1 Introduction

1.1 Overview

The objective of this document is to deliver, demonstrate and document the first implementation of the C-lyrics software product to the customer. It includes detailed description of the code base management policies, mapping to the project management plan, and mappings to the design document for verification purposes. A startup guide to the deliverable software is also included to guide the customer

1.2. Scope

The intended audience of this document however is for the client and the future maintainers of the code base who will presumably be hired by the client. Ideally, the document should clarify the implementation techniques by making them predictable given that they follow the prior documents on design and management and this is the guiding principle with which the document is drafted. Consequently, some sections will include designs and descriptions taken from these prior documents on design and management.

1.3 Definitions and Acronyms

@Please fill in from prior documents, copy pasting not needed as the text in correct format is in Sebastien's computer@

2 Verification

2.1 System Architecture

The system architecture remains mostly unchanged from that which is described in section 2 of the C-lyrics design document. The frontend and backend are split into different repositories in the Github code base much like their division as separate components in the system architecture diagram in section 2 of the design document. The only exception is that LyricFind will not be used as an external API because we were unable to acquire an API key to use the service. This fact simplifies some UML class and component designs.

2.2 Data Flow Deviation

Since LyricFind will not be used as an external API, our original data flow is different than what we actually built. This issue is detailed in Mark Krant's email on February 28, which can be referenced in full in the appendix. LyricFind was thought to be a quick and easy API call to find all the lyrics for songs of a specified artist, however, it is not a free or reliable service and we did not have the budget to pay for access to its functionality. The solution we came up with is to use a web scraper to get lyrics from LyricsFreak.com. The draw back from this solution is that it takes many seconds to load lyrics depending on the number of songs the queried artist has.

2.3 System Design

While some changes were made that affect the design process of C-lyrics, the overall system design remains unchanged. Many aspects like Use Cases for example remain completely the same.

3 Process Documentation

3.1 Overview of Processes

Based on section 2.2.2, of the PMP, headlined "Staff and Personnel Plan", each member of the development team was originally assigned tasks milestones based on their "prior knowledge" with the technologies being used, such as PHP for example. However, given certain limitations and incidents which will be described in the following sections, significant but not detrimental deviations from the processes described in the PMP were undertaken. These deviations were naturally taken in order to assure the timely and expected completion of the software implementation, and the completion of milestones as outlined in the PMP.

3.2 Code Base Management

The whole code of the project is hosted and managed by the git program as mentioned in section 5.2 of the PMP. Specifically, the project has been divided into three main repositories; the frontend, the backend and a third repository for general purpose. All three of the repositories are hosted on GitHub.com. Milestones for the frontend and backend are kept within each respective repository. Note again that some deviations have occured as mentioned above, but within the context of this section, the deviations in questions are in the usage of milestones and issue tracking within the Github.com service. Access to the code base will be provided to the client either upon request (during development phase) or upon completion and verification of the final product

3.3 Original Milestone Assignments

Below is an excerpt from the PMP with complete descriptions of milestones D-F, these lettering naming conventions will from now on be used to refer to the respective milestones. D-F concern the implementation phase of C-lyrics. A-C concern the design phase which is as of this date completed, and milestones G-I concern the testing and final delivery phase which as of the date of this publication is due on March 11, 2015.

- 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

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 of the PMP, risk factors can be found in section 6 of the PMP titled “Risk Management Plan”. Also note that these dates are subject to change by the client and are only tentative (dates are from project schedule given by client, where it is noted that they are subject to change).

To re-iterate the information from the above chart taken from the PMP, and to conform the the following sub sections presentation of changes to the assigned milestones, refer to the below list of milestones and assignees: *D: Justine, Milad* E: Kelsey, Jeff *F: Mark, Sebastien

3.3 Current Milestone Assignments and Deviations

3.3.1 Milestone Deviations

Changes made to milestone assignees:

- D: Sebastien/Justine
- E: Sebastien/Kelsey
- F: Sebastien

The above changes in assignees were a direct consequence of changes to a team member’s expected contribution to the implementation. Milad Gueramian did not have access to the required development tools due to the loss of visual output from his Hewlett Packard laptop computer which was expected for use in development of software covered in milestone D. Consequently, it became necessary to send the machine in for repairs and this team member’s role was repurposed, which is explained later in this section.

Added milestones not specifically accounted for in the PMP:

- (New) Backend: Mark
- (New) Documentation: Milad, Jeff

As mentioned before, the PMP states that the development team process relies on a principle of matching member tasks based on his/her level of prior experience. Through the evolution of our processes, the development team, in an effort to be efficient, relies ever more on this principle. While the time allocation and estimation of milestones D-F were correct at the conception of the PMP, articulation of specific details were not. Therefore, new milestones have since been added to clearly separate and assign tasks to the appropriate team member based on prior experience. Furthermore, the task for the creation of this document was not specifically taken into account in the PMP. Consequently, Jeff Kang and Milad Gueramian-due to the latter's inability to contribute to milestone D-were reassigned to the new Documentation milestone.

3.3.2 Deviations in Project Monitoring Plan

Issue Tracking on Github

In section 5.4 of the PMP, it is stated that the Github.com issue tracking system will be used to monitor milestones. Indeed, this is a fitting management and documentation tool since the future maintainers of the code base will want to have access to the development history of the software. The milestones and assignees were not added in this system until very recently and all at once, which is not ideal because there is no smooth timeline documenting events as they occurred. However, section 5.4 also indicates that Google Docs will be used as a means of tracking issues. This is still true and has helped the development team to accurately keep track of issues and milestones and to retroactively add them in the Github issue tracker and progress reporting system. Furthermore, other communication methods described in earlier documents such as a group text messaging thread and a Google email messaging thread have been useful in keeping track of progress. Therefore, the effects of this deviation were not detrimental in making the planned progress.

3.3.3 Deviation's Cause By Change in Timeline

The client pushed up the due date for the product implementation deliverable from Wednesday March 4, 2015 to Monday March 2, 2015. This put considerable constraints on the development team in completing the milestones set for the original due date at the time of the new, March 2 due date. Consequently, the development team decided to scrap plans for adding some added advantages and assurances to the client and software. Sphinx as a documentation management tool, outlined in section 5.1 of the PMP, will not be used. The impact of this deviation is that the documentation provided by Sphinx form the source code comments will not be created and therefor not available on the code base in Github.

Testing efforts were also seriously hampered by the constrained timeline. The tests were not as inclusive as was previously planned. As stated in section 5.3 of the PMP on software quality assurance, we wanted to adopt the test Driven Development Methodology because, "this aims to provide an additional measure for the advancement of the project as well as an assurance on its quality". Because of the earlier due date, quality assurance is hampered because we did not have the necessary time to run a full coverage of test cases.

4 Startup Instructions

4.1 Background

5. Appendices

5.1 Issue Tracking And Progress Reporting

5.1.1 Gmail Communications

From: Sebastien Arnold, February 28, 2015

So, I just pushed some code on the organization repo. (https://github.com/C-Lyrics/frontend) It is not fully working yet, and there are still a few things to implement, but we are in good shape. From what I understood, Mark has been making some progress as well on the PHP, but we have troubles getting the actual lyrics for a given song, as well as performance issues.

Performance should not be that much of a problem, as we can always speed things up. Regarding the songs issue, I found two unofficial APIs for RapGenius: http://blog.edforson.me/introducing-genius-api-rapgenius-api-as-a-service/ and more complete, but more difficult to understand: http://api.genius.com/search?q=the+recipe

It would be very good if we could have the backend online as well, on the repo of the organization. So that we can have a look at how we want to integrate both parts of the application. You should all be invited to join, if you were not shoot me an email.

Also, someone should begin to take of the document we have to submit for the implementation. Milad, I suggest you do it, that will be easier for you than to code stuff, as you don't have your usual tools for coding.

From Mark Krant, February 28, 2015 Just committed the back end. It works, but to an extent. Since I could not find a free lyrics API, I used a webscraper framework to go to a page for each song by an artist and get the lyrics from there. Only problem is that it takes way too long, and Sebastien said it can be done using asynchronous calls, but I still need to look into that more because I have no idea how to do that. A band with like 15-20 songs might take 10-15 seconds to load, but 150+ songs will cause it to time-out (it times out after 120 seconds). If anyone has suggestions / knows how to do it feel free to edit the code. It is under /backend/templates/getSongs.php

Also i did not do error checking for the back end yet. so if you try to make a word cloud with an artist name "sdfgh" then it will blow up. it's a quick fix so ill do it soon