ILLINI HARMONICS

PROJECT SUMMARY

Our project is an online platform that will be designed for users to engage publicly with their music tastes through posting their brief ratings of music and artists. The website will function as both a social media platform, and a personal music rating repository. Users will be able to create their own personalized profiles, with the ability to provide crucial information on their musical tastes via Spotify and potentially Apple Music integration. Primarily speaking, through the Spotify API, users will be able to view their listening history from spotify, as well as any of their frequently listened to songs so that they can quickly and easily rate them. Furthermore, users will be able to interact with each other and engage in sharing their thoughts about different music through profile discovery and through existing connections. Through linking their Spotify account with their profile, users can quickly find their friends on our platform, or invite their friends to the platform. Additionally, users will be able to discover new people through a discovery hub, containing reviews and ratings from other people on the platform. Through this, users can like or respond to music critiques by other users, and they can follow other people if they are interested in seeing that person's ratings more often.

The primary functionality of the site will be through a simplistic rating page, where the user simply has to scroll through their listening history and click values on the scale of 5 stars for each song. This brevity and ease of access seen in the reviewing process is what makes our site unique and approachable for users, as nobody would sensibly want to write written reviews for all their songs and artists. This page will additionally contain a feed for listened-to artists, where artist ratings may also be pushed to the web server. Moreover, all of the data regarding the user's ratings of songs can be used to provide search recommendations on the site and to tailor shown songs more carefully. With more data on the user's opinions of music, better music recommendations can be provided to the user, allowing them to explore music that best fits their tastes based on genre and rating overlaps from other users. This feature is secondary to our intention of making the site a rating-oriented application, but it provides extra depth to our implementation that would make sense given the data we plan on collecting.

APPLICATION DESCRIPTION

People bond over reviews. Watching a movie and sharing opinions or eating at a nice restaurant with proclaimed dishes or reviewing a newly released album are all social activities in which we all participate. However, unlike reviewing movies or food, there is no simple way to review music. The movie industry has websites like Rotten Tomatoes and IMDB which provide large databases of reviews created by many types of people. Similarly, the restaurant industry has websites such as Yelp or OpenTable. In both cases, it is incredibly easy for any user to log on, write a review, and post it on a forum where others can interact with it. There is no comparable platform for the music industry. No accessible way exists to quickly and efficiently post music reviews. Additionally, many common music apps such as Spotify or Apple Music lack features to help a user find new music. It is difficult or impossible to find songs based on popularity or any kind of review or rating metric. As a result, song recommendations are often inaccurate. It is clear that the music industry is primed for a breakout platform which will allow all users to rate music and interact with other users in a more efficient manner.

The goal of this project is to solve all of these problems by creating an efficient and user-friendly platform to facilitate the reviewing and sharing of music. We plan to use available Spotify data and user listening histories to allow each user to rate their recently listened music. This process is significantly easier than using a third party to write a formal review. Additionally, other Spotify users are then able to view the posted reviews and interact with them. By accumulating a large user base, each song's rating will eventually accurately represent the public perception, allowing for much better recommendations. Combining these features into one service will most aptly address the problem which we aim to solve.

CREATIVE COMPONENT

We believe that it's important for the user to not only be recommended songs that they might like, but to also be able to discover songs manually. For this reason, we believe that it would be cool to implement filters for different songs such that the user can search for songs that might fit a specific group. For example, if a user needed a happy and uplifting song to cheer them up, then they could filter happy songs into their search to be able to find a list of songs that fit their current mood. Moreover, if a user was planning on hitting the gym to work out, they can filter their discovery to only show songs from a "hype or workout" group, in which the songs elicit excitement before entering the gym. The way in which we would achieve this seems difficult to implement, and is not necessarily included in our minimum viable product, but nonetheless is something that we are striving for.

To implement this, we would need to make a set of mood groups with which we split all of the music on the platform into. The way in which we would identify which song goes in which groups initially would rely on data from the spotify API regarding things like genre and tempo. Furthermore, we would be able to improve upon this feature with data of users' playlists to see which songs are typically grouped together and why, in addition to users' reviews of songs, and what emotions they elicited from them.

USEFULNESS DESCRIPTION

Illini harmonics, a music reviewers' application, bridges the gap between movie reviewing and music reviewing. Compared to movies with a vast range of categories, the musical world is built by great composers and passionate singers, and enjoying and reviewing music works should not be neglected. Illini harmonic allows users to manage music pieces as easily as they want. Using spotify API to query and interact with spotify databases, the application is designed so that users can focus on their history list, and rate the pieces they like. Illini harmonic also encourages users to share their thoughts and comments of their favorite pieces with friends and family. Additionally, the application also provides access and operations to spotify music lists, that being said, users can choose from the highest rated or most popular music from the spotify list without necessarily opening spotify itself, instead they can enjoy the music pieces and review them in Illini harmonic all together.

The most similar application that is comparable to ours would be Musicboard. Musicboard is also a music reviewing application, the application provides a place where music listeners can post their reviews. However, this application is too complicated for someone who wants to enjoy music; Illini harmonic, on the other hand, will be pure and clean, users can post their reviews on any pieces without being disturbed by pop-up ads or fancy interfaces. Furthermore, users can rate their music through simply scrolling and pressing a single button, rather than typing a full-blown review like Musicboard encourages. Illini harmonic also has built-in access to spotify, where users can more easily choose the songs they want to rate based on their listening history.

DATA SOURCE (REALNESS) DESCRIPTION

Within our application, we plan on utilizing the Spotify API in order to perform the majority of our data retrieval. We may also extend our user accessibility by using the Apple Music API for getting user listening data. This would be helpful as many people either have one or the other, so we wouldn't want to exclude users from our website's core purpose: efficient and simple music reviewing based on your listening history. Hence, the Spotify and Apple Music APIs would serve as our primary two databases. We could utilize others if we need to access things such as artist or album images, but both of those APIs are already capable of doing so.

In terms of describing the technical aspects of these databases, both databases return JSON files upon query requests. Therefore, we could easily use Python as our backend and convert these JSON files into Python dictionaries, which would then be transferred into our custom SQL databases. The API databases that we plan on using will return various types of information, such as: user listening history, artist names, artist profile pictures, song names, song cover pictures, and streaming counts for all queried songs. Inherently these queries all return different cardinalities and degrees, and some of the information we need may be returned conjointly in a single query. But this may all be found out from exploring the Spotify Web API website. For example, the user listening history query is listed on the website and has a maximum cardinality of 50 songs, and a degree of roughly 27 various returned attributes. This same process of searching the API help website would also work for any other functions we need to query with.

Now, the degree of these requests may seem large, but it's important to keep in mind that we are using our own datasets that are simply taking the API data, and narrowing it down only to the attributes that we need. Our own datasets will consist of the following tables thus far:

1. A table with all users’ basic profile information
2. A table with users’ song ratings
3. A table with artists, their songs, and genres
4. A table with all songs that have been listened to or rated by users in addition to their averaged user rating and total streams
5. A table connecting users to their friends that they’ve added
6. A table with user login information

FUNCTIONALITY DESCRIPTION

The primary queries and functionality that will occur in our project pertains to user listening history. This query will occur when the user connects their Spotify API token to the website upon registration or profile editing. It will retrieve music and artist listening history so we can then provide a streamlined rating system in our website’s “rate” page. This query will additionally occur when the user either first loads the website, or when they open pages that require their history to be updated (rate and profile pages). Optimally however, we would simply update their history upon the webpage loading initially so we can ensure that it gets updated. Furthermore, we could also query every user in our user database either daily or hourly just to make sure that listening activity is being consistently updated on the site. This is important as our site is intended to function partially as a social media site.

In our application, users can also query the database of the most popular or highest ranked music pieces according to the Spotify database. This type of query would be used for sorting our popularity tabs for both songs and artists. Each user may additionally have update queries regarding their friends as they invite friends, family, and other users to the website. Such updates will append other users to a user’s friend table, which will tell our website to display their friends’ listening activity on pages such as the discover page.

Other types of queries besides update and insert queries exist in our website’s functionality, for instance, the delete query. It may be possible that a user wants to delete their account from our website, so we must ensure that we have the proper delete functionality that can handle cascading or nulling entries in our web server’s database related to the user. Other than the delete query, we will also have search queries within our website’s search bar atop the navigation panel. This will allow users to search between artists, users, and songs for creating manual ratings or for viewing user profiles. We will also likely have to utilize triggers and constraints in order to ensure that all relevant tables are updating upon a query, and that they are updating within the proper constraints that we define. For instance, a single song cannot have multiple reviews from the same person, thus we cannot make a query that will insert another entry for the same person. This would be resolved either through constraints or updating the data instead of inserting a new entry. Similarly, triggers could be utilized upon updating a user’s song reviews in order to make sure that the objective song rating in the song table is adjusted as the average value of all user reviews.

UI MOCKUP

The following three images are example mockups that we will model our user interface after. The first image provides a glimpse at the discover page, wherein we have the options to scroll through popular songs based on stream count, or the opportunity to scroll through the songs with the highest ratings. This page will also provide other users' activity - such as friends or popular people - on the side as well as some genres that users can sort through. The layout of this mockup allows easily for more categories of songs to be added as the user scrolls down. It also allows us to add a more user centralized feed for actual user reviews below the popular and highest rated sections.

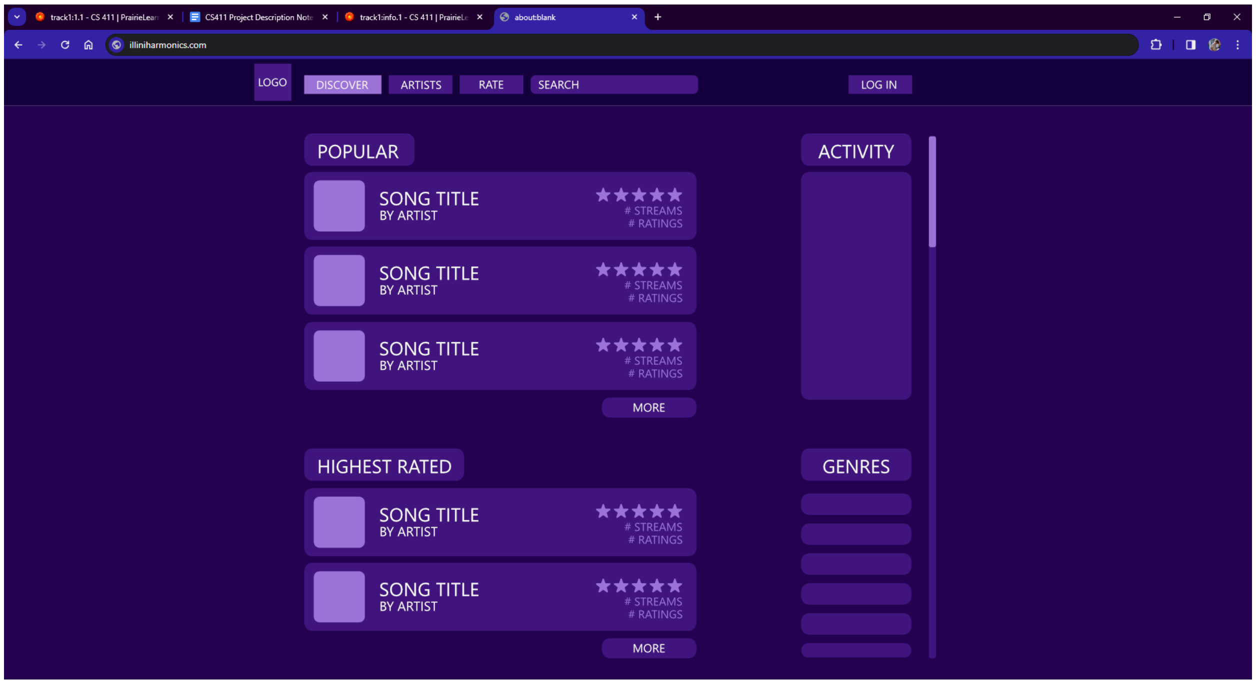

Figure 1. Discover page mockup

The second image shows the second tab of our web application, wherein users can directly look at artists based on their popularity in streams or user ratings. Two tabs exist at the top of this page in order to allow filtering by either option interchangeably. It may also be noted at this point that in all of these images, a search tab alongside a login tab is visible at the navigator atop the site. The search tab will allow users who haven't connected their Spotify API token to manually search for songs or artists and review them. Furthermore, the login button will provide the system through which we allow users to access their reviews and profile. Once logged in, the login button will be replaced by the user's profile, where they'll be able to view their own activity on the site: music ratings, artist ratings, friends, etc.

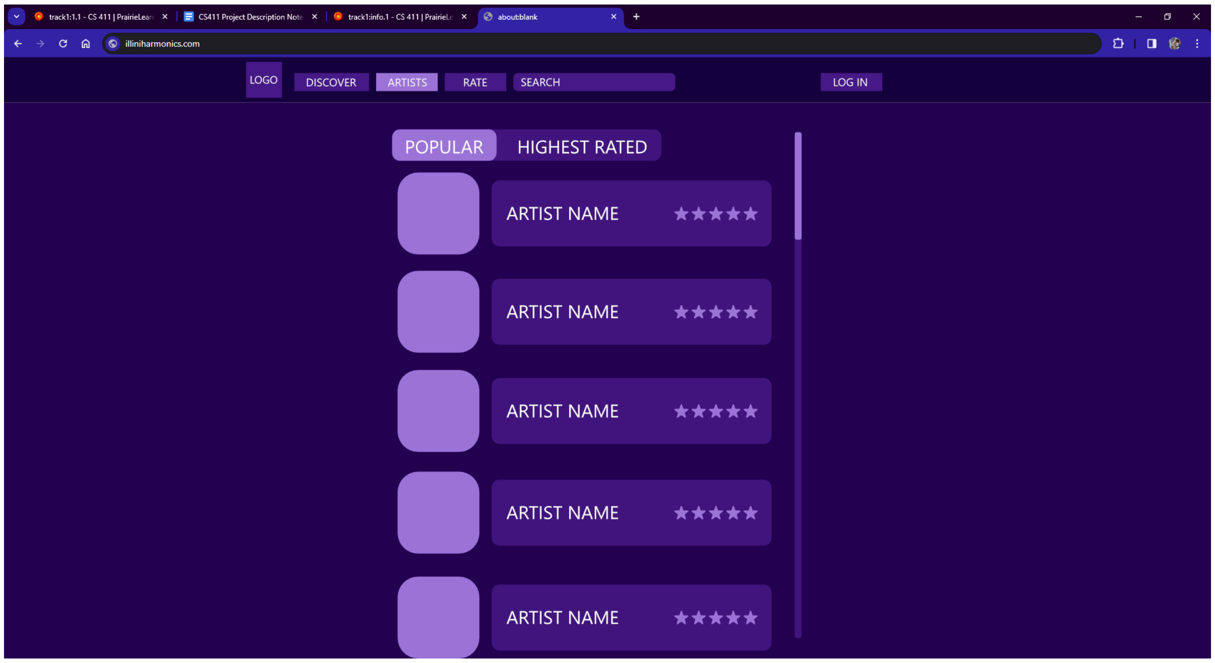

Figure 2. Artists page mockup

The third image shows the rate tab, wherein users can rate either songs or artists. Within the image, the ability to switch between artists and listening history isn't apparent but it will be once the tab is implemented. Here, the user may click any of the stars on the right of the artist in order to push their rating to the web server. This simple implementation of reviewing music encapsulates our vision for sharing music opinions, where it isn't lengthy and doesn't require a written review that nobody will realistically read.

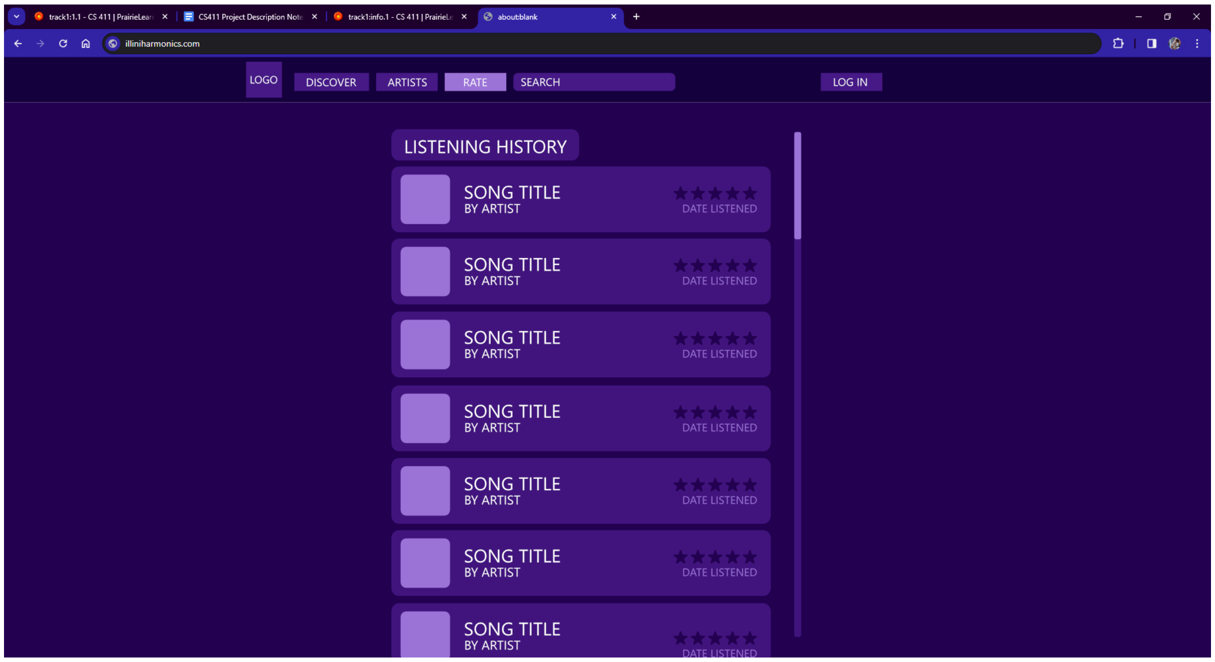

Figure 3. Rate page mockup

PROJECT WORK DISTRIBUTION

We anticipate that the workload of the project will fall into the following four rough categories: Front End Development, Web Hosting, Database Management, and Back End Development. For each category, we will appoint a team leader who will be responsible for managing the group's efforts. Michael will manage the end, Jacob will be responsible for database management, and Paul and Joe will conjointly lead web hosting and backend segments of the project. By letting each team member lead a certain aspect of the project, we aim to create a defined focal point of study for each member while still allowing for full teamwork and cooperation within the group. Our goal in this project is for each member to fully understand the work behind developing a web application, so we will not assign one task to just one member. Additionally, to maintain a rigid schedule we have established team meetings at a frequency of at least once per week. This schedule should allow us to stay up to date with the development of our project and guarantee that all necessary deadlines will be met.