JJ Abides and Peter Stanton

March 11, 2018

CSS 490

Professor Dimpsey

# Program 5: Final Project – Documentation

## **URL**

<http://spotifysearchengine.azurewebsites.net/>

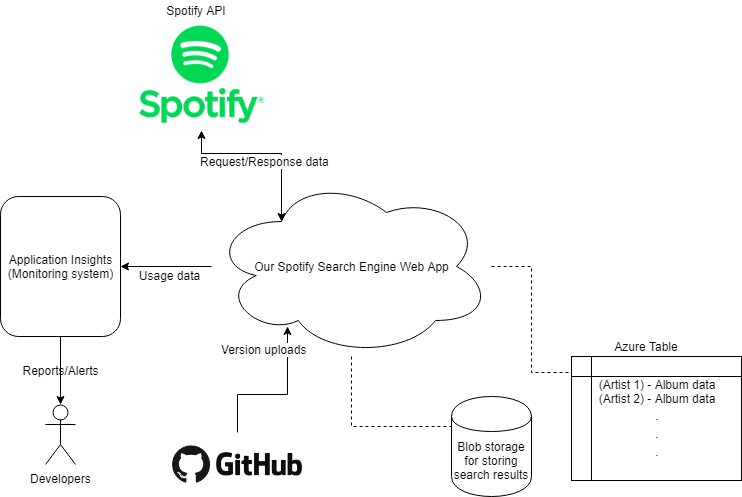
## **What is our Cloud Service?**

Our cloud service is a web app called The Spotify Search Engine, which takes a search input for a song artist, utilizes Spotify’s API to get a list of artists with the given name, Musixmatch to retrieve a sample of lyrics, and can store an artist’s album and song data into an Azure Table for querying and analytics.

## **Services Utilized**

* Azure Web Apps – For running our website
* Blob Storage – For storing all searches to form a statistic on popular searches
* Azure Table Storage – For storing the data about an artist and making queries
* GitHub – For version control
* Azure Application Insights – For monitoring app usage and sending alerts
* Spotify API – For utilizing Spotify’s login and obtaining song artist data
* MusixMatch – for retrieving song lyrics.
* Spotify – Acts as a repository of artists, tracks, and albums for users to consult.

## **Design Diagram**



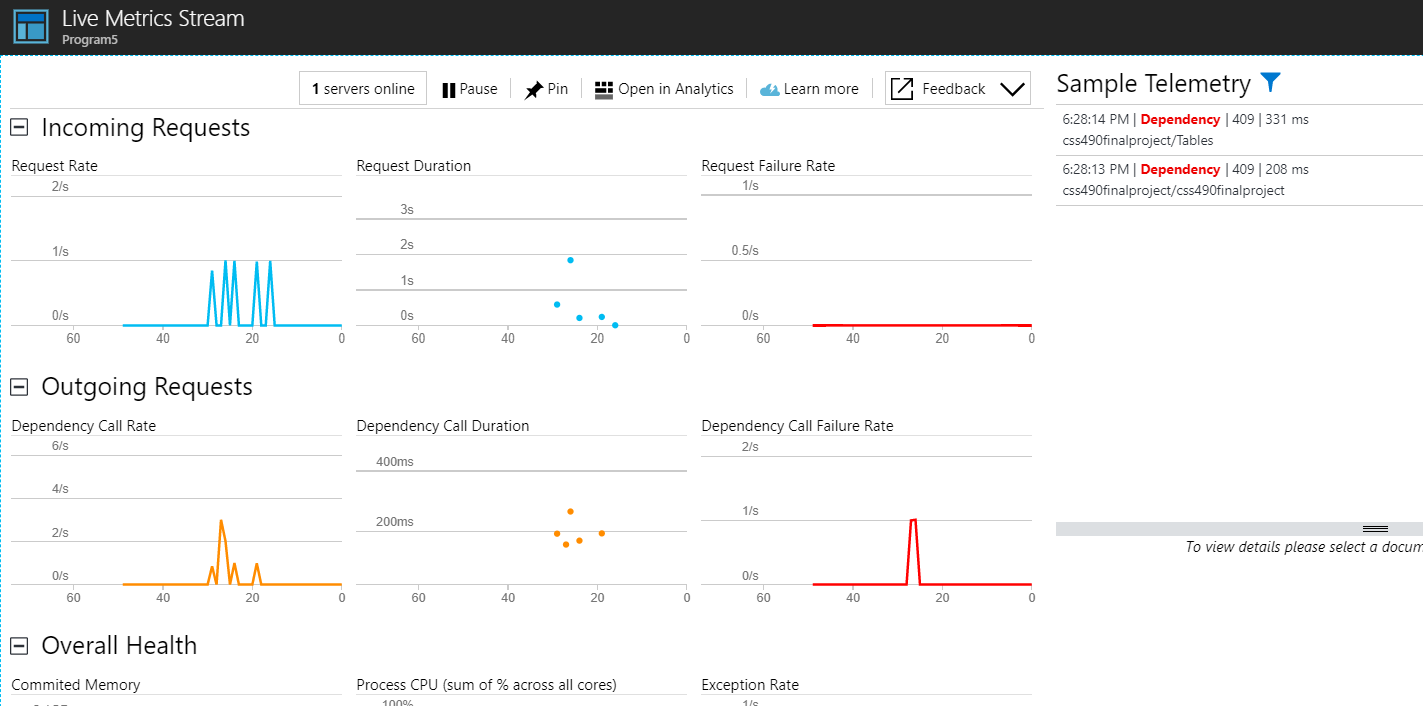
## **Why did we choose Azure?**

We used Azure to build our program 4 web app and we’re basing our Spotify Search Engine web app based on what we learned from program 4. So, we chose Azure because we are most familiar with it for web app development.

## **Monitoring the System for Availability**

We monitor the web app through the Azure dash board and the Application Insights service. Application Insights provides us with a complete overview of metrics such as performance, failures, usage, and availability. We use Application Insight’s Smart Detection feature to email me (JJ) and Peter in case of any issues.

Sample of Live Metrics Stream in Application Insights



## **Service-Level Agreement (SLA)**

Because we are using the free tier service to run our web app, we are not guaranteed an availability percentage.

As stated by Microsoft Azure’s SLA for App Services:

We guarantee that Web Apps running in a customer subscription will be available 99.95% of the time. No SLA is provided for Mobile Apps, Logic Apps, or API Apps while such services are still in Preview or for Apps under either the Free or Shared tiers.

If we can assume that we will at least have 99.95% availability for running our web app using Azure Web Apps, then we could multiply that availability with the availability of all the other services we are using (that are required to run the app) to get an availability percentage for the whole web app.

* Blob storage availability: 99.99%
* Table Storage availability: 99.99%

Thus, the potential availability of our app is 99.93%.

Spotify and MusixMatch do not provide SLAs.

## **How Our App Will Scale with Load**

Because we are using the free tier service, our web app will not auto scale. If we were to scale it, however, we would shift our tier service to B1 Basic to improve processing, availability, and storage size. Auto scaling would also be enabled for scaling out.

## **Discussion**

This was a difficult assignment that resulted in significant refactoring as we went along. Originally, this project was to include player functionality through Spotify connected devices such as phones, smart speakers, or desktop clients. However, playback APIs are still in development, and we could not make Spotify’s OAuth service work with our ASP.NET web application. While the original implementation of playback works in a locally debugging mode, when published to Azure attempting to authenticate to Spotify generates a 502-bad gateway error. I suspect this issue is caused by the need for OAuth to redirect to http://localhost to notify the application that authentication was successful. I am guessing that an Azure app service is not the ideal cloud product to host this sort of functionality. Unfortunately, Spotify has very restrictive requirements for authentication, and without OAuth, we could only utilize the least-privileged level of authentication that does not support playback.