Marc Castro

Spotify API Web App

"Semi-Wrapped"

Documentation

Description

Semi-Wrapped is a Web application that is created to allow users to view data from their spotify accounts and see their top tracks, artists, and genres across the span of 6 months. The application pulls data using the Spotify API specifications and uses code specifications from the github repository formally titled "Spotify-Web-API-Node". This application is built using the react framework, along with backend implementation via Express.js and Node.js for server side token handling that is pulled from the Spotify API itself.

Tools, Frameworks, and Languages:

- Languages:
  - HTML & CSS
  - JavaScript
  - JSON
- Frameworks:
  - React.js
  - Express.js
  - Axios
  - Bootstrap
- Tools:
  - Node.js
  - Visual Studio Code
  - Recharts
    - https://recharts.org/en-US/
  - Netlify
    - https://www.netlify.com/?attr=homepage-modal
  - Spotify API
    - https://developer.spotify.com/documentation/web-api
  - spotify-web-api-node
    - github: https://www.npmjs.com/package/spotify-web-api-node

Components & Functions (Client):

App.js

- Central environment that handles switching between dashboard and login prompt upon entering the app.
- If "code" exists to access the Spotify API data and the Spotify user account, changes from login to Dashboard

ProcessGenres.js

- Function file for processing the genres into an array of overarching genres, adding unique genres that do not fall into the "overarch" category into the final array
    - ProcessGenres = (topArtists) => {}
        - Takes the array of top artists pulled from the Spotify API and partitions through the genres listed under the artist and creates a new array for them, then passes that array to "genreArrayFilter".
        - Returns the data sorted from highest to lowest occurrence.
    - genreArrayFilter = (genreArray, edmSubGenres) => {}
        - Takes genre array generated from ProcessGenres and partitions through to categorize by overarch/unique genres.
        - edmSubGenres is an array uniquely created to read for the various categories of edm via looking for keywords within the genre name (Ex. "melodic" is found in "melodic bass").
        - Runs a forEach loop to read if it is an EDM sub genre and breaks if true, reads to see if it is part of one of the overarches specified and breaks if true, and if neither are true it will push it to the final array because it is a unique genre (Ex. "bubblegrudge" is a unique genre, so it will end up in the final array that is returned).

Dashboard.js

- Component file that uses the auth code obtained from useAuth.js to pull data from the Spotify API for display.
- Upon creation from App.js, it fires 3 useEffect functions that grab and process the data into key-value pair arrays that are stored in variables appropriately named "topArtists", "topTracks", and profile using react useState hooks.
    - topArtists is also passed in its useEffect hook to ProcessGenres in order to obtain the user's top genres list, the result of which is stored in variable "topGenres".
    - If values are not done being obtained, the file will wait until all data is gathered before displaying.

- Return value is HTML elements from the Bootstrap framework and is formatted via Bootstrap's grid system of rows and columns, as well as a color associated linear gradient that is tethered to the top overarching genre of the user as the background.

Login.js

- Component file that is rendered if there is no login code provided and/or if the application is just opened.
- Contains the authorization URL provided by the Spotify API documentation, as well as just a button to allow the user to proceed to the Spotify login page.

ChartCard.js

- Component file that returns Bootstrap elements to render the space for the Recharts Pie Chart I have implemented.
- Called from Dashboard.js and takes the genre data that is calculated by ProcessGenres.
- Calls the GenreChart file to create the pie chart and passes the data into said file.

GenreChart.js

- Component file obtained from recharts that takes data from ChartCard.js and displays it in an animated pie chart with hoverable components that display the genre and the percentage it takes of the pie chart.
- Pie chart will always start with the user's highest genre opened and displayed at the center of the chart.
- Return value is Recharts components that help render the chart to the destination, which is the ChartCard.

TopCard.js

- Component file that uses Bootstrap card elements to display data that is passed in.
- File takes a props component that contains the array of data to be processed and the title of the card.
- Primary purpose is to show either the top artist or the top track of the user.
- File will render the first element from the array passed, displaying the image src saved within the element, the name, and tethering the image to the Spotify link of the appropriate element, whether that be the song or the artist page.

ListCard.js

- Similar to TopCard.js, but uses the Bootstrap ListGroup elements as well as the Card elements.
- Component displays the next 5 (positions 1-5) of the array passed in, displaying the image, name, and tethered url of each element using the JavaScript .map function.

UserProfile.js

- Similar to ListCard and TopCard, but used specifically for the user's profile information in order to display the profile image and username of the user that has signed in.

useAuth.js

- Uses axios to push a post request to the domain with endpoint "/login" and pulls the Spotify API access token, refresh token, and the time it takes for the access token to expire.
- Also uses the same premise for the refresh token with endpoint "/refresh" to pull a new access token from the Spotify API to refresh the access token once it expires.
- Variables are stored in useState hooks with names appropriately called "accessToken", "refreshToken", and "expiresIn".

Example: