**Software Requirements and Design Document**

**For**

**Group Jingle**

Version 1.0

**Authors**:

Jack Splaine

Dante Coupet

Kohin Khandwalla

Gary Bowen

John Barden

# Overview (5 points)

Uses the Django framework to create a web application. The general use of the app is you type in the name of a song and it displays information about the song.

Firstly we use the Spotify API to get the song name, artist name, album name, song cover, song duration, featured artist, release date, and genres. This is done in the Spotifyxx.py file. The Genius API gets the lyrics of the song and the page views, this is gotten in Geniusxx.py. The page views then calls functions from the master\_results.py and this file compiles data from the other files and then returns it to the view to be showed

# Functional Requirements (10 points)

# Functional Django website, using the Django framework to get a working website to host our app – HIGH

* 1. Spotify API, using the Spotify API to gather the required information to be displayed – HIGH
  2. Genius API, using the Genius API to rather the required information to be displayed – MEDIUM
  3. Youtube API, using the Youtube API to rather the required information to be displayed – LOW
  4. Webpage Linking, having all webpages linked together, and made sure that required information is passed to appropriate webpages – HIGH

# Non-functional Requirements (10 points)

# Website Design and visuals, having website look good and have logos and visuals

* 1. User intuitiveness, making sure website works and isn’t confusing to a user
  2. Lyrical analysis, having a program that gives an analysis of a given songs lyrics
  3. Most searched, implementing a data base to store songs to be used to keep track of most search for songs
  4. Web hosting, host website using a service to people can access and use website from anywhere

# 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)

*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)

*List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*