PrePARED BY: GROUP 6

GRoup members:

ABHISHEK DUTTA

APOORV KHAIRNAR

CHIRAAG SUBRAMANIAN

PRANJAL KARANKAR

PUSHKAR LADHE

RAJAT SINHA

SACHIN MAGAR

Project Plan

Show Me: Related Work

October 31, 2017

Table of Contents

[Project Description 2](#_Toc497255601)

[Overview (CO-5): 2](#_Toc497255602)

[Key Requirements (CO-5): 2](#_Toc497255603)

[Deliverables (CO-5, CO-6): 2](#_Toc497255604)

[Acronyms and abbreviations (CO-7): 3](#_Toc497255605)

[Design and Architecture (CO-1, CO-3) 4](#_Toc497255606)

[Implementation Strategy 8](#_Toc497255607)

[Schedule / Timeline (CO-2): 10](#_Toc497255608)

[Required Hardware (CO-2): 12](#_Toc497255609)

[THird party content (CO-2): 12](#_Toc497255610)

[Quality (CO-2): 12](#_Toc497255611)

[Other Special considerations (CO-7, Co-3): 13](#_Toc497255612)

[process 14](#_Toc497255613)

[Process Description and justification (Co-2) 14](#_Toc497255614)

[Tools (CO-2): 14](#_Toc497255615)

[Roles and Responsibilities (CO-2): 14](#_Toc497255616)

[Location of Project Artifacts (CO-2): 14](#_Toc497255617)

[Sponsor communications (CO-7): 15](#_Toc497255618)

[Risk management 16](#_Toc497255619)

[identified Potential risks (CO-2): 16](#_Toc497255620)

[mitigation strategies (CO-2, Co-3): 16](#_Toc497255621)

# Project Description

## Overview (CO-5):

Show Me is proposed to be a community driven application with the goal to enable researchers find related work (citations in the papers) and help fellow researchers by providing feedback on the citations.

In the field of research and discovery, it is often cumbersome for the researchers to browse through different research papers and keep track of relations between them. It takes a lot of time and effort to study the paper and explore the cited work, since the researchers may not understand the extent of relevancy of the cited papers without reading them. This may hinder the progress of their research work. We, as a team of Software Engineering graduate students, are proposing an elegant solution to this problem which will aim to address the difficulties faced by the researchers.

Show Me is a web application which will allow the users to browse and navigate through different categories of research papers and visualize relations among them as a network graph. Nodes in the network graph represents the papers and the edges represents the relations between the papers. Users can provide their feedback for the relations between papers in the form of comments. They also have the option to upvote or downvote the relations as per their understanding. The application is directed to the researchers as end users in their research area.

As a part of our final deliverable for this semester, we will submit Project Plan, Software Requirements Specification document and working prototype capturing the main functionality of our application. In the next semester, we will submit the final application with the functionalities as per the client requirements.

## Key Requirements (CO-5):

* Establish a relationship between technical papers in the form of a graph.

The core requirement for the project is to build a graph that relates research papers to each other’s and allows researchers to navigate freely amongst them.

* Host a web application

This graph is to be made available in a web application for users to navigate through. Thus, the next requirement is to build a web application that renders this graph and shows it to users in the web browser.

* Crowdsourcing the weightage of relationship.

We are also required to allow the users to provide feedback on the relationship between two papers based on the relevance of research. The application will use the feedback to assign weights to the relations between papers.

* User profile maintenance.

To make sure that bots do not affect the crowdsourcing, implement authentication using OAuth.

* Define and implement a crawling infrastructure.
* Enable users to search papers using a search box
* Provision for upvoting and downvoting the relation between two papers.

The upvote and downvote would allow users to weigh the relationship between two papers. Upvote would increase the weightage of the relation wherein the downvote would allow them to reduce the strength of the relation.

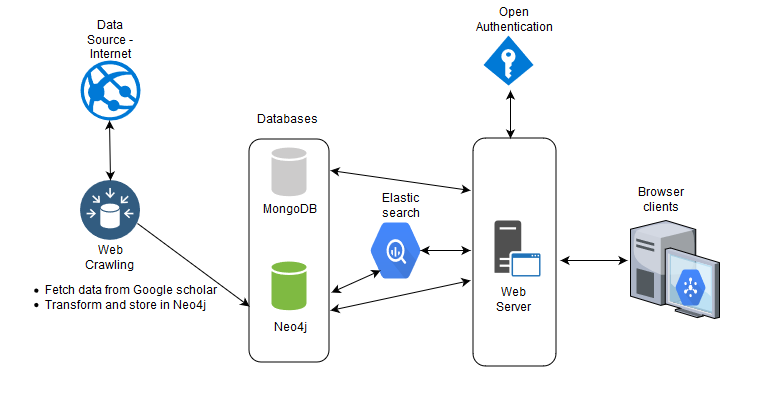
## Deliverables (CO-5, CO-6):

* Establish relationship among papers based on their references and crowdsourced rating in a graph database.
* Network graph visualization of papers and their interrelations based on citations.
* Enable users to provide feed-back (upvote/downvote, comment) for the relationship between papers.
* Crawl and scrap research paper websites (e.g. Google scholar) for data gathering.
* Create graph database comprising of relationship between research papers.

## Acronyms and abbreviations (CO-7):

|  |  |
| --- | --- |
| Abbreviation | Full Form |
| SPA | Single Page Application |
| URL | Universal Resource Locator |
| MVC | Model-View-Controller |
| UI | User Interface |

# Design and Architecture (CO-1, CO-3)



Description:

1. **Web Crawlers**

* The Web Crawlers will be used to crawl pages on the web or to fetch information about the papers and their citations.
* Google Scholar is a possible source to get data about research papers. The information about the papers and their interrelations will be stored in Neo4j database.

1. **Database**

* The application will rely on MongoDB and Neo4j for storage.
* MongoDB: MongoDB is a document-database that will be used to store user’s feedback (upvotes /downvotes and comments) on the relation between papers
* Neo4j: Neo4j is a graph database that will be used to store the papers and the citations (as relations) among the papers. The graph database aligns well with the requirement to persist information about papers and their interrelations.

1. **Web Server**

* A Node web server will contain the business logic to drive the application.
* All the CRUD (Create-Read-Update-Delete) operations are routed via the web server
* The web server in tandem with Elasticsearch will handle requests to search papers based on paper names, author names, etc.
* The web server would interact with MongoDB and Neo4j to retrieve information about papers and their relations and store feedback for the citations.
* The web-server will use the OAuth protocol to authenticate users via social applications (e.g. GitHub, LinkedIn, Facebook)
* Express framework will be used to implement the MVC pattern

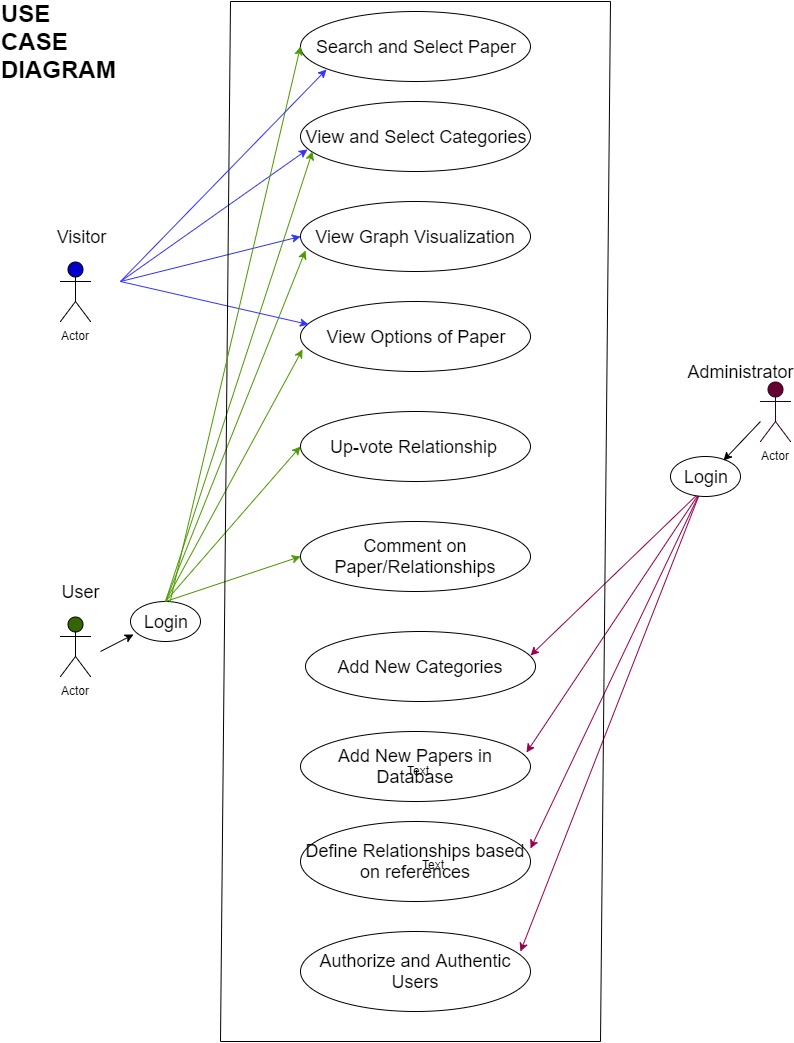
1. **Elastic Search**

* Component to facilitate search functionality all over the data.

1. **User Interface**

* The user interface will provide a means for the end user to search papers, visualize relations in the form of a weighted network graph (to depict the extent of relevancy of the citations) and provide feedback for the citations.
* The interface will comprise of a single page web application achieved by following the MVC pattern.
* Angular JS framework will be used to implement the MVC pattern. Bootstrap library will be used to make the application responsive across different screen sizes. Cytoscape.js library will be used for network visualizations.

**Use case Diagram**



**Description**:

The proposed system is a web application with three actors performing actions and interacting with the application.

**Visitor (Actor)** -  Visitors have limited access to the functionalities of the application. Visitors are not allowed to add likes and comments without logging into the application however they can view the likes and comments.

* Visitors can search any paper served by the application either by using search bar or manually browsing and selecting the paper from the categories dropdown.
* Visitors can view graph of the selected paper. Graph generated for the selected paper depicts the relationship between the selected papers and its adjacent nodes. Adjacent nodes are either the papers referenced in selected paper or the papers which has referred it.
* Visitor can select any related paper and view options for additional information about the paper. These options include the title of the paper, author of the paper and URL to access the paper.

**User (Actor)** - Users are the authenticated and authorized end user of the application. They have access to all the functionalities which are available to visitors of the application.

* Additionally, Users can upvote any relationship among the papers depicted by the graph by simply using the options menu.
* Users also have the authorization to add and view comments/likes from other users on the papers and relationships.

**Administrator (Actor)** -  Administrator is responsible for the maintenance of the application. In addition to all the accessibilities which are available to User and Visitor, Admin can also add new categories of the paper, add new papers in the database and define relationship between them.

* To maintain the integrity of the application, Admin can also restrict or unauthorize Users of the application for any malicious activities with the application.
* Admin also have access to remove and add comments on any of the paper/relationships.

# Implementation Strategy

Based on the project description and post requirement gathering and refinement phase, we identified following functional and nonfunctional requirements of the application.

Functional requirements are the core functionalities of the application. We have identified following major tasks to capture the functional requirements.

1. **Web crawling / Dataset hunting**

**Scope:**

* Initially, we decided to work with a manually created dataset containing fifty entries of papers in a specific domain.
* The information we recorded for these papers has following attributes for each paper.   paper-id, title, author, domain, referenced-from, link to document.
* Next, we plan to construct a crawler and let it run over a decent span of time to collect information regarding papers in multiple domains.
* So, we plan to construct a prototype with the manually collected data and later for the final project use the complete dataset created by our crawler.

**Tools:**

* We plan to use python. We chose python because some of our team members already have an experience in developing web crawlers in python.

**Expected Time:**

* Crawling part is highly dependent on the source from where we are going to crawl papers. Expected time: 1 Week/source

1. **Database development**

**Scope:**

* We have decided to use two databases for our application namely MongoDB and Neo4J.
* Neo4J is a graph database which will be used to store all the information regarding the papers we collect, along with the relationship between them.
* Papers will be represented as nodes and the relationships between papers as edges in the graph database.
* We will also store the strength of relationship between papers, in Neo4J. This strength is determined by number of upvotes and downvotes.
* The strength of the relationship will signify the relevance of the connected papers.
* MongoDB is used to store the user profiles and the user’s comments on the relationship between two papers.
* Currently, we are doing research to capture all the requirement in a single database Neo4J.

**Tools:**

* Neo4J
* MongoDB

**Expected Time:**

* We have constructed a basic structure in Neo4J using the manual dataset.

1. **User Authentication using Open Authentication (OAuth)**

**Scope:**

* OAuth will get the user profile information from either of the social networking sites.
* Whenever user tries to comment or upvote/downvote a link, OAuth will try to first get the user authenticated from a different provider and allow the user to proceed further.
* Web Server will map and route the requests accordingly.

**Tools:**

* We are planning to use express-oauth-server, node-oauth2-server library with express js.

**Expected Time:**

* Implementation and integration should not take more than 2 to 3 weeks of time.
* End to end testing can take 1- 2 weeks’ time depending on the progress of other components.

1. **Web server development**

**Scope:**

* The web server holds the business logic in this application.
* We are planning to use the MVC architecture for developing this application.
* It is a thick server application i.e. the business logic remains at server end.
* Web server is responsible for routing and invoking database queries from front end and fetching the result from backend to the UI.
* Web server would also maintain the user profiles and make sure there is not bot attack on the crowdsourced application.

**Tools:**

* We are using node based express js for developing the web server.
* We are also implementing Elasticsearch for enhancing the throughput of the application
* We are using OAuth to enable third party authentication

**Expected Time:**

* The web server has a large chunk of work and hence we have three dedicated server-side developers and two helping developers from database and front-end side**.**

1. **User Interface Design**

**Scope:**

* Any web application is incomplete without attractive and user-friendly user interface.
* Since this application is crowdsourced and expect high volume of user traffic, it is best advisable to create an intriguing human interface.
* We are planning to deploy best practices of the Human Computer Interface in the UI design
* UI developer is responsible for creating templates of all the web pages expected in the lifecycle of the proposed application.
* Moreover, UI development includes making the application as much intuitive as possible.

**Tools**:

* Most of the functional requirements of the application are incorporated in the User Interface design. Hence, we have decided to use Angular 4 framework.
* Angular offers MVC design pattern implementation which helps maintaining the modularity of the application.
* Also, it is possible achieve Single Page Application behavior for this application using the Angular framework, which can be used to enhance the responsiveness of the application.

**Expected Time:**

* As front-end development addresses the major part of the requirements, we have decided to have two dedicated and two supporting developers.
* It is expected to develop the user interface of the prototypal (beta) system in 4 to 5 weeks. The user interface of the actual system will be built on the prototypal UI and it is an incremental development process as new features and requirements come along.

1. **Visualizations**

**Scope:**

* Visualizations are one of the major functional requirement and feature of the application.
* The best visualization for the given requirements can be achieved using the network graph visualization since it can clearly depict the picture of the nodes (papers) and edges (relationships among papers).
* Moreover, the network graph will be interactive and will perform designed actions based on the user events. For instance, when user hover on any node, they can view the basic details about the paper. Similarly, when user clicks on the edge, they can upvote/downvote the relationship using the options menu.
* Visualization component of the application expects an input data from the web server, in the form of JSON object. This JSON object will be parsed and displayed in the form of graph.

**Tools:**

* We plan to use Cytoscape library for the development of the visualizations. Cytoscape is a javascript library that enables visualization of network graphs.
* Cytoscape provides high-level APIs that allows developers to create and manipulate network graphs. The library is open-source with good documentation. The library is actively maintained/ developed.
* The decision to select Cytoscape as visualization library was taken after performing comparative analysis between D3.js, Alchemy.js, and Cytoscape.js.

**Expected Time:**

* Once frontend user interface foundation is setup, visualization using the input JSON string from the web server will take tentatively one week assuming two developers working on this task.

Non-functional requirements are the parameters which can be used to judge the operation of the system, responsiveness, availability, reliability etc. In addition, nonfunctional requirements can be related to software engineering processes or any specific tools used during the development of the application. We have identified following non-functional requirements based on the software engineering process requirements and development tools.

1. Performance
2. Security
3. Continuous Integration
4. Version Control
5. Agile Process Management

## Schedule / Timeline (CO-2):

We have divided our entire project lifecycle into 5 different phases, dividing two semesters into 4 equal halves and one during the winter semester break. The deliverables and the timelines have been agreed upon with our sponsor in our last meeting. Please find below details related to milestones and deliverables.

1. **1st phase (Sep 20, 2017 - Oct 31st, 2017)**

This is the initial phase of our project. In this phase, we focused on understanding the problem description, gathering project requirements, discussed further among the team members and the sponsor further to refine our requirements, split up roles among ourselves, prepare relevant software artifacts collectively as a team, POC on each technology specific to our project by each of team member in the relevant technologies we are assigned.   Below are the deliverables that we plan to submit.

* Project Plan
* High Level Architecture Diagram
* Sample UI Mockups
* Proof of Concept
* Minutes of Meeting with sponsor

**2. 2nd Phase (Nov 1st, 2017 - Nov 31st, 2017)**

This is the second phase in our project lifecycle where we will be focused in building a working prototype and refine the documents further. The prototype will be built based on the core functionality requirement working on a static data. It will be presented with a limited capability addressing the main problem described. We will also finalize our other software documents with few minor modifications. Below are the deliverables that we plan to submit.

* Software Requirements Document
* Presentation on the working prototype to sponsor
* Presentation on our work for this project to the class so far

**3. 3rd Phase (Dec 1st, 2017 - Jan 10th, 2018)**

In this phase, we are planning to setup crawlers and parsers that would run for the entire phase which would collect data from different sources over the internet creating a large collection of database. Our application will run on this data and present it to the end users. Larger the data size, better it is for our application to run. So, probably we would continue to run them in 4th phase of our life cycle too. Few of the team members will be maintaining the scripts during the entire phase and make sure it is working fine. There are no deliverables in this phase.

**4. 4th Phase (Jan 10th, 2018 - Feb 25th, 2018)**

We will resume back our work in this phase on further enhancing the prototype to add more features to the application and continuous testing with the data captured. We will be in constant touch with the sponsor to get it reviewed and make any changes if needed. This process will go on in a circular fashion with all sorts of unit testing, integration testing, and system testing to make the model better. We will try to accommodate any last minute’s changes specified by the sponsor. As of deliverables, it will be mainly status reports and demo of our application.

**5. 5th Phase (Feb 26th,2018 - March 6th, 2018)**

This is the shortest and the final phase in the project lifecycle which will be utilized for final testing of our application, organize project software artifacts, and submit our project to SCORE. We have a buffer period of 8 to 9 days to fix any bugs if present. The key deliverables will be the project folder with the below artifacts:

* Executable of our application
* Software Requirements Specification doc
* Software Design Document doc
* Software Testing Document
* READ ME

## Required Hardware (CO-2):

We are not planning to use any hardware for this project so far.

## THird party content (CO-2):

We are not planning to use any third-party content for this project so far.

## Quality (CO-2):

Quality Goals

* **Code Reuse**

Make as much code reuse as possible by dividing major tasks into small, independent functions which can be reuse anywhere in the code, as per requirement. For any related requirements, we plan to develop APIs which can be imported in other programs.

* **Dependency Management**

Use of MEAN (Mongo, Express, Angular, and Node) technology stack for development, enables us with simple mechanism for dependency management. All the external node modules can be listed in the package.json file.

* **Modular Development-**

Majority of the project focuses on the front-end development and visualization. Hence, we have chosen Angular framework because of its component based modeling.

* **Easy Maintenance**

Employ best coding practices, comments for each piece of code developed. Maintain proper documentation for each component of the web application.

* **Robust Architecture-**

MVC architecture implementation allows to modularize the code and distribute responsibilities among different components

* **Reliability**

Application should work as per the requirements specification. The papers and their citations should be correctly displayed as network graph [of nodes and links]. User feedback in the form of upvotes/downvotes and comment should be correctly reflected in the system. Non-Authenticated and Non-Authorized users should not be able to change the state of the system [by providing anonymous feedback]. A suite of unit-tests will be run to check if the application design aligns with requirements and handles failures gracefully.

Quality Metrics

* **Response time**

Calculate average response time for each functionality under benchmark conditions

* **Intuitiveness:**

Take opinion survey of the end users on the beta version of the application before release

* **Unit tests**

Write unit tests for each functionality developed across the applications. The tests will be used to check if the application is designed to meet the requirements specification.

* **Documentation**

Code should follow proper conventions as per the programming language. Code should be well be structured and should include appropriate comments.

* **Load Balancing (Future scope)**

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.

## Other Special considerations (CO-7, Co-3):

There are no special considerations to be made for our project.

# process

## Process Description and justification (Co-2)

We are using agile based Scrum methodology for the project. The team has chosen the method because of the dynamic nature of the project. Most members in the team have worked in scrum based teams previously which allows them to easily adapt to the process. The iterative process allows to start with a Proof-of-Concept and add features incrementally to enable a seamless delivery of the product.

## Tools (CO-2):

* **Taiga** - We are using Taiga for our Agile SCRUM methodology to keep track of our progress and to assign tasks to team members.
* **GitHub** - We are using GitHub for our code repository and version control software.
* **GitLab** - We will be using it for continuous integration tool.
* **Slack -** We are using Slack channel for team communications over chat.

## Roles and Responsibilities (CO-2):

We have divided the roles and responsibilities in three different teams based on our architecture which are Frontend UI team, Middleware team, and Backend Team. Frontend team manages the responsibility of designing and developing the look and feel of the application with a greater user experience. The Middleware team is responsible for designing and developing the application server which will serve the requests from the frontend, process it and send it to respective backend component. The backend team takes care of storing the data in backend DBs, by designing to schema and architecture inside the DB and storing the crawled data based on the relation among them. They are also responsible for crawling data from different sources and store it into the Databases.

Below are the individual roles assigned to each team member:

1. Pushkar Ladhe (Middleware) - Server-side web development and software process management.
2. Rajat Sinha(Backend) - Graph Database development and management and minute of meeting writer.
3. Sachin Magar(Backend) - Graph Database development, web crawler development and Infrastructure setup/maintenance.
4. Pranjal Karankar(Frontend) - User interface design and development, Infrastructure setup / maintenance.
5. Chiraag Subramanian(Frontend) - User interface development and high-level system design.
6. Abhishek Dutta(Middleware) - Server-side Development, integration of OAuth and managing infrastructure.
7. Apoorv Khairnar(Middleware) -  Server-Side Development, integration of Elasticsearch and Neo4j with the middleware

## Location of Project Artifacts (CO-2):

* We are using GitHub repository for this project. We chose GitHub since we all have some initial experience using it for version control. It also helps in efficient collaboration and continuous integration in fast paced environment. This project will enhance our knowledge and experience with GitHub.
* We are planning to send this project for Score competition, hence we have hosted the project in a private repository.

## Sponsor communications (CO-7):

* We decided to have skype call once a month with sponsor. We discuss our progress till date with sponsor in this meeting. Sponsor then provides his feedback on the content as well as discuss the areas which can be improved. We also use this meeting to discuss roadblocks and ways to resolve them. This meeting is crucial for question answer session. We also have a slack channel where we post quick updates for the sponsor. If we need to convey/ask something with high priority, we communicate with sponsor via email.

# Risk management

## identified Potential risks (CO-2):

* Scarcity of data is the potential risk for this project. From our discussion with the sponsor, we are planning to collect the data using web scrapers. One problem with using scrapers is some websites don’t allow scraping on them. We also don’t have a way to source for collecting the data, so a lot of research is needed in finding the source and validating if it suffice the purpose. Most of the website which allow scraping doesn’t have a large data. Web scraping is time consuming process and considering the volume of data we need for this project, we will have to run our scrapers for longer times. For this, we will need a dedicated system where we can run our programs. This system must be fault tolerant to ensure the programs run without interruption, so we cannot use our own systems.
* Secondly, we don’t have a dedicated and stable infrastructure where we can host our infrastructure properly which might affect the overall project performance. We might have to compromise with using shared resources which doesn’t guarantee the availability of our service  24/7

## mitigation strategies (CO-2, Co-3):

We are researching on the sources to get the data from. Even if there is no single source where we can get majority of data from. We can divide the crawling process in smaller chunks for multiple sources. This will slow down the data gathering process which will eventually push the completion date further. We will also have to dedicate an additional resource from our team for crawling, so it will slow down the progress in other components of the project.

If we don’t have a robust system for hosting the crawlers, we will have to rely on our own systems where we may face the issues of inadvertent failures. This may slow down the crawling process