The Pennsylvania State University

The College of Information Sciences and Technology

Senior IST440W Capstone Project

Audio Spot Final Report

Prepared by

Cameron Sotoodeh & Sean McSweeney

# 

# **ABSTRACT**

This report encompasses the progression and explanation of our senior capstone project Audio Spot. Audio Spot is a web service that allows users to log in to a website that will output their music listening history back to them, like the wrapped service that Spotify provides for their users at the end of each year. We believe that it is important for users to have access to their streaming data instantly so they can keep up with their listening trends and the news of their favorite genres of music. Our project uses Webflow and its integration tools to allow us to use the Spotify and Twitter APIs to our advantage. As it stands now, our service allows our users to see their recently liked songs and the news of their favorite artists. Due to many constraints including time, understanding of web development, and expertise, our service is more limited than we both intended and for our future goals. Future iterations of our project aim to include communication among social features, room for advertising/profit, and a more immersive experience into the world of one’s music listening history.

# **TABLE OF CONTENTS**

Contents

[**ABSTRACT** 1](#_Toc101468385)

[**TABLE OF CONTENTS** 2](#_Toc101468386)

[**LIST OF FIGURES** 4](#_Toc101468387)

[**LIST OF TABLES** 5](#_Toc101468388)

[**INTRODUCTION** 6](#_Toc101468389)

[Problem Statement 6](#_Toc101468390)

[Motivation 6](#_Toc101468391)

[Objective 6](#_Toc101468392)

[End User Needs 6](#_Toc101468393)

[**LITERATURE SURVEY** 7](#_Toc101468394)

[Introduction 7](#_Toc101468395)

[Assessment of Available Solutions and Techniques 7](#_Toc101468396)

[Pros and Cons of Each Approach or Technique 8](#_Toc101468397)

[**REQUIREMENT SPECIFICATIONS** 9](#_Toc101468398)

[Market Requirement Analysis 9](#_Toc101468399)

[Design Requirement Analysis 9](#_Toc101468400)

[Constraints 9](#_Toc101468401)

[Assumptions 10](#_Toc101468402)

[Outcome Criteria 10](#_Toc101468403)

[Risks 10](#_Toc101468404)

[**SYSTEM DEVELOPMENT** 12](#_Toc101468405)

[Concept Generation 12](#_Toc101468406)

[Functional Decomposition 15](#_Toc101468407)

[Principle Operation 18](#_Toc101468408)

[18](#_Toc101468409)

[Design Process 19](#_Toc101468410)

[**CONCLUSION** 33](#_Toc101468411)

[Changes in Development 33](#_Toc101468412)

[Challenges 33](#_Toc101468413)

[Lessons Learned 34](#_Toc101468414)

[**FUTURE WORK** 35](#_Toc101468415)

[**REFERENCES** 36](#_Toc101468416)

# 

# 

# **LIST OF FIGURES**

1. Technical Flow Chart – Inputting Spotify Credentials (Figure 2.1) 15
2. Technical Flow Chart – Song Feed (Figure 2.2) 16
3. Technical Flow Chart – News Feed (Figure 2.3) 17
4. Operational Flow Chart (Figure 3.1) 18
5. Work Breakdown Structure (Figure 4.1) 21
6. User Interface Design (Figure 5.1) 22
7. Database Backend – Twitter (Figure 5.2) 23
8. Database Backend – Spotify (Figure 5.3) 23
9. Spotify Developer Account (Figure 5.4) 24
10. Spotify API Request (Figure 5.5) 25
11. Spotify Song Metadata (Figure 5.6) 26
12. Twitter API Request (Figure 5.7) 26
13. Tweet Metadata (Figure 5.8) 27
14. Website - Homepage (Figure 5.9) 28
15. Website - audio Screener Page (Figure 5.10) 29
16. Website – Music Feed Page (Figure 5.11) 30
17. Website – News Feed Page (Figure 5.12) 31
18. Website – Contact Page (Figure 5.13) 32
19. Testing – Logical Case #1 (Figure 6.1) 33
20. Testing – Logical Case #2 (Figure 6.2) 34
21. Testing – Technical Case (Figure 6.3) 35
22. Testing – Twitter API Error Code (Figure 6.4) 35

# **LIST OF TABLES**

1. Gantt Chart – Original Version (Table 1.1) 12
2. Gannt Chart – Final Version (Table 1.2) 13

# **INTRODUCTION**

## Problem Statement

A large frustration and inconvenience among Spotify users are that the “Wrapped” service is only available for a short time once a year. There is no easy way for Spotify users to view their listening activity data in real time with any organization. In this growing age of digital convergence and information, it is fair that users can see their analytics in real time and access this whenever they feel.

## Motivation

After concluding that a service like this did not exist, we thought it would be interesting to take on the challenge ourselves. We wanted to provide a platform that linked with a music streaming account and was able to output the same information that is used to create Spotify’s Wrapped service. We believed that users would be interested in using this platform not only to view their listening data, but we thought that down the line, there could be social media applications and fan-artists involvement applications as well.

## Objective

The main objective of our project was to develop a platform whether it be a mobile app or a website to provide the users of Spotify their music listening data in real time. We wanted our platform to be easy to use, understand, and incorporate in daily life and accentuate the music listeners' experience. We set out to build this platform from scratch and integrate the Spotify API to bring some of their services data to our platform.

## End User Needs

We realized that our users were more likely to be avid music listeners and people that are more interested in the music industry. We found that users like these would be more likely to be interested in sharing and learning about what they are listening to rather than a less common user that doesn’t care about the numbers or data behind. This meant that our users would have high expectations about our service and would most likely expect the service to give them their data in an easy-to-understand way and in a prompt fashion. With this in mind, we were able to start to develop, plan, and build.

# 

# **LITERATURE SURVEY**

# Introduction

The two largest players in the music streaming industry, have reported growing popularity with their release of users’ annual rankings of their music listening history. This demonstrates the growing demand from music streamers for the ability to view and share their music data.

We seek to develop a tool that allows people to discover what mutual music tastes people have with their friends. This tool, starting as a website, would allow people to sign in through their Apple Music or Spotify accounts to create an account. Integrating with the Spotify and Apple Music APIs, when creating an account each user’s data of their entire music streaming history would be uploaded to our database. From this point, you can add yourself to groups with any of your friends or other people who have created accounts. Doing so will direct our program to cross-reference each person in that group's music history and pinpoint any similar songs, genres, and artists you have listened to and identify them. We aim to provide music listeners a cross-platform tool they can use to automatically detect what music they and their friends both like, regardless of the streaming service you use.

# Assessment of Available Solutions and Techniques

There have been several attempts of applications and features on larger music platforms to try and socialize the music listening experience, but virtually all of them have failed. First, we have major streaming services: Apple Music and Spotify. iTunes, the precursor to Apple Music, first launched an in-house social recommendation platform called Ping in 2010. The service failed after a couple of years and was replaced with Apple integrating other established social media platforms like Twitter and Facebook. Spotify has a limited number of built-in features that can be applied socially, but they are not very popular as the Spotify UI is not designed for social communication. Another issue with the major streaming platforms releasing social features is that any of a user’s friends that use a different platform are already eliminated from being able to access those features.

There are also instances of social media companies launching music-related features. A couple years after Ping, Twitter launched the #Music feature, which aggregated tweets about music that they think a particular user would like. That feature was eliminated after a year. The most relevant competitor in the social music space was an app launched in 2015 called Cymbal, which was the Instagram for music sharing. The app ended up shutting down after a couple years because they could not provide something unique enough to make people want to download and use another social network. In response to their failure, we will be focusing on leveraging people’s entire music history through the data that’s already been recorded by their streaming service. We would like to partially automate the process of finding mutual music tastes with your friends using each person’s existing music data. Considering that, we are more of a layer 2 solution to music discovery rather than just another social platform.

# 

# Pros and Cons of Each Approach or Technique

**Pros:**

One of the pros of our approach and technique is the fact that since Apple Music and Spotify deal with data that is so sophisticated and prevalent, the collaborative playlists and recommended songs are very accurate. Along with this, Apple Music and Spotify have large existing user pools and are very compatible with iPhone UI which is where most users access their music streaming. Finally, both Apple and Spotify contain very loyal fan bases that have a long history of dealing with their respective platforms.

**Cons:**

One of the cons of our approach and technique is the lack of cross-platform capabilities. Since the streaming industry is a business, different platforms do not cross well which makes the applications of mixing streams very difficult. This is also true for their social integrations; Spotify users are not able to follow and see Apple music members and vice-versa. Another con we found was that it is hard to introduce new technologies in this industry. A lot of respect and loyalty has already been grown so trying to switch users to something else has failed for many companies time and time again. Another con we came across was the fact that streaming services were never built to be social platforms. While there are some social applications to their service, the main goal of their operations is to stream music and not connect user to user. Along with this, streaming services fail to provide customization of group listening preferences.

# **REQUIREMENT SPECIFICATIONS**

## Market Requirement Analysis

The market of streaming services in the music industry has been exponentially growing in the past decade. Early bloomers like Pandora paved the way for other services to become one of the most important apps on users’ phones today. At any given minute, there are at least 750,000 tracks being streamed at the same time. Since Spotify currently has 32% of the market share (more than any other large streaming service) our team thought that we would find the most success enhancing one of their services. While future iterations of our design might involve the inclusion and branching out of other streaming services, we wanted to establish our technology and vision around Spotify primarily.

## Design Requirement Analysis

When designing our interface, we had certain priorities over others. The priority of our project was making sure that Spotify users were able to simply log in with their accounts and have the information presented to them. This way, there would be less user-based interaction with the site, and it would feel like more of a service, growing our customer satisfaction rating. \

Aside from this, we also wanted users to have access to other information that just their listening history. This sparked our thinking into the creation of a live feed of twitter content. This twitter content would be integrated within our site to provide a live playback of genre-specific content and news that was tailored to each user.

Having both things in mind, we thought that a website would be the best way to display and access this information. Despite our future efforts to create a mobile application that is compatible in the same way, we felt that users would have an easier and more hands-free experience accessing our service through a computer and the world wide web.

## Constraints

Constraints during this project affected us in every step of development, planning, and implementation. Our biggest constraints that we faced was our prior skillset of coding, our understanding of the project requirements, the time frame, and our reliance on Spotify to administer us with a developer account.

Coming into our project, our team that only consisted of two people, lacked the skills needed to develop a website of this complexity level. Because of this, we were stressed with finding a way to make it work. This included countless hours of research, testing, trials, error, and setbacks. The design of our project and its intended services had not been finalized until a month before the required due date. This caused our team to fall behind in the development and focus too much effort on the missed deliverables.

The requirements of our project that we set for ourselves were too far out of reach and likely impossible to achieve in the time frame that we had. Because of this, we always found ourselves trying to test our service before it was ready to be tested. This led to many project designs, UI redesigns, and even entire coding language switches. The version of our project now is based on over a dozen failed versions of our idea and we found that it was the only way to meet deadlines and create a project that would serve for the requirements of the class.

Given we had more time to learn the skills of web development and new coding languages, I think that our team would have excelled. We spent most of the semester on pure research and learning so that we could finally implement a working design. It was a very frustrating model as we had the creativity and passion behind our project, yet lacked the technical skills required to implement our ideas. Nevertheless, we prevailed among the conditions and designed to the best of our ability the time and resources we had on hand.

Something less out of our control was the API integrations that we required within our project. Since we were using Spotify’s services, we concluded that it was necessary to obtain a developer account from Spotify. This meant that we had to go through an application process for Spotify to approve us for an account and so that we could start gathering data from their database. This is something that set us back and required us to wait on the development of some of our project deliverables.

## Assumptions

For this project our assumptions were targeted in different areas. One was the outcome of our project while the other was the challenges that we faced along the way. From the beginning we assumed that we would start to run into problems when we began designing the site. Again, alluding to our lack of technical skills, we always knew that we would have issues implementing our services within this format. We assumed that our final outcome would be more polished of a website. We assumed that we would have not only an overview on someone's listening data, but also a social aspect within our sight that allowed users to share and converse about their findings. After the development concluded on our site, we realized that we were skewing away from our initial assumptions with the progress that we made within our given time frame.

## Outcome Criteria

The criteria outcome for our service is based on two things. The first is being able to show users their listening data in real time based on their Spotify interactions. The second outcome of our criteria that we are testing is if users can see twitter updates from genre-related accounts that support their favorite artists.

## Risks

There are several risks associated with our project. The first risk is the safety of our users' data and accounts within our service. During some of the testing stages, we found that user info was leaked, and we had to rebound with a more secure mode of storing account information within our site. Another risk that we face is from Spotify themselves and them not being happy with the continued collaboration between our site and their service. Aside from this, we also face risks of login errors, expired accounts, and ensuring that correct data is being displayed for each user. Our idea is new and unique which means we also face outside pressures of rejection and the potential of our idea being stolen.

# **SYSTEM DEVELOPMENT**

## Concept Generation

Both of us have a common interest in music and wanted to focus our project on that topic because of the lack of innovation that’s occurred in the music industry in recent years. With the advent of music streaming at the turn of the century, a few large platforms, most notably Spotify and Apple Music, have dominated this market and have left little room for competitors. Because of the monopolistic control these few large companies have on the music industry; little innovation has occurred since. This is particularly noticeable today when the world has had so much innovation and growth with social networks and social media platforms, which have made little to no impact on how we listen to or share music.

The most notable social innovation within music that we’ve seen this century is the creation of Spotify Wrapped and Apple Music Replay. These are two features launched by their respective platforms that provide a summary of each user's music streaming data for the year including total minutes of music listened as well as your most listened to songs and artists. These features saw unprecedented popularity and according to MoEngage’s customer insights report, “Spotify's mobile app downloads increased by 21% in the first week.” Our team recognized this trend and the growing demand behind features that help socialize the music listening and discovery experience and knew we wanted to focus on that aspect of music.

We sought to conduct some experiments with current social features on music streaming platforms. In doing so our team realized that there is virtually zero integration of music platforms and social media platforms. We also found that this extends to communication between different streaming platforms. Cameron is an Apple Music user and Sean is a Spotify user, and they quickly realized that it is impossible for them to share and play songs with one another or even see what each other is listening to without creating an account on the opposite platform. These realizations planted the seed for our idea because we knew we wanted to provide music streamers, like us, the ability to see what users of different platforms are listening to and external tools to provide more context to their music.

With all the current limitations in mind, we wanted to solve some of those technical problems while also innovating the type of solution we could build. To do this we tried to ask ourselves what does this growing trend of music socialization that Wrapped and Replay introduced signify? A RockContent.com blog post explained, “After a period in which the digital became essential for the cultivation of personal and professional relationships, this year’s Spotify Wrapped brought possibilities for interactions and connections between users, artists and the brand.” At this point we knew we had to focus on the connection and community aspect that music provides because that is the essence of why people listen to music in the first place. Thus, we knew we wanted to build a platform, unprejudiced of streaming service, that lets people build followings and communities based on music tastes.

System Planning

With a general idea thought out, we then focused on researching and planning the technical requirements for building something like this. We knew we wanted to use at least one API to provide access to some data and we knew we wanted to make a web application rather than a mobile app because of our desire to make our service easily accessible.

After some preliminary research, we knew we wanted to focus on the Spotify API because they provide the most information and resources for developers. To start that process, we read in-depth all the Spotify documentation relating to our idea including reading their developer tools about their Web API and the process on how to authorize your app, request, and retrieve JSON metadata. Because of our team's inexperience with APIs, we also did some thorough research on the REST principles and architecture.

As far as incorporating some other form of music-related media into our idea, we knew we wanted to tap into some other existing network to pull additional data that could be of value to users. We decided that a news feed section, pulling related news updates based on music genres would be beneficial to a music community. Based on our research and analysis, we determined that the most viable option to provide this information would be by utilizing the Twitter API and pulling live tweets from music-focused accounts. We corroborated those conclusions with existing system planning and Gantt chart examples from prior IST440W class projects and broke down the categories of work required into front end, back end, middleware, and deployment with their respective subtasks for each section. These changed over time as our features and their feasibility shifted, but it provided a good foundation for our system planning.

With all of that in mind, we created our first project Gantt chart:

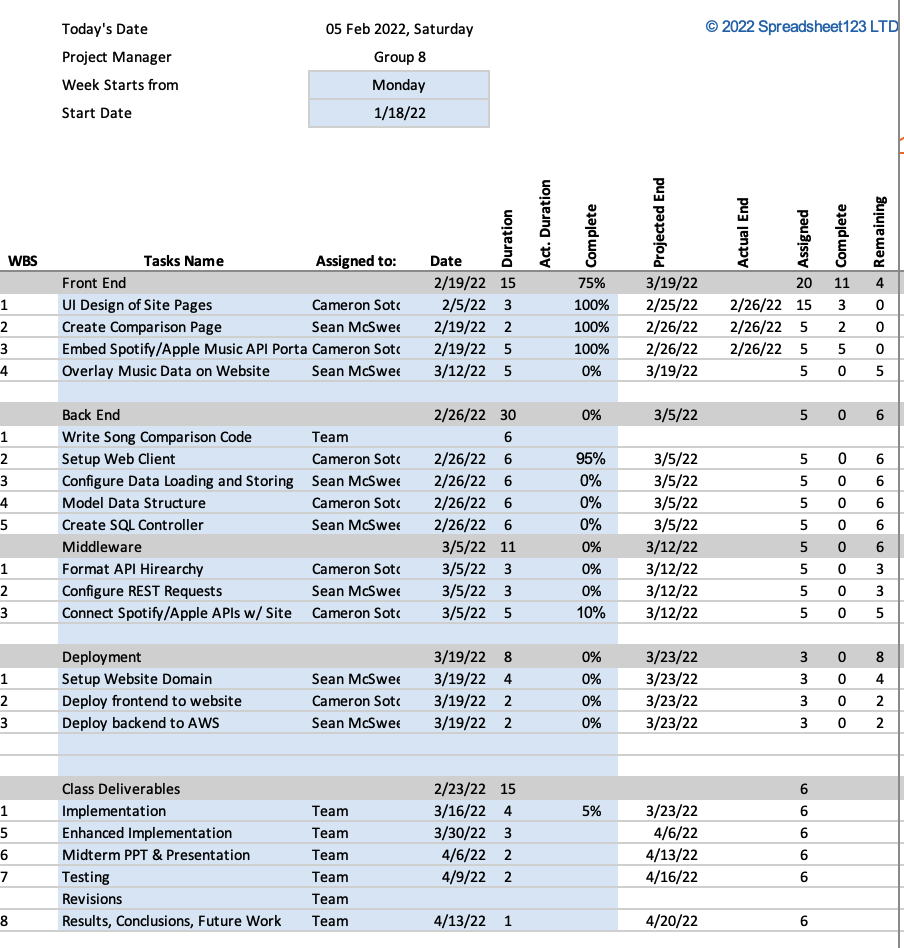


Table 1.1: The image above depicts our original project Gantt chart based on the work breakdown structure of our initial idea.

Our initial Gantt chart broke down our sections of required work into the front-end, back-end, middleware, and deployment, with an additional section for their respective class deliverables. The front-end phase featured the UI design, creation of the required webpages, and overlaying the data on the website. For our back-end phase, we initially intended the deliverables to include coding a song comparison program, setting up the web client, configuring the loading and storage of the data, modelling the data structure, and creating an SQL controller for our database. For the middleware, we knew if we wanted to utilize APIs, we had to format the API hierarchy, configure the REST requests, and integrate the Spotify API. We broke down the deployment into 3 steps, which include setting up a website domain, deploying the front-end to the website, and deploying the backend to a cloud service provider.

*Table 1.2: The figure below is our final Gantt chart with all the updated deliverables.*

Graphical user interface, table

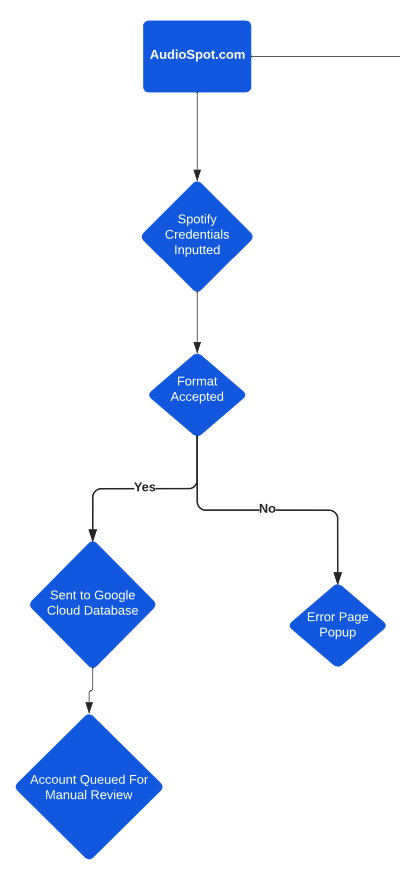
Description automatically generated

For our final Gantt chart, we removed all the tasks that were no longer feasible to the updated project such as “Create SQL Controller.” As the feasibility of our initial end-goal changed, we chose already integrated tools with the Webflow software to serve as our database. Because of the thorough research we had done on the technical requirements throughout the project, much of these tasks remained the same before and after shifting the idea.

## Functional Decomposition

The breakdown of the functionality of AudioSpot is quite straightforward with three main use-cases. The functional decomposition is illustrated in the below technical flow chart, which shows the technical paths taken by AudioSpot for the three main operations: inputting Spotify credentials, the song feed, and the music feed.

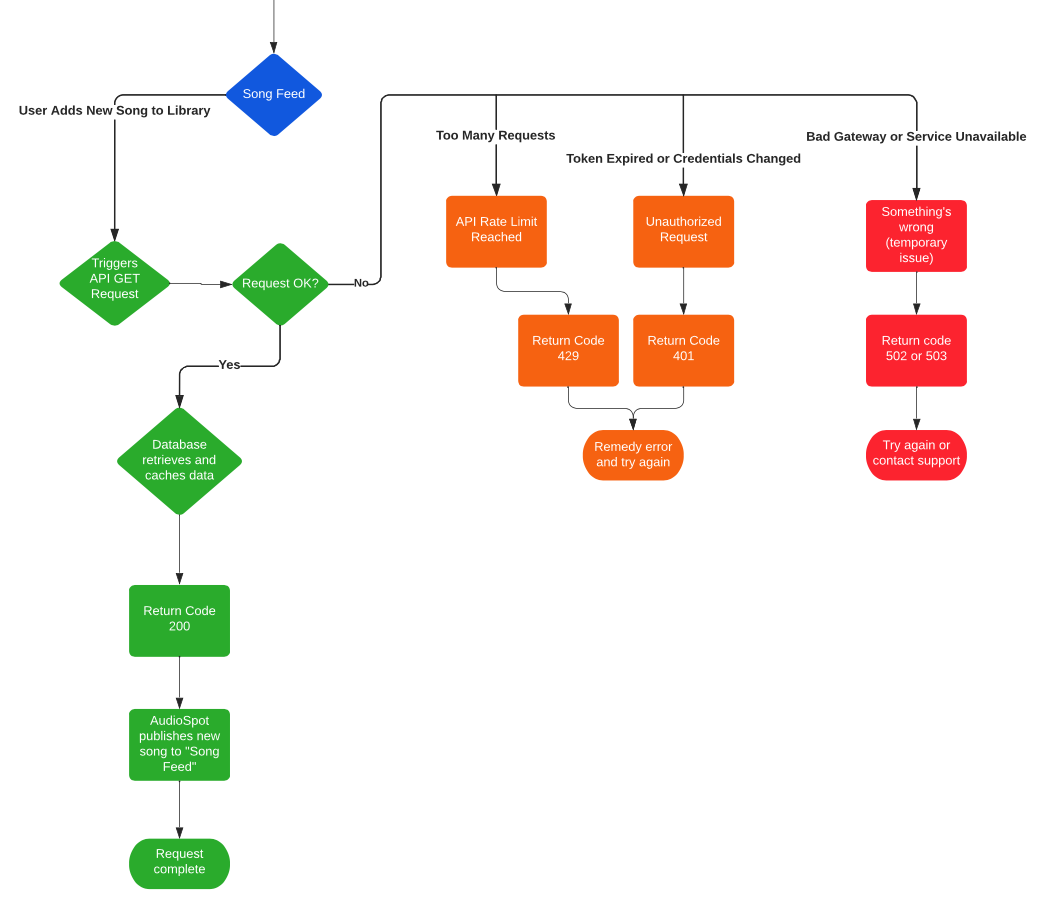
When a user visits the AudioSpot website there are three main tasks they can do. The first of which is inputting your Spotify account credentials to receive a detailed analysis of your music data, and the chance to be selected as the “Featured Fan of the Month.” This is the most straightforward path because there are not many technical steps required. A user inputs their Spotify account credentials and hits the “Submit” button. If they are in the proper format, a “Thank you! Your submission has been received!” message pops up, and the information is sent to our Google Cloud database, which we are notified of through email. Then the AudioSpot team pulls, views, and analyzes that user’s Spotify data manually. That manual review is where the AudioSpot team determines key trends and analytics behind their users' music data, which is used to select the “featured fan of the month.” If the format is inaccurate, an error page will popup.



*Figure 2.1: The image above depicts the technical flow of AudioSpot when a user inputs their Spotify credentials.*

**Populating the Song Feed:**

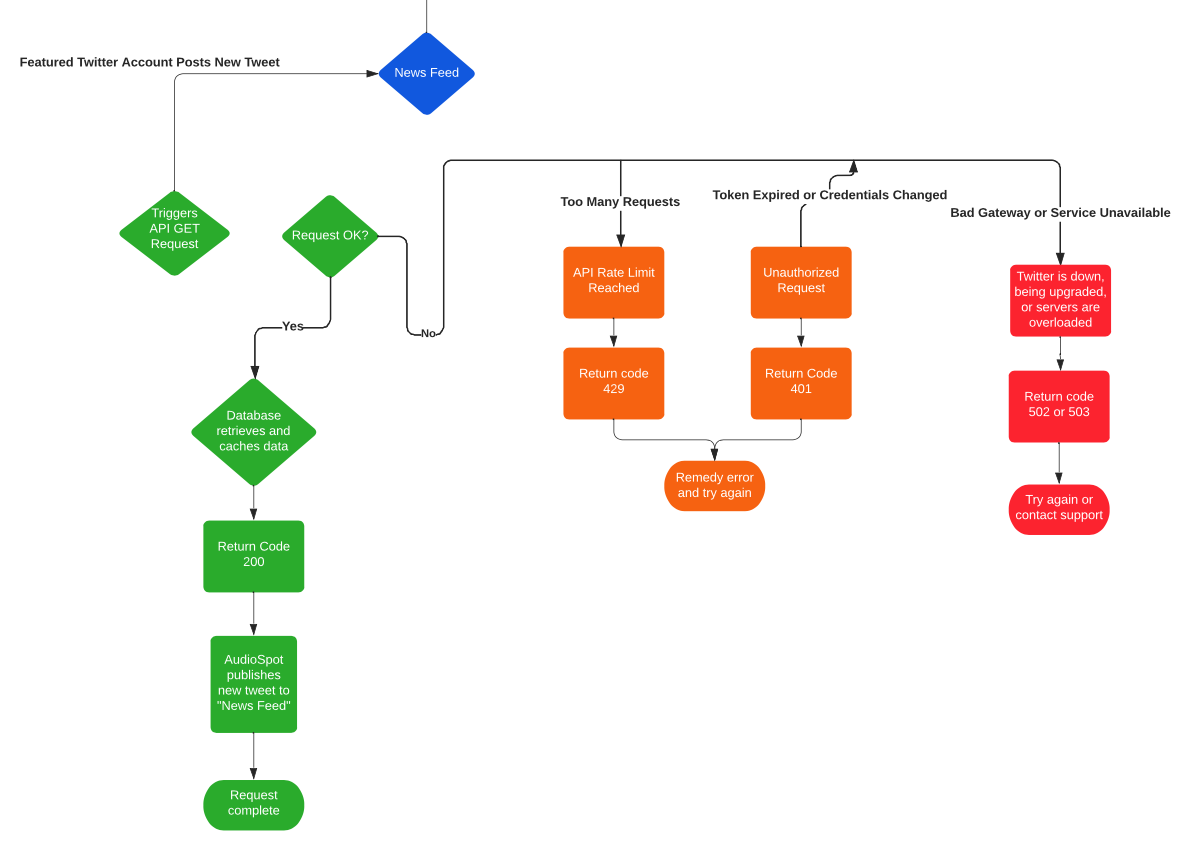
*Figure 2.2: The figure below depicts the technical flow of AudioSpot for the “Song Feed” feature, when adding a new song.*



For the song feed, AudioSpot pulls data from the “Featured Fan of the Month’s” Spotify account via their API. The user action that serves as the trigger for the API request is that user adding a new song to their Spotify library. When the user does that and the request is authorized and goes through, our database retrieves and cashes that new song data. Spotify returns a status code 200 to signify the data was pulled successfully, and AudioSpot instantaneously publishes that new song information to the “Song Feed.” If there is an issue with the request like the rate limit being reached, the credentials being changed, or an internal Spotify issue, the corresponding code of the error is returned, and the AudioSpot team is notified of that internally.

**Populating the News Feed:**

*Figure 2.3: The figure below depicts the technical flow of AudioSpot for the “News Feed” feature.*



For the news feed, the AudioSpot team chooses one or multiple Twitter music news accounts at a time based on the most popular genre of the “Featured Fan of the Month.” Each time that Twitter account tweets, it triggers our GET request to their API. If that request is authorized, our database retrieves and caches that new tweet data and returns a status code 200 to confirm its success. Then the AudioSpot site instantaneously publishes the contents of the new tweet and a link to the tweet on the “News Feed.” If there is an error with the request like reaching the rate limit, a token expiring, or a temporary Twitter issue, the data is not pulled and the AudioSpot team is informed of the respective error internally.

## **Principle Operation**

Operational/User Flow Chart

## 

*Figure 3.1: The figure above shows the steps for the two main use-cases a user would take on AudioSpot.*

*The first is for an average end-user and the second is for the “Featured Fan.”*

## 

The principal operation begins when a user navigates to audio-spot.webflow.io. When they visit the site, they can navigate the pages and view any desired information they want including the “Featured Fan of the Month’s” song feed, the news feed, as well as an about and contact page. If the user wants to, they can input their Spotify login credentials to us, so that we can access and analyze their Spotify data and for a chance to be chosen as the “Featured Fan of the Month.” When a user inputs their Spotify credentials, they will receive an email response featuring a detailed analysis of their Spotify streaming data. If AudioSpot determines they are active enough to be selected as our “Featured Fan” we will inform them of this and of all it entails and ask for their permission manually.

If a user is chosen as the “Featured Fan of the Month” their operational flow chart changes because they have additional responsibilities. After the user has inputted their Spotify credentials and are chosen as the “Featured Fan,” their job is to listen to music on their Spotify account as they normally would. Each time that user adds a new song to their Spotify library, our API request is triggered, and that song information is published to the song feed on the website. Also, as that user listens to genres, the Twitter accounts that are chosen for the news feed change based on that user’s patterns.

## Design Process

For our design process, we referenced a Khan Academy programming project planning guide, in which we determined the phases of our project as follows:

1. What do you want to make?
2. What technology will you use?
3. What features will it include?
4. But what features must it include?
5. How will you implement it?
6. What's your timeline?

Source: https://www.khanacademy.org/computing/computer-programming/programming/good-practices/a/planning-a-programming-project

1) We established what we wanted to build and what our goals with it were in the brainstorming and literature review stages.

2) The next step was figuring out what technology we would be using, which we determined with our research in the system planning phase above. We broke down all the technical requirements of all the technology our final project would require. We did so by researching the developer documentation for the Spotify and Twitter APIs and existing GitHub projects and YouTube tutorials using similar resources we would be using. Through this we were able to define and understand what utilizing these different applications and technologies would require.

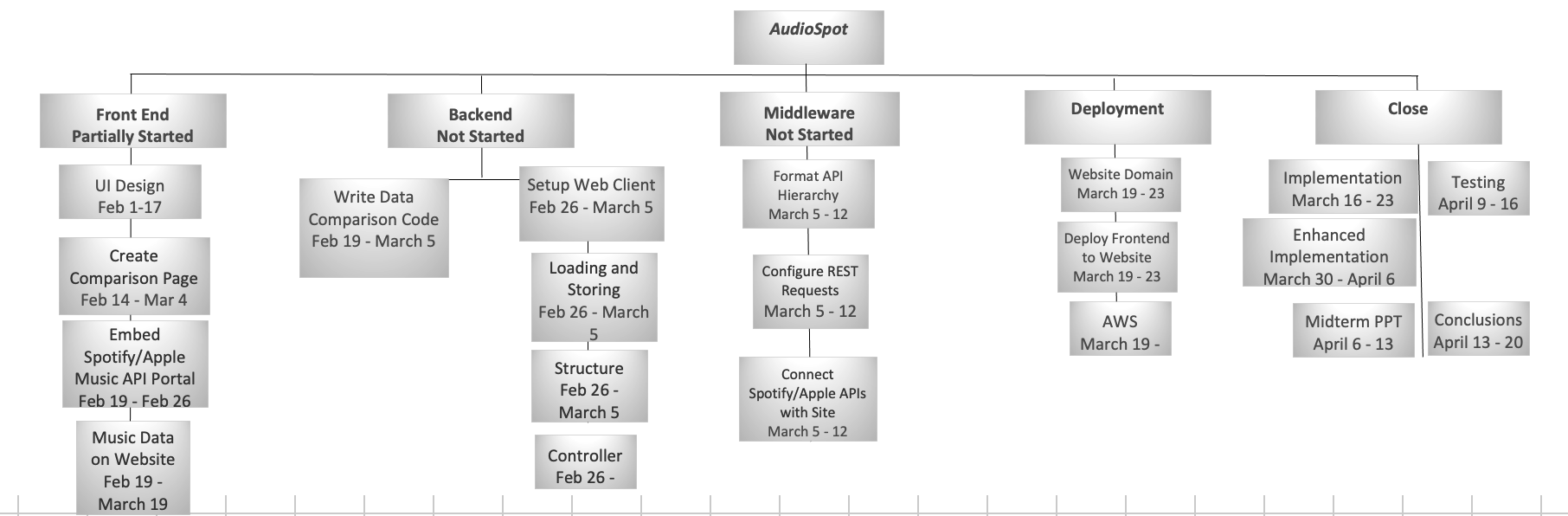
3) To determine the features, we could include in our project, we conducted an analysis based on the previous “technology” step and what we would be enabled to provide with each tool we were using. We came up with numerous initial features based of our technologies, which included:

* Pulling all available user-related metadata from each user’s Spotify accounts
* Pulling any tweet or tweet-related information from any public Twitter account
* Proprietary features built using this data that we would have to code ourselves
  + Comparing people’s music data to find similarities
  + Grouping people based on their music data
  + Generating recommended music based on users’ data
  + Linking Twitter accounts that a user would find of value based on their music data

4) Determining the features, we “must” include was an essential step for us because the features we could build with the APIs and technologies we were using is endless but determining what was feasible for this project was a tougher task. We were limited by several factors including time, money, our skill sets, existing knowledge, and unknown difficulties with using and integrating the technologies. We concluded that our final project had to include the following features:

* Some data from the Spotify API
* Some data from the Twitter API
* Some user-interface

5) For the implementation and timeline, we had to determine what technical end-products we needed to make our essential features work. Based on the work in prior stages, we knew our final project would require some sort of database to store the Spotify data, some code to pull data using the API, and a front-end website to display and allow users to access it all. We researched and studied the work breakdown structures of similar IT projects that utilize APIs to help base that process and required time for each step. We then created a list of every technical step we had to do for each technology we intended to use, cross-referenced that with the class timeline, and developed the following Work Breakdown Structure:

*Figure 4.1: The above image is our original project Work Breakdown Structure for our initial idea.*

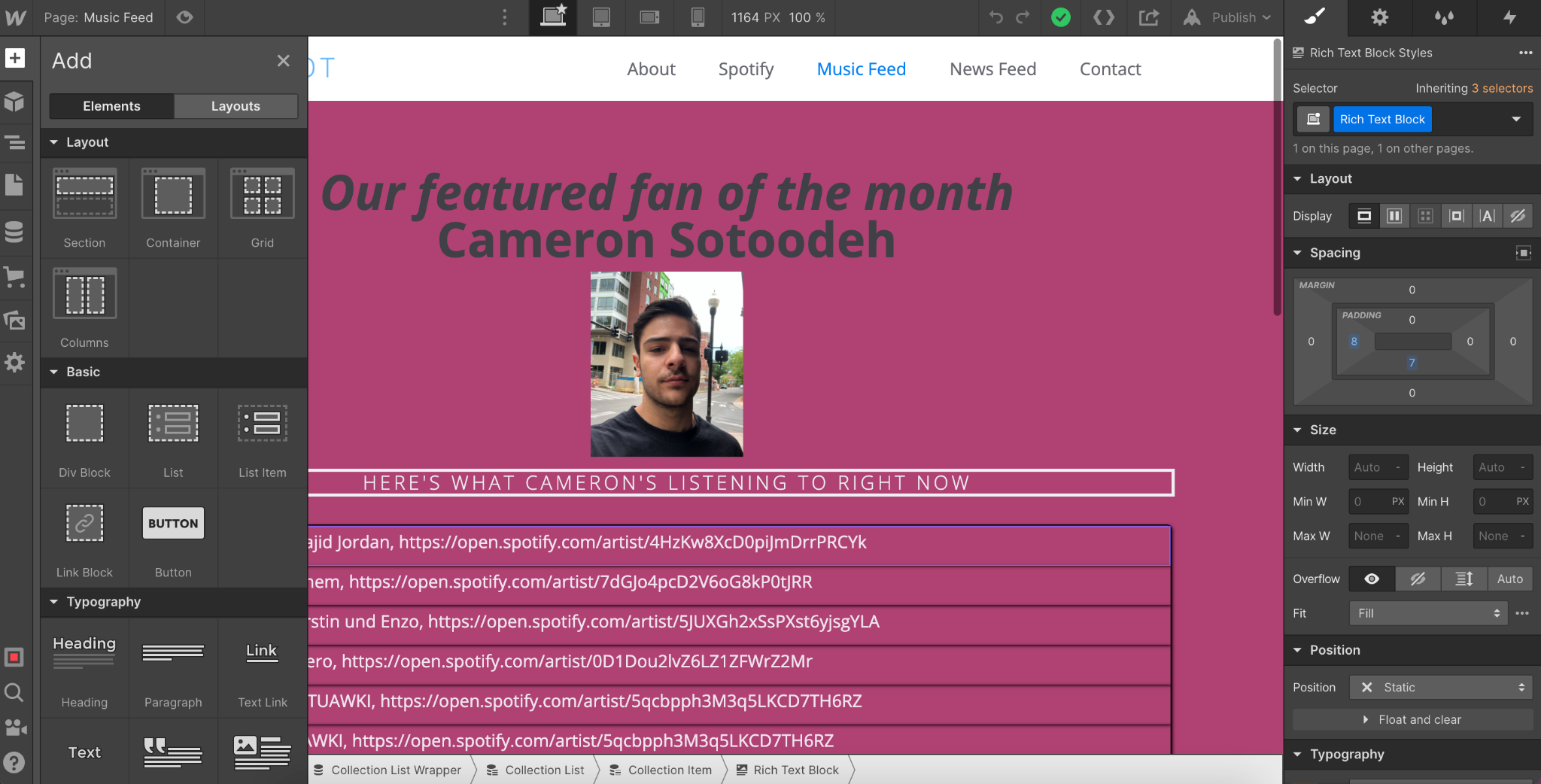
#### 

#### 

#### 

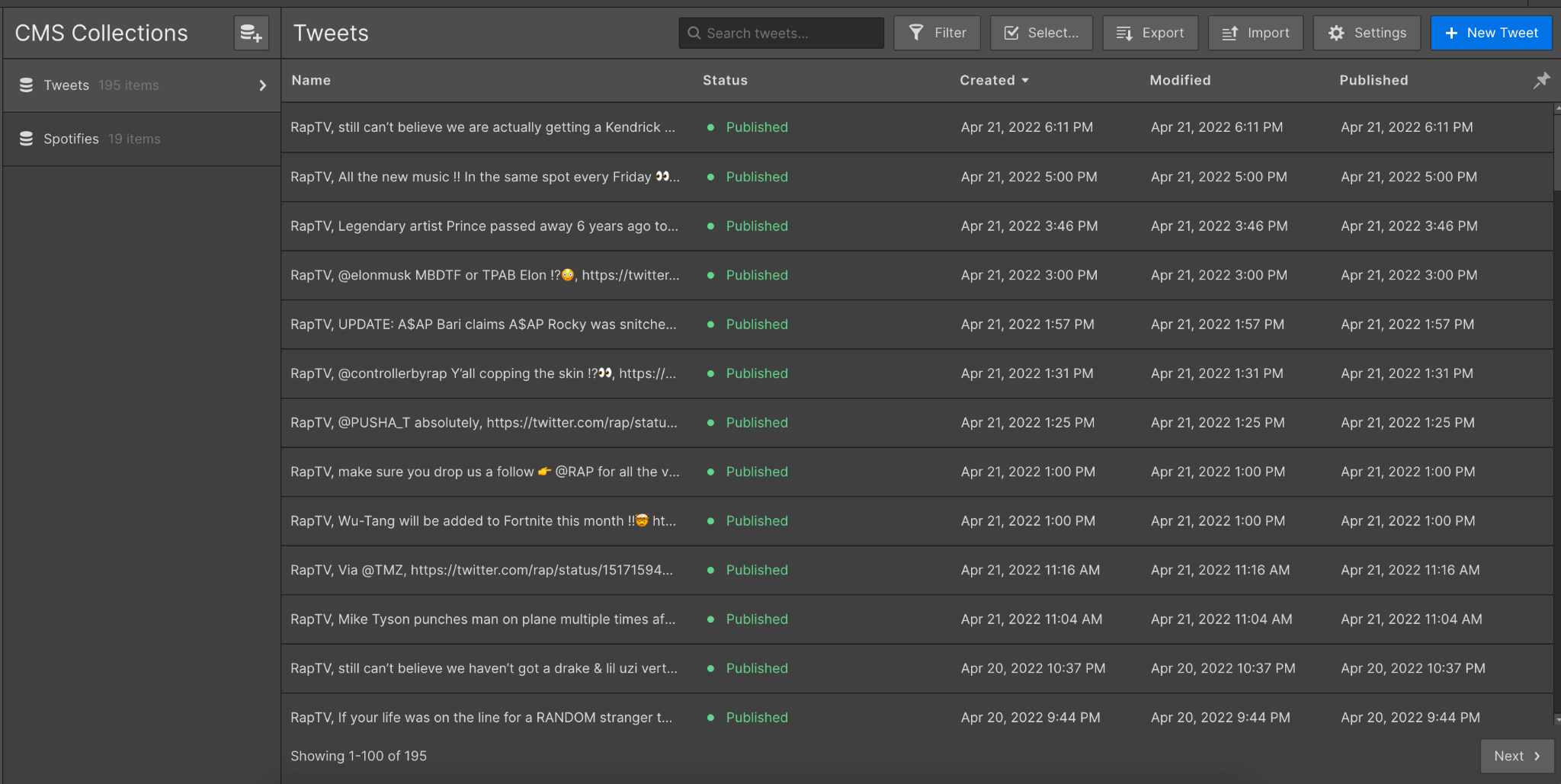
#### System Design & Choices from Design

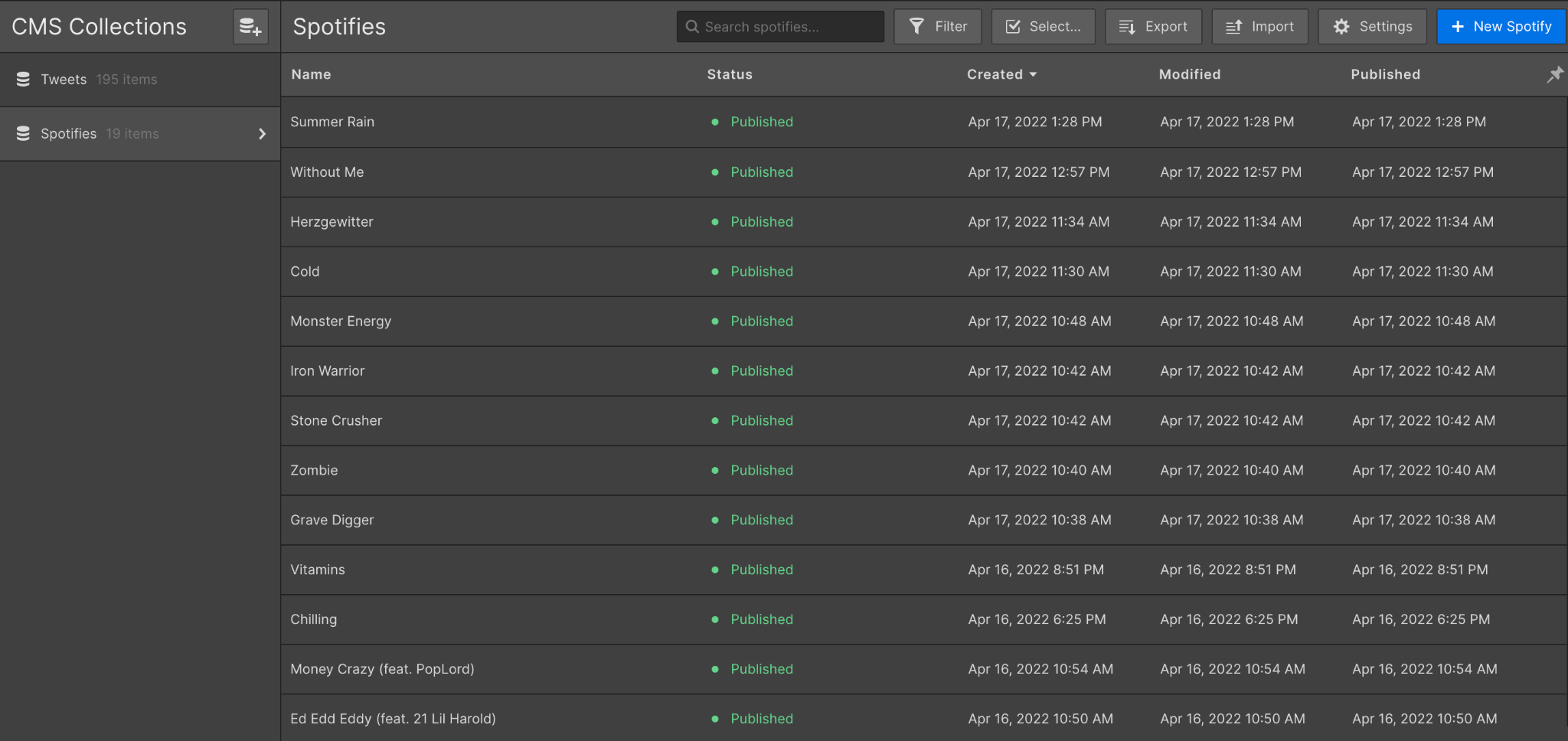
Because of our only two-member group and our limited skillset, we sought to make every aspect of our system design as simplistic and coherent as possible to ensure we could get it functioning properly ourselves. For the front end, we chose to utilize the visual editor platform, Webflow, to build, design, and integrate our website because it provided the most technical utility for our skill levels. We knew there would be several different aspects of our website design including live feeds displaying data from APIs, multiple input forms for Spotify credentials and inquiries, as well as external linking, all with a clear and pleasing user-interface. Webflow provided us with a wide range of comprehensive tools to build all of this.

*Figure 5.1: The image above displays the Webflow visual editor platform we used to design our UI.*

For the backend, Webflow provides a feature called CMS Collections, which serves as our database. We built a collection for each pool of data we were pulling, which is then dynamically stored and displayed, as we direct it, based on our needs for the site. We created a CMS collection for the Spotify song data for our “Song Feed” and a collection for the Tweet data for our “News Feed.” We then connect these collection lists with their respective sections and containers on the website to display the data.

*Figure 5.2: The image below shows the CMS Collection List for all our retrieved tweets from Twitter.*

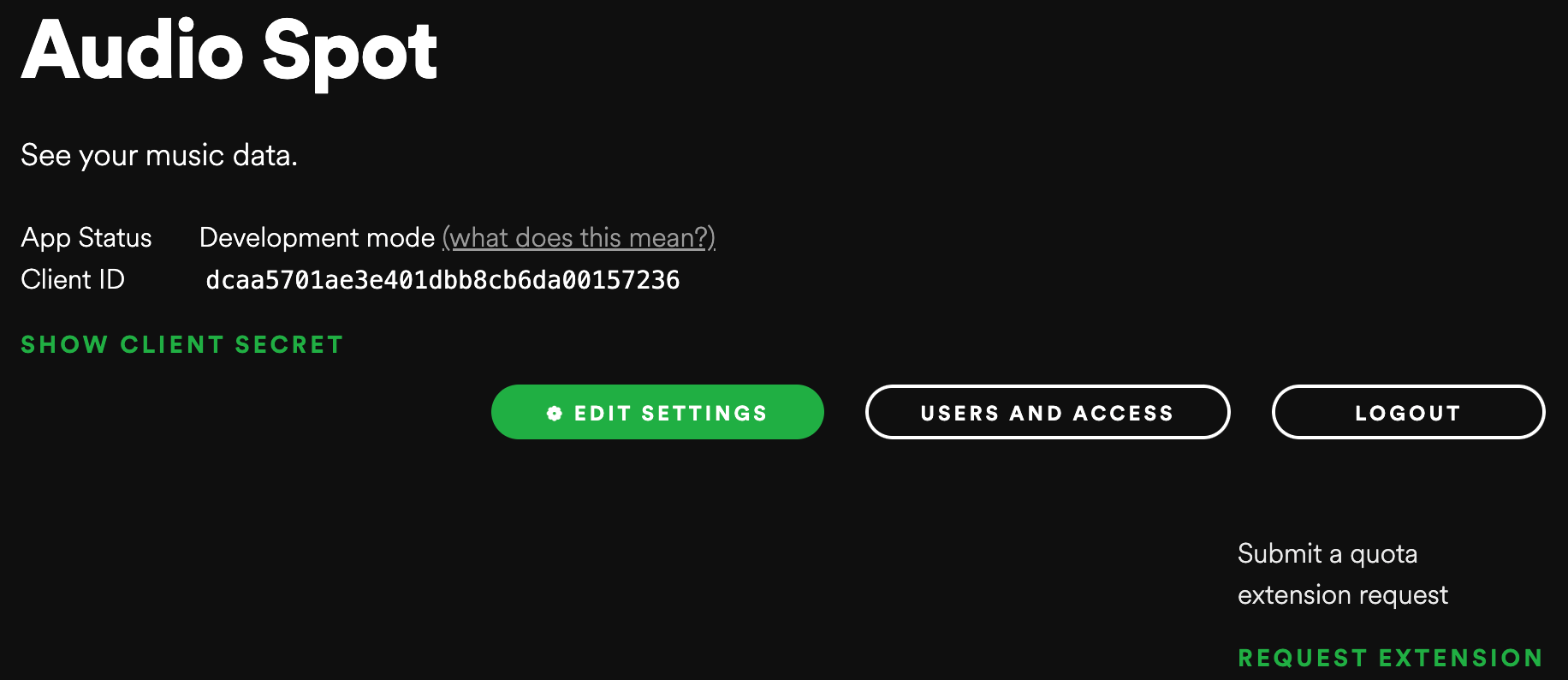


*Figure 5.3: The image below shows the CMS Collection List for all our retrieved songs from Spotify.*

For the middleware, we needed some method to make the API requests and successfully pull the JSON data from Spotify and Twitter. Getting this to function properly was the hardest and most lengthy stage of this project for us because of our nonexistent web and API development experience. Initially, we wrote a program in JavaScript, which we registered as a Spotify application to try and request data. With this, we were able to successfully authorize a request, request and return access and refresh tokens, and use those tokens in our GET requests to pull data. Where we fell short with this program was, we were unable to display the Spotify scopes and prompt a user login, which was essential to this version of our idea.

After our code for the API requests failed, we decided to take an alternative path more fit for our skill sets. We utilized the 3rd party workflow automation tool, Zapier, to run our API requests and pull the necessary data. We set up triggers that a selected end-user (our featured fan of the month) would execute on their own to trigger the API to request that data and publish it to our website. For the Spotify API, we designed it to request data each time the featured user likes a new song or adds a new song to their library. For the Twitter API, we select one or two Twitter music news accounts to follow, based on the featured user’s popular music, and each time that account tweets we pull the data and it’s published to our site.

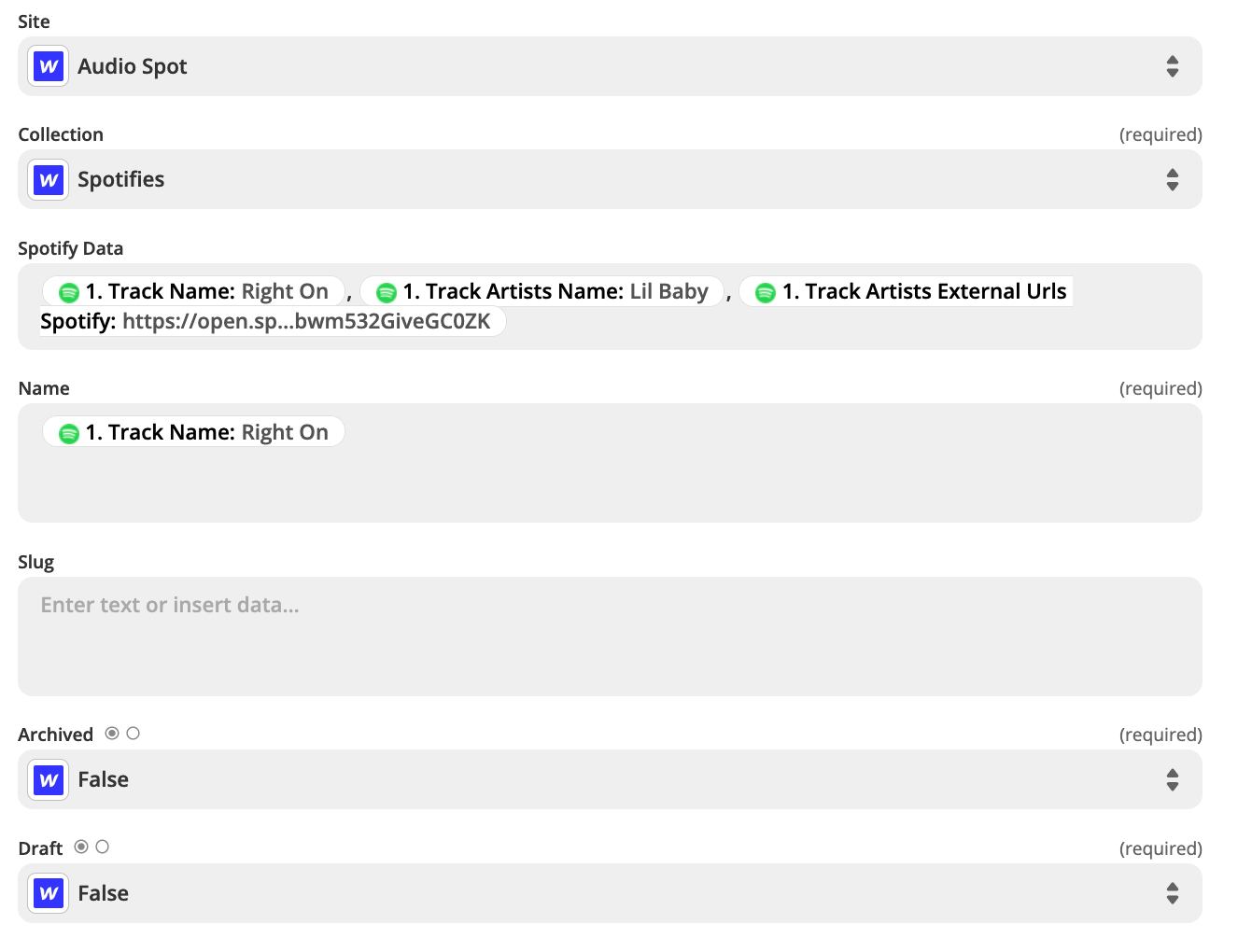
*Figure 5.4: The image below shows our registered application in a Spotify Developer Account.*



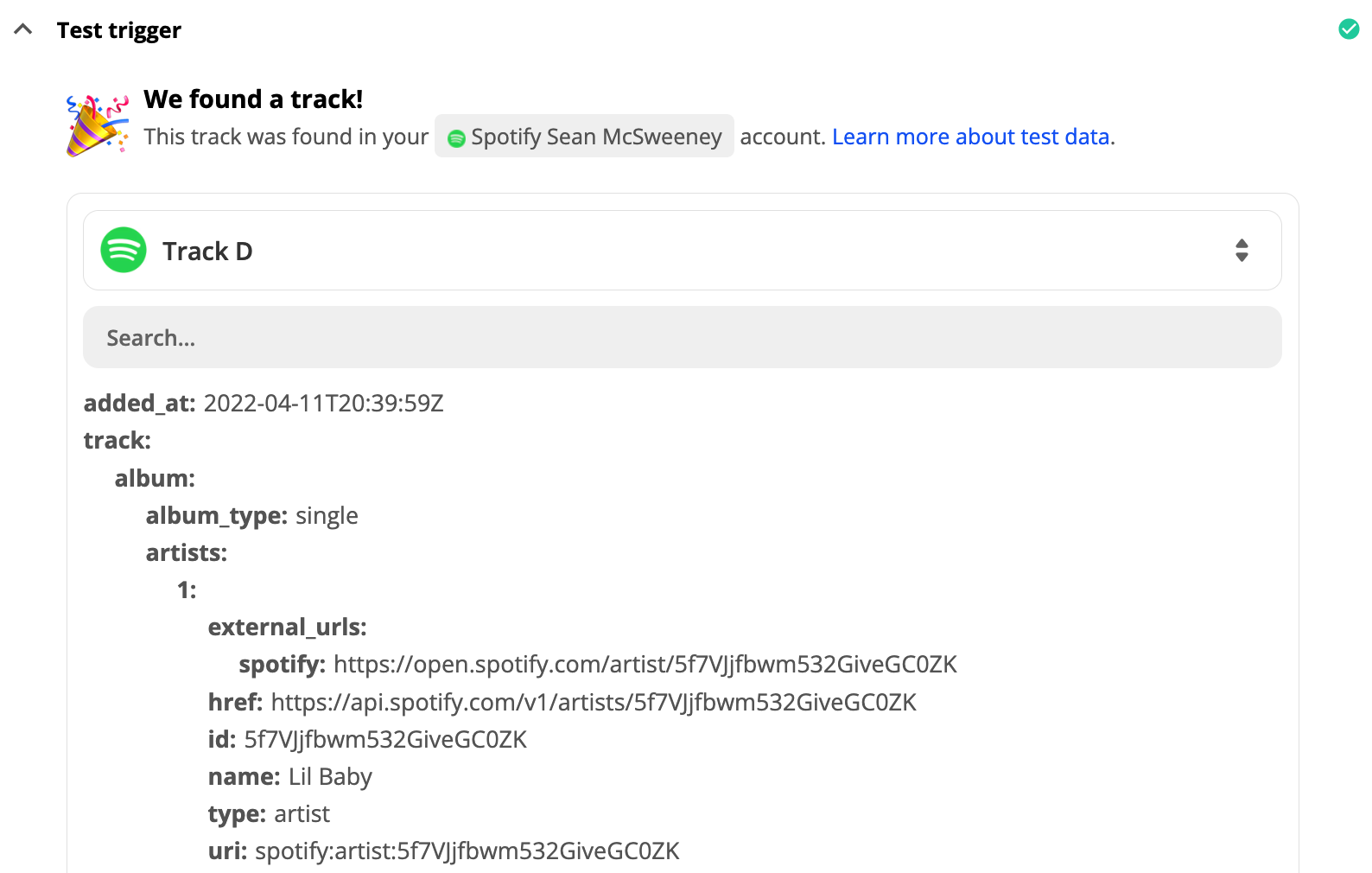
The be able to access the Spotify API at all, you need to create a developer account with them and register your application. To do so for the Web API, you provide the URL to your website and, in return, are provided with a ClientID and a Client Secret key, which you need to reference in your code to authorize and generate access tokens.

The following images show the details from the API integration tool we used, Zapier, to set up the API requests and pull data from Spotify and Twitter. When setting up the request, we chose to pull the name of the song, the name of the artists, and the Spotify URL so users can easily access the song if needed. Figure 5.5 depicts the types of metadata and their formatting that is available when pulling song data from the Spotify API.

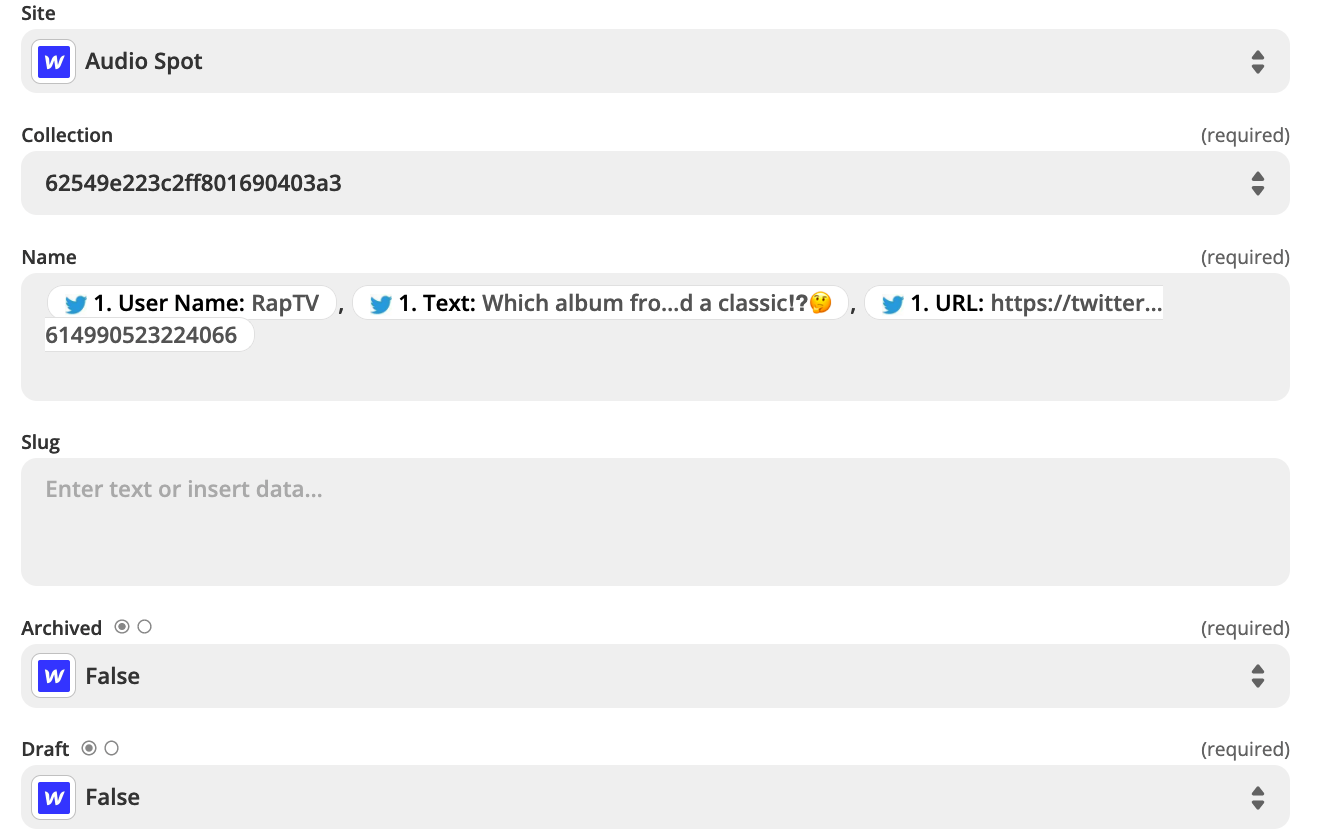
*Figure 5.5: This image depicts the data points we chose to pull from Spotify and where we are storing it.*



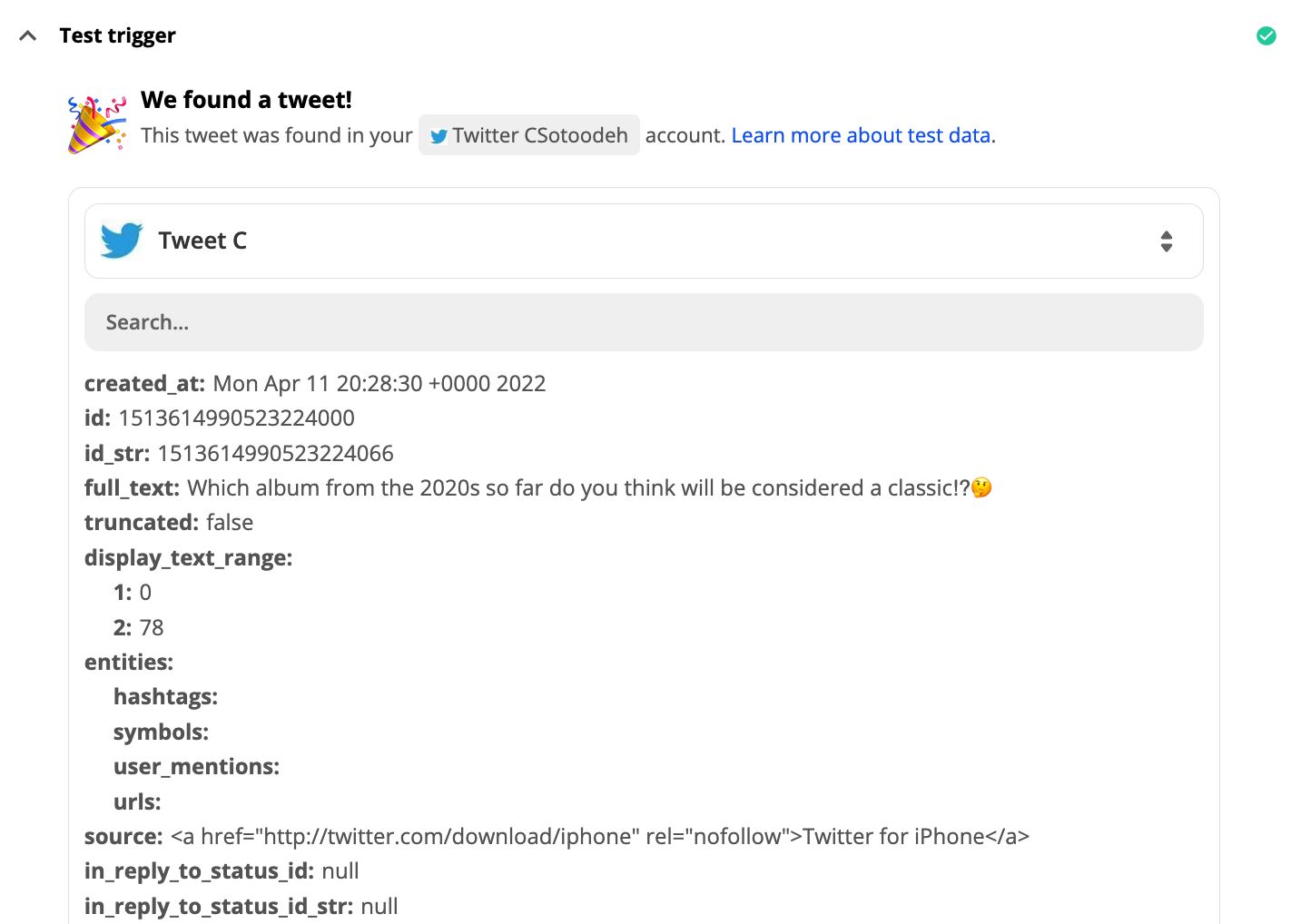
*Figure 5.6: The image below depicts the retrieved song metadata from the Spotify API.*



*Figure 5.7: This image depicts the data we are choosing to pull from Twitter and where we store it.*

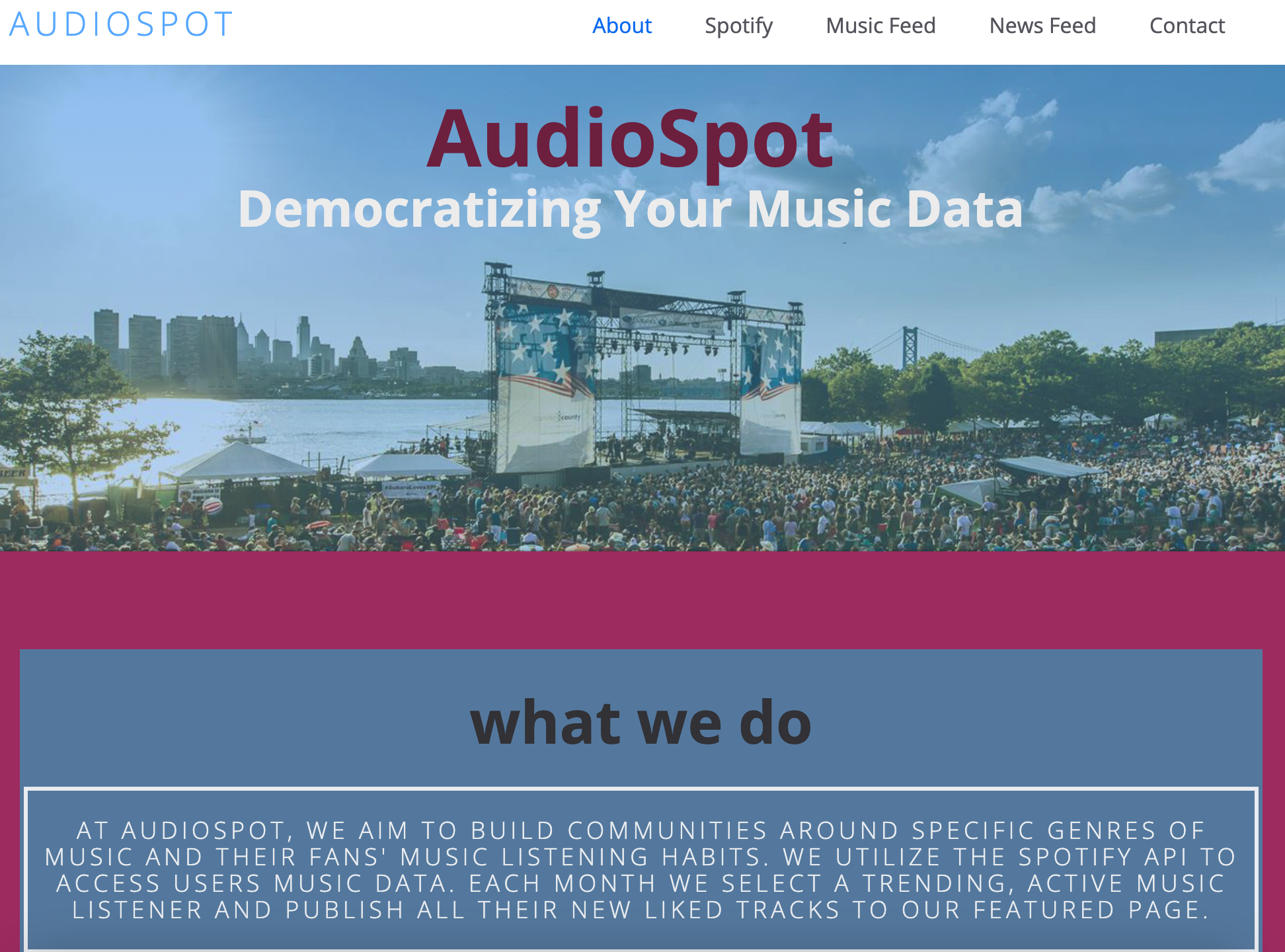


*Figure 5.8: This image depicts the metadata we receive from the Twitter API.*



For the deployment of this web app, we also utilized the Webflow tool because it is where we designed our site, stored our data, and integrated the API requests with. We referred to research done on the psychology of color for website design to base our color scheme and referred to similar websites to base our formatting off. Our goal with this was to design the most cohesive and user-friendly UX as possible to ensure users would navigate to all the pages on the website. After our UI/UX design was complete, we published our website using Webflow’s hosting tool, and is now accessible for any user on any browser.

*Figure 5.9: The image below is a screenshot of the home/about page of our website.*



For the home page, we wanted it to be as simplistic and aesthetically pleasing as possible to draw the user in from the start. We include a photo banner of a music festival to emphasize our purpose as a company, which is bringing people together over their music tastes. We also provide our slogan, “Democratizing Your Music Data” because providing users access to detailed insights about their music data whenever they want is AudioSpot’s mission. Lastly, we provide a brief description of what AudioSpot does to provide some context about what the site is, but not too much to intimidate or bore the user.

Graphical user interface, website

Description automatically generated

Figure 5.10: The above figure is a screenshot of the “Spotify” page on the AudioSpot website, which is where users enter their Spotify login credentials.

We position this “Audio Screener” tool as the first main page in the navigation bar because we want to draw users to it, which is why we chose a bright orange color for the background here. We also chose not to combine this, or any of the features, on the same page to emphasize simplicity and organization, and to not scare a user away from a section in case it’s too intimidating. On this page, users will input their Spotify credentials so we can access and provide them with a detailed analysis and insights of their streaming data. We also store that user’s data and use it as the basis of our “Fan of the Month” and genre selections by cross-referencing everyone’s music and determining key trends and what people want to see.

Graphical user interface, text

Description automatically generated

*Figure 5.11: This image is a screenshot of the “Music Feed” page on the AudioSpot website, which features a live list of all the new songs a “Featured Fan” adds to their Spotify library.*

The next page is the “Music Feed” where users can go to see every new song the “Featured Fan of the Month” is adding to their library in real time. Based on the analysis of all the users’ streaming data, each month the AudioSpot team selects one Spotify users to be the “Featured Fan of the Month.” This person is selected based on how active they are on Spotify, how often they are adding new songs to their library, and how trendy their music tastes are. When a user is selected, they are notified of the terms of the featured fan “Music Feed,” and we update our integrated Spotify API to their account credentials. We set the API to trigger a song request each time the featured user adds a new song to their library, at which point our website automatically published the name of the song, the artists, and a Spotify link to it.

Text, calendar

Description automatically generated with medium confidence

*Figure 5.12: This image is a screenshot of the “News Feed” page on the AudioSpot website, which features a live list of tweets from (in this example) a rap music news Twitter account.*

The next page is the “News Feed” tab, which features a live feed of music-related news, pulled from a selected Twitter account, based on the featured fan’s genre preferences. When selecting a new featured fan, the AudioSpot team determines the most favored genre of that user, and tailors the news feed to cover music-based news related to that genre. For the above example in figure [], that user’s most popular genre of music was rap, so we set the API request to pull data from the popular rap music news twitter account, RapTV. Each time this account posted a new tweet, our API GET request would be triggered, and the text of the tweet, along with the account name and a link to the tweet, are automatically retrieved and published to the website.

Graphical user interface

Description automatically generated

*Figure 5.13: This image depicts the “Contact” page on the AudioSpot website, which features a submission form where users can input their contact info and a message about any inquiries.*

The last tab is the “Contact” page with a simple submission form where users can input their name, email, and a message to send to the AudioSpot customer support team. Webflow sends us each submission here directly to our email where we can view everything they inputted. Considering AudioSpot is still in the early stages of development, this contact page is a crucial feature of our website because it provides users with an opportunity to give us feedback on the UX/UI, the current features, areas of improvement, and just general comments, which is essential to our team if we want to continue to develop AudioSpot and see it succeed in the long term.

Testing

Because of the integrated design of AudioSpot, testing of the functionality was quite simple and is broken down into two components: Logical and Technical errors.

**Logical (End-User Errors):**

Since virtually all the functionality of AudioSpot is done by the AudioSpot team on the backend, there is little room for error by an end-user. If a user inputs their Spotify credentials to us accurately and is chosen as the “Featured Fan of the Month,” then the only remaining actions required by them are adding new songs to their Spotify library. We’ve tested this with multiple users and Spotify accounts and multiple types of devices including the Spotify website on a PC, Mac, iOS, and android as well as the Spotify application on Android and iOS. These are our findings:

Case 1: The “Featured Fan of the Month’s” account is jeopardized.

In this scenario, an outsider gained access to the Spotify account of our “Featured Fan of the Month.” This could have been a hacker or a friend who had access to the user’s Spotify account at some point. This unknown person was playing songs of their own on the user’s account. They also added new songs to the user’s library, which triggered our API request and caused AudioSpot to post several songs unknown to the user.

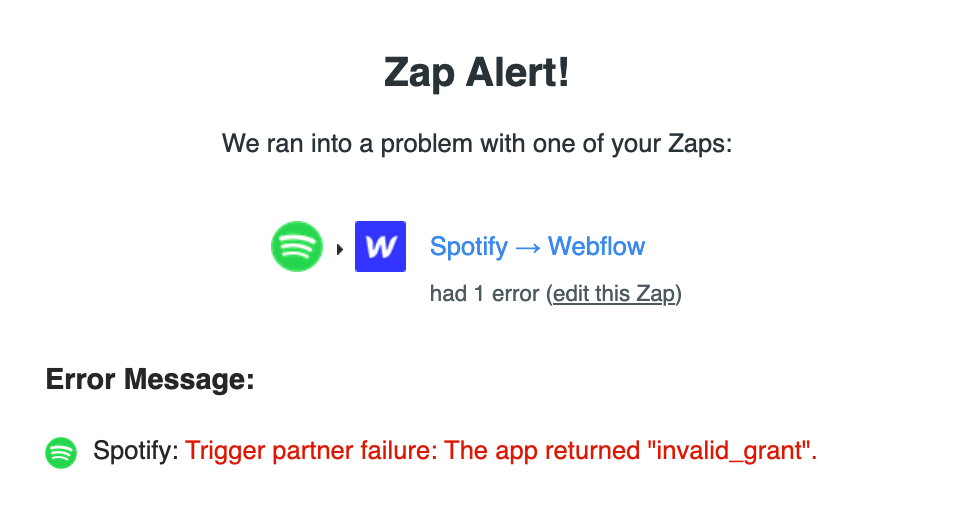
*Text

Description automatically generatedFigure 6.1: The image below is a screen shot of the AudioSpot song feed with all the unknown songs that the unauthorized user was adding to the “Featured Fan’s” library.*

Case 2: The “Featured Fan of the Month” changes their Spotify login credentials.

In this scenario, the “Featured Fan” changed their Spotify login credentials, which prevents us from being able to access their data. In this case, the user was either unaware that it would cause AudioSpot to lose access, or they didn’t want us to have access anymore. Either way, when the user does this, we are notified directly by Zapier of the status error that Spotify sends when making unauthorized requests, which is a status code 401. The song feed on the AudioSpot website will not change when this error occurs, but it will no longer have new songs to post and will remain the same until the issue is fixed. When this occurs, Zapier immediately notifies us, by email, of the issue.

*Figure 6.2: The image below shows the invalid grant request returned by Spotify, which Zapier notifies us of.*



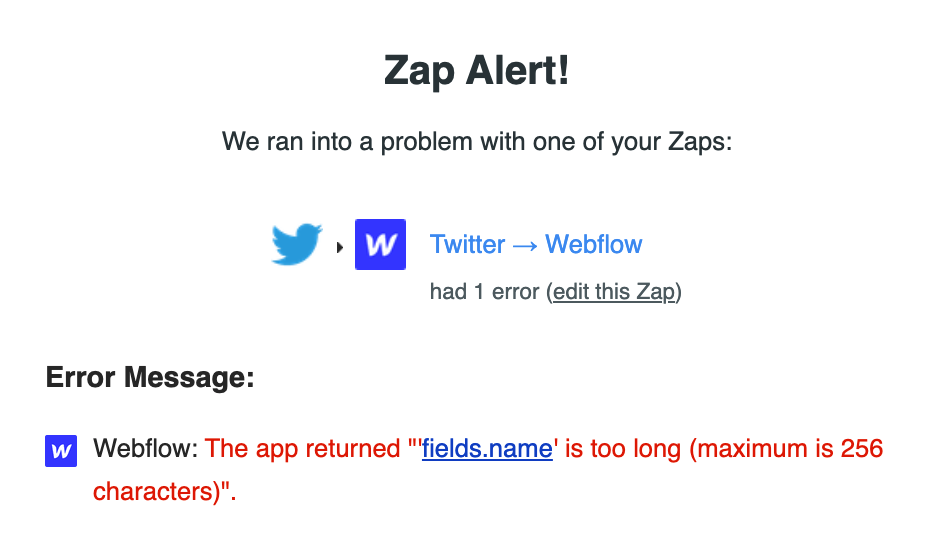
The only remedy we have to this issue is to disconnect the Spotify API so that unknown songs are not continued to be posted while the user’s account is still jeopardized. In this scenario, we would notify the user immediately of their jeopardized account and require them to change their password to regain control and kick out the intruder.

**Technical Testing:**

Case 1: The tweet from a selected Twitter account is unretrievable.

In this scenario, one of the tweets from a selected Twitter account, for our news feed, is unable to be retrieved. Because of the set formatting rules of our CMS collection list on our website, there are certain tweets that cannot be displayed. An example would be if the tweet or some of the metadata being pulled from that tweet is longer than the set character limit. In this case, the “name” field that was being pulled, was longer than 256 characters, and our website could not store or publish it.

*Figure 6.3: The below image shows the error from Zapier, which signifies that the name field is too long for Webflow to handle.*



*Figure 6.4: The below image is from the Twitter API documentation and explains the technical issue encountered that comes with the 120-error code.*

Graphical user interface

Description automatically generated with low confidence

This is a technical error within the AudioSpot middleware because our collection database is formatted to only hold up to 256 characters. For purposes of this project, we were unable to create a larger collection list to hold larger tweets like the one in Figure 6.4, which is a necessary patch we will have to make in our future development. In this case, this tweet is not retrieved by our request, and we are unable to publish it to our website.

# **CONCLUSION**

## Changes in Development

Through the entirety of our project, we suffered through numerous changes in development. The main change that we noticed recurring was the base that our platform would run on. For the sake of this project, we wanted something that would suffice as a website that offered tools to our users. In doing this, we played around with different website builders, coding languages, and other outlets to try and piece our information together.

Another change in development that we faced was with UI design. Initially, we thought that we organized our website in the proper way. However, upon meeting with our professor, he thought it would be worth the time to look over both the color spectrum of our design, along with the organization of our website tools. This in mind, we went through lengths to create a friendly UI experience that was both visually and mechanically sound.

## Challenges

After the research is where we hit most of our issues upon the realization that we would most likely have to code extensively, if not everything in this website. Neither one of us had any proper training on web development or the integration of Spotify’s APIs which provoked a long, frustrating journey of seeking this knowledge.

We knew what we wanted to do, we knew that there was a way to do it, but we had trouble putting the two together. We worked alongside computer science majors and other people who had a lot of experience developing websites and or programming things in general. They too said that our idea was very possible but led us to dead ends. It feels that we had to start our project from scratch nearly every day to come to where we are now.

Through many different attempts, we finally found success with webflow. A website building platform that allowed us to make the API integrations that were needed to make our website function how we wanted. After this, we followed forums, videos, and other sources of knowledge to sharpen our skills with the interface and perform to the best of our abilities. The website we have now is not exactly what we envisioned from the beginning, but we are proud in both ourselves and our work that we were able to put out something that does something similar to our initial goal.

To summarize, the main challenges we faced were with the timeline, our level of web development/integration skills, and the strength of our team. We feel that with a longer time frame and more group members, simple struggles that are faced could have been handled in a quicker and more efficient fashion.

## Lessons Learned

Firstly, we believe that we should have chosen an idea that would not require as much integration and hard code that we had to do with this project. It seems we were overcomplicating the complexity requirements of the project and spending less time on accepting the fact that we did not have the proper skillset.

Another thing that we learned was the importance of focusing work on the more important pieces of the project instead of over analyzing small details. With the help of the professor, we were able to pivot our direction so that we made large developmental changes for the better in a short amount of time. In doing this, we were able to knock out those small challenges along the way.

On a more positive note, through our extensive research and learning, we were able to become more acquainted with web design, development, and integrations. This is something that we had little to no experience about but after the course of this project, we now have a decent understanding of these things. It seemed like if we were making slow progress through the semester, we were still picking up useful information to use later in life in our careers.

On an organizational note, we learned that it was helpful to plan out our deliverables, especially when it came down to the integrations we had to create. It gave us both a time frame and task which we were able to follow. Having said this, we still needed to update our deliverables as we went through developmental changes which caused confusion and frustration at points of our journey.

# 

# **FUTURE WORK**

While our version of AudioSpot is in its early stages of publication, we are eager to continue with this journey that we have started. We see points of improvement that can harvest profits, fan interaction, social media functions, and other improvements. We envision AudioSpot being exactly what it is titled, The spot for Audio. With the correct management and careful progression of our site, we think that it can grow to become one of the most visited sights in the world and offer a unique user experience.

For Audio Spot’s future, we see a large amount of investments that would drive up customer satisfaction and possible profit. One of the first long term ideas that we had was for the website to generate some sort of revenue. This could be possible in several different streams such as artists being able to promote themselves through our website, or a paid subscription-based account that would provide users even more data and insights about their listening history.

Another possible revenue stream would be advertising through our website for companies that are either already involved in the music industry or just general companies that would be interested in seeing their banner ads posted.

To optimize our website to our best abilities and drive customer satisfaction up, we think that AudioSpot should become a social hub for music listeners alike to be able to interact and view each other. This could come in the form of a web and app based social media platform so that you can see what your friends, family, and others are listening to as well as the most popular music listening trends of the day, week, and year. With this extra layer of connection between music listeners and their peers, users would be able to discover new artists, songs, albums and more.

With AudioSpot only being accessible to Spotify at the moment, there is a large number of people, such as Apple Music listeners, which cannot use our website. We plan in the future to further our integrations with other streaming services such as Apple Music, YouTube, Soundcloud, Pandora and more. This way, AudioSpot will act as the main platform responsible for tracking music data and allowing connectivity, sharing, and discovery through music.

# 

# **REFERENCES**

*All the listed references below were active and accessed as of April 22, 2022.*

1. https://spdload.com/blog/7-music-app-ideas-for-founders/
2. https://theappsolutions.com/blog/development/develop-spotify-like-app/
3. <https://www.digitalmusicnews.com/2021/03/30/most-popular-music-streaming-services-2021/>
4. https://tracxn.com/d/trending-themes/Startups-in-Music-Social-Network
5. https://developer.apple.com/documentation/applemusicapi/
6. https://developer.spotify.com/documentation/web-api/
7. https://www.chosic.com/playlist-generator/
8. https://spotalike.com/
9. <https://www.workingdata.co.uk/how-to-cross-reference-spreadsheet-data-using-vlookup/>
10. <https://prezi.com/p/lcjz4ubhrpgx/spotify-social-media-presentation/>
11. <https://www.slideshare.net/skempi/skempi-music-streaming-platform-artist-presentation>
12. <https://slidesgo.com/theme/music-streaming-services-market-analysis>
13. <https://www.youtube.com/watch?v=CKq8yFEzoyI>
14. <https://www.youtube.com/watch?v=c5sWvP9h3s8>
15. <https://www.youtube.com/watch?v=FNeZsxb4UDY>
16. <https://www.youtube.com/watch?v=0MXamRhLtLY>
17. <https://www.behance.net/search/projects/?search=social%20music%20app>
18. <https://www.atlantis-press.com/article/25329.pdf>
19. <https://www.daitm.org.in/wp-content/uploads/2019/04/Gr.-11Melody-CONCLUSION.pdf>
20. https://www.researchgate.net/publication/233783701\_Music\_Discovery\_with\_Social\_Networks
21. https://www.businessinsider.com/guides/tech/best-music-streaming-service-subscription
22. https://www.phonearena.com/news/these-are-the-most-used-music-streaming-services-in-the-world\_id137933
23. https://www.orientsoftware.com/services/database-development/
24. <https://www.upwork.com/resources/data-base-technology>
25. https://variety.com/2021/music/news/global-song-streams-up-25-percent-mrc-international-music-study-1235075965/
26. https://www.transparencymarketresearch.com/music-streaming-market.html
27. https://dribbble.com/tags/music\_streaming
28. https://www.docsity.com/en/ppt-for-presentation-on-online-music-website/7638657/
29. https://courses.cs.washington.edu/courses/cse403/08wi/projects/sds/web\_music\_player.ppt
30. https://prezi.com/p/rmnucjpslwi8/music-streaming-presentation/
31. https://www.youtube.com/watch?v=eVb1gOvLpRo
32. https://www.youtube.com/watch?v=lhE52mFS6gQ
33. https://www.youtube.com/watch?v=7hPzoc5UgAg
34. https://callmeismaelbarry.com/studio-ios-project
35. https://github.com/akessela/Sparkify-Capstone-Project
36. https://www.oliversyreyes.com/raze-case-study