Pradyumna Mukunda
Fall 2019 – Sprint 3 Individual Report

CS 8803 Mobile Application and Services
Georgia Tech, Atlanta, Georgia, USA

Team Name:
Beat Harmony
Project Name:
Beat Harmony
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 Project Overview

Following section gives description of Problem the team is trying to solve, Domain research and user interview results, approaches considered and details of chosen approach

1.1 Problem:

Novelty seeking music-heads cannot find the fresh new music they need because:

- *Music-heads are stuck in echo chambers as a result of the naive recommendation algorithms of Youtube and Spotify.*
- *Music-heads are not connected to curators who share in their unique tastes.*

1.2 Domain Research, User Interviews

Customer discovery interview was conducted by Ankit Verma who is the Team lead for this project as part of First Interview assessment [1]. Following insights could be derived out of the work:

- Major platforms used for finding music are – Spotify, Soundcloud, Reddit, Youtube, Friends or Word-of-mouth, Social media (Facebook, Twitter)
- Participants felt the current algorithmic solution suggest something based on their previous choice and can be repetitive. Music-heads worry that they may miss out on other gems as it could be in the bottom of list.
- Participants want music discovery platforms to suggest new music based on the choices of people with similar taste or on their own established tastes
- The majority of participants value ease and convenience for music discovery. Participants feel like they have to put in too much effort to find new music
- Person who seeks novelty in music and digs for obscure music is more likely to be dissatisfied with current solutions than someone who is fine with listening to what they already are familiar with.

- Many interviewees seemed to imply that getting new music recommendations from their friends was their most reliable method of getting to hear new music that they really like.
- There is no one size fits all when it comes to music discovery as each person has different preferences based on demographics

There is a significant trade-off between convenience and high-quality discovery, and yet users opt for convenience in every case. It seems like the solutions that the interviewees use all currently strike a balance between speed, convenience, and finding music that is “just good enough for now”. What is missing for most individuals is a style of discovery that closely resembles word-of-mouth recommendations from friends, but users are not willing to sacrifice convenience to get there. There seems to not yet be a solution that can find a balance between convenience and quality organic, word-of-mouth, style discovery, and our team wants to explore a potential viable product in this area.

1.3 Solution Approach Initial

Each team member came up with three separate approaches to solve the problem. All suggested approaches were rated based on following criteria

- Ease of Implementation
- Customer Convenience
- Taking recommendations from Friends
- Having connectivity to platforms

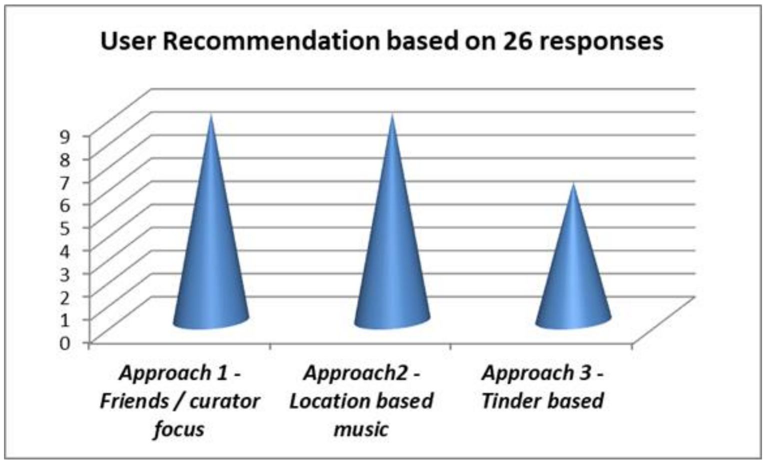
Three approaches were selected and presented as part of Sprint1 based on the above criteria**.** **Approach 1 below is very close to the approach I suggested by me as part of individual effort**. Mood based feature in my approach was not accepted by team as it involved predicting user mood. My idea was to predict user mood based on time of the day, user location setting, user current activity such as driving / workout / dining, etc. As it can be a hard task, team did not include this in the selected approach.

Approach presented in Sprint1 are as below:

1. Social-Media platform with each user’s ‘trusted’ curators as the focus
    a) user type A acts as a music ‘seeker’ and sees recommendations from the people they trust most
    b) user type B acts as a music curator and posts recommendations for others
2. Location based music curation
    a) Uses user’s geographic data to recommend music.
    b) Users may submit song recommendations for current location and have other users vote.
3. Tinder for music recommendations

1.4 Feedback from Sprint1 presentation and Additional interviews

This section is contributed by me. Post the Sprint1 presentation, we received feedback and also many students in the class gave their preference for the approach. Graph above summarizes the ratings received for each of the three approaches.



As can be seen from the chart above, 9 participants preferred Approach 1 which has Friends / Curator focus and Approach 2 (Location based music) and 6 participants preferred Approach 3 which was Tinder like functionality for music. In addition to the above, separate limited interviews were held to get user preference for each of the approach.

(This section is contributed by me) Following is the summary of questions / Feedback received as part of Sprint1 Peer review feedback and our reply**.**

You should think about users who dont necessarily want to discover a new genre of music but rather just stay within their niche of music they already know

Team should clarify how their solutions will differentiate amongst competitors that already have music recommendation algorithms."

"I still don't see how this problem is not already solved by existing problems. If the focus is creating a social media platform based on music--this already exists and you can add friends and 'trusted' users there already.

An app that curates music without playing it is not enough

Based on the responses, team decided to go ahead with developing full blown prototype based on Approach 1 with few changes
Social-Media platform with each user's 'trusted' curators as the focus

- Tinder like function to match and connect people with similar music taste
- Integrate Spotify so that user can play music

1.5 Selected Approach Details (Team decision with my participation)

Selected Approach is 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.

Features

- Social feeds with scroll
- Trusted source recommends higher in the feed order
- Trusted source identification method
    - Explicit follow of curator
    - marked trusted source
    - number of posts listened or liked by user
- Spotify Integration
- Setup User / Curator music profile
- Up-vote Playlist / Curator / User post
- Tinder for matching users with same music taste

Figure below shows the conceptual view of the selected approach. (**Conceptual view is created by me).** It identifies the activities of Music seeker and Curator, Functions of the app and displays. It will have integration with Spotify database to play music. Local stores will include user profile and precompiled list storage.

Conceptual View

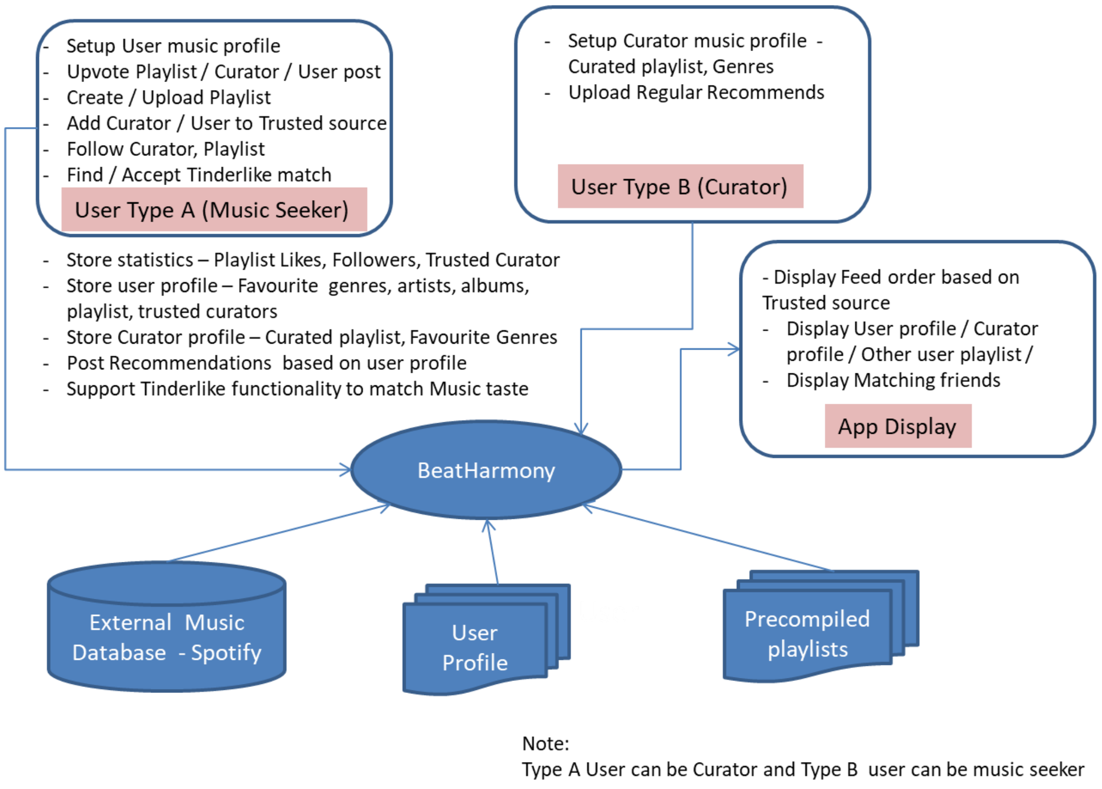
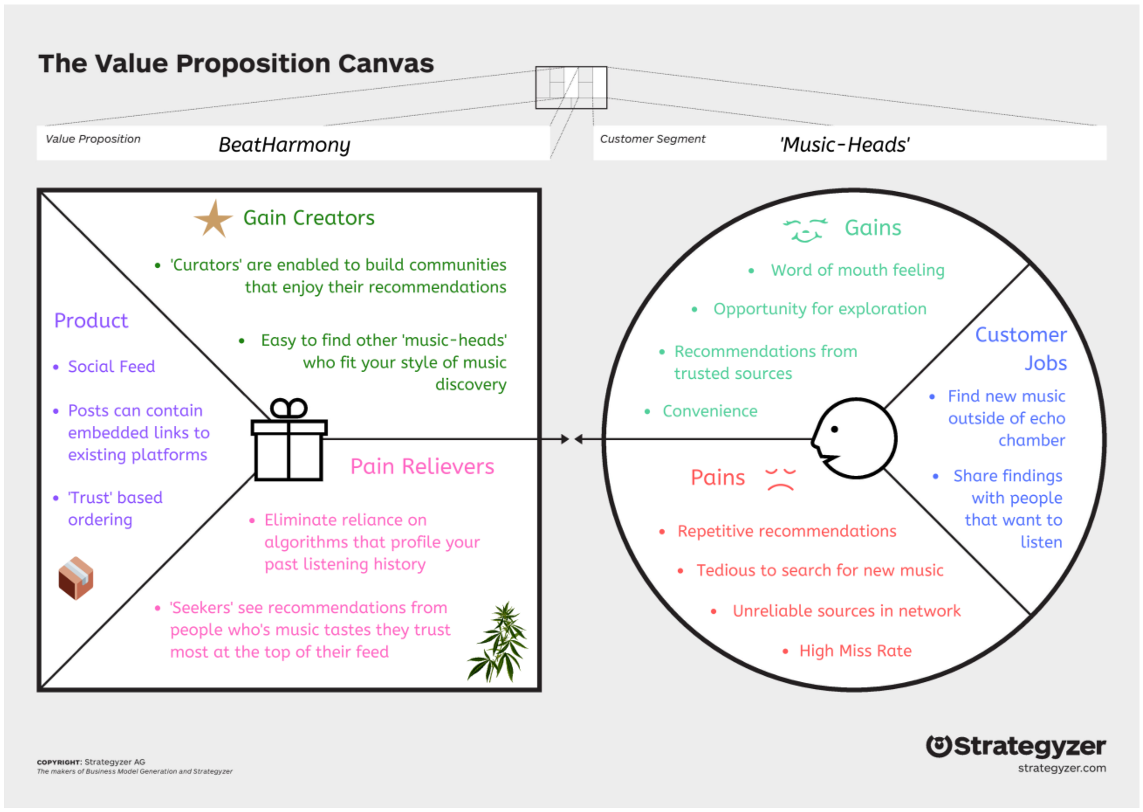

- Setup User music profile
- Upvote Playlist / Curator / User post
- Create / Upload Playlist
- Add Curator / User to Trusted source
- Follow Curator, Playlist
- Find / Accept Tinderlike match
User Type A (Music Seeker)
- Setup Curator music profile - Curated playlist, Genres
- Upload Regular Recommends
User Type B (Curator)
- Store statistics – Playlist Likes, Followers, Trusted Curator
- Store user profile – Favourite genres, artists, albums, playlist, trusted curators
- Store Curator profile – Curated playlist, Favourite Genres
- Post Recommendations based on user profile
- Support Tinderlike functionality to match Music taste
- Display Feed order based on Trusted source
- Display User profile / Curator profile / Other user playlist /
- Display Matching friends
App Display
BeatHarmony
External Music Database - Spotify
User Profile
Precompiled playlists
Note:
Type A User can be Curator and Type B user can be music seeker

Figure below shows the Value proposition Canvas (Team design)


The Value Proposition Canvas
Value Proposition
BeatHarmony
Customer Segment
'Music-Heads'
Gain Creators
'Curators' are enabled to build communities that enjoy their recommendations
Easy to find other 'music-heads' who fit your style of music discovery
Product
Social Feed
Posts can contain embedded links to existing platforms
'Trust' based ordering
Pain Relievers
Eliminate reliance on algorithms that profile your past listening history
'Seekers' see recommendations from people who's music tastes they trust most at the top of their feed
Gains
Word of mouth feeling
Opportunity for exploration
Recommendations from trusted sources
Convenience
Customer Jobs
Find new music outside of echo chamber
Share findings with people that want to listen
Pains
Repetitive recommendations
Tedious to search for new music
Unreliable sources in network
High Miss Rate
COPYRIGHT: Strategyzer AG
The makers of Business Model Generation and Strategyzer
Strategyzer
strategyzer.com

Value seen for customer are

- Word of mouth feeling
- Opportunity to explore new music
- Recommendation from trusted sources
- Convenience

Business value

- Music curators are enabled to build community that enjoy their recommendations
- Easy to find music heads who fits your style of music

1.6 Feedback from Sprint 2

What they are trying to build is another streaming service similar to Spotify. The problem is that I don't think they do anything to differentiate themselves from it. Also they do not have the capital to become a company like Spotify. Alternatively, they could be a product that works in conjunction with Spotify, but as brought up in class, I don't know how likely people are to get an additional application just to find recommendations outside of Spotify

The social element is already built in to Spotify. In response to this they said the social aspect of Spotify is something people tend to not participate in but I think if people don't want to participate it in Spotify why would they want to participate it in an app that takes them outside and then back into Spotify.

In Spotify Recommendations are based on

- listening history
- search history
- Playlists containing songs you like

Our approach is based on a style of discovery that closely resembles word-of-mouth recommendations from friends. Instead of having an algorithm recommend music for you which again is based on learning history, our team envisages that quality organic music can only be found by human curators who are connoisseurs in music. Curators and music-lovers love to connect and share their likings, comments, recommendations.. This platform allows direct contact between the two. Adding Tinder like functionality is to create a social network of like minded music circle which is private and not all encompassing as in Spotify

It seems your app will have its own login/account creation process. You should consider using Facebook to authenticate to your system. This way you can pull in people friends and obtain more information about them.

Excellent suggestion. We are considering this suggestion for inclusion

The team should explain a bit how they plan to market the product. I'm still a bit confused as to whether it's a service that they plan to add with major music providers, or a standalone app? If it's the latter, how do they plan to market it, when everyone uses Spotify

Idea to have a standalone app which has plugins to music providers like Spotify

2 Learning Prototype – Sprint 2

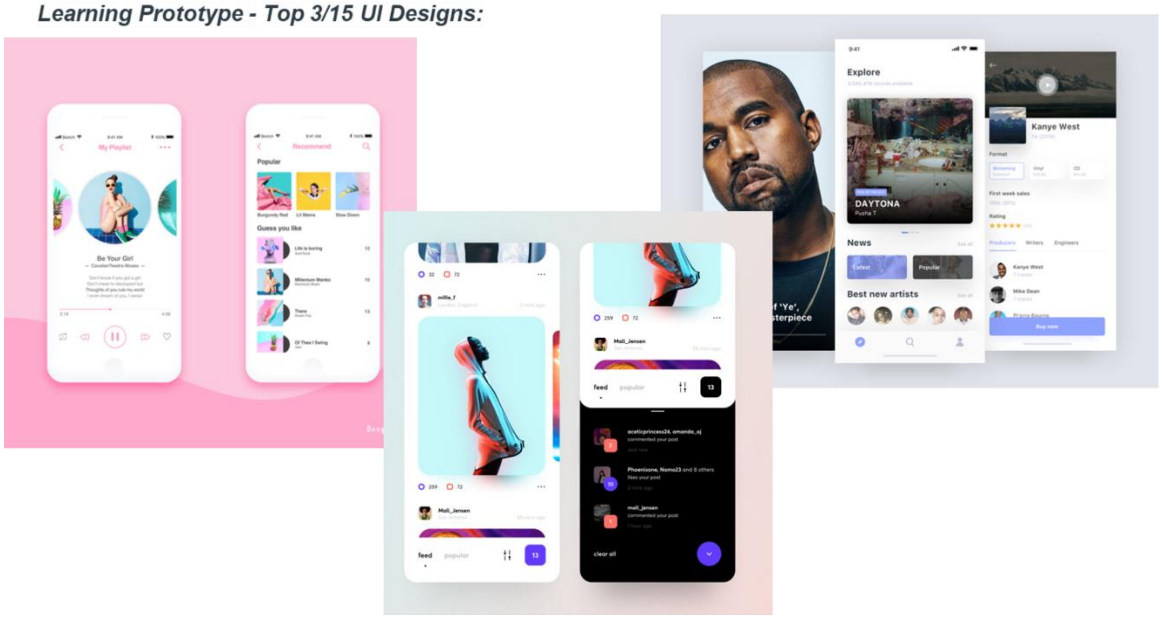
Goals of learning prototype in Sprint2 was

- 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.

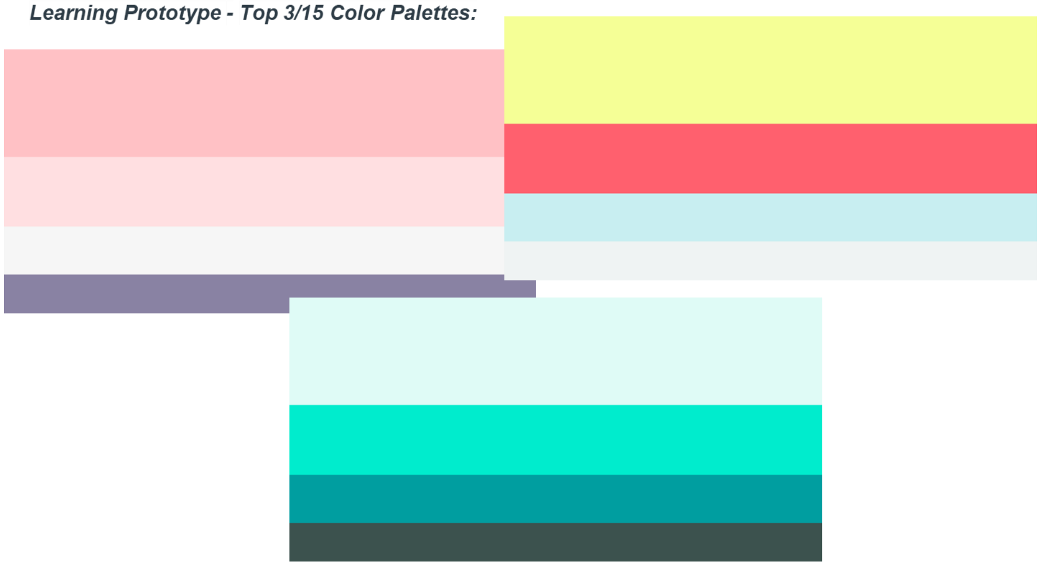
About 28 people were interviewed to collect their preference on 15 UI designs and 15 Color palettes. Based on above survey we narrowed down to the top 3 UI design and color palette.

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

Learning Prototype - Top 3/15 UI Designs:



Learning Prototype - Top 3/15 Color Palettes:



3 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

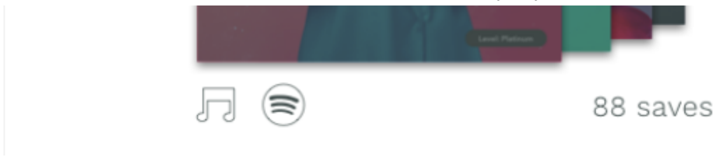
In addition, we created a clickable prototype of mockup screens / UI and planning to get user feedback. Insights we would look for in the prototype are

- Understanding pains and gains for the two primary users – Music listeners and Music Curators
- Understand the different styles of user interactions

Interview questions are listed in appendix 1

3.1 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:
   i. Find a different icon for liking a post, because the music note is actually Apple Music's logo.
   ii. Put the number of saves close to the "like" button
   iii. 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:
   i. Switch the live chat section to an asynchronous comments section.
   ii. 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 on 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.

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

4 Summary of key activities in Sprint3

Important decisions taken (**Team Decision)**

- Decide on a hosting service for the web server
- Decide on a platform for the web server
- Decide on authentication
- Decide what music API we want to hit (for now)
- Finalize the decision on a mobile development platform
- Decide what web server we want to use (Tomcat, Glassfish, etc.)
- Decide if we want to use a noSQL option (mongoDB preferred)

Big tasks we accomplished for sprint 3

- Set up and configure server architecture
  - Set up physical server
  - Set up web server environment
  - Set up backing database environment
- Set up and configure dev environment for mobile
- Create Git org repository
- Develop interactive mobile UI on InVision
  - Screens
    - Login
    - Home page with feed
    - Profile and Playlist page
  - Test UI with users
- Set up authentication **(Contributed by me**)
- Develop initial web services **(Contributed by me)**
  - Develop dummy web services to test
  - Set up database tables
  - Expand web services to interact with database
- Create design doc for app functionality
- Set up development environment for front end
- Consolidate User feedback and incorporate suggestions in approach **(Done by me)**

5 Description of significant changes/project pivots

5.1 Pitch

BeatHarmony is a music curation service built to help users find new music through social connections. 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.

5.2 Updated Design Pivot

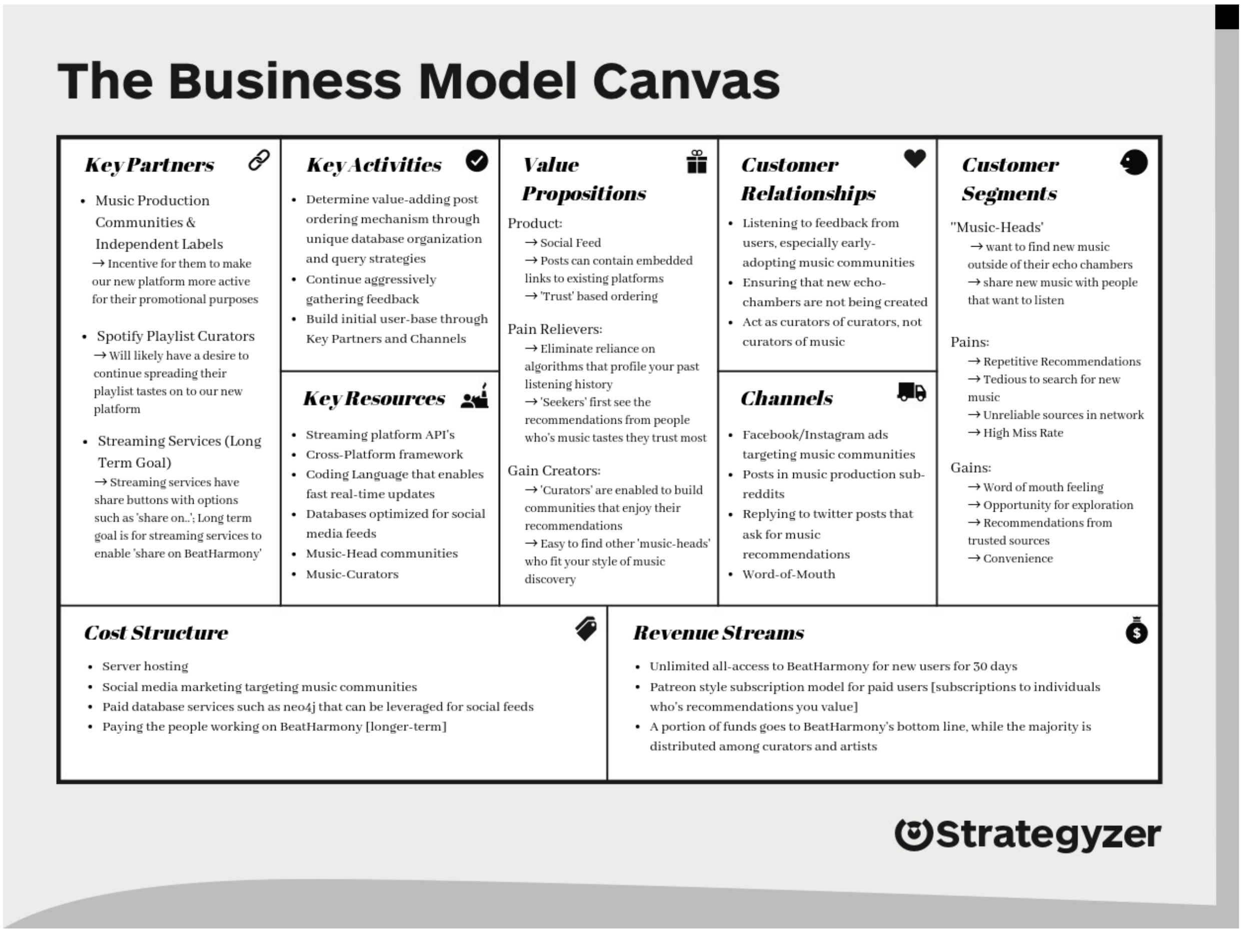
Music playlists are a primarily solo activity.

- *Creating playlists.*
- *Listening to playlists.*

Pivot: Redefine playlists as a living 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.*

5.3 Business Model Canvas



The Business Model Canvas

Key Partners

- 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

- 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

Key Resources

- Streaming platform API's
- Cross-Platform framework
- Coding Language that enables fast real-time updates
- Databases optimized for social media feeds
- Music-Head communities
- Music-Curators

Value Propositions

Product:
→ Social Feed
→ Posts can contain embedded links to existing platforms
→ 'Trust' based ordering

Pain Relievers:
→ Eliminate reliance on algorithms that profile your past listening history
→ 'Seekers' first see the recommendations from people who's music tastes they trust most

Gain Creators:
→ '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

- 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

Channels

- Facebook/Instagram ads targeting music communities
- Posts in music production sub-reddits
- Replying to twitter posts that ask for music recommendations
- Word-of-Mouth

Customer Segments

"Music-Heads'
→ want to find new music outside of their echo chambers
→ share new music with people that want to listen

Pains:
→ Repetitive Recommendations
→ Tedious to search for new music
→ Unreliable sources in network
→ High Miss Rate

Gains:
→ Word of mouth feeling
→ Opportunity for exploration
→ Recommendations from trusted sources
→ Convenience

Cost Structure

- 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

- 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

Value seen for customer are

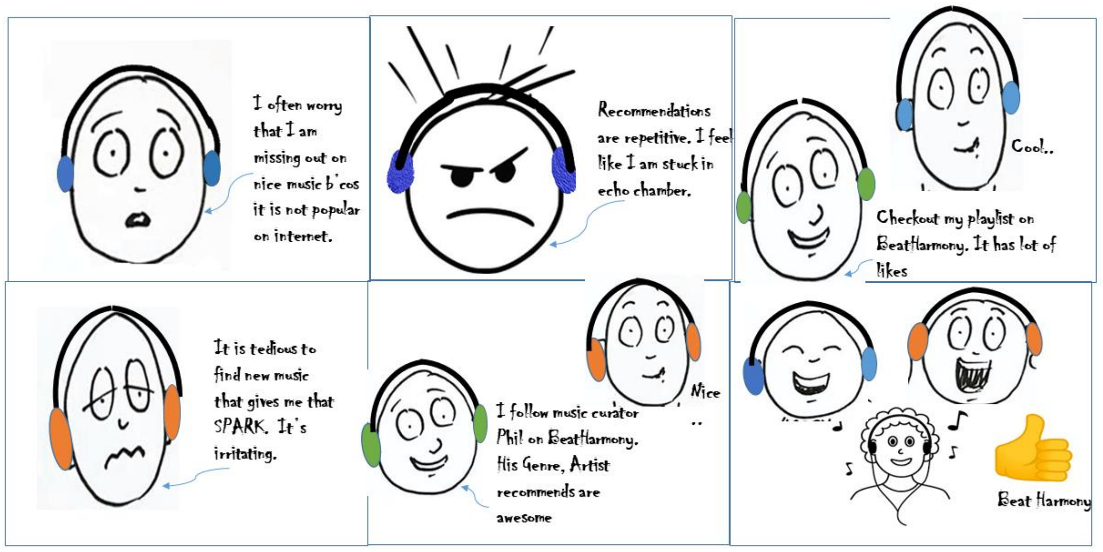
- Word of mouth feeling
- Opportunity to explore new music
- Recommendation from trusted sources
- Convenience

Business value

- Music curators are enabled to build community that enjoy their recommendations

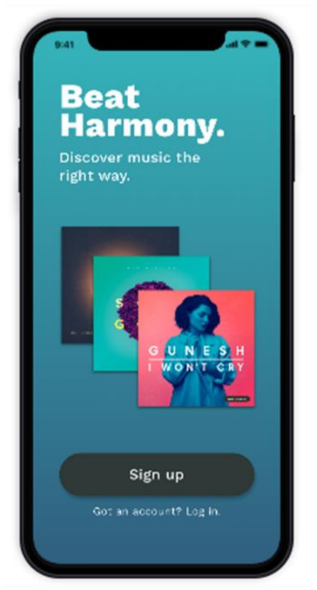
Easy to find music heads who fits your style of music

5.4 Storyboard (This storyboard is created by me)

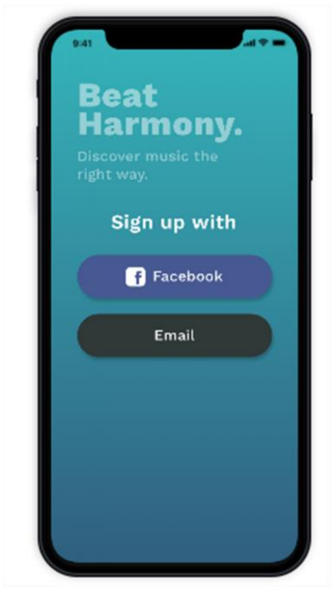

I often worry that I am missing out on nice music b'cos it is not popular on internet.
Recommendations are repetitive. I feel like I am stuck in echo chamber.
Cool..
Checkout my playlist on BeatHarmony. It has lot of likes
It is tedious to find new music that gives me that SPARK. It's irritating.
Nice
I follow music curator Phil on BeatHarmony. His Genre, Artist recommends are awesome
Beat Harmony

5.5 Mockup

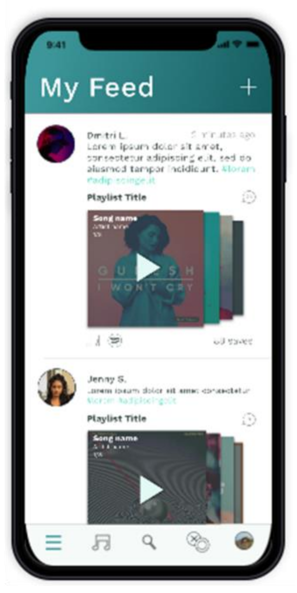
This section lists the updated mockup screens. Following are the signup screen. User can also signup with their media accounts. Home page after signing is also shown



Signup

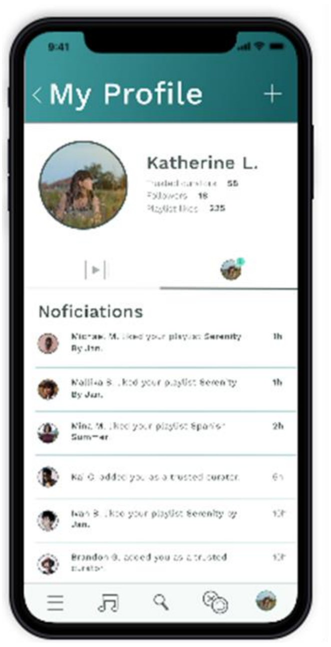


Signup with social media

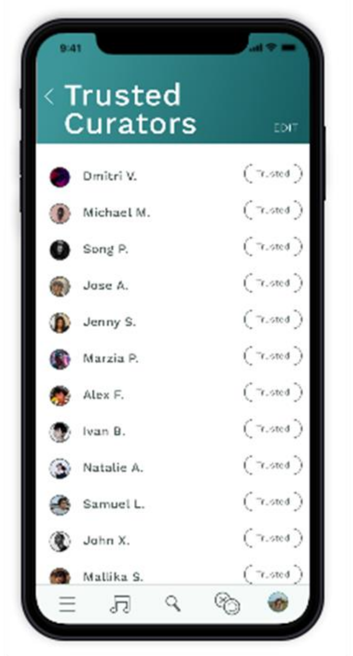


Home Page

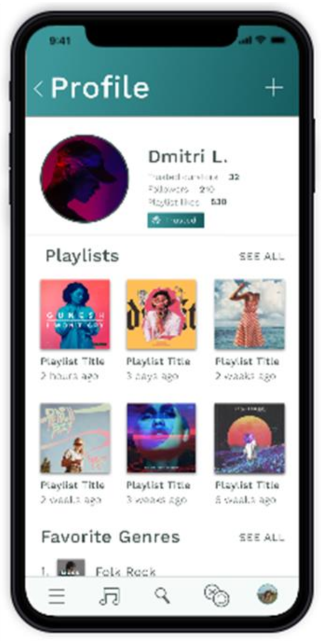
Following screens show details of profile is displayed. App can display the listed of trusted curators and we can also check profile of curators or other user



My profile

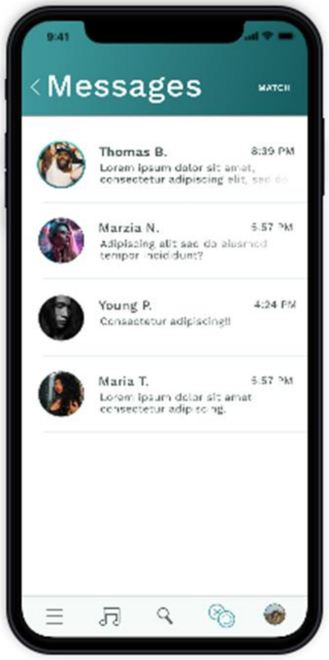


My Trusted curators

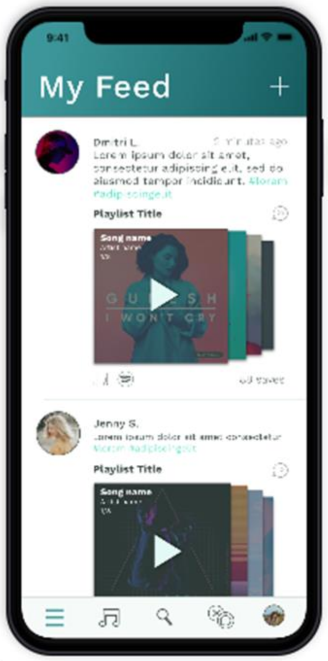


Check other users profile

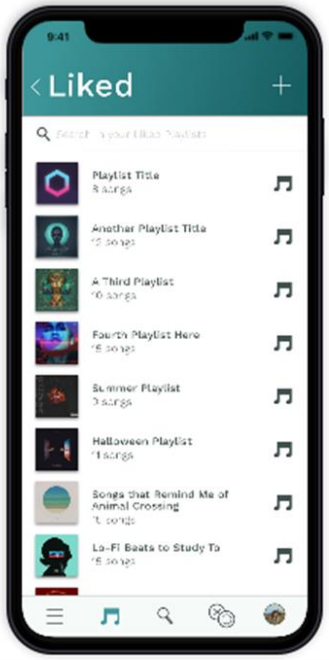
Following screens show how My messages, My feed and My liked playlist is displayed



My messages

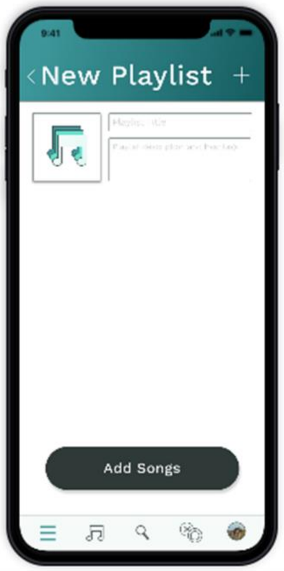


My Feed

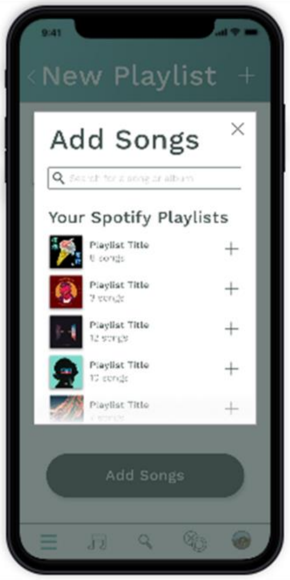


My Liked playlist

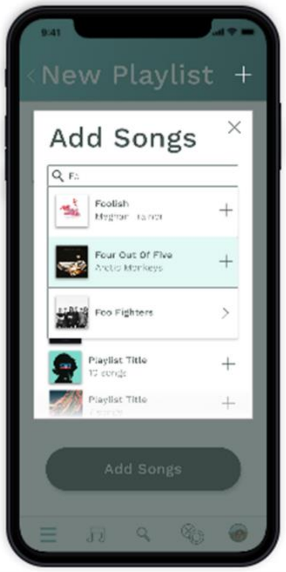
Following screens show how new playlist is created, how songs can be searched and added to the new playlist and display saved playlist



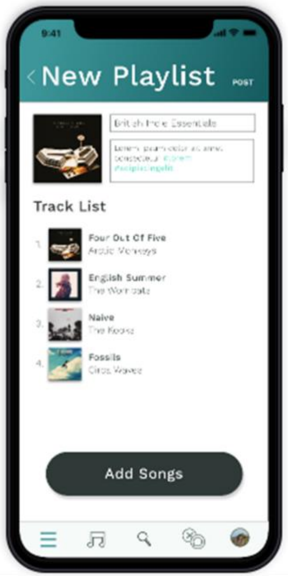
Create New playlist



Add songs to playlist

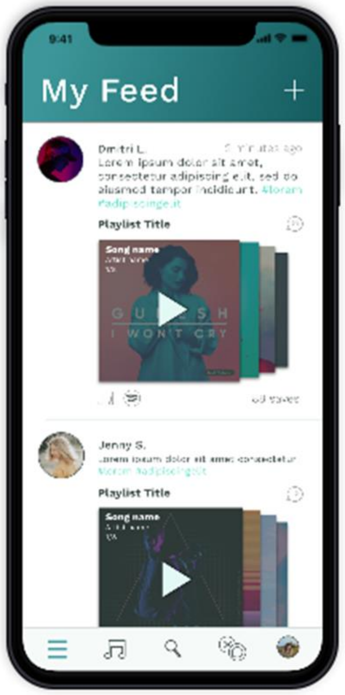


Add songs to playlist

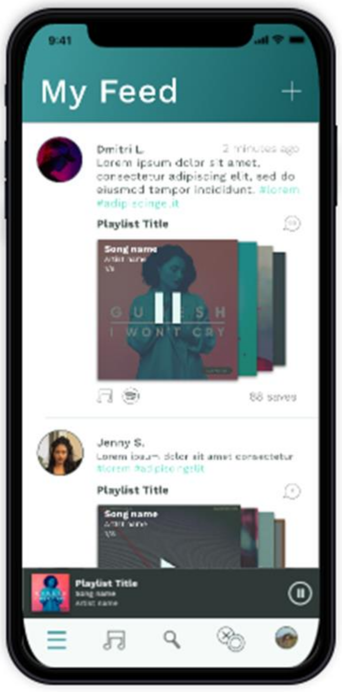


New playlist created

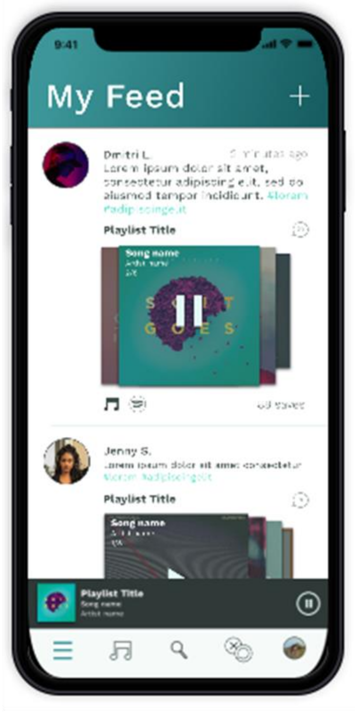
Following screens show how a song can be selected to Play, song being played can be paused and we can also display next song to be played.



Play song

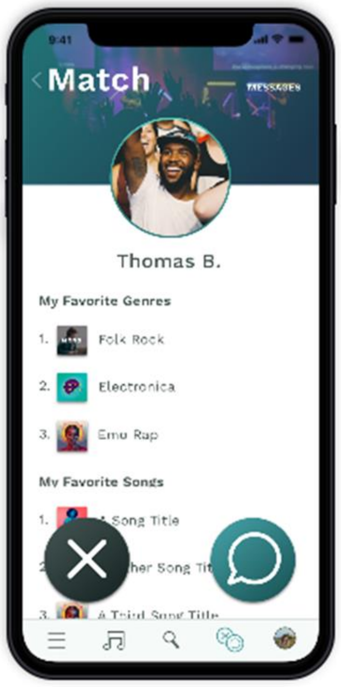


Playing song

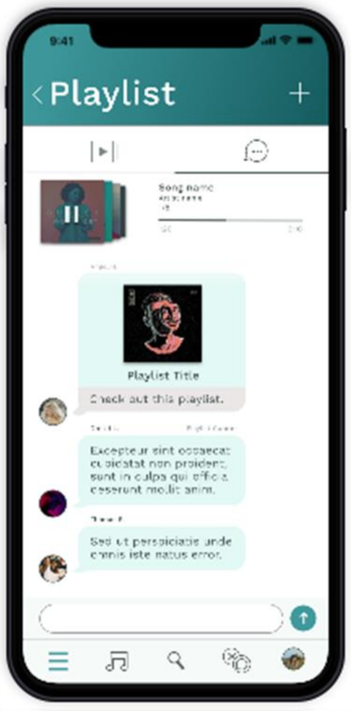


Next song

Screens below show how Tinder like functionality is supported, showing how a match is displayed and one can decide to accept or reject match. One can send chat messages thru the app.



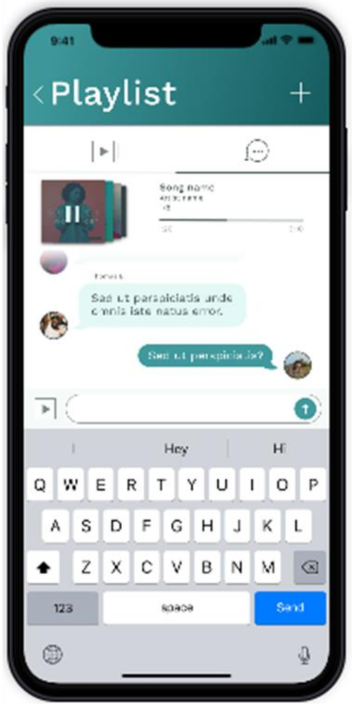
Music Tinder



Check out this playlist.

Sed ut perspiciatis unde omnis iste natus error.

Chat

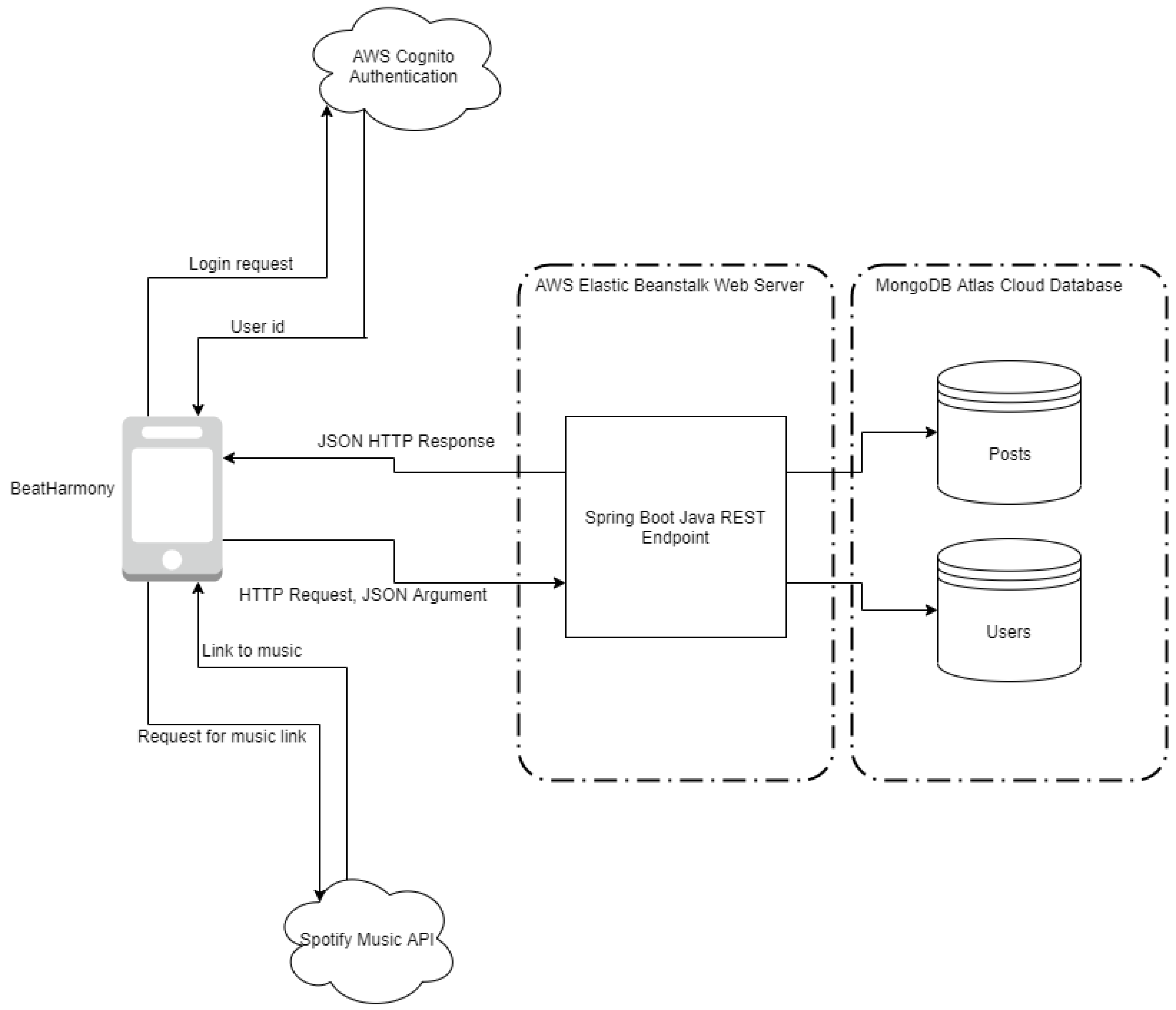


Sed ut perspiciatis unde omnis iste natus error.

Sed ut perspiciatis?

Chat send message

5.6 Updated Architecture Diagram


AWS Cognito Authentication
Login request
User id
AWS Elastic Beanstalk Web Server
MongoDB Atlas Cloud Database
JSON HTTP Response
BeatHarmony
Spring Boot Java REST Endpoint
Posts
Users
HTTP Request, JSON Argument
Link to music
Request for music link
Spotify Music API

5.7 Technology Decisions

We have decided on the following technologies for each of the components.

- **Frontend:**
  - React Native as it is platform neutral
  - Luckily, we have getstream.io (which helps with social networking functionality) has a free tier and React Native integrations
- **Backend**
  - A Spring Boot Java REST backend hosted on AWS Elastic Beanstalk
  - A MongoDB NoSQL database hosted on AWS and managed by MongoDB Atlas
  - Spotify Music API
  - AWS Cognito Authentication Service

Server

We decided to use AWS Elastic Beanstalk. It allows us to upload a Java Spring Boot project to it and run on Tomcat with no other config required. We will try using dynamoDB but will stick with MongoDB to start with. If we go graph-based, it wouldn't be that much of a change. Spring Data has neo4j functionality available if we want to use it. We have been using MongoDB with the Spring Data plugin, and it's so easy to set up. CRUD operations don't even require an implementation of the repository interface, just the interface itself. We will add more advanced queries as time goes on. As of now, we have got the web service layer on top of the data access layer. We have demoed that in Postman.

Finally backend is written with Java, assisted by Spring Boot, packaged by Maven and deployed on Tomcat.

Here are the features which led to the technology decisions

Representational state transfer (**REST**)

REST is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called *RESTful* Web services (RWS), provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. In a RESTful Web service, requests made to a resource's URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format. When HTTP is used, as is most common, the operations (HTTP methods) available are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE

AWS Elastic beanstalk

Applications deployed in the cloud need memory, computing power and an operating system to run. Creating and administering these items can take a lot of work and maintenance. AWS Elastic beanstalk can take a lot of the setup work out of development/deployment and can save developers and companies time and hassle. AWS Elastic Beanstalk is an orchestration service that abstracts away some of these hardware resources and details (e.g. setting up AWS components), while still allowing the developer a range of choices when it comes to OS and programming language.

AWS Elastic Beanstalk supports multiple languages, which, includes, but is not limited to, Java, PHP, .NET and Docker

Spring Boot

The Java Spring Boot framework provides a powerful set of tools for web development on both the front-end and back-end. It has built-in configuration for security and database access, as well as simple request mappings. This framework is highly configurable and makes web development extremely simple.

MongoDB

Mongo is quickly growing in popularity among NoSQL databases. Read more about NoSQL and its benefits here. MongoDB, specifically, is perfect for developing a REST API with Spring Boot for a couple of key reasons:

- Data is stored in MongoDB as JSON
- Spring Boot provides a powerful MongoDB connector

NoSQL

When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:

- Large volumes of rapidly changing structured, semi-structured, and unstructured data
- Agile sprints, quick schema iteration, and frequent code pushes
- Object-oriented programming that is easy to use and flexible
- Geographically distributed scale-out architecture instead of expensive, monolithic architecture

Maven is a build automation tool used primarily for Java projects**.** Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, there are several areas of concern that Maven attempts to deal with:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information – eg change log document, mailing list for project..
- Providing guidelines for best practices development - eg, specification, execution, and reporting of unit tests are part of the normal build cycle using Maven
- Allowing transparent migration to new Maven features

5.8 Design

Functionality Pillars

- FINDING Playlists
- LISTENING to Playlists
- CREATING Playlists
- SOCIALLY Driven

Services/Platforms to Research

- Spotify
- iHeartRadio
- Twitch
- YouTube Live Streams
    - i.e. LoFi Chill Beats to Study To
- Discord
    - Mee6 Plugin

5.8.1 Core App Loop

The core app loop is broken up into 3 segments: finding playlists, creating playlists, and listening to playlists. All features within the application are dedicated to fulfilling 1 (or more) of these core app segments.

- Listening should be supported by finding and creating.
- Creating should be supported by listening and finding.
- Finding should be supported by creating and listening..

Typical App Session

- Boot up the app.
- Login to your user profile.
- You click on the “Playlist” tab from the bottom UI bar.
    - Playlists are sorted by most recent.
- You click on your most recently played playlist.
    - A Punk Rock playlist that one of your friends curates.
- “All the Small Things” by Blink-182 is playing.
- Someone in the playlists chat mentions that they’re part of a great Blink-182 playlist. The user links the playlist in the chat.
- You click the playlist link and see that “I Miss You” is playing. You decide to hop over to that playlist.
- After several minutes of listening you decide to subscribe to the playlist.
- You see the user’s username who recommended the playlist. They’re one of the moderators of the playlist. You click their user profile.
- You are able to see their profile picture, username, followers, following, and their recent activity.
- You decide to browse their recent activity to see what playlists they’re apart of.
- In the process, you decide to follow them. Their activity now shows up in your feed.

5.8.2 Features

User Profile [feature]

User profiles are the core driving force behind the social aspect of this application. User profiles contain data specific to an individual user.

- User Data
    - Display Name
    - Username
    - Profile Picture
    - Bio
    - Followers List
    - Following List
    - Pinned Item
        - Song
        - Album
        - Artist
        - Playlist

- Recent Activity Feed
  - Feed of created/managed playlists
    - Display the current song playing & last message sent in this playlist server.
  - Feed of followed playlists
    - Display the current song playing & last message sent in this playlist server.

Playlist Servers [feature]
Playlists are privately curated, socially driven *radio* communities designed for users to interact with and contribute to.

- All playlists are public, unless set to private (invite only).
- Playlists have owners and moderators (assigned by the owner).
  - Users can see the owner of the playlist, and access their profile.
  - Owners can create moderators that can moderate chat, and add/remove music from the playlist
  - Owners/mods can name the server.
- Playlists can contain songs from all platforms.
  - Spotify
  - Soundcloud
  - Youtube Music
- Playlists have perpetual chat rooms.
  - Users can link things in chat.
    - Songs
    - Other Playlists
    - Other Users
  - Users can recommend songs to be added to the playlist.
    - This populates a list of recommend songs that the owner/mods can see.
- Users can subscribe to a playlist.

Social Feed Tab [page]
Pillar: FINDING Playlists
Pillar: LISTENING to Playlists
Pillar: SOCIALLY Driven

- See what your friends are doing.
  - What they're listening to.
  - What they subscribe to.
  - Who they follow.

Trending Tab [page]
Pillar: FINDING Playlists
Pillar: LISTENING to Playlists
Pillar: SOCIALLY Driven

- See what people who aren't your friends are doing.

        - What they're listening to.
        - What they're subscribing to.
        - Who they're following.

Playlist Tab [page]
Pillar: LISTENING to Playlists
Pillar: CREATING Playlists

- Do things yourself.
    - Show subscribed playlist servers.
    - Find a list of private liked songs.
    - Create a playlist.

Search Tab [page]
Pillar: FINDING Playlists
Pillar: LISTENING to Playlists

- Find specific playlists, users, or songs.
    - Generates a list that can be filtered by users, playlists, or songs.
    - Searching for a song can display popular playlists that this song is associated with.

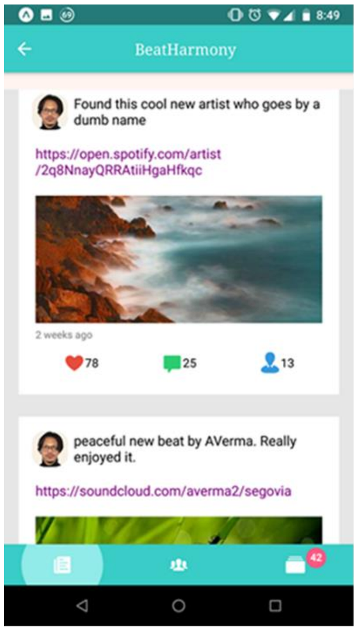
Profile Tab [page]
Pillar: SOCIALLY Driven

- View your profile page.
    - Display Name
    - Username
    - Profile Picture
    - Bio
    - Followers List
    - Following List
    - Pinned Item
        - Song
        - Album
        - Artist
        - Playlist
    - Recent Activity Feed
- Update your personal / account info.

6 Front End details

This section displays the current status of front end.

Existing posts with a URL link and an image can be displayed in a feed. URLs are also clickable and will take you to the song/playlist/artist in the link. Like/comment is dummy and not functional yet.


BeatHarmony

Found this cool new artist who goes by a dumb name

https://open.spotify.com/artist/2q8NnayQRRAtiiHgaHfkqc

2 weeks ago

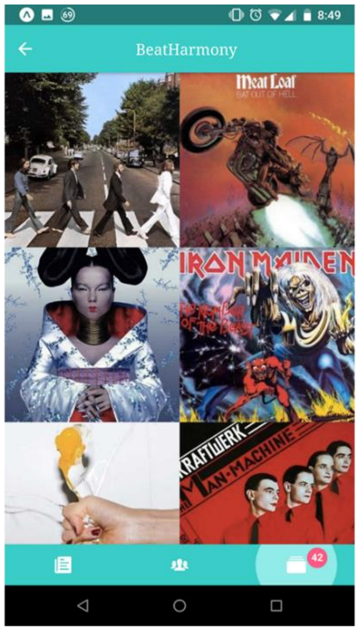
78 25 13

peaceful new beat by AVerma. Really enjoyed it.

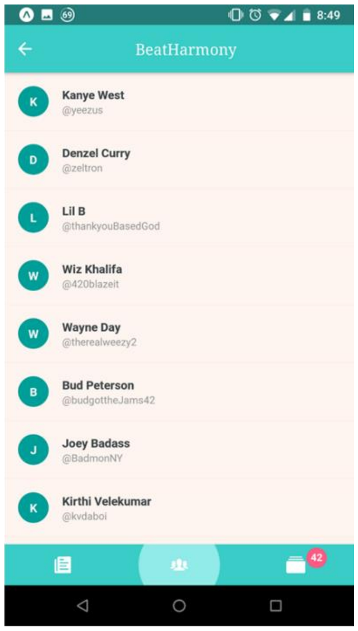
https://soundcloud.com/averma2/segovia

42

My Feed



My Saved playlist


BeatHarmony

K	Kanye West @yeezus
D	Denzel Curry @zeltron
L	Lil B @thankyouBasedGod
W	Wiz Khalifa @420blazeit
W	Wayne Day @therealweezy2
B	Bud Peterson @budgottheJams42
J	Joey Badass @BadmonNY
K	Kirthi Velekumar @kvdaboi

42

My Trusted friends

7 Backend Details

Sample backend is created. It has a sample UserService and UserController so everyone can see what the backend will be like programmatically.

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. Master has working Java Spring Boot REST API with a dummy database, for demonstration purposes. The MongoDB implementation has already been merged back to master.

Here are the details of master. It starts with a package description, and below that, a class description.

Master Details

- 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:

- Install an IDE of your choice, (eg IntelliJ)
- Install MongoDB. (see in Google)
- 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

8 Data Stores

Collections we need:

- 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

9 Concerns

MongoDB will be great for tasks like storing post info and user info

- We may run into issues with mongo once we need to use relationship information between posts and users (but we can cross that bridge when we get there).

- We still think a graph database would be optimal for a social network, but since graph databases are newer and less popular, we may find less people to help us with our problems, and so MongoDB is a good starting point.

- If we ever run into issues managing mongo, but want to continue with a noSQL datastore, we can consider dynamoDB by Amazon. It is similar to mongo, and we think it would integrate well with the rest of the AWS services we will be using.

Conclusion: We will continue with mongo, but stay aware of possible issues and don't get complacent. We are thinking we continue with MongoDB for now, unless we decided to go graph-based, which wouldn't be that much of a change. Spring Data has neo4j functionality available if we want to use it.

Graph stores are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.

10 Appendix 1: Interview Protocol

Introduction

1. How are you doing today?
2. Study introduction
    a. We are working on a project that aims to improve the music discovery process. Thank you so much for taking the time to talk with me. Data from this conversation will not be used outside the scope of this project and all recordings and other potentially identifiable data will be deleted. This interview will be entirely confidential. Is it okay if I audio record you and record your screen? Can you please share your screen with me?
    b. We'll start off by asking you a few questions about you.
        i. Can you tell me a bit about the role of music in your life currently?
        ii. What platforms do you currently use to discover new music?
            1. Why do you use these platforms?
            2. How often do you find a song that you like enough to add to your music library?
        iii. What is your favorite part of the music discovery process?
        iv. What is your least favorite part of the music discovery process?

Great! Thank you so much! Now we will introduce you to the application we have designed.
We're working on a music discovery application that allows you to discover new playlists from users that you add as "trusted curators". This is a feed-based application much like social media. You can listen to playlists, save playlists, discuss playlists with other users, and share your own playlists. You can also export playlists to Spotify. There is also a matching feature win which here you can match with a user with similar music taste and share a song with them.

For the sake of this study, we will be using an online prototype. We would like to remind you that we are not evaluating you by any means. It is the system that we are evaluating. Feel free to criticize it or appreciate it. Do you have any questions?

Great! Let's get started.

First, go ahead and scroll through the feed. You may use the system as if you were using it in real life. You can try to interact with the playlist itself in the first post.

We would appreciate if you could answer some follow up questions now about the task you just performed.

1. How was the process of scrolling through the feed?
    a. Walk me through what you're seeing here.
    b. What do you think of how the posts are structured?
    c. Go ahead and click on the play button.
        i. What did you expect clicking the play button to do?

    d. Try to save the song.
2. What do you feel about how you interact with the system while performing the task?
3. Are there any other thoughts you have on this page?

Next, go ahead and click on the description of the first post.

1. What did you think would happen?
2. Walk me through what you see on this page.
3. How do you feel about the information that is displayed here?
4. What do you think you will see if you click on the second tab?
5. What do you feel about how you interact with the system while performing the task?
6. Are there any other thoughts you have on this page?

Go ahead and click on the chat.

1. Walk me through what you see here.
2. Are there any other thoughts you have on this page?

Go ahead and create a new playlist.

1. Walk me through what you're seeing.
2. How was that process for you?
    a. What was easy?
    b. What was difficult?
3. Are there any other thoughts you have on this page?

Great. Now you can go ahead and look at your saved playlists.

1. What do you think about the information that is being displayed on this page?
2. Are there any other thoughts you have on this page?

Go ahead and use search.

1. Walk me through what you're seeing.
2. How do you feel about the information that is being displayed here?
3. Go ahead and click the search bar.
4. How was this process?
5. Are there any other thoughts you have on this page?

Go ahead and click on the matching service. [Explain matching concept again]

1. How do you feel about the matching concept?
    a. Do you think you would or wouldn't use this part of the application if it existed in real life?
2. What do you think about the information that was displayed on this page?
3. How do you feel about this information in terms of whether or not you would want to match with someone?
4. Are there any other thoughts you have on this page?

Click on the messages section.

1. Walk me through what you're seeing here.
2. How many unread messages are on this screen?
    a. What information led you to believe that this message was unread?
3. Are there any other thoughts you have on this page?

Navigate to your profile. Take a look at the information on this page.

1. How do you feel about the types of information that are displayed on your profile?
    a. Do you feel that this type of information is representative of your music taste?
    b. Are there any other thoughts you have on this page?
2. View your notifications.
3. How do you feel about this profile section in general?
4. Click on trusted curators.
    a. How do you feel about how this info is displayed?
    b. Are there any other thoughts you have on this page?

5. Go ahead and look at the first person's profile on this list.
    a. What did you think would happen?
    b. Tell me what you're looking at now.
6. What would lead you to add someone as a trusted curator?

Awesome! We have a few more questions about the overall system

1. What are your overall impressions of the system?
2. What do you like most about the design concept?
3. What do you like least about the design concept?
4. If we gave you a magic wand and you could make one significant change to the design concept, what change would you make?
5. How do you feel about the visuals of the system?
6. How does this compare to other streaming services you use?
7. How often do you see yourself using this system in real life?
8. Do you have any other questions or comments about this concept or your experiences with it?