SpoTweety

SpotTweety is a web application which is planned, designed and implemented for Software Development Practice (SWE 573) 2017 Spring project.

SpoTweety uses Twitter API as a data source. It collects tweets from Twitter which is shared via any Spotify App. After analyzing the data it shows the most popular songs of that area depending on the share rate.

Author: Fatih Aracı

Repository: https://github.com/fatiharaci/Spring2017Swe573

Note: Please ask any change requests on Github's issue tracker…

Functional Requirements

1. User Accesability
   1.1. Any user shall be able to access the web page. No login required.
2. Top Songs Search Operations
   2.1. User shall enter location info as coordinates.
   2.2. User shall be able to select location in order to provide coordinates.
   2.3. User shall be able to see trend of songs for a given time-line.
3. Collected Data Operations
   3.1. System shall be able to analyze any tweet collected.
   3.2. System shall be able to count number of occurances of songs on tweets.
   3.3. System shall be able to compare the occurances of songs in order to create top 20 songs.
4. Results
   4.1. User shall be able to see the top 20 popular songs as a list.
   4.2. User shall be able to listen the songs directly from the list.

Non-functional requirements

1. The system shall be both resource-efficient and scalable. That is, as the number of users increases, the system, by adding new resources, should work as if there is only one user. This scalability should be satisfied in a distributed manner.

2. The system should be usable. In other words, users should not need to be trained to use the system.
3. The system should adopt and follow certain quality standards such as W3C Html/XHtml Markup Standards, most common coding/documenting styles(for instance PEP8 for Python)
4. The system shall have responsive user interfaces so that it can be more accessible. It shall support modern web browsers such as Firefox or Chrome and their corresponding mobile versions.
5. The system shall use and contain only open source technologies, libraries, and tools.
6. The system shall be fast enough not to bother its users. Any user interaction should not take longer than 3 seconds in a local development environment, except 3rd party API transactions.
7. The system shall expose an HTTP RESTful API which supports every user interactions which can also be done via the user interfaces of the system. This API shall have complete documentation.
8. The system shall have unit tests and functional tests for the back-end side. Test coverage shall be at least 70%
9. The system shall be able to be deployed in a virtual environment which is completely isolated from operating systems.