.

Andorid Specific Implementation Designs

ListView

ListViews are being used to allow for simple naviagtion in the lists for top-songs, and genres respectively. Clickable implementation was added for fragment transitions.

WebViews

The WebView is using an embed technique rather than a straight URL to load data. This allows for the webview to be more or less static and makes the scroll of the webpage more manageable for the user rther than traditional WebViews, resulting in the webview not being scrolled, but rather the scrollview that it is encapsulated within. The WebView is used for the youtube embed links and the graphs. They are dynamically added to the fragment rather than statically as anywhere between 2-8 webviews can be used on one song-graph page depending on the individual song and the number of charts that it made it on.

BottomNaviagtionView

The bottom naviagtion view is used for the menu layouts to allow for simple navigation bewtween fragments in the main layout.

Fragments

In order to simplify the UI and maintain consistency fragments are used in the main layout rather than individual activities.

Future Development

Artist Integration

The original concept of the app was to provide not only a top-song list by genre, but also a top-artist list. Due to time constraints as well as FireBase implementation difficulties we only have a top-song data structure enabled. That being said, top-artist structures are not far away. We have the information available and formatted within our SQLAlchemy database as it stands, so the update script can come in the next patch.

Time & Dates

In a future instantiation of the app we would like to have time capsule information available: To be able to see who dominated the 90s, 2000s, 80s, or 60s on the charts. As billboard first started producing charts in 1953, we believe that generational information would be fascinating to see. We also want to make the slider dynamic so that the user could determine where they want to see chart domination times.

Search Terms

We want to be able to implement a search bar for either Artist or Songs. Again, an easy fix which could be implemented in the next update.

Firebase Update Scripts

ROB's Information Goes HERE

References

[1] www.billboard.com

[2] www.github.com/guoguo12/billboard-charts

[3] https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

[4] https://firebase.google.com/

[5] https://plot.ly/python/axes/