Project Stratus
Sprint 1 Deliverable

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

Table of Contents

1. Executive Summary

This document outlines the motivations behind the actions taken during the first sprint of Project 2 and documents the work accomplished during the sprint. The overview details the information about the document that is referenced within the document and clarifies ambiguous parts of the document. The product backlog is a prioritized list of features that will be implemented during the lifecycle of the project. The sprint backlog is a prioritized list of features from the product backlog that have been implemented during the first sprint. The sprint backlog for this document includes the search feature, status bar feature, and specify number of articles parsed feature. The task chart breaks down the sprint backlog into tasks that are to be accomplished during the sprint that will when completed indicate that the sprint implementation is complete. The burndown chart indicates the rate at which the team completed the tasks on the task chart to visually indicate the pace of the sprint. The scrum meeting notes document the information exchanged between the team during the stand up meetings. For more documentation than is provided within this document visit the github repository linked in the reference table in section 2.3 and navigate to the folder named “sprint1_documentation”.

2. Overview

2.1 Purpose

The purpose of this document is to track and document the work planned and completed in the first sprint of the project.

2.2 Intended Audience

The intended audience for this document is the product owners and the development team.

2.3 References

Reference	Relevant Information
GitHub Repository	https://github.com/kvanland/csci310TeamP
Cumulyrics	Previous project for CSCI 310. The group for this project is the exact same group that worked on Cumulyrics, so the entire Cumulyrics code base is available for reuse.
Product Backlog Excel Sheet	https://docs.google.com/spreadsheets/d/1IrTDg63ZMwNC10liWfWb5Cs50-KmELNTij1JuijRXHE/edit?usp=sharing

2.4 Definitions

Term	Definition
Product Backlog	A prioritized list of features or requirements for the system to have some time during development.
Sprint Backlog	A sublist of the Product Backlog that contains the features or requirements that the developers will tackle during the current sprint.

3. Product Backlog

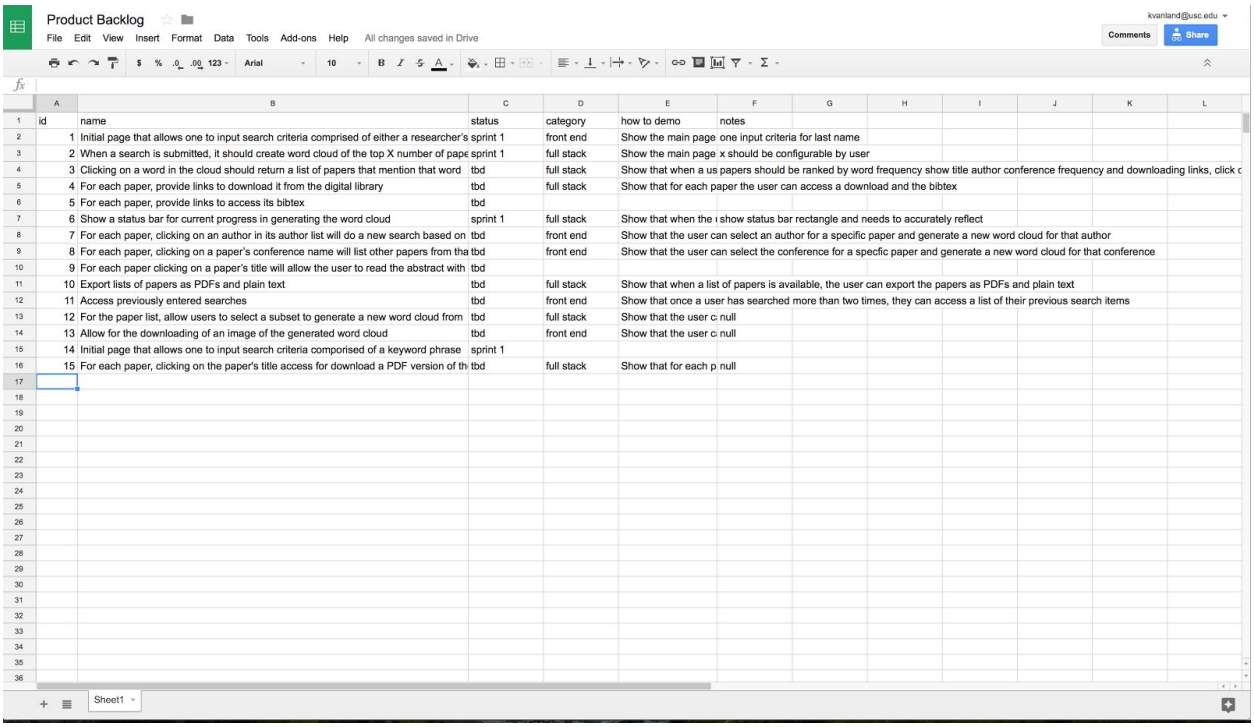
The ordering of the Backlog reflects the priority of each feature where number one is the highest priority and fifteen is the lowest priority.

Project description: Web application to display word clouds associated with researchers' papers that have been published in the ACM or IEEE digital libraries and allow navigation to inspect papers and authors.

1. Initial page that allows one to input search criteria comprised of either a researcher's last name.
2. When a search is submitted, it should create word cloud of the top X number of papers in the ACM and IEEE digital library that match the provided criteria. (X is configurable by the user)
3. Clicking on a word in the cloud should return a list of papers that mention that word
4. For each paper, provide link to download it from the digital library
5. For each paper, provide link to access its bibtex
6. Show a status bar for current progress in generating the word cloud (needs to reflect actual progress)
7. For each paper, clicking on an author in its author list will do a new search based on that author
8. For each paper, clicking on a paper's conference name will list other papers from that conference.
9. For each paper clicking on the paper's title will allow the user to read the abstract with the word highlighted.
10. Export lists of papers as PDFs and plain text
11. Access previously entered searches
12. For the paper list, allow users to select a subset to generate a new word cloud from

13. Allow for the downloading of an image of the generated word cloud
14. Initial page that allows one to input search criteria comprised of a keyword phrase
15. For each paper, clicking on the paper's title allows for download of a PDF version of the article with the word highlighted

Image of Product Backlog within an excel format. Link in the reference table.



Product Backlog

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

id	name	status	category	how to demo	notes
1	Initial page that allows one to input search criteria comprised of either a researcher's	sprint 1	front end	Show the main page	one input criteria for last name
2	When a search is submitted, it should create word cloud of the top X number of pape	sprint 1	full stack	Show the main page	x should be configurable by user
3	Clicking on a word in the cloud should return a list of papers that mention that word	tbd	full stack	Show that when a us	papers should be ranked by word frequency show title author conference frequency and downloading links, click
4	For each paper, provide links to download it from the digital library	tbd	full stack	Show that for each p	aper the user can access a download and the bibtex
5	For each paper, provide links to access its bibtex	tbd			
6	Show a status bar for current progress in generating the word cloud	sprint 1	full stack	Show that when the	show status bar rectangle and needs to accurately reflect
7	For each paper, clicking on an author in its author list will do a new search based on	tbd	front end	Show that the user can	select an author for a specific paper and generate a new word cloud for that author
8	For each paper, clicking on a paper's conference name will list other papers from tha	tbd	front end	Show that the user can	select the conference for a specfic paper and generate a new word cloud for that conference
9	For each paper clicking on a paper's title will allow the user to read the abstract with	tbd			
10	Export lists of papers as PDFs and plain text	tbd	full stack	Show that when a list of papers is available, the user can export the papers as PDFs and plain text	
11	Access previously entered searches	tbd	front end	Show that once a user has searched more than two times, they can access a list of their previous search items	
12	For the paper list, allow users to select a subset to generate a new word cloud from	tbd	full stack	Show that the user c	null
13	Allow for the downloading of an image of the generated word cloud	tbd	front end	Show that the user c	null
14	Initial page that allows one to input search criteria comporised of a keyword phrase	sprint 1			
15	For each paper, clicking on the paper's title access for download a PDF version of th	tbd	full stack	Show that for each p	null

Sheet1

4. Sprint Backlog

1. Initial page that allows one to input search criteria comprised of a researcher's last name
2. When a search is submitted, it should create word cloud of the top X number of papers in the ACM and IEEE digital library that match the provided criteria *(using only the abstract text)
3. Initial page that allows one to input search criteria comprised of a keyword phrase
4. Show a status bar for current progress in generating the word cloud

4.1 Sprint Planning

The team decided that this sprint would contain mostly the framework from which to build the rest of the system with the main functionality only included. The reasoning behind the lighter workload for the first sprint is a large workload in other classes for multiple team members. Two team members had midterms, two other team members had family obligations during the sprint, and one team member had another project to work on. Given these other

obligations the team agreed to complete the framework for the system with an emphasis on making it easy to add features in future sprints.

With the constraints on the first sprint in mind, the team decided to move the above features from the product backlog to the sprint backlog. The first three features are considered the main functionality of the desired system so the team focused on making those parts of the system as simple and easy to add additions to as possible. Although not directly on the sprint backlog, the team did design the system architecture to include an article list for each search query that can be used to complete some of the other features in the product backlog in future sprints. The fourth and final feature on the sprint backlog is the status bar which would have been more difficult to implement in the future so it was added to the backlog out of necessity.

The * note at the end of the second feature mentions that the word cloud is generated from the abstracts of each article because it would have taken too much time to implement the pdf parsing for the full text which will be saved for the next sprint.

5. Task Chart

List of Tasks

LEVEL 1 PRIORITY

- Spike on PHP servers
- Spike on IEEE and ACM APIs
- Spike on system architecture
- Create backend class structure
- Create unit tests for search and word cloud generation for IEEE database
- Create unit tests for search and word cloud generation for ACM database
- Create test cases for search feature (TDD)
- Modify javascript.js from Cumulyrics to include new AJAX calls
- Modify javascript.js from Cumulyrics for navigation
- Modify index.html from Cumulyrics to new design
- create status bar tests

LEVEL 2 PRIORITY

- Decide on how server will know when certain tasks are finished for status bar
- Decide on JSON object replies to different requests
- Implement status bar javascript
- Implement new functions not present in Cumulyrics (Front end)

LEVEL 3 PRIORITY

- Implement search by keyword for IEEE database
- Implement search by author for IEEE database
- Implement backend word cloud generation for data from IEEE database using only abstract
- Implement search by keyword for ACM database

- Implement search by author for ACM database
- Implement backend word cloud generation for data from ACM database using only abstracts
- Implement status bar tests
  LEVEL 4 PRIORITY
- Implement status bar backend
- Talk to TA about previously entered searches
- Develop Front-end skeleton (reusing code from Cumulyrics)
- Integrate APIs into front-end skeleton to enable search with top 20 papers included
- Implement test cases for search feature
- Develop method to specify number of papers included in word cloud

Discarded task	Reason
Implement listener class and skeleton methods for handling all different types of server requests	The system architecture does not have a need for a separate class to handle all different requests. The backend consists of multiple scripts rather than a server architecture.
Create test cases for number of papers feature (TDD)	The tests for this feature are integrated into the other tests and the team felt that there was no need to have a separate test suite for this specific feature.

There are pictures of the excel task chart daily starting from 3/27/17 within the githhub repository for the project. The reason for documenting the task chart so late into the sprint was a recommendation from the professor with reference to the documentation.

5.1 Task Reasoning

5.1.1 Level 1 Priority Tasks

These tasks are the simplest and most essential tasks to starting the project. They are tasks Team 19 have to get done before anything else. For that reason Team 19 has to research use of PHP servers, the IEEE and ACM API's, and effective system architectures for this type of system, so that Team 19 can get a general plan of how the system should be structured and then implemented. Then, there are the the simple backend tasks to be one, which contain creating the backend class structure and the unit tests for the application's main functionality. This will set up a framework for the future backend implementation. Since the front end can be very similar to that of last project, the first thing to be done should be making subtle changes to the old code to get the new application's front end framework working.

5.1.2 Level 2 Priority Tasks

Level 2 tasks contain the server-browser communication. This is a very important piece because all of the logic is located in the php backend. Therefore, once Team 19 gets the basic structure for the frontend and backend down, the tasks are to make sure they can communicate.

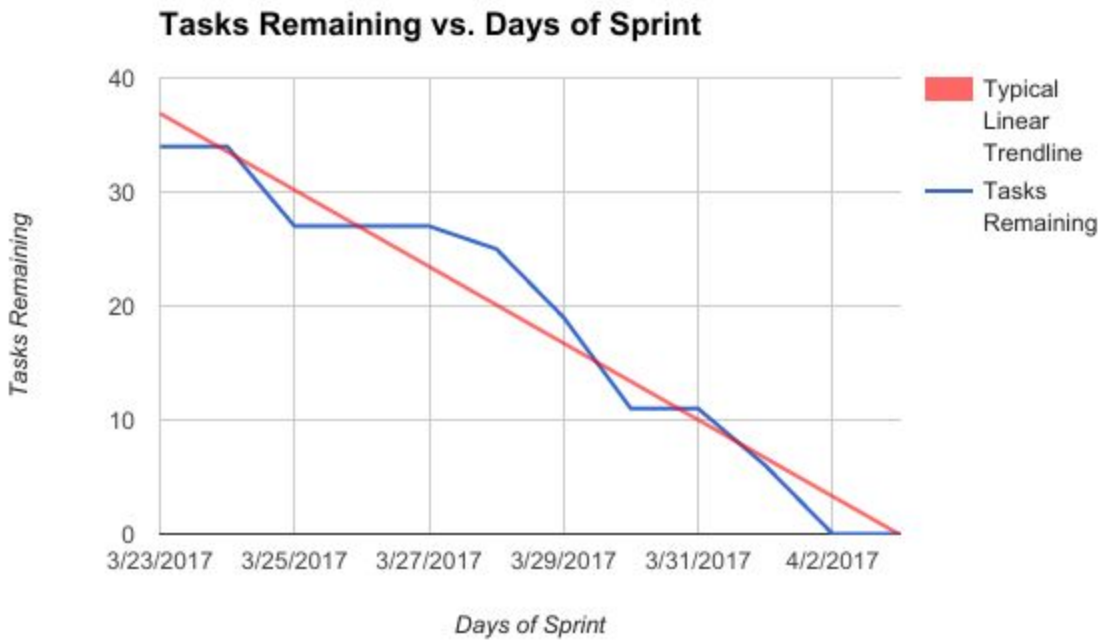
5.1.3 Level 3 Priority Tasks

As a team, Team 19 decided to tackle the main functionality of the application, creating a word cloud from a search term, first. Therefore, the the next most important tasks involve implementing the logic to access online APIs and turn the data from there into a word cloud. Since parsing PDFs could be difficult, the task in this sprint only asks to create the word cloud using abstracts, which will most likely be easier, and then Team 19 can parse the PDF instead of abstract in a future sprint.

5.1.4 Level 4 Priority Tasks

These are different tasks that Team 19 is striving to get done, and would be greatly beneficial, but are either not entirely necessary in the completion of the project or can be pushed off until a future sprint if need be.

6. Burndown Chart



The burndown chart shows that the team generally followed the XP practice of sustainable pace. The team gradually completed tasks at a sustainable pace throughout the sprint in order to prevent a large workload in the final days of the sprint. This trend can also be obtained from observing the task chart screenshots in the github repository under sprint1_documentation/taskChart.

7. Scrum Meeting Notes

There are pictures from each scrum meeting in the github repository.

3/25/17

What did you accomplish since the last meeting?

Kyle - Completed the spike on the ACM digital library and failed to find an official API but did find crossref API to access the ACM digital library

Noah - Completed the spike on the PDF parsing and found pdfparser.org

Alec F - Considered different system architectures

Alec S - Considered different system architectures

Andrew - Completed the spike on the IEEE digital library and found the IEEExplore API

What will you accomplish before the next meeting?

Kyle - Reorganize the task chart and burndown chart documentation

Noah - Figure out the final system architecture plan

Alec F - Work on understanding the crossref and IEEExplore APIs and how to integrate them into the backend section of the system

Alec S - Figure out the final system architecture plan

Andrew - Work on understanding the crossref and IEEExplore APIs and how to integrate them into the backend section of the system

Are there any obstacles that stand between you and getting your plan done?

Kyle - No

Noah - We need to understand whether or not we need a persistent server or just scripts

Alec F - Need to know whether or not the crossref API can access all ACM articles

Alec S - The status bar might alter the general architecture we have in mind so I need to research how to implement a status bar before finalizing the architecture

Andrew - Similar to what Alec said

3/26/17

What did you accomplish since the last meeting?

Kyle - Reorganized the task chart and burndown chart

Noah - Figured out that scripts without a persistent server will work

Alec F - Figured out that the crossref API is the best bet

Alec S - Concluded that the status bar would work with just php scripts

Andrew - Figured out that the crossref API is the best bet

What will you accomplish before the next meeting?

Kyle - Create the test cases for the black box testing

Noah - Repurpose code from last project to use for the current project

Alec F - Begin making unit tests for the backend by figuring out the expected results for various test cases

Alec S - Begin making unit tests for the backend by figuring out the expected results for various test cases

Andrew - Repurpose code from last project to use for the current project

Are there any obstacles that stand between you and getting your plan done?

Kyle - None

Noah - None

Alec F - It will be difficult to figure out the exact expected results for some API calls but it will be possible just time consuming

Alec S - None

Andrew - Similar to Alec

3/27/17

What did you accomplish since the last meeting?

Kyle - Finished the feature files for the behat black box testing

Noah - Finished reorganizing the frontend code

Alec F - Found a good amount of data for test cases

Alec S - Finished reorganizing the frontend code

Andrew - Found a good amount of data for test cases

What will you accomplish before the next meeting?

Kyle - Decide on JSON format and implement status bar in css and javascript

Noah - Implement status bar in css and javascript

Alec F - Finish unit tests for the php and create skeleton for backend

Alec S - Decide on JSON format and improve documentation

Andrew - Finish unit tests for the php and create skeleton for backend

Are there any obstacles that stand between you and getting your plan done?

Kyle - No

Noah - No

Alec F - No

Alec S - No

Andrew - No

3/28/17

What did you accomplish since the last meeting?

Kyle - Implemented the status bar in css and javascript

Noah - Implemented the status bar in css and javascript

Alec F - Finished the unit tests and made the skeleton for the backend

Alec S - Decided on JSON and improved documentation

Andrew - Finished the unit tests and made the skeleton for the backend

What will you accomplish before the next meeting?

Kyle - Finish the Ajax calls and status bar implementation

Noah - Solidify the backend structure documentation and documentation

Alec F - Continue working on the backend communication with the crossref API and IEEExplore

Alec S - Finish the Ajax calls and status bar implementation

Andrew - Continue working on the backend communication with the crossref API and IEEExplore

Are there any obstacles that stand between you and getting your plan done?

Kyle - The implementation might not be compatible with the backend and will only know once we test the integration

Noah - No

Alec F - It will be difficult to figure out how to get all the information from the urls given by the crossref API

Alec S - Similar to what Kyle said

Andrew - IEEExplore will be easy but ACM articles could be more difficult

3/30/17

What did you accomplish since the last meeting?

Kyle - Finished Ajax code and status bar implementation

Noah - n/a

Alec F - Finished communication with APIs

Alec S - Finished Ajax code and status bar implementation

Andrew - Finished communication with APIs

What will you accomplish before the next meeting?

Kyle - Alter color scheme and reformat the front end code

Noah - n/a

Alec F - Finish the backend with abstracts

Alec S - Work on documentation

Andrew - Finish the backend with abstracts and include part for number of articles

Are there any obstacles that stand between you and getting your plan done?

Kyle - None

Noah - n/a

Alec F - It could take more time than expected depending on if I can locate the abstracts with php/curl

Alec S - None

Andrew - Finishing my part of the backend depends on Alec finishing his part

4/1/17

What did you accomplish since the last meeting?

Kyle - Created the option to choose the number of articles used to generate the wordcloud

Noah - Worked on documentation

Alec F - Finished the backend parsing the abstract

Alec S - Created the option to choose the number of articles used to generate the wordcloud

Andrew - Altered the backend to receive the number of articles to be parsed

What will you accomplish before the next meeting?

Kyle - Finish documentation and making sure the backend and front end work together

Noah - Finish documentation and making sure the backend and front end work together

Alec F - Anything needed before the deadline

Alec S - Finish documentation and making sure the backend and front end work together

Andrew - Respond to backend bugs that are found by other team member while away

Are there any obstacles that stand between you and getting your plan done?

Kyle - We might need to drop the status bar implementation if we can't figure out how to get it to work

Noah - We could run into trouble with too little documentation

Alec F - I will be away for the day and am very limited in my ability to contribute tomorrow

Alec S - I can only start working late into the day due to other commitments

Andrew - I will be away for the day but can work remotely if needed

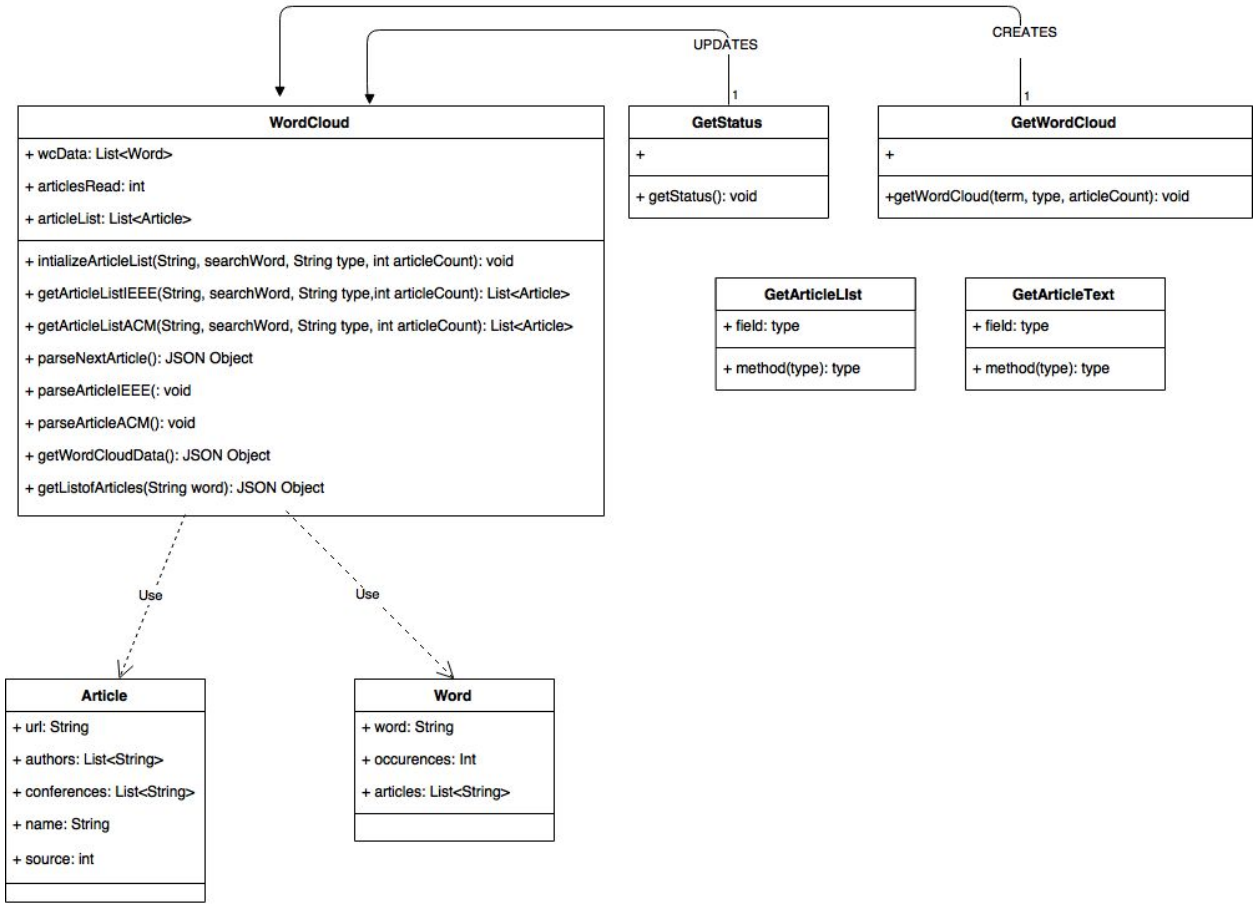
Pair Programming

Evidence of the team pair programming can be found in the github repository under "sprint1_documentation/pairProgramming" which contains images from the times that team members pair programmed.

Test Driven-Development

Evidence of the team utilizing the XP practice of test driven development can be found in the github repository by observing the first number of commits to the repository that contain the feature files for the black box testing and the unit tests for the white box testing. These commits show that the team wrote tests before code and made sure that the code was written with the tests in mind.

Class Diagram



The class diagram represents the logic behind the backend part of the system. The WordCloud class represents a global shared object between all the php files which contains all the information needed to generate a word cloud and the article list associated with that word cloud. The reason for the lack of front end diagram is that majority of the computing logic resides in the backend section of the system.

The interaction between the frontend and backend that is used in the system developed during this sprint is limited to ajax calls to GetStatus and GetWordCloud. The frontend calls GetWordCloud to initiate the backend API calls to gather the articles to be used for the word cloud and then continuously calls GetStatus to poll for the progress of the parsing until GetStatus returns the relevant word cloud data. The reasoning for including classes related to the article list despite the lack of necessity in the sprint backlog is because the design for the first sprint is intended to make it easier to add features later on and these classes relate to that future proofing.

Notes on the topics that were assigned to be spiked.

Spike Topic	Notes
IEEE	Research into available API's to search the IEEE database of research papers showed that there is a IEEE Xplore XML Search API that works perfectly to satisfy the project's requirements. It is accessible through HTTP requests using structured URL queries and delivers an XML result. The API can be can search by Author, Title, and Abstract. In order to access the API the user must be an IEEE subscriber. Team P should be able to it through the USC Library. Lastly, the API can retrieve at most 200 papers for a single request and a request can only contain a maximum of 10 words.
System Architecture	Research different methods for serving the application with a PHP backend and a HTML/Javascript frontend. It is most efficient to make as much use of the previously existing code base (Cumulyrics) as possible. Thus, the recommended architecture for this project is to use the same architecture used in Cumulyrics. PHP will be used to retrieve the information needed to meet the project requirements, and serve the information to the browser via AJAX calls. The information will then be manipulated and displayed using Javascript, CSS, and HTML.
ACM	There is no official API to access the ACM digital library. There is however an API available via https://github.com/CrossRef/rest-api-doc/blob/master/rest_api.md that can access the ACM digital library using the member number 320. This API allows the system to search the ACM digital library based on Author, Term, or Conference title. The API gives URLs to the article pages but most articles are in PDF format so the team needs to figure out how to parse text in a PDF via a third party library most likely.
PDF Parsing	We can use the library at https://github.com/smalot/pdfparser