Our testing approach holds a combination of objective tests and subjective tests that may be difficult to quantify. Our plan is to first test the necessary, quantifiable test cases, such as verifying that our API endpoints are functional, that are databases are connected, and that performance is accurate. After this, we will be testing the result of our similarity algorithm, which is at least in part subjective, such that users are [illegible]as representative of both users' tastes. Finally, we will make sure that the UI is aesthetically pleasing and intuitive.

Test Cases:

1) **Fast Load Time Test Case;**
   This test is to ensure that users are admitted to the webapp quickly
   This test will ensure that all users apps load in less than a minute
   Input; user login / Auth
   Output; Home page
   Normal Boundary
   Black Box
   Performance
   Unit Test

2) **Access Spotify Data**
   This is the test to make sure that the program can successfully pull data from Spotify.
   This test will access data from Spotify after a user authorizes our program.
   Input: User Login / OAuth
   Output: Interface with user data
   Normal Boundary
   White Box
   Functional
   Integration Test

3) **Data is Saved to Database**
   This test is to ensure that data is properly saved upon new user login/user data expiration.
   Input: user login/Auth
   Output: User data is stored properly in database
   Normal Boundary
   Blackbox
   Functional
   Unit

4) **Database Connection Test**
   This test is to ensure that our program can connect to a cloud database to store data in.
   Input: database login
   Output: successful connection
   Normal Boundary
   Blackbox
   Functional
   Integration

5) **Plot Display Test**
   The purpose of this test is to verify that, upon request, a user will be able to see themselves upon other users and examine how similar they are to each other. The plot should be interactive and believable based on the songs that are available upon user request.
   Input: User's song data
   Output: interactive plot of users and their top songs, similarity score
   Normal
   Whitebox
   Functional
   Integration

6) **Spider Chart Works**
   This test is to ensure a user's homepage contains a spider chart of their Super Genres
   This test makes sure the user has a spider chart of their super genres displayed when they log into *SpotifyStats*.
   Input: User's Super Genre distribution
   Output: Super Genre Spider Chart
   Normal Boundary
   White Box
   Functional
   Integration

7) **Super Genre Map Helper**
   This is to ensure that a users songs are mapped to the correct genre
   This will cross reference users subgenres to the super genre map to make sure their genre distributions are accurate
   Input: List of users genres from top 100 songs
   Output: Genre Distribution
   Normal Boundary
   Black Box
   Functional

Unit Test

8) **User Similarity Ranking**

This is to ensure that user similarity/dissimilarity rankings work to subjective and objective criteria
Will use test users and manual data to validate rankings
inputs: test users that have only listened to specified genre playlists
outputs: test users rankings compared to other users
Normal boundary
White Box
Functional
Integration

9) **User Interface Data Transfer Efficiently**

Ensure efficient data transfer from backend to front end graphs
Input: Data objects from parser
Output: Data objects in front end
Normal Boundary
Black Box
Performance
Unit test

10) **User Interface Data Transfer Correctly**

Ensure clean data transfer from middleware/parser to front end graphs
Input: Data objects from parser
Output: Data objects in front end / Correct graphs
Normal Boundary
Black Box
Functional
Unit test

11) **User Accessibility Test**

Ensure that our site is easily navigable and understandable by the average user.
Input: Non-developer user navigating the website
Output: Review of the site's accessibility
Normal Boundary
White Box
Performance
Integration

Test Matrix

	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
1	**Normal**	**Black**	**Performance**	**Unit**
2	**Normal**	**White**	**Functional**	**Integration**
3	**Normal**	**Black**	**Functional**	**Unit**
4	**Normal**	**Black**	**Functional**	**Integration**
5	**Normal**	**White**	**Functional**	**Integration**
6	**Norma**	**White**	**Functional**	**Integration**
7	**Normal**	**Black**	**Functional**	**Unit**
8	**Normal**	**White**	**Functional**	**Integration**
9	**Normal**	**Black**	**Perfromance**	**Unit**
10	**Normal**	**Black**	**Functional**	**Unit**