CS 121 - MOBILE APPLICATIONS

CRATE DIGGER

TINDER FOR MUSIC RECOMMENDATIONS

By Jack Bauman, Carly, Young Choe, and Ismael

Introduction

This mobile application is a system for music lovers to quickly and easily discover new music. Alike a dating app where the user can swipe either right or left according to their preferences, Crate Digger allow users the freedom to discover new music at ease in this similar fashion. The app will present a constant stream of music recommendations, presented with album artwork, information, and a snippet of the song, and our algorithm will obtain further songs based on related artists of the songs users have liked. With the Spotify Android SDK, calls to the powerful Spotify API provide all the content we will pull from, and we will also provide an easy avenue for users to save the songs they are enjoy to their Spotify libraries.

Purpose

There are millions upon millions of songs and artists accessible online, including plenty of amazing tracks that lie just outside our own realm of musical knowledge. Our app aims to get users out of a musical rut and aid them in finding a constant stream of new music and rarities based on their relationships with music they already like.

Main Features

- Crate Digger will provide a constant stream of music recommendations to a user along with an easy and intuitive way to rate the music.
- Users can easily view previously liked songs, save them to your Spotify library, and share them with friends.
- Users can view statistics about your music taste, as well as tweak the algorithm directly in order to make your recommendations more relevant.

App Flow

The app will consist of 3 main views.

1.) *Main Music View* - This will be the primary view of the app, where the user is presented with music in a simple and visually pleasing way, and will be able to give feedback on whether they like it. The app presents one song at a time to the user, complete with the track's album artwork, information (artist, name, year), and plays a 30-second sample snippet to the user. The view consists of physical "like" and "dislike" buttons, as well as swiping "left" or "right." Liked songs are added to the user's saved

music list, and its related artists and songs are added to the pool of next presentable songs.

2.) *Liked Music View* - Accessible from the top navigation bar of the Main View, this will present a table view of all the songs that the user has liked, along with small album art images and basic info. Users have the option to "Open/Save the song in their Spotify library", "Share the song with a friend" over a messaging app, or "Delete" the song from their saved list.

3.) *Stats and Setting View* - Also accessible from the Main View navigation bar, this view will show genre statistics for "Most Liked" and "Least Liked" genres over time. There will also be a table view listing genres of songs, with the option to "Show me more" of said genre. The algorithm will add more examples of a selected genre to pool of next songs.

In terms of how the app's algorithm will recommend songs:

- Our app will store and handle songs by their unique Spotify URI provided by the Spotify SDK framework. Calls to the Spotify SDK using these URIs allow us to access a track's metadata and 30-second sample, as well as easily convert the URI to a link to the track in the Spotify app. There will be no need for our users to log in with a Spotify account, as the Spotify client ID for Crate Digger allows us to access this data.
- Songs will be presented somewhat randomly from a pool of music generated by a user's likes and dislikes. A user liking a song will trigger an API call to get the song's related artists' URIs to the pool of upcoming recommendations.
- A user tapping "Show me More" of a particular genre will also add more of these songs to the pool. These are accessible through Spotify API calls. This will fix issues if there are not enough enjoyable songs being presented to a user through our algorithm.
- We will store URIs of songs we have shown in a hash table and will verify that a user has not already been shown the same song before presenting it to them.

High Level Technical Architecture

- A hash table to store songs (by Spotify URI) we have already shown to users.
- A dictionary to store liked songs and their metadata.
- Spotify SDK framework embedded.

Libraries

We will be using Android Studio for the app's development. Crate Digger will integrate Spotify's open source, free-to-use Android Software Development Kit and obtain music metadata through calls to the Spotify API. To call the Spotify API from our app, we will use the Spotify Remote Library SDK.