Abstract

Music is central to our lives and culture, and most songwriters draw inspiration from the physical world around them when writing lyrics. Because of this, song lyrics are filled with references to real-world streets, buildings, neighborhoods, natural formations, and other landmarks. Listeners may be curious about the meaning behind a place alluded to in a song. Residents of a certain town or neighborhood may want to explore music that was written about the area in which they live. Both are possible with this project, which allows the visualization and exploration of location references in music.

Lyric Map is a public, interactive, user-editable website that represents location references in music geographically on a map. Users can explore the map, viewing pins that contain lyrics and other song information, and listen to the referenced song without leaving the site. In order to add new references to the map, users create a free account that keeps track of the lyrics that they've added. The site administrator has additional actions available to them, including downloading backups of site data and deleting wrong or offensive lyric entries. The site is freely hosted on Heroku and is built on the Google Maps Javascript API, the Flask Python framework, and a Redis data store. Possible further areas to be explored include expansion of account features, allowing
the connection of Lyric Map to other websites relating to music, or an analysis of the different geospatial distribution of different musical artists or genres.

**Background**

The idea for this project was sparked by my habit of reading and looking up the meaning of lyrics in order to better understand songs and artists when listening to music. In order to truly understand the emotional message of a song, a listener must put him or herself in the frame of reference of the artist, and a large part of that frame is the geographical location. Real-world locations carry immense weight – they make up our homes, schools, and sites of important life events. A map that would allow one to explore the world around them through music (and explore the world of music through the world around them) was a natural offshoot of my interest.

Other than creating a web map, a concurrent goal for the project was to explore new technologies and frameworks, and use the technologies I have some experience with in new ways. In this way, I would be able to create something of use while expanding my skills. At each step in the process, I weighed the options and, from among the many different options, selected technologies that would both fit the use case and allow me to gain experience as a developer.

**Screenshots/walkthrough**

A visitor to the site will first encounter the main map page, which is centered on the United States. The map allows for dragging, zooming in and out, and switching from the map view to a satellite view. In addition, a location search box located in the upper right allows users to easily move the map to a particular address.
Fig. 1: Lyric Map’s main page as shown prior to logging in.

Users that are not logged in, while not able to add lyrics to the map, can explore the lyrics already added by clicking on one of the pins. An info window appears on click with the lyric information, including the artist name, the album name, the year released, an embedded link to play the song on Spotify (if available), the name of the user who added the lyric, and the time the lyric was added.
Fig. 2: An example lyric pin with information displayed.

If users that are not logged in attempt to add a pin to the map (by clicking on the map), they will be shown a message that prompts them to log in.

Fig. 3: The log in prompt.
To register for an account, log in to an account, or view the “about” page, users simply use the options in the navigation bar above the map. Once a user is logged in, they are shown basic information about their account (their chosen nickname and the number of pins that they have added). In addition, logged in users can add a pin by clicking to the correct location on the map. They will be presented with a form to fill in the details of the lyric.

![Fig. 4: The form for submitting a lyric pin.](image)

![Fig. 5: The message displayed on successful lyric pin form submission.](image)
If a user is logged in as the administrator, they will also be presented with the option to delete any of the pins (in case of abuse).

![Image of admin's view of a lyric pin, including the option to delete the lyric pin (at bottom).]

**Lyric:** *Amarillo by morning, up from San Antone*

**Song Title:** Amarillo By Morning

**Artist:** George Strait

**Album:** Strait From The Heart

**Year:** 1982

*Pin added by: dave on 5/3/2017, 8:52:05 PM*

**Delete**

Fig. 6: The admin’s view of a lyric pin, including the option to delete the lyric pin (at bottom).

**Technical details**

The site is hosted on Heroku, a web hosting platform. The main factor in deciding this route was the price – for web apps that get less than a certain amount of traffic,
Heroku is free. In addition, if Lyric Map becomes suddenly more popular, Heroku’s infrastructure would allow service to be maintained and the app to scale smoothly.

Flask was chosen to develop with because it is a popular and accessible yet flexible and robust web framework. In addition, Flask is a technology that I have wanted to experiment with for some time, and it fits the project perfectly. Similarly, out of the possible ways to store and handle the data, Redis was chosen for its relative simplicity, widespread adoption, and possibility for further use in the future. Furthermore, Python libraries exist for Redis with good documentation and ease of use.

The data in the Redis store is backed up both by Heroku’s own built-in replicas as well as periodic JSON dumps of the database. When a user with admin permissions requests a backup (by sending a request to /snapshot), a JSON dump is timestamped and saved to disk. In addition, hashed and salted passwords are stored in the user database rather than plaintext passwords to prevent a malicious intrusion gaining access to passwords.

In order to interact with other services (like Spotify), API keys had to be obtained and registered. To keep these keys private, they are saved in environmental variables and read at runtime so that they do not appear in the source code. When a song is saved to the database, a search of the song title and artist is made on Spotify. If a match is found, the URI of the Spotify song is stored along with the rest of the lyric data so that the Spotify song preview can be displayed on request.

Upon loading the map, only the latitude and longitude of each of the lyric pins are sent over at first. Only when a lyric pin is clicked on is the rest of the data (lyric, artist, song, Spotify embed code, etc.) finally requested and displayed. The reason behind this
system is to reduce the size of the payload requested on each page load. If all the data were requested at once, the page would load very slowly, especially on slow connections and in the case that there are hundreds or thousands of lyric pins. One tradeoff of not including all the data is that less customization of the lyric pin appearance can be made – if the artist or genre data were also sent over initially, then the pins could be color coded by some other factor instead of being a uniform color. I decided to keep page load times as low as possible, especially for the main pages.

The Google Maps API is used as the basis of Lyric Map, albeit with a slightly different appearance because of a custom theme. The info windows that display with the lyric information are based on the info window objects provided in the Google Maps API. In addition, Google’s geocoding service is used for the location search box to geocode queries and pan to the correct location.

**Future Possibilities**

There are a number of areas for possible future development of Lyric Map. For one, the features that logged in users get from having an account could be expanded. For example, a “leaderboard” of the best contributors or the ability to customize an account page might draw more users to the site and make them more active contributors. In addition, connecting Lyric Map to other music-related apps may make for a more rewarding experience. In areas where the lyrics are dense enough, songs could be automatically played on a music streaming service when close enough to them, providing the soundtrack to a road trip, for example. Finally, once the dataset is large enough, the lyric and location data collected on Lyric Map may provide some interesting opportunities for analysis. For example, locations could be analyzed by
genre or artist to see whether certain genres favor certain areas of the country, or to find which artist has the most diverse geographic references. Since the lyric and location data will always be available publically, others in the future may be able to perform this type of inquiry.

Acknowledgements

First, I’d like to thank Professor Piskac for her help and guidance on this project. In addition, it goes without saying that this project would not have been possible (or nearly as enjoyable to complete) without the work of the developers whose work I built on in using Flask, Heroku, Redis, and Google Maps. In particular, I found the Google Maps API documentation very clear and informing.