

Music-Merge Design Synopsis

Team Members:

Kelly McVeigh | kemc7914@colorado.edu | KellyMcVeigh

Noah Schwartz | nosc1301@colorado.edu | NoahBSchwartz

Eli Jones | eljo7407@colorado.edu | elfjones

Aayush Shrestha | aash8639@colorado.edu | aaaayush21

Daniel Alemayehu | daal6363@coloraod.edu | DTAlemayehu01

Toby Moore | tomo7434@colorado.edu | T0blerone

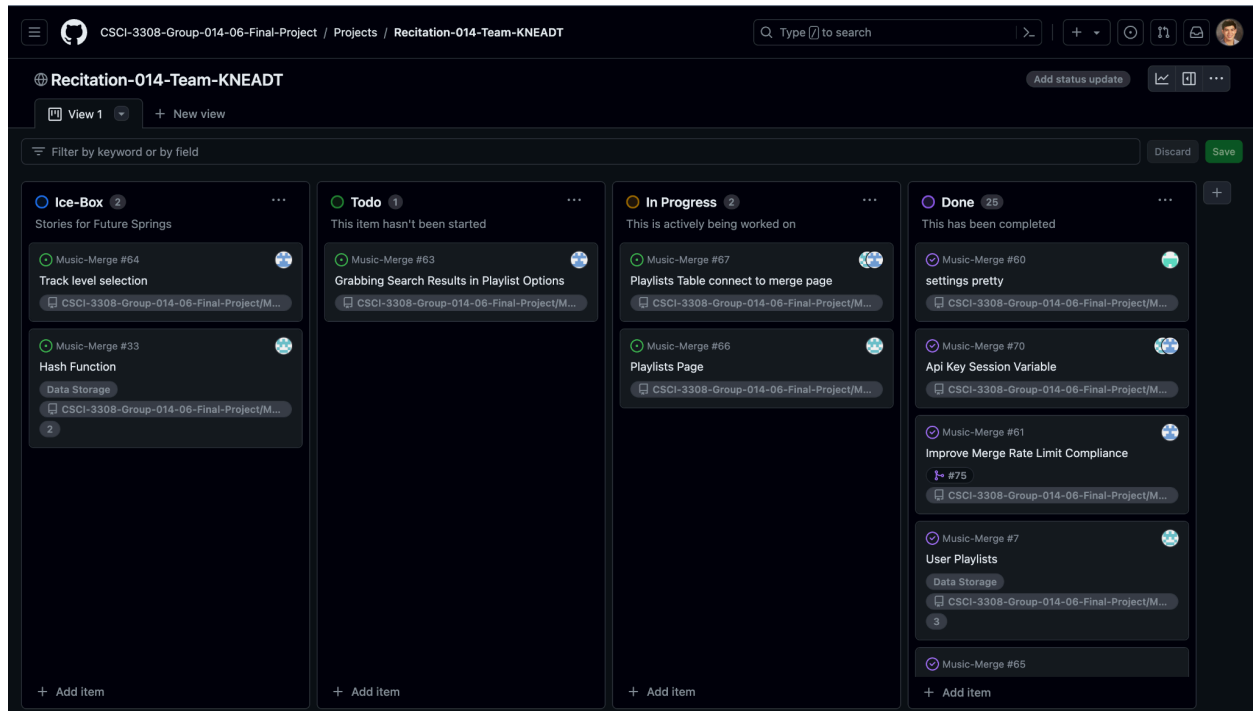
Application Description:

One big issue with many modern music streaming platforms is the ability to merge two playlists into one. Our application will allow users to look at their playlist library and select two playlists they want to combine. This feature will become even more powerful when it's used collaboratively: our site gives users the ability to merge their playlists with their friends' playlists. This emphasis on making music new and shareable again is at the core of our application.

The first step is a login page allowing users to register to the site and link their Spotify account. This is crucial because after they have entered their information, we'll be able to access all of their music libraries. Once their personal information is merged, the site will give them the option to share it with their friends. They can simply enter the username and password of another person in the database and then choose one of their playlists and one of their friends' playlists to merge.

Our strategy for increasing novelty is to allow the user the choice of how to generate the merged playlists. They can choose to import only a couple of songs from each playlist or they can choose to "enhance" their playlists. This will be done by using the API to generate suggestions for songs that fit the aesthetic of both original playlists. The last step in this process is writing the finalized playlist back to the user's Spotify account so that they can stream the music easily.

Project Tracker: [Link to our Project Board](#)



Video

https://github.com/CSCI-3308-Group-014-06-Final-Project/Music-Merge/blob/main/MilestoneSubmissions/Music_Merge_Demo.txt

VCS:

<https://github.com/CSCI-3308-Group-014-06-Final-Project/Music-Merge/tree/main>

Contributions:

Eli - I mainly worked with the login page, some in the back end and some in the front end. I ensured that the post I used HTML, Javascript, and CSS to carry out my tasks.

Toby - I created the settings page, and added some get and post api routes to index.js to help implement the login and settings functionality. I also made it so the options you choose in the settings page take effect within the merge.

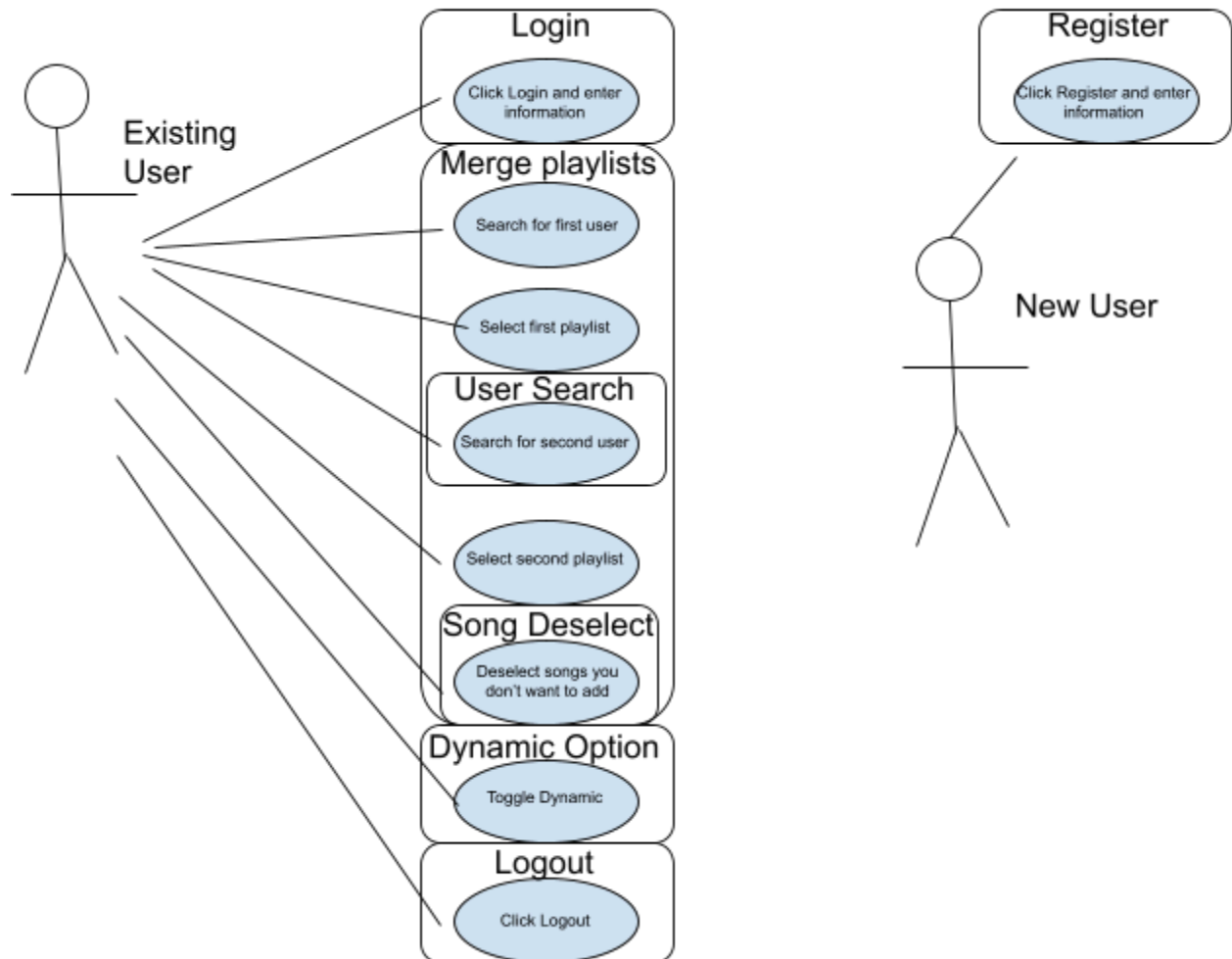
Aayush - I worked on the navbar, using HTML and bootstrap with handlebars, to make it responsive to the size of the users screen, also added user authentication flag, where the navbar switches it's menu based on whether the user is logged in or not, and middleware authentication for various pages.

Noah - I worked on almost everything that had to do with the API and the music merge page. I was responsible for authenticating the user and dealing with all of the relevant data (and creating the pipeline that funneled the user from registration to spotify to login to music merge), I created an API route to get all user playlists and built the front-end to display them, and I sent the user's selection data to be stored in our database. I also built the home page for the project.

Daniel - I was also a principal contributor to the API, I'm responsible for the merge route primarily along with some work in the discovery route. I also have work fixing a variety of minor bugs that cropped up in the project. I also set up the github and cleaned up any forgotten issues/epics on top of managing releases. I also manage the hosting for the project.

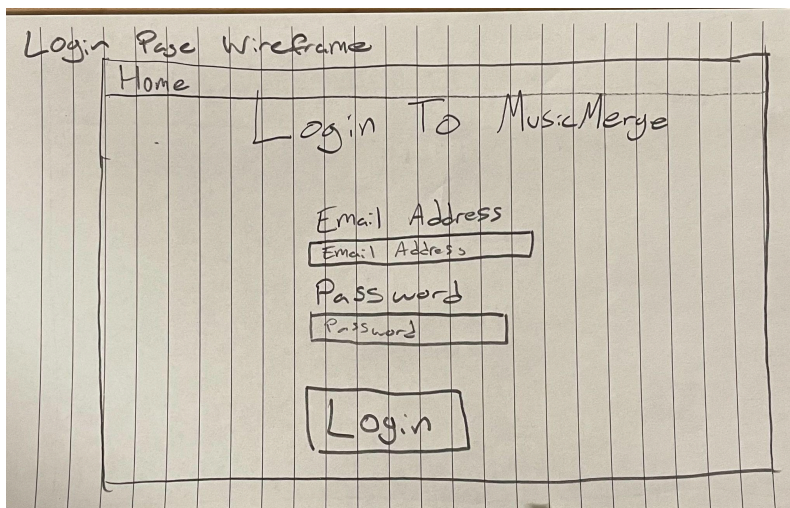
Kelly - I implemented the "Your Merged Playlists" page, front and back end, as well as the database functionality. I was responsible for writing and debugging the SQL queries, and debugging the register and login functionality. We experienced numerous issues with API and the registration, as well as database tables and calls, so I successfully debugged and problemsolved many of those issues. I, in addition to Noah, worked as a task manager and helped to keep everyone on track, on the same page, and having a balanced plate of tasks. We communicated timelines, meeting notes, and task layouts, and kept the git projects page up to date.

Use Case Diagram:




Wireframes:

Login Page:



Register Page:

Register



Link: 

Create
account

Discover/Home Page:


Home Logout

Library

Select	Playlist 1	
<input type="checkbox"/>		
<input checked="" type="checkbox"/>	Playlist 2	

Viewing Playlist:

HomeLogoutLogout



Playlist C

- 1.
- 2.
- 3.
- 4.
- 5.

Create newview in Spotify

Test Results:

Test 1 -

Users should be able to register for Music-Merge by linking their Spotify account.

Test Data: we will use Spotify account: possiblynamedtoby.

Acceptance Criteria:

- A user cannot submit the register form without completing all of the mandatory fields.

Mandatory fields include:

- Spotify Account
- Information from the form is stored in the user database.
- Localhost will be used to simulate a realistic user environment.
- Toby will act as the User Acceptance Tester.

Observation

- The user saw the create account button from the homepage of the website, the login page.

- Upon being prompted to enter their spotify username, and then log in to spotify the user complied.
- User logged into the newly created account, and was let into the home page, showing that their info was stored in the database.
- There is no deviation from the expected actions in the registration process.
- No changes were made based on this test.

Test 2 -

Users should be able to login to Music-Merge by entering the data they previously used to register.

Test Data: we will use Spotify account: possiblynamedtoby.

Acceptance Criteria:

- A user cannot submit the login form without completing all of the mandatory fields.

Mandatory fields include:

- Spotify Account
- The login page will not work unless the account is correct.
- Information from the form is cross-referenced with the user database.
- Localhost will be used to simulate a realistic user environment.
- Kelly will act as the User Acceptance Tester.

Observation

- The user was placed on the login page automatically upon opening the website.
- The user entered the username that they previously registered.
- User was placed on the home page, indicating that there was an interface with the database, and login was successful.
- There was no deviation from the expected actions in the login process.
- No changes were made based on this test.

Test 3 -

Users should be able to navigate to the merge page, check 2 playlists, click merge, and receive a message saying "Playlists Merged."

Test Data: we will use Spotify account: possiblynamedtoby. We will select the first 2 playlists to merge.

Acceptance Criteria:

- A user cannot submit the merge request without completing all of the mandatory steps.

Steps include:

- Checking playlist #1
- Checking playlist #2
- Selecting "Merge Playlists"

- Information from the form is used to make a call to the API which creates a new playlist. This playlist is stored in the playlists database.
- Localhost will be used to simulate a realistic user environment.
- Noah will act as the User Acceptance Tester.

Observation

- From the homepage, the user selected the merge playlist button.
- The user selected two playlists that they wanted to merge
- The user selected the merge playlist button.
- No feedback was given from the button, which was confusing for the user as it was unclear if the process was started or not.
- The playlist appeared in the users spotify account, indicating that the API interface worked successfully.
- We added a message to notify the user of the merge as a result of this test.

Test 4 -

Users should be able to alter the settings that will take effect when they merge their playlists together. These settings include the publicity of the playlist, and the name of the playlist.

Test data: We will use Spotify account: possiblynamedtoby, and then alter the settings page to change the name to "Test123". We will then merge 2 playlists.

Acceptance Criteria:

The resulting playlist must have the title "Test123".

Information from the settings form and merge form should be used to make an API call, and create a new playlist.

Danny will act as the User Acceptance Tester.

Observation -

- From the homepage, the user selected the settings button.
- The user changed the Playlist title option to "Test123".
- The user pressed save, and was returned to the homepage.
- User selected merge playlists.
- User selected the playlists they would like to merge, and pressed merge.
- Inside their spotify account, the resulting playlist appeared with the name "Test123".
- This indicates that the API functionality worked as expected, and the settings page is functional.
- No changes were made based on this test.

Deployment:

<http://daal-csci-project.eastus.cloudapp.azure.com:3000/discover?>