1. Introduction

1.1 Project Overview

The primary objective of this project is to grasp a better understanding of the lifecycle of software development. For this specific project, objectives would include creating a software system that would allow a user to search for an artist of their choice and generate a word cloud containing the most popular used words from their lyrics database. This program will be called C-lyrics, and set to be delivered and readily available for public use by March 11, 2015.

There are three major milestones that will be completed and have subsections under each. These milestones consist of the completion of all requirements set for each interface page, such as the home page, songs page, and lyrics page. More details on these milestones can be read in section 4.2. The master schedule will include five phases of product development. These phases include designing the user interface, implementing each function within the interfaces, testing these interfaces, adding finishing touches, checking with the client, and submitting. A more detailed Gantt chart will be located in section 4.1. A more detailed description of costs can be found in section 3.

By completion, contributing members of this project should be able to fully understand a product lifecycle as well as carrying out a complete deliverable to all stakeholders, Dr. William G. Halfond and Sonal Mahajan.

1.2 Project Deliverables

The contributing authors to this documentation will create a fully functioning software program that will allow anyone to specify any artist and generate a word cloud comprised of that artist's most frequently used words. This software program will also have the functionality to list the specified artist's songs containing any word from the generated word cloud, and will allow the user to select one of these listed songs to display its lyrics. This program will be delivered as a product called C-lyrics. The client, Dr. William G. Halfond, shall receive all documentation associated with the product as well as the product itself on March 11, 2015 at THH 208 and on Blackboard. Only a single copy of each deliverable shall be provided. These deliverables shall include software deliverables and document deliverables. The software deliverable will include a working software that will satisfy all set requirements and specifications made at the beginning the project, and delivered online through Blackboard. Document deliverables will include this current document as well as past and previous documents to completely satisfy the clients' request.

1.3 Evolution of the SPMP

The SPMP for C-lyrics will be under a version control tool, GIT, monitored by all contributing members, so any changes will be made to the plan itself. The clients will be updated when significant changes have been made to the prototype. The client and development team will be able to access the updated document.

1.4 Reference Materials

[1] IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998. [2] "word cloud". Oxforddictionaries.com. 2015. http://www.oxforddictionaries.com/us/definition/american_english/word-cloud (January 31, 2015) [3] EchoNest API documentation. 2011. http://developer.echonest.com/docs/v4/index.html#overview (January 29, 2015) [4] Van Vliet, Hans. Software Engineering: Principles and Practice. 3rd ed. Indianapolis, Indiana: Wiley, 2008. 170-71. ##1.5 Definitions and Acronyms TODO

2. Process and Organization

2.1 Software Lifecycle Process

The software lifecycle process for this project will be based on the Waterfall Model. In this model, each phase of software development needs to be completed before attempting to work on the next phase. Testing is the last phase of the process which takes place after the software has been implemented.

2.2 Team Organization

The organizational structure of our team is flat with no specified leader or hierarchy. Our team is structured by collaboration across all members resulting in group decisions on who will work on each part of the overall project. All internal deadlines and deliverables are set as a team and communicated to everyone during team meetings and emails. Each member of the team is equally responsible for every finished deliverable as there is no team lead for any specific component.

2.2.1 Current Group Process

The current group process in place for the development team is to meet as an entire group whenever it is deemed necessary by a general consensus. The consensus is gathered by means of group text messaging. At each meeting, the development team decides the tasks to be completed, progress made as of the time of the meeting and delegates future tasks to each member for completion. This process has proved itself highly effective as of the date of this publication and it is closely estimated to continue through the entire life cycle of the software development process for C-lyrics.

2.2.2 Staff and Personnel Plan

The C-Lyrics team will consist of the members of the development team. Each member is proficient in object oriented design and concepts, and has some familiarity with web development. Knowledge of git and other version control methods is a must. The duration of need will be until the last milestone and deliverable is reached. Each member will be assigned tasks based on his or her prior knowledge. Since there will be no training, members will have to research on the job if they are insufficiently prepared. Personnel will not be phased out and retention will not be a factor.

2.2.3 Staff Allocation

2.2.4 Team Member Description

This section provides a brief description of the members of the development team and their relative experiences. Each member is a student at the University of Southern California and is affiliated with the Viterbi School of Engineering Computer Science Department. For additional information on each member, refer to **appendix** for resumes.

3. Cost Estimation

3.1 Cost Influence

The cost influencing factors that have been considered follow the standard COCOMO II model for estimating project effort. These include the four main categories of product factors, platform factors, personnel factors, and project factors. The detailed list of cost drivers and the associated estimations can be found in section 3.3 of this document.

3.2 Cost Schedule Relationship

Cost and schedule are related based on the fact that this particular project contains no defined wages in terms of fiscal costs. Therefore, the only real “cost” is the effort and time that will be put into creating the C-lyrics project. In other words, the schedule is perfectly correlated with the cost of the project.

3.3 Development Costs

Our development costs have been estimated using the COCOMO II Model. The following results are estimates of the cost drivers that are applied to the model [4].

—-check table—- Product Factors Reliability required: very low, 0.75 Database size: low, 0.93

Product complexity: low, 0.88 Required reusability: nominal, 1.00 Documentation needs: very high, 1.13

The project requires very low reliability, since it would only pose a slight inconvenience if the C-lyrics platform encountered problems and was not available. We will most likely not need a database since the lyrics generation will be based on an online API, therefore the database size is listed as low. The low product complexity was selected based on an average of the nominal control operations level and the very low computational operations level. The reuse of components in the product is nominal, since components only need to be reused within the project. The documentation relative to the complexity of our project is very high.

—check table— Platform Factors Execution-time constraints: nominal, 1.0 Main storage constraints: nominal, 1.0 Platform volatility: low, 0.87

The execution-time constraints are nominal since we will require less than 50% of available execution time. The main storage constraints are nominal since we will require less than 50% of available system storage. Platform volatility is low since the hardware and software required to run our program will not need frequent updates.

—check table— Personnel Factors Analyst capability: very low, 1.50 Programmer capability: low, 1.16 Application experience: very low, 1.22 Platform experience: low, 1.10 Language and tool experience: low, 1.12 Personnel continuity: very high, 0.84

Analyst capability is very low since the group's ability to create detailed, high-level design for software projects is minimal. Programmer capability is low because of the lack of significant industry programming experience of the team. Language and tool experience is low since most of the team's programming experience centers around one or two functional languages and little experience with software tools. Personnel continuity is very high since all members of the team will continue to be working together over the course of the project.

—check table— Project Factors Use of software tools: low, 1.12 Multi-site development: very high, 0.84 Required development schedule: very high, 1.0

The use of software tools is designated as low because we will not be using highly developed tools that are structured for the purposes of our project. The multi-site development rating is very high, since the group has consistent communication both in person and through electronic channels. The required development schedule is listed as very high since the implementation phase of our project exceeds 160% of the expected time to complete the project.

The effort in terms of person-months is calculated as the KSLOC multiplied by the product of all cost drivers according to COCOMO II. With an estimated KSLOC of 1, the number of person-months that this calculation returns is 1.247.

3.4 External Costs

Some of the external costs that are not directly factored into the project costs include: purchasing the class textbook for research, programming resources, and transportation costs for group meetings.

4. Schedule, Milestones, and Deliverables

4.1 Schedule

4.1.1 Activity Network

(See section 4.2 for the specific task encompassed in each milestone.)

4.2 Project Milestones and Deliverables

The process model for our project is the Waterfall Model, meaning we will finish each big software development phase before we start the next. We will finish requirements engineering before starting the project design, which will be finished before we start implementation, which will be finished before we start testing, which will be finished before we arrive at the final phase of project, maintenance.

Our project initiation was being assigned this project by Dr. Halfond. The first step in this project was beginning to work on the Software Requirements Specification. The subsequent milestones and deliverables are listed below.

Deliverables and Milestones:

- Project Management Plan. 2/9/15
- There are no specific milestones for this deliverable as it will be worked on by all team members and completed in full by February 9th.

- Design. 2/18/15*
- Milestones A-C
  - A: Design of the Home Page
    - A.1: Search bar
    - A.2: WC generation
    - A.3: Share button
    - A.4: Add to Cloud
  - B: Design of the Songs Page
    - B.1: List of songs
    - B.2: Back to Home button
  - C: Design of the Lyrics Page
    - C.1: Lyrics displayed on page
    - C.2: Back to Songs button
    - C.3: Back to Home button
- Implementation. 3/4/15*
- Milestones D-F
  - D: Implementation of the Home Page
    - D.1: Search bar with autocomplete functionality when typing in an artist's name
    - D.2: WC generation with words that can be selected to take the user to the Songs Page
    - D.3: Share button to upload the WC to Facebook
    - D.4: Add to Cloud button to create a new WC based off of words commonly used by both of the specified artists
  - E: Implementation of the Songs Page
    - E.1: List of songs sorted by how frequently the selected word is used in each song
    - E.2: Song titles in list able to be selected, taking the user to the Lyrics page
    - E.3: Back to Home button takes the user back to the Home Page with the WC still displayed and the artist's name still in the Search Bar
  - F: Implementation of the Lyrics Page
    - F.1: Lyrics displayed on page with the selected word highlighted every time it appears in the song
    - F.2: Back to Songs button takes the user back to the Songs Page with the same list of songs still displayed in the same order

        - F.3: Back to Home button takes the user back to the Home Page with the WC still displayed and the artist's name still in the Search Bar

- Testing and Final Delivery. 3/11/15*
- Milestones G-J
    - G: Testing of the Home Page
        - G.1: Search bar with autocomplete functionality when typing in an artist's name
        - G.2: WC generation with words that can be selected to take the user to the Songs Page
        - G.3: Share button to upload the WC to Facebook
        - G.4: Add to Cloud button to create a new WC based off of words commonly used by both of the specified artists
    - H: Testing of the Songs Page
        - H.1: List of songs sorted by how frequently the selected word is used in each song
        - H.2: Song titles in list able to be selected, taking the user to the Lyrics page
        - H.3: Back to Home button takes the user back to the Home Page with the WC still displayed and the artist's name still in the Search Bar
    - I: Testing of the Lyrics Page
        - I.1: Lyrics displayed on page with the selected word highlighted every time it appears in the song
        - I.2: Back to Songs button takes the user back to the Songs Page with the same list of songs still displayed in the same order
        - I.3: Back to Home button takes the user back to the Home Page with the WC still displayed and the artist's name still in the Search Bar
    - J: Testing of the entire product to ensure all pages work together as specified in the SRS

Deliverables with an asterisk (*) by the due date indicate that there are risk factors that may alter the completion date. These risk factors include, but are not limited to, changing requirements for each deliverable and the deliverables taking either more or less time than expected. Due dates for each milestone are listed in the schedule in section 4.1.

The termination activity of our project is presenting the completed product to both Dr. Halfond and Ms. Mahajan and ensuring that both of them are satisfied with our work. However, if this is not accomplished by the final project deadline, 3/11/15, then our project will be terminated regardless of the customer's satisfaction.

5. Configuration Management Plans

5.1 Product Documentation Management

Formal report formats between the client and development team will be structured according to IEEE standards and use LaTeX as a document preparation system for the client. The code documentation will be generated by Sphinx, which provides an easy way to create intelligent and beautiful documentation. In particular, it allows the documentation to be generated in several formats (eg, HTML, LaTeX, PDF) directly from the comments in the source code. Therefore, the documentation will partly be integrated in the code base and will be versioned by git as well. These processes will be used for the Software Requirements Specifications (SRS), Software Project Management Plan (SPMP), and a Design Document (DD).

5.2 Code Base Management

The whole code of the project will hosted and managed by the git program. Specifically, the project is going to be divided in three main repositories; the frontend, the backend and a third repository for general purpose. All three of the repositories will be hosted by GitHub.com and they will belong to the C-Lyrics organisation. This will allow the development team to better coordinate their effort while creating the software, and takes care of most of the merges in the code base. As stated in Section 5.3, the GitHub repositories will be coupled with Travis CI for continuous integration.

5.3 Software Quality Assurance Plan

Reviews and audits will take place after every milestone completed. The development team will review the progress made thus far to make sure deadlines will be met, while observing and complying with the details of the project management plan. If there are noticeable shortcomings and deviations in the current progress of the project from what has been decided in the project management plan, audits to the project will have to be made.

In addition, the software will be developed according the Test Driven Development methodology. This aims to provide an additional measure for the advancement of the project as well as an assurance on its quality. By writing the basic tests first and then implementing the functions so that they pass the tests, the goal of the development process is slightly modified. Instead of aiming to implement every requirement and functionality, the objective is to have all the pre-written test pass. To keep track of such advancements, GitHub will be coupled with Travis CI allowing continuous integration along the process. Finally some code coverage tools (ie, JSCoverage, PHP_Unit) will be used in order to keep track of the percentage of total coverage on the project.

In order to keep track of the progress, the development team will use some a custom metric. Essentially, the metric mixes the objectivity of the test coverage as well as the personal insights of the team members. Each team member will give a grade from 1 to 5 to each one of the other members, based on the amount of work they accomplished. For each member's score will be averaged and the average of these averages will be the final score of the team. This score is then combined with the amount of tests that passes as well as the amount of code coverage. Hopefully, this combination of both an objective metric as well as the team's feeling on how they are doing provides an accurate estimation of the progress on the project. The mathematical details and some additional reasons for the use of this metric are included in the Appendix.

5.4 Project Monitoring Plan

Github's issue tracker and progress reporting system, along with Google Docs, will be used as the primary project monitoring mechanism among team members. Tasks will be assigned to each team member with a description and a project milestone. Each issue completion will correspond to a milestone set in the project management plan. If there is a change in the milestone contents and dates, the issue tracker will be updated in accordance and strictly follow the most up to date plan. Once the task is completed, the team member will submit the deliverables to Github and mark the task in the issue tracker as completed.

6. Risk Management Plan

6.1 Risk Identification

Risk Type Possible Risks Technology The maximum number of queries allowed on EchoNest is reached during testing. The server that is hosting the domain goes down. The website traffic is overloaded, causing a crash. The real-time performance of the software is inadequate. People Staff members do not have the required technical skills demanded by the project and must spend time to learn them. One or more members of the staff are seriously ill or incapable of working. Staff members who lack experience are unable to learn quick enough to meet deadlines. Organisational One or more new staff members are added to the development team. One or more of the current staff members quits. A flat organization produces a lack of accountability and confusion, resulting in delayed deliverables. The waterfall method is ineffective due to changing customer demands and uncertainties. Tools The chosen CASE tools cannot be integrated into the software without serious restructuring. Requirements The client and stakeholder want to alter small details of the product. The client and stakeholder want to restructure the product entirely. Requirements from the client are misinterpreted during implementation. The development team implements features that looks aesthetically pleasing, but is unwanted by the client. Estimation The software takes longer to develop than previously anticipated. The number of people using the software is greater than what was previously decided by the client and stakeholder.

6.2 Risk Analysis

No. Risk Probability Effects 1 The maximum number of queries allowed on EchoNest is reached during testing. Very Low Catastrophic 2 The server that is hosting the domain crashes. Low Serious 3 The website traffic is overloaded, causing a crash. Very Low Serious 4 Staff members do not have the required technical skills demanded by the project and must spend time to learn them. Moderate Tolerable 5 One or more members of the staff are seriously ill or incapable of working. Low Serious 6 Staff members who lack experience are unable to learn quick enough to meet deadlines. Low Serious 7 One or more new staff members are added to the development team. Low Insignificant 8 One or more of the current staff members quits. Low Tolerable 9 The chosen CASE tools cannot be integrated into the software without serious restructuring. Moderate Tolerable 10 The client and stakeholder request to alter small details of the product. High Tolerable 11 The client and stakeholder want to restructure the product entirely. Moderate Catastrophic 12 The software takes longer to develop than previously anticipated. High Tolerable 13 The number of users is greater than what was previously decided by the client and stakeholder. Moderate Tolerable 14 A flat organization produces a lack of accountability and confusion, resulting in delayed deliverables. Moderate Serious 15 The waterfall method is ineffective due to changing customer demands and uncertainties. Moderate Tolerable 16 The real-time performance of the software is inadequate. Moderate Serious 17 Requirements from the client are misinterpreted during implementation. High Tolerable 18 The development team implements features that looks aesthetically pleasing, but is unwanted by the client. Moderate Tolerable

6.3 Risk Planning

Risk No. Strategy Type Solution 1 Avoidance Make smarter test cases that cover more areas of vulnerability in a smaller amount of API queries. 2 Avoidance Host the software on a reliable server, or perhaps change the hosting location based off of previous crashing problems.. 3 Avoidance Select a more robust hosting service. 4 Contingency Plan Allocate staff members to develop parts of the project that best suits their skills. If no one is familiar with the language or practices required, work in teams of two to promote collaboration 5 Contingency Plan Assign the tasks of the ill staff member to various other members, while the ill staff member can update the new members on the current state of the task. 6 Contingency Plan Delay the deliverable output, or submit the incomplete deliverable and patch the mistakes later when given more time. 7 Minimisation Update the new staff member with the current state of the software and introduce low level tasks to help get him or her up to speed. 8 Minimisation Allocate the work of the quitting staff member among the remaining team members. 9 Contingency Plan Find new tools to fill the needs of the software. 10 Contingency Plan Update previous documents with the new information and fill out a revision history of each document. Change the code to fit the demands and, depending on the magnitude of the desired changes, restructuring of the software may be necessary. 11 Contingency Plan Depending on the amount of change desired by the client, the staff may have to create new documents entirely and essentially start a new project. 12 Contingency Plan Change the deliverable dates to compensate for a new estimated time using COCOMO. 13 Minimisation To compensate for more users than previously intended, the software may have to be altered to more effectively handle storage, the EchoNest API licensing may have to be purchased to allow more requests, and the server will need to be able to handle more requests. 14 Minimisation Hold team members more accountable for his or her own tasks. Part of this can be achieved by instilling a productive atmosphere during meetings and other official work times. If there continues to be problems with a flat organization, a hierarchy may need to be introduced to inspire more efficient work. 15 Contingency Plan The development team will have to work around the inefficiencies of the waterfall method. Switching to Agile methodology is not an option. 16 Avoidance The development team will need to structure the code in a way that produces results in the quickest way possible. If it is too slow, restructuring may be required. Caching results is one way to speed up results. 17 Avoidance If the requirements specified by the client is different than what is implemented, the development team will have to fix the unwanted components. This can be avoided by following the requirements document closely, assuming the client specified everything during the formation of the requirements document. 18 Avoidance The development team should follow the requirements document regardless of the opinions of the team on the specifications of components.

6.4 Risk Monitoring

Throughout the development cycle, the team will have to monitor each risk regularly and prepare for it depending on the estimated likelihood that it will occur. Each member will be aware of the risk pertaining to his or her tasks and discuss the chance that they occur at each meeting. Many risks do not involve the actual implementation and cannot be anticipated, therefore these risks can be ignored during team meetings. Larger risks for a particular milestone or deliverable can be noted in the github issue tracker along with the regular team meeting procedure.

7. Appendices

A.1: Appendix 1: Notes From Meetings With Customer

A.2: Appendix 2: Metrics

A.3: Appendix 3: COCOMO Details

A.4: Appendix 4: Resumes? **if section is removed, remove resume reference in team description