Group 20 - Robert Acebedo, JJ Whitehurst, Andrew Jung

1100 Intro to ITWS

**Term Project Writeup**

**Description of Project**

Our project, named Wrapped Now, gives the user a webpage that allows them to view their Spotify listening data in a customizable infographic. The website uses the Spotify Web API which allows us to retrieve data from the Spotify database. The data retrieved is shown to the user in an infographic and the user can change the parameters such as the time frame for the listening history period and the top items shown. These top items range from the top songs, top artists, top genres, and top albums. The user can also go to the stats page for a longer list of top songs for the past year. These are the basic functions of our term project.

**Proposal Summary**

The idea for this project came from our initial brainstorm on term projects we could do. Initially, we wanted to do a Spotify player that would bypass ads. We switched gears to create a listening history graphic generator as that would better fit the term project guidelines and also be legal. Our main competitor and inspiration for our term project was Spotify Wrapped. Spotify Wrapped is an end-of-the-year recap of your listening history for that year on Spotify. We wanted to improve on this by allowing users to access their listening data in a nice-looking infographic any time of the year. The Spotify Web API gives documentation for allowing the developer to gather the user’s top items in a time specified. We also wanted to areas 1 and 2 as our focus and outlined a basic IA. All of these items later changed with time as we progressed in the project.

**Project Plan**

Our plan for this project consists of first, its purpose. It will allow the user to retrieve their top artist and top songs from a period they choose. This will be like the official Spotify Wrapped but unlike the official one, this can be used anytime throughout the year.

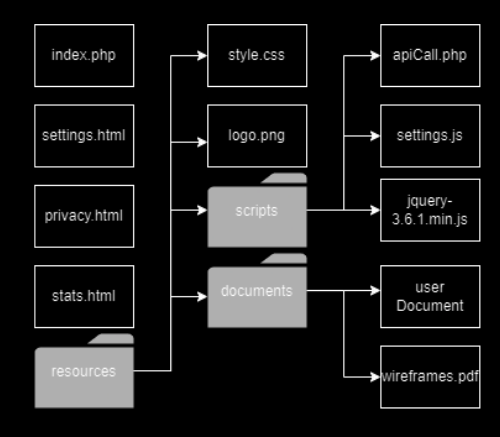
Something that we have discovered was the [Spotify media kit](https://newsroom.spotify.com/media-kit/). This is an official resource that is provided by Spotify giving access to all the graphics they use for all Spotify products. We will use this resource as inspiration for designing our wraps. This will help us in designing the front end and give the user the same feel as the Spotify Wrapped.

First, we will lay out HTML for the main/landing page and settle on a color scheme to use. The color scheme and page layout will be consistent throughout all pages. We then need to review the Spotify API documentation as it will be utilized to get the data from the user.

When we worked on our wireframe for the page layout we decided to have 4 pages. In our proposal and scope write-up, we had our IA set up only to have one .html file. Including a setting page, a privacy policy, and a final product page. The settings.html page correlates with the 3rd wireframe. This will be where the user chooses the preferences for the final product such as having top artists or top tracks and the listening history length. The product.html page will be the product our project is creating and will put all the information we get from the Spotify database to show the user. The privacy.html page will just be our privacy policy for those who are interested. The IA also has the documentation from lab 7 which includes the user personas we are targeting and the wireframes for our site.

The assignments and deadlines for this term project are presentations on April 22nd, and final term packages, reports, and peer reviews on the 22nd as well. We aim to complete these with sufficient time before at the latest to make sure we can review everything before it’s done.

Our plan for how the work will be divided is still the same as when we submitted the proposal. That is, we plan to have Andrew focus on the front-end HTML and CSS, Robert on back-end JavaScript, and JJ will do parts of both as needed. As the project continues, we will further reevaluate depending on what needs to be worked on.

**Internet Architecture**

Shown to the right is our final IA for this term project. In our initial project proposal, we only outlined one HTML page in the root and a single JavaScript file. This changed as our project progressed. In our IA we have all the html and php files in the root directory and also a folder for resources. The index.php file holds the login function for the site. This is where users log in to Spotify and we can gain the access token to make calls to the web API. The folder for resources houses the CSS file for styling and also the images we used for the site. There is another folder for the scripts that are used by the site such as the API calls and the JavaScript for user interactivity. The second folder just holds documentation for our website such as the wireframes we created and also the project proposals.

**Project Summary**

In the actual execution of the project, we stuck to the original plan on the front end but ended up deviating from the plan to only use JavaScript for the backend coding due to how OAuth 2.0 handles user data. The Spotify Web API uses OAuth 2.0 to handle logging in to obtain user data. After the user logs in to Spotify, they are redirected to the page to build their Wrapped with an access code in the URL. Since the Spotify Web API uses the Proof Key for Code Exchange (PKCE) extension to OAuth 2.0, this access code is useless on its own. When the API call is made, AJAX sends the endpoint, decided by the user’s selection on the dropdown menus, as well as the access code to a PHP script in a post request. This script then sends the access code along with other verification data to the Spotify API’s access token endpoint, which trades the code for an access token. This token is considered sensitive data, as it can access all of a user’s data on its own, which is why it is only ever stored on the server. This token is then saved as a session variable, allowing the token to persist between webpages as well as reload, as well as skip the code-token trade if a user has already visited the website and the token has not expired. The access token is used in a cURL to get a request to the endpoint that was determined based on user input to return the user data. This data ends up returned back to the JS script which made the AJAX post request as a JSON string. The relevant data is extracted and added to the list of items that the user will see on the webpage.

From this project, we all gained experience in many different areas of programming. First, frontend design and development using HTML and CSS to create a visually appealing, user-friendly frontend. We also gained a lot of practice with JavaScript and jQuery to allow for user input and interactions with our website. Understanding PHP was a necessity to successfully call the Spotify API and store the authorization token in session storage. Finally, when data was returned as a JSON, we were able to practice reading JSON and then formatting it onto the webpage.

**Challenges We Faced**

Most of the challenges we faced going into the project were related to debugging backend code, specifically the Spotify API call. While we were able to make a working version using only client-side JavaScript, handling the access token client-side turned out to be a security risk for the user. If we were to use JavaScript to handle the API call, it would have needed to be NodeJS, which was not allowed for this project. Our only option left was PHP, which none of us were the most experienced in using. This inexperience led to many bugs in the code, which were only exacerbated by the AJAX call adding another layer and further slowing the debugging process. However, it was eventually debugged after figuring out how to pass the API error messages into the AJAX call.

After making the API call work correctly, we needed to properly parse down the JSON and format it based on the user’s inputs. This became difficult because of the massive JSON returned, with many key-value pairs that became difficult to sift through. Furthermore, getting specific data required accessing many nested arrays, and bugs in this process was difficult to debug as it was often due to going down an incorrect route of key-value pairs. Finally, there was the issue of writing the code specific to the type of data requested. The options the user could select would take data from one of two possible API calls, but the route of key-value pairs to get to the data was different. This required writing many functions that were very similar but only differed in paths.

**Future Plans**

Our project has the potential to expand given more time and freedom in our stack. Firstly, due to the limitations of vanilla JavaScript and the stack we were permitted to use, allowing the user to share their wrapped via download or social media proved to be very difficult. This was because the image displayed was a URL accesses from the JSON Spotify returned, and libraries that render an HTML div to a download PNG cannot render an image from a URL.

Another avenue to expand is the user customization options for their wrapped. At the moment, all the user can choose is the type of data, the time frame in which the data is from, and colors. Allowing the user to have more control over their graphic would be ideal. An idea would be to give the user a blank canvas and allow them to draw their own wrapped template.

Expanding to other platforms would be ideal to make this as accessible for everyone. Many people prefer Apple Music over Spotify, so creating compatibility with Apple Music would increase the number of users that can use our website.