Cumulyrics

Project Plan

Team 19

Kyle Van Landingham, Alec Schule, Alec Fong, Andrew Zolintakis, Noah Bergman

Executive Summary

This document presents a plan for developing the Cumulyrics application. Development costs were estimated at 4.5 person months using the COCOMO II model. Based on the cost estimate and stated software requirements, the project was broken down into milestones and individual tasks. These tasks were then assigned and scheduled to the appropriate team members. In Section 5, appropriate project artifacts are identified. All artifacts have stated quality benchmarks and acceptable levels of quality. Throughout the development process, team members will use Google documents to collaborate on written reports, GitHub for version control, and Asana to track the project's progress. Significant risks to the development process were also identified. These risks are presented in Section 8 of this document, as well as a plan to track and mitigate them.

Purpose

The purpose of the Project Plan document is to utilize the Software Requirements Specification artifact to develop a Project Management Plan. This process is concerned with ensuring that the system is delivered on time, on budget, and meets the requirements specified in the SRS.

During the construction of this document a plan will be develop such that if followed will allow the team to meet all project objectives. This aspect of the document will involve cost estimation, schedule and milestone development, staffing and personnel decisions, and the development of more plans. Cost estimation will be completed with the help of COCOMO II to develop the plan with cost limitations in mind. Staffing and personnel decisions are important because the members of the team do not have equivalent skill sets so deciding how to most efficiently utilize team members will reduce the difficulty of developing the project. Setting achievable and trackable milestones and developing a corresponding schedule aid in monitoring the development of the project. Contingency plans also allow flexibility for the team to pivot toward if there are complications with the plan the team is currently following.

The second aspect of this project plan is to ensure that the project is going to plan and to identify risks. Activities to ensure that the team is following the project plan include monitoring of capital spent, tracking the schedule and milestone achievements, and assessing the quality of artifacts produced. Capital spent on the project can be roughly equated to effort put into the development of the system. If the team follows the schedule and reaches the milestones then it is a good indicator that the project plan is being followed. Even if the team achieves the other two goals the quality of their progress is also an important factor in determining how well the team is adhering to the project plan.

Lastly, termination analysis will help the team improve processes related to project planning. This part of the document will summarize information about the project based on relevant information gathered from project information. Accomplishing this task allows the team to adjust this process for future projects.

Intended Audience

This document is intended for the stakeholders, which include the CPs, TAs, and professor, and the developers,Team 19.

References

Reference	Relevant Information
Github Repository	https://github.com/kvanland/csci310Team19
Planning and Measurement Lecture Notes	Slides created by Professor Halfond describing the desired format for a project plan document. Accessible via Blackboard.

Definitions

Term	Definition
COCOMO	The Constructive Cost Model or COCOMO is a procedural software cost estimation model developed by Barry W. Boehm.
ASANA	Asana is a software system that enables teams to assign tasks, track progress, and manage projects.
SRS	The Software Requirements Specification is the initial document created in our project process that describes all requirements the system must have.
Fog Index	The measurement of the average length of words and sentences within a document. The higher the fog index the higher the complexity of the document analyzed.

1. Process and organization

The software lifecycle process for this system will be following the waterfall model. The waterfall model gives us specific dates on which we will produce a new deliverable in its entirety. The waterfall model for software development consists of phases that each flow to the next including, requirements, project planning, design, implementation, testing, and maintenance. Changes to phases previously done will cascade and affect the phases farther down the model waterfall structure. This software development process generates a large amount of documentation and planning to reduce the possibility of errors or complications further in the development process.

Our group is comprised of five developers who all share the task of, developing, testing, and delivering the project. One developer will be given the title of project manager who will lead the team meetings in order to ensure every developer is on the same page with regards to the project plan. The project manager will also approve and submit all documentation produced by the team. The shared Github repository will also be managed by the project manager in order to guarantee structural continuity for all documentation and code stored by the team. All team members are in charge of keeping the project on schedule as well as making sure they develop a fully functioning deliverable that meets all of the requirements specified in the SRS, the project manager just takes lead in these aspects of the project.

The team will meet at minimum every three days to collaborate on the documentation and or code. During these meetings the project manager will lead team in assessing the current progress of the project and keep track with brief notes on the topics discussed and actions taken during the meeting. In between meetings, each team member is expected to contribute toward the current project tasks on their own time. The project manager will check in on team members individually between meetings to make sure each team member is on track to complete the current task they are assigned to contribute toward. At any time the project manager can decide to assign tasks to additional developers.

2. Cost estimation

We are using COCOMO II to estimate resource requirements.
Source lines of code are used to determine the cost and time our project will take. We have calculated by analogy that our project will take roughly 1396 lines of source code. We analyzed a similar open source project that generated word clouds along with a template we made in html for the front page. The open source project found here, https://github.com/jasondavies/d3-cloud, uses 980 lines of code to generate the cloud and has far greater functionality and customazation then our project requires. We are estimating that 200 lines will be used for web scraping and database generation based on prior experience and projects done by Alec Schule. Analyzing the API code and projects (https://github.com/rhnvrm/lyric-api) we can estimate our back end php code will take 200 lines. It will take less than 50 lines to implement facebook share features as well as autocomplete with ajax. Our HTML template for the home/front page of our site took 104 lines. We estimate the 3 other pages will take 3 times the amount of HTML code. Thus 196 is derived from 1796 = 980 + 100 + 100*3 + 200 + 200 + 50.

Software Scale Drivers

Scale Drivers	**Level**	**Reason**
Precedentedness	High	There are many examples of word cloud generators with open source code along with similar software architecture and scalability as what we are attempting to achieve.
Development Flexibility	Low	The requirements put forth for review are fairly static and thus the development flexibility is low.
Architecture / Risk Resolution	High	We have many contingencies for a myriad of risks as seen in section 8.
Team Cohesion	Nominal	Our team works well with each other, however we don’t have much experience working with each other
Process Maturity	Low	There is a document that has been reviewed and approved by the supervisor or the approving authority as the standard process. But it may be doubtful that the activity being performed is as per the document. There may be a process drift or some change since the document was drafted. CMM 1(upper half)

Software Cost Drivers

Cost Drivers	**Level**	**Reason**
Required Software Reliability	Very Low	This software is expected to run reliably for all of the standard use cases. However,

		it is not meant to be hyper reliable as a software does not control or affect critical systems. So failure to perform the intended function will only be a slight inconvenience.
Database Size	High	Our database will hold the majority of lyrics and artists so it has a high number of bytes stored to source lines of code ratio.
Product Complexity	Low	This software mostly revolves around two systems, word cloud generation, and lyrics/artists query. The system is a based on a basic server client model.
Developed for Reusability	Low	This software project is not being designed with code reusability in mind.
Documentation Match to Lifecycle Needs	Low	The planning documentation and design documentation need to be finished.
Analyst Capability	Nominal	We are capable analyst but lack many years of experience.
Programmer Capability	Nominal	Capable programmers but lack experience.
Personnel Continuity	Very High	As the project is slated to last half a semester, we are expecting very low personnel turnover.
Application Experience	Very Low	Our team has little experience building applications like our projects
Platform Experience	Low	Our team has very little

		experience with web design.
Language and Toolset Experience	Low	A few of our team members have experience with php, javascript, and ajax.
Time Constraint	Nominal	The time constraint is average given the amount of planning and coding time we have along with the amount of team members.
Storage Constraint	Nominal	Storage is nominal as we estimate our database will take up less than a GB.
Platform Volatility	Low	Web applications are not volatile.
Use of Software Tools	Nominal	We will be using some software tools
Multisite Development	Low	We will be developing at one site, USC
Required Development Schedule	Nominal	We have a development schedule that is semi rigid as seen in section 3.

Person Month
A person month is described as 1 hours of work per day. A work week is days a week.

Effort = 4.5 person months
Number of persons = 5
4.5/5 ~ 27 days

3. Schedule, milestones, and deliverables

3.1 Milestones

1. Finished Project Plan
2. Finished Design Document
3. Front-end working
4. Back-end working
5. Fully Functional Website

6. All tests passed

3.2 Project Tasks

3.2.1 Document Design Tasks

A. Develop high level architecture of system
B. Develop detailed design of system
C. Create full design document
D. Validate and confirm document

3.2.2 Implementation Tasks

E. Create the html/css framework for the website.
F. Create animations and navigation functionality for the website. Create method to handle data for song listing screen and lyric screen.
G. Create script to initialize database and methods to access data.
H. Set up and run the scraping of the APIs to gather data.
I. Figure out how to stem lyrics.
J. Develop the algorithm to find the top 250 words of artist(s) and handle requests for that data.

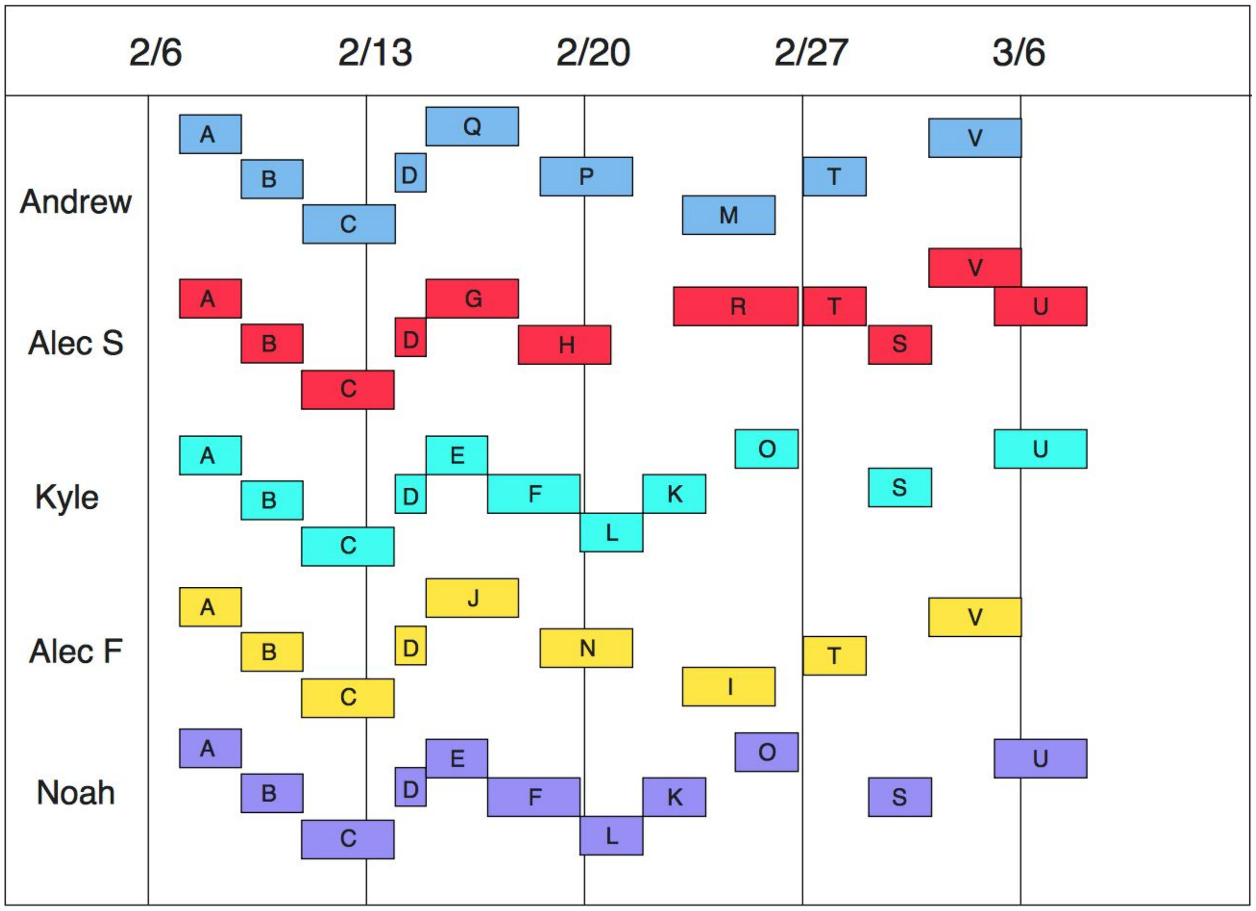
K. Establish method to request data from server and and to process requests.
L. Develop the process for generating the word cloud visually and ability to alter the color.
M. Develop autocomplete and identical artist drop down functionality.
N. Connect share to facebook button to Facebook API.
O. Bring all front-end features together
P. Establish method for getting all songs containing a word and sending that data to the client.
Q. Establish method for getting the lyrics of a song and sending that data to the client.
R. Optimize efficiency to meet requirements.

3.2.3 Testing Tasks

S. Create front-end tests
T. Create back-end tests
U. Run front-end tests and debug
V. Run back-end tests and debug

Task	Duration (days)	Dependencies
A	2	M1 (Milestone 1)

B	2	A
C	3	A,B
D	1	C
E	2	M2
F	4	E
G	3	M2
H	3	G
I	3	M2
J	3	M2
K	2	G,H
L	2	M2
M	3	G,H,K
N	3	E
O	2	E,F,M, N
P	3	M2
Q	3	M2
R	2	J,M,P,Q
S	2	M3
T	2	M4
U	3	S
V	3	T



4. Staff and personnel plan

Team Structure:

We will have a flat team structure with a designated Project Manager. We will make design and implementation decisions as a team. We will also keep our task structure fluid to allow team members to ask for help from other team members if needed.

Team Members:

Andrew Zolintakis- Junior CSBA major at USC. He has three years of programming experience, however no experience with developing websites.

Kyle Van Landingham - Sophomore studying computer science at the University of Southern California. Has no experience with web development but does have experience with mobile development.

Noah Bergman - Junior studying Computer Science at University of Southern California, has limited experience developing with PHP and JavaScript.

Alec Fong- Sophomore studying Computer Science and Finance at University of Southern California, has experience developing using Rails.

Alec Schule- Junior studying computer science at the University of Southern California. Has past work experience in web development using PHP and Javascript.

Roles:
Andrew Zolintakis - Back-End Developer
Kyle Van Landingham - Project Manager and Front-End Developer
Noah Bergman - Front-End Developer
Alec Fong - Full Stack Developer
Alec Schule - Back-End Developer and QA Engineer

Task Scheduling and Effort:
See task duration/dependency chart and task schedule in Section 3

5. Software quality assurance plan

Artifacts

1. Project Plan
    a. Measure of quality: The level of quality shall be determined by how well the project plan fulfills the requirements outlined in the planning and measurement lecture notes.
    b. Acceptable level of quality: An acceptable project plan shall fulfill every requirement given in the planning and measurement lecture notes.
2. Design Document
    a. Measure of quality: The level of quality shall be determined by how well the design document follows Professor Halfond's lecture notes.
    b. Acceptable level of quality: An acceptable design document shall fulfill every requirement given by Professor Halfond.
3. Web Server
    a. Measure of quality: The level of quality of the web server shall be measured by its ability to support all tasks required for the remainder of the development process.
    b. Acceptable level of quality: An acceptable web server shall be fully functional to the extent that it can support all planned development goals.
4. Front-end Code
    a. Measure of quality: The level of quality shall be measured by how many requirements requiring front-end code are satisfied.
    b. Acceptable level of quality: Acceptable front-end code shall satisfy all requirements that depend on front-end code.
5. Back-end Code
    a. Measure of quality: The level of quality shall be measured by how many requirements requiring back-end code are satisfied.

   b. Acceptable level of quality: Acceptable back-end code shall satisfy all requirements that depend on back-end code.

6. Database Complete
   a. Measure of quality: The quality of the database shall be measured by what portion of information needed to implement planned design is present in the database.
   b. Acceptable level of quality: An acceptable database shall have all information present in it required to complete all requirements.
7. Completed Website
   a. Measure of quality: The level of quality shall be measured by how many requirements are satisfied by the completed website.
   b. Acceptable level of quality: An acceptable complete website shall satisfy every requirement given in the requirements document.
8. Test Cases
   a. Measure of quality: The level of quality shall be determined by how adequately the test cases cover any scenario that may arise in using the final product.
   b. Acceptable level of quality: Acceptable test cases shall be ones that, if all passed, guarantee that the final product will perform as intended in all scenarios.
9. Testing Phase Completed
   a. Measure of quality: The level of quality shall be determined by how many test cases are passed by the end of the testing phase.
   b. Acceptable level of quality: An acceptable testing phase is one in which every test case is passed by the end of the phase.

6. Configuration management plans

Due to the large amount of documentation and code generated throughout the development process the team will need an agreed upon way to track and store that data. Storage for the artifacts generated by this project will be through Github where each team member will have access and push privileges to a shared repository. To view the repository for the project follow the link provided in the references section of the document.

Generating documents for the project will be done through Google Drive. Google Drive allows the team to work simultaneously on the document and collaborate in real time to generate artifacts. Each team member will be assigned specific sections of documentation to work on and complete to the best of their understanding and ability without direct in person collaboration. At team meetings continuity will be considered as team members discuss overlapping aspects of the documentation to ensure the document is consistent. Refinement will be done both in meetings and after meetings individually. Finally before the due date of a document, the team will meet in person to overview and check for consistency in the document before submitting in pdf for via blackboard.

Version control for the documents will start at version 1.0 and each time there is a change deemed worthy of a version update a new pdf will be generated from Google Drive and the version number will increase by 0.1. Version numbers are reflected by the version number appended to the end of the document file name e.g. “Project Plan v1.0.pdf”. There will be a “Documentation” folder within the shared repository that will contain folders for each document artifact and within each of those folders are all the version pdfs for those documents. All versions for a specific document will be contained within its specified folder within the “Documentation” folder of the repository. Notes from each meeting will also be stored within the “Documentation” folder so all team members and stakeholders will be able to review the contents of the meetings.

Github will also be used for version control and storage with respect to the code generated for the system. All team member will have access and push privileges to the same shared github repository used for documentation. In order to ensure smooth merging with the different versions of code each team member has on their local drives each task will be done on a separate branch. Once a task is completed it will be merged with the master branch so the code on the master branch will always be functional. The code will be separated into two folders, one for the backend server code and one for the front end code.

At each meeting the team will assess the work completed since the last meeting and the project manager will approve branch merging and document version updates.

Additionally with regards to code guidelines, the team has agreed to provide descriptions for each function in javascript and each method in php. This process will be implemented to improve the readability of the code generated as well as allow other developers to get acquainted with the source code quickly. Comments will also be required to explain code that is not obvious in its functionality. Common coding practices such as refraining from hard coding constants and descriptive variable names will be enforced. Variable naming conventions will be required as follows, all uppercase separated with underscores for global variables and camelcase for all other variables. These naming conventions will be enforced to ensure continuity in the code generated for the project.

With regards to html and css guidelines, the front end should use percentages when sizing and positioning elements. In general the html and css should follow responsive web design principles to ensure that the website remains functional and visually appealing on multiple screen sizes.

7. Project monitoring plan

The team will use Asana to monitor the project schedule and achievement of milestones. Asana is a software system that enables teams to assign tasks, track progress on tasks, and manage projects. Thus, the primary metric for tracking the progress of the project will be task completion progress in Asana. Asana will also serve to ensure all team members understand their responsibilities and are aware of deadlines for tasks assigned to them. In the case that a team member falls behind on their responsibilities, Asana will notify the team, making it easy for other team members to notice and discuss what should be done to keep the project on schedule. Tasks input into Asana will come from section three of the project plan -- Schedule, Milestones, and Deliverables.

In addition to Asana, at each meeting the Project Manager will check up with each developer on how they are doing on their current task. He will assess their progress, quality of work, and likelihood of finishing the task on time. If it seems probable that a team member is unable to meet the deadline for their current task, other team members can be assigned to assist in order to keep the project on schedule. In the unfortunate case in which all members fall behind on their tasks, an emergency meeting will be held to at least ensure that high priority requirements will be completed by the final project dealing. This shall only be done as a last resort.

After each artifact is produced, the team will meet and assess the quality of the artifact. The primary metric for doing so will be measures of quality and acceptable levels of quality described in the software quality assurance plan. In addition, for written documents, the fog index will be used as a metric for how difficult the document is to understand. For each document, the team will strive to keep the fog index under level 12, as an acceptable standard of readability. For code based artifacts, the team will also ensure the written code meets an acceptable standard of quality. To accomplish this, the team will have a group code review for each completed task. These code reviews will take place at the same time as the regular group meetings and will serve to identify obvious flaws in the code, ensure all developers understand what the code accomplishes, and ensure that developers use good coding practices. Good coding practices include self documenting variable and function names, sufficiently commented code blocks, and modular, object-oriented code design. One metric for measuring good variable and function names that we will use is the average length of all identifiers in written code. Longer variable and function names are much more likely to be self documenting and have a clear purpose in the code. Acceptable lengths will be 20 characters for function names and 10 characters for variable names.

8. Risk management

8.1 Risk Identification and Analysis

Risk	Probability	Effects
3rd party API's lock us out for too many calls	Moderate	Tolerable
3rd party API's go down	Low	Catastrophic
3rd party API's take too much time	Moderate	Severe

Team member get sick and cannot complete task	Low	Tolerable
Requirement changes requiring major design changes	Moderate	Severe
Database capacity not adequate	Low	Catastrophic
Team Member gets distracted by other schoolwork	Moderate	Tolerable
The time required to develop the software is underestimated.	High	Serious

8.2 Risk Planning

Risk	Minimization and Avoidance Plan
3rd party API's lock us out for too many calls.	When possible, we will try to store data in our system by running our scripts calling the API's at the beginning of implementation. That way we can stop relying on the API's early on.
3rd party API's go down	The team will use the minimum number of API's to accomplish the requirements.
3rd party API's take too much time	To monitor this risk the team will make multiple test calls to the API's in order to assess the speed of the API calls. The team will also use the minimum number of API's to reduce dependencies on 3rd party software.
Team member gets sick and cannot complete a task	Everyone will do their part and try to stay healthy by getting an adequate amount of sleep and avoiding sick individuals outside of team meetings.
Requirement changes requiring major design changes	We will constantly be in contact with the stakeholders, so that if they request changes to the requirements it will be earlier rather than later. This will minimize the extra time it takes to change a requirement.
Database capacity not	We will fill our database at the beginning of the

adequate	implementation, which will allow us to identify this risk early on.
Team Member gets distracted by other schoolwork	Team members will be aware of schedule conflicts ahead of time.
The time required to develop the software is underestimated	We will stick to our implementation plan to ensure our project stays on schedule. We will also voice any concerns that a task might take longer than expected early so that we can adjust our plans accordingly.

8.3 Risk Monitoring

Risk	Monitor Plan
3rd party API's lock us out for too many calls	The team will assess the data received by the API's utilized for the project and if there are no results given valid input then it would be a possibility that the API locked the system out for too many calls. If we run into trouble we can get another API key or switch to a backup API to get the necessary information.
3rd party API's go down	The team will assess the data received by the API's utilized for the project and if there are no results given valid input then it would be a possibility that the API is down. In the event of this risk occurring the team will pivot to a backup API that would accomplish the same task.
3rd party API's take too much time	The team will assess the speed at which data can be scraped from the 3rd party API's and determine whether or not the system can collect all the necessary data in a timely manner. If the API calls do not allow the system to meet performance requirements the team will store the data in the database so each request by a user can be fulfilled within a more reasonable time frame.
Team member gets sick and cannot complete a task	At each meeting each team member will assess their current health and inform the others if there is the possibility of a complication. If team member is unable to work while sick, other team members will cover his tasks.
Requirement changes	Constant contact with stakeholders will allow the team to

requiring major design changes	assess whether or not the requirements specified in the SRS match the needs of the stakeholders throughout the development process. If there are requirements changes the team will need to rewrite the SRS and all documents that follow the SRS on the waterfall lifecycle.
Database capacity not adequate	At each meeting the team will observe the capacity of the database and assess whether or not the system will be able to store that amount including future data. If this risk occurs the team will pivot to storing less data and using more API calls in real time.
Team member gets distracted by other schoolwork	At each meeting the team will assess their other obligations between the current meeting and the next meeting. Team members will be open to covering other members' tasks in the event that another team member has other work to complete.
The time required to develop the software is underestimated	The team as a whole will assess the project progress at each meeting and adjust the speed and or number of people working on specific tasks according to the project progress. The project manager will monitor progress in between team meetings through slack and Asana.