TEAM 068-CSSN REWIND

a. **Project Title**

REWIND

b. **Project Summary: It should be a 1-2 paragraph description of what your project is.**

REWIND aims to utilize the Spotify API to curate playlists of popular songs based on different attributes. By separating the data into songs, artists, and users, we will be able to efficiently sort through all the data on Spotify to make the best playlists for each user's wants.

The UI for our project is going to be simple, but effective. We want it to look good next to the actual Spotify interface and be easy for anyone to use. To use the playlist generator, each user must create a login that will connect them to all the playlists that they generated in the past. REWIND will make it easy for anyone to find music based off of what they already like and will generate an actual scannable Spotify code and link so they can make it an addition to their actual Spotify account.

c. **Description of an application of your choice. State as clearly as possible what you want to do. What problem do you want to solve, etc.?**

The application will be a music playlist generator. The web app will generate playlists of popular music for the user depending on preferences like genre, artists, time-period, number of songs, etc. The generated playlist will create a link which can be used on Spotify.

This application makes it easier to generate personalized playlists for users. Currently music apps don’t necessarily create playlists for you depending on time-period preferences or genre preferences. This will allow users to listen to songs with more personalisation. It can also simulate what music would be available to you if Spotify existed in the 1900s.

d. **Usefulness. Explain as clearly as possible why your chosen application is useful. Make sure to answer the following questions: Are there any similar websites/applications out there? If so, what are they, and how is yours different?**

The two alternatives to an app like this would be to create your own playlist manually or find someone else’s collection of songs online. This

app takes away the tediousness of choosing your own songs and adds more personalization to your playlist as compared to using someone else's playlist. It also makes it easy to sync the generated playlist with a widely used application like Spotify.

e. **Realness. Describe what your data is and where you will get it.**

   To obtain the data we will be utilizing the Spotify API. Using the GET requests given on the Spotify Developers Console, we will obtain information on the Artists, Tracks, Albums, etc., in the form of a json file. This data will contain information such as follower count, popularity score, Image link, unique ID's, release date, etc. We will then create a User and Playlist table, and use the POST requests available to create the playlist on spotify.

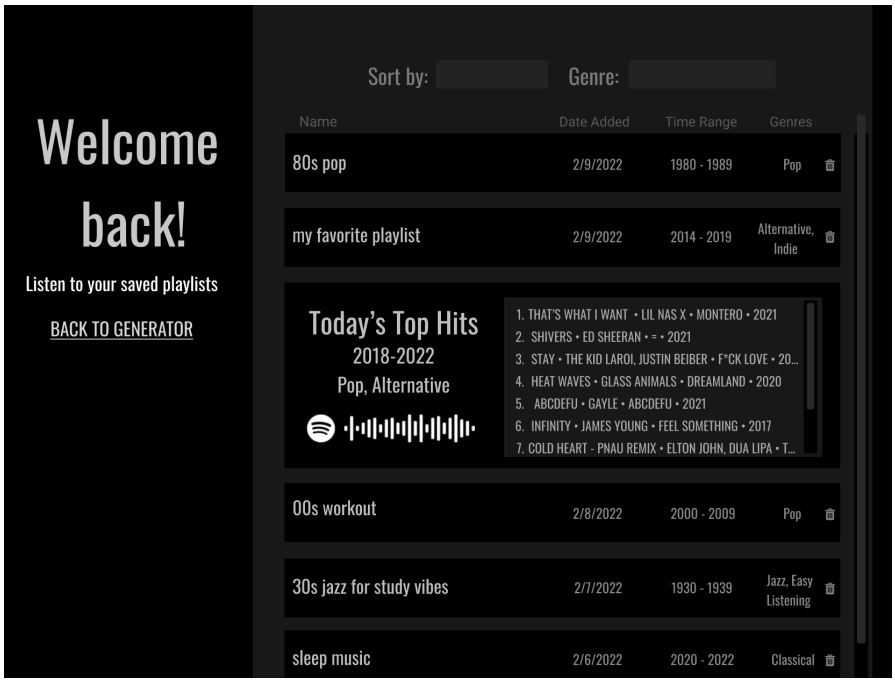
f. **Description of the functionality that your website offers. This is where you talk about what the website delivers. Talk about how a user would interact with the application (i.e. things that one could create, delete, update, or search for). Read the requirements for stages 4 and 5 to see what other functionalities you want to provide to the users. You should include:**
   i. **Describe what data is stored in the database. (Where is the data from, what attributes and information would be stored?)**
   ii. **What are the basic functions of your web application? (What can users of this website do? Which simple and complex features are there?)**
   iii. **What would be a good creative component (function) that can improve the functionality of your application? (What is something cool that you want to include? How are you planning to achieve it?**

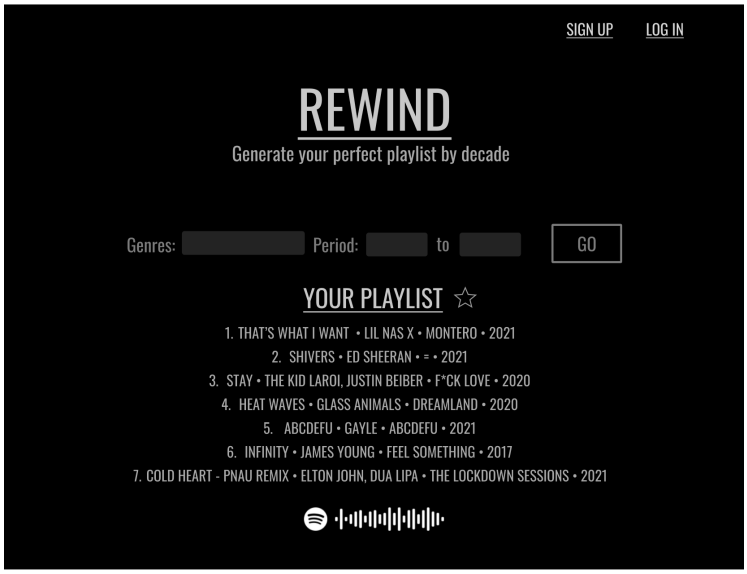
   The website will have a few entities. Some of these entities and their attributes are as follows:

   Artists: **ArtistId**, Name, Genre, Followers, Image, Popularity, …
   Songs: **SongId**, Name, Genre, Number of plays, image, releaseDate, album, ArtistId, …
   Playlists: **PlaylistId**, UserId, link, num_songs, years, genres, …
   Users: UserId, Firstname, LastName, LastPlaylistId, …
   The data for the Artists and Songs in the above list will be obtained using the Spotify API.

On the website, users will be able to add constraints such as a range for years or a list of genres. This would create a playlist whose link would be saved in the playlist table along with the user who queried it, and their constraints. A user could search and delete from their list of playlists. Furthermore, we will be able to obtain the number of playlists for a given user or find the total songs in all their playlists. We could also find their most common genre or find the oldest song on all their playlists.
The creative component for our website would be the ability to create a real playlist on Spotify that the User can listen to. Furthermore, with enough users, we can also find the most popular year requested or the most popular genre for a given year. Finally, another interesting component of our website would be the utilization of the popularity score of songs to create these playlists, instead of randomly generating them.

g. **A low fidelity UI mockup: What do you imagine your final application's interface might look like? A PowerPoint slide or a pencil sketch on a piece of paper works!**





h. **Project work distribution: Who would be responsible for each of the tasks or subtasks?**
List of the person responsible for which exact functionalities in section f. Explain how backend systems will be distributed across members. Be as specific as possible as this could be part of the final peer evaluation metrics.

The project can be split up into 4 umbrella categories: database setup, querying, frontend development, and authorization. We have divided each category into subtasks that can be evenly distributed between the four members of the team. However, it is most likely that we will not follow this plan precisely as we want to collaborate with each other possible- instead, each member will maintain ownership of the tasks they are assigned while also working with others members to accomplish them. The distribution of tasks is written below and color coded as follows:

Nayonika	Sreyas	Sarah	Chirag

1. Database Setup
    - Users
        - Attributes: UserId, Email, Password (encrypted), Playlists
    - Playlists
        - Attributes: PlaylistId, Years, Genres, Songs
        - Songs

        - Attributes: SongId, Name, Genre, numberOfPlays, Image, releaseDate, Album, Artist
    - Artists
        - Attributes: ArtistId, Name, Genre, Followers, Image, Popularity
2. Queries
    - Create Playlist
    - Create User
    - Get Playlist
    - Get User
    - Update Playlist
    - Delete Playlist
    - Update User (optional)
    - Delete User (optional)
    - Get Song, Artist (optional)
3. Frontend Development
    - Home Page
        - Playlist generation bar
        - Generated playlist
    - Profile Page
        - Playlist summary component
        - Expanded playlist component
        - Sort/filter bar
    - Login Page
    - CSS - everyone
4. Authorization - everyone