Project #1
Testing Report

Group #6
Yawen (Maggie) Cao
Xiaoyang (Leo) Liao
Juntao (Kenneth) Shen
Jiachang (Ernest) Xu
Mengqian (Michelle) Yu
Lifan Zhao

Executive Summary

This document serves as the testing report for Lyrics Cloud, based on the *Software Requirements Specification* document provided by Team 17 of CSCI 310 (Software Engineering). This document (1) lists all the black-box test cases along with validations against requirements specification (a.k.a. the *Software Requirements Specification* document provided by Team 17), (2) lists all the white-box test cases along with validations and reports of coverage, and (3) provides process-related documentation and cross-references with this group's planning (a.k.a. the *Project Management Plan* document submitted by this group). These activities explained by this document are divided into three chapters: Chapter 2: Black-Box Testing, Chapter 3: White-Box Testing, and Chapter 4: Description of Testing Process.

Chapter 2 emphasizes on black-box testing techniques of the testing phase of this project. This group uses Cucumber as the primary tool of black-box testing. All the test cases of black-box testing are listed in this chapter, along with their instruction and information, including designation, setup/start state, input, expected output, and actual output. The test-case generation process of black-box testing correlates to the 10 major functional requirements in the requirements specification (see Section 3.2 of *Software Requirements Specification* provided by Team 17). This group uses the behavioral logic behind each of the 10 major functional requirements to generate each black-box test case. Each black-box test case is validated against each smallest piece of functional requirement in *Software Requirement Specification*.

Chapter 3 emphasizes on white-box testing techniques of the testing phase of this project. This group uses PHPUnit as the primary tool of white-box testing. This chapter includes the description of test case generation process, validation of all the test cases, and report of coverage. This project includes two PHP files: sortFrequency.php and wordcount.php. This group test every internal method inside each PHP file separately, and then tests the PHP file collectively as an individual class.

Chapter 4 describes this group's testing process, and ensures that this group is following a well-planned process instead of mindlessly hacking. Most of this group's testing process is compliant with the original planning (a.k.a. the *Project Management Plan* document submitted by this group). However, due to unforeseen complexity of the actual testing process, this group has to rearrange subteams; due to the postponed testing deadline, this group rescheduled the testing process, but finished the testing process according to the new schedule.

Table of Contents

1. Introduction

1.1 Project Overview and Purpose

Upon the STAKEHOLDERS' request, this project is designed to generate word clouds for lyrics of specified artists. This system is called Lyrics Cloud. This document specifically walks through every detail about the implementation process of this project. All the implemented features are based on the Software Requirements Specification document provided by Group #17 of CSCI 310 (Software Engineering).

This document explains how this group uses black-box testing technique, and white-box testing technique to complete the testing phase of this group. Specifically, this document lists all the test cases, describes the test case generation process, validates all the test cases, and provides reports of coverage if necessary.

1.2 Intended Audiences

This document is drafted for the intended viewing of all the STAKEHOLDERS, who are the teaching staff of CSCI 310 (Software Engineering), led by Professor William G. J. Halfond.

1.3 Definitions and Acronyms

STAKEHOLDERs: the teaching staff of CSCI 310 (Software Engineering), led by Professor William G. J. Halfond.

***Software Requirements Specification (SRS)*:** the document generated during the requirements engineering phase of this project. The current version of the *Software Requirements Specification* document is the one drafted by Team 17.

***Project Management Plan (PMP)*:** the document generated during the planning phase of this project. The current version of the *Project Management Plan* document is the one submitted by this group.

***Software Design*:** the document generated during the design phase of this project. The current version of the *Software Design* document is the one submitted by this group.

***Implementation Report*:** the document generated during the implementation phase of this project. The current version of the *Implementation Report* document is the one submitted by this group.

Lyrics Cloud: the system which will generate a clickable word cloud for the lyrics of a specified artist. Please refer to the *Software Requirements Specification* document drafted by Team 17 for details.

Client: the user who uses the Lyrics Cloud.

Subteam: the 2-person organizational structure under this group.

Subteam 1: the first of the 3 subteams from this group, composed by Ernest and Leo.

Subteam 2: the second of the 3 subteams from this group, composed by Lifan and Mengqian.
Subteam 3: the third of the 3 subteams from this group, composed by Juntao and Yawen.

1.4 References

[1] Team 17. “Project #1 Software Requirements Specification”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHUjdnTDhabi1IWlk. Accessed: March 7th, 2017.

[2] This Group. “Project #1 Project Management Plan”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHZ3BycjlSS00zeDQ. Accessed: March 7th, 2017.

[3] This Group. “Project #1 Software Design”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHdThBWFE0OEt5YUU. Accessed: March 7th, 2017.

[4] This Group. “Project #1 Implementation Report”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHQVFBZ3ZfM0QzWFU. Accesses: March 7th, 2017.

[5] Spotify AB. “Spotify Developer”. [Online]. Available: https://developer.spotify.com/. Accessed: March 7th, 2017.

[6] Musixmatch. “Musixmatch Developer API”. [Online]. Available: https://developer.musixmatch.com/. Accessed: March 7th, 2017.

[7] Facebook. “Facebook for Developers”. [Online]. Available: https://developers.facebook.com/. Accessed: March 7th, 2017.

2. Black-Box Testing

This chapter emphasizes on black-box testing techniques of the testing phase of this project. This group uses Cucumber as the primary tool of black-box testing. This chapter includes all the test cases of black-box testing, the description of test case generation process, and validation of all the test cases.

2.1 Test Cases of Black-Box Testing

This section lists all the test cases of black-box testing, and their instruction and information, including designation, setup/start state, input, expected output, and actual output. The test-case generation process of black-box testing correlates to the 10 major functional requirements in the requirements specification (see Section 3.2 of *Software Requirements Specification* provided by Team 17). This group uses the behavioral logic behind each of the 10 major functional requirements to generate each black-box test case. For information about correlation between each test case and each major functional requirement, please proceed to Section 2.2 (Validation of Black-Box Testing) of this document for details.

2.1.1 Black-Box Test Suite 01: Artist Search Bar

Designation	Black-Box Test Case 01.1 (abbr. BTC_01.1)
Reference	REQ-1, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	N/A
Expected Output	The Artist Search Bar is empty, and is in NO-SELECTION state.
Actual Output	The Artist Search Bar is empty, and is in NO-SELECTION state.

Designation	Black-Box Test Case 01.2 (abbr. BTC_01.2)
Reference	REQ-2, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user clicks the Search Button.
Input	The user types a word in the Artist Search Bar.
Expected Output	The Artist Search Bar is editable.
Actual Output	The Artist Search Bar is editable.

Designation	Black-Box Test Case 01.3 (abbr. BTC_01.3)
Reference	REQ-3, Subsection 3.2, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	The user types a word in the Artist Search Bar.
Expected Output	The Artist Search Bar is in NO-SELECTION state.
Actual Output	The Artist Search Bar is in NO-SELECTION state.

Designation	Black-Box Test Case 01.4.1 (abbr. BTC_01.4.1)
Reference	REQ-4, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page. The Artist Search Bar is in NO-SELECTION state.
Input	The user types 1 character in the Artist Search Bar.
Expected Output	The suggestion drop-down list is not available.
Actual Output	The suggestion drop-down list is not available.

Designation	Black-Box Test Case 01.4.2 (abbr. BTC_01.4.2)
Reference	REQ-4, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page. The Artist Search Bar is in NO-SELECTION state.
Input	The user types 2 characters in the Artist Search Bar.
Expected Output	The suggestion drop-down list is not available.
Actual Output	The suggestion drop-down list is not available.

Designation	Black-Box Test Case 01.4.3 (abbr. BTC_01.4.3)
Reference	REQ-4, Subsection 3.2.3, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page. The Artist Search Bar is in NO-SELECTION state.

Input	The user types 3 characters in the Artist Search Bar.
Expected Output	The suggestion drop-down list is available.
Actual Output	The suggestion drop-down list is available.

Designation	Black-Box Test Case 01.4.4 (abbr. BTC_01.4.4)
Reference	REQ-4, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page. The Artist Search Bar is in NO-SELECTION state.
Input	The user types 4 characters in the Artist Search Bar.
Expected Output	The suggestion drop-down list is available.
Actual Output	The suggestion drop-down list is available.

Designation	Black-Box Test Case 01.5 (abbr. BTC_01.5)
Reference	REQ-5, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	The user types “Cold” in the Artist Search Bar.
Expected Output	The suggestions drop-down contain at least 3 artists.
Actual Output	The suggestions drop-down contain at least 3 artists.

Designation	Black-Box Test Case 01.6 (abbr. BTC_01.6)
Reference	REQ-6, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	The user types “Cold” in the Artist Search Bar.
Expected Output	Each artist in the suggestions drop-down contain an artist name and an image of the artist.
Actual Output	Each artist in the suggestions drop-down contain an artist name.

Designation	Black-Box Test Case 01.7 (abbr. BTC_01.7)
Reference	REQ-7, Subsection 3.2.1, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	1. The user types “Cold’ in the Artist Search Bar. 2. The user clicks “ColdPlay” in the suggestions drop-down.
Expected Output	1. The Artist Search Bar is updated with “ColdPlay”. 2. The user navigates to the Word Cloud Page.
Actual Output	1. The Artist Search Bar is updated with “ColdPlay”. 2. The user navigates to the Word Cloud Page.

2.1.2 Black-Box Test Suite 02: Search Button

Designation	Black-Box Test Case 02.1 (abbr. BTC_02.1)
Reference	REQ-1, Subsection 3.2.2, *Software Requirements Specification*
Setup/Start State	The Artist Search Bar is empty.
Input	1. The user inputs some text into the Artist Search Bar. 2. The user deletes all the text in the Artist Search Bar.
Expected Output	The Search Button is not clickable initially but it becomes clickable after the user inputs any text. The Search button is clickable until the user clears off the text in Artist Search Bar.
Actual Output	The Search Button is not clickable initially but it becomes clickable after the user inputs any text. The Search button is clickable until the user clears off the text in Artist Search Bar.

Designation	Black-Box Test Case 02.2 (abbr. BTC_02.2)
Reference	REQ-2, Subsection 3.2.2, *Software Requirements Specification*
Setup/Start State	Artist Search Bar has a artist name the user wants to search.
Input	The user click the Search Button.
Expected Output	The user navigates to the Word Cloud Page.
Actual Output	The user navigates to the Word Cloud Page.

Designation	Black-Box Test Case 02.3 (abbr. BTC_02.3)
Reference	REQ-3, Subsection 3.2.2, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page that is generated in Black-Box Test Case 02.2 (abbr. BTC_02.2)
Input	N/A
Expected Output	The Word Cloud Title is shown with the content of selected artist.
Actual Output	No title.

2.1.3 Black-Box Test Suite 03: Word Could Title

Designation	Black-Box Test Case 03.1 (abbr. BTC_03.1)
Reference	REQ-1, Subsection 3.2.3, *Software Requirements Specification*
Setup/Start State	The use is on the Artist Search Page, with neither names entered nor word clouds generated.
Input	1. Enter and choose the name of a valid artist. 2. Click the Search Button. 3. Enter and choose the names of more valid artists. 4. Click the Add Button.
Expected Output	The label's text lists the name(s) of the inputted artist(s), delimited by commas.
Actual Output	The label's text lists the name(s) of the inputted artist(s), but not delimited by commas.

2.1.4 Black-Box Test Suite 04: Word Cloud

Designation	Black-Box Test Case 04.1 (abbr. BTC_04.1)
Reference	REQ-1, Subsection 3.2.4, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	N/A
Expected Output	The Word Cloud does not contain any Commonly Used Word.
Actual Output	The Word Cloud does not contain any Commonly Used Word.

Designation	Black-Box Test Case 04.2 (abbr. BTC_04.2)
Reference	REQ-2, Subsection 3.2.4, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	1. The user types “ColdPlay” in the Artist Search Bar. 2. The user clicks Artist Search Button.
Expected Output	1. The user navigates to the Word Cloud Page. 2. The Word Cloud contains 250 words omitting any Commonly Used Word.
Actual Output	1. The user navigates to the Word Cloud Page. 2. The Word Cloud contains less than 250 words omitting any Commonly Used Word.

Designation	Black-Box Test Case 04.3 (abbr. BTC_04.3)
Reference	REQ-3, Subsection 3.2.4, *Software Requirements Specification*
Setup/Start State	The user is on the Artist Search Page.
Input	1. The user types “Duru, Adanna” in the Artist Search Bar. 2. The user clicks Artist Search Button.
Expected Output	1. The user navigates to the Word Cloud Page. 2. The Word Cloud contains 250 words omitting any Commonly Used Word.
Actual Output	1. The user navigates to the Word Cloud Page. 2. The Word Cloud contains less than 250 words omitting any Commonly Used Word.

Designation	Black-Box Test Case 04.4 (abbr. BTC_04.4)
Reference	REQ-4, Subsection 3.2.4, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	N/A
Expected Output	The words in the Word Cloud are horizontal.

Actual Output	The words in the Word Cloud are horizontal.

Designation	Black-Box Test Case 04.5 (abbr. BTC_04.5)
Reference	REQ-5, Subsection 3.2.5, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	N/A
Expected Output	The Word Cloud is rectangular.
Actual Output	The Word Cloud is rectangular.

Designation	Black-Box Test Case 04.6 (abbr. BTC_04.6)
Reference	REQ-6, Subsection 3.2.6, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	N/A
Expected Output	The words in the Word Cloud are colorful.
Actual Output	The words in the Word Cloud are black and white.

Designation	Black-Box Test Case 04.7 (abbr. BTC_04.7)
Reference	REQ-7, Subsection 3.2.7, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	N/A
Expected Output	The words in the Word Cloud are horizontal.
Actual Output	The words in the Word Cloud are horizontal.

Designation	Black-Box Test Case 04.8 (abbr. BTC_04.8)
Reference	REQ-8, Subsection 3.2.8, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.

Input	N/A
Expected Output	The size of the word is proportional to the frequency of the word in the lyrics of the artist.
Actual Output	The size of the word is proportional to the frequency of the word in the lyrics of the artist.

Designation	Black-Box Test Case 04.9 (abbr. BTC_04.9)
Reference	REQ-9, Subsection 3.2.9, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page.
Input	1. The user types “ColdPlay”. 2. The user clicks “time” in the Word Cloud.
Expected Output	1. The user navigates to the Song List Page. 2. In the Song List Page, the Song List Title and Song List are initialized with the artist(s) and the clicked word.
Actual Output	1. The user navigates to the Song List Page. 2. In the Song List Page, the Song List Title and Song List are initialized with the artist(s) and the clicked word.

2.1.5 Black-Box Test Suite 05: Add Button

Designation	Black-Box Test Case 05.1 (abbr. BTC_05.1)
Reference	REQ-1, Subsection 3.2.5, *Software Requirements Specification*
Setup/Start State	The Artist Search Bar is empty.
Input	1. The user inputs some text into the Artist Search Bar. 2. The user deletes all the text in the Artist Search Bar.
Expected Output	The Add Button is not clickable initially but it becomes clickable after the user inputs any text. The Add button is clickable until the user clears off the text in Artist Search Bar.
Actual Output	The Add Button is not clickable initially but it becomes clickable after the user inputs any text. The Add button is clickable until the user clears off the text in Artist Search Bar.

Designation	Black-Box Test Case 05.2 (abbr. BTC_05.2)
Reference	REQ-2, Subsection 3.2.5, *Software Requirements Specification*
Setup/Start State	The user finishes Black-Box Test Case 05.1. The user stays on that Word Cloud Page
Input	The user clicks the Add Button.
Expected Output	After seconds of processing, the Word Cloud Title refreshes itself with the previous artists' names and the selected names.
Actual Output	After seconds of processing, the Word Cloud Title refreshes itself with the previous artists' names and the selected names.

Designation	Black-Box Test Case 05.3 (abbr. BTC_05.3)
Reference	REQ-3, Subsection 3.2.5, *Software Requirements Specification*
Setup/Start State	The user finishes Black-Box Test Case 05.1. The user stays on that Word Cloud Page.
Input	The user clicks the Add Button.
Expected Output	After seconds of processing, the word cloud refreshes itself with a new word cloud based on the previous artist and the selected artist.
Actual Output	After seconds of processing, the word cloud refreshes itself with a new word cloud based on the previous artist and the selected artist.

2.1.6 Black-Box Test Suite 06: Share Button

Designation	Black-Box Test Case 06.1 (abbr. BTC_06.1)
Reference	REQ-1, Subsection 3.2.6, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page with a generated Word Cloud.
Input	The user clicks on the Share Button.
Expected Output	The user is redirected to a Facebook website with the action of creating a new post.
Actual Output	The user is not redirected to a Facebook website with the action of creating a new post.

Designation	Black-Box Test Case 06.2 (abbr. BTC_06.2)
Reference	REQ-2, Subsection 3.2.6, *Software Requirements Specification*
Setup/Start State	The user is on the Word Coud Page with a generated Word Cloud
Input	The user clicks the Share Button.
Expected Output	An image of the Word Cloud and a text which lists the artist(s) are ready to post.
Actual Output	Share Button is not working.

Designation	Black-Box Test Case 06.3.1 (abbr. BTC_06.3.1)
Reference	REQ-3, Subsection 3.2.6, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page with a generated Word Cloud.
Input	1. The user clicks the Share Button. 2. The user inputs with correct Facebook username and password.
Expected Output	1. The user login is accepted. 2. The user successfully shares the Word Cloud image.
Actual Output	The user login fails.

Designation	Black-Box Test Case 06.3.2 (abbr. BTC_06.3.2)
Reference	REQ-3, Subsection 3.2.6, *Software Requirements Specification*
Setup/Start State	The user is on the Word Cloud Page with a generated Word Cloud.
Input	3. The user clicks the Share Button. 4. The user inputs with incorrect Facebook username and password.
Expected Output	The user login is rejected.
Actual Output	Share Button is not working.

2.1.7 Black-Box Test Suite 07: Song List Title

Designation	Black-Box Test Case 07.1 (abbr. BTC_07.1)

Reference	REQ-1, Subsection 3.2.7, *Software Requirements Specification*
Setup/Start State	The user is on the Song List Page.
Input	N/A
Expected Output	The label's text consists of the word that Song List Title is initialized with.
Actual Output	The label's text consists of the word that Song List Title is initialized with.

2.1.8 Black-Box Test Suite 08: Song List

Designation	Black-Box Test Case 08.1 (abbr. BTC_08.1)
Reference	REQ-1, Subsection 3.2.8, *Software Requirements Specification*
Setup/Start State	The user is on the Song List Page with a keyword "time" and a specified artist "ColdPlay".
Input	N/A
Expected Output	The list contains all songs by the artist with the keyword.
Actual Output	The list contains all songs by the artist with the keyword.

Designation	Black-Box Test Case 08.2 (abbr. BTC_08.2)
Reference	REQ-2, Subsection 3.2.8, *Software Requirements Specification*
Setup/Start State	The user is on the Song List Page with a keyword "time" and a specified artist "ColdPlay".
Input	N/A
Expected Output	The songs are sorted according to the frequency of the keyword in each song, descending.
Actual Output	The songs are sorted according to the frequency of the keyword in each song, descending.

Designation	Black-Box Test Case 08.3 (abbr. BTC_08.3)
Reference	REQ-3, Subsection 3.2.8, *Software Requirements Specification*

Setup/Start State	The user is on the Song List Page with a keyword “time” and a specified artist “ColdPlay”.
Input	N/A
Expected Output	Each entry in the song list consists of the title of the corresponding song and the number of frequency of the keyword in that song.
Actual Output	Each entry in the song list consists of the title of the corresponding song and the number of frequency of the keyword in that song.

Designation	Black-Box Test Case 08.4 (abbr. BTC_08.4)
Reference	REQ-4, Subsection 3.2.8, *Software Requirements Specification*
Setup/Start State	The user is on the Song List Page with a keyword “time” and a specified artist “ColdPlay”.
Input	The user clicks any one of the song titles.
Expected Output	The user navigates to the Lyrics Page.
Actual Output	The user navigates to the Lyrics Page.

Designation	Black-Box Test Case 08.5 (abbr. BTC_08.5)
Reference	REQ-5, Subsection 3.2.8, *Software Requirements Specification*
Setup/Start State	The user is on the Lyrics Page.
Input	N/A
Expected Output	The Lyrics Title and Lyrics are initialized with the word and clicked song.
Actual Output	The Lyrics Title and Lyrics are initialized with the word and clicked song.

2.1.9 Black-Box Test Suite 09: Lyrics Title

Designation	Black-Box Test Case 09.1 (abbr. BTC_09.1)
Reference	REQ-1 & REQ-2, Subsection 3.2.9, *Software Requirements Specification*

Setup/Start State	The user is on the Lyrics Page.
Input	N/A
Expected Output	The label's text consists of the word the Song List Title is initialized with.
Actual Output	The label's text consists of the word the Song List Title is initialized with.

2.1.10 Black-Box Test Suite 10: Lyrics

Designation	Black-Box Test Case 10.1 (abbr. BTC_10.1)
Reference	REQ-1, Subsection 3.2.10, *Software Requirements Specification*
Setup/Start State	The user is on the Lyrics Page.
Input	N/A
Expected Output	The text area contains the lyrics of the song that the Lyrics feature is initialized with.
Actual Output	The text area contains the lyrics of the song that the Lyrics feature is initialized with.

Designation	Black-Box Test Case 10.2 (abbr. BTC_10.2)
Reference	REQ-2, Subsection 3.2.10, *Software Requirements Specification*
Setup/Start State	The user is on the Lyrics Page.
Input	N/A
Expected Output	In the text area, any occurrence of the word that the Lyrics feature is initialized with are highlighted in yellow.
Actual Output	In the text area, any occurrence of the word that the Lyrics feature is initialized with are highlighted in yellow.

2.2 Validation of Black-Box Testing

This section validates all the test cases against the requirements specification (a.k.a. *Software Requirements Specification*). Specifically, this section explains how each black-box test

case reflects one of the many functional requirements in the *Software Requirements Specification*.

2.2.1 Black-Box Test Suite 01: Artist Search Bar

According to the requirements specification (see Subsection 3.2.1 of *Software Requirements Specification* provided by Team 17), the Artist Search Bar is required to remain editable, to display the suggestion drop-down list, and to maintain a state of NO-SELECTION unless clicking on an artist from the suggestion drop-down list.

BTC_01.1 correlates to REQ-1. It examines whether the Artist Search Bar, by default, is empty and has a state of NO-SELECTION.

BTC_01.2 correlates to REQ-2. It examines whether the Artist Search Bar is always editable by the user.

BTC_01.3 correlates to REQ-3. It examines whether the Artist Search Bar maintains a state of NO-SELECTION, while the user is editing it.

BTC_01.4.1 through BTC_01.4.4 correlate to REQ-4. They examines whether the suggestion drop-down list is available after 1, 2, 3, 4 or more characters are entered into the Artist Search Bar.

BTC_01.5 correlates to REQ-5. It examines whether the suggestion drop-down list contains at least 3 names of approximation.

BTC_01.6 correlates to REQ-6. It examines whether each artist in the suggestion drop-down list has his/her name and image displayed.

BTC_01.7 correlates to REQ-7. It examines whether the Artist Search Bar adopts a state of YES-SELECTION, after the user clicks on one artist in the suggestion drop-down list.

2.2.2 Black-Box Test Suite 02: Search Button

According to the requirements specification (see Subsection 3.2.2 of *Software Requirements Specification* provided by Team 17), the Search Button is required to maintain unclickable unless the Artist Search Bar has a state of YES-SELECTION, and to navigated to a Word Cloud Page with Word Cloud displayed.

BTC_02.1 correlates to REQ-1. It examines whether the Search Button remains unclickable unless the Artist Search Bar has a state of YES-SELECTION.

BTC_02.2 correlates to REQ-2. It examines whether the user is navigated to a Word Cloud Page, if he/she clicks the Search Button.

BTC_02.3 correlates to REQ-3. It examines whether the Word Cloud Page displays the Word Cloud Title, which contains the name(s) of selected artist(s), and the Word Cloud generated for selected artists.

2.2.3 Black-Box Test Suite 03: Word Cloud Title

According to the requirements specification (see Subsection 3.2.3 of *Software Requirements Specification* provided by Team 17), the Word Cloud Title is required to list the name(s) of the specified artist(s).

BTC_03.1 correlates to REQ-1. It examines whether the Word Cloud Title display the names of all the specified artists after one or more selections of artists.

2.2.4 Black-Box Test Suite 04: Word Cloud

According to the requirements specification (see Subsection 3.2.4 of *Software Requirements Specifications* provided by Team 17), the Word Cloud is required to omit commonly used words, to contain the 250 most frequently occuring words, to maintain a horizontal, rectangular, colorful, clickable display.

BTC_04.1 correlates to REQ-1. It examines whether the Word Cloud omites commonly used words (e.g. articles, pronouns, and prepositions).

BTC_04.2 correlates to REQ-2. It examines whether the Word Cloud contains the 250 most frequently occuring words from the lyrics of the selected artist(s).

BTC_04.3 correlates to REQ-3. It examines whether the Word Cloud contains as many words as possible is the lyrics of the artist(s) contains less than 250 eligible words.

BTC_04.4 correlates to REQ-4. It examines whether the Word Cloud is horizontal.

BTC_04.5 correlates to REQ-5. It examines whether the Word Cloud is rectangular.

BTC_04.6 correlates to REQ-6. It examines whether the Word Cloud is colorful.

BTC_04.7 correlates to REQ-7. It examines whether the size of the keywords in the Word Cloud is proportional to their frequencies in the lyrics of the selected artist(s).

BTC_04.8 correlates to REQ-8. It examines whether the user is navigated to a Song List Page, if he/she clicks one of the keywords in the Word Cloud.

BTC_04.9 correlates to REQ-9. It examines whether the Song List Page consists of a Song List Title, which displays the selected keyword, and the Song List, which contains all the songs by the specified artist(s) that contains the selected keywords.

2.2.5 Black-Box Test Suite 05: Add Button

According to the requirements specification (see Subsection 3.2.5 of *Software Requirements Specification* provided by Team 17), the Add Button is required to maintain unclickable unless the Artist Search Bar has a state of YES-SELECTION, and to add the selected artist to the Word Cloud Page, with a regenerated Word Cloud.

BTC_05.1 correlates to REQ-1. It examines whether the Add Button remains unclickable unless the Artist Search Bar has a state of YES-SELECTION.

BTC_05.2 correlates to REQ-2. It examines whether the selected artist is added to the Word Cloud Page.

BTC_05.3 correlates to REQ-3. It examines whether the selected artist is added to the Word Cloud Page.

2.2.6 Black-Box Test Suite 06: Share Button

According to the requirements specification (see Subsection 3.2.6 of *Software Requirements Specification* provided by Team 17), the Share Button is required to redirect the user to the Facebook website, and create a new post with an image of the Word Cloud and a text of the list of the selected artists.

BTC_06.1 correlates to REQ-1. It examines whether the user is redirected to the Facebook website, if he/she clicks the Share Button.

BTC_06.2 correlates to REQ-2. It examines whether the creation of a new Facebook post consists of an image of the Word Cloud and a text of the list of the selected artists.

BTC_06.3.1 through BTC_06.3.2 correlate to REQ-3. They examine whether login functionality is handled by the Facebook website.

2.2.7 Black-Box Test Suite 07: Song List Title

According to the requirements specification (see Subsection 3.2.7 of *Software Requirements Specification* provided by Team 17), the Song List Title is required to display the selected keyword.

BTC_07.1 correlates to REQ-1. It examines whether the Song List Title displays the selected keyword.

2.2.8 Black-Box Test Suite 08: Song List

According to the requirements specification (see Subsection 3.2.8 of *Software Requirements Specification* provided by Team 17), the Song List is required to display a list of all the songs, by the specified artist(s), which contains the selected keyword.

BTC_08.1 correlates to REQ-1. It examines whether the Song List displays all the songs by the artist(s), which contain the selected keyword.

BTC_08.2 correlates to REQ-2. It examines whether the Song List is sorted in the descending order of the frequency of the selected keyword in each song.

BTC_08.3 correlates to REQ-3. It examines whether each entry of the Song List consists of the title of a song in that list, followed by the frequency of the selected keyword in that song.

BTC_08.4 correlates to REQ-4. It examines whether the user is navigated to a Lyrics Page, if he/she clicks on the title of a song in the Song List.

BTC_08.5 correlates to REQ-5. It examines whether the Lyrics Page displays a Lyrics Title and the Lyrics.

2.2.9 Black-Box Test Suite 09: Lyrics Title

According to the requirements specification (see Subsection 3.2.9 of *Software Requirements Specification* provided by Team 17), the Lyrics Title is required to display the title of the selected song, and the name of the artist who sings that song, in the format "SONG_TITLE by ARTIST_NAME".

BTC_09.1 correlates to REQ-1 and REQ-2. It examines whether the Lyrics title displays the title of the selected song, and the name of the artist who sings that song, in the format "SONG_TITLE by ARTIST_NAME".

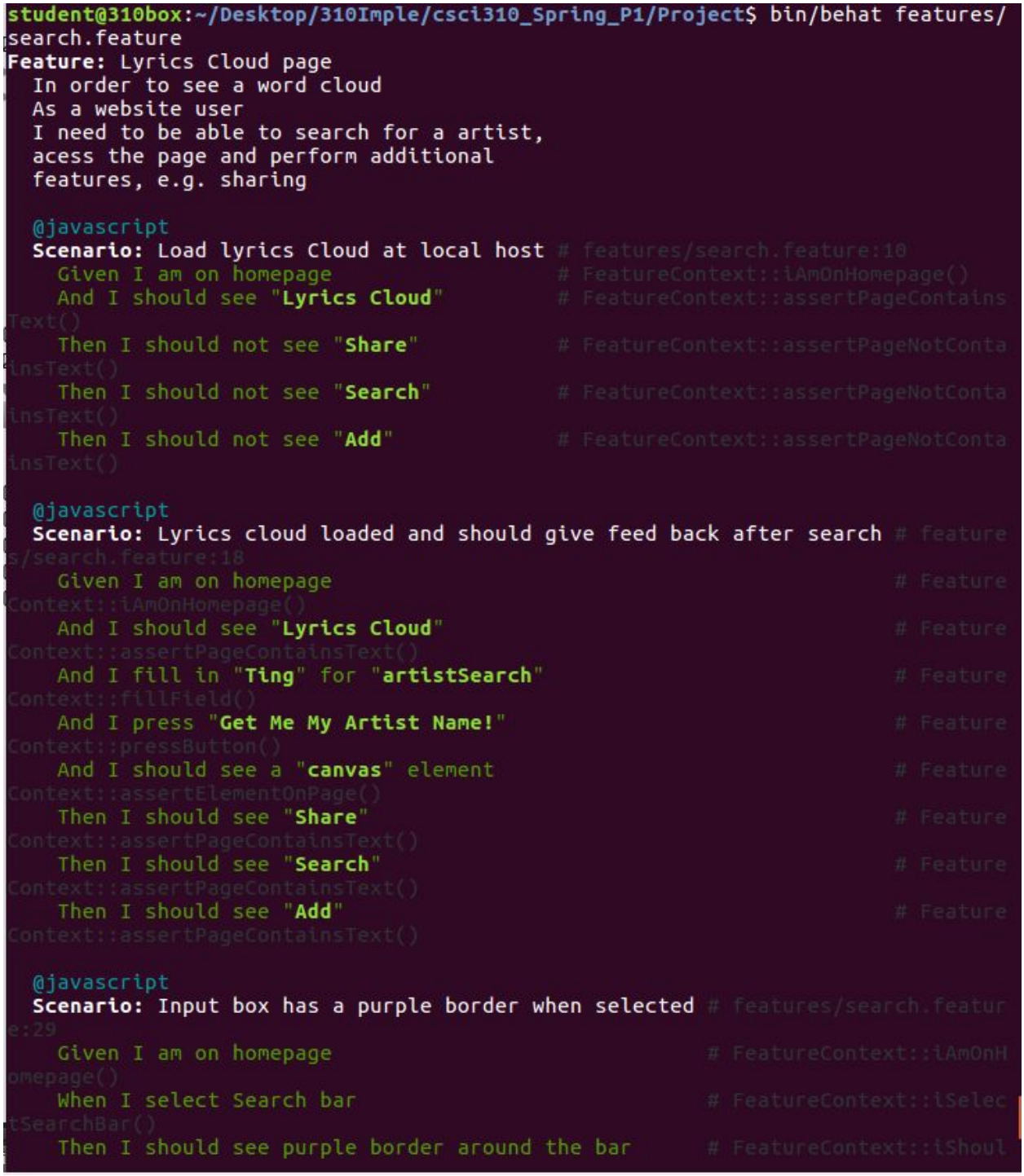
2.2.10 Black-Box Test Suite 10: Lyrics

According to the requirements specification (see Subsection 3.2.10 of *Software Requirements Specification* provided by Team 17), the Lyrics is required to display the lyrics string of the selected song with the specified keyword highlighted in yellow.

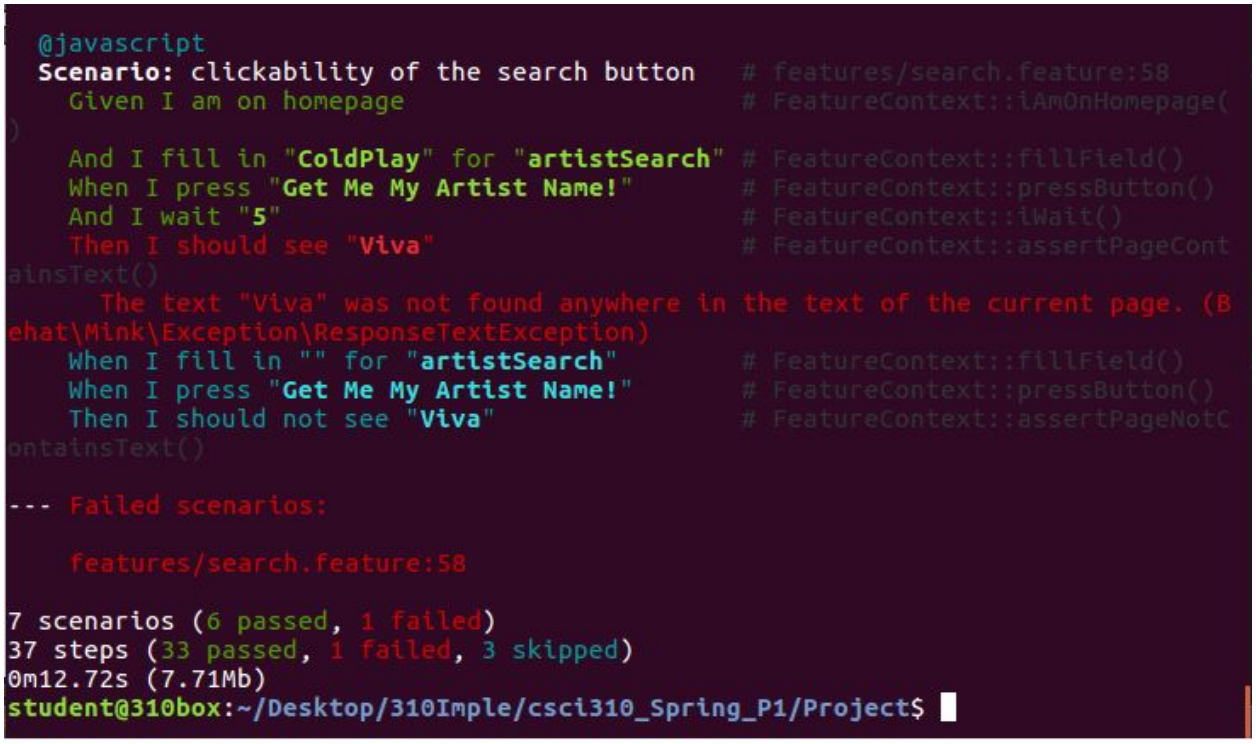
BTC_10.1 correlates to REQ-1. It examines whether the Lyrics displays the lyrics string of the selected song.

BTC_10.2 correlates to REQ-2. It examines whether the Lyrics highlights all the occurrences of the specified keyword in yellow.

2.2.11 Screenshots of Black-Box Testing



```
dSeePurpleBorderAroundTheBar()
    Then I should see background "gray"                  # FeatureContext::iShoul
dSeeBackground()
    And I should see text "black"                        # FeatureContext::iShoul
dSeeText()
lime Text
  @javascript
  Scenario: Zero item in search input bar at beginning # features/search.feature
:38
    Given I am on homepage                              # FeatureContext::iAmOnHo
mepage()
    And I should see "Lyrics Cloud"                     # FeatureContext::assertP
ageContainsText()
    Then the "artistSearch" field should contain ""     # FeatureContext::assertF
ieldContains()

  @javascript
  Scenario: Search bar is edible                        # features/search.featur
e:44
    Given I am on homepage                              # FeatureContext::iAmOnH
omepage()
    And I fill in "Ting" for "artistSearch"             # FeatureContext::fillFi
eld()
    Then the "artistSearch" field should contain "Ting" # FeatureContext::assert
FieldContains()

  @javascript
  Scenario: search is usable                          # features/search.feature:50
    Given I am on homepage                            # FeatureContext::iAmOnHomepage
()
    And I fill in "Cold" for "artistSearch"           # FeatureContext::fillField()
    When I fill in "ColdPlay" for "artistSearch"      # FeatureContext::fillField()
    And I press "Get Me My Artist Name!"              # FeatureContext::pressButton()
    Then I should see a "canvas" element              # FeatureContext::assertElement
OnPage()

  @javascript
  Scenario: clickability of the search button    # features/search.feature:58
    Given I am on homepage                       # FeatureContext::iAmOnHomepage(
)
    And I fill in "ColdPlay" for "artistSearch" # FeatureContext::fillField()
    When I press "Get Me My Artist Name!"       # FeatureContext::pressButton()
    And I wait "5"                              # FeatureContext::iWait()
    Then I should see "Viva"                    # FeatureContext::assertPageCont
ainsText()
      The text "Viva" was not found anywhere in the text of the current page. (B
ehat\Mink\Exception\ResponseTextException)
    When I fill in "" for "artistSearch"        # FeatureContext::fillField()
    When I press "Get Me My Artist Name!"       # FeatureContext::pressButton()
    Then I should not see "Viva"                # FeatureContext::assertPageNotC
```


@javascript
Scenario: clickability of the search button # features/search.feature:58
Given I am on homepage # FeatureContext::iAmOnHomepage()
And I fill in "ColdPlay" for "artistSearch" # FeatureContext::fillField()
When I press "Get Me My Artist Name!" # FeatureContext::pressButton()
And I wait "5" # FeatureContext::iWait()
Then I should see "Viva" # FeatureContext::assertPageContainsText()
The text "Viva" was not found anywhere in the text of the current page. (Behat\Mink\Exception\ResponseTextException)
When I fill in "" for "artistSearch" # FeatureContext::fillField()
When I press "Get Me My Artist Name!" # FeatureContext::pressButton()
Then I should not see "Viva" # FeatureContext::assertPageNotContainsText()
--- Failed scenarios:
features/search.feature:58
7 scenarios (6 passed, 1 failed)
37 steps (33 passed, 1 failed, 3 skipped)
0m12.72s (7.71Mb)
student@310box:~/Desktop/310Imple/csci310_Spring_P1/Project$

```
student@310box:~/Desktop/310Imple/csci310_Spring_P1/Project$ bin/behat features/
share.feature
Feature: Share
  In order to share a generated
  word cloud. As a website user
  I need to be able to share the
  generated word cloud to
  Facebook with the premission
  from the me.

  @javascript
  Scenario: The user is not logged in          # features/share.feature:11
    Given I am on homepage                     # FeatureContext::iAmOnHomepage()
    And I fill in "Justin" for "artistSearch"  # FeatureContext::fillField()
    And I press "Get Me My Artist Name!"       # FeatureContext::pressButton()
    And I should see a "canvas" element        # FeatureContext::assertElementOn
Page()
    When I press "Share"                       # FeatureContext::pressButton()
    Then I should see text matching "Facebook" # FeatureContext::assertPageMatch
esText()
      The pattern "Facebook" was not found anywhere in the text of the current p
age. (Behat\Mink\Exception\ResponseTextException)
    And I should see text matching "Login in"  # FeatureContext::assertPageMatch
esText()

  @javascript
  Scenario: The user is logged in              # features/share.feature:22
    Given I am on homepage                     # FeatureContext::iAmOnHomepage()
    And I fill in "Justin" for "artistSearch"  # FeatureContext::fillField()
    And I press "Get Me My Artist Name!"       # FeatureContext::pressButton()
    And I should see a "canvas" element        # FeatureContext::assertElementOn
Page()
    When I press "Share"                       # FeatureContext::pressButton()
    Then I should see text matching "Post"     # FeatureContext::assertPageMatch
esText()
      The pattern "Post" was not found anywhere in the text of the current page.
 (Behat\Mink\Exception\ResponseTextException)
    And I should not see text matching "error" # FeatureContext::assertPageNotMa
tchesText()

--- Failed scenarios:

    features/share.feature:11
    features/share.feature:22

2 scenarios (2 failed)
14 steps (10 passed, 2 failed, 2 skipped)
0m5.50s (7.70Mb)
```

3. White-Box Testing

This chapter emphasizes on white-box testing techniques of the testing phase of this project. This group uses PHPUnit as the primary tool of white-box testing. This chapter includes the description of test case generation process, validation of all the test cases, and report of coverage.

3.1 Generation Process of White-Box Testing

This section explains the test generation process of white-box testing.

3.1.1 White-Box Test Suite 01: frequencySort.php

There is two methods to test in this class: getFrequency() and filterBadWords().

getFrequency() requires a lyrics string, and finds the frequencies of unique words. Therefore, this test suite passes in a lyrics string to test this method.

filterBadWords() requires a lyrics string and a list of predetermined bad words, and filters any of those bad words from this lyrics string. Therefore, this test suite passes in a lyrics string and a list of predetermined bad words to test method.

This test suit also test these two methods collectively to see if these two methods can work together.

3.1.2 White-Box Test Suite 02: wordcount.php

There is one method to test in this class: getFrequency().

This method requires a lyrics string and a specified keyword, and calculates the frequency of the specified keyword in this lyrics string. Therefore, this test suite passes in a lyrics string and a specified keyword to test this method.

3.1.3. White-Box Test Suite 03: request.js

This is a javascript file includes several consequent functions: getArtists, getAllAlbums, getTracks, getLyrics, parseLyrics and generateWordCloud. Since generateWordCloud serves as a drawing function and parseLyrics as a simple call function to frequencySort.php, we don't have White-Box Test on them (which should be done in Black Box Testing).

getArtists() requires an input name from searchbox and find the artist_id. Therefore, this test suite passes in a name and expect to get a id back.

getAllAlbums() requires an input of artist_id from previous function and find the all albums. Therefore, a list of album_ids is expected. There is some branches inside. It should check the correctness of parameter. It should check if the artist only has songs with lyrics.

getTracks() requires a list of album_id from previous function and find the all track_ids. Therefore, a list of track_ids is expected to be returned. It should check if the songs has lyrics or not.

getLyrics() requires a track_ids and find the lyrics through Musixmatch API. Therefore, a string variable is expected to be returned. It should check if the song is English or not by the parameter in API.

3.2 Validation of White-Box Testing

This section validates all the test suites of white-box testing, and reports their coverage.

3.2.1 White-Box Test Suite 01: frequencySort.php

Function	Input	Expected	Actual
frequencySort.php:: getFrequency	["abc","abc","abc","abc","bcd","bcd","bcd","abc","cde","cde"];	["abc"=>5, "bcd"=>3, "cde"=>2]	["abc"=>5, "bcd"=>3, "cde"=>2]
frequencySort.php:: getFrequency	["abc","abc","bcd","bcd","bcd","abc","cde","cde","abc","abc"];	["abc"=>5, "bcd"=>3, "cde"=>2]	["abc"=>5, "bcd"=>3, "cde"=>2]

Since we have no branches in getFrequency function we only test it with two cases of different orders and expected to get the same output.

Function	Input	Expected	Actual
frequencySort.php:: filterBadWords	["abc"]	["abc"]	["abc"]
frequencySort.php:: filterBadWords	["after", "abc", "bcd", "again", "a", "all"]	["abc", "bcd"];	["abc", "bcd"];

There is one branch judging if the word is trivial or not. Therefore, we should give one non-trivial input and one trivial input, expecting the program can get rid of the trivial word perfectly.

Function	Input	Expected	Actual
frequencySort.php:: getMap	["abc abc abc", "bcd bcd abc", " cde cde bcd", "def"]	[["word"=>"abc", "count"=>4], ["word"=>"bcd", "count"=>3], ["word"=>"cde", "count"=>2], ["word"=>"def", "count"=>1]]	[["word"=>"abc", "count"=>4], ["word"=>"bcd", "count"=>3], ["word"=>"cde", "count"=>2], ["word"=>"def", "count"=>1]]

frequencySort.php:: getMap	["again again", "almost already also", "always am among", "an"]	[]	empty

This function serves as a general execution of frequencySort, calling filterBadWords and getFrequency functions. Expect to get sorted map without trivial word. However, we failed in the second testcase because it returned a null rather than an empty array.

3.2.2 White-Box Test Suite 02: wordcount.php

Function	Input	Expected	Actual
wordcount.php:: getnubmer	"abc abc abc bcd bcd abc cde cde bcd def" & "abc"	4	4

This function takes in a string and a word in order to count the occurrence of that word in the string. No branching or loop need to do white box. So we only have one test case.

3.2.3 White-Box Test Suite 03: request.js

getArtists

Function	Input	Expected	Actual
request.js:: getArtist	"ColdPlay"	Integer of ColdPlay artist id	Integer of ColdPlay artist id
request.js:: getArtist	"aasdf"	Undefined Variable	Undefined Variable

In the case we divide the situation to two, based on branch: searching of the existed artists and searching on non-existed artists. The expected outputs were returned.

getAllAlbums

Function	Input	Expected	Actual
request.js:: getAllAlbums	"56083" (id for Amy Winehouse)	Larger than zero(get something in first place)	Larger than zero
request.js:: getAllAlbums	"00000" (id for no one)	0	0
request.js:: getAllAlbums	"56083" (id for Amy Winehouse)	Exactly 50 albums	Less than 50 albums were get

We test on one branch of whether the artist_ids is fake or not, though this can be asure if calling via getArtist. We passed the first two tests. However, we were not be able to get all the albums, it might be the reason of default page size by Musixmatch API.

getTracks

Function	Input	Expected	Actual
request.js:: getTracks	"13801029" (an album_id of Amy Winehouse)	11	11
request.js:: getTracks	"344265" (an album_id of no one)	0	0

We use two different artist_id to fit the different branch. There is only 11 songs in that album and 0 song exists for that artist_id never exist. We get the correct output.

getLyrics

Function	Input	Expected	Actual
request.js:: getLyrics	"30645728"(track_id with english lyrics)	A string variable	A string
request.js:: getLyrics	"30645743"(track_id with foreign lyrics)	''	empty

We use two different track_id to fit the different branch. There is one song with english lyrics and one song with foreign lyrics. In the branch we get rid of the foreign language. Therefore, we get an empty result rather than empty character back, probably the reason of forgetting to initialize variables.We get the correct output.

3.2.4 Report of Coverage

We originally plan to use phpunit-coverage to get the coverage report automatically. However, the versions of xdebug and phpunit are not compatible with each other. Therefore we demonstrate the coverage of testing code.

frequencySort.php:

Covered: all the main functions. 90%

Not covered: json get parameter from javascript 10%

Wordcount.php:

Covered: all the main functions. 95%

Not covered: json get parameter from javascript 5%

Request.js:

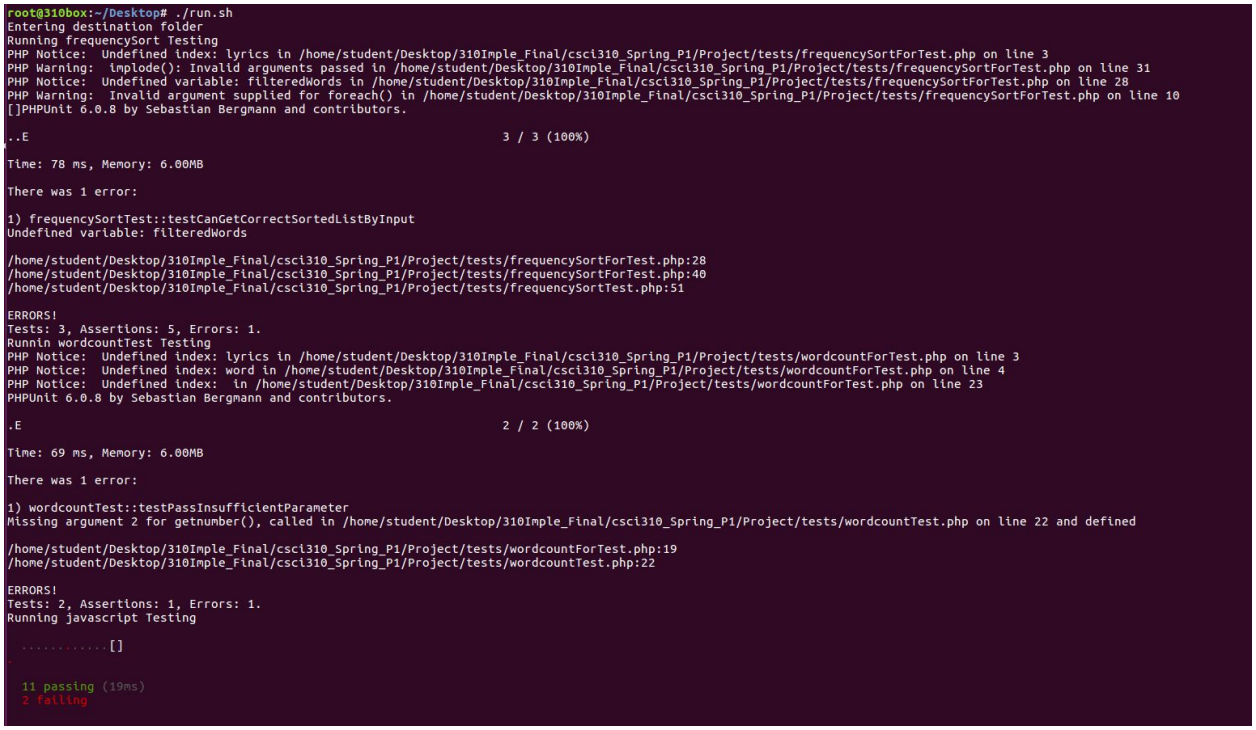
Covered: most of the main functions. 80%

Not covered: Implicit Autocomplete generation code 5%
Implicit word cloud generation code 5%
Store cache to LocalStorage 5%
Duplicate call function 5%

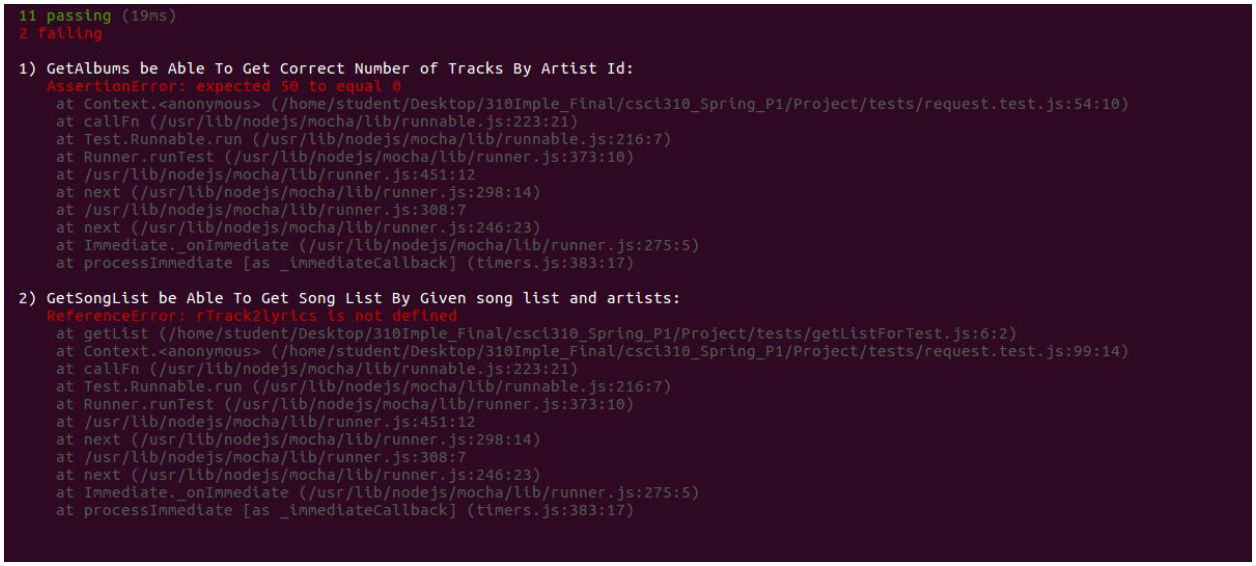
Songlist.js:

Covered: all the main functions 100%

3.2.5 Screenshots of White-Box Testing



```
root@310box:~/Desktop# ./run.sh
Entering destination folder
Running frequencySort Testing
PHP Notice:  Undefined index: lyrics in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php on line 3
PHP Warning:  implode(): Invalid arguments passed in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php on line 31
PHP Notice:  Undefined variable: filteredWords in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php on line 28
PHP Warning:  Invalid argument supplied for foreach() in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php on line 10
[]PHPUnit 6.0.8 by Sebastian Bergmann and contributors.

..E                                                                 3 / 3 (100%)

Time: 78 ms, Memory: 6.00MB

There was 1 error:

1) frequencySortTest::testCanGetCorrectSortedListByInput
Undefined variable: filteredWords

/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php:28
/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortForTest.php:40
/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/frequencySortTest.php:51

ERRORS!
Tests: 3, Assertions: 5, Errors: 1.
Runnin wordcountTest Testing
PHP Notice:  Undefined index: lyrics in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountForTest.php on line 3
PHP Notice:  Undefined index: word in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountForTest.php on line 4
PHP Notice:  Undefined index:  in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountForTest.php on line 23
PHPUnit 6.0.8 by Sebastian Bergmann and contributors.

.E                                                                  2 / 2 (100%)

Time: 69 ms, Memory: 6.00MB

There was 1 error:

1) wordcountTest::testPassInsufficientParameter
Missing argument 2 for getnumber(), called in /home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountTest.php on line 22 and defined

/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountForTest.php:19
/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/wordcountTest.php:22

ERRORS!
Tests: 2, Assertions: 1, Errors: 1.
Running javascript Testing

  ...........[]


  11 passing (19ms)
  2 failing
```



```
  11 passing (19ms)
  2 failing

  1) GetAlbums be Able To Get Correct Number of Tracks By Artist Id:
     AssertionError: expected 50 to equal 0
      at Context.<anonymous> (/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/request.test.js:54:10)
      at callFn (/usr/lib/nodejs/mocha/lib/runnable.js:223:21)
      at Test.Runnable.run (/usr/lib/nodejs/mocha/lib/runnable.js:216:7)
      at Runner.runTest (/usr/lib/nodejs/mocha/lib/runner.js:373:10)
      at /usr/lib/nodejs/mocha/lib/runner.js:451:12
      at next (/usr/lib/nodejs/mocha/lib/runner.js:298:14)
      at /usr/lib/nodejs/mocha/lib/runner.js:308:7
      at next (/usr/lib/nodejs/mocha/lib/runner.js:246:23)
      at Immediate._onImmediate (/usr/lib/nodejs/mocha/lib/runner.js:275:5)
      at processImmediate [as _immediateCallback] (timers.js:383:17)

  2) GetSongList be Able To Get Song List By Given song list and artists:
     ReferenceError: rTrack2Lyrics is not defined
      at getList (/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/getListForTest.js:6:2)
      at Context.<anonymous> (/home/student/Desktop/310Imple_Final/csci310_Spring_P1/Project/tests/request.test.js:99:14)
      at callFn (/usr/lib/nodejs/mocha/lib/runnable.js:223:21)
      at Test.Runnable.run (/usr/lib/nodejs/mocha/lib/runnable.js:216:7)
      at Runner.runTest (/usr/lib/nodejs/mocha/lib/runner.js:373:10)
      at /usr/lib/nodejs/mocha/lib/runner.js:451:12
      at next (/usr/lib/nodejs/mocha/lib/runner.js:298:14)
      at /usr/lib/nodejs/mocha/lib/runner.js:308:7
      at next (/usr/lib/nodejs/mocha/lib/runner.js:246:23)
      at Immediate._onImmediate (/usr/lib/nodejs/mocha/lib/runner.js:275:5)
      at processImmediate [as _immediateCallback] (timers.js:383:17)
```

4. Description of Testing Process

4.1 Compliance with Project Management Plan

This section provides the process details of the testing phase of this project, and examines how the testing process complies with the *Project Management Plan.*

4.1.1 Process and Organization

According to the original planning (see Section 2.2 and Section 2.3 of *Project Management Plan*), this group is divided into three 2-person subteams, in order to divide and conquer. However, in order to better adapt to the testing process, this group rearranges the composition of each subteam. Please proceed to Subsection 4.2.1 (Rearrangement of Subteams) of this document for details.

4.1.2 Schedules, Milestones, and Deliverables

According to the original planning (see Subsection 4.1.1 and Subsection 4.1.2 of *Project Management Plan*), tasks T28 through T35 are scheduled for the testing phase of this project, and milestone M6 (complete testing) is due on Tuesday, March 7th, 2017. However, since both the implementation and testing deadlines were extended, this group postpones the schedule and milestone for the testing phase of this project. Please proceed to Subsection 4.2.2 (Postponed Testing Schedule) of this document for details.

4.1.3 Quality Assurance

According to the original planning (see Subsection 6.1.5 of *Project Management Plan*), this group is required to submit one *Testing Report* document for the testing phase of this project. This document is the one submitted for the testing phase. This document specifies all the test cases, using Cucumber for black-box testing, and using PHPUnit for white-box testing; this document is drafted professionally, with all the chapters, sections, and subsections properly and systematically numbered, and composed by elements such as an executive summary and a table of content, etc.; this document lists test cases for all the functional requirements (see Section 3.2 of *Software Requirements Specification* provided by Team 17), and all the classes. Therefore, this group is compliant on the terms of quality assurance during the testing phase of this project.

4.1.4 Configuration Management

According to the original planning (see Section 7.2 of *Project Management Plan*), this group is supposed to use BitBucket for version control. However, due to developers' inadequate experience with BitBucket, this group decided to permanently replace BitBucket with GitHub during the implementation phase of this project (see Subsection 3.2.3 of *Project Management*

Plan). This group still uses GitHub for version control. Therefore, this group is compliant on the terms of configuration management during the testing phase of this project.

4.1.5 Risk Management

According to the original planning (see Section 9.2 of *Project Management Plan*), this group is required to monitor all the pre-identified and analyzed risks. During the testing process, this group pay close attention to all the risks, but none of them escalates its probabilities. Therefore, this group is compliant on the terms of risk management during the testing phase of this project.

4.2 Deviation from Project Management Plan

This section provides the details of the actual testing process, which deviate from the Project Management Plan. Each instance of deviation is examined via (1) the origin plan of the intended task, (2) the actual execution of the intended task, and (3) the reasoning behind such a change of plan.

4.2.1 Rearrangement of Subteams

Because of the unforeseen complexity of the testing process, this group decides to rearrange the organizational structure. In general, all three subteams will be responsible for their assigned task of black-box testing, including test case design and execution. In addition, Ernest breaks out from Subteam 1, in order to devote most of his workload on drafting the *Testing Report* document, and Leo focuses on the design and execution of black-box testing tasks assigned to Subteam 1; Lifan breaks out from Subteam 2, in order to devote most of his workload on the design and execution of white-box testing, and Mengqian focuses on the design and execution of black-box testing tasks assigned to Subteam 2.

4.2.2 Postponed Testing Schedule

Because the testing deadline was extended to Thursday, March 9th, 2017, and the testing phase couldn't be started until the completion of the implementation phase, this group adjusts the testing schedule accordingly (see Subsection 4.1.2 of *Project Management Plan*).

This group has the first testing meeting on Sunday night, March 5th, 2017 (designated Testing Meeting #1). Before the Testing Meeting #1, each subteam completes their assigned test case design task: Subteam 1 completes task T28 (primarily by Leo); Subteam 2 completes task T29 (primarily by Mengqian); Subteam 3 completes task T30. During the Testing Meeting #1, this group collectively completes task T31.

This group has the second testing meeting on Wednesday night, March 8th, 2017 (designated Testing Meeting #2). Before the Testing Meeting #2, each subteam completes their assigned test case running tasks: Subteam 1 completes task T32 (primarily by Leo); Subteam 2

completes task T33 (primarily by Mengqian); Subteam 3 completes task T34. During the Testing Meeting #2, this group collectively completes task T35.

Milestone M6 is reached, when this group delivers the working system, all the test cases (including both black-box testing and white-box testing), and the *Testing Report* document on Thursday, March 9th, 2017..