

Pradyumna Mukunda

Fall 2019 – Sprint 5 Individual Report

CS 8803 Mobile Application and Services

Georgia Tech, Atlanta, Georgia, USA

Part I

1. Team Name and Members

Team Name:

Beat Harmony

Project Name:

Harmonize

Team Members:

- Ankit Verma (averma46@gatech.edu)
- Rishma Mendhekar rmendhekar3@gatech.edu
- Christian Graham cgraham47@gatech.edu
- Justin Higgins jhiggins@gatech.edu
- Pradyumna Mukunda pmukunda3@gatech.edu

1. Updated Project Overview

Problem Statement

Existing music platforms create echo chambers based on the music you already listen to.

- Recommendation algorithms have a hard time breaking out of the information they're fed.
- Music-heads are not connected to curators who share in their unique tastes

Music streaming services like Spotify, YouTube Music, etc. collect user data in order to recommend new music. This causes echo chambers of the same genres of music being recommended back to you.

Services such as Spotify, try to break out of this with curated playlists tailored to your mood or activity, but in practice you're still likely to skip a song that's not in a genre you enjoy. Our team is working to recreate the organic experience of hearing a song in your friends car that you love, but existing services wouldn't recommend to you.

Pitch

Harmonize is a collaborative music platform centered around finding, creating, and listening to music playlists together. Harmonize is like a music curation service built to help users find new music through social connections. Music streaming services like Spotify, YouTube Music, etc. use algorithms to recommend new music which can cause echo chambers where the same style of music is recommended to you over and over again. To break out of this we're redefining playlists as a collaborative effort. Up until this point music playlists have been a historically solo activity where you create and listen to them alone. Harmonize is turning playlists into socially driven music hubs designed to recreate the feeling of hopping into your friends car and hearing your new favorite song for the first time

Current Design Pivot

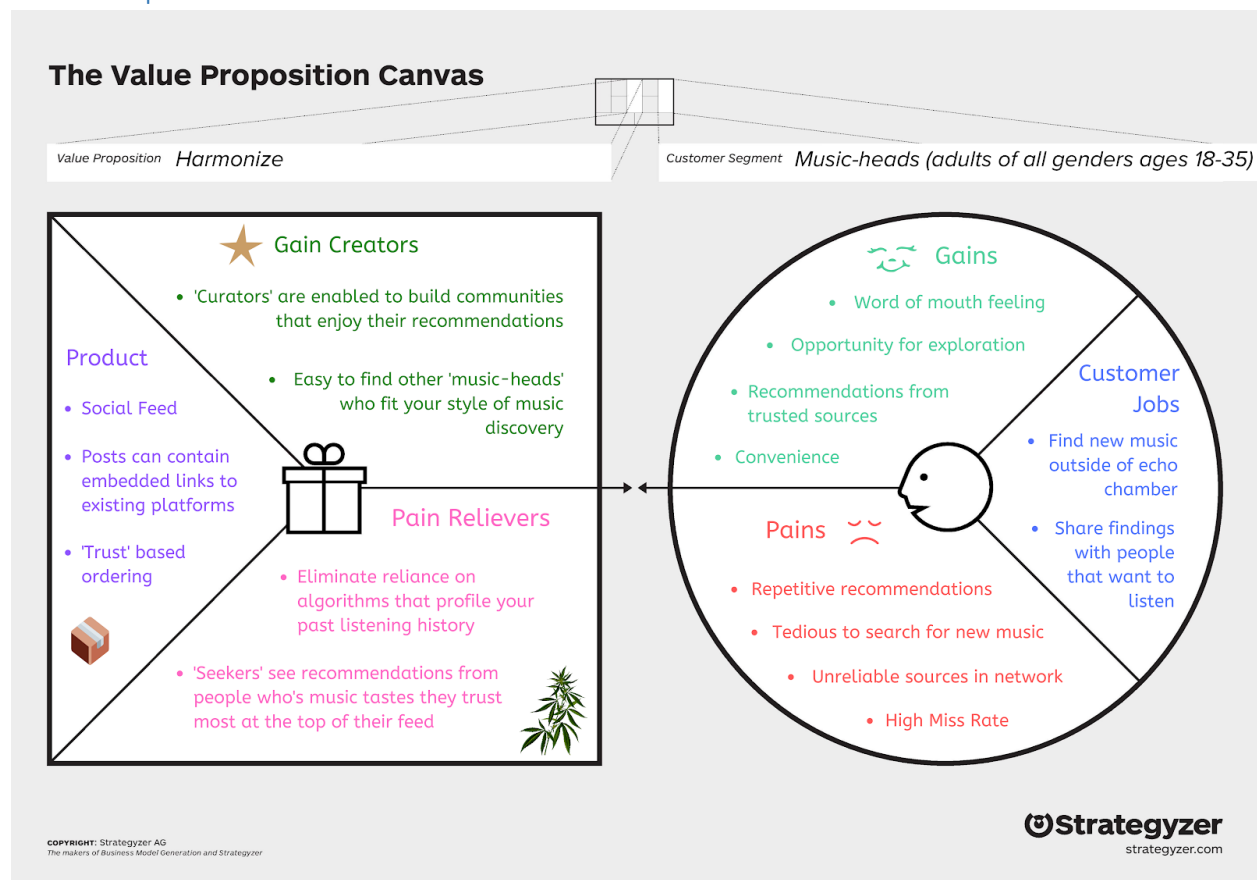
Pivot: Redefine playlists as a living, communal affair.

- Every playlist is a living space with an owner, live chat, and an ever changing playlist of songs.








- Users can chat, link to other playlists, recommend songs to add/remove from the playlist, and more.
- With a focus on **creating** playlists, **finding** playlists, and **listening** to playlists.

Each playlist is a social community where users can collaboratively add/remove songs, interact through a live comments section, and link to other playlists. Harmonize is designed to work with, not against, the music services you already love and use. A Harmonize playlist can contain songs from multiple music streaming platforms. In a single playlist a user can switch from Spotify to SoundCloud and never skip a beat. Whether you're looking for socially driven music discovery or a place for your friends to collaborate on a kickass party playlist, Harmonize makes it easy to combine all your favorite music streaming services into one community driven place

Value Proposition



The Business Model Canvas

Key Partners  <ul style="list-style-type: none"> Music Production Communities & Independent Labels → Incentive for them to make our new platform more active for their promotional purposes Spotify Playlist Curators → Will likely have a desire to continue spreading their playlist tastes on to our new platform Streaming Services (Long Term Goal) → Streaming services have share buttons with options such as 'share on...'. Long term goal is for streaming services to enable 'share on BeatHarmony' 	Key Activities  <ul style="list-style-type: none"> Determine value-adding post ordering mechanism through unique database organization and query strategies Continue aggressively gathering feedback Build initial user-base through Key Partners and Channels 	Value Propositions  <p>Product:</p> <ul style="list-style-type: none"> → Social Feed → Posts can contain embedded links to existing platforms → 'Trust' based ordering <p>Pain Relievers:</p> <ul style="list-style-type: none"> → Eliminate reliance on algorithms that profile your past listening history → 'Seekers' first see the recommendations from people who's music tastes they trust most <p>Gain Creators:</p> <ul style="list-style-type: none"> → 'Curators' are enabled to build communities that enjoy their recommendations → Easy to find other 'music-heads' who fit your style of music discovery 	Customer Relationships  <ul style="list-style-type: none"> Listening to feedback from users, especially early-adopting music communities Ensuring that new echo-chambers are not being created Act as curators of curators, not curators of music 	Customer Segments  <p>"Music-Heads"</p> <ul style="list-style-type: none"> → want to find new music outside of their echo chambers → share new music with people that want to listen <p>Pains:</p> <ul style="list-style-type: none"> → Repetitive Recommendations → Tedious to search for new music → Unreliable sources in network → High Miss Rate <p>Gains:</p> <ul style="list-style-type: none"> → Word of mouth feeling → Opportunity for exploration → Recommendations from trusted sources → Convenience
Cost Structure  <ul style="list-style-type: none"> Server hosting Social media marketing targeting music communities Paid database services such as neo4j that can be leveraged for social feeds Paying the people working on BeatHarmony [longer-term] 		Revenue Streams  <ul style="list-style-type: none"> Unlimited all-access to BeatHarmony for new users for 30 days Patreon style subscription model for paid users [subscriptions to individuals who's recommendations you value] A portion of funds goes to BeatHarmony's bottom line, while the majority is distributed among curators and artists 		

 Strategyzer

3. Learnings from previous learning prototypes

Learning Prototype – Sprint 2

Goals of learning prototype in Sprint2 was

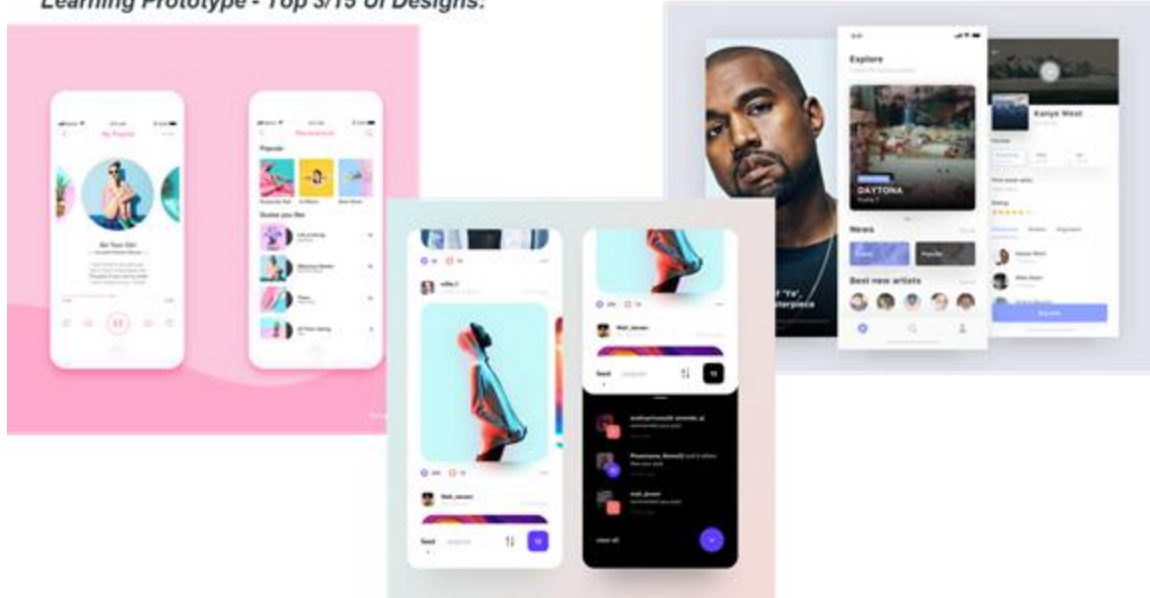
- Collect customer data referencing our overall product design.
- Collect customer data for a future interactive learning prototype.
- Narrow in on a UI design.
- Narrow in on a color palette or aesthetic style.

About 28 people were interviewed to collect their preference on 15 UI designs and 15 Color palettes.

Based on above survey we narrowed down to the top 3 UI design and color palette.

Top 3 UI designs and color palette selected from above survey is listed below

Learning Prototype - Top 3/15 UI Designs:



Learning Prototype – Sprint 3

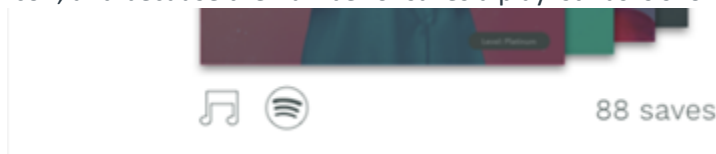
Sprint3 activities were based on following main activities

- Creating Basic Design document along with use case updates
- Creating server architecture
- Decide on technologies for each of the resources in the system

In addition, we created a clickable prototype of mockup screens / UI and planning to get user feedback.

Insights gained

1. All of the participants thought that the “like” button was the Apple Music logo and that it would export the playlist to Apple Music. This is because it is situated next to the export to Spotify icon, and because the number of saves a playlist has is shown on the opposite side of the post.



Design ideas to address this concern:

- Find a different icon for liking a post, because the music note is actually Apple Music’s logo.
 - Put the number of saves close to the “like” button
 - Change the wording of “saves” to “likes” so that language is consistent throughout the app.
2. All four participants thought that the live chat section was actually a comments section before they click on it. Three out of four participants also stated that they would prefer comments over a live chat; two preferred comments because it is easier to follow. With a live chat they felt like they would not be able to see or take part in discussions that happened some time ago, whereas with comment threads, discussions are preserved and easier to find and sort through.

Design ideas to address this concern:

- Switch the live chat section to an asynchronous comments section.

- If we continue using the live chat, the chat can split up into subchats/groups based on topic.
3. A few participants said that they would prefer being able to message any user directly from their profile because this would allow them to start conversations with interesting users immediately. At the moment, users can only message people that they have matched with the matching feature.

Design ideas to address this concern: Implement messaging functionality for every user regardless of whether they have matched or not. My main concern here is where users will be able to access these messages because the menu bar already has a number of icons, and the top right corner of the app holds the icon for creating playlists.

- Two users said that they would like to see peoples' top artists on their match profiles.
- Two users said that it was hard to differentiate between the unread and read states for messages and notifications.

Learning Prototype – Sprint4

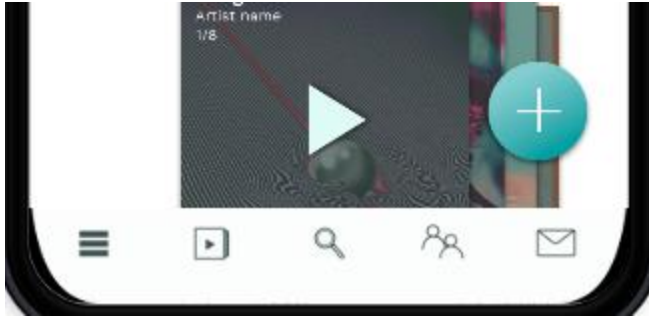
Design (Rishma)

During Sprint 5, there were two branches of design. One branch continued from Sprint 4's user feedback, while the other one was an MVP design that was developed after meetings with the rest of the team to determine our MVP for Sprint 5. During Sprint 4, Rishma ran a usability survey with 14 participants on the most recent InVision prototype. We received the following feedback which resulted in design changes-

77% of participants reported that contributing to a playlist was easy or very easy. However, 3 participants expected the + icon in the top left corner to let them contribute a playlist rather than create a new playlist. Four participants stated the playlist user should have more control over who adds songs and which songs are added.

Design pivot:

1. Privacy settings have been added to playlists. The screens can be viewed here by clicking "Add Contributors" - https://projects.invisionapp.com/share/C2UGJH2AN7H#/screens/387311462_Homepage
2. The "Create Playlist" button has been moved from the top corner and now floats at the bottom right of the screen. Also, it only appears on the Feed page to make its function clear. Although mimicking the floating functionality was not possible in InVision, the intent was for the blue circle to stay in the same position on screen while a user is scrolling through their feed.



a.

34% of participants reported that it was not easy to find their messages. 3 participants stated that they expected the messages to be in the bottom menu bar. 6 participants thought that the Matches icon in the menu bar would take them to the messages section. 11/14 participants were able to correctly identify the number of unread messages on the page.

Design pivot:

1. Change the matches icon to be more obvious

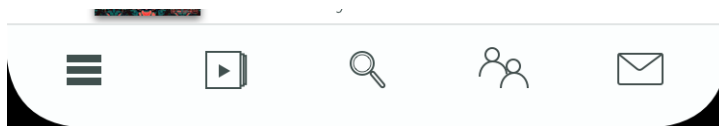


a.

2. Put messages in the menu bar

The application navigation was rearranged so that messaging is now in the menu bar and users can access their profile on the top right corner of the screen.

i. New menu:



ii. Old menu:



iii. New header:



The MVP design was created based on team discussions about what would be feasible for Sprint 5. Because development was already far along by the end of Sprint 4, we chose not to incorporate the sprint 5 changes into the MVP so that developers could focus on implementing basic features like Spotify integration and UI responsiveness. The MVP prototype was created based on the MVP specs Christian built out based on our team meetings. The features of the MVP prototype can be found in section 3 of this report. You can view it on InVision here:

[https://invis.io/C2UGJH2AN7H#/393239421 MVP Home](https://invis.io/C2UGJH2AN7H#/393239421_MVP_Home)

To toggle between the flow influenced by feedback and the MVP flow, visit this link -

[https://invis.io/C2UGJH2AN7H#/393823335 Prototype Menu](https://invis.io/C2UGJH2AN7H#/393823335_Prototype_Menu)

We also received peer feedback from our presentation in class. Our main takeaway from this was that multiple people said that they would prefer the MVP application not to open Spotify when playing a playlist. In response to this feedback, we researched ways to play a Spotify playlist while staying on the Harmonize app.

4. New learning prototype description

MVP Specification

We created a minimum viable prototype considering the time constraints. Our MVP has the following features -

- Sign In Page
 - Sign in button which opens Facebook login
- Bottom Nav Bar
 - Social Feed Tab
 - Playlist Tab
 - Explore Tab
 - User Profile Tab
- Social Feed Tab
 - Lists of posts
 - Sorted chronologically
 - Create new post button
 - Social Feed Post Object
 - Contains
 - Spotify song URI / spotify playlist URI / (Harmonize Playlist)
 - Optional user text
 - Like button
 - User Comments
 - List of comments associated with post object
 - Comments have an author and replies
- Playlist tab
 - List of existing playlists you have liked or have joined
 - Choose to play existing playlist
 - Create new playlist button
 - Create New Playlist Page
 - Functionality

- Search spotify
 - Add song from spotify
 - Remove song
 - Playlist object
 - List of spotify URI's
 - Owner user
 - List of users who have joined the playlist
 - User comments
 - List of user comments
 - Playlist Page
 - Displays the playlist object
 - Contribute to Playlist Button
 - Leave playlist button if already joined
 - Play button
 - Comments
- Explore Tab
 - Search bar
 - 3 recent searches
 - Trending playlists
- User Profile Tab
 - User Profile Contains
 - Name
 - Bio
 - Followers
 - Following
 - Personal Feed
 - Logout Button

Goals of Learning prototype was basically to complete the implementation so that it can be handed over to users for beta testing. We did not have time to collect any user feedback on this prototype

- Explore Tab Functionality
 - NOTE: The explore tab allows users to search for songs. Once a song has been searched for, the user can choose to add that song to one of their existing playlists.
 - [BUTTON] Search Bar
 - Allow users to type text. After every letter, search Spotify for the queried string.
 - Display a list of Song Objects returned by Spotify
 - [BUTTON] "Add Song"
 - Add button to every Song Object. Allow users to add searched songs to their User Objects list of Joined Playlist Objects
- User Profile Tab Functionality
 - NOTE: The user profile tab displays the current users profile picture, name, bio, followers, following, and social feed.
 - Displays User Name
 - Displays User Bio
 - Displays User Number of Following
 - Displays User Number of Followers
 - Displays List of User Post Objects

- [BUTTON] "Link Spotify"
 - Allows the user to link their Spotify Account for music playback.
- Playlist Page Functionality
 - NOTE: Every playlist object has a playlist page. The playlist page is where the user can join, leave, add/remove songs, and start music playback. This page is accessible by clicking on a playlist from the "social feed" or the "playlist tab".
 - Displays Information from the Playlist Object
 - [BUTTON] "Join Playlist"
 - [BUTTON] "Play"
 - Starts playlist playback from the first song in the Playlist Object
 - [BUTTON] Comments Button
 - Displays a list of Comment Objects associated with the Playlist Object
 - [BUTTON] Add Song
 - Takes the user to Explore Tab, and opens the search bar.
 - Users may then search for songs and add them to joined/created playlists.
- Social Feed Tab Functionality
 - NOTE: The social feed tab is where users can view posts from other users, create a post them-self, and view a feed of what their friends are doing within the app.
 - Displays a sorted list of Post Objects from followed accounts.
 - [BUTTON] "Create New Post"
 - Allows users to create a new Post Object
- Playlist Tab Functionality
 - NOTE: The playlist tab is where users go to view the playlist they have created or have joined. This page allows users to select playlist objects and go to specific playlist pages.
 - Display list of created or joined Playlist Objects.
 - Clicking a Playlist Object in the list, opens that Playlist Object page.
 - [BUTTON] "Create New Playlist"
 - Allows users to create a new Playlist Object
 - Button created a new Playlist Object with nothing in it, and takes the user to that empty Playlist Object page.
- Create User Objects
 - CONTAINS:
 - User Name
 - Display Pic
 - User Bio
 - User Followers
 - User Following
 - List of Post Objects by User
 - List of Posts liked by User
 - Linked Spotify Account ???
- Create Comment Objects
 - CONTAINS:
 - Owner User Object
 - Time Stamp (date - time)
 - Text
- Create Song Objects
 - CONTAINS:
 - Streaming Service URI

- Title (pulled from service)
- Artist (pulled from service)
- Song Length (pulled from service)
- Album (pulled from service)
- Album Art (pulled from service)
- Create Playlist Objects
 - CONTAINS:
 - Owner User Object
 - Playlist Title (text)
 - Playlist Art (pulled from the Album Art of the first song in the list of Song Objects)
 - List of Song Objects
 - List of User Objects who have joined the playlist
 - List of Comment Objects
- Create Post Objects
 - CONTAINS:
 - Owner User Object
 - User Post Text
 - Number of User Likes
 - List of User Objects who have liked
 - Post Time Stamp (date - time)
 - Playlist Object

Design (Rishma)

Design changes were made according to Sprint 4 feedback

(<https://projects.invisionapp.com/share/C2UGJH2AN7H#/screens/387311462>) and based on the MVP specs (<https://projects.invisionapp.com/share/C2UGJH2AN7H#/screens/393239421>). The design changes made based on Sprint 4 feedback are considered to be future work in terms of development.

The changes that inspired these new designs are discussed in sections 2 and 3 of this document.

Additionally, a new logo was created for CIC and to create a cohesive image for Harmonize. The logo is inspired by audio waveforms to solidly connect our brand image to music.

HARMONIZE

I also created a poster for CIC based on the waveform theme. This poster displays Christian's product description and monetization strategy. The poster can be found here -

https://drive.google.com/file/d/18HyCR2tqP_DnDYNU-6Jd2P6MovMjqriM/view

I also updated our website, beatharmony.squarespace.com, to reflect our logo and colors in preparation for CIC.

Finally, I updated our coded MVP to better reflect our design by adding color and images.

CIC approach and Feedback

Our entry Harmonize fits in Players and Fans category.

CIC judging session was held on 13th November 2019 at Centergy Building, Georgia Tech. There were about 18 teams mainly from our class who participated in the competition. We prepared a poster and

created a website for the competition. Christian lead this effort, but we all contributed to the event thru reviews, being part of demo and logistics.

Website is at <https://beatharmony.squarespace.com/>

We did not receive any negative or conceptual feedback from the CIC judges or attendees, but one of the judges did ask us about a marketing plan which we had not previously considered. Ankit created a marketing plan according to which we would capitalize on existing communities of music heads. This includes advertising on Twitter to users that are interested in and follow music accounts and music-sharing subreddits such as r/hiphopheads.

5. Summary of the team's key activities for Sprint5

During this sprint, we had 4 main goals.

1. Research and understand Spotify integration
2. Decide on the features that will be part of the MVP
 - a. Christian created a list of the features needed for the MVP resulting from team discussion. For Sprint 5, we wanted to prioritize the core functionality of Harmonize. To this end, we decided to focus on the feed, Spotify integration, comments, liked playlists, the explore tab, and the user profile.
3. Work on CIC materials
 - a. For CIC, Rishma created a logo and poster and updated the Harmonize website.
4. Finish MVP:
 - a. Ankit and Pradyumna took the lead on developing the front end of the MVP. Based on MVP specification, Pradyumna created complete UI (My feed page, Playlist page, Search page, User profile page..), made each element touchable, added navigation function and integrated with backend (POST /GET). Pradyumna also added function wherein touching a song will open it in Spotify. Ankit added Login page with authentication, connection to Spotify account.
 - b. Justin built out the back end to accommodate playlists

Monetization Strategy by Christian

Christian also created a monetization strategy in preparation for CIC. Similar to Spotify, Harmonize will utilize a freemium license/subscription-based monetization model. This means that all users, including free, will be able to make use of Harmonize' unique features, ensuring that our application maintains as wide a reach as possible. Free users will experience intermittent ads at an unspecified frequency during music playback. While premium (subscribed) users can experience music playback with no interruption.

To incentivize free users to try the premium subscription, Harmonize will offer a standard 30-day free trial period at which point the premium subscription will be \$2.99/month. It is important that our price remains competitive as we offer a service in addendum to existing paid services.

Our monthly burn rate is expected to be lower than similar music streaming platforms, because Harmonize is simply a collection of Uniform Resource Identifiers (URI) with basic text based social

media layered on top. There is no need to pay for music hosting as that is not hosted on our end. One of our primary points of concern is that we are doubling the number of ads for our users. If a free user on our service is a free user on another service, there's a chance that the user would experience double the amount of ads as they would be receiving ads from two separate services. To avoid this we are narrowing our target audience to premium users on services. Thus they will not experience ads on their home platform. This is a major concern that we are actively looking into avoiding.

6. Detailed documentation

Problem Statement

Existing music platforms create echo chambers based on the music you already listen to.

- Recommendation algorithms have a hard time breaking out of the information they're fed.
- Music-heads are not connected to curators who share in their unique tastes

Music streaming services like Spotify, YouTube Music, etc. collect user data in order to recommend new music. This causes echo chambers of the same genres of music being recommended back to you.

Services such as Spotify, try to break out of this with curated playlists tailored to your mood or activity, but in practice you're still likely to skip a song that's not in a genre you enjoy. Our team is working to recreate the organic experience of hearing a song in your friends car that you love, but existing services wouldn't recommend to you.

Pitch

Harmonize is a collaborative music platform centered around finding, creating, and listening to music playlists together. Harmonize is like a music curation service built to help users find new music through social connections. Music streaming services like Spotify, YouTube Music, etc. use algorithms to recommend new music which can cause echo chambers where the same style of music is recommended to you over and over again. To break out of this we're redefining playlists as a collaborative effort. Up until this point music playlists have been a historically solo activity where you create and listen to them alone. Harmonize is turning playlists into socially driven music hubs designed to recreate the feeling of hopping into your friends car and hearing your new favorite song for the first time

Current Design Pivot

Pivot: Redefine playlists as a living, communal affair.

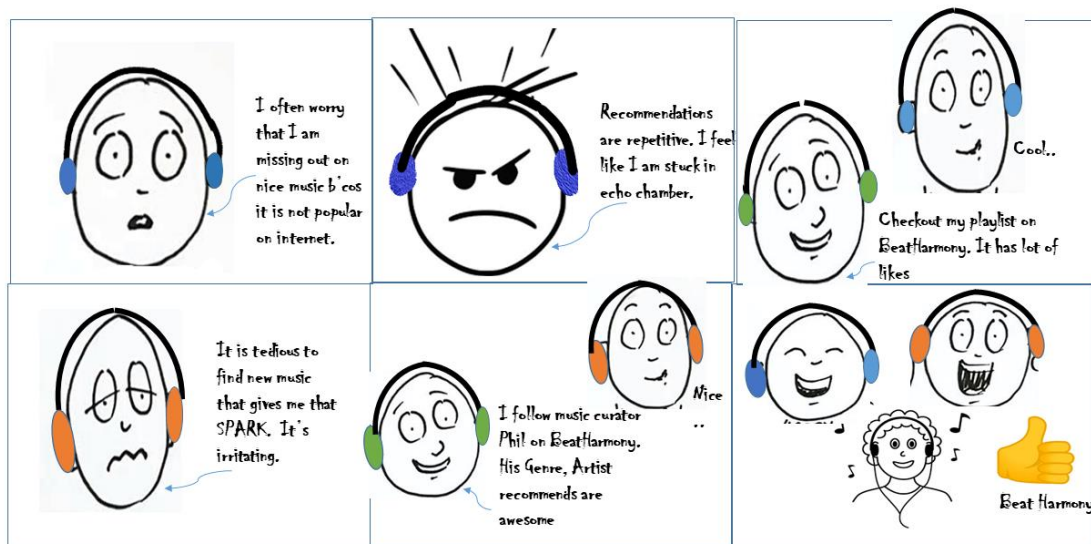
- Every playlist is a living space with an owner, live chat, and an ever changing playlist of songs.
- Users can chat, link to other playlists, recommend songs to add/remove from the playlist, and more.
- With a focus on **creating** playlists, **finding** playlists, and **listening** to playlists.

Each playlist is a social community where users can collaboratively add/remove songs, interact through a live comments section, and link to other playlists. Harmonize is designed to work with, not against, the music services you already love and use. A Harmonize playlist can contain songs from multiple music streaming platforms. In a single playlist a user can switch from Spotify to SoundCloud and never skip a

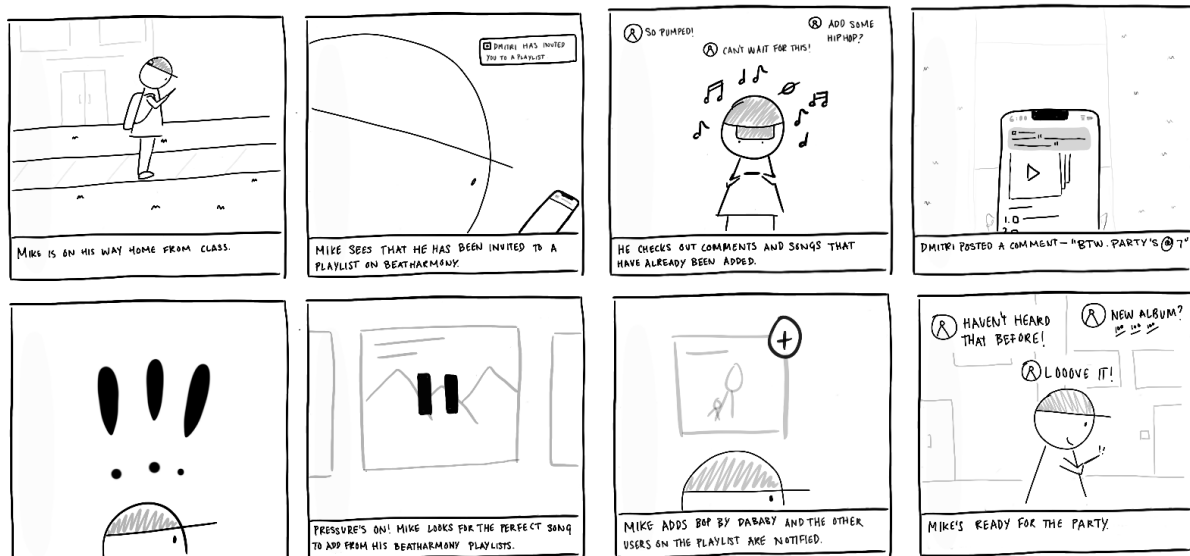
beat. Whether you're looking for socially driven music discovery or a place for your friends to collaborate on a kickass party playlist, Harmonize makes it easy to combine all your favorite music streaming services into one community driven place

Storyboard (Pradyumna / Reshma)

Our initial storyboard at conceptual stage was as below (Pradyumna)



New storyboard based on our design pivot (collaborative playlists) and Christian's video script. (Reshma)



Task Breakdown and Planning (Christian)

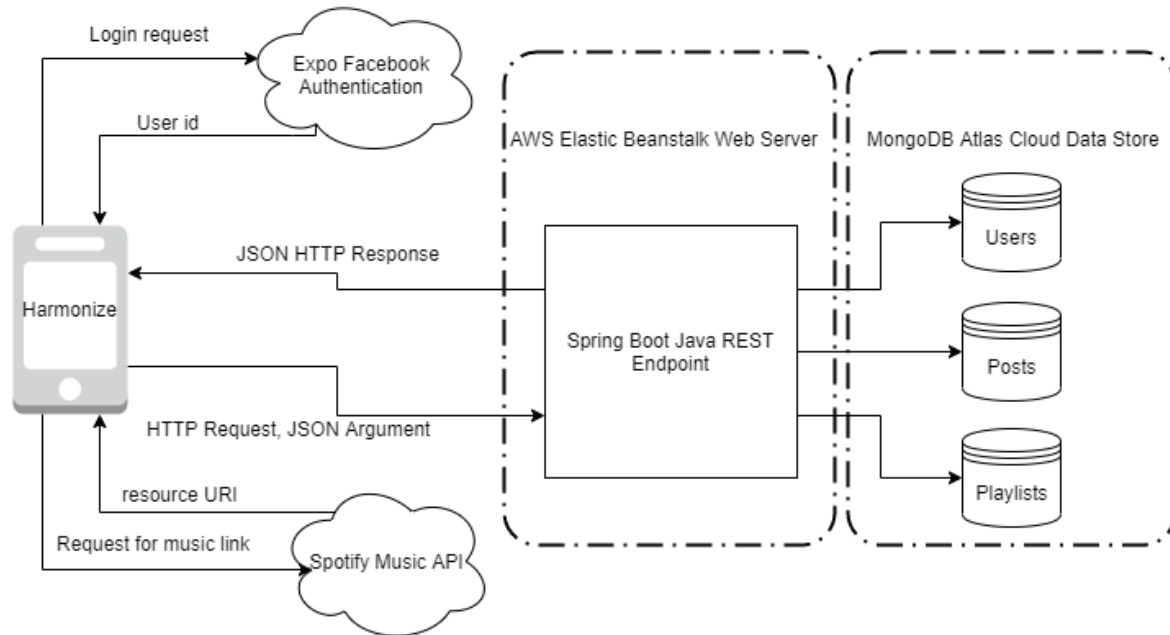
To complete the MVP for Sprint 5, the team had multiple meetings to determine the tasks to be completed and how to break them down into manageable chunks. As a result of these meetings and the updated MVP InVision prototype, Christian created a task breakdown on Trello to determine which objects and functionality had yet to be developed. Team assigned the tasks to themselves and communicated on Trello / Slack.

The screenshot shows a Trello board for a project named "BeatHarmony". The board is organized into four main columns: "To-Do [FUNCTIONALITY]", "To-Do [OBJECTS]", "In Progress", and "DONE".

- To-Do [FUNCTIONALITY]:** Contains three cards with mobile app mockups. The first card is titled "[FUNCTIONALITY] Music Playback". The second card is titled "[FUNCTIONALITY] Explore Tab Functionality". The third card is titled "My Profile" and shows a user profile for "Katherine L.". Each card has a "+ Add another card" button at the bottom.
- To-Do [OBJECTS]:** Contains five cards with titles: "[OBJECT] Create User Object", "[OBJECT] Create Comment Object", "[OBJECT] Create Song Object", "[OBJECT] Create Playlist Object", and "[OBJECT] Create Post Object". Each card has a "+ Add another card" button at the bottom.
- In Progress:** Contains two cards with mobile app mockups. The first card is titled "[FUNCTIONALITY] Playlist Page Functionality". The second card is titled "My Feed". Each card has a "+ Add another card" button at the bottom.
- DONE:** Contains five cards with titles: "Create Bottom Navigation Bar", "Create Social Feed Tab", "Create Sign In Page", "Create Sign In Button - linked to Facebook Login", and "Social Feed Tab - Create 'new post button'". Each card has a "+ Add another card" button at the bottom.

The board header includes the project name "BeatHarmony", a star icon, and several status indicators: "MAS Music", "Free", "Team Visible", and an "Invite" button. There are also several profile icons and a "P" icon in the header area.

Architecture / Technology / Platform choices



Our implementation has been true to our planned architecture since Sprint 3, with one exception: the addition of a Playlists repository so that playlists can be searched for independently. With our design pivot into living social playlists, these playlists need to be pulled from data stores independently, along with their associated comments and social interactions. We made a change in Sprint4 to use Expo's built-in Facebook authentication for our app instead of sourcing authentication through Amazon Web Services. This change was made because we learned new information about Expo and how it supports Facebook authentication more conveniently than AWS Cognito, especially for prototyping purposes.

Another thing we learnt is that the spotify API is useless for embedding and playing spotify stuff through an app. It's real strengths lie in data extraction, searching spotify's catalog, and powering spotify functionality through outside apps (follow someone on spotify , etc)

We will try in terms of functionality in using our app's search bar to search spotify's catalog through their API.

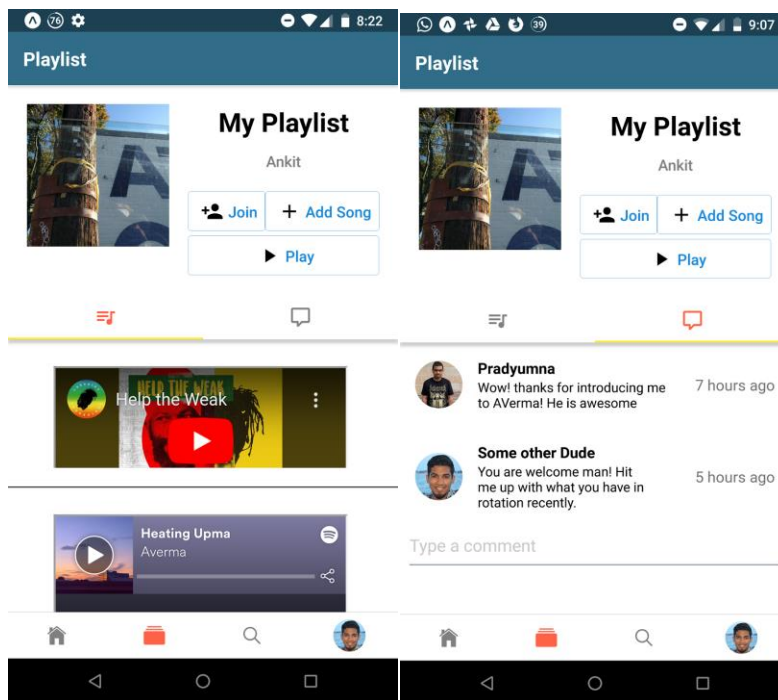
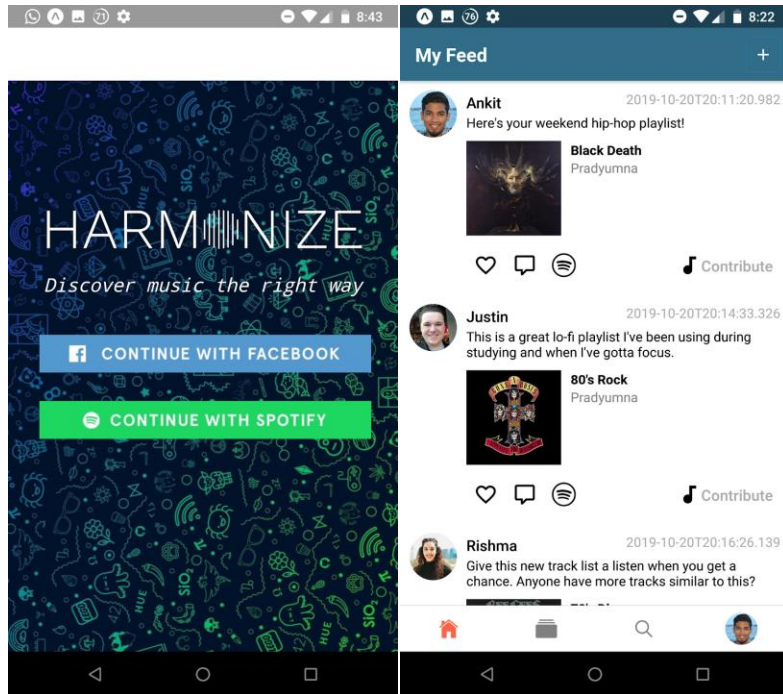
The entire final architecture is as follows:

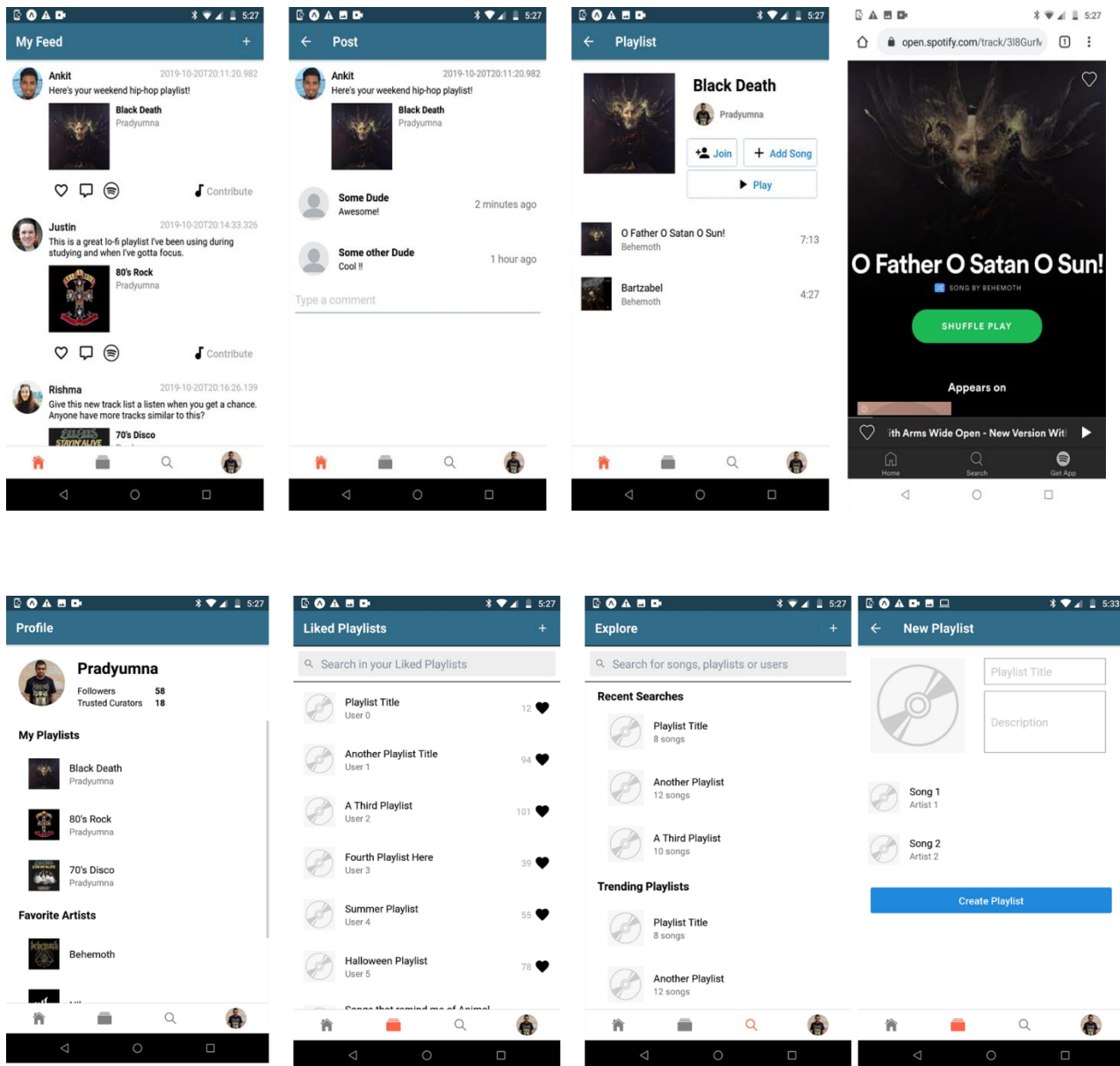
- React Native frontend mobile application
- Built-in Expo Facebook Authentication
- Spotify API integration for resource retrieval
- Spring Boot Java REST Endpoint
 - Hosted on Amazon Web Services Elastic Beanstalk Scalable Web Server
- MongoDB NoSQL Data Store
 - Hosted on MongoDB Atlas Managed Cloud

Front End UI (Pradyumna and Ankit)

During this sprint, significant strides were made on the front end of our application. Pradyumna and Ankit worked on building out functionality on the front end. We made the following progress from Sprint 4:

- Landing/Login
 - Users can now login to Harmonize through both Facebook and Spotify. Our app leverages Expo's AuthSession (<https://docs.expo.io/versions/latest/sdk/auth-session/>) to make authentication calls to Facebook and Spotify's APIs.
- Home/Feed (first icon on menu bar)
 - Users can now click on a post from their feed to open the post.
- Playlist object page
 - Playlists can contain embedded & playable links from multiple streaming platforms
 - Users can play the playlist.
 - Users can join the playlist to contribute songs.
 - Displays Information from the Playlist Object
 - Tracklist tab shows songs on the playlists
 - Comments tab shows new comments.
- Liked Playlists (second icon on menu bar)
 - Displays list of created or joined Playlist Objects.
 - Clicking a Playlist Object in the list opens that Playlist Object page.
 - "Like" icon has been changed to a heart in response to user feedback from Sprint 4.
- Explore
 - The explore tab allows users to search for songs. Once a song has been searched for, the user can choose to add that song to one of their existing playlists.
- User profile
 - User can see follower information
 - how many people they follow
 - how many people follow them
 - User can see their music profile in the profile tab.
 - Displays user's playlists
 - Displays user's favorite artists
 - Displays user's favorite genres
 - Users can see their notifications in the notification tab.
 - A user can view notifications about people liking their playlist posts and contributing to their playlist.
 - Clicking on a song will link to Spotify





Back End (Justin)

The back end basics have been set up for some time now. We have implemented a full-feature Java Spring back end. This endpoint is hosted on Amazon Web Services and backed by a MongoDB NoSQL data store, hosted on MongoDB Atlas Cloud Management Service. From Sprint 4, we have made the following specific changes:

- Set up Playlists repository
 - Implemented basic CRUD functionality
- Added comment functionality to existing Playlists and Posts repositories
- Added additional information to be stored in the Users repository including Spotify link and owned playlists

We made these changes so that our front end can access individual playlists, separate from posts, and render their contents, including comments. The overall construction of the back end is as follows:

- Posts REST Controller -> Posts Repository
- Users REST Controller -> Users Repository
- Playlists REST Controller -> Playlists Repository
- Utility code to create non-resource-based responses to front end

Complete list of API / data stores is attached in Appendix 1

BMC / Value proposition

There is no significant change to the BMC /Value proposition in this sprint. Updated BMC / Value Proposition is show in overview section

7. Future Direction

Our initial project scope was: A feed-focused social media style app based on sharing playlists. A user can act as a seeker or curator at any time. Users can mark other users as 'trusted' music sources and increase another user's trust rating by upvoting their posts. We added Spotify integration to play music and tinder based functionality based on user feedback.

As our project evolved, we sort of changed our focus to user centered playlist to collaborative playlists with updated Pivot as:

Redefine playlists as a living, communal affair.

- Every playlist is a living space with an owner, live chat, and an ever changing playlist of songs.
- Users can chat, link to other playlists, recommend songs to add/remove from the playlist, and more.
- With a focus on **creating** playlists, **finding** playlists, and **listening** to playlists.

Due to timing constraints we dropped following items from our initial vision for MVP

- Tinder based functionality
- Curator focus

If we were to continue this project I would add following features

- Begin by implementing the features that were left out of the MVP, which were matching (Tinder based) and messaging. Although these functions are not essential for music discovery through the app, they do encourage music discovery through social connection with other music fans. Including these functions would also help us separate ourselves from other music services. We would also allow users to connect their accounts from other streaming services than Spotify, which we decided to focus on for the scope of the semester.
- Adding back curator focus – this we think will help the app monetarily and also differentiate from collaborative playlist of Spotify and individuals collaborating separately
- We would also build out the collaborative features more extensively. Currently, all playlists are open for collaboration. A number of users gave us feedback in Sprint 4 stating that they wanted

customizable collaborative playlist features so that they could control whether people can contribute to a playlist and who exactly can contribute

- Another next step would involve putting the built-out application in the hands of users for beta testing. This would allow us to uncover and fix bugs, as well as understand how successfully the users are able to discover music with Harmonize. Although we have been able to test the UI of the application intensively, without a real community of users, we have not been able to verify how often users would discover a new song that they really enjoy.
- Many functionality of UI elements in Front end is not completely functional as they did not have correct back end support. Would go back and fix it.

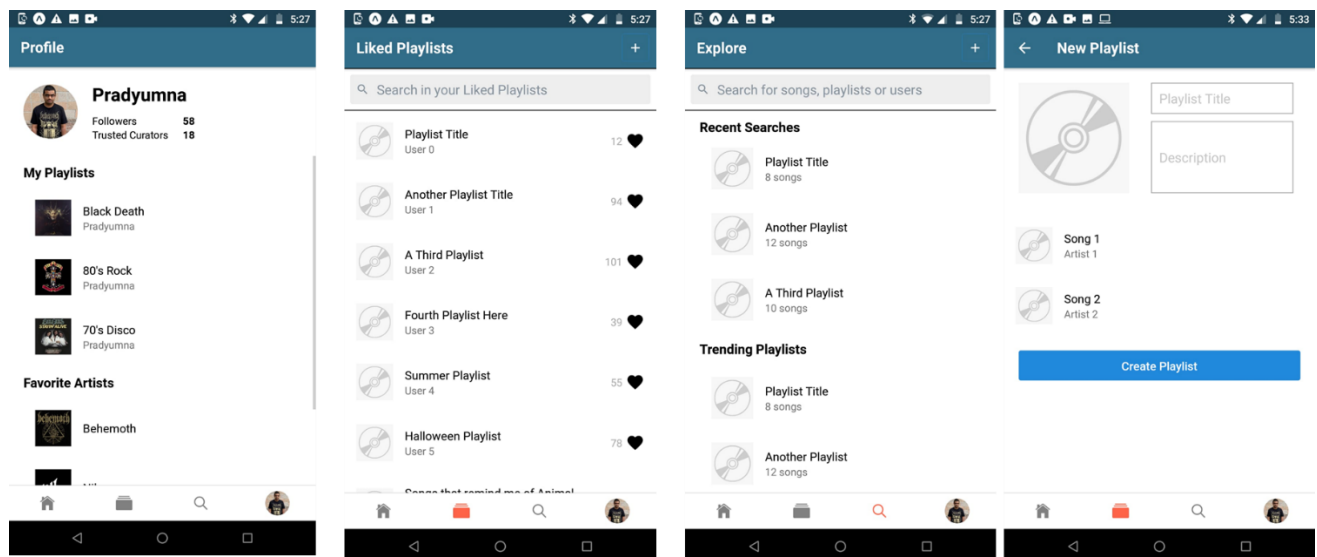
Business wise, it is sort of clear that this idea has a very small edge over competitors. Users would be hesitant to make this as a standalone application given a very small edge. I would probably rethink how we can rebuild the model to make this app over existing streaming service like Spotify as an add-on for a small premium fee.

Part II (Done individually, should only include individual content)

Ankit and myself (Pradyumna) took the lead on developing the front end of the MVP. Based on MVP specification, Pradyumna created complete UI (My feed page, Playlist page, Search page, User profile page..), made each element touchable, added navigation function and integrated with backend (POST /GET) and did basic Spotify integration to open a song.

Following are my specific contributions to Sprint5

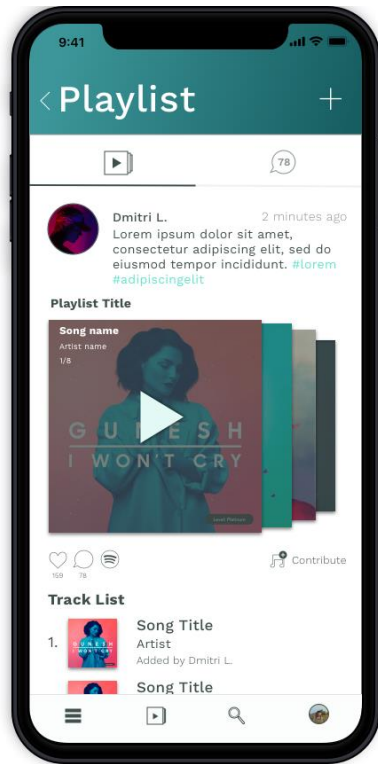
- Created code and tested complete front end as per MVP specification. Some of the functionality still needs coding in back end and hence is partly functional.
- Create Bottom Navigation Bar
 - Add social feed button tab to bottom navigation bar
 - Add playlist tab to bottom navigation bar
 - Add Explore tab to bottom navigation bar
 - Add current user profile to bottom navigation bar
- Create Social feed tab
 - Create new post button on Social feed tab
- Create Explore page
- Making UI elements touchable
- Playlist tab – Create new playlist button'
- Integrated Backend for POST / GET
- Navigate to user profile by clicking user profile picture
- Spotify integration – Can click song to open it on Spotify



Following screenshots show the design requirement and its actual implementation

Playlist page functionality

Design



NOTE: Every playlist object has a playlist page. The playlist page is where the user can join, leave, add/remove songs, and start music playback. This page is accessible by clicking on a playlist from the "social feed" or the "playlist tab".

FUNCTIONALITY:

Displays Information from the Playlist Object

[BUTTON] "Join Playlist"

[BUTTON] "Play"

-Starts playlist playback from the first song in the Playlist Object

[BUTTON] Comments Button

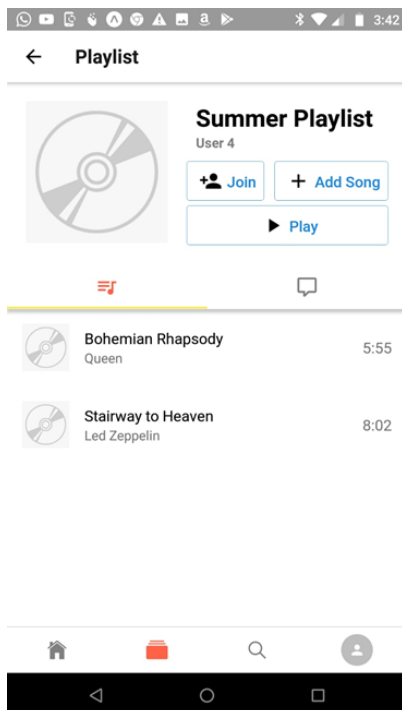
-Displays a list of Comment Objects associated with the Playlist Object

[BUTTON] Add Song

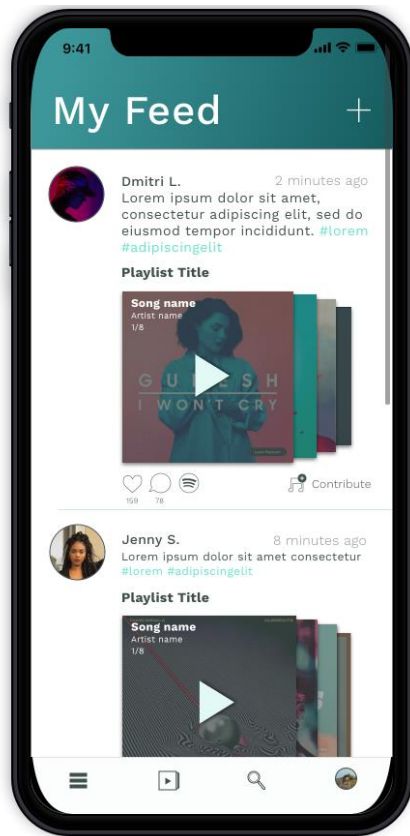
-Takes the user to Explore Tab, and opens the search bar.

-Users may then search for songs and add them to joined/created playlists.

Implementation



MyFeed page functionality (Design)



NOTE: The social feed tab is where users can view posts from other users, create a post them-self, and view a feed of what their friends are doing within the app.

FUNCTIONALITY:

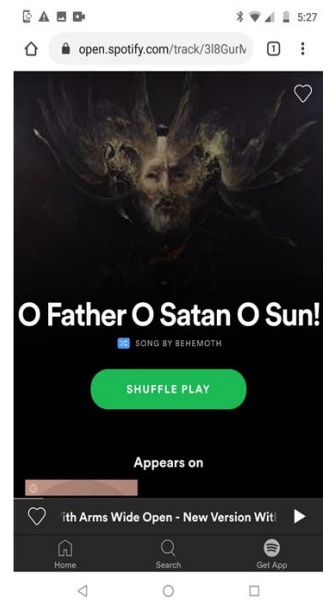
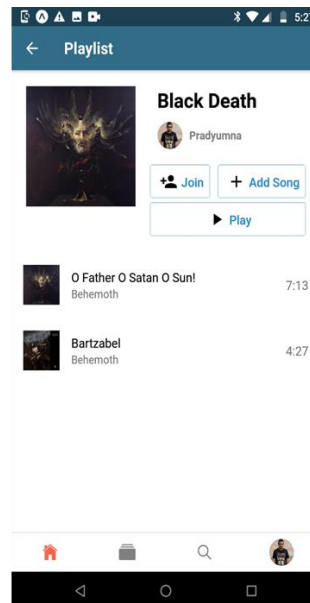
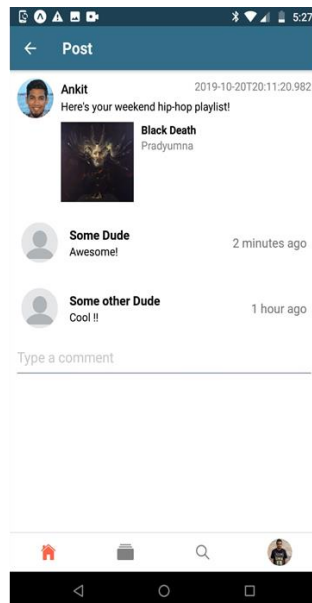
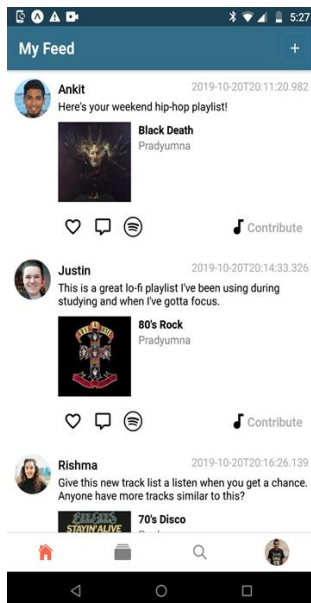
Displays a sorted list of Post Objects from followed accounts.

[BUTTON] "Create New Post"

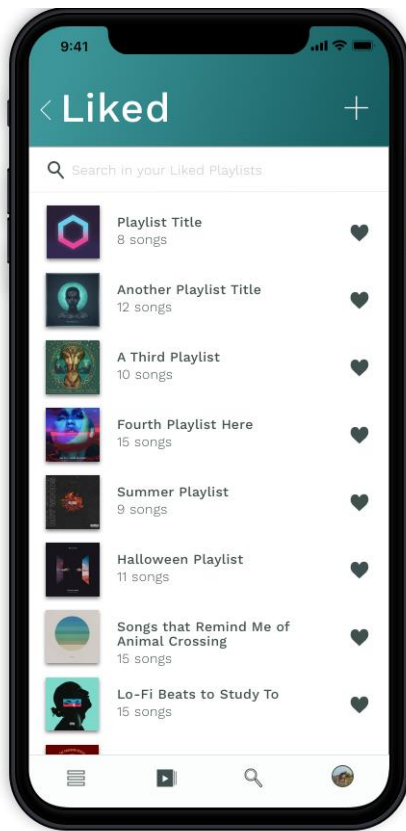
-Allows users to create a new Post Object

Implementation

Clicking on a song will open the song in Spotify.



Liked Playlist (Design)



NOTE: The playlist tab is where users go to view the playlist they have created or have joined. This page allows users to select playlist objects and go to specific playlist pages.

FUNCTIONALITY:

Display list of created or joined Playlist Objects.

-Clicking a Playlist Object in the list, opens that Playlist Object page.

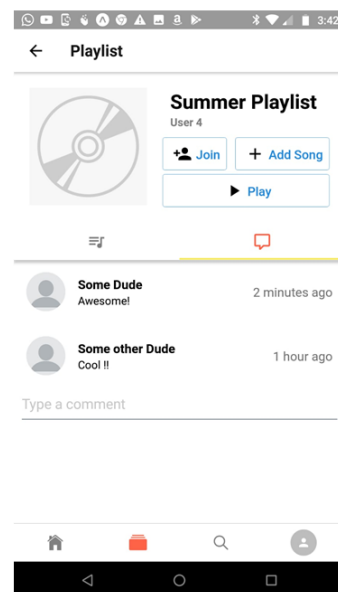
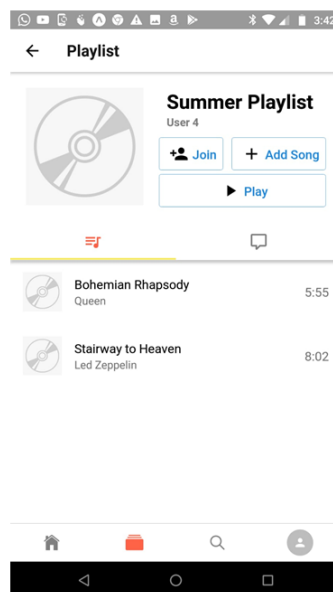
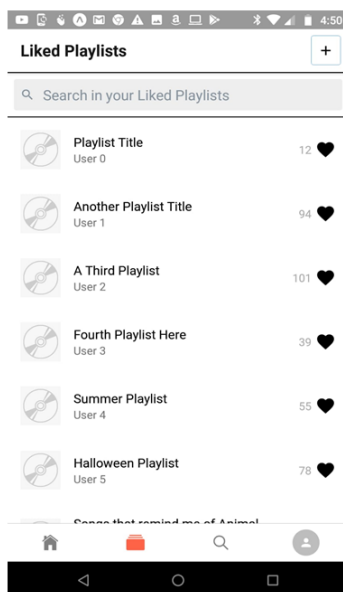
[BUTTON] "Create New Playlist"

-Allows users to create a new Playlist Object

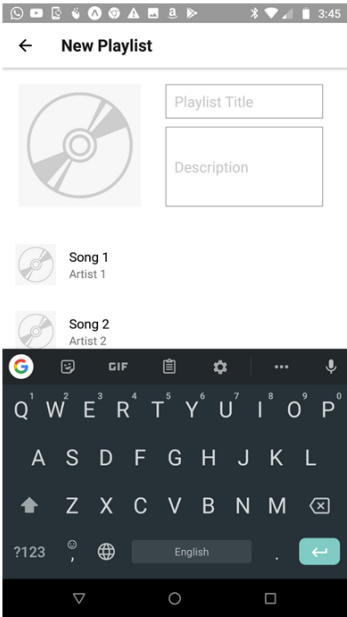
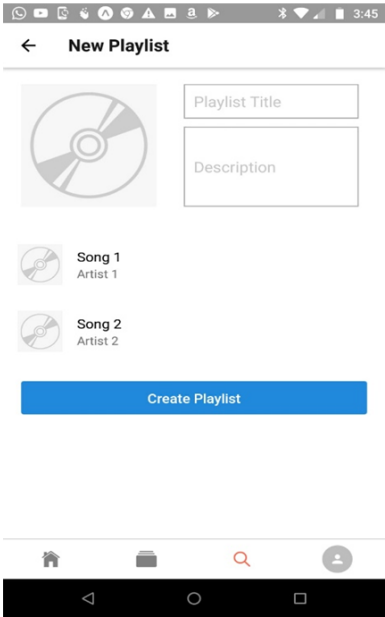
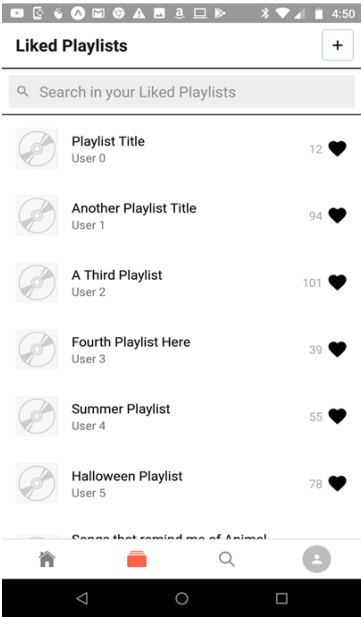
-Button created a new Playlist Object with nothing in it, and takes the user to that empty Playlist Object page.

Implementation

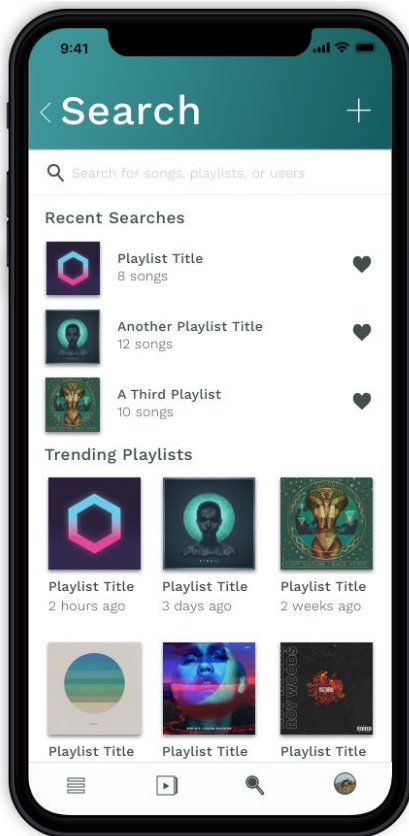
Displaying specific playlist from liked playlist, notifications for specific playlist



New playlist creation; display keyboard for typing description / title



Explore Tab Functionality (Design)



NOTE: The explore tab allows users to search for songs. Once a song has been searched for, the user can choose to add that song to one of their existing playlists.

FUNCTIONALITY:

[BUTTON] Search Bar

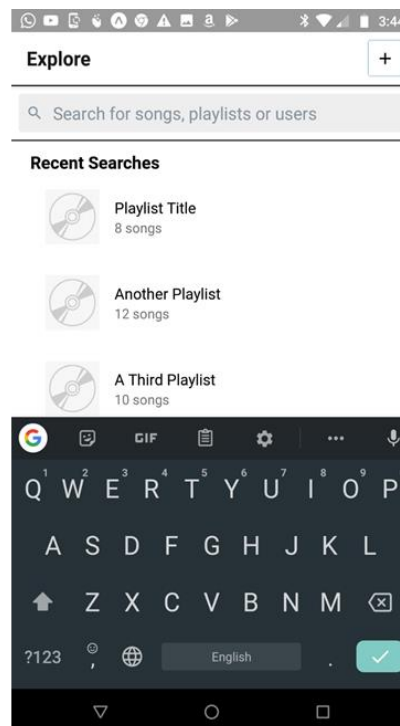
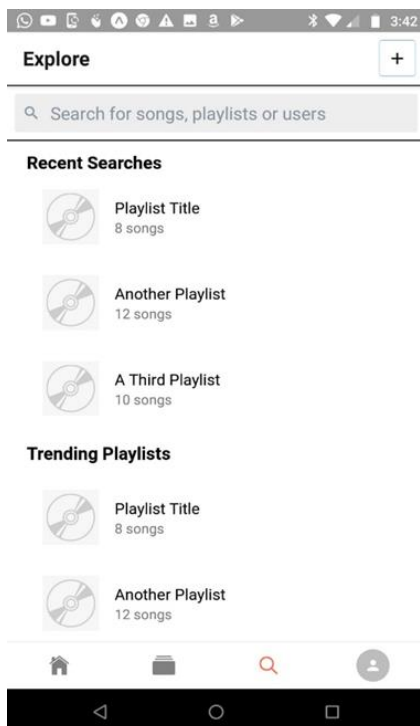
-Allow users to type text. After every letter, search Spotify for the queried string.

Display a list of Song Objects returned by Spotify

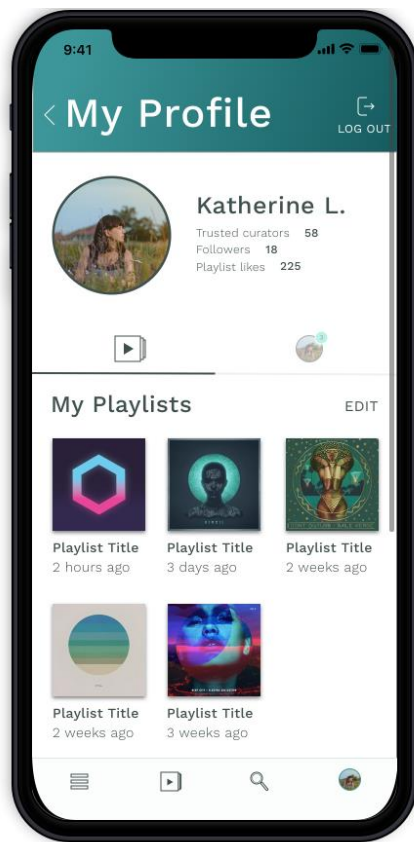
[BUTTON] "Add Song"

-Add button to every Song Object. Allow users to add searched songs to their User Objects list of Joined Playlist Objects

Implementation



User Profile Tab Functionality (Design)



NOTE: The user profile tab displays the current users profile picture, name, bio, followers, following, and social feed.

FUNCTIONALITY:

Displays User Name

Displays User Bio

Displays User Number of Following

Displays User Number of Followers

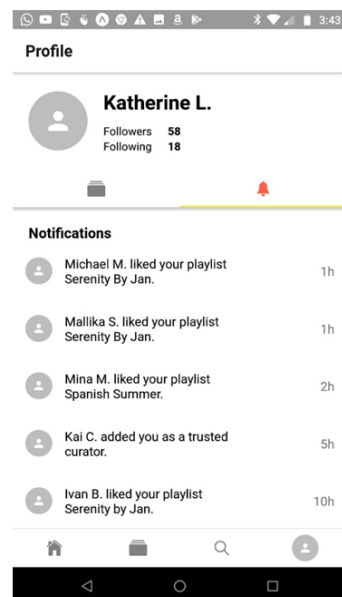
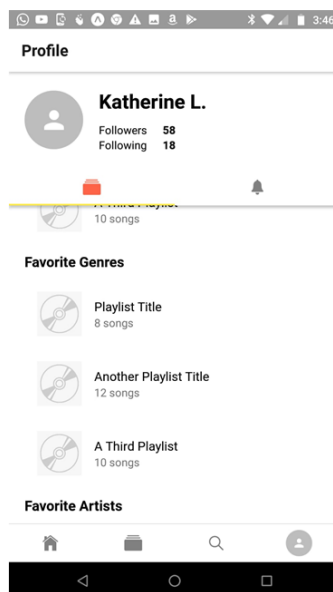
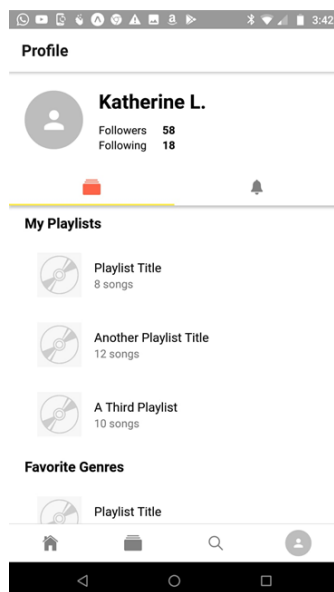
Displays List of User Post Objects

[BUTTON] "Link Spotify"

-Allows the user to link their Spotify Account for music playback.

Implementation

My profile page displaying My playlist, My favorite Genres, My favorite artists and notifications received by user



Navigation bar functionality showing My feed, Liked playlist, Explore page and User profile

My Feed

Dmitri L.

2 minutes ago

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt. #lorem #adipiscingelit

Playlist Title

User 0

40

78

Contribute

Jenny S.

8 minutes ago

>Lorem ipsum dolor sit amet consectetur #lorem #adipiscingelit

Another Playlist Title

User 1

32

12

Contribute

Samuel L.

14 minutes ago

>Lorem ipsum dolor sit?? #lorem #consectetur #adipiscingelit

A Third Playlist

Liked Playlists

Search in your Liked Playlists

Playlist Title

User 0

12

Another Playlist Title

User 1

94

A Third Playlist

User 2

101

Fourth Playlist Here

User 3

39

Summer Playlist

User 4

55

Halloween Playlist

User 5

78

Explore

Search for songs, playlists or users

Recent Searches

Playlist Title

8 songs

Another Playlist

12 songs

A Third Playlist

10 songs

Trending Playlists

Playlist Title

8 songs

Another Playlist

12 songs

Profile

Katherine L.

Followers 58

Following 18

My Playlists

Playlist Title

8 songs

Another Playlist Title

12 songs

A Third Playlist

10 songs

Favorite Genres

Playlist Title

Reflections on CIC

The Convergence Innovation Competition (CIC) is Georgia Tech's annual student competition. The CIC's mission is to enable students to develop innovative applications and services that are commercially viable. Winning entries include a working end-to-end prototype which operates on converged services, media, networks, services, and platforms as well as a value proposition model. Following categories of solutions were identified for the competition.

Climate Solutions

This category is focused on developing practical solutions related to climate change. Solutions may contribute to limiting carbon emissions; coping with environmental, social, and/or business changes; or promoting a greater understanding of how climate change impacts wellbeing and prosperity.

Health on the Move

This category promotes the vision that all individuals—regardless of age, socioeconomic status or health—fully engage in life; sustaining relationships, independence and quality of life while continuing to learn, grow, and contribute to society. Innovation in this category will support health and wellbeing in any phase of life.

Players and Fans

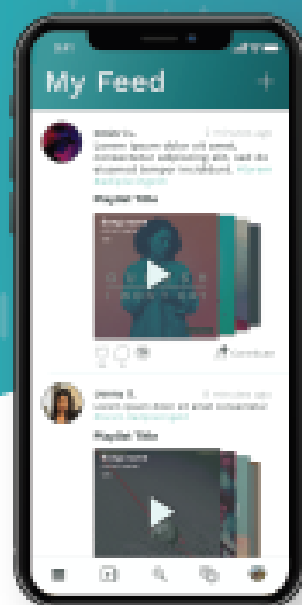
This category encompasses innovations that benefit athletes, artists, entertainers, sports gamers, fans, team owners, or professionals whether at the amateur, youth, or professional level. Solutions could focus on improving performance, expanding access and interest in the craft, lowering financial or other barriers to participation, skills acquisition, injury prevention, and enhancing the fan experience.

Our entry for CIC

Our entry Harmonize fits in Players and Fans category.

CIC judging session was held on 13th November 2019 at Centergy Building, Georgia Tech. There were about 18 teams mainly from our class who participated in the competition. We prepared a poster and created a website for the competition. Christian lead this effort, but we all contributed to the event thru reviews, being part of demo and logistics.

Website is at <https://beatharmony.squarespace.com/>

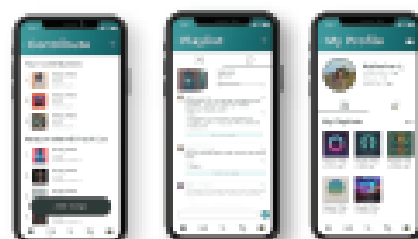


HARMONIZE

A playlist sharing platform by BeatHarmony

Harmonize is a collaborative music platform centered around finding, creating, and listening to music playlists together.

Music streaming services like Spotify, YouTube Music, etc. use algorithms to recommend new music which can cause echo chambers where the same style of music is recommended to you over and over again. To break out of this we're redefining playlists as a collaborative effort. Up until this point music playlists have been a historically solo activity where you create and listen to them alone. Harmonize is turning playlists into socially driven music hubs designed to recreate the feeling of hopping into your friend's car and hearing your new favorite song for the first time.



Each playlist is a social community where users can collaboratively add/remove songs, interact through a live comments section, and link to other playlists.



Harmonize playlists can contain songs from multiple music streaming platforms. In a single playlist a user can switch from Spotify to SoundCloud and never skip a beat.



Harmonize will utilize a freemium (free+subscription-based) model. This means that all users, including free, will be able to make use of Harmonize's unique features.

Whether you're looking for socially driven music discovery or a place for your friends to collaborate on a kickass party playlist, Harmonize makes it easy to combine all your favorite music streaming services into one community driven place.

Learn more at beatharmony.squarespace.com



CIC was a great chance for us to present our ideas to people outside of this class and get feedback from industry, community, and campus experts. There were about 5 judges who visited our stall.

Some of the feedback received from CIC

- Most acknowledged the existence of echo chamber music problem and were appreciative of our innovative solution
- Most attendees were happy with social collaborative playlist idea, mockup and our monetization plan
- One of the judges asked about the marketing plan which we had not considered in our plan

Learnings from CIC:

When we formulated the idea for this project, we were thinking only about innovation. Being part of CIC and observing the approach of other projects, we understood the areas which are currently important for the industry eg. solving global climate problem, help elderly patients and issues which are need of the hour which can generate monetary benefit. We also understood basics of entrepreneurship, steps to consider to make a successful business and having a long term plan for the product. Although we did not win the competition, we gained a lot thru learnings from participation. Last but not least, the snacks provided were awesome!!

Reflections on Sprint5

As mentioned previously, I was initially scheduled to work on backend along with Justin. I switched to Front end in Sprint4. When I joined Ankit for Front end development, I felt the code was not structured well. I basically, started from scratch and set up a correct framework and structure to the application. There was a learning curve as I was new to React Native and Mobile application development. I feel I accomplished a lot by understanding the tools and techniques used for mobile application development. I also learnt about making business models and basics of entrepreneurship.

As part of previous sprints, I had learnt Spring Framework, REST API and gained an understanding of how to set up architecture for mobile application on backend / front end, tools involved and testing methods. As I switched to Front end in Sprint4, I got more hands on experience with React Native tool, Expo snack development and Expo client testing platform. Working in group, I also learnt about prototyping methods using Invision and user feedback strategies.

Of all the Sprints, Sprint5 was the most challenging personally for me. Reason was very clear that we need to get something done by Nov 18th.

In Sprint 5, I learnt how to pass data between screens and method used to navigate between screens on ReactNative. I learnt how to integrate with backend using POST and GET requests. I also transferred from Expo Snack on server to local environment as others were following same method. I also researched and added code so that when we click on a song, it opens in Spotify.

As mentioned in the course website, this course requires a lot of interaction with group. Due to conflicting deadlines from other courses for other group members, it is hard to keep track of progress

and also to maintain strict deadlines for the work assigned. Many times we ended up doing things at the last minute, sometimes without much discussion. I felt this could have been better.

Another thing our group struggled was because of not having formal regular meetings. We were meeting for short time after the class and our interaction was thru Trello for assigning tasks, Slack for communication (which created delay as people were not always responding quickly), github for co-development. We left for individuals to create the Trello card and assign to themselves without a formal check as to if tasks are finally getting completed and what additional help is needed. Adhoc meetings were scheduled towards end, but I think it was too late and did not have full attendance. Sometimes the information flow was also late. Due to all these reasons, we had to cutdown on many planned features in final MVP spec. In Sprint4 and Sprint5 I did assume leadership role to some extent to bring things together. This was unlike my other projects, where we had many meetings and was managed better.

Even with this limitation, I think our group has turned out excellent presentations throughout and did the best we can for MVP demo.

Appendix 1

Backend Detailed document

Here's some general information about backend implementation. The backend is written with Java, assisted by Spring Boot, packaged by Maven and deployed on Tomcat.

There are two primary branches: master and mongodb-impl-posts. The MongoDB implementation has already been merged back to master. I'll focus on what master has in it. I'll start with a package description, and below that, a class description.

- Com.beatharmony
 - RestApplication - landing class to boot the application, for now, contains dummy server setup information
 - .controller - HTTP endpoint code, includes all primary serviceable request code
 - UserController - contains all REST code to interact with the User repository
 - Things to know:
 - Annotations above methods in controllers indicate what type of mapping it is for (GET, POST, PUT, DELETE) and the path to access that service
 - PathVariable annotations in method arguments mean that the last part of the path is an argument to the method
 - RequestBody annotations in Method arguments mean that the argument is an object parsed from a JSON string
 - .data - Repository interfaces, defines what repositories store what
 - UserRepository - interface that defines the MongoDB repository to store users
 - .model - Java objects that define fields that are stored in repositories
 - User - the object that defines what fields a User has in the database, also contains setters, getters and other access methods to properly update fields
 - .util - Contains other useful methods and objects
 - StringResponse - storage method to help package interaction between client and server, simply contains a string field to store a string input between client and server, used as return type and argument in services

To set up your environment, you'll want to do the following:

- Install an IDE of your choice, mine is IntelliJ.
- Install MongoDB. Google it.
- Install Postman to test out HTTP requests.

Services are listed first by primary data store and then by HTTP request protocol. Each individual service is listed by path and other relevant information. Values in brackets are path variables. Request body values exist where noted.

UserController:

- GET
 - /users - get all users, this is a service only available for API testing

- /users/id/{id} - get a user by id
- /users/name/first/{name} - get a user by first name; more exhaustive name search coming next sprint
- /users/username/{username} - get a user by user name
- /users/{id}/trusted - get a user's trusted users list
- POST
 - /users - create a new user, JSON body contains fields
- PUT
 - /users/addtrusted/{id} - add a trusted user to id in the path, JSON body contains id of which user to add to the list
 - /users/removetrusted{id} - remove a trusted user from id in the path, JSON body contains id of which user to remove from the list
- DELETE
 - /users/{id} - delete a user, mostly used for testing currently

PostController:

- GET
 - /posts - return all posts
 - /posts/{id} - get a post by id
- POST
 - /posts - add a new post, JSON body contains post information and fields
- DELETE
 - /posts - delete a post by id, JSON body contains id of post to delete

Data Store information

MongoDB, noSQL database, store data as json

Collections:

- Posts
 - Post id (database-created id)
 - Posted by (key to user)
 - Post text (string)
 - Liked by (list of user keys who like the post)
- Users
 - User id (database-created id)
 - Username
 - Name (actual name in words)
 - Email
 - Trusted users
 - Interest profile (list of genres and values from 0-1, 0 being no interest, 1 being full interest)

Possible HTTP request frameworks:

GET

app/users - returns all relevant information about a user, basic searching, only accessible through sub-requests

- Sub-requests:
 - app/users/id/(user id)
 - app/users/username/(username)

app/posts/(post id) - returns post id, posted by user, post text, list of users who have liked post

- Sub-requests:
 - app/posts/(post id)/by - returns who created the post
 - app/posts/(post id)/text - returns post text
 - app/posts/(post id)/likes - returns list of users (user ids only) who have liked said post

POST

app/newpost - creates a new post

- Arguments:
 - User - user who is creating the post
 - Post text - text inside post

PUT

app/likepost - adds a like to a post

- Arguments:
 - Post id - post that is being liked
 - User id - user that is liking a post

app/unlikepost - removes a like from a post

- Arguments:
 - Post id - post that is being unliked
 - User id - user that is unliking a post, must exist in the liked list of the post

app/editpost - edits text of post

- Arguments:
 - Post id - post that is being edited
 - User id - user that is editing a post, must match user who created post
 - Text - new text to display in post

app/user/(user id)/addtrusted - adds new trusted user to user's trusted list

- Arguments:
 - User id - user that is being added to list, must not be present in list

app/user/(user id)/removetrusted - removes trusted user from user's trusted list

- Arguments:
 - User id - user that is being removed from list, must be present in list

DELETE

app/deletepost/(post id) - delete an existing post

- Arguments:
 - Post id - post that is being deleted
 - User id - user that is requesting deletion of a post, must match user who created post