**Software Requirements and Design Document**

**For**

**Group <Jingle>**

Version 3.0

**Authors**:

John Barden

Jack Splaine

Dante Coupet

Gary Bowen

Kohin Khandwalla

# Overview (5 points)

Jingle is a web application that runs a music search engine. It is built using python and the Django framework. The function of the project is to accept a search query and return the songs it finds along with a variety of relevant information one might wish to know about the song including but not limited to the album, release date, lyrics, related YouTube videos, links to relevant web pages, etc.

To retrieve the aforementioned information, the backend code uses API’s to return information relevant to search queries, as well as the most listened to songs(according to our API sources). Specifically, the ‘Spotipy’ API, Genius API, and YouTube Data API are used in unison in the project.

# Functional Requirements (10 points)

1.) Allow the user to easily navigate between pages. – HIGH

2.) Allow the user to execute a search without error. – HIGH

2.) Spotify API: Efficiently return relevant information to allow the user to find the song they wish to see easily – HIGH.

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

4.) YouTube API: When it is possible, find and return videos relevant to the queried song and present them to the user. – MEDIUM

5.) Feedback System: Allow a user to input a statement into the page and return it to a file belonging to one of us so that we can look at it. - MEDIUM

# Non-functional Requirements (10 points)

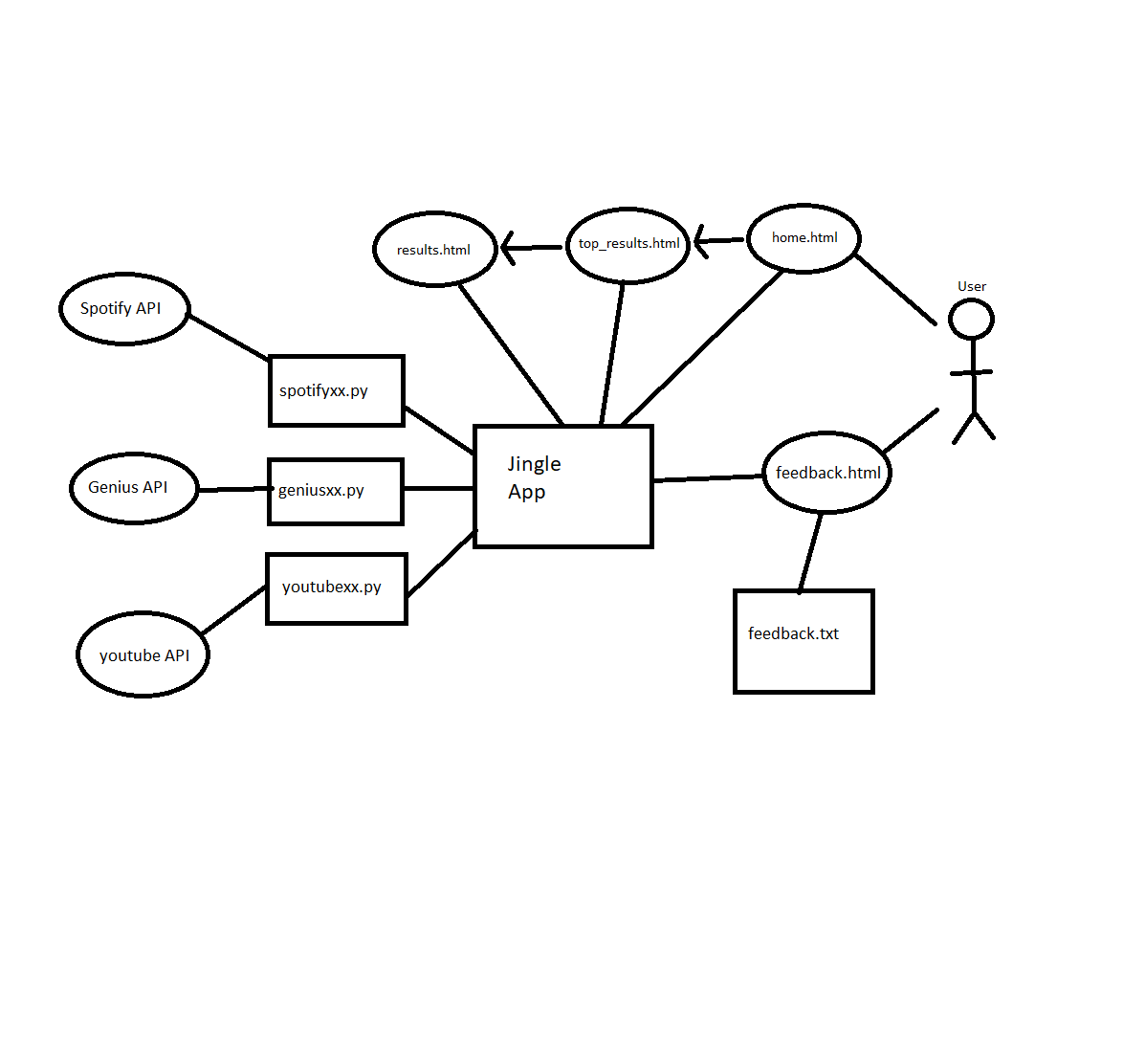
1.) Site aesthetics – Have the website look presentable, be visually pleasing, have some animation, and not look outdated

2.) User-interface – Ensure that the website is not at all difficult to navigate and controls are not counter-intuitive

3.) Search Efficiency – Ensure that searches do not take an unreasonable amount of time to return the requested information.

4.) Reliability – Keep up with the functionality of API’s being used, and return as much information as possible even if an API cannot be used.

# Use Case Diagram (10 points)



About.html

About.html

about

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

# 

# Operating Environment (5 points)

This app will run on a webserver(localhost if not online) using the Django framework. It is using Django version 3.0.4. It is written in Python, specifically Python 3. It 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 product is not currently hosted online at the time of submission. It is run on a localhost server for testing and demonstration. It can run on any common web browser using any platform that can run the said browser.

Website Link: <http://jingle.herokuapp.com/>

Website is exact same as our final product. Due to the use of a site hosting that was free and simple some features(mainly visuals) are not the same. The website is supposed to be for fun and demonstration and not an accurate representation of our final product (our video and source code is)

# Assumptions and Dependencies (5 points)

There are parts of the source code that are dependent on the Spotify, YouTube Data, and Beautiful Soup Libraries and APIs being put to use in our program. If any of these were to stop working, the application would no longer be able to obtain the information from that specific extension of the code. For the application to run smoothly and without errors, these APIs must function properly. The server has only been run on a Mac or PC platform for testing and is only expected to run on these for the now. Page layouts can vary in some ways depending on what type of machine the application is being used on and the screen resolution of said machine.