sounDJam

Sanat Deshpande, Lionel Eisenberg, Jack Karyo,
Alex Knowlton, Jonathan Wang

Overview

How many times have you and your friends fought over the aux cord when you’re all hanging out or hosting a party?

sounDJam involves everyone in the process of creating the best possible playlist. Whether you enjoy Nickelback or Taylor Swift, you have to power to influence everyone's listening experience. While anyone can share their song, you are still the DJ of your own party, and can customize and cater your queue to every setting.

Goals

Create

- A user-friendly, robust mobile app
- Implement our core features

Learn

- Flutter & Dart
- Process of building a mobile app from scratch
- Industry standard development practices

Initial Vision

We wanted to create an app for friends to queue up their favorite songs without having to pass around the aux cord. A sketch of our initial vision was as follows:

- One user with their device playing music (DJ) who controls each session and can blacklist genres that he/she does not enjoy, and set other restrictions.
- Regular users can downvote (boo) songs and add songs of their choice to the queue.
- Queue will have an algorithm that automatically reorders songs based on the DJ’s settings and users’ feedback.

Original Feature List

DJ

- play next
- dj queue
- time limit: give a time-limit for all songs
- amount of songs allowed to add: amount of songs each user is allowed to add in an hour
- blacklist: create a blacklist of songs, artists or genres
- stylize: choose what genres you like/don't like
- ability to end the party
- Oauth

Implented
Not implemented due to Spotify
Not implemented

Users

- add songs
- join session
- boo songs
- Identifier

General

- Input a code to join (we moved to QR codes instead)
- Code lasts 24 hours or until party ends.
- Sharing code through social media or sms/imessage
- Curated Queue using our own algorithm

Extended Features

- add song to personal playlist
- QR code to join party
- QR code can be displayed on DJ phone or chromecast
- Equalizer
- closing time
- anarchy mode

Backend Architecture

- Java, Maven, Javalin
- MongoDB
- Heroku
- JUnit, Mockito, Power Mock
- Travis CI, Postman
- Spotify API

Queue Algorithm

1. Add Song to end of queue.
2. Score each Song with how well they fit the requested settings.
3. Add each Song to a graph where a node represents a Song
4. Create a complete graph by creating edges between each Song, and weighing the edge by how well two Songs would fit consecutively (2nd scoring function).
5. Start at the currently playing Song and traverse the graph:
   a. Adjust the weight of each edge by the score of the destination node.
   b. Implement nearest neighbour algorithm, go to the node with the “closest” next node that has not already been visited.
   c. Stop when all nodes have been visited.
6. Traversal order is order in which to play Songs.

Very similar to TSP but more efficient for big queue size

Algorithm Scoring (Normalizing 5 Attributes)

Song to Song Score

- Danceability, Happiness, Energy
  - Spotify API returns these scores from 0.0 to 1.0
- Song Duration and Tempo
  - We normalize these to 0.0 to 1.0, using the larger song’s attribute as a baseline

Algorithm Scoring (Normalizing 7 Attributes)

Song to DJ Setting score

- Danceability, Happiness, Energy
  - Spotify API returns these scores from 0.0 to 1.0
- Tempo and Duration
  - Treat setting bounds as plus or minus 2 sigma of the middle value, and normalize to a z-score
  - Taylor Series Approximation of Gaussian CDF to compute score from 0.0 to 1.0
    - No closed form of the integral of a Gaussian probability density function
- Boo Score
  - Total Boos / Total Potential Boos
- Time Since Requested
  - Normalizes based on longest-waiting song and shortest-waiting song

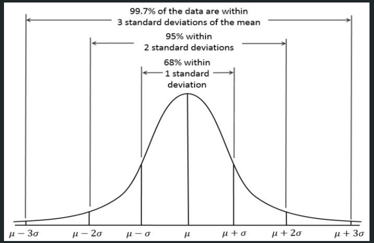


Image Source: https://math.berkeley.edu/~scanlon/m16bs04/ln/16blec31ns/index.html
Image Source: https://en.wikipedia.org/wiki/Normal_distribution

$$
\begin{aligned}
G(x) &= \int \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}x^2} dx \\
&= \frac{1}{\sqrt{2\pi}} \int \sum_{n=0}^{\infty} \frac{(-1)^n}{n!2^n} x^{2n} dx \\
&= \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n}{n!2^n(2n+1)} x^{2n+1} \\
&= \frac{1}{\sqrt{2\pi}} \left(x - \frac{1}{6}x^3 + \frac{1}{40}x^5 - \frac{1}{336}x^7 + \cdots\right)
\end{aligned}
$$

Algorithm Scoring (Final Score)

Softmax Scores and Compute Cross Entropy For Both Scores

- $$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$
- $$H(p,q) = -\sum_{x} p(x)\,\log q(x)$$

Combing Scores

- α * Song-to-Song + β * Song-to-Settings
- Weighted through experimentation

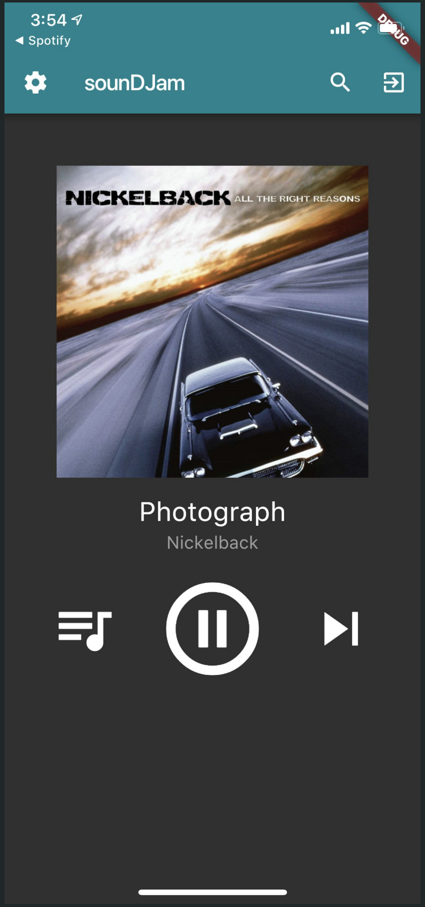
Image Source: https://en.wikipedia.org/wiki/Cross_entropy
Image Source: https://en.wikipedia.org/wiki/Softmax_function

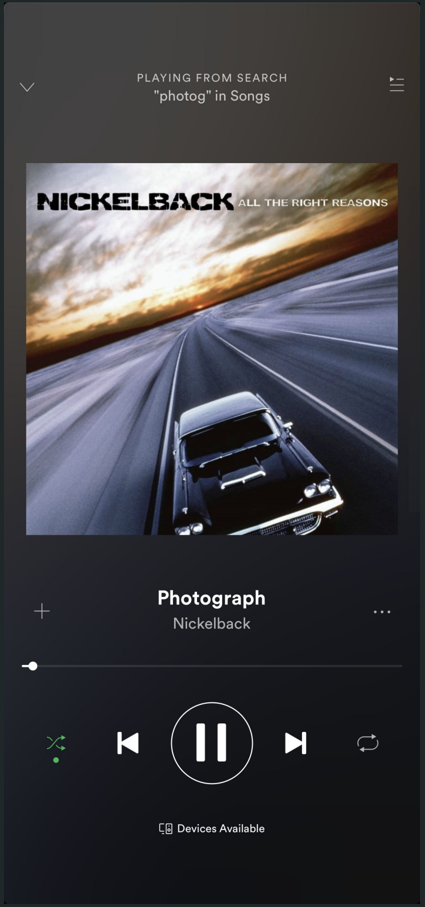
Frontend Architecture

- Flutter, written in Dart
    - Ports easily to both iOS and Android devices
    - We did have to write some platform-specific code (Objective-C, Java) in order to get the player to function properly
- Widget testing to accompany each view
    - Build widgets (components) on the fly and dynamically test layout and interactivity

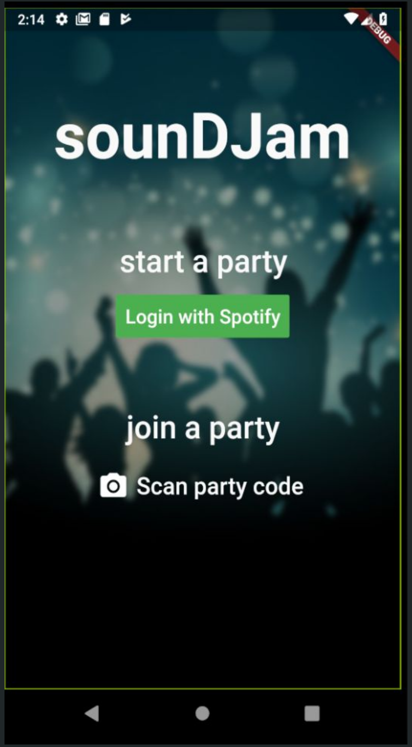
Design

We wanted to make our app familiar to Spotify users. As a result, the layout of our user interface shares similarities with Spotify.

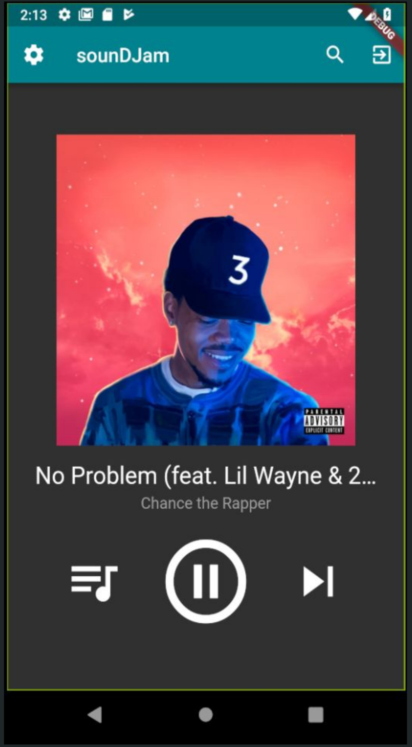




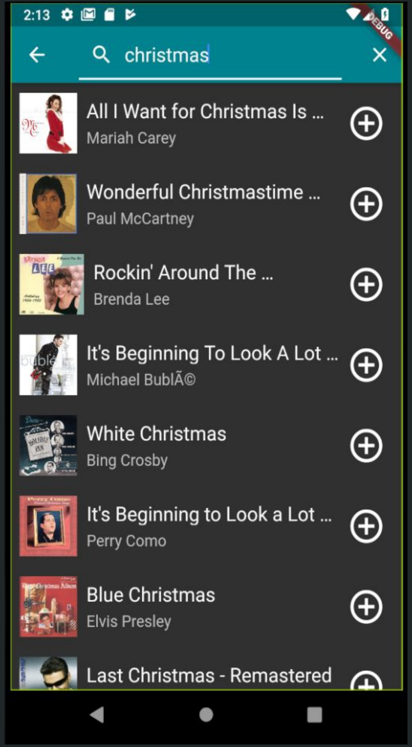
Some UI Screenshots



Landing Page

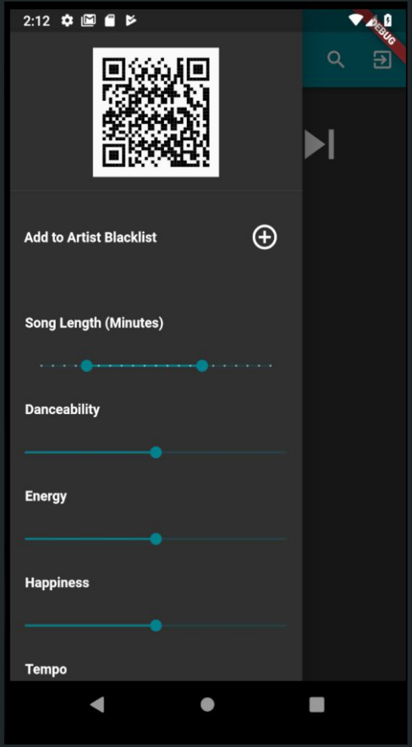


Player



All I Want for Christmas Is ... — Mariah Carey
Wonderful Christmastime ... — Paul McCartney
Rockin' Around The ... — Brenda Lee
It's Beginning To Look A Lot ... — Michael BublÃ©
White Christmas — Bing Crosby
It's Beginning to Look a Lot ... — Perry Como
Blue Christmas — Elvis Presley
Last Christmas - Remastered

Search



Settings

Demo

sounDJam

Next Steps - “Iteration 7”

- More Integration and/or E2E tests to increase code-coverage.
- Integrate a Complete Hamiltonian Path algorithm for smaller sized queues to curate more optimally.
  - Held-Karp Algorithm: $O(n^2 2^n)$
- Flesh out social media features (sharing codes, sharing favorites etc.)
- Add more optimization and personalization options to the DJ
  - Seasonal settings like a christmas ambiance settings option.
- Go more in depth in all the features offered by flutter.
- Scrubbing
- Refactoring
- Handle more native operations regarding application state
- Tweak hyperparameters for Graph Algorithm