Project #1
Implementation 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 implementation report for Lyrics Cloud, based on the *Software Requirements Specification* document provided by Team 17 of CSCI 310 (Software Engineering). This document (1) validates this group's actually implemented system against this group's original design (a.k.a. the *Software Design* document submitted by this group), and (2) describes this group's implementation process and cross-references with this group's planning (a.k.a. the *Project Management Plan* document submitted by this group). These two activities explained by this document are arranged in two chapters: Chapter 2 (Validation of Implementation), and Chapter 3 (Description of Implementation Process).

Chapter 2 validates the actual implementation again this group's original design (a.k.a. the *Software Design* document submitted by this group). Most part of the actually implemented system complies with the original design. This group examines the correct compliance using the 7 types of diagrams drafted in the original design: (1) the use case diagram, (2) the data flow diagram, (3) the state machine diagram, (4) the sequence diagram, (5) communication diagram, (6) component diagram, and (7) the class diagram. There also exist 3 major deviations from the original design: (1) replacement of Spotify API with Musixmatch API, (2) adoption of existing API for word cloud generation, and (3) functionality of search suggestion.

Chapter 3 describes this group's implementation, and ensures that this group is following a predetermined process instead of mindlessly hacking. Most of this group's implementation process is in good compliance with the original planning (a.k.a. the *Project Management Plan* document submitted by this group). Due to time constraint, there is an extension on the implementation deadline. In consequence, 3 major activities deviates from the original planning: (1) selection of Lifan as the decision-making leader during this phase, (2) rescheduled implementation milestones due to postponed deadline, and (3) replacement of BitBucket with GitHub for version control. Alos, in terms of risk management, the probability of 2 pre-identified and analyzed risks escalated during the implementation process: (1) potential course withdrawal by Mengqian, and (2) Juntao and Leo's limited availability. However, this group manages to resolves and de-escalates these risks eventually, so that they do not pose catastrophic effect on the implementation process.

Chapter 4 is the startup (running) instruction for this group's actually implemented system. This is a detailed list of step-by-step instructions. Following the instructions, the client will be able to run this system as the STAKEHOLDERS request.

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 validates the actual implemented system with the Software Design Document that this group submitted on February 16th, 2017, describes the implementation process that this group adopted, and provides a detailed instruction of how to run system.

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.

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: February 24th, 2017.

[2] This Group. “Project #1 Project Management Plan”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHZ3BycjlSS00zeDQ. Accessed: February 24th, 2017.

[3] This Group. “Project #1 Software Design”. [Online]. Available: https://drive.google.com/open?id=0B4dQvo2bfyPHdThBWFE0OEt5YUU. Accessed: February 24th, 2017.

[4] Spotify AB. “Spotify Developer”. [Online]. Available: https://developer.spotify.com/. Accessed: February 24th, 2017.

[5] Musixmatch. “Musixmatch Developer API”. [Online]. Available: https://developer.musixmatch.com/. Accessed: February 24th, 2017.

[6] Facebook. “Facebook for Developers”. [Online]. Available: https://developers.facebook.com/. Accessed: February 24th, 2017.

2. Validation of Implementation

2.1 Compliance with Software Design

This section provides the details of all the implementations that comply with the original design (a.k.a. the *Software Design* document). Specifically, this section validate the actually implemented system against each diagram drafted in the original design.

2.1.1 Use Case Diagram

The use case diagram in the original design (see Section 2.1 of *Software Design*) explains the general idea behind each user's action. To summarize, the general process for each use action is that when client's action stimulates an request through the web portal, the Lyrics Cloud system will return a result and display it on the web portal (request-return cycle).

The actual implementation of this system is written in HTML, PHP and JavaScript. This JavaScript handles the request-return cycle (described in the last paragraph). The web portal is written is HTML. There are some blocks of code, which represent the request-return cycle. These blocks of code are written in JavaScript, and will not be triggered unless client explicitly acted on the components, which these blocks of code relate to, on the web portal of this system.

2.1.2 Data Flow Diagram

The data flow diagram in the original design (see Section 2.2 of *Software Design*) explains how data flows through this system, and between this system and APIs. Each data flow cycle corresponds to one request-return cycle (described in Subsection 2.1.1 of this document). Since JavaScript handles each cycle, and these JavaScript blocks of code won't be triggered unless client acts, the actual implementation complies well with the original design in terms of data flow.

The only deviation from the original data flow diagram was that this group decided to replace Spotify API with Musixmatch API, because of Spotify API's inadequacy of developer access of lyrics database. Please proceed to Subsection 2.1.6 (Component Diagram) and Subsection 2.2.1 (Musixmatch API) of this document for details.

2.1.3 State Machine Diagram

The state machine diagram of the original design (see Section 2.3 of *Software Design*) explains the different states of this system. In the actual implementation, each action by client will trigger a corresponding block of JavaScript code. After one specific block of JavaScript code is executed, the result will be displayed on the web page, which represents the corresponding state.

2.1.4 Sequence Diagram

The sequence diagram of the original design (see Section 2.4 of *Software Design*) is a further expansion of the use case diagram (see Section 2.1 of *Software Design*, and Subsection 2.1.1 of this document), explaining the sequence in which each action is able to be executed by client. In the actual implementation, the sequence of action is arranged in the following order:

(1) Client navigates to the homepage of the web portal of this system.
(2) Client enters the name of the artist he wants to search for.
(3) This system suggests search keywords.
(4) Client clicks the Search Button after entering a valid artist's name.
(5) This system displays word cloud for the lyrics of the specified artists.
(6) Client can either add an artist into the word cloud (by entering a new name, and then clicking Add Button), or share the word cloud to Facebook, or clicks a word on the word cloud.
(7) If client clicks a word on the word cloud, this system searches for the songs that contain this keyword, and displays the returned list of songs with frequency of this keyword in each song on the next page
(8) If client clicks a song, this system displays the lyrics of this song with the previously chosen keyword highlighted on the next page.

This sequence of action complies with the sequence diagram in the original design.

2.1.5 Communication Diagram

The seven communication diagrams in the original design (see Section 2.5 of *Software Design*) visualize the exchange of stimulus and result display between the Client and the Web Portal of this system, and the communication of request and return message between the Web Portal of this system and API Servers.

2.1.6 Component Diagram

There are two component diagrams in the original design (see Section 2.6 of *Software Design*). The first component diagram involves Spotify API (see Subsection 2.6.1 of *Software Design*, designated Component Diagram #1 in this document); the second component diagram involves Facebook API (see Subsection 2.6.2 of *Software Design*, designated Component Diagram #2 in this document).

Component Diagram #1 was re-designed in the implementation phase. The new design of Component Diagram #1 replaces Spotify API with Musixmatch API. For details and justification of such a deviation, please proceed to Subsection 2.2.1 (Musixmatch API) of this document.

Component Diagram #2 is implemented in the way of its original design. Whenever the client clicks the Share Button, this system sends a share request to Facebook API Server, which will return a true/false result of whether such an action succeeds.

2.1.7 Class Diagram

The class diagrams in the original design (see Section 2.7 of *Software Design*) focuses mainly on the WebPage class, the Button class, and their respective derived classes. In the actual implementation, this group wrote one HTML file for each derived WebPage class in the original design (see Subsection 2.7.1 of *Software Design*). Each block of JavaScript code maps to the functionality of one derived Button class in the original design (see Subsection 2.7.2 of *Software Design*). Therefore, the actually implemented system is in good compliance with the original design, in terms of the class diagrams.

2.2 Deviation from Software Design

This section provides the details of any implemented features that deviate from the original design (a.k.a. the *Software Design* document). Each instance of deviation will be justified by (1) description of the original design, (2) description of the actual implementation, and (3) why this group chose this change.

2.2.1 Musixmatch API

The original design (see Subsection 2.6.1 of *Software Design*) of artist search, keyword search, track search, and lyrics fetch involves Spotify API. However, after starting the implementation phase, Subteam 1 discovered that the methods provided by Spotify API do not grant developer access to Spotify API's lyrics database. The most Spotify API can do is to return a boolean value (true/false) whether one track has explicit lyrics. Obviously such a method is inadequate for the implementation of this system.

For the actual implementation of this system, this group opted for Musixmatch API, which grants paid access to its lyrics database. However, this group use the free version of Musixmatch API which merely grants 30% of the lyrics. There exist three methods in Musixmatch API, which justify this deviation from the original design (see Subsection 2.6.1 of *Software Design*). Musixmatch API allows search for tracks using a specified artist's name as the search parameter, so that the returned track list contains each track's lyrics, which fulfills the functional requirement of the Search Button (see Subsection 3.2.2 of *Software Requirements Specification* drafted by Team 17); Musixmatch API allows search for tracks using a specified artist's name, and a specified keyword as the search parameters, which fulfills the function requirement of clicking a word in the word cloud (see REQ-7, Subsection 3.2.4.3 of *Software Requirements Specification* drafted by Team 17); Musixmatch API allows search for lyrics using a specified track's track_id, designated by Musixmatch API, as the search parameters, which fulfills the functional requirement of clicking on the song (see REQ-4, Subsection 3.2.8.3 of *Software Requirements Specification* drafted by Team 17).

However, there still exists one problem with the Musixmatch API. The nationality of any artist is undefined in search result, because Musixmatch API doesn't grant developer access to any artist's nationality

2.2.2 API for Word Cloud Generation

The original design (a.k.a. the *Software Design* document) did not foresee the complication in actual implementation. However, after starting the implementation phase, this group found that it is extremely complicated to implement our own world cloud generation. Therefore, Subteam 2, who are tasked to implement our own algorithm for word cloud generation, did research on APIs, and found this API for word cloud generation, wordcloud2.js, designed and implemented by Tim Chien. This API basically takes two major input parameters: a string of text, and a style parameter. For the purpose of this project, what this group did for the actual implementation of this system was to consolidate all the lyrics of the specified artist(s), pass them into the wordcloud2.js API, and specify the style parameter.

2.2.3 Functionality of Search Suggestion

This dropdown list are filled with suggested artists' names, which are produced by Musixmatch autocomplete API. When the user types characters in the search box, the API call instantly depend on user input and generate a suggestion list. Moreover, if the user click on any artist on the suggestion list, it automatically fills the search box with the selected name. This group thinks it is very user-friendly and can improve the user experience.

3. Description of Implementation Process

3.1 Compliance with Project Management Plan

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

3.1.1 Process and Organization

During the implementation phase, this group still remains the 6-person size, and maintains the flat organizational structure, composed of three 2-person subteams (see Section 2.2 of *Project Management Plan*). The 2 developers from the same Subteam worked closely on their assigned tasks. The pair programming mechanism inside each subteam worked quite well as this project progressed, because when the difficulties arose, the 2 developers from the same Subteam could resolve the problems collectively. For example, when Ernest (from Subteam 1) found out that Spotify API couldn't grant developer access to lyrics, Subteam 1 decided to replace Spotify API with Musixmatch API (see Subsection 2.2.1 of this document), and coordinated research on Musixmatch API.

However, on the matter of organizational management, there were some deviation from this group's planning (see Section 2.3 of *Project Management Plan*). Please proceed to Subsection 3.2.1 (Changes on Organizational Management) of this document for details.

3.1.2 Schedules, Milestones, and Deliverables

Because the midterm for CSCI 310 (Software Engineering) was on Wednesday, February 22nd, 2017, and the Virtual Machine for the implementation phase was not available until Thursday, February 23nd, 2017, One major STAKEHOLDER, Professor William G. J. Halfond decided to postpone the implementation deadline from Monday, February 28th, 2017 to Thursday, March 2nd, 2017. These unforeseen situations left this group in a position of poor compliance with the original scheduling (see Chapter 4 of *Project Management Plan*). Please proceed to Subsection 3.2.2 (Postponed Implementation Schedule) of this document for details.

3.1.3 Quality Assurance

This group complied well with our Software Quality Assurance Plan (see Chapter 6 of *Project Management Plan*) in this implementation phase. According to our Software Quality Assurance Plan, this implementation phase would deliver two artifacts by Monday, February 27th, 2017 (now postponed to Thursday, March 2nd, 2017): the *Implementation Report* document, and an implemented working system. This group submitted these two deliverables on time.

The *Implementation Report* document (see Subsection 6.1.4 of *Project Management Plan*) was drafted professionally. This document contains a Executive Summary, and a Table of Content; all the chapters, sections and subsections of this document are highly formatted, and labeled in numerical bulletins. This document validates the actual implementation against the original design (a.k.a the *Software Design* document), to ensure that this group was building the right system. This document examines how the actual implementation complies with the original planning (a.k.a. the *Project Management Plan* document), to ensure that this group was following a process, instead of mindlessly hacking.

3.1.4 Configuration Management

According to the original planning (see Section 7.2 of *Project Management Plan*), this group would control different version of this system by using BitBucket, as this project progresses within the implementation phase. In reality, this group opted for Github. Please proceed to Section 3.2.3 (Use of GitHub for Version Control) for details.

3.1.5 Risk Management

This group was monitoring all the pre-identified and analyzed risks (see Section 9.2 of *Project Management Plan*) through the entire implementation phase. During this implementation phase, the probability of two major risks escalated, so that they raised this group's alarm: (1) course withdrawal, and (2) developers' availability.

Shortly after the midterm, Mengqian became a candidate for course withdrawal. She was afraid that she had done poorly on her midterm, so that she might fail the course. Responding to her anxiety, this group rallied for a short meeting in the afternoon of the day of the midterm, so that we could calm her down, and keep her motivated to stay with this group. This attempt de-escalated the risk of Mengqian's withdrawal from this course.

Before the midterm, Juntao and Leo notified this group that they had another big project due on Monday, February 27th, 2017 (the original deadline for Project #1 Implementation), so that they were afraid they would have very limited availability for both projects. This group informed the STAKEHOLDERS to negotiate a postpone of the implementation deadline. As a result, the major STAKEHOLDER, Professor William G. J. Halfond, decided to extend the deadline to Thursday, March 2nd, 2017. Therefore, Juntao and Leo could focus more on their own independent projects first, and rejoin this group later will full availability. This attempt of negotiation to postpone the implementation deadline lead to an extension, which resolved the risk of Juntao and Leo's availability due to collision of deadlines.

3.2 Deviation from Project Management Plan

This section provides the details of the actual implementation 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.

3.2.1 Changes on Organizational Management

Due to the nature of flat organizational structure, there exists an person with absolute authority in terms of decision making. During the implementation phase, we found it necessary to define a group leader (of decision making). Because Lifan was in charge of communicating with STAKEHOLDERS before, and best understood the requirements, planning, and design among all 6 developers, he was elected the decision-making leader of this group for the implementation phase. He managed the overall progress of this group, and ensured that every developer was on the same page, and that this project would be delivered on time.

3.2.2 Postponed Implementation Schedule

Due the unforeseen situations described in Subsection 3.1.2 (Schedules, Milestones, and Deliverables) of this document, this group had to postponed the implementation schedule. This group tried our best to catch up the original schedule as much as possible.

During the implementation phase, this group first rallied on Saturday afternoon, February 25th, 2017 (designated Implementation Meeting #1). The major activities for this first implementation meeting were to ensure that each Subteam was fully aware of their assigned tasks and the urgency of this phase, and to address any question that any developers had. This group had 3 more implementation meetings, on Sunday evening, February 26th, 2017 (designated Implementation Meeting #2), on Tuesday evening, February 28th, 2017 (designated Implementation Meeting #3), and on Wednesday evening, March 1st, 2017 (designated Implementation Meeting #4).

In terms of schedule milestones (see Section 4.2 of *Project Management Plan*), this group reached Milestone M3 (Basic Function Complete) by the end of Implementation Meeting #2, reached Milestone M4 (Advanced Function Complete) by the end of Implementation Meeting # 3, and reached Milestone M5 (Complete Implementation) by the end of Implementation Meeting #4.

3.2.3 Use of GitHub for Version Control

After this group conducted a survey on developers' respective familiarity on BitBucket and GitHub, we discovered that not all of the developers of this group had adequate experience with BitBucket. On the contrary, since all the developers from this group had taken CSCI 104 and CSCI 201, which granted developers sufficient experience of working with GitHub, this group decided to switch from BitBucket to GitHub. By using GitHub for version control, this group would save the trouble of push-and-pull collision due to inadequate familiarity with BitBucket.

4. Startup Instructions

Step 1	In Virtualbox, go to “File -> Import Appliance” in the menu bar.
Step 2	Click the “Open Appliance” button to select your OVA file.
Step 3	Once you have selected the file, click “Next”.
Step 4	Scroll through the configuration list and click “import”
Step 5	Select the new VM in your list and click green arrow “show”.