Music For You

Travis Aelvoet, An Nguyen, Perry Scott, Andrew Smith, Leif Thomas

April 23, 2021

Introduction:

The purpose of the website is to categorize popular music and show where a musical artist will be. The website will sort the music into several different categories, or “pillars”. The pillars of the project are artist, album, popular songs, genre, and release date. The website focuses on allowing the user to connect with the music they love.

Once a user selects an artist, song, genre, or decade, the website will then provide a deep dive into the selected option. This will give the user a better insight into the music they love, while also providing them with the information they need to find more music they will like.

For example, the user can search for the band. The website will then provide the user with relevant information about the artists that they may be interested in listening to. Once a user has found artists that they like, they will be able to find more similar artists.

The priority of the website is to display many rows of information that relate to the most popular songs on Spotify. This information will allow users of the website to learn interesting trivia about their favorite music. This website also serves as a trial of using the database software postgresql.

The tools used on the project are GitLab, Google Cloud Platform, Bootstrap, Javascript, Python, Flask, SQLAlchemy and PostgreSQL.

Using Flask and SQLAlchemy we are able to use Python to display dynamic web pages that contain information about music. This information was gathered from the Spotify API. The Spotify API provided easy to access JSON files that we were able to convert into a SQL database.

Design:

APIs Used:

- https://api.spotify.com

The spotify API is an API that uses REST principles. To use the API we needed to request a valid OAUTH token, which was granted automatically. This API provides several endpoints that provide various information about music. The API then returns all of the requested information in the form of a JSON file.

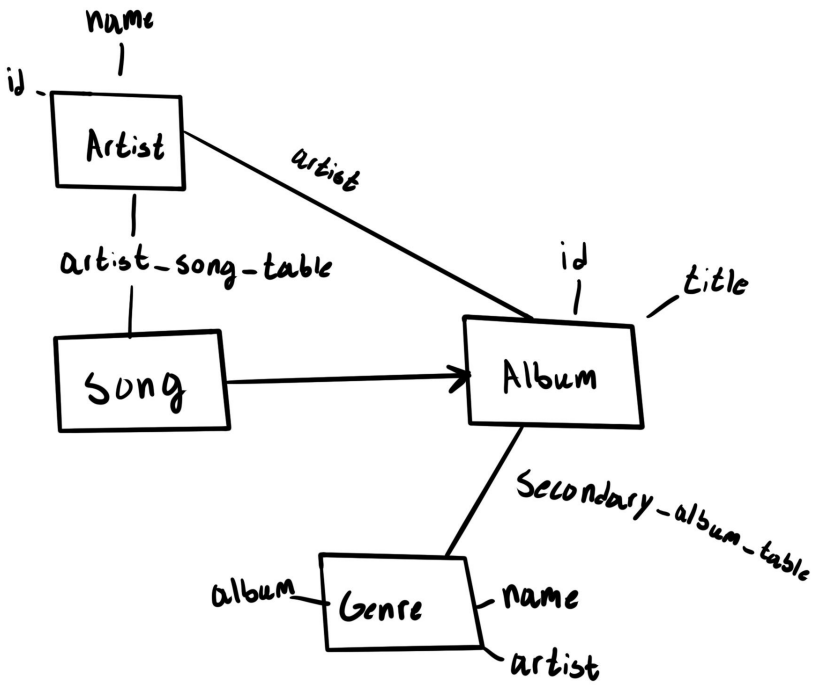
To use this data we set up a python script that would request the data for the top 100 songs. To do this the script calls the playlist endpoint "https://api.spotify.com/v1/playlist/" and references the appropriate playlist ID for Spotify's top 100, in this case the ID is "6UeSakyzhiEt4NB3UAd6NQ", making the API call "https://api.spotify.com/v1/playlist/6UeSakyzhiEt4NB3UAd6NQ". From this data we are primarily interested in retrieving the artists of said songs, so we extract the artist section of the JSON given by the API call and its internal Spotify ID into a dictionary that uses the ArtistID as a key and the Artist's name as the Value. This should give more than 100 artists as many songs feature other artists.

After retrieving the artist name and ID of the top 100 songs on Spotify, we do another API call that retrieves the artist's top tracks and the album that the track is in. To do this we iterate through the dictionary of ArtistIDs and do an API request in the format of "https://api.spotify.com/v1/artists/'artistKey'/top-tracks/" where 'artistkey' is replaced by the ID of the Artist from the dictionary of artists that we are iterating through. All of this information is appended together into a list of dictionaries that each have three keys; "track" that has the song name as its value, "album" that has the album name as its value, and "artist" that has the artist name as its value. This list of dictionaries is then dumped into the database. We do this so that the information is more easily referenced. By doing this we also help to future proof the website, so that a change in the API will not break the website later.

Database Models:

The SQL database pulls its information from Spotify made from its API data. To do this it runs the API data through a python script that parses through the API data to return the artist, album, and song. This then assigns an id to each entry, ignoring repeating entries.

Our Database has 5 tables; song_to_artist, song_to_album, artist, song, album.



This illustrates songs the relation between songs to artist, songs to album, and artist to album. Along with its relation to genre. Each artist can make multiple albums, and multiple songs. While an album can also have multiple songs and multiple artists. A song can have multiple artists. An album can have multiple genres.

An artist has two primary attributes ID and name. An album is similar to an artist as it has two primary attributes, ID and title. A song also has two attributes, ID and title.

Songs is dependent on album. Artist has two primary attributes, id and name. Album has two primary attributes, id and title. Song has the attribute name. The tables are all linked together in the database.

Because of the way these tables are linked to each other, we are able to reference the information in one table to another. This allows us to display a table on the website that contains the song name, album name, artist name, and genre name. Since song to album is a many to one relationship we are able to reference an album and receive all of its songs. A similar relationship also extends to artist to songs and artist to album. However, a song can have multiple artists and can be in multiple albums, such as a greatest hits album, so the database allows the website to have the same entry be referenced in multiple ways.

Search Capabilities

The Music For You database can be queried in several different ways. All of the query models are available to the user in the navbar on the top of the webpage. The first model is by Artist, where all of the artists in the website's database are available in a searchable table. This query for all Artists is done upon opening the "artists" page, and on this page there is a paginated table that displays each artist in the database along with their top song. In this table, the entries per page can be controlled with a text box marked "Show X Entries" in the top left corner. For each artist, their top song is also displayed, which is also data that is available to be searched in the top right search bar. If the user wants to search for a particular artist, they can search for them in the search bar, and the jQuery table plugin that controls the table data quickly queries for relevant data. The second model is the song model. This model is on its own web page called the "song" page, and with each song in the database, the album that it appears on, the artist it is made by, and the release date of the song are all available in the table. The table has the same jQuery functionality that is used in the artists page as well. The third model by Album and is on the "album" page. On this page, the album name, album artist, top song from the album, and album release date are all available in the jQuery table, again with the same functionalities as the other pages. The fourth model is the genre model, which also is on its own webpage called the "genre" page. It also utilizes the jQuery plugin table, and the user can see all genre names and relevant artists and of course this table can be queried using the search bar. Lastly, there is a search bar in the top left of the navbar on every page that allows the user to search the entirety of the database, which is a combination of all of the data represented in the four previous models: song, artist, genre, and album.

It should be mentioned that each entry in the database is a link that can be clicked. Depending on whether a song, artist, album, or genre is selected, the user will be sent to a page detailing the name of the entry they selected, with a layout of the entries relationships, and a picture if the selected entry is an album or song name.

API Using Postman

Using Postman we were able to create a collection of API endpoints. Because MusicForYou.com is rendered with jinja templates, the API passes the database as an HTML file rather than a JSON as our Flask app passes the database information directly into the html template. The API requests are searchbyartist, searchbysongs, searchbyalbum, searchbygenre, and getartist.

Tools:

Front-End:

When a user first enters the site, they will be greeted by the homepage- the splash page. At the top of the display there is a Navbar that presents the user with the option to either go to an about page where they can learn about the project or they can go to any of the model pages. In the middle of the splash page is a carousel that contains pictures of various musical artists. Once the user scrolls down they are presented with tabs implemented using Bootstrap that contain the varying ways to search for music. These pages are linked using Flask so that they can be dynamic. The Tabs available are Artists, Generes, and Songs.

- “All” displays all the artists available and lets the users freely browse for their artists.
- “Generes” displays the type of music the artist performs.
- “Songs” displays the top songs of Artists.

The homepage uses Flask to connect to other pages such as songs, artists, etc. It uses render_template to connect to the HTML of the other pages. To create the page aesthetics we employ Bootstrap, so that all of the elements are consistent and look modern.

Format of Pages

To browse by artist, click on the Artists tab. On the web page- artistPage.html, there is a search by artist name, genre, top songs, album and released date.

The artist page has a search bar for users to search for the artists. The page is divided into two parts: the navigation bar and main body. The navigation bar consists of links that consist of the about page, artist, songs, and genres, along with a link to the homepage and a search bar. The second part is the main information box. In this, it displays all information about the searched artist. The artist page displays the discography of the artist. It also has the picture and description of the artist. The page contains a table that contains information on the artist. The website also shows related music and artists of the searched artist. It also shows related music of the artists or artists of similar genres. The information box contains a table. The table is used to display the information in an informative and pleasing way in the middle of the page. The file artistsTable.html contains the information about the table. In artistTable.html, for example, “artist name” is a set of entities, where each entity is a name and an entry. You can see on the artistsTable.html what the artist's name and model looks like. The page

is also clickable, if clicking on an artist, song, or genre, the webpage will lead to a page containing the information about the subject.

Similarly, to browse songs, the user would click on the songs tab in the navbar. Similar to the artist page, the song page consists of two parts, the navigation bar and the main body. The navigation bar is used for searching different songs and displaying them. The search bar will show the result of the search. The second part is the main information box. It consists of a table that contains information about the top artists on spotify. For example, “songs” is a set of entities, where each entity is a song. You can see on the songsTable.html what the song model looks like. It would include artist name, genre, top songs related to the artists, origin, and also released dates. Along with the searched songs, it would also show related information. While browsing songs, it will also include song names and albums with similar songs and genres.

There is also a page called “All.” This page consists of all songs and artists that allows for users to have flexibility when they browse. The set up of this page is similar to other pages, with a search bar and the main information table. It contains song name, album, artist genre and release date. All the tables from the web page are sortable in alphabetic order from any of the columns. The last page of the website is an about page. The about page contains information of all of the members who are working on the project and their basic information (such as majors, interests). Alongside information on the group members, the about page also contains information about the project itself, including issues and commits from the GitLab repository. On this page you can also find the results of Unit Tests run on the project. The page linked to the main page and displayed on the navigation bar.

Back-End:

The group collaborates using a shared GitLab repository. This provides versioning control as well as keeping all team members on the same page during development. The development environment that the website runs off contains Flask, flask_sqlalchemy, requests, psycopg2-binary, Click, Jinja2, pep8, SQLAlchemy, gunicorn, and coverage.

Flask is used to serve python code into an html document. The use of Flask has allowed us to create dynamic web pages whose data can be changed depending on the information provided. Alongside Flask the website employs SQLAlchemy so that we can interface with the SQL database using python. Together the website is able to use python to serve information that is stored in an SQL database to the website itself.

To properly document the project we used pydoc to output a file that shows how each portion of the main python file works. To ensure that the website is working properly we

also employ several unit tests throughout the project. These unit tests allow us to see that the website is behaving as expected, even in edge cases.

When displaying query data for our 4 main models, tables were used that utilized a jQuery table plugin that was extremely robust for displaying and querying information in the table. The user can type a string into the search box in the top right corner of each table to quickly query for matching strings in any of the columns/rows of the table. The table is paginated if the number of rows exceeds the number in the top left “Show X Entries” box. This number can be changed to show the desired amount of entries per table page.

Hosting:

The website is hosted on Google Cloud Platform (GCP). To set up the GCP:

1. Create a GCP project,
2. Create an app.yaml file,
3. Install the gcloud sdk
4. Push the app to the app engine.

Namecheap:

The website domain name was purchased from namecheap.com for free using a student promo. The domain name of our website which is hosted on GCP is called musicforyou.me. The following steps give the website visibility for both of the URL’s of musicforyou.me and www.musicforyou.me.

The first step in using a Namecheap domain on a GCP hosted website is to enable Google Cloud DNS API under the API manager found in the GCP interface. Once this is completed, a new zone is created under the Networking tab. We then named the zone “MusicForYou" and set the DNS Name to the Namecheap purchased domain name of musicforyou.me. Next, we added an A record set which was under the name MusicForYou.me, and added the corresponding IPv4 address that our GCP website was being hosted on. Additionally, we added an additional recordset under the name www.MusicForYou.me and the record type of this was a CNAME record with the canonical name “MusicForYou.me”

In Namecheap under the domain list, we accessed our purchased domain name of musicforyou.me, and under “name servers", we added the google server domain url’s under Custom DNS tab. These urls came from the data section under our recordset for musicforyou.me.

User Stories:

In order to give our website the greatest relevance to the user, we wrote user stories, assigned time to completion for each user story, and equally distributed these user stories to the team, keeping in mind that the total time to completion for each team member should be relatively similar.

User Story #	User Story	Time to Completion (hrs)	Assigned Member
1	As a website visitor, I want to be able to search through the songs to find my favorites.	1	Travis Aelvoet
2	As a website visitor, I want to be able to find artists that match genres I already like.	1	Travis Aelvoet
3	As a website visitor, I want to find songs by my favorite artists.	1	Travis Aelvoet
4	As a website visitor, I want to find albums by my favorite artists.	1	Ann Nguyen
5	As a website visitor, I want to search across songs, artists, genres, and albums.	2	Leif Thomas
6	As a website visitor, I want to find	1	Leif Thomas

	albums my favorite artist appears on.		
7	As a website visitor, I want to be able to search the songs page for my favorite tunes.	1	Andrew Smith
8	As a website visitor, I want to be able to find the music that best matches my query with a universal search.	2	Andrew Smith
9	As a website visitor, I want to be able to see album, artist, and song artwork for each page.	2	Ann Nguyen
10	As a website visitor, I want the pages to link to one another so I can explore the site and find new music.	2	Perry Scott
11	As a website visitor, I want each page with multiple results to have a search box that doesn't require reloading the page.	.5	Perry Scott