Spotify Playlist Generator

Developer's Guide

by Wenzhe Lin, Matthew Lam

---

1. Project Overview

The purpose of this project is to create a system that generates personalized playlists by analyzing user prompts and leveraging Spotify's audio features. The project aims to enhance the user experience by recommending music that aligns with specific emotions, activities, or scenarios (e.g., "workout," "relax," "sad"). The end goal is to create software that serves as a retention tool for music streaming users or as an entry tool for new users to get into music listening.

Spotify Playlist Generator (SPG) is integrated with the popular streaming service "Spotify" to create seamless software for users. The design is to mimic Spotify's easy-to-navigate user interface to ensure ease of use for our targeted consumers and stakeholders. This software's services rely upon Spotify's web API to retrieve data, get recommendations, and play recommended songs directly onto Spotify's platform. Moving forward, we hope to include personal authentication to include account-specific features such as share and save playlists or to get personally curated playlists based on listening habits.

This project was created under the coursework of CIS 434 - Software Engineering at Cleveland State University. While creating this software project, certain skills such as web API integration became necessary. Other skills include front-end design, back-end algorithms, and logic/data handling.

Audience

Anyone can use the playlist generator. It is for anyone who needs a playlist, or they are trying to add new music to their playlists. For version 1.0, no Spotify account is required to utilize SPG.

Tech Stack:

- Front-end code includes HTML, CSS, Javascript
- Back-end code includes Python, node.js
- APIs use Spotify Web API
- Other tools used are GitHub, Spotipy, pip, flask

2. Setting up the environment

1.) Install a Visual Studio Code (or any IDE) for your specified device, macOS, Linux, Windows, or whatever operating system you are using.
2.) Install the extension called “live server” which allows you to host a server from your operating system, this should open in Chrome, Firefox, or whatever browser you use. To open a live server, two-finger press on macOS on any HTML file. On Windows right click on any HTML file.
3.) Download the project source code from GitHub https://github.com/ZekuuMatthew/SongPlaylistGenerator
4.) Open the project press Control p or Command p on macOS and type clone into the search bar. Clone the project and all the files and pictures should appear.

Using Spotify Playlist Generator

1.) Install software
- Node.js (v14 or later): For running the backend server.
- Python 3 (v3.8 or later): To execute the recommendationgetter.py script.
- pip: For managing Python packages.

2.) Python packages, run these commands(make sure pip is installed):
- "npm install"
- Spotipy: command is "pip install Spotipy" Once pip is installed
- "pip install flask"
- "pip install requests"

3.) Ensure every file is properly downloaded and in the right folder

4.) Run server
- type "node node.mjs" once everything is set up to start the backend server

5.) Open the live server and test our software! (make sure the extension for the live server is installed for VScode)

For a more detailed guide, please refer to “usersguide.txt”

3. Prerequisites

Software Requirements

-Node.js (v16 or higher)
-Git
-Visual Studio Code (or any preferred IDE)

Accounts and API access

1.) Go to Spotify Developer Dashboard
2.) Create an account
3.) Create a new app and receive your Client ID and Client Secret (for grading purposes, we have included a client ID and Secret directly into the code since the software is not for public deployment)

4. Frontend development

HTML was used to create each web page, currently including the home page, artist, and generated playlist page. **CSS** was used to style the web pages adding colors, buttons, and links and making it look neat overall. A little bit of **JavaScript** was used to implement the search bar, when the user presses the search bar icon it will generate the playlist, if they are on a computer and type the “enter” key it will also generate the playlist.

```
├── frontend
│   ├── index.html        # Main homepage
│   ├── artists.html      # Page displaying artists
│   ├── generate.html     # Playlist generator page
│   ├── styles.css        # Frontend styling
│   ├── app.js            # JavaScript for client-side functionality
```

5. Backend development

The backend comprises several Python libraries, node(node.mjs), and communication with Spotify API.

```
├── backend
│   ├── node.mjs          # Backend API using Node.js
│   ├── recommendationgetter.py  # Python script for generating playlists
│   ├── .env           # Environment variables for Spotify API keys
```

"recommendationgetter.py" handles the user input and determines the category of music and genre of music to search for. Once the Python script determines the right type of music to search for, it communicates with Spotify API to search for songs that fit the criteria (track features). "Limit_per_genre = x" can easily be modified to search for more or fewer songs per search.
"Node. mjs" is the backend server that communicates between the user inputs to the backend algorithms. To run the server, "node nodde.mjs".
".env" contains Client ID and Client Secret (not in our project, because not public deployment)

6. Testing

The app uses Spotify's get /search and post /playlists endpoints to search for songs and create playlists. Manually test the playlist generation by entering moods or activities including phrases and verifying the output. Use browser developer tools to monitor API responses.

Testing tools

- Hardcoded playlists can be used if the front end properly displays playlists for users
- Postman tests to see if our backend is properly communicating with Spotify's API by checking for endpoint connections
- Various keyword inputs, view outputs

Test cases

- **Test Case 1**: Search for the keyword "Workout"
- **Test Case 2**: Click "Generate playlist"
- **Test Case 3**: enter an API request and it returns errors
- **Test Case 4**: The search input is empty
- **Test Case 5**: The user input contains incoherent words, ex: "asdfg"
- **Test Case 6**: User inputs contradicting keywords, ex: "happy"+"sad"
- **Test Case 7**: Navigational buttons taking us to expected pages (ex: home goes to homepage)
- **Test Case 8**: 50-song playlist generates
- **Test Case 9**: Request sent to generate a song playlist
    - **Test Case 1**: A playlist containing high-energy songs suitable for a workout. All songs should have at least 120 bpm.

- **Test Case 2**: A playlist with 50 songs is generated and a “Save to Spotify” button
- **Test Case 3**: Error message displayed to the user. Prompt the user then try again later.
- **Test Case 4**: Warning message: "Please enter a mood or activity to generate a playlist"
- **Test Case 5**: Warning message: "Please enter a mood or activity to generate a playlist"
- **Test Case 6**: Warning message: "Input contradicted, may result in unexpected results" Prompt user if they want to proceed
- **Test Case 7**: If we click the home button then we expect to be on the home page, some go for other interactive buttons such
- **Test Case 8**: The expected output is to be able to view the generated playlist
- **Test Case 9**: The back end handles reading song data from Spotify API. print onto CSV to verify the successful reading

- **Test Case 1**: Successful
- **Test Case 2**: Successful
- **Test Case 3**: Successful
- **Test Case 4**: Successful
- **Test Case 5**: Successful
- **Test Case 6**: Successful
- **Test Case 7**: Successful
- **Test Case 8**: Successful
- **Test Case 9**: Successful

As the project scales, more test cases need to be created to ensure the project meets requirements. Starting point for browsing: “index.html”. Hardcoded playlist and inputs tests for proper communication between the frontend, and backend and ensure playlists are properly displayed.

7. Troubleshooting

Error “Invalid Spotify credentials”
Solution verify that your ClientID and Client Secret in the .env file

8. Future contributions

Spotify’s API contains several endpoints that hold endless possibilities for scaling Song Playlist Generator. As music streaming platforms grow and more music genres pop up, SPG can constantly be updated to keep up with new music and genres of music. Spotify API also offers development with audiobooks and podcasts.

Future contributions

- Recommend audiobooks, podcasts, and any other Spotify content
- Input song IDs and recommend similar songs that share similar genre, or track features such as valence or tempo.
- Natural language processor for handling user inputs instead of using a keyword bank for determining recommended categories and genres. This results in no longer relying on a work bank, case sensitivity, and minor spelling error
- Artificial intelligence integration for better song recommendation.
- Large server databases for storing use data, music/song data, and a history for users to view past generated playlists

Authenticated features(user sign-in with Spotify username and password)

- Save the playlist into the user account library
- Share the playlist with other Spotify users
- Spotify holds user data for listening habits, we can use this to create individually curated playlists
- Community of Spotify and music listeners. Interact with other users, and artists, share musical tastes, and view each other's profile.

END