Project #1
Project Management Plan

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

Executive Summary

This document lays out the details on planning and management of this project, based on the Software Requirements Specification Document that this group submitted on January 30th, 2017. Components of this document include 1 introductory chapter, 8 content-intense chapters, and 1 Appendix.

To get started, this group has a small staff size (6 persons), maintaining a flat organizational structure, which is constituted by three 2-person subteams. This unique organizational design features pair programming, shared responsibility, and specialized job for certain talented persons.

This group uses the COCOMO II model standard provided by USC CSSE to estimate the cost and effort necessary to complete this project. According to this group's estimation, one software engineer working for one month (Person-month) costs $3000. This project expects to be completed for a total cost of approximately $86393, for an effort of 28.8 Person-months.

Moving forward, this group sets 5 milestones for project scheduling: basic design, complete design, basic function complete, advanced function complete, and complete implementation. During the waterfall lifecycle of this project, this group has generated or will generate the following artifacts: Software Requirements Specification (SRS), Project Management Plan (PMP), Design Document, Implementation Document, Testing and Delivery Document, Code, Development Notes, Testing Code, and Webpage.

All 8 contents-intense chapters are concluded by this group's risk management plans, which identify and analyze potential risks ahead, and draft plans to counter and monitor these risks. All identified risks include course withdrawal, changes to requirements, underestimated time consumption, members' availability, subteams rearrangement, inadequacy in required programming language, integration problem, fulfill performance requirements, and time complexity of large input dataset.

In the end, the one appendix explains in details about the standard for comparison of qualitative measurement of factors of cost, using the COCOMO II model provided by USC CSSE.

For those who may concern, please continue to main contents for more detailed explanation about the planning and management of this project.

Table of Contents

1. Introduction

1.1 Project Overview

Upon the STAKEHOLDERS' request, this project is designed to generate word clouds for lyrics of specified artists. This document specifically walks through every detail about the planning of this project.

1.2 Project Deliverables

The deliverables of this project include Software Requirements Specification (already compiled and submitted to STAKEHOLDERS), Project Management Plan (this document), architecture and design document, implementation document, and testing and delivery document.

1.3 Evolution of the PMP

This document is the very first scheduled version of Project #1's Project Management Plan.

1.4 References

[1] "COCOMO® II cost driver and scale driver help,". [Online]. Available: http://sunset.usc.edu/research/COCOMOII/expert_cocomo/drivers.html. Accessed: Feb. 7, 2017.

[2] Posted and M. Rouse, "What is capability maturity model (CMM)? - definition from WhatIs.com," SearchSoftwareQuality, 2016. [Online]. Available: http://searchsoftwarequality.techtarget.com/definition/Capability-Maturity-Model. Accessed: Feb. 7, 2017.

[3]"Frequently asked questions regarding the usage of person months,". [Online]. Available: https://grants.nih.gov/grants/policy/person_months_faqs.htm#1039. Accessed: Feb. 7, 2017.

[4] Posted and M. Rouse, "What is waterfall model? - definition from WhatIs.com," SearchSoftwareQuality, 2007. [Online]. Available: http://searchsoftwarequality.techtarget.com/definition/waterfall-model. Accessed: Feb. 7, 2017.

1.5 Definitions and Acronyms

This Group: Group #6 of CSCI-310 (Software Engineering) Project #1 for the semester Spring 2017, including Jiachang (Ernest) Xu, Xiaoyang (Leo) Liao, Lifan Zhao, Mengqiao (Michielle) Yu, Juntao (Kenneth) Shen, Yawen (Maggie) Cao.

Subteam: a 2-person organization 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.

COCOMO: short for "the COnstructive COst MOdel", and it can be used as a framework for cost estimation and related activities

Person Months: is the metric for expressing the effort (amount of time) PI(s), faculty and other senior personnel devote to a specific project. The effort is based on the type of appointment of the individual with the organization;

CSSE: short for Center for System and Software Engineering.

USC: an Acronym for University of Southern California.

PREC: stands for "Precedentedness", a scale factor that measures the degree to which a system is new and past experience applies.

FLEX: stands for "Development Flexibility", a scale factor that scales the need to conform with specified requirements.

RESL: stands for "Architecture/Risk Resolution", a scale factor that measures degree of design thoroughness and risk elimination.

TEAM: stands for "Team Cohesion", a scale factor that scales the need to synchronize stakeholders and minimize conflict.

PMAT: stands for "Process Maturity", Software Engineering Institute's Capability Maturity Model process maturity rating.

RELY: stands for "Required Software Reliability", a product factor that measures the extent to which the software must perform its intended function over a period of time.

DATA: stands for "Database Size", a product factor that attempts to capture the affect large data requirements have on product development.

CLPX: stands for "Product Complexity", a product factor. Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.

RUSE: stands for "Required Reusability", a product factor that accounts for the additional effort needed to construct components intended for reuse on the current or future projects.

DOCU: stands for "Documentation match to life-cycle needs", the rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs.

TIME: stands for "Execution Time Constraint", a platform factor of the execution time constraint imposed upon a software system.

STOR: stands for "Main Storage Constraint", a platform factor that represents the degree of main storage constraint imposed on a software system or subsystem.

PVOL: stands for “Platform Volatility”, a platform factor that means the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.

ACAP: stands for “Analyst Capability”. Analysts are personnel that work on requirements, high level design and detailed design.

PCAP: stands for “Programmer Capability” Current trends continue to emphasize the importance of highly capable analysts.

AEXP: stands for “Applications Experience”, is dependent on the level of applications experience of the project team developing the software system or subsystem.

PEXP: stands for “Platform Experience” The Post-Architecture model broadens the productivity influence of PEXP, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.

LTEX: stands for “Language and Tool Experience”, is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem.

PCON: stands for “Personnel Continuity”, a personnel factor in terms of the project's annual personnel turnover.

TOOL: stands for “Use of Software Tools”, software tools that have improved significantly since the 1970's projects.

SITE: stands for “Multisite Development”. Given the increasing frequency of multisite developments, and indications that multisite development effects are significant.

SCED: stands for “Required Development Schedule”, a project factor that measures the schedule constraint imposed on the project team developing the software.

2. Process and Organization

2.1 Software Lifecycle Process

This group uses Waterfall methodology for software lifecycle process. Waterfall methodology is a linear approach where events are scheduled in a sequence. First off, it is system and software requirements stage while gathering product requirement document. Secondly, it comes to analysis stage while resulting in models, schema, and business rules. Then, the next stage is design, which results in software architecture. Afterwards, it is a stage of coding where software development, proving and integration of software are carried on. Then there is testing stage which intends to debug defects and fix problems. Finally, it is a stage for installation, migration, and maintenance of completed system.

2.2 Organizational Structure

Due to the small size of this group (6 people in total), this group chooses a flat organizational structure. Every person in this group is equally superior in the matter of organizational chart. However, the group decides to break down to three 2-person subteams: Ernest and Leo form Subteam 1; Lifan and Mengqian form Subteam 2; Juntao and Yawen form Subteam 3. Two members of the same subteam work collectively on one same task each time.

2.3 Organizational Management

This group borrows the idea of pair programming to design the organizational structure mentioned above. When the two members from the same subteam collaborate on one task, they supervise each other, and pay close attention in case any mistakes arise. On the level of inter-subteam collaboration, all three subteams supervise each other to complete their assigned tasks on time.

In order to enforce organizational integrity, and ensure clear communication, there are specific persons to perform specific kinds of jobs. Ernest is in charge of scheduling group meetings, reserving rooms for meetings, and writing executive summaries; Lifan is in charge of communication with STAKEHOLDERS, and verbal compilation. As the project progresses, there may be more specific jobs assigned. In case that any of these specific persons were not able to perform his corresponding jobs, his subteam partner would take over until his return.

2.4 Organizational Process

The idea of organizational process is very similar to Divide and Conquer Algorithm. When a massive assignment is assigned to the group or when the group wants to fulfill one specific purpose, a massive assignment will first break into parts and subteams accomplish small goals. Inside each subteam, two team members collaborate together to achieve the goal. Among

different subteams, partial products are integrated as a whole which serves for the group. The concept is that complex problems are divided up based upon different features that can be resolved by each individual. Whereas a problem appears to be too complicated for single team member, it is very convenient to gain assistance from the same subteam.

3. Cost Estimation

3.1 Purpose

This chapter identifies and evaluates each factor of cost, and any cause that could affects productivity. By using USC CSSE's COCOMO II model, Section 3.2 assigns rating to each factor of cost. Additionally, Section 3.3 measures factors of productivity, and explains the reasoning behind each qualitative measurements. Section 3.5 finally calculate the cost equivalence of one Person-month, and estimates the required efforts in the measure of Person-months.

3.2 Factors of Cost

This section discusses the factors that affects this group's estimation on cost of building this system, from a managerial point of view, using USC CSSE's COCOMO II model. For information about standard of comparison, please refer to Appendix I.

3.2.1 Scale Factors

Factors	Rating	Reasoning
Precedentedness (PREC)	Nominal	This group have fair understanding of product objectives and some members have related experience.
Development Flexibility (FLEX)	Very Low	Given the fact that this project follows a waterfall process, it is too rigid to have room flexible to changes to requirements.
Architecture/Risk Resolution (RESL)	Nominal	Risk Identification (9.1) thoroughly identifies most risks. However, the development schedule does not assign adequate amount of time for establishing architecture.
Team Cohesion (TEAM)	High	Cooperation among

		members of this group is encouraged during the whole process, but it still has slightly more room for more effective communication
Process Maturity (PMAT)	Nominal	PMAT is determined through evaluation based on Capability Maturity Model. It is at the *defined* level, which means this group along with STAKEHOLDERs have developed a standard software process through greater attention to documentation, standardization, and integration.

3.2.2 Product Factors

Factors	Rating	Reasoning
Reliability (RELY)	Very Low	If this project failed, there would only very little losses.
Data (DATA)	Nominal	This system doesn't possess its own database. This system will retrieve data from Spotify and Facebook API, which are reliable applications.
Complexity (CPLX)	Nominal	This system requires simple nesting, use of math and statistical algorithm, simple error checking, and simple use of widget set.

Reusability (RUSE)	Nominal	Since the Spotify API and Facebook API that needs to be integrated to this system are designed to obey the Principle of Modularity, if any changes were proposed, this system would only need to be modified across project.
Documentation (DOCU)	Nominal	This project has or will have Software Requirements Specification, Project Management Plan, Architecture and Design Document, Implementation Document, and Delivery and Testing Document. These documents are right-sized for this project, which follows a waterfall model.

3.2.3 Platform Factors

Factors	Rating	Reasoning
Time Constraint (TIME)	Extra High	This system has a very strict performance requirement on execution time, which requires 1-second response time on generating word clouds.
Storage Constraint (STOR)	Nominal	This system only has a web portal. Therefore, it only poses an average needs for storage.
Platform Volatility (PVOL)	Nominal	This system requires

		integration with Spotify API and Facebook API, which are reliable applications that are normally volatile.

3.2.4 Personnel Factors

Factors	Rating	Reasoning
Analyst Capability (ACAP)	Nominal	Because all members of this group have taken a number of algorithm courses from USC, on average this group has an average-level capability on analysis.
Program Capability (PCAP)	Nominal	Because all members of this group have taken a number of programming courses from USC, on average this group has an average-level capability on programming.
Application Experience (APEX)	Low	Given the fact that all members of this group have all completed the course CSCI-201, all the members have equivalently 6-month similiar experience.
Platform Experience (PLEX)	Low	Given the fact that all members of this group have all completed the course CSCI-201, all the members have equivalently 6-month similiar experience.
Language and Tool Experience (LTEX)	Very Low	All members of this group have very little

		experience with the required programming languages (i.e. PHP and JavaScript).
Personnel Continuity (PCON)	Very Low	All members of this group stay focused on the project for the entire length of an semester (i.e. 4 months).

3.2.5 Project Factors

Factors	Rating	Reasoning
Software Tools (TOOL)	Very High	Spotify API and Facebook API are highly reliable applications.
Multisite Development (SITE)	Very High	All members of this group all rally at a reserved study room in Leavey Library during every group meeting. This team uses email, Google Drive, projectors to facilitate the organizational process
Required Schedule (SCED)	Nominal	This group believes that this project could be completed on time at the right pace.

3.3 Factors of Productivity

This section discusses the factors that facilitate or slow down this group's productivity, from a managerial point of view.

3.3.1 Consistency: Moderate

Given the peer supervision mechanism of this group, members supervise each other, so that personal preference toward certain people in a hierarchy is avoided.

3.3.2 Respect: Moderate

All group members had known each other prior to the start of this project. Every group member's personality is well known. Therefore, each person knows how to interact with the rest of the group, without violating a certain person's political position, religious belief, etc.

3.3.3 Group Structure: Very High

As discussed in Section 2.2 (Organizational Structure), this group is constructed based on a flat organizational structure, composed of three 2-person subteams. Such a group structure possesses the advantages of divide and conquer, pair programming, and peer supervision. Besides, there are certain group members, who serve to complete certain special tasks, in order to facilitate this project's progress. For details, please refer back to Section 2.3 (Organizational Management).

3.3.4 Group Composition: High

The way of dividing this group into subteams is very reasonable since every two members in a subteam has complementary personality and skill set. One can offer the greatest help to the other among all members in the group. More importantly, this group has a very balanced sex ratio that plays crucial role in interaction control and facilitating communication.

3.3.5 Physical Work Environment: High

Objectively, USC provides many study places which offer great study experience during group discussion and individual study. Study rooms require reservations so that no one disturbs and everyone can stay focused. There are even more study places options for subteams and individuals since one or two people don't require a large-capacity room

3.3.6 Available Communication Channel: Moderate

This group has many different forms of communications between group members, such as face to face, email, Wechat messages and more. Everyone is fast at replying questions and eager to contribute through group chat channel by expressing unique idea. Most importantly, members are easily accessible for the most time of a day.

3.4 Relation between Cost and Schedule

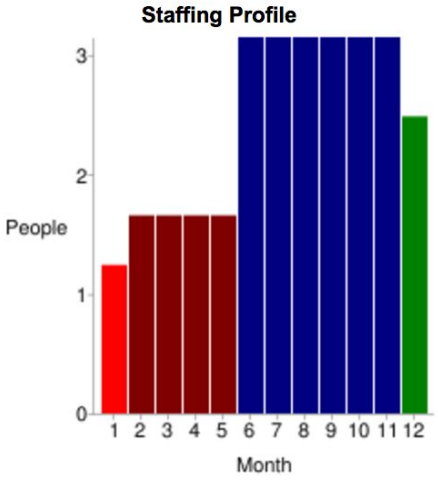
The Schedule is a crucial factor that influence cost directly and that's why having a consistent schedule is a good approach to limit cost. As the schedule prolongs and procrastinates

the date for completed system, the cost will increase due to the fact that more hours of development need more compensations. Nevertheless, under the condition that the quality of the product remains unchanged, a more compact schedule will more likely reduce the cost by shortening the total hours of development.

3.5 Person-Month (PM)

With a careful observation of similar projects, it is safe to determine that this system will have approximately 5000 source lines of codes, and 1000 of those will be modified. As for Person Month (PM), including salary, benefits, IT support, utilities and office space, the estimated cost for having one software engineer (group member) work a month would be $3000 .

According to this group's estimation from Section 3.2 (Factors of Cost), using USC CSSE's COCOMO II model. this group calculates that the estimated effort for this project is 28.8 Person-months, and the length of schedule will be 11.1 months. Total cost for this project will be $86393.



Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.7	1.4	1.2	$5184
Elaboration	6.9	4.2	1.7	$20734
Construction	21.9	7.0	3.1	$65659
Transition	3.5	1.4	2.5	$10367

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.8	2.2	0.5
Environment/CM	0.2	0.6	1.1	0.2
Requirements	0.7	1.2	1.8	0.1
Design	0.3	2.5	3.5	0.1
Implementation	0.1	0.9	7.4	0.7
Assessment	0.1	0.7	5.3	0.8
Deployment	0.1	0.2	0.7	1.0

For further details, please continue to Section 5.1 (Effort Estimation).

4. Schedule, Milestones, and Deliverables

4.1 Project Scheduling

This section shows how staff is allocated to complete this project across its lifecycle, and relations among tasks.

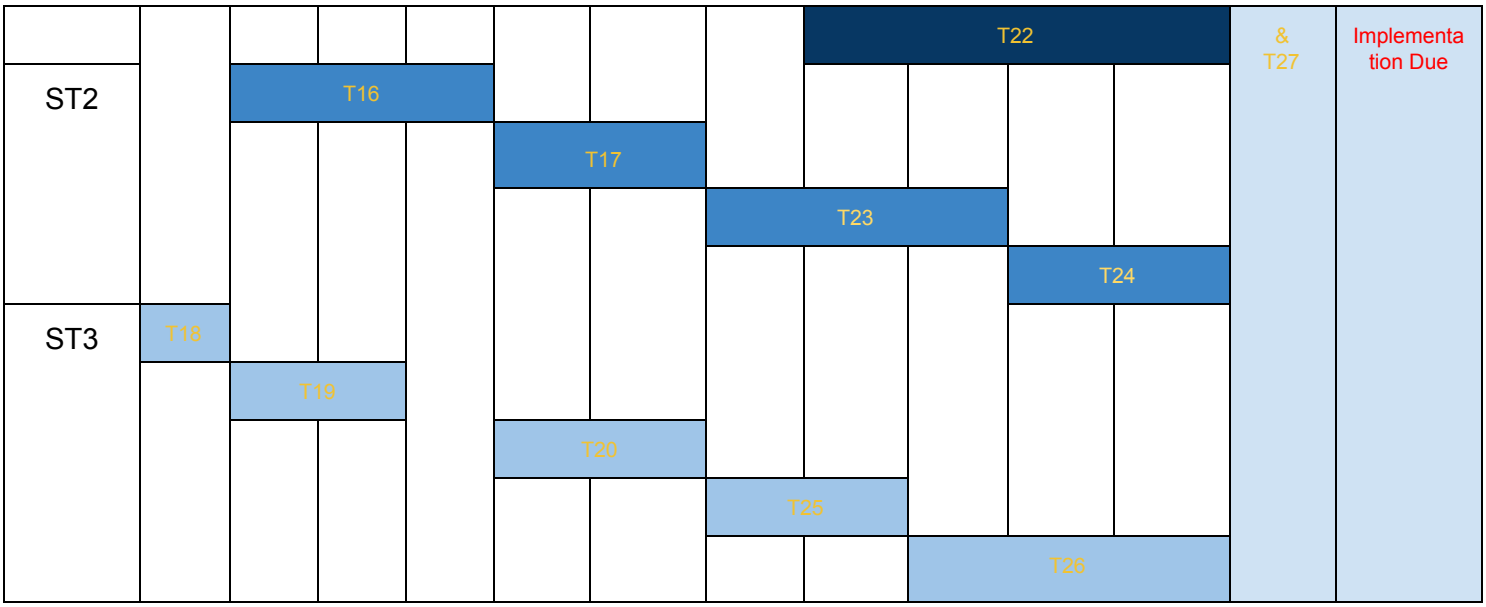
4.1.1 Staff Allocation

The following chart shows how staff is allocated to complete tasks during design phase. For detailed description of each task, please continue to Subsection 4.1.2 (Task Durations and Dependencies).

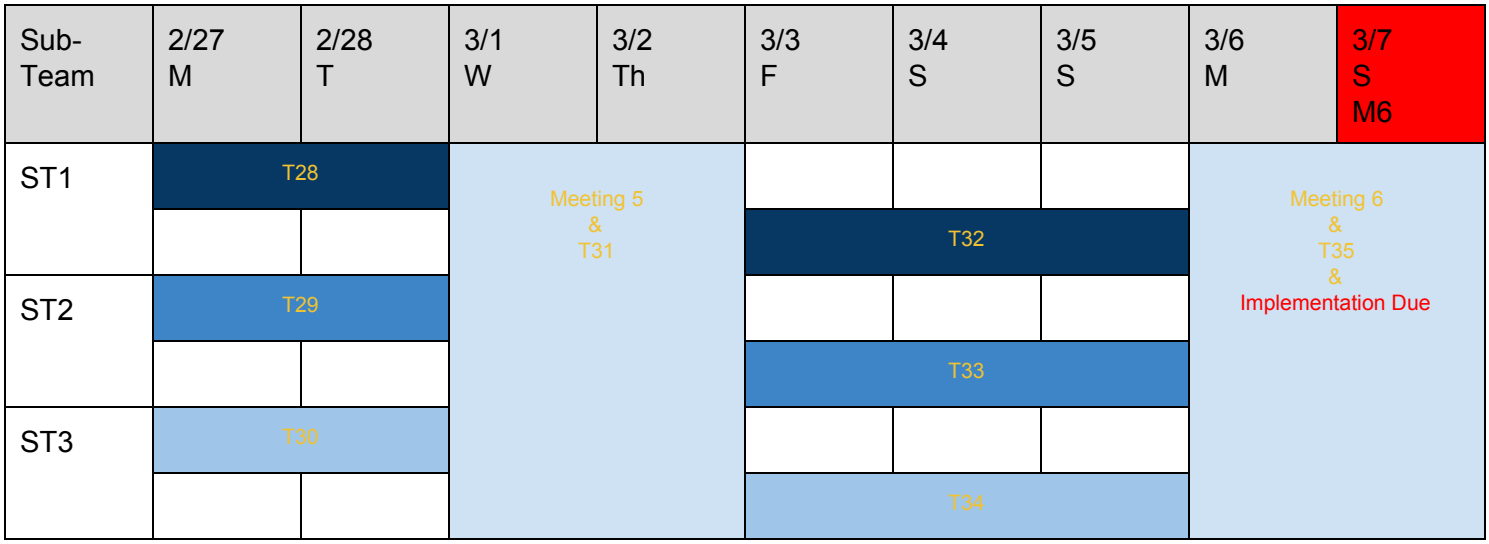
Sub-Team	2/7T	2/8W	2/9Th	2/10F	2/11S (M1)	2/12S	2/13M	2/14T (M2)
ST1	T1		T2		Meeting 1 & T7	T8		Meeting 2 & T11 & Design Due
ST2	T3		T4			T9		
ST3	T5		T6			T10		

The following chart shows how staff is allocated to complete tasks during implementation phase. For detailed description of each task, please continue to Subsection 4.1.2 (Task Durations and Dependencies).

Sub-Team	2/15 W	2/16 TH	2/17 F	2/18 S	2/19 S	2/20M (M3)	2/21 T	2/22 W	2/23 TH	2/24F	2/25S (M4)	2/26 S	2/27 M(M5)
ST1	T12		T13	T14	T15		T21	T22				Meeting 3	Meeting 4 &
							T21						



The following chart shows how staff is allocated to complete tasks during testing phase. For detailed description of each task, please continue to Subsection 4.1.2 (Task Durations and Dependencies).



4.1.2 Task Durations and Dependencies

The following chart describes each task that is scheduled across the lifecycle of this project.

Task	Description
T1	Basic Study on Searching API
T2	Design Search Function

T3	Basic Study on Word Cloud API
T4	Design Word Cloud Function
T5	Basic Study on Web Page Construction
T6	Design Web Page
T7	First Design Integration
T8	Revise Search Design
T9	Revise Word Cloud Design
T10	Revise Web Page Design
T11	Final Design Integration and Complete Documentation
T12	Search Function 1: Get a list of song names by the given artist name
T13	Search Function 2: Get Entire lyrics by the given song name and a given artist
T14	Search Function 3: Output a map of words & frequency by given text
T15	Given an artist and a word, output a map that maps each song of the artist to the frequency of the specified word in each song.
T16	Word Cloud Implementation 1: Generate a word cloud image.
T17	Word Cloud Implementation 2: Make the word cloud image clickable. If the user clicks a word in the image, the system shall print the word.
T18	Build a basic web page for Team2
T19	Construct basic layout and components of the three required web pages
T20	Combine Search Function with Generate word cloud (unclickable) function & Integrate Home Page's search box with Search Functionality

T21	Implement ADD functionality
T22	Profile Distinguishable
T23	Auto-Fill
T24	Implement Back functionality
T25	Integrate Second Page from the word clicked & Integrate Third Page with Second one
T26	Implement Share Button
T27	Final Implementation Integration
T28	Design test cases for search functionality, “ADD” button and profile distinguishability.
T29	Design test cases for word cloud implementation, auto-fill, and “Back” buttons on the Second Page and the Third Page.
T30	Design test cases for UI implementation of all required pages.
T31	Discuss and integrate test cases and design a testing plan.
T32	Run test cases for search functionality, “ADD” button and auto fill. Debug for discovered bugs.
T33	Run test cases for word cloud implementation, profile distinguishability, and “Back” buttons on the Second Page and the Third Page. Debug discovered bugs.
T34	Run test cases for UI implementation of all required pages. Debug discovered bugs.
T35	Go through and all test cases and debug as a group. Debug discovered bugs.

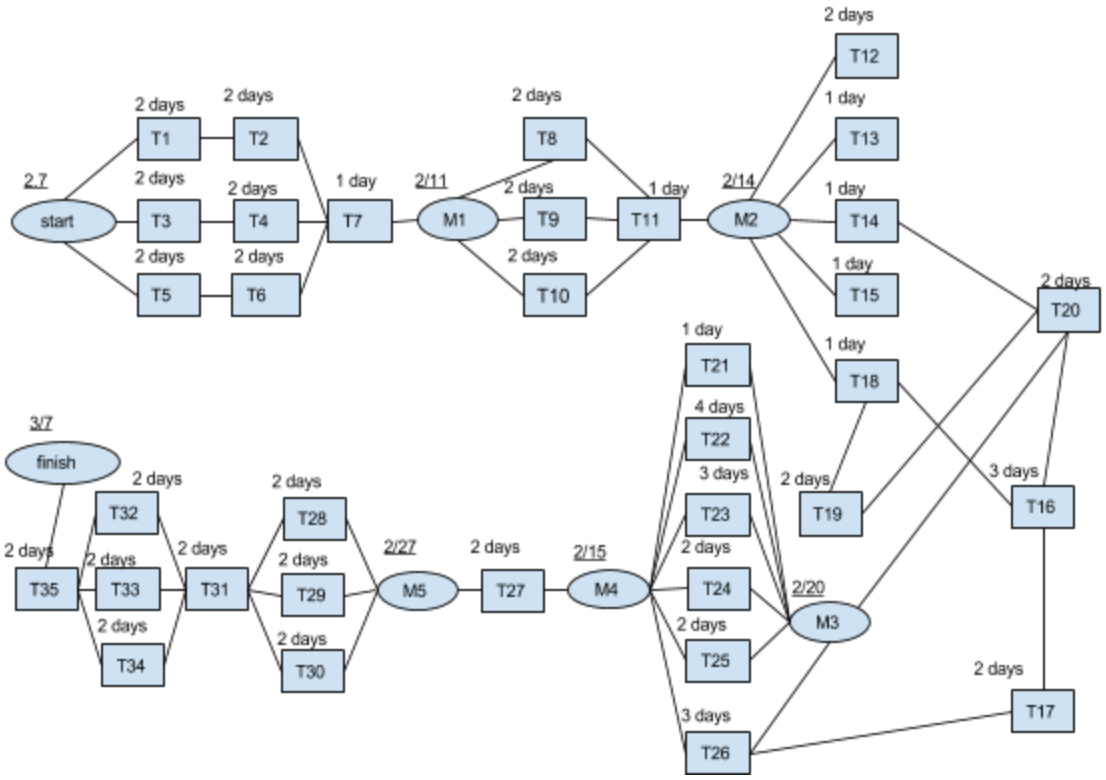
The following chart estimates the necessary task and duration of each task, and its dependency on other tasks, if there is any.

Task	Effort	Duration(Days)	Dependencies

T1	2	2	
T2	2	2	T1
T3	2	2	
T4	2	2	T3
T5	2	2	
T6	2	2	T5
T7	6	1	T2, T4, T6
T8	3	2	T7(M1)
T9	3	2	T7(M1)
T10	3	2	T7(M1)
T11	6	1	T8,T9,T10
T12	4	2	T11(M2)
T13	2	1	T11 (M2)
T14	2	1	T11 (M2)
T15	2	1	T11 (M2)
T16	6	3	T18
T17	5	2	T16
T18	2	1	T11 (M2)
T19	4	2	T18
T20	5	2	T14,T16,T19
T21	2	1	T20(M3)
T22	8	4	T20 (M3)
T23	6	3	T20 (M3)
T24	4	2	T20 (M3)
T25	4	2	T20 (M3)

T26	8	3	T17, T20
T27	6	1	T21, T22, T23, T24, T25, T26 (M4)
T28	4	2	T27 (M5)
T29	4	2	T27 (M5)
T30	4	2	T27 (M5)
T31	12	2	T28, T29, T30
T32	4	2	T31
T33	4	2	T31
T34	4	2	T31
T35	12	2	T32, T33, T34

4.1.3 Activity Network



4.2 Schedule Milestones

Milestone	Description	Criteria	Deadline
M1	Basic Design	Each subteam handles the detail design of their corresponding tasks. The whole team integrates these designs into one software architecture design and concludes a coding standard for implementation.	2/11 After the meeting
M2	Complete Design	Each subteam submits the revised detail design and combines them together to complete	
M3	Basic Function Complete	T12, T13, T14, T15, T16, T17, T18, T19, T20 completed. The project shall be able to search for a valid artist and generate an unclickable word cloud of the lyrics of the specified artist's all songs. If the artist does not exist in the database, the error page shall display. The second and third page shall be displayed with hard coded input.	2/20 11:59PM
M4	Advanced Function Complete	T221, T22, T23, T24, T26 completed. After viewing the word cloud of an artist, the user can click "ADD" button and search for another artist; the project shall generate a word cloud of all the lyrics of the two artists' songs. When user searches the artist by typing in the search box, the dropdown list of the search box shall generate a list of all possible artist names. When the user clicks the artist name in the dropdown list, the project shall generate a word cloud of the lyrics of the specified artist's all songs. If two artists have the same name, the dropdown list shall show each artist's profile in order to distinguish them. When the user clicks the artist name and profile in the dropdown list, the project shall generate a word cloud of the lyrics of the specified artist's all songs. By clicking the word in the word cloud, the user shall navigate to the Second Page. By clicking the "Back" button in the second page, the user shall navigate to the Home	2/25 11:59PM

		Page. By clicking the “Back to Word Cloud” button in the third page, the user shall navigate to the Home Page. By clicking the “Back to the Song List” in the third page, the user shall navigate to the Second Page. By clicking the “Share” button below the word cloud, the user shall be able to share the word cloud on facebook.	
M5	Complete Implementation	Each member of the group check if the functionalities have met the requirements. Stakeholders shall have viewed the system.	2/27 11:59PM

5. Staff and Personnel Plan

5.1 Effort Estimation

The process of the project will be separated into three section: Design, Implementation and Testing. During the design section, each subteam will first be given 2 days to start a basic study on what they are going to implement. Then each subteam will design the detail functions and structures of their own tasks in the following 2 days. And then, the whole team will meet together, integrate three subteams' detail designs and come up to a coding standard that will be used in the future implementation(1 day). Each subteam will revise their detail design of this project in the next 2 days. All team members are expected to contribute at least two hours each to their own section during this period. Effort estimation in this time will be 3 Person-Month. At last, we will put our detail design together and form the complete version.

During the implementation section, all the tasks are respectively assigned to three subteams. Before Milestone 3, Subteam 1 will take care of all the searching functionalities in the first five days. Subteam 3 will first construct a basic web page for Subteam 2 on the first day and then they will continue work on the construction of web pages and integrate other teams' work. Subteam 2 who will be in charge of word cloud implementation will spent the next four days on it. Everyone will be expected to contribute at least four hours to their tasks and complete their work by Milestone 3. After Milestone 3, Subteam 1 will be in charge of profile-distinguishable functionality (4 days) and add button (1 day) implementation; Subteam 2 will take charge of auto-fill (3 days) and back button functionality (2 days); Subteam 3 will be responsible for integration of all web pages (3 days) and implementation of share button (2 days). Each member is supposed to spend at least four hours on their task and be sure to complete all the task by Milestone 5. There will be two buffer day before the actual due date. In that period time, the whole team will meet together to check bugs, find errors and integrate the whole project all together. We expect to spend 5 hours together during that weekend. Effort estimation in this time will be 14 Person-Month.

During the testing section, each team member are expect to write their own testing cases of the entire project in three days. And we will have a meeting to combine all of the testing cases together and run all of the testing cases. This process might take one day to combine and the other to run all the cases. After running all the testing case, we will revise the project code to be perfection in the following three days. And at last we will run the testing cases again and complete the documentation in the last two days. During this period of time, each member is expected to effort at least three hours per day on revising and testing. Effort estimation in this time will be 8 Person-Month.

Communication is vital to a group project. Since we divide the entire team into three subteams. The communications among each subteam and between each subteam members are required. One hour meeting or facetime meeting are required between two members of a subteam in order to inform the process of each subteam task. We will have a two-hour meeting every three days besides the scheduled meetings on project plan.

5.2 Staffing Requirements and Needs

Team members are required to have basic knowledge of C++, Java and data structure. The optimal needs is being proficient in PHP or Javascript language. The capability of fast-learning to new API and other new knowledge. They have to do basic study on web construction and different API that is going to be used.

All team members are required to accomplish all their tasks on time. If encountering any problem that he or she cannot solve, he or she should report that to the whole team, seeking the help without hesitation.

As the team member of this project, each team member have to contribute at least two hours per day during design period, four hours per day during implementation period and two hours per during testing period. An one-hour communication between subteam members is required. During the weekends, the project needs all subteam to spend at least 4 hours together to integrate their tasks.

5.3 Team Structure

The whole team will be divided into three subteams to which two members are assigned.

Ernest and Leo form Subteam 1;

Lifan and Mengqian form Subteam 2;

Juntao and Yawen form Subteam 3.

These three subteams are in a flat structure where each subteam takes responsibility for its own tasks. Two members of the same subteam collaborate together, supervise each other's process and take over the other's work if possible. Meanwhile, Subteam 3 which takes responsible for Web Page construction will monitor the entire project process.

5.4 Description of Team Members

5.4.1 Ernest's Skills and Work Experience

Skills:

Languages: C++, Java, HTML, MySQL, Python

Operating System: Mac OS X, Windows, Linux(Ubuntu), IOS(phone), Android

Tools: GitHub, Virtual Machine, Visual Studio, Xcode, Eclipse

Experience:

Teaching Assistant, ITP 165: Introduction to C++ Programming January 2017 - present

5.4.2 Leo's Skills and Work Experience

Skills

Languages: C++, C, Java, HTML, CSS, JavaScript, MySQL, Python, Matlab
Operating System: Mac OS X, Windows, Linux(Ubuntu), IOS(phone), Android
Tools: GitHub, Virtual Machine, Visual Studio, Xcode, Eclipse, Tomcat Apache, Spring MVC

Experience

Software Engineering Intern June 2016 – August 2016
91 Jinrong CO., Ltd, Beijing, China
Develop a dynamic website using spring MVC that can store, modify, delete, search personal information.

5.4.3 Lifan's Skills and Work Experience

Skills

Programming Language: C++/C#, Java, PHP, Bash
Operating System: Window, Linux (Centos, Ubuntu)
Software: Unity, Eclipse, Microsoft Office, Microsoft Visual Studio, Android Studio, GitHub

Experience

Software Engineering Internship June 2016 – August 2016
Keen (Shanghai) Cloud Computing Technology Co., Ltd, Shanghai, China
Using Exploiting tools to find and exploit the vulnerabilities of smart home devices.
Responsible disclosure these vulnerabilities to the manufacturers.
Use Android Studio and dSploit to develop an app that can automatically unlock a WIFI-connected safe box.

Developer Internship July 2015 – Aug 2015
Goto45.com Shanghai, China
Build up the remote office system
Assist construct the internal website and UI
Build up a Wiki website for new employees to learn rules

5.4.4 Mengqian's Skills and Work Experience

Skills

Languages: C++, C, Python, MATLAB, Java, HTML, JavaScript, SQL
Operating System: Window, Linux(Centos, Ubuntu), Mac OS X, IOS

Software: Eclipse, Microsoft Office, Microsoft Visual Studio, GitHub

Experience

Software Developer Intern Summer 2015
Southwest Securities, Chongqing, China
Built a program that estimates VaR in 100 days based on analysis of historical data

Research Assistant & Software Developer Summer 2016
Lenovo Core Technology Lab, Beijing, China
Built a NVME tool to transmit data using C

5.4.5 Juntao's Skills and Work Experience

Skills

Languages: Java, JavaScript, Swift, C#, C++, PHP, HTML
Design Software: Maya, Photoshop, Fuse, Final Cut Pro
Version Control: SourceTree, Git, Perforce
Software: Visual Studio, Xcode, Eclipse, MySQL, Unreal 4, Unity

Experience

Quality Assurance Intern
Shanghai Jiacheng Packaging Machinery Manufacturing CO., LTD 2016
Part of the team that made plans for website development for the company.
Part of the team that examine and produce videos for the company.
Implemented a HTML5 focused web for advertisements on Wechat.

5.4.6 Yawen's Computer Science Experience

Yawen have had taken USC computer science course CS103 in which learnt the Cplusplus knowledge, CS104 where learnt the data structure and CS210 in which learnt Java language and experience and complete a team project.

6. Software Quality Assurance Plan

6.1 List of Artifacts

The following artifacts will be produced during the lifecycle of this project.

- Software Requirements Specification (SRS)
- Project Management Plan (PMP)
- Design Document
- Implementation Document
- Testing and Delivery Document
- Code
- Development Notes
- Testing Code
- Webpage

6.1.1 Software Requirements Specification (SRS)

Description	A document that lists the requirements of the software product that we intend to build
Standard of Quality	Understandability: The document should be clear to the readers. Correctness: The information presented on the document should have no logical mistake and fulfill client's requirement. Professionalism: The document should be professional.
Quality Measurement	1.The Software Requirement Specification (SRS) has a clear layout and overviews throughout the document, as it follows the IEEE standards. 2.The SRS contains everything that had been negotiated between the developers and stakeholders. 3.The SRS follows the constraints that are stated on the professionalism slide.
Activities	This artifacts will be handed in on January 30th, 2017 and handed back to the developer if there are mistakes be to fixed.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be changed. Instead, changes will be mentioned and corrected in the next patch or documents depends on the current phase.

6.1.2 Project Management Plan (PMP)

Description	A formal plan that defines the execution, monitoring and controlling of the project
Standard of Quality	Professionalism: The document should be professional. Thoroughness: The document should provide enough information in each section. Clarity: Each sections of the document should be discussed unambiguously. Justifiable: Each sections should include justifications for each presentations given.
Quality Measurement	1.The Project Management Plan follows the constraints that are stated on the professionalism slide. 2.The Project Management Plan has provide in-depth explanations on the topics presented. 3.The Project Management Plan has discussed all the contents with great details with no ambiguities. 4.The Project Management Plan has been justified in details on the reasons of why certain methods are used.
Activities	This artifacts will be handed in on February 7th, 2017 and handed back to the developer if there are mistakes be to fixed.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be changed. Instead, changes will be mentioned and corrected in the next patch or documents depends on the current phase.

6.1.3 Architecture and Design Document

Description	A document that provides overall guidance for the development team on the architecture of the software project
Standard of Quality	Professionalism: The document should be professional. Correctness: it measures the correctness of the information are presented compare to the negotiations and documents between the developer and stakeholders. Clarity: Each sections of the document should be discussed unambiguously. Justifiable: Each sections should include justifications for each presentations given. Thoroughness: The document should provide enough information in each section.

Quality Measurement	1.The Architecture and Design Document follows the constraints that are stated on the professionalism slide. 2.The Architecture and Design Document must deliver a detailed and accurate design for the software. 3.The Architecture and Design Document must follow the plans that are discussed in class. 4.The Architecture and Design Document must provide justifications for all designs including PMP. 5.The Architecture and Design Document must provide an appropriate amount of details on the design.
Activities	This artifacts will be handed in on February 15th, 2017 and handed back to the developer if there are mistakes be to fixed.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be changed. Instead, changes will be mentioned and corrected in the next patch or documents depends on the current phase.

6.1.4 Implementation Document

Description	A document that defines the details of implementing the software that is specified in the early negotiations and documents.
Standard of Quality	Professionalism: The document should be professional. Thoroughness: The document should provide enough information in each section. Clarity: Each sections of the document should be discussed unambiguously. Understandability: The document should be clear to the readers.
Quality Measurement	1. Implementation document should follow the guidelines from the professionalism slides. 2. Implementation document should describe how each component will be implemented with details. 3. Each section in implementation document will be discussed with clear words. 4. Implementation document has a clear layout and overviews throughout the document.
Activities	This artifacts will be handed in on February 27th, 2017 and handed back to the developer if there are mistakes be to fixed.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be

	changed. Instead, changes will be mentioned and corrected in the next patch or documents depends on the current phase.

6.1.5 Testing and Delivery Document

Description	A document that specify the testing cases that will be used before the delivery of the software.
Standard of Quality	Professionalism: The document should meet the professionalism requirement. Thoroughness: The document should provide enough information in each section.
Quality Measurement	1. Testing and Delivery Document should meet the requirements on the professionalism slides. 2. Testing and Delivery Document should test every functionalities of the product and consider every cases and scenarios that may occur.
Activities	This artifacts will be handed in on March 8th, 2017 and handed back to the developer if there are mistakes be to fixed.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be changed. Instead, changes will be mentioned and corrected in the next patch or documents depends on the current phase.

6.1.6 Code

Description	Lines of codes written in PHP and JavaScript that will become the final product of this software.
Standard of Quality	Correctness: it measures if all the requirements have been fulfilled. Readability: it measures if the codes generated is easy to follow Reusability: it measures how the codes can be developed in the future. Maintainability: it measures how easy the software will be maintained.
Quality Measurement	1. The code should have meet the proposed lines of code. 2. The code should have low cyclomatic complexity or within limit specified by stakeholders. 3. The code should have low Length of identifiers or within limit specified by stakeholders. 4. The code should have no Error message after run or compiled.

	5. The code should have low depth of inheritance tree for future development or within limit specified by stakeholders.
Activities	Before each milestones discussed in section 4.2, there will be 3 sub-teams that will regroup and grade the codes written by other sub-teams. Their code will be graded by the standards listed above. If some teams are falling behind, we can identify the problem and help that person or groups together. If there are major problem in the codes, we will have three sub-teams run debugger to figure out the problem. If some of programmers in the team decide to leave, we will assess the situation and discuss with the stakeholder on which parts of the program features will be neglected for that person changes. Each testing will be recorded into a text document named "Developer Notes" and stored in each updates on Bitbucket.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will not be changed. Instead, changes will be mentioned and corrected in the next update.

6.1.7 Development Notes

Description	Notes that are included in every major development updates in Bitbucket. It contains the major changes to the software and any lines of code changes if new requirements is added.
Standard of Quality	Thoroughness: The document should provide enough information in each section. Clarity: Each sections of this document should be stated unambiguously. Justifiable: Each changes made should include justifications. Completeness: It measures the completeness of updating this document.
Quality Measurement	1.This document should clearly states the changes made for this/ current updates. 2.This document should be clear about this contents. 3.This document should includes justifications on the modifications. 4.This document should always be included in each updates for users, stakeholders and future reference.
Activities	Before start the testing for current development, development group will randomly assign a group member to assess the quality of this document. It will avoid mentioning modification more than once. If there is any changes to that particular Developer Notes, these

	changes will be added to the end of current Developer Notes. This document will always store in a new folder along with the new code.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, this document will be changed accordingly.

6.1.8 Testing code

Description	Testing code that is generated for each specific feature of the software. It will run the software in different environment and cases to test the software's reliability, compatibility and correctness.
Standard of Quality	Clarity: it measures if the testing codes are commented properly Justifiable: it measures the intention of creating certain testing codes. Completeness: It measures the completeness of each test features.
Quality Measurement	1.Each testing code should have comments about it. 2.Each testing code should contains description on its intended testing areas. 3.Every features discussed in the negotiations should have its own testing code.
Activities	Before start the testing for current development, development group will randomly assign a group member to assess the quality of this document. It will avoid mentioning modification more than once. If there is any changes to that particular Developer Notes, these changes will be added to the end of current Developer Notes. This document will always store in a new folder along with the new code.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, these testing codes will be modified for new requirements.

6.1.9 Webpage

Description	The final working product of the software, a web, that can search artists by their names to generate a word cloud.
Standard of Quality	Completeness: It measures the completeness of each features discussed in the negotiations. Correctness: it measures if all the requirements have been correctly

	implemented. Professionalism: The final product should be professional.
Quality Measurement	1.All features discussed in the negotiations and documents should be completely implemented. 2.All features discussed in the negotiations should be implemented correctly. 3.The web should look professional to stakeholders and the developing team.
Activities	At each milestone mentioned in the 4.2 section, all team members will meet together to test out the features that are supposed to be implemented at that phase. We will demonstrate that feature constantly with the stakeholders, CPs and see if that one is identical to their requirements. In the middle of the developing process, the developing team will also meet with stakeholder, Professor Halfond, to ensure the quality. Two weeks before the software deadline, developing team will once again meet with both CP and Professor Halfond to have a final quality check on the final product. If there are discrepancies in the process of developing at an early phase, the developing team will fix that issues as soon as the next patch update. The developing, however, will try their best to ensure the quality and fulfill the requirements given. If something happens at the very end phase of development, we will have a negotiation with the stakeholders to see if there is any compromise the developers and the stakeholders can make. Before the software is released, it will be tested at least by all the programmers in the team and also by other testers (USC students) in the field.
Changes	If terms and conditions discussed in this document will be updated in future meetings with the stakeholders, changes will be made primarily by negotiations. Notes will be added to the developing notes.

7. Configuration Management Plan

7.1 System Building

In the process of developing this software, we will divide the group into three subteams, and use the technique of pair programming. The group members will meet at Leavey Library to collaborate and generate a balanced and efficient plan for each subteam according to the Architecture and Design Document. Each subteam will separate the tasks, finish their assigned sections and commit to their sub-branch on the group's Github repository. Two days prior to the deadline for each submissions, all the subteams will meet, compile and merge all the codes to one complete file and commit it to BitBucket as a updated version of the software.

7.2 Version Control

All the versions generated in the process of development will be stored in Bitbucket. Each version will be differentiated by their names of their versions and their dates of release and put into a separate folder. Each folder contains two separate folders that contains 1) all the source code 2) all the relevant documentations that is used in the process of developing. It also contains a readme file that specify the changes made in the updated version and relevant notes for that specific patch. On the developer's side, each developer has his or her own repository in Github. They maintain their commits to the project and merge on the day of meeting, and merge conflicts will be solved on the day of merging.

7.3 Change Management

For the modifications of the requirements on the project, there will be an immediate follow up by the developers with the stakeholders. A more specific lists of modifications, a updated document, will be generated for the stakeholders and developers. The list will include the things that will be changed, timing adjustment for the new requirements, personal movements of any kind to support this or these modifications. If those modifications or some cannot be fulfilled with the given time frame, more meetings will be held by the developers with the stakeholders to discuss the details of the softwares until a final list is created. Developers will meet to discuss how the development will proceed with the new requirements. The updated list will be place under the document folder in the most recent patch on Bitbucket.

7.4 Release Management

Two lists consists of major and minor programming mistakes will be generated in the developing process before each updates is pushed to the Bitbucket. One for PHP programming language and another for JavaScript language. The two lists will be included in the documentation folder on Bitbucket. Each subteam's work will be examined if there is any

programming errors in other subteam's works by going over these 2 lists. Then developers will focus on the tests that are listed on the testing documentations to test the functionalities that were implemented in the specific update. Each developer should focus on another developer's code to identify if their codes are clear, correct and satisfy the requirements.

8. Project Monitoring Plan

The chapter identifies the approach of monitoring the process of the projects.Metrics are defined in the description before the table. Specific measurements for each task are provided in the second column.

8.1 Effort Monitoring Plan

The metrics of the following plan is PM. And since we student currently doesn't have a fix timetable for working, we decide to self-estimate our effort time on this project. Monitors are supposed to use this measurement to value the effort estimation on this project.

Title	Monitoring and Measurement	Way of Prevention
Effort Monitoring	On every three-day meeting, everyone should give his or her actual time that effort on his or her own tasks during the previous three days.	N/A

8.2 Coding Monitoring Plan

The metrics of the following plan is task. And a specific measurement will be given to verify the completeness of one task. Monitors are supposed to use these measurement to value the progress of the project.

Task	Monitoring and Measurement	Way of Prevention
T1	Each member does research on Searching API. Two members communicate and exchange their knowledge about the API and how to design their task in order to monitor the research process of each other.	The daily meeting between two subteam members.
T2	Draw a diagram about the searching functionality.	The diagram should be uploaded to google doc before meeting.
T3	Each member does research on Word Cloud API. Two members communicate and exchange their knowledge about the API and how to design their task.	The daily meeting between two subteam members.
T4	Draw a diagram about the word cloud functionality.	The diagram should be upload to google doc before meeting.

T5	Each member does research on web page construction. Two members communicate and exchange their knowledge about the web page and how to design their task.	The daily meeting between two subteam members.
T6	Draw a sketch about how three web pages would look like and where each element should be placed.	The diagram should be upload to google doc before meeting.
T7 & Meeting 1	The whole team should come up with a diagram about the entire project.	
T8	Two members should write the detailed searching functions and variables that will be implemented in the project.	The detail design should be upload to google doc before meeting.
T9	Two members should write the detailed word cloud functions and variables that would be used to build the project.	The detail design should be upload to google doc before meeting.
T10	Two members should write the detailed web page frame and UI for the project	The detail design should be upload to google doc before meeting.
T11 & Meeting 2	The whole team should unite three subteams' detailed design and different variable names that other subteams can use. The result should be a complete documentation with detailed design.	Subteam 3 should group the meeting and manage the progress of the meeting.
Design Due/ Implementation Start	Measurement	
	Input	Output
T12	An artist name	A list of songs name
T13	A list of songs name	A list of entire lyrics
T14	A list of entire lyrics	A map that maps each word to its frequency
T15	An artist name	A map that maps each song of

	and a word	the artist to the frequency of the specified word in each song.	
T16	A map of word corresponding to their frequencies	A image version of word cloud	Monitored by Subteam 3 whose next task has dependency on this task.
T17	1. A map that maps each word to its frequency 2. Click on one word	Print the word on screen.	Monitored by Subteam 3 whose next task has dependency on this task.
T18	N/A	A web page with header, body and tail.	Monitored by Subteam 2 whose next task has dependency on this task.
T19	The url address	Three web pages satisfied the requirements	
T20	An artist name	An image version word cloud on the Home Page	Have the complete task upload to Bitbucket. Monitored by Subteam 1&2.
T21	Enter another artist's name in search box and Click ADD	New result combined with the previous result	Have the complete task upload to Bitbucket. Monitored by Subteam 3.
T22	A name with duplicate artists	Different profiles are shown in the drop down list	Have the complete task upload to Bitbucket. Monitored by Subteam 3.
T23	An incomplete artist name	Potential artist names are displayed in the drop down list	Have the complete task upload to Bitbucket. Monitored by Subteam 3.

T24	Click the Back Button	Navigate back to the previous page	Have the complete task upload to Bitbucket. Monitored by Subteam 3.
T25	Click on one word in the word cloud & Click one of the song in Second Page	Navigate to the target page.	Have the complete task upload to Bitbucket. Monitored by Subteam 1&2.
T26	Click the button	Facebook API shows up and the user can share the word cloud image to Facebook timeline	Have the complete task upload to Bitbucket. Monitored by Subteam 1&2.
T27 & Meeting 3	N/A	The complete project that can run perfectly with correct testing cases.	Subteam 3 should group the meeting and manage to complete the entire implementation one day before due.
Implementation Due/ Testing Start	Monitoring and Measurement		
T28	Each team member writes a list of test cases and expected results and exchange ideas with the other team member. Bring the list to next meeting.		Each team member should upload their test case into google doc before meeting.
T29	Each team member writes a list of test cases and expected results and exchange ideas with the other team member. Bring the list to next meeting.		Each team member should upload their test case into google doc before meeting.
T30	Each team member writes a list of test cases and expected results and exchange ideas with the other team member. Bring the list to next meeting.		Each team member should upload their test case into google doc before meeting.
T31 & Meeting 5	The whole team should get an entire test cases for this project. Each testing case will be compiled.		

T32	Form a checklist of test cases. After fixing one of the bugs, one of the subteam members should check for that case and inform the other member. The checklist should be filled before the next meeting.	Each team should upload their checklist before fixing bugs and continue update the checklist.
T33	Form a checklist of test cases. After fixing one of the bugs, one of the subteam members should check for that case and inform the other member. The checklist should be filled before the next meeting.	Each team should upload their checklist before fixing bugs and continue update the checklist.
T34	Form a checklist of test cases. After fixing one of the bugs, one of the subteam members should check for that case and inform the other member. The checklist should be filled before the next meeting.	Each team should upload their checklist before fixing bugs and continue update the checklist.
T35 & Meeting 6	Run all the testing cases again. Complete the deliverable documentation.	Documentation should be completed 12 hours before due in case of other emergencies.

9. Risk Management

A risk is a probability that some unpreferable situation or adverse circumstance will occur. This chapter explains this team's method in identifying risks, and drafting plans to minimize their effects on this project.

9.1 Risk Identification

Risk Type	Possible Risks
Technology	(1) Software development encounters some difficulties that cause this system to fail in performance requirements (i.e. inefficient algorithm and lengthy runtime) (2) The database for this system is not eligible for processing large amount of operations as needed. (3) The design of this system violates the guideline of loose coupling, which makes the integration of front-end and back-end components difficult.
People	(1) Some group members may not be available during project/exam-intense periods of the semester. (2) Some group members are not proficient in the programming languages (i.e. PHP and JavaScript). (3) Some people may withdraw from this course in the middle of the semester.
Organizational	(1) Subteams are rearranged and new subteams need more time to familiarize with the material.
Tools	(1) Integration of Spotify API and Facebook API is difficult to implement.
Requirements	(1) Changes to requirements, which will cause major modification to original designs, are proposed past the design phase. (2) STAKEHOLDERS fail to understand this system's vulnerability to significant requirement changes.
Estimation	(1) Effort required for a key feature of the original design is significantly underestimated.

9.2 Risk Analysis

Risk Description	Probability	Effects
One or more group members withdraw from this course	Moderate	Catastrophic

in the middle of the semester		
Changes to requirements, which will cause major modification to original designs, are proposed after the original designs are submitted.	Low	Catastrophic
Integration of Spotify API is difficult to implement.	High	Serious
The time consumption of software development is underestimated.	High	Serious
Some group members may not be available during project/exam-intense periods of the semester.	High	Serious
The database for this system is not eligible for processing large amount of operations as needed.	Low	Serious
Some group members are not proficient in the programming languages (i.e. PHP and JavaScript).	High	Tolerable
Software development encounters some difficulties that cause this system to fail in performance requirements (i.e. inefficient algorithm and lengthy runtime)	Moderate	Tolerable
The design of this system violates the guideline of loose coupling, which makes the integration of front-end and back-end components difficult.	Low	Tolerable
Integration of Facebook API is difficult to implement.	Low	Tolerable
Subteams are rearranged and new subteams need more time to familiarize with the material.	Low	Insignificant

9.3 Risk Planning

This section considers each risk and develops a corresponding strategy to identify and counter that risk, providing at least one avoidance strategy, one minimization strategy, and one contingency.

9.3.1 Course Withdrawal

- Avoidance Strategy:
  This group will summon every member and person who withdraws. The leaving person will express his/her concern and current dilemma that enforces the withdraw. Group members first reflect on themselves and fix the internal issues if the problem can be

resolved inside the team. If there are external factors beyond the control, this group will seek help from STAKEHOLDERs.

- Minimization Strategy:
  Knowing withdraw is inevitable, the rest group members re-assign tasks on each individual and have the most appropriate member continue the work of the leaving member.
- Contingency Plan:
  A specific contingency plan is prepared beforehand. In case it happens, the other member from the same subteam will attempt to finish two thirds of work assigned to the whole subteam. The other one third of work will be distributed to other subteams. Everyone else in the group is willing to assist that alone person unconditionally.

9.3.2 Changes to Requirements

- Avoidance Strategy:
  With the clear understanding of how changes to requirement delay the schedule, this group needs to negotiate with STAKEHOLDERs who propose the new changes. The purpose of negotiation is to hopefully retain the original requirements and avoid the new changes.
- Minimization Strategy:
  Minimization strategy is used if STAKEHOLDERs refuse to cut off new requirements that is catastrophy to original design. This group will negotiate again with similarity between new and old requirements, and demonstrate which feature can be implemented without overturning the original rules.
- Contingency Plan:
  It is very important for this group to cease what ongoing tasks they are dealing with. A immediate meeting between this group and STAKEHOLDERs will be hold where . If STAKEHOLDERs insist of having new requirement, this group will produce a upgraded version that satisfies new requirements.

9.3.3 Underestimated Time Consumption

- Avoidance Strategy:
  In order to completely avoid the time underestimation, the approach for this group could be analyze the reason behind. To solve the certain problem, there are various way. A very efficient method could be to reduce unnecessary portion of codes and neglect some irrelevant conditions without violating the requirements.
- Minimization Strategy:
  This group will discuss to figure out the reason that causes the delay and come up with a solution plan. Additionally, this group should reschedule the time line so that the delay is minimized to the least and notify STAKEHOLDERs about the potential delay.

- Contingency Plan:
  This group should provide an updated version of schedule that suffice the most parts of STAKEHOLDERs requirements. The new version of schedule must be carried out as soon as possible.

9.3.4 Members' Availability

- Avoidance Strategy:
  If that member can manage assignment with no time conflicts and become more efficient on problem solving, this risk can be avoided and the schedule won't be affected by personal issues such as midterms.
- Minimization Strategy:
  It is quite understandable for everyone that personal business would be time-consuming. Other members can split up the task to alleviate the burden on that person and carry out the original plan as scheduled.
- Contingency Plan:
  When this risk happens, the other subteam member has to fill the vacancy and assure the assignment is deliverable and completed in time. On the other hand, the person who leaves for midterms should fully acknowledge what phase of software development is and keep himself up to the pace of the development.

9.3.5 Subteams Rearrangement

- Avoidance Strategy:
  Due to the fact that it is inefficient to familiarize members with the new assignment, it is the best for group's interest that subteams remain same throughout the whole development process. In order to avoid the subteams rearrangement in the first place, a group meeting will be hold and figure out the reason for the rearrangement. Finally, this group will come up with an alternative solution that achieved the same goal without breaking down original subteams.
- Minimization Strategy:
  Since minimization strategy is executed after avoidance strategy fails, it is most important to familiarize subteam members and the techniques that will be used in assignments. New formed subteams should spend more time doing pair coding and have
- Contingency Plan:
  This team must consider the case where subteams are rearranged due to certain reasons which can be categorized as a PEOPLE type of risk, so that a specific avoidance strategy is prepared in advance.

9.3.6 Inadequate Proficiency in PHP and JavaScript

- Avoidance Strategy:
  If STAKEHOLDERS would like to entirely avoid the potential delay that such a inadequacy might cause, this group recommends to outsource the design and implementation phase to experienced project managers and software engineers, who are much more proficient in PHP and JavaScript.
- Minimization Strategy:
  The reality in this group is that most of the members are not familiar with PHP or JavaScript. However, all of this group's members are very proficient in Java programming, data structure, and object-oriented programming, which makes this group a team of fast-learning programmers. This group will learn PHP and JavaScript during the design phase, in order to minimize the delay of project delivery that such an inadequacy might cause.
- Contingency Plan:
  In case such an inadequacy resulted a significant delay, this group would inform STAKEHOLDERS of the potential delay in advance. This group would negotiate with STAKEHOLDERS to modify the project schedule, and to lift the delivery deadline of this project delivery.

9.3.7 "Loose Coupling" Principle: Frontend-Backend Integration, and Adoption of Spotify API and Facebook API

- Avoidance Strategy:
  To avoid the difficulty to properly integrate frontend and backend components, and Spotify API and Facebook API, this group will follow the "Loose Coupling" principle. During the time period of component integrations, this principle will ensure the smoothness of inter-subteam collaboration.
- Minimization Strategy:
  The "Loose Coupling" principle can be well executed to minimize the risk of difficult integration, because there exists a peer-supervision mechanism at the intra- and inter-subteam level. To be specific, two persons, who belong to the same subteam, will advise and supervise each on the design and implementation of this system.
- Contingency Plan:
  If following "Loose Coupling" principle still posed the risk of difficult integration, this group would have to reconsider the original design, and negotiate with STAKEHOLDERS to lift the delivery deadlines.

9.3.8 Fulfill Performance Requirements

- Avoidance Strategy:
  Following the "Loose Coupling" principle described above, this group has the capability

to rewrite a specific algorithm or to optimize a specific block of code. For example, if the original design planned to use bubble sort algorithm to implement the lyrics ranking algorithm, and failed the 1-second response requirement, this team has the capability to effectively and efficiently rewrite the lyrics ranking algorithm, by changing the bubble sort algorithm to merge sort algorithm.

- Minimization Strategy:
  During design and implementation phase of this project, this group encourages all members to design and implement the best known algorithm of the this objective. For example, inside this project's lyrics ranking algorithm, which requires sorting algorithm, this system should implement merge sort instead of bubble sort.
- Contingency Plan:
  If the above avoidance strategy and minimization strategy still failed the performance requirements, this group will inform the STAKEHOLDERS, and negotiate to lift the performance requirements on response time.

9.3.9 Process Large Dataset

- Avoidance Strategy:
  This group encourages algorithm analysis on designs. This group makes sure that no exponential algorithms are used in this project, which might make the system constantly fail on large input dataset.
- Minimization Strategy:
  The primary concern about large dataset is runtime. Following the minimization strategy that helps to fulfill performance requirements, this group encourages to choose the best known algorithms, so that this system has better time complexity.
- Contingency Plan:
  In case that this system failed on large dataset, this group would negotiate with STAKEHOLDERS about extending the deadlines, and reconsider this project's original design.

9.4 Risk Monitoring

9.4.1 Management Progress Meeting Schedule

This group will conduct two-hour management progress meetings regularly three times each week. The schedules are 10:00 am - 12:00 pm every Tuesday and Thursday, and 1:00 pm - 3:00 pm every Sunday.

9.4.2 Management Progress Meeting Agenda (Emphasis on Risk Monitoring)

- Assess each identified risk and its probability given the current progress

- Assess whether the effect of each identified risk has changed
- Assess whether any unidentified risks will arise

Appendix I. Factors of Cost

Scale Drivers

	Very Low	Low	Nominal	High	Very High	Extra High
PREC	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar
FLEX	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals
RESL	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
TEAM	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
PAMT	Weighted average of "YES" answers to CMM Maturity Questionnaire					

Product Factors

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	Slight inconvenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life	
DATA		D/P < 10	10 <= D/P < 100	100 <= D/P < 1000	1000 <= D/P < 10000	
CPLX	Not compex	Little complex	Somewhat complex	Much complex	Very complex	Extremely complex
RUSE		None	Across project	Across program	Across product line	Across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-size to life-cycle needs	Excessive to life-cycle needs	Very excessive to life-cycle needs	

Platform Factors

	Very Low	Low	Nominal	High	Very High	Extra High
TIME			<= 50% use of available execution time	70%	85%	95%

STOR			<= 50% use of available storage	70%	85%	95%
PVOL		Major change every 12 months; minor change every 1 month	Major: 6 months; minor: 2 weeks	Major: 2 months; minor: 1 week	major : 2 weeks; minor 2 days	

Personnel Factors

	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
APEX	<= 2 months	6 months	1 year	3 years	6 years	
PLEX	<= 2 months	6 months	1 year	3 years	6 years	
LTEX	<= 2 months	6 months	1 year	3 years	6 years	
PCON	48% per year	24% per year	12% per year	6% per year	3% per year	

Project Factors

	Very Low	Low	Nominal	High	Very High	Extra High
TOOL	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature lifecycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
SITE	International; some phone, mail	Multi-city and multi-company; individual phone, FAX	Multi-city or multi-company; narrowband email	Same city or metro area; wideband electronic communication	Same building or complex; wideband electronic communication, occasional video conference	Fully; interactive multimedia
SCED	75% of nominal	85%	100%	130%	160%	