Riffle - Spotify Listening Insights
Team Name: Riffle
Members: Braeden Watkins, Chris Ramos, Tyler Cartier

Introduction

Overview

Riffle is a web-based application that uses the publicly available Spotify API to provide users with useful statistics about their listening habits. The app aims to enhance user engagement with their music data while also serving as a learning opportunity for the development team in API integration and React/JavaScript development.

Motivation

The primary motivation behind this project is to familiarize ourselves with APIs and learn the basics of React. Our overall motivation is to learn common systems used in industry while still creating something that could be useful to our daily lives as well as others. All of the team members enjoy listening to music and wanted to develop something that could make our listening efforts more efficient. Specifically, we wanted to find a way to adapt our playlists to better fit our tastes, using statistics to do such.

Approach

We began by spending our time focusing on implementing a sound backend that was capable of handling token authentication without fail, so that making API calls would be seamless. This took up the majority of our time and was rather challenging, but our thorough work in this area resulted in not having to patch backend bugs later on. After authentication, we focused on making user-specific API calls and began to store user data in our chosen database (Supabase). From here, we implemented one of our major features, WebPlayback, and successfully created almost a Spotify player replica. At this point, the only thing left to do was implement our import feature and, in turn, our skip feature, resulting in a completed project that has already shown to provide benefit to our listening habits.

Major Changes

The application was originally supposed to be a cross-platform application using React Native with the Expo framework; however, making authentication between applications possible was becoming extremely difficult. When one component was fixed, it would only cause issues with another. Due to these issues, the team decided that it would be more beneficial to resort to

creating a web-only application. In making this decision, we converted our project away from React Native with Expo to React with Next.js, as it contains more of the web-based features we needed to obtain the desired application. One of our proudest features, the WebPlayback feature, would not have been possible without switching over to React.

Results

The team is very satisfied with the application and is very happy with the result. We quite honestly exceeded expectations and actually have plans to continue development after this course to continue making improvements and provide even more benefits to our potential users. Our main goal at the beginning of the semester was to have a functioning skip feature so that we were able to track how many times every song we listened to had been skipped. This would allow users to remove overly skipped songs from their playlists, and we were ultimately able to accomplish this. Because of this completed feature, the team deems this project a success and is happy with not only the product but also how much we learned about React and web app development.

Customer Value (no changes)

There were no changes to what we wanted to provide to the customer. The main features we wanted users to be able to have access to have been implemented and are not only loved by the developers, but testers also seem rather impressed with the application as well.

Technology

As mentioned before, the architecture changed drastically. Initially, our goal was to create a cross-platform application, but we realized early on that it would take us too long to finish. We then switched our architecture to React with Next.js, focusing on only making a web version of our application. Our application also utilizes a database that is tied to the backend, in which we make api calls to reference user statistics and artist information. The implementation of the application was fairly straightforward. While at the start, none of us had experience with React Native and Expo, we could make steady progress after switching, as we had experience with HTML, CSS, and JS. The frameworks handled the page linking, so all we had to do was create the UI and functionality for the pages. Although we didn’t put time aside specifically for designing the UI before coding it, we quickly determined what we wanted it to look like.

Our testing consisted of implementing an API call and displaying it in basic JSON format to make sure we got the information we wanted. After confirming the information, we began extracting it from the object and formatting it to look presentable. We did this for each API call we made. It was crucial to make sure we were actually getting the information and testing the API calls so that users did not get errors when information was missing. We made sure that there were no NULL values presented to the user in the way we implement the API calls. This proved helpful, and we were able to complete the look of the pages more quickly due to it.

As shown in the image below, every track interaction has a value, whether it be 0 or something else. This image shows the layout of the main table in our database. It holds information regarding the user's interaction with a certain song. We use the information stored in the play_data column and track_duration to calculate all of the stats for that specific song. The information for these 2 columns is gathered from the JSON that the user imports and from api calls.

id int8	u... t..	track_id text	listen_count int8	skip_co... i...	play_data jsonb	track_duration int8	minutes_listened float8
4985	22n6ht...	7AEcphXUR52...	2	0	{"plays":[380053,380053]	380053	12.6684333333333
4745	22n6ht...	1pvBm1ydKLXo...	6	1	{"plays":[104672,81912,86	167106	13.0353333333333
4732	22n6ht...	2l3dDtrbMbQ...	5	2	{"plays":[106488,68997,16	174407	12.1117
4729	22n6ht...	6mWxaBCDvJ...	5	0	{"plays":[338287,187397,18	187397	18.13125
4723	22n6ht...	3s3Eo4rb6aq3...	10	5	{"plays":[177228,29482,177	177228	26.5014166666667
4689	22n6ht...	7KjpKG8xORV...	8	1	{"plays":[2946,184025,186	186960	21.8122
4611	22n6ht...	0BA3uoKlu9C...	8	2	{"plays":[210826,210826,5	210826	24.7223833333333
4570	22n6ht...	0McjONE4eqk...	12	1	{"plays":[265186,265186,2	265186	45.8064
4555	22n6ht...	5c9wBtVuRjnr...	3	1	{"plays":[228000,27074,22	228000	11.8512333333333
4549	22n6ht...	4RtUedkLzskA...	7	2	{"plays":[174080,174080,17	174080	20.7690333333333
4548	22n6ht...	2IlsiiMXVFJpB...	8	1	{"plays":[241887,241887,24	241887	32.3275166666667
4479	22n6ht...	1AOLchfVKydX...	9	1	{"plays":[209146,2364,209	209146	28.0166666666667
4437	22n6ht...	1jfiv2N9KAx79...	11	1	{"plays":[150732,114708,67	178625	22.1722166666667
4386	22n6ht...	5w3knJ0bVSb...	15	1	{"plays":[186320,186320,18	186320	46.63235
4291	22n6ht...	6dpCPillXgu3J...	5	1	{"plays":[216653,216652,2	216653	18.0938833333333
4279	22n6ht...	0QjnzOVvDG2...	4	3	{"plays":[192000,4421,192	192000	12.26825
4214	22n6ht...	58yR1fuvNzr0...	25	5	{"plays":[240057,240057,2	240057	92.9081666666667
4212	22n6ht...	28hs7031Cjcz6...	9	6	{"plays":[194876,194876,12	194876	30.9374166666667
4195	22n6ht...	4qTVvlH3AdR...	17	3	{"plays":[185026,185026,3	185026	54.7174
4143	22n6ht...	6e38ee2ioja5C...	4	2	{"plays":[206999,206999,	206999	13.8179666666667
4138	22n6ht...	1JpvGHw0UO...	21	7	{"plays":[205400,205400,	205400	65.4059333333333

Team

Role of Members

The project was led by Braeden Watkins and Chris Ramos, and the two did the bulk of the work. Both Braeden and Chris contributed to almost all areas of the application (backend and frontend), with Chris having a heavier focus on backend while Braeden had a heavier focus on frontend UI and making user-specific API calls. Both Braeden and Chris are responsible for the token authentication used to make API calls, while Chris is solely responsible for the import and database functionality. Braeden is responsible for the stats page, although both Braeden and Chris implemented the WebPlayback feature together. Tyler is responsible for the landing page.

Rotating Roles

The roles of team members almost flowed with the state of the project. It was necessary to primarily focus on backend token authentication first so that issues related to authentication would not have to be dealt with later. Once backend authentication was sound, the team then moved to frontend functionality and connected the backend components with the UI. Essentially, the team decided that implementing feature by feature in an incremental fashion would be more efficient (and it proved to be). We did not split up entirely, implement our own features, and hope we would be able to properly connect them later on.

Project Management

The only goal that we did not complete to our satisfaction was our app being cross-platform. This is something that we originally wanted and set out to do, but are also not upset that it was not able to come to fruition. The reason we scrapped the cross-platform feature was due to its complex nature and the limited time constraint of the course. After deciding to make a web-only application, the team was able to stay on schedule and make solid incremental progress. This incremental strategy resulted in all original goals being met, other than the cross-platform functionality.

Reflection

Issues

The original learning phase was quite challenging, and the design process could have been planned and researched better. The team did not do enough research about the challenges of implementing a cross-platform application, specifically related to Spotify and the process used to

authenticate a user. The team just started focusing on properly authenticating with the web version without considering Android and IOS. We picked a framework that is designed for cross-platform and accepted this blindly while just assuming this fact would make the transition easy (it was not). If we were more careful and did more in-depth research before beginning, the scrapping of cross-platform functionality may not have been necessary.

Positives

After the move away from cross-platform and the challenges of learning the frameworks and languages had been overcome, the development process went quite smoothly. We were able to implement all of the features that we wanted to and even made them better than we had originally imagined. Overall, the development phase was rather effective, and everyone is happy with the results. The team spent ample time testing the application, which turned out to be very beneficial. Small bugs, such as particular areas of buttons not being able to be clicked and invalid input that varied between users, were all able to be patched due to sufficient testing. Some of these bugs were application-breaking, but because of testing, the team believes we have a working product. Team management was also an overall success, as can be seen due to all deadlines and goals of the project being met.

Success?

Overall, the team does consider Riffle to be a success. There are no aspects of the project that we are unhappy with. We have a fully functioning application that has proven to be quite useful in adapting the team's listening habits to become even more refined to our individual tastes. There could certainly be additions to the applications, as goes for any application out there, and there are active plans to continue development over the Summer. Most of these plans come in the form of refinement of features, but additional features, such as adding users' Spotify playlists to the apps, have been planned as well. The structure and architecture of the application have been created such that future development should be smooth. Hopefully, our structurally sound application will eventually be able to be pushed into development.