CS411 Section A5 Team 5

Prof. Donham

9 May 2022

Weatherify: Final Documentation

In the beginning, our group had two initial ideas: 1. Playing music matching the weather and 2. Generating recipes from a user's available groceries. We were interested in utilizing the Spotify API, so we decided to pursue the weather-playlist generator option. The main goal of our project was to be able to create a Spotify playlist based on the weather in a user's location in order to conveniently customize music selections for the user. For our technology stack, we decided to use a Python Flask backend with a React frontend framework.

To implement all of the features for the application, we researched several APIs, including Open Weather API and Spotify API. Fortunately enough, there were numerous weather and music APIs to choose from that were available online for free.

- **Weather API's:**
  - Weather API: https://www.weatherapi.com/
  - Yandex Weather API: https://yandex.com/dev/weather/
  - OpenWeatherMap: https://openweathermap.org/api
- **Music / Spotify API's:**
  - Deezer: https://developers.deezer.com/
  - Watson: https://www.ibm.com/cloud/watson-natural-language-understanding

Fig.1 Possible APIs we researched in the beginning of our project

After researching the tools we would use to create the project, we came up with the following user stories:

1. As a user, I want to be able to log into my account and access the application.
2. As a user, I want to be able to check the weather conditions and temperature in my area for today so that I can know the weather around me.
3. As a user, I want to be able to choose from suggested playlists without the need to generate my own in order to have quicker access to music.
4. As a user, I want to view and modify my list of previously made playlists (e.g., delete unnecessary playlists so that I do not clutter my account with old/unwanted playlists).
5. As a user, I want to be able to create a Spotify playlist for the weather in a different chosen location so that I can explore other areas and other music.

We implemented functionality for most of the user stories above, but had to drop some ideas due to time constraints. Although our application saves each generated playlist (associated with a user) to the database, users are not able to see and modify their previously created playlists. For future implementation, the ability to view and modify a user's list of saved playlists would be ideal, so that users can visualize all their playlists and favorite and delete any playlist. We would also add further UI improvements, such as displaying the user's playlists on their profile page and displaying a map on the search page so that users can drop a marker on any part of the world, view the weather, and generate a playlist accordingly. Lastly, we would implement a language filter that allows users to specify the language that they would like their songs to be in, allowing users in different countries to also make use of our application.

We did run into difficulties during the development process, the first being that since the sentiment value of Spotify API tracks range from 0 to 1, we had to convert weather values into

sentiment values ranging from 0 to 1. We successfully were able to hardcode values for each possible weather category (sunny, cloudy, rainy, etc.) in Open Weather API, and used these weather categories to select tracks by weather. Additionally, we ran into many CORS errors arising because of proxy issues when sending data between the React app and Flask server. We were able to resolve these by using axios instead of fetch to retrieve data from the backend. Another obstacle was when we struggled to implement Google OAuth to log users into our application, especially when trying to use JWT tokens. Instead of passing the token from the authenticating server to the React app in the cookies header, we decided to pass the token through the URL as a URL parameter and store it in local storage to persist.

Another challenge was the database. The initial structure of the database wasn't too hard, we just needed to save their email address and the playlists they created under the keys of a userID. However, we were quickly met with a problem. We weren't sure how to access the currently logged in user's playlists without the userID, which was lost after being saved initially to the database. One solution our group member mentioned was "let us save the user's email in a global variable at the instance of logging in, so we can save that as the primary key in the database, and then kind of just save all the playlists they make along with it under their email" and that is actually how we solved this problem.

Overall, we achieved most of the functionality that we strived to do for the scope of our project this semester. While there are plenty of features that can be implemented in the future, our main goal was to create a platform that allows users to generate Spotify playlists based on the current weather and we achieved this. We learned a lot through working in our team and are all satisfied with the final product.