AUX

Van Brantley
Advisor: Robert Dondero

Introduction

While many people may not realize it, listening to music is a very social activity. Whether it be at a party, or a small gathering of friends, there is often music playing in the background, setting the mood for the event. However due to a lack of a prominent online platform for music sharing, this social activity is mostly limited to in-person encounters. Despite there being social platforms for seemingly everything today, surprisingly there is no well-known online social hub for music. Yet there are countless people eager to share the music they've taken the time to discover, and even more people yearning to be introduced to new music to diversify and expand their music libraries. The problem that my project is addressing is that there is no place for people to fully display their taste in music and explore those of others. The goal of my project is to create a platform named AUX where people can display and recommend their favorite music to others in a way that is easy to digest for the recommendation recipients. Ultimately, I want to build community through music, where people can discover new music that they will love from one another. As pretty much everyone listens to music, this project has the potential to affect many people by introducing new music to them and broadening their musical horizons.

Problem Background and Related Work

There are other social music applications out there on the App Store which are products targeted at the same goal as mine. The two apps that I found were Bopdrop and Revo. They are both slightly different from one another, and also different from the application that I plan to make. Bopdrop operates on the song level, where users post individual songs that they recommend to their friends. Revo, on the other hand, operates mainly on the playlist level, where users create a playlist in their music streaming app (either Spotify or Apple Music) and display them on their profile. Revo also has a "Moments" feature where users can post a song snippet that is played over a video that they record.

As Revo's "Moments" functionality can be achieved using the vastly more popular social media platform Instagram, and Spotify and Apple Music users are able to view their friends' playlists in those apps, Revo offers its users very little value that is unique to their app. While Bopdrop does succeed in providing users utility that streaming platforms fail to provide, the fact that they operate on the individual song level limits this utility. When recommending a singular song to your friends, the expected provided value to them is quite small. Either a recipient of a recommendation does not like the song, and they are provided no value, or they do like the song, and they get the value that one new song provides them. If they have never heard the song or artist before, it is likely that they will want to check out more songs by that artist. In this case, they would need to go to their music streaming app and take the time to look through their discography themselves.

Approach

My approach to achieving my stated goal of creating a platform where people can display and recommend their favorite music to others will be focused one step above the song, the album. By doing so, I will be able to maximize the value that a recommendation can provide to a recipient. Instead of posting a singular song for their friends to see, users on AUX will post album reviews. When making an album review, a user will be shown the full track list for the album, which they will narrow down to only the tracks that they like on the album. Once a user has this list, they will be able to rearrange it in the order of their favorites. By doing this, a user is essentially able to create their own personalized package of songs from the album that they believe will have the best chance of turning their friends into fans of the artist.

On the receiving end of a recommendation, users will be able to go through their friend's curated list and listen to 30 second previews of each song. This way they can get a feeling for the song without having to spend the time listening to the entire track. By comparing an example of an exchange on AUX versus the one described above for Bopdrop, we can see that a recommendation holds much more expected value to a recipient in my application.

In my application there are three possible scenarios that can occur when one receives a recommendation. They can either like none of the songs in the list, and they are provided no value, or they can like some of the songs in the list, and they get the value provided to them by those few songs. What is different about my app is that it has the potential to provide a recipient with extreme value. The last scenario is that they love all or most songs in the list, and can now confidently consider themselves as a fan of the artist of the album. They now know that they

should take the time to listen to all of their previous works before this album and all of their future work. And they know this all because they listen to a handful of 30 second previews ordered by their friend.

Implementation

Technology Stack

To implement this project, I used a pure JavaScript technology stack. For my database, I used MongoDB, which strays away from the conventional database structure of using relational tables to hold data, and instead stores collections of JSON objects. For my application's backend, I used Node.js and Express.js. Node is a JavaScript runtime environment that enables JavaScript to be run on a machine as opposed to the browser. Express.js is a framework built on top of Node that uses Node functions and structures to enable API construction. For my application's frontend, I used Facebook's UI library React.js. These three stack layers worked together nicely since they are all JavaScript frameworks, which enabled data to be seamlessly transferred as JSON objects from one layer to another.

Progress Steps

The progress that I made throughout the semester can be broken up into four sections; planning, prototype, MVP, and stretch goals.

First, I went through a planning and preparation stage where I outlined the steps for my project. This involved identifying my minimum viable product (MVP), which is the simplest version of the application that could provide benefit to users. I decided that my MVP would be a system

that allowed users to post album recommendations, look up other users of the system, view their album recommendations, and save songs to Spotify. In addition to outlining my MVP in the planning stage, I also identified some stretch goals, which are functionalities that could be implemented after successfully completing the MVP. Some of the stretch goals that I came up with were being able to friend other users, and having a feed which consists of all of the recommendations that your friends have posted. The last thing that I did during the planning stage was make pencil and paper drawings of the different screens that I wanted the application to have. By showing these drawings to future users of the system, I was able to lock down the user interface design.

Once I had my MVP and stretch goals mapped out, I went about transforming the plan I made into reality. I needed to make a prototype that connected all layers of the technology stack together and was hosted on the internet. This prototype would need to access data stored in my database, and provide the data to my application's frontend so that it can be displayed to the user. First I had to get the user connected to Spotify. Luckily Spotify made this process easy, as they use OAuth 2.0. Under this authorization flow, users are redirected to a Spotify sign in page, and once they supply their credentials, they are redirected back to my application with an access token (Spotify for Developers). This access token is unique to the user, and must be supplied to any calls to Spotify's API to retrieve data from their catalog. To get my database involved, I started by hard-coding some album recommendations into my database. Upon pressing a button in my application's frontend, a function call was triggered in my application's middle layer, which facilitated the retrieval of the data from my database, and this data was provided back to my frontend to be displayed to the user. By completing this prototype, I had a skinny logic path through all of the layers of my application, which gave me a foundation upon which to build.

From this prototype, I was able to reach my MVP. A big step in accomplishing this was being able to post data from my middle layer to my database so that data did not have to be hard-coded. Thanks to MongoDB's extensive function library and documentation, this task was much more straightforward than I was originally anticipating. In order to create my MVP, I needed to implement two searches, an album search and a user search. My album search took an input from the user and returned the album's in Spotify's catalog that matched their query. The user search simply queried my users collection in my database and returned the users whose usernames matched the given input.

Since completing the MVP I have been able to implement a few of my stretch goals, namely friending and feed. While I did not get around to completing all of the goals that I laid out in the beginning of the semester, I ended up implementing additional features that I had not yet conceived in September. For example, users can add albums that they want to review in the future to their "To Review" list, and can store their friends' recommendations in their "To Listen" list for future inspection. I plan to continue implementing features for the website in the spring semester and beyond.

Challenges

Throughout the course of developing my project, I ran into many challenges. Two of the biggest that I faced occurred when trying to host the website on the internet, and having multiple users signed into the system simultaneously.

In attempting to host my project on the internet using the hosting service Heroku, I ran into the issue of hosting a two server application. By using React and Express for my frontend and backend respectively, each ran on their own server. However, to host a project on Heroku, it must be a single server. In order to solve this problem, I was able to have my Express backend serve my React frontend by using something called a Heroku postbuild. The postbuild runs a script which retrieves the frontend project. Thus, I was able to point Heroku to the backend, which then fetches and provides the frontend along with itself.

Another issue that I came across in my development was the occurrence of a global user. If multiple users were accessing the site at the same time, all users would see the data of the last user to sign into the system. This problem was due to the fact that I was using a local database in my backend to store the user's information once they signed into Spotify. Whenever a new user would log in, their information would overwrite the previous user's information and would be saved instead. To solve this issue, I replaced my database with session variables to store user information once they signed in. Session variables are used to store information that corresponds to a specific browser instance, which is exactly what I needed.

Evaluation

To test my system, I went through a peer evaluation process. I created a list of tasks to be performed on my system, and observed users going through the tasks over a Zoom call as they shared their screen. I encouraged users to talk aloud throughout the process, so that I could get a sense for what they were thinking, their likes, and dislikes for the system. The list of tasks and notes from each evaluation session can be found in the appendix at the end of my report.

In general, the evaluation was a success. Although the steps in my task list were quite brief and did not offer much detail as to how to perform the tasks, my evaluators were able to perform them with ease. I attribute this success to the explanatory messages within my system that point users in the right direction. The biggest takeaway from my three evaluation sessions was that the website could use some stylistic updates. It was brought to my attention that some items were not centered in the page, and certain elements were not visually appealing. I was also given a myriad of future feature suggestions from my evaluators, which was great to see. It appears that users would enjoy a "Like" feature for reviews as well as the ability to comment on other users' reviews. All of my evaluators said that they were impressed with the application, and would use the system in the future to share and recommend music.

Summary

Conclusions

Throughout my time developing this application, I certainly learned a lot about web development. I had always wondered what it would take to build a social media application, and to my surprise, it was not as daunting as I had originally anticipated. I learned a lot about JavaScript specifically, which is something that I am extremely grateful for. While it was intimidating going with a pure JavaScript stack with little prior knowledge about the language, I am glad that I stuck with it. Typically, there is a lot of tedious work that has to be done in order to put data in and out of JSON format when using other languages, but with JavaScript, this work is not necessary!

Being able to share the website with my friends and have it facilitate new music discovery is something that brings me great joy. I am very grateful that I had the opportunity to develop this application for my senior independent work.

Future Work

This is a project that I am very passionate about, and I plan on continuing to enhance it for years to come. Next semester I hope to conduct another session of independent work where I shift my focus to making an iPhone application that incorporates both Spotify and Apple Music. I also want to expand the website to have album and artist pages, where users can see how other users of the system collectively rank songs. I think it would also be interesting to incorporate machine learning into the project to gauge user's music tastes to better recommend music to them.

Acknowledgements

I would like to extend a special thanks to my advisor, Professor Robert Dondero. His COS333 course taught me the fundamentals of web development, and I am glad that I got to expand upon the foundation of my knowledge under his guidance. Professor Dondero did a great job of keeping me on track, and never hesitated to let me know when I needed to pick up the pace with my development. Without Professor Dondero's direction, AUX would not be the website that you see today; it would likely be a mere thought in my brain.

Works Cited

"Authorization Guide." *Spotify for Developers*, developer.spotify.com/documentation/general/guides/authorization-guide/.

AUX User's Guide

AUX is a website for Spotify users to share and discover music. Users can publish reviews of their favorite albums to showcase their musical tastes, and easily digest the recommendations of their friends to expand their musical horizons.

Below are some use cases for the application and the steps required to complete them.

Log in

In order to see what the website has to offer, you first have to sign in with your Spotify account.

Note: This application is best suited for viewing on your desktop in a full browser window using Google Chrome.

- Navigate to https://aux-social-music.herokuapp.com
- Click the purple "Log In" button
- Enter Spotify credentials and press the green "Log In" button

Create an AUX ID

Create a unique username so that you can be identified and your friends can find you.

- Type a username that contains no spaces in the text box
- Press the blue "Submit" button to the left of the text box
- Observe that you are directed to your profile page with your username displayed on the top left of the screen

Post a review

Share an album that you love with your friends. Make a list of the songs you like on the album in order of your favorite so that your friends can enjoy it too.

- Click on the "Album Search" button in the navigation bar
- Type in the name of your favorite album in the search bar
- Click on the name of the album in the search results
- Add the songs that you like on the album to "My List" by clicking the right arrow buttons in their respective rows
- Reorder songs in order of your favorite by pressing the up and down arrows for the songs in "My List"
- Give the album a score from 0-10 by typing in the text field to the right of the word "Rating"

- Press the green “Publish” button

Edit a review

Turns out you changed your mind about your review. You want to update it so your friends get the picture for the album.

- Click on the album art of the album that you just published a review for
- Add one to your score by selecting the up arrow in text box to the right of the word “Rating”
- Move a song down one slot in “My List” by pressing the down arrow in its row
- Press the green “Update” button under the word “Rating”
- Click on the album art of the album on your Profile page
- Observe the rating has changed and the song ordering has been changed in “My List”

Delete a review

Turns out you don’t really like the album that you just made a review for and you don’t want it to be present on your profile anymore.

- Click on the album art of the album that you just edited your review for
- Press the red “Delete” button below the green “Update” button
- Observe that the album art no longer appears on your profile page

Follow a user

What good is a social music application without any friends? Follow a user so that you can easily see the reviews that they post.

- Press the “User Search” button in the navigation bar
- Type “van.brantley” into the search bar
- Click on the search result “van.brantley”
- Press the green “Follow” button
- Observe that the “Follow” button changes to “Unfollow” and the user’s Followers count increased by one

View a review

Let’s see what music your friends are listening to.

- Click on the “Profile” tab in the navigation bar
- Click the “Following” button in the upper right part of the screen
- Notice that the user “van.brantley” is listed
- Select on the name “van.brantley”
- Click on the top left album cover on their profile page
- Observe that the songs that the user selected are displayed under “van.brantley’s Picks”

Preview a song

A song’s name can only tell you so much. Listen to a preview of a song to tell whether or not you like it.

- Click the play button (sideways triangle) for the first album track to listen to a 30 second preview
- Exit the tab that was opened to play the preview

Save to Spotify

You like some songs that you’ve discovered in one of your friend’s review. You want a way to save the songs for future listening.

- Press the right arrow button for the first 3 songs under “van.brantley’s Picks” to move them to the “Songs to Save” list
- Press the green “Save to Spotify” button
- Open a new tab in your browser
- Navigate to https://open.spotify.com
- Click on the “Liked Songs” playlist
- Notice that the selected songs have been added to the playlist

View your feed

You want to see all of the reviews that your friends have been posting.

- Click the “Feed” tab in the navigation bar
- Observe that the user van.brantley’s reviews are displayed

Add a review to your To Listen list

You see your friends posting reviews that you are interested in checking out, but you don’t have the time to listen to them now. You want to store them someplace to listen to them later.

- Click the name of the third album result in your feed
- Press the light blue “Listen Later” button under the green “Save to Spotify” button
- Click the “To Listen” tab in the navigation bar
- Observe that the added review appears as a result in the list
- Click on the name of the album
- Notice that you are taken back to the user’s review for the album
- Press the back page button in your browser to return to your To Listen list

Remove a review from your To Listen list

You’ve listened to the review that you stored in your To Listen list and no longer need to save it.

- Press the red X button to the right of the artist name
- Notice that the album is removed from the list

Add an album to your To Review list

You have an album that you know your friends will love, but you don’t have the time to post a review for it now. You want to store the album someplace so you do not forget to review it.

- Click the “Album Search” tab in the navigation bar
- Type “lonerism” into the search bar
- Click on the title “Lonerism” in the first search result
- Click the grey “Review Later” button
- Click the “To Review” tab in the navigation bar
- Click the album name “Lonerism”
- Notice that you are taken to the page to create a review for this album
- Press the back page button in your browser to return to your To Review list

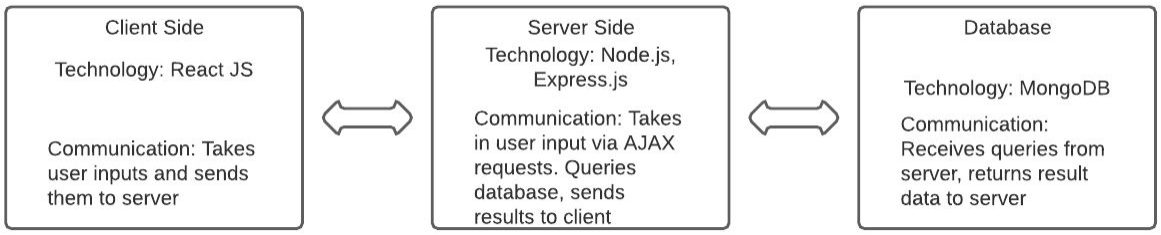
Remove an album from your To Review list

You’ve posted a review for the album in your To Review list or you simply do not want to review it anymore. You no longer need to save it.

- Press the red X button to the right of the artist name
- Notice that the album is removed from the list

AUX Programmer's Guide

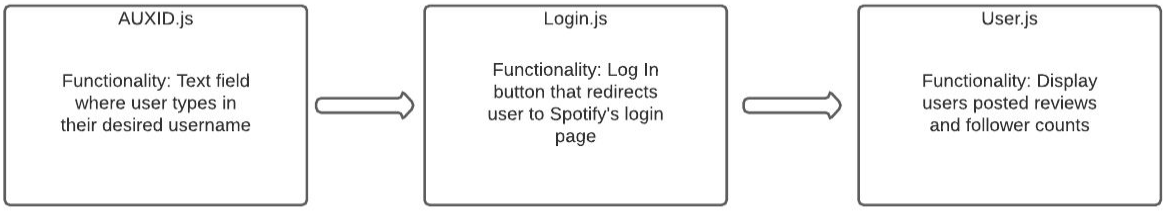
Top Level



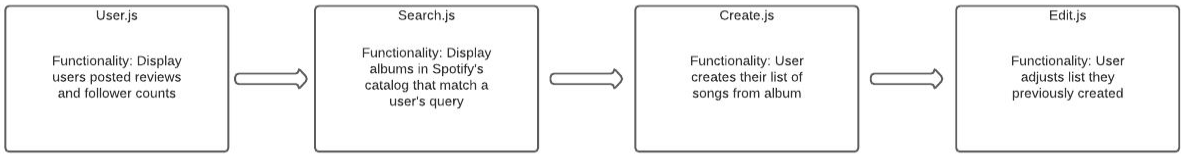
Second Level

Client Side

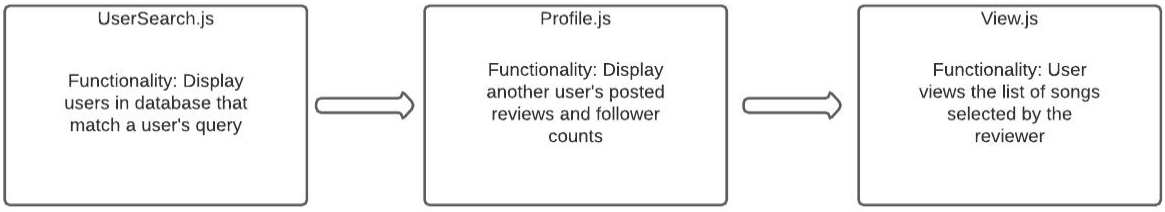
Log In Progression



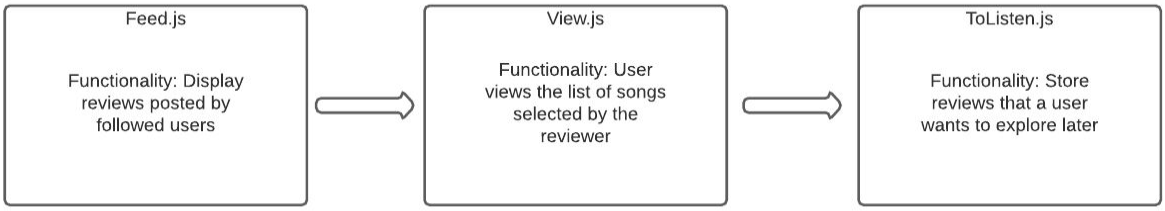
Create Review Progression



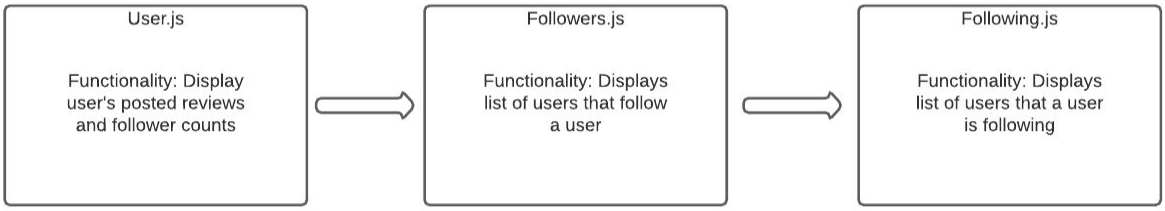
View Review Progression



View Feed Progression



View Follow Lists Progression



Server Side

ENDPOINT	METHOD	DESCRIPTION
/login	GET	Authorize user by having them sign into their Spotify account
/checkID/:auxID	GET	Check to see if a user's desired username is already taken by another user
/api/auxID	POST	Enter a user into the user's collection in the database
/callback	GET	Set user identifying session variables after logging into Spotify and redirect user to profile page
/checkCredentials	GET	Check to make sure that a user is currently logged into Spotify and has an access token that is not expired
/tracks/:albumID	GET	Retrieve an album's tracks
/search/:query	GET	Search Spotify's album catalog given a query
/usersearch/:auxID	GET	Search user's collection in database for users matching a given query
/list/:auxID/:albumID	GET	Retrieve a user's review for a specified album
/reviews	GET	Retrieve all of a user's reviews
/reviews/:auxID	GET	Retrieve all reviews posted by a specified user
/addReview	POST	Add a review to the lists collection in the database
/deleteReview	POST	Delete a review from the lists collection in the database
/updateReview	POST	Update a review in the lists collection in the database
/toListen	GET	Retrieve a user's To Listen list from the users collection in the database
/addToListen	POST	Add a review to a user's To Listen list

/removeToListen	POST	Remove a review from a user's To Listen list
/toReview	GET	Retrieve a user's To Review list from the users collection in the database
/addToReview	POST	Add an album to a user's To Review list

/removeToReview	POST	Remove an album from a user's To Review list
/saveTracks	PUT	Save tracks to a user's Liked Songs playlist in Spotify
/followUser	POST	Follow a user
/unfollowUser	POST	Unfollow a user
/checkFriendship/:user1/:user 2	GET	Check to see if a user follows another user to determine appearance of friend button
/getUser/:auxID	GET	Retrieve a user's data from the users collection in the database
/getFollowers/:auxID	GET	Retrieve the list of users that follow a user
/getFollowing/:auxID	GET	Retrieve the list of users that a user follows
/feed	GET	Retrieve all reviews posted by users that a user follows

Database

Collections: lists, sessions, users

Schemas

- Lists - stores all reviews posted on the system

```
{
    _id: ObjectId,              (Unique ID assigned by Mongo)
    auxID: String,              (AUXID of the reviewer)
    album_id: String,           (Album's Spotify ID)
    album_name: String,         (Album's name)
    album_artist: String,       (Album's artist)
    album_art: String,          (URL of Album's artwork)
    num_album_tracks: Int,      (Number of tracks on the album)
    score: Double,              (Score that user gave the album)
    percentage: Double,         (Percentage of songs included in user's list)
    user_list: Array            (Array of tracks that user included in their list)
        {
            title: String,          (Title of the track)
            id: String,             (Track's Spotify ID)
            track_number: Int,      (Track's index on the album)
            preview_url: String,    (URL of Spotify's 30 second track preview)
            index: Int              (Track's index in user's list)
        }
}
```

- Sessions - stores the active session variables for users

```
{
    _id: ObjectId,              (Unique ID assigned by Mongo)
    expires: Date,              (Date that the session expires)
    session: {
            spotifyID: String,      (Spotify ID of the signed in user)
            accessToken: String,    (Spotify access token used for API requests)
            auxID: String,          (AUXID of the signed in user)
    }

}
```

- Users - stores user information

```
{
    _id: String,                    (Unique ID assigned by Mongo)
    auxID: String,                  (AUXID of the user)
    spotifyID: String,              (Spotify ID of the user)
    following: Array
        {
            auxID: String,          (AUXID of the user being followed)
        },
    followers: Array
        {
            auxID: String,          (AUXID of the user that is following)
        },
    to_listen: Array
        {
            name: String,           (Name of the album)
            artist: String,         (Artist of the album)
            art: String,            (URL of album artwork)
            albumID: String,        (Spotify ID of the album)
            reviewer: String,       (AUXID of the reviewer)
        },
    to_review: Array
        {
            name: String,           (Name of the album)
            artist: String,         (Artist of the album)
            art: String,            (URL of album artwork)
            albumID: String,        (Spotify ID of the album)
            num_tracks: Int,        (Number of tracks on the album)
            Release: String         (Year that the album was released)
        },

}
```

Appendix

Task List

- Log in
- Create an AUX ID
- Publish a review
- Edit your review
- Follow the user “van.brantley”
- View Van’s most recent review
- Preview a song
- Save songs to Spotify
- View your feed
- Add a user’s review to your “To Listen” list
- Remove a review from your “To Listen” list
- Add an album to your “To Review” list
- Remove an album from your “To Review” list
- View your Followers list
- View your Following list

Aaron Lichtblau Evaluation Session Notes

- Login page looks clean, straight forward
- Create a username message doesn’t need an exclamation mark
- Publish first review message doesn’t need exclamation mark
- Message doesn’t need comma - “Click the Album search tab to create your first review”
- Search result options make sense, intuitive
- Rating should be in line with Publish and Review Later buttons
- Follow button changed, can tell you’re now following the user
- Likes the blank screen when previewing a song
- Wasn’t sure at first how to Save Songs to Spotify, then saw the “Songs to Save” title of the blank list
- Not exactly sure what save to Spotify means - does it create a new playlist in Spotify with those songs in it?
- About the redirection after saving songs to spotify - some sort of message is needed
- Likes the idea of message area displaying message for a couple of seconds
- When viewing another user’s review, there should be a link to their profile somewhere (could go under the artist’s name, make sure it’s blue/looks like a link)
    - The title of the album should be a link to that album’s create page

- Was expecting to see his own reviews in the feed like you would on facebook/instagram
- Annoyed that the names of the albums and the word “Feed” aren’t aligned
- Feed table could have headers (user, name, album, artist)
- Navbar should be sticky
- Buttons being Listen Later/Review Later and lists being called To Listen and To Review makes sense
- Followers/Following pages could use some touching up on the interface design, result bars are really long

Brian Young Evaluation Session Notes

- Cut down on the number of scopes, people could be wary of the amount of control you have, and you don’t actually use all of them
- Very easy set up, likes that there is no email confirmation
- Went to publish without giving the album a score
- Should have a popup that prevents user from publishing if they haven’t changed the Rating
- Kept pressing the up arrow thinking that focus would stay with the same song
- Impressed with the ability to save songs to spotify
- Should have an add to a specific playlist feature instead of just save to Liked Songs playlist
- Asked if feed was chronological, followed another user to see if the reviews mixed in -- they did
- Could have one album review in focus at a time in the feed, scroll through the recommendations
- Likes the title is highlighted in blue in the feed
- Likes the pop up message that comes up when adding to “To Listen” list
- Likes the message that pops up once nothing is in the To Listen list anymore
- Can see himself using this, good way to keep up with what people are listening to
- Doesn’t use the Spotify created explore playlists
- Would like to see likes, comments on reviews

Caleb Hart-Ruderman Evaluation Session Notes

- Assumes the rating is out of 10
- Figured out that the right arrow moves to my list
- Added all of the songs to his list, didn’t think to just do the ones he likes
- Knew to press the play button to preview
- Assumed following lists were on the profile page
- Put the score as a column in the feed table

- Put Following/Followers lists in the navbar
- Explanation/abstract for the review
- Click on the top left AUX button in navbar should take you to the home page, shouldn't make you log in again
- "Select all" button for Save To Spotify
- Upvote/Downvote or Likes/Fire for viewing a review
- Comment on reviews
- Links to user profile pages in Feed and View pages
- Likes the idea of the Comparison Dashboard for albums that you've both reviewed
- Album pages that rank songs based on number of times included, and for each song show the percentage of people that included it in their list
- Save songs to specific playlists that they have in Spotify, not just Liked Songs
- Being able to click on the whole row in feed, instead of just the album's name
- If you've already viewed an album, there's a little indication of that next to the album
- Recommend an album to one of your friends directly, gets sent to their inbox
- If user posts review for album in their To Review it should be removed