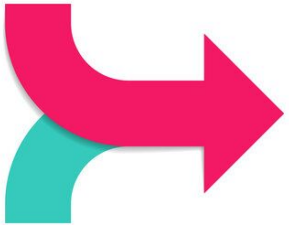
Music-Merge

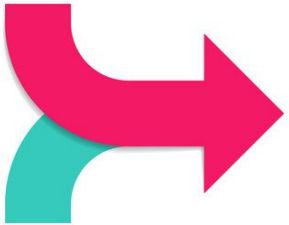
By: Aayush, Danny, Eli, Kelly, Noah, Toby

What is Music Merge ?



- With this project, we wanted to implement the ability to merge two different playlists into one playlist.
- This feature will become even more powerful when it’s used collaboratively.
  - Friends and families could make a collaborative playlist.
  - Not having to add every song manually, to create a playlist.
- Our vision is to revolutionize the way people curate and enjoy their playlists and have something of their own that expresses both parties. Our goal is to seamlessly merge their playlists, and have a dynamic and personalized music experience, with a click of a button.

Why is This Important?



- In spotify, a user can create multiple playlists, but what if they wanted to combine those playlists into a single playlist?
- Currently, this isn't easily possible in spotify, so we created an app for it
- This app allows users to easily log into their spotify accounts, merge pre existing playlists, and then see the updated playlist in their spotify account
- Not only is it a cool feature, but it also allows users to collaborate and share their joy of music with one another by merging public playlists

Features

- Create a newly merged playlist directly from the web application.
  - The user is able to take two of their personal playlists and use our web apps merge functionality to create a new playlist containing all of the songs from both of the merged playlists
- The playlist is automatically synced with the users account in spotify.
  - When the user merges the playlist, it creates a new playlist that shows up in the their personal spotify account and they can listen to it right away
- Customize your playlist directly from the web application.
  - The user can edit songs and change names of playlists directly from our web app, making managing playlists easier than before

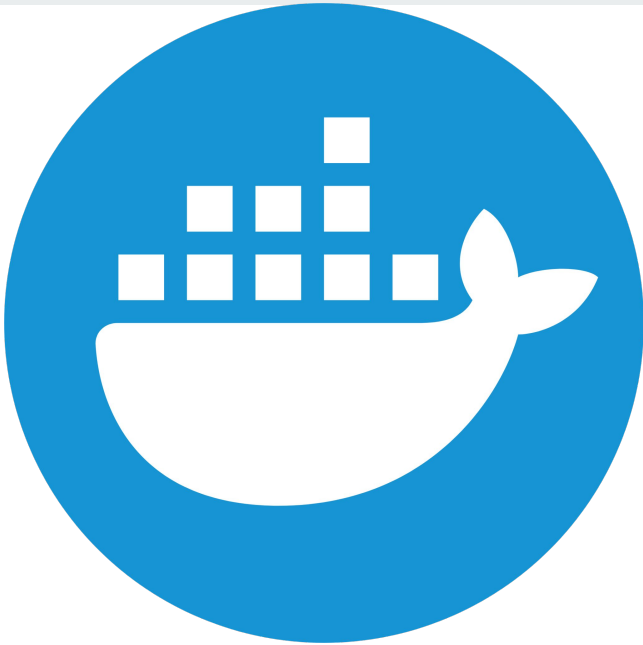
Azure

- We are using Azure to deploy our project.
- We give Azure a 5 star rating because it has a free-tier for it’s end user testing which was very useful for us to ensure everything is fully functional.

Visual Studio Code

- The code editor that we used to build our project
- 5 stars for usefulness. It has been a simple and well thought out software that has lots of extension integration for more niche uses cases.

Docker



- Docker is a containerization software we used to simplify running our software between different environments. It allowed us to all have the same usability on our separate devices, and would allow a user to have the same.
- I would give this software 2.5 stars. We ran into a couple of issues with it that we were hung up on but once everything was sorted it was very useful.
    - Things like containers sticking around for too long
    - Containers running on the same ports
    - Containers not being updated on git pulls
    - Usually solvable with docker compose down -v, but hard for novices

Node.js

- We used Node.js as our runtime environment for this project, which allowed us to execute javascript code locally, giving us the ability to develop and test our application.
- Node.js gets 5 stars as we could not have created this project without using it, or something similar to it. It was easy to understand, and vitally important.

Spotify API - Authentication

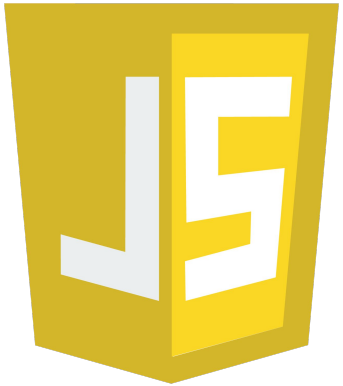
- The Spotify API's authentication setup was difficult but secure.
- The user had to be passed to a private page to personally authenticate
- Then a token was returned to us
- We also had to deal with devmode vs extended quota mode (right now our api key is quite limited because we're only students)
- The rating is 4 stars

Spotify API - Merge

- The purpose of using the Spotify API was to give us a way to interact with an actual users spotify account, and manipulate their library. This is obviously vital to a project such as ours.
- Spotify's API for merging gets a 5 star rating, as the documentation was very thorough on it, and they made it simple to implement all we were trying to do.

Languages Used

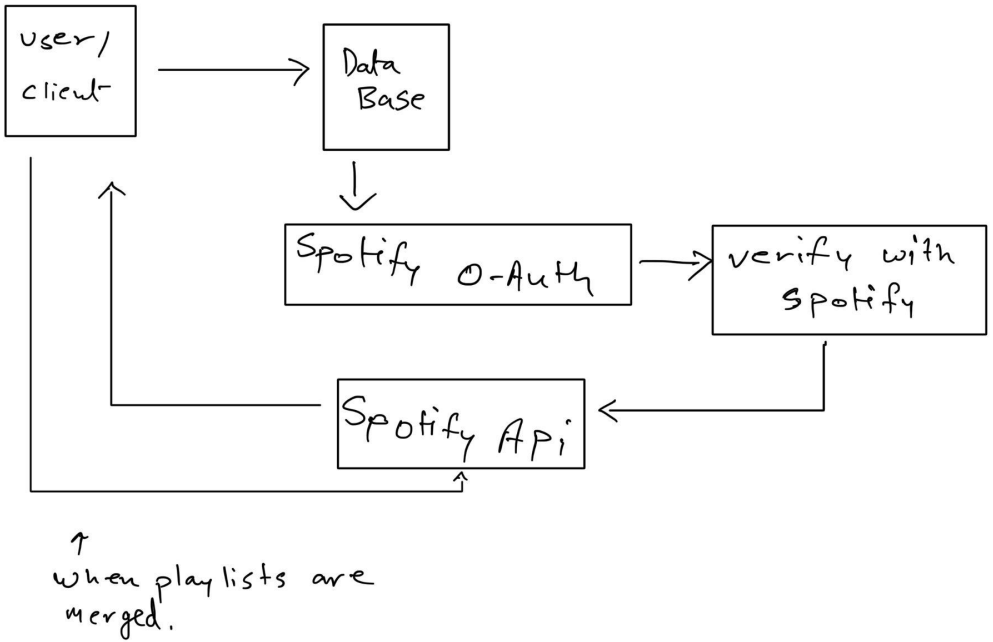
- HTML
- CSS
- JavaScript
- SQL



Architecture Layout

Tools Used In Each Layer of the Architecture:

Front End - Bootstrap, Handlebars
Middle Ware - Express-Session
Back End - Spotify API, Node.js



Methodologies Used

- Pair programming
- Peer code reviews with the TA
- Iterative weekly meetings with the group
- Iterative coding style paired with agile coding

Demo

- Time to walkthrough the Merge Music site!

Challenges & Solutions

- Spotify API Routes - Noah
  - All the api routes are written in TypeScript.
  - Solution: We had to adapt TypeScript into JavaScript.
- Merge API Routes - Danny
  - Solution : Lots of hours of coding
- Unintended files were committed to the GitHub repository.
  - Solution : Increase merge main into the branches.
- Github Miscommunications - Kelly
  - One person pushed a change, overriding other's changes
  - 2 people working on same feature
  - Solution: use the project board to coordinate which features people work on. Remind everyone to pull often

Future Plan

- Collaborate with different Spotify Accounts
  - Being able to merge your playlist with another user
- Not limiting ourselves to just Spotify
  - Being able to merge playlists in Apple Music, and other music streaming platforms.
- Cross-collaborate with various applications.
  - Merging a playlist from Spotify with a playlist from Apple Music
  - Users will be able to select which application(s) they would like to push their playlists to.
- In-app enhancements
  - System to check for repeating songs from different playlists.
  - User being able to pick which songs they don't want to merge.
  - More customization to the playlist, such as a playlist image.

Conclusion

- Seamlessly merge two playlist into one.
- Don’t have to worry about transfer playlist to the app.

Q&A

Any questions?