**Software Requirements and Design Document**

**For**

**Group Jingle**

Version 1.0

**Authors**:

John Barden

Jack Splaine

Kohin Khandwalla

Gary Bowen

Dante Coupet

# Overview (5 points)

This web application uses is built on the Django framework. The function of the app is to accept a query regarding a song, while returning a plethora of information, including genius lyrics, the album cover and track information, and links to related YouTube videos.

The application utilizes the genius API to find song lyrics, the spotify API to find song information, and the youtube API to find related videos. These all work in unison to return all of the information to the user on the results page.

# Functional Requirements (10 points)

1.) Allow the user to easily navigate between pages and execute a search without error – HIGH.

2.) Spotify API: Effectively return relevant information to allow the user to find the song they wish to look at quickly and easily – HIGH.

3.) Genius API: Find the correct lyrics to the song that was clicked on by the user – MEDIUM.

4.) YouTube API: find relevant videos involving or about the song and present them neatly to the user – MEDIUM.

# Non-functional Requirements (10 points)

1.) Aesthetics – have the website be visually pleasing, avoid an outdated ui, and

Have some pleasing animations/effects.

2.) User-interface - make sure the website is easy to navigate and the controls are

Intuitive

3.) Search efficiency – searches should not take long to return the appropriate amount of information

*.*

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

# Class Diagram and/or Sequence Diagrams (15 points)

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

# Operating Environment (5 points)

The app will be running on a web server using the Django Framework. It is currently using Django version 3.0.4. It is also written in Python using Python 3. It also uses the Spotipy python library and the Beautiful Soup 4.0 Library, and the YouTube Data API version 3, which is called through a search done by the Google API service. The final product is planned to be hosted by a web hosting company. It is also being tested primarily on Mac computers being ran through the terminal, with our windows users using PyCharm.

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

# Assumptions and Dependencies (5 points)

Some aspects of the code are dependent on the Spotify, YouTube Data. and Beautiful Soup python libraries. If any of these were to be shut down or stop working, parts of the code would be affected*.* The whole application is running on the Django framework, so all aspects and libraries of the framework must be functional. The server is also mainly ran on Mac for testing, but windows uses slightly different versions of some of the software that could cause issues while testing by individual group members.