Playlist Archiver Presentation 1

Progress Report

Presentation Contents

Overview

Prototype

Format of Archive Files

Primary Objectives

Secondary Objectives

Tech Stack / Software Requirements

Accessibility / Hardware Requirements

Project Timeline

Web App Progress

Codebase + Accessing The Web App

Overview

The Playlist Archiver is React-based web app that will allow you to input a link for a Spotify playlist and in turn be able to download an archive file containing all the relevant information of each specific song within the playlist in json (Javascript Object Notation) format. Since playlists are often deleted or lost by the creator, it seemed like a helpful tool for those who wish to hold on the content of playlists they did not create or exist on accounts they no longer use.

When the user provides the web app with a link to any public Spotify playlist, this link is then used to retrieve all of the relevant playlist data from the Spotify API. From there the data is processed using different data structures and then formatted into a json file, which is made available for download on the website. This allows users to save the data of any given playlist to their own machine, which could be useful for people who use streaming services other than Spotify, or people who would like to have a permanent copy of the playlist without fear of it being deleted or made private by its original creator.

Prototype

The final version of the Playlist Archiver will be accessible via a URL like any other website, but to validate the idea before creating a frontend, a prototype was created which operated locally from the command line.

```
nayatrov@ariahas-MBP naya-utils % node sp-pl-archiver.js
Enter the Spotify playlist URL or ID: https://open.spotify.com/playlist/034XigRF
v5CSbUpNccqcqY?si=825bcbc463a4449a
Playlist data saved to Halcyon Days-playlist.json
```

As you can see in the sample output above, the prototype was a single javascript file and run via node.js similar to how python files are ran from the command line. After running the file it prompted the user for a link, and then after successfully saving the file it outputs the success message seen at the last line of the sample output. “Halcyon Days” was the name of the playlist, which is appended to the beginning of the output file name.

Format of Archive Files

To the right you can see the resulting json file from the code ran in the last slide. Each individual song within the playlist is stored as its own Javascript object with the four properties of title, album, artist, and spotifyLink. They are stored within the json file in the same order they appear within the playlist.

{} Halcyon Days-playlist.json > ...


```
[
  {
    "title": "I Need A Forest Fire",
    "album": "The Colour In Anything",
    "artist": "James Blake, Bon Iver",
    "spotifyLink": "https://open.spotify.com/track/0TLAptKgYxe7F0KewWH6X6"
  },
  {
    "title": "Avril 14th",
    "album": "Drukqs",
    "artist": "Aphex Twin",
    "spotifyLink": "https://open.spotify.com/track/1uaGSDFsLdReQgg8p7Obwh"
  },
  {
    "title": "Reality Surf",
    "album": "333",
    "artist": "Bladee",
    "spotifyLink": "https://open.spotify.com/track/6HJszgJO19tAKff7X40ggp"
  },
  {
    "title": "MOJABI GHOST",
    "album": "DATA",
    "artist": "Tainy, Bad Bunny",
    "spotifyLink": "https://open.spotify.com/track/4eMKD8MRroxCqugpsxCCNb"
  },
```

Primary Objectives

01 Implement the original prototype so that it functions from the server-side of the web app and is capable of being passed values from the front-end.

02 Build the frontend using React/Nextjs, and make sure that the values inputted into the frontend can correctly be passed to the backend.

03 Ensure the file is downloadable after being retrieved from the backend and fix any bugs that arise as we begin testing.

Secondary Objectives

01 Use CSS to style the web app and make it more user friendly.

02 Add support for streaming services other than Spotify as well (Apple Music, Tidal, etc.)

03 Add functionality to create playlists on your own account using the archive files as input.

Tech Stack / Software Requirements

Backend

Node.js and Javascript are being used on the server-side (backend) for retrieving data from the Spotify API, processing it, and then making it available to the frontend

Frontend

For the website portion (frontend) I am using React for the foundation, and CSS to style it. You can think of React as a Javascript framework that is essentially HTML with much more functionality. I am also using Next.js for better SEO (search engine optimization) thanks to it providing server-side rendering and static website generation.

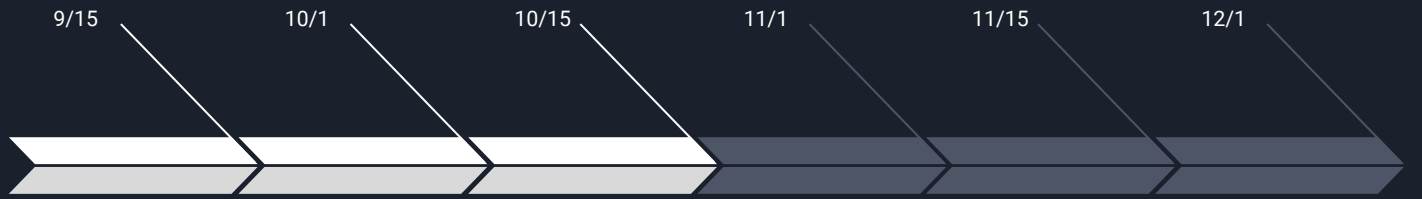
APIs / Node Modules

The only API the app is interacting with at the moment is the Spotify API but this may expand to other streaming service APIs like Apple in the future. The relevant node modules the app is using at the moment are ‘dotenv’ for obscuring API keys, ‘axios’ for sending requests to the Spotify API, and ‘‘fs’ for writing the data to a file on the server once it is retrieved and processed.

Accessibility / Hardware Requirements

This project will be accessible via any device that can connect to the internet and download .json files. All that is required is a link to a public Spotify playlist. The project is hosted on Vercel.

Project timeline



Topic Selection

Decided on my project topic and which language would be best to implement the project.

Prototype + Structuring

Created a prototype for the main project functionality, and then began working on structuring a frontend around it.

First Presentation

Preparing and giving the first presentation while also making progress on implementing the main functionality of the project.

Frontend + Adjustments

The main functionality of the project should be implemented by this point and I should be focusing mainly on building the frontend around it.

Cleaning Up

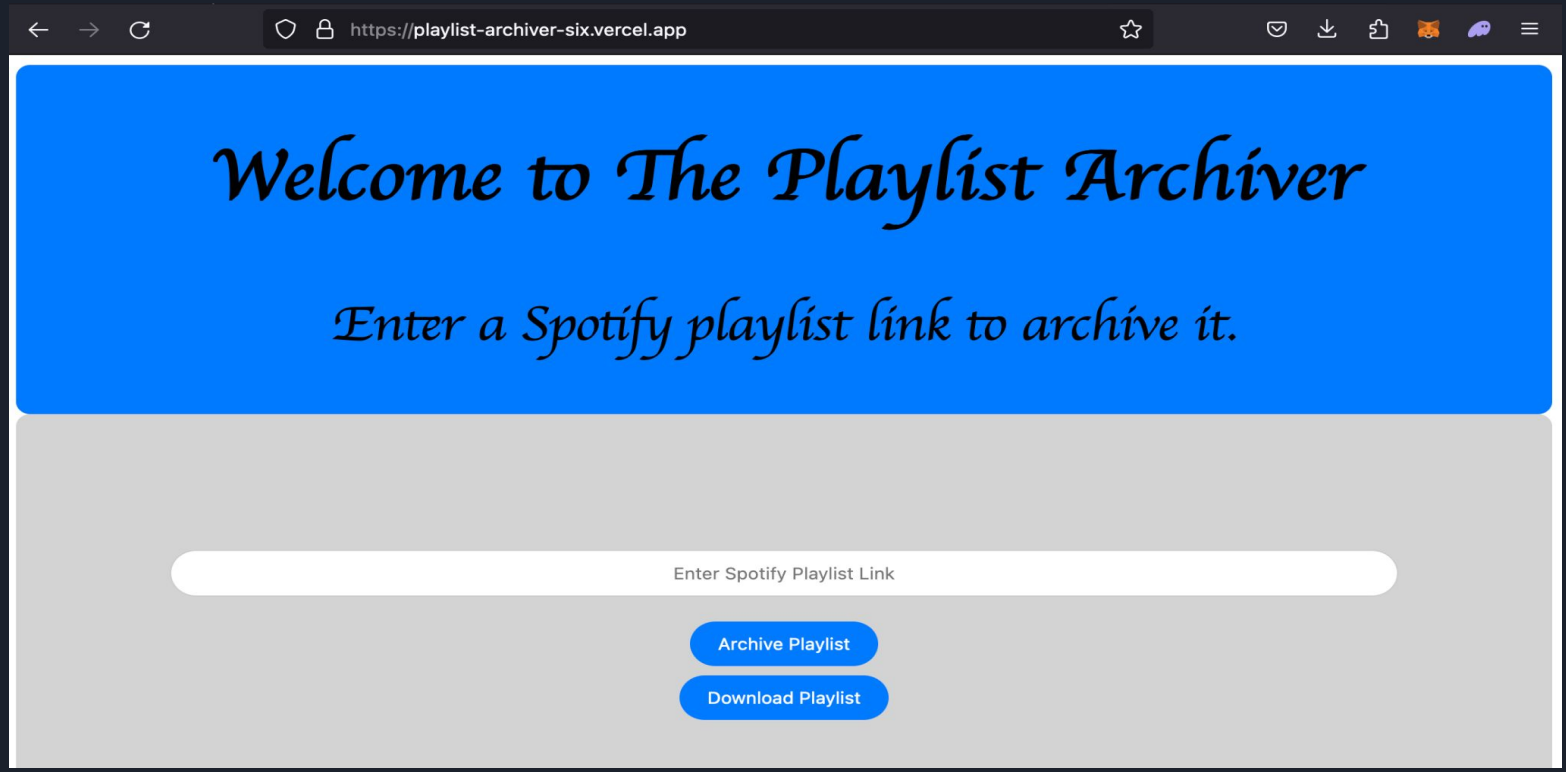
The vast majority of the project should be finished up by this point, and most of the work at this point should consist of refining the CSS and fixing any bugs that arise.

Final Presentation

Presenting the finished product in its entirety as well as turning in all documentation and the codebase. The project should be accessible via url at this point for future users.

Web App Progress

Below is the basic layout of the web app as it is right now. It's only a rough prototype, and after all the functionality is put in place I will use CSS to style it further.



Codebase + Accessing the Web App

The code I have so far can be found within the github repository at:

www.github.com/tenjotenge/playlist-archiver

I have deployed the web app using Vercel so you can access it via the link in the top right corner of the Github repository under “About.”