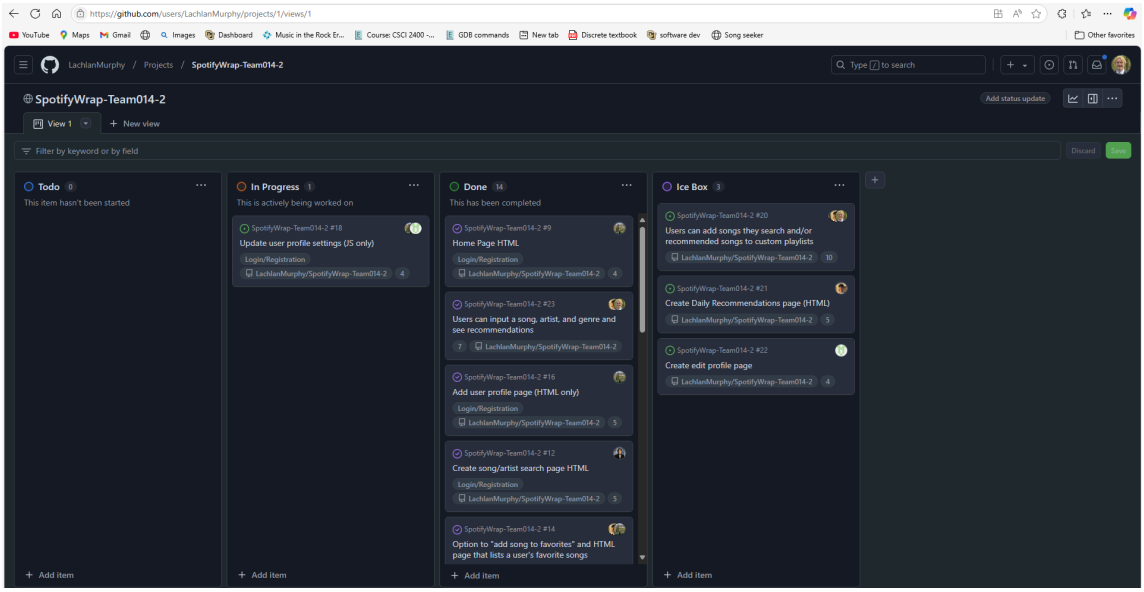
SongSeeker

Group Members:
Gavin Petruzzi - gape5988
Lachlan Murphy - LachlanMurphy
Cody Mattox - CodyMattox
Brady Gaona - Braeden464
Cole Younoszai - Coledy2004
Joshua Wright - joshwright04

Project Description:
SongSeeker is an application that users can leverage to search for artists or songs in order to get recommendations for new songs they might enjoy. SongSeeker also has a “favorite” feature so users can save songs or recommendations to listen to them later. The first screen of the website displays a login/register page where users can choose to create an account with SongSeeker or login if they already have one. Before the user can access any page on SongSeeker, they must login. Once a user does log in, they are instantly greeted with a home page. At the bottom of the page, the user will see instructions to navigate the application and two options to either “search for songs by artist name” or “search for songs by song name.” In both search pages, the search results display what users look up and give recommendations based on what they look up too. All songs are displayed with an album cover picture which can be clicked on if the user wants to open Spotify and play that song. At the top of the screen, a navbar presents users with options to go home, view their favorites, logout, and view their profile.

Project Tracker:
[View 1 · SpotifyWrap-Team014-2]



Demo video:
https://drive.google.com/file/d/1HZvDQt8Y1WSV-1B5f828hL3bA-XcO4ZK/view?usp=sharing

VCS:
https://github.com/LachlanMurphy/SpotifyWrap-Team014-2.git

Individual Contributions:

Cody Mattox:

- My main contributions to the project were focused around the APIs we were using. I developed the different search and recommendations functions using Spotify's API, and after Spotify deleted some endpoints we were using, Gavin and I quickly implemented new recommendations with the Last.fm API. I also used handlebars and HTML to create forms to make sure the data I was retrieving was being rendered to our pages correctly. I made minor additions in the HTML adding the feature that allows you to click on the album picture and it brings up the song in Spotify. I worked very closely with Gavin on all of the javascript we wrote for the APIs we were using.

Gavin Petruzzi:

- My contributions to the project involved primarily the javascript file as well as handlebars within our artist search and song search HTML pages to correctly display data when pages render. I was also responsible for authenticating with Spotify's API and worked closely with Cody to implement initial endpoint functions for search and recommendations. When Spotify unexpectedly changed their API, Cody and I worked quickly to implement a new recommendations function with Last.fm's API. Finally, I wrote much of the documentation related to our final product, including the README file, final release notes, and weekly progress reports in MilestoneSubmissions.

Josh Wright:

- My main role revolved around CSS and HTML, however I also worked on the search songs API endpoint, developing the feature that allowed multiple songs to appear in song search. Improved general formatting of code around the website that made code easier to read and improved runtime. Created the rough draft of the website design that was later used to create the HTML and CSS styling of the home page. Created the HTML and CSS styling for the song search page, as well as the recommendations page. Contributed to various other endpoints in the

index.js file in order to make accessing Spotify and Last.fm's song database possible/easier to access.

Cole Younoszai:

- During this project, I was responsible for the front end development. I designed the HTML, CSS and handlebars for every one of our pages. I worked a lot with other group members to make sure the frontend was compatible with the backend of our project. I made the color theme green originally and changed it to orange when we rebranded. Many times during our project we had to change plans for various reasons and this often resulted in format modifications which I always handled. One example of this was splitting our initial home page functions into two cards that each took the user to a page with a different function. I also dabbled in the JS files for two reasons. First, to help display user data in the profile page when registering. Also, to get songs and artists printing correctly when searches were made.

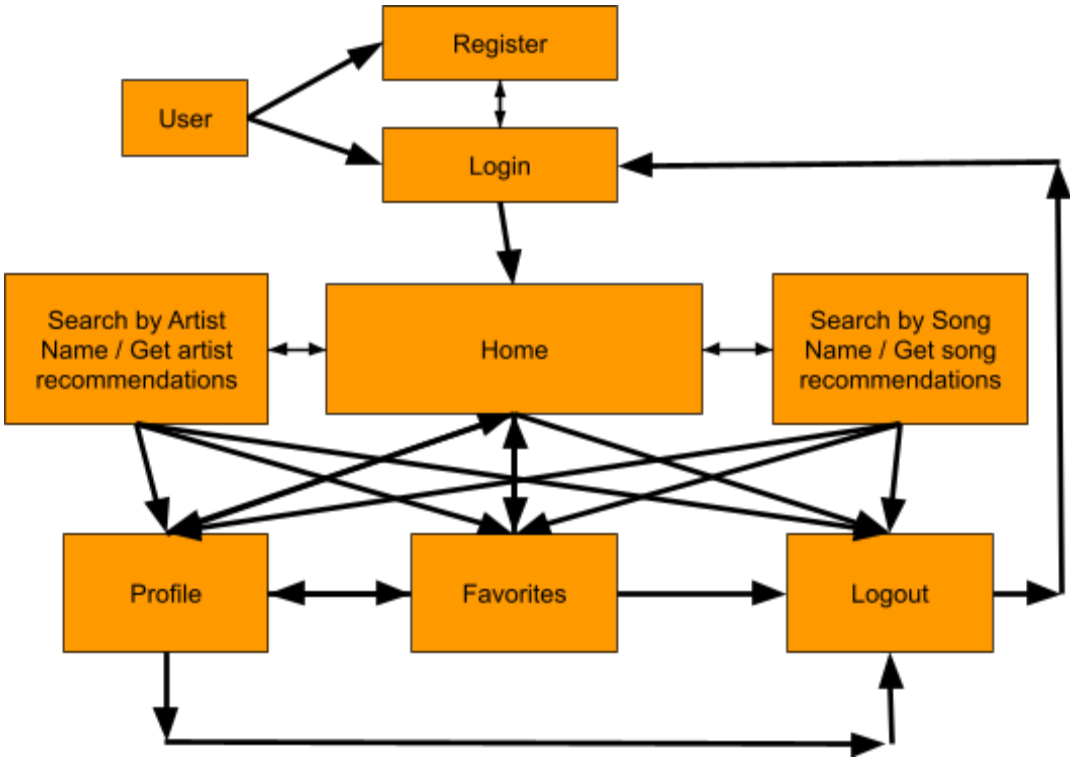
Lachlan Murphy:

- My main role during this project consisted of maintaining the server end of the website. This included setting up get and post requests for our website so that users could interact with the server and our SQL database including adding users to the database, creating sessions for each user, adding favorite songs to a user's account, and middleware. I also contributed to a large portion of the APIs, especially debugging how the APIs worked with the server. I had some minor contributions to the HTML elements. This was mostly so that the HTML could communicate with the server in a consistent fashion.

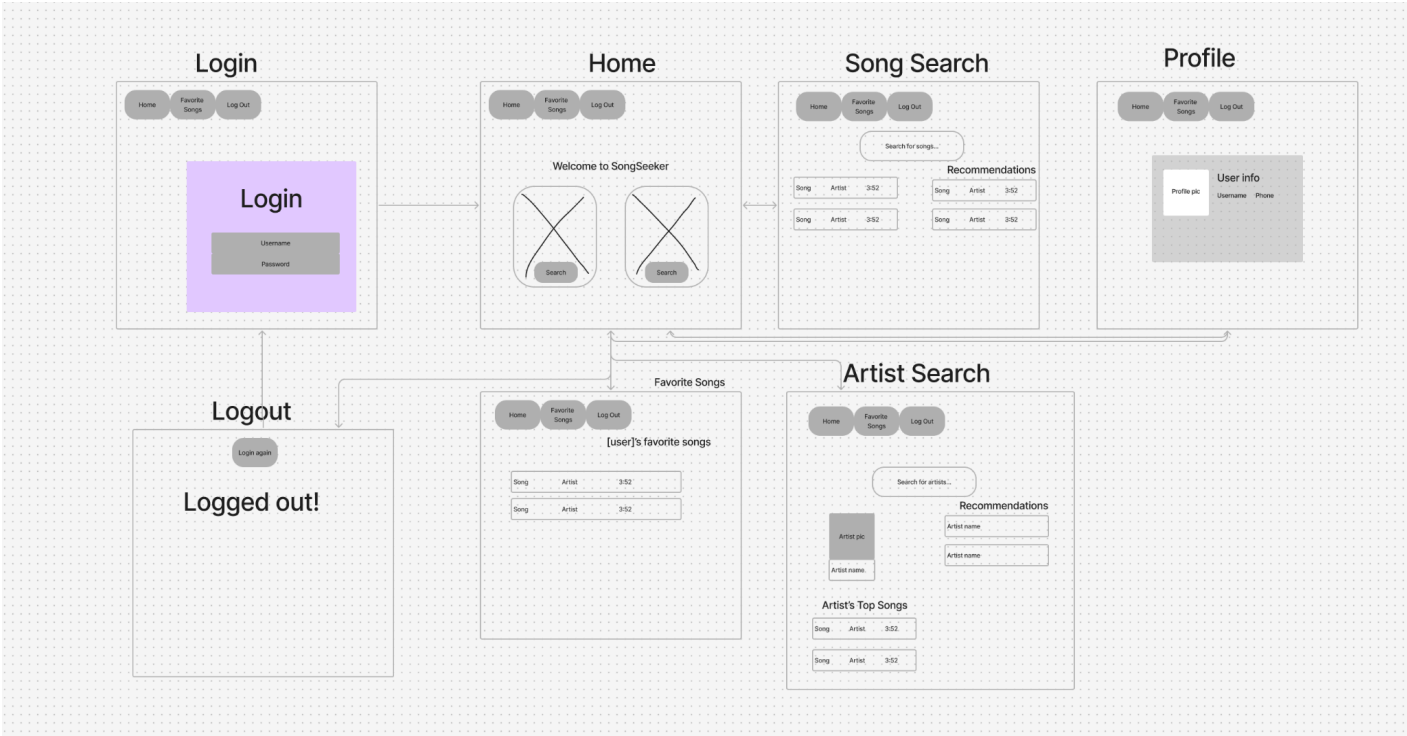
Brady Gaona:

- My main role during this project was maintaining SQL databases and creating all of the tables that we needed for this project. This included setting up the register and login pages so the information from those pages would be saved. I also worked on inserting data into SQL tables. During this project I also worked on some endpoints needed for the profile page and the search songs page. I also created a logo for SongSeeker. I also worked on some of the HTML for favoriting songs. Some other contributions that I made were creating an edit profile page that ended up being scrapped and taking the information from the register page and putting it in the profile page.

Use Case Diagram:



Wireframes:



Test Results:

For our user acceptance testing, we tested our server as well as a few of our endpoints with both positive and negative test cases. When running the application locally, all of our tests run and their results are printed in the terminal. By doing this, we were able to ensure that key features of our application worked as intended while we were continuously making changes to our repository. If any of the tests fail, the application cancels before being run on localhost:3000. The first and most important test was on the server. If the server successfully runs, a welcome message is printed. The other endpoint tests we used are listed below:

- Registration Endpoint: A user must be registered before they can login to SongSeeker
    - Mandatory Steps:
        - User must register with their full name, phone number, username, and password
        - A new username being registered must be unique
    - Positive test case:
        - Returns a success status when a user registers a valid new account and that account is added to the database
    - Negative test case:
        - User attempts to register under a username that already exists
- Login Endpoint: A user must login before they can access the home page, favorites page, profile page, search pages, and logout button.
    - Mandatory Steps:
        - User must login with a username and password that have been previously registered under an account
    - Positive test case:
        - Returns a success status and displays a welcome message when a user logs in
    - Negative test case:
        - A non-registered user should not be able to login and an error status should be returned. They will be asked to try again or to register an account
- Favorite Endpoint: A logged in user should be able to favorite any songs and/or song recommendations
    - Mandatory Steps:
        - A user must be logged in

        - Song results in either of the search pages need to be rendered. Each will have a “favorite” button next to them that can be clicked
    - Positive test case:
        - A user who is logged in can favorite any song rendered with Spotify’s API
    - Negative test case:
        - If not logged in, favoriting any song should not be possible. In this case, a user should be instructed to login

Link to deployed application:
SongSeeker