ShowMe: Show Me Related Work

Summary Report

Kishan Kumar Ganguly

Eusha Kadir

Aquib Azmain

Moumita Asad

Rafed Muhammad Yasir

January 15, 2018

Contents

List of Figures

1 Introduction

SCORE 2018 is a worldwide competition organized by the 40th International Conference on Software Engineering (ICSE) for promoting Software Engineering practice. Student teams are required to design and implement one among the several projects given, proving their skills in software engineering.

Our team consists of a masters student and four undergrad students from the Institute of Information Technology, University of Dhaka. The task of the masters student was to administer the students during the development process.

We have selected "ShowMe: Show me related work", which is a project for finding related research papers. We chose it as our project because we saw in it the possibilities immediately. At our institute we have a group of students and teachers very much dedicated to research work. We have seen their efforts in doing literature reviews by reading many research papers at a time. We thought that doing this project would help them to some extent, making the task of exploring papers a little easier.

It contains requirement analysis, architectural design, tradeoffs and choices, management plan, source code management, time management, implementation and platform choices, testing methods, challenges faced and lessons learned.

2 Requirement Analysis

2.1 Overview

ShowMe is a web application that will help the research community to find related work and understand relationship among them.

Whenever a user searches a paper, a list of relevant papers will be shown by taking data from Google Scholar. A user may also make an advanced search by specifying the following criteria:

- Articles with all of the words
- Articles with the exact phrase
- Articles with at least one of the words
- Articles without the words
- Words occur in the title/anywhere in the article
- Articles authored by
- Articles published in
- Articles dated between

When an item on the list is clicked, a citation graph of the paper will be shown. Each node in the graph will represent a scholarly information. If an article cites another article, a directed edge to the cited article will be shown. Hovering mouse on a node will show following information of the article:

- title
- author
- journal
- pages
- year

Clicking on a node, the pdf of the publication will be shown. If the pdf is not available the user will be taken to the site where the paper was found.
Clicking on an edge, user can view text snippets of how one paper cites another. Edges will be color coded according to relationship among papers. A user can rate an edge to mark the relatedness between two papers. A user can rate an edge out of 5 according to strong or weak relationship among the papers. Only a logged in user can rate an edge. A user may sign up using an email address or Google account.

The generated graph can be filtered by searching. A search key can be applied on the title, author, year and journal. The user can select if the filter applied will remove nodes from the graph or apply a different color. Users can view recently made search queries also.

2.2 Assumption

- Metadata of papers will be available (title, author, journal, pages, year of the article).
- Number of requests to be made to Google Scholar will not exceed the request limit of Google Scholar.

2.3 Scope

- We will use information from the search results provided by Google Scholar.
- Reference lists will be extracted from PDFs. However, not all PDFs are available as some are paid. We will extract from PDFs that are shown on Google Scholar.

2.4 Requirement Specification

2.4.1. Use cases

The following use cases show the list of events that take place between the users and the system to accomplish the goal.
Level 0: ShowMe
Primary actors: Unauthenticated user, Authenticated user.

Goal in context: The diagram represents the whole ShowMe application.

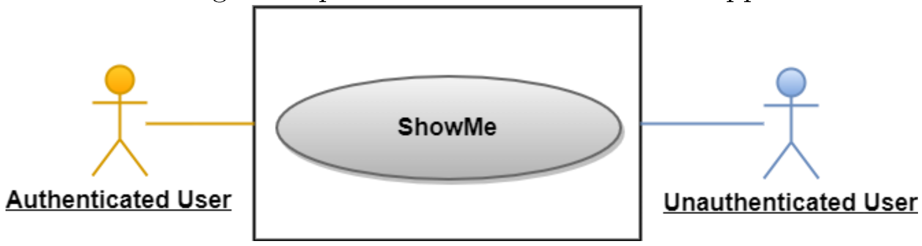


Figure 1: Use case level 0 - ShowMe

Level 1: Modules of ShowMe
Primary actors: Unauthenticated user, Authenticated user.
Goal in context: The diagram shows all the modules of the ShowMe.

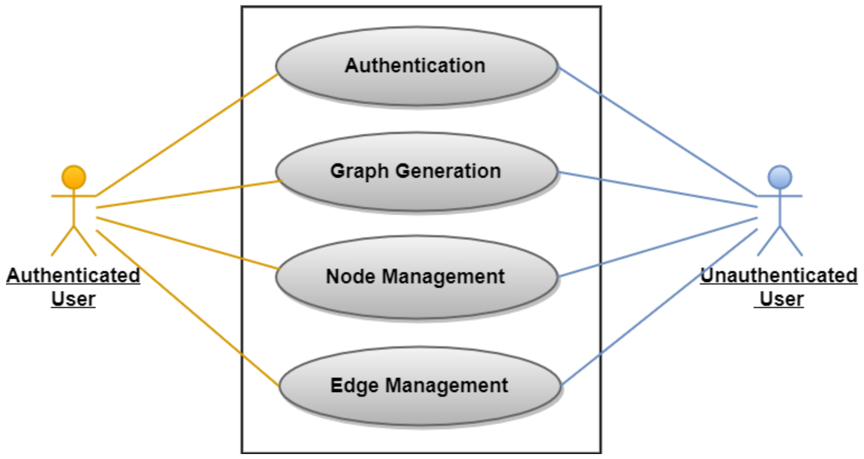


Figure 2: Use case Level 1 - Modules of ShowMe

There are four modules in ShowMe application.
Level 1.1: Authentication
Level 1.2: Graph Generation
Level 1.3: Node Management
Level 1.4: Edge Management

Level 1.1: Authentication
Primary actors: Unauthenticated user, Authenticated user.
Goal in context: The diagram refers to the details of the Authentication module of level 1.

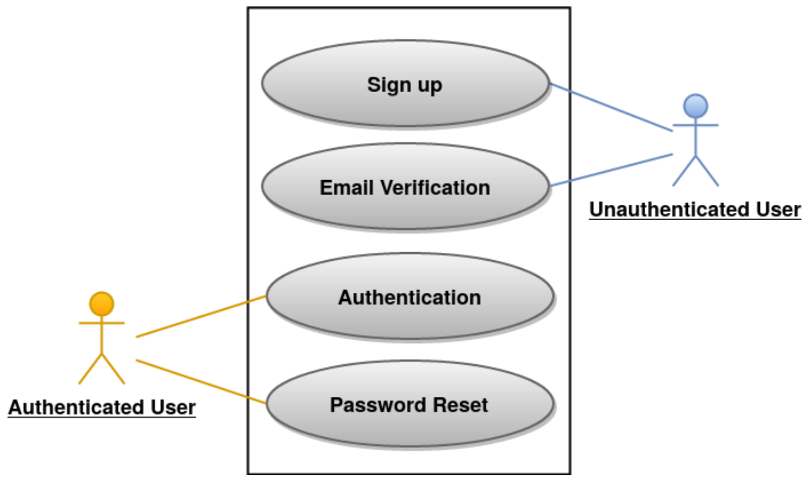


Figure 3: Use case Level 1.1 - Authentication

Actions and Replies

A1: Unauthenticated user enters email address and password to register.
R1: System checks whether any personal account exists under the same email or not. If the request is valid, the applicant will receive a confirmation email.
A2: Authenticated user enters email address and password or use google account to authenticate.
R2: He/she is allowed to enter into the system upon entering correct credentials.
A3: Authenticated user wants to reset password.
R3: System allows to reset the password through email.

Level 1.2: Graph Generation

Primary actors: Unauthenticated user, Authenticated user.
Goal in context: The diagram refers to the details of the Graph Generation module of level 1.

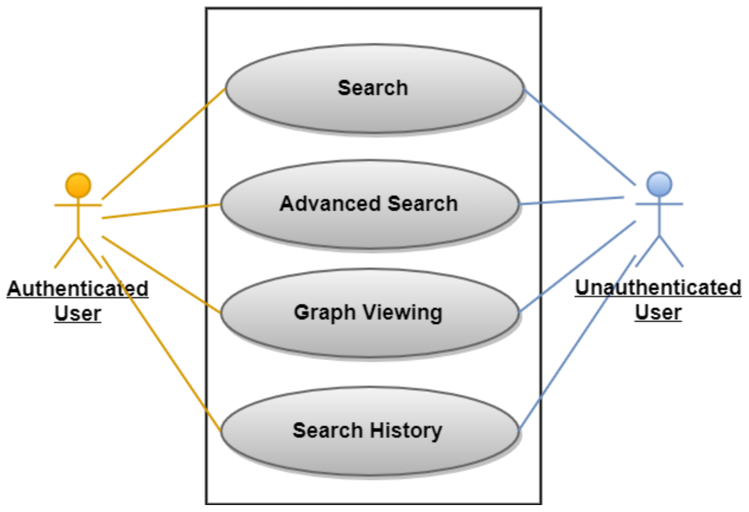


Figure 4: Use case Lavel 1.2 - Graph Generation

Actions and Replies

A1: Unauthenticated user or Authenticated user searches for papers with or without advanced search.

R1: System provides result by taking data from Google Scholar. This result shows a list of related papers.

A2: Unauthenticated user or Authenticated user clicks on a paper from the list.

R2: System generates citation graph of the paper.

A3: Unauthenticated user or Authenticated user wants to view recently searched papers.

R3: System provides the list of recently searched papers.

Level 1.3: Node Management

Primary actors: Unauthenticated user, Authenticated user.

Goal in context: The diagram refers to the details of the node management module of level 1.

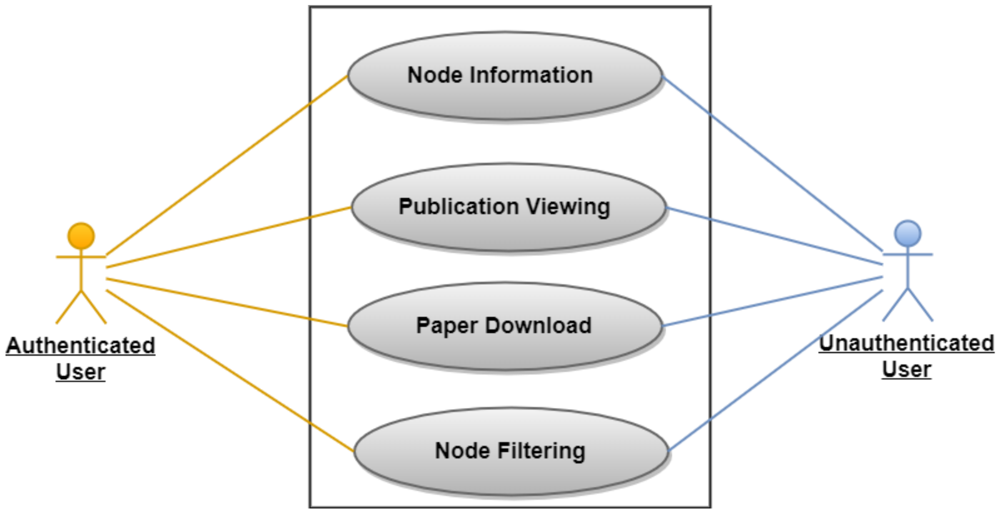


Figure 5: Use case Level 1.3 - Node Management

Actions and Replies

A1: User hovers mouse on a node.

R1: information about a pdf (title, author, journal, pages, year) will be shown.

A2: User clicks on a node.

R2: The pdf of the publication will be shown. If the pdf is unavailable, the user will be taken to the site where the paper was found.

A3: User clicks on downloading paper.

R3: Paper will be downloaded.

A4: User wants to filter graph.

R4: Graph is filtered upon selected criteria.

Level 1.4: Edge Management

Primary actors: Unauthenticated user, Authenticated user.

Goal in context: The diagram refers to the details of the Edge Management module of level 1.

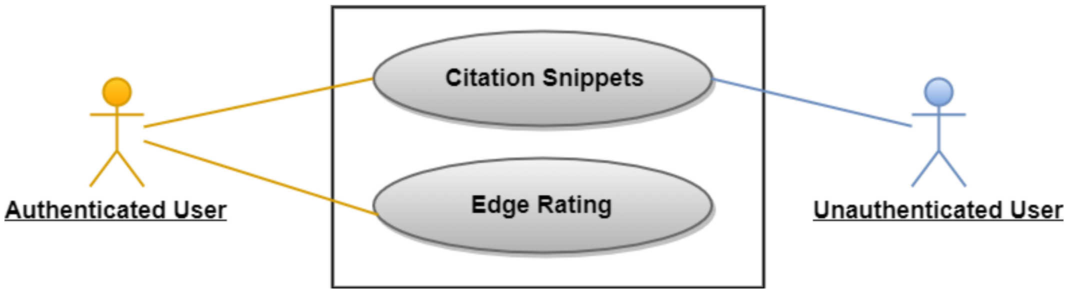


Figure 6: Use case Level 1.4 - Edge Management

Actions and Replies

A1: Unauthenticated user or Authenticated user wants to view text snippets of citations.

R1: System shows citation snippets.

A2: Authenticated users rate edges.

R2: System stores rating.

2.4.2. Activity Diagrams

Activity diagram represents the complete flow of a particular use case.

Figure 7 represents the activity diagram of level 1.1 Authentication module.

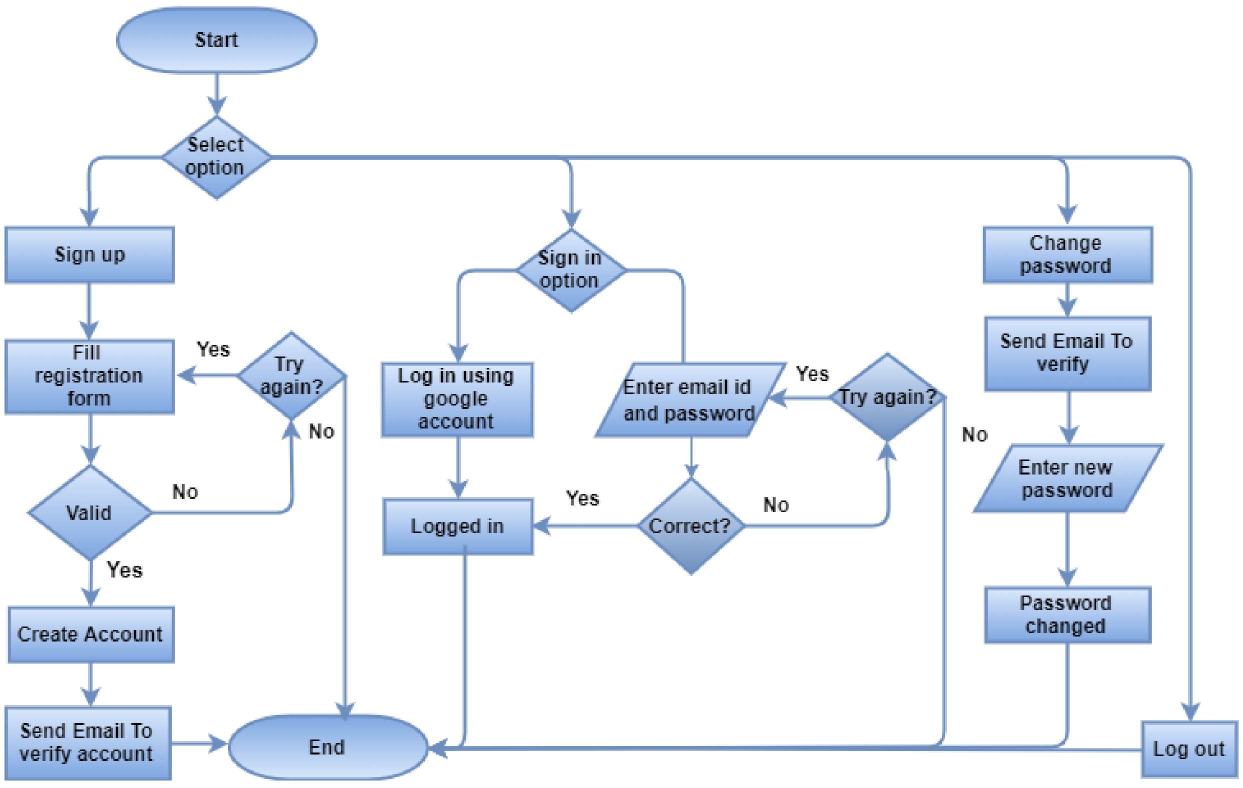


Figure 7: Activity diagram of Authentication module

Figure 8 represents the activity diagram of level 1.2: Graph Generation module.

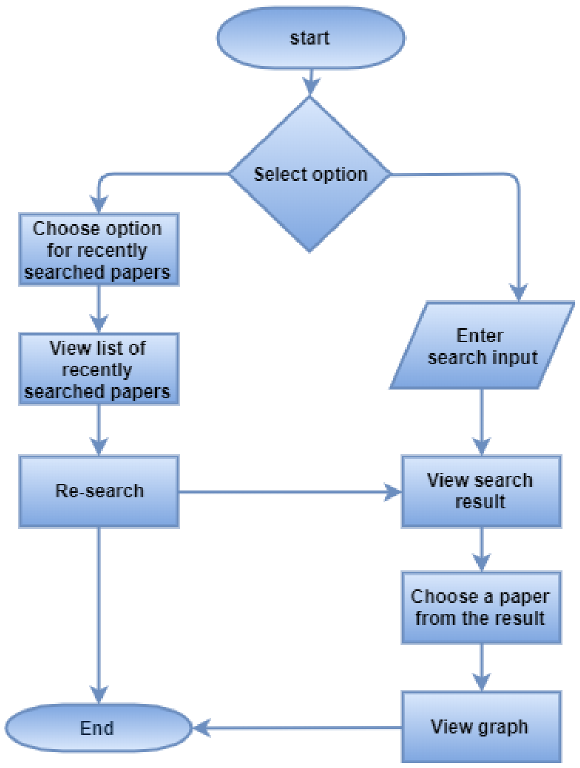

start
Select option
Choose option for recently searched papers
View list of recently searched papers
Re-search
Enter search input
View search result
Choose a paper from the result
View graph
End

Figure 8: Activity diagram of Graph Generation module

Figure 9 represents the activity diagram of level 1.3: Node Management module.

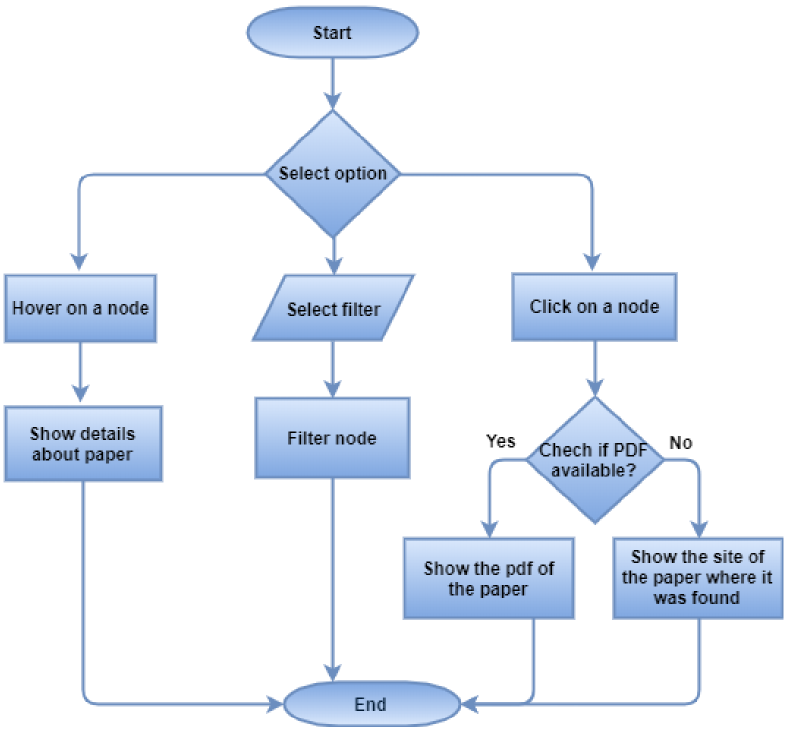


Figure 9: Activity diagram of Node Management module

Figure 10 represents the activity diagram of level 1.4: Edge Management module.

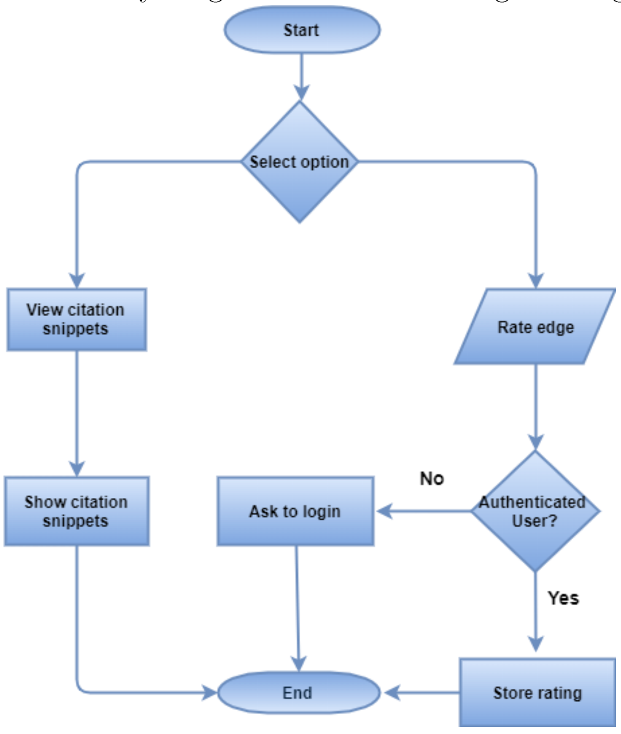


Figure 10: Activity diagram of Edge Management module

2.4.3. Entity Relationship Diagram

Figure 11 shows Entity Relationship diagram of showme project. For simplicity only primary keys are shown in the ER diagram.

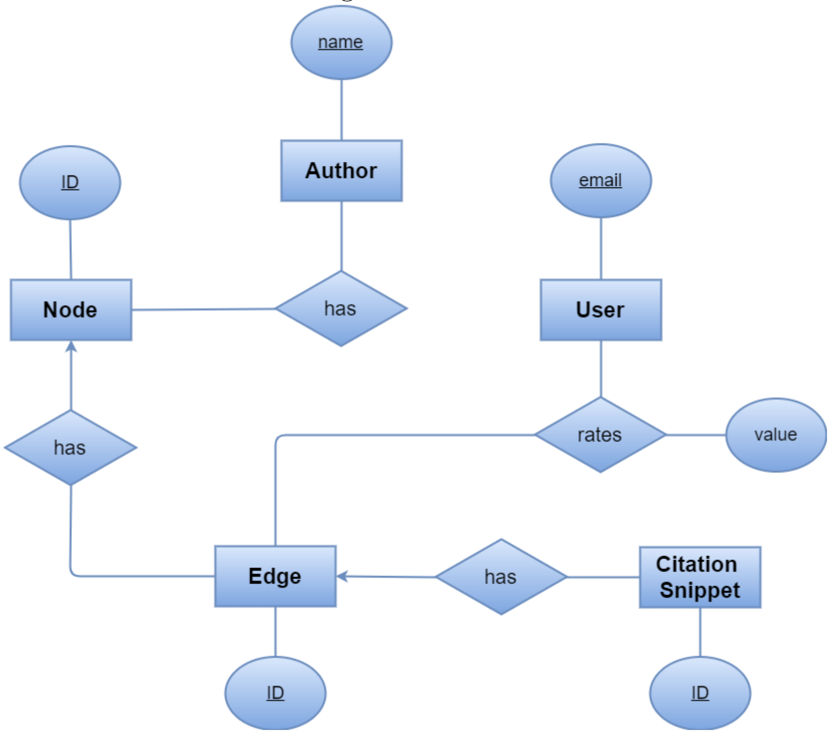


Figure 11: Entity Relationship Diagram of ShowMe

2.4.4. Schema Tables

We have derived the following tables from the ER diagram (Figure :11)

User

Attribute	Type	Size
email	varchar	50
password	varchar	50
oauth_provider	varchar	20
oauth_uid	varchar	50
fname	varchar	30
lname	varchar	30
created	datetime	

Node

Attribute	Type	Size
ID	varchar	30
Title	varchar	120
journal	varchar	100
volume	varchar	10
pages	varchar	10
year	year	
pdfLink	varchar	100

Author

Attribute	Type	Size
name	varchar	50
Node_ID	varchar	30

Edge

Attribute	Type	Size
ID	varchar	30
SourceNode_ID	varchar	30
TargetNode_ID	varchar	30

Citation Snippet

Attribute	Type	Size
ID	varchar	30
Edge_ID	varchar	50
text	varchar	300

Rating

Attribute	Type	Size
Email_ID	varchar	30
Edge_ID	varchar	30
value	tinyint	

3 Architectural Design

Our application is based on the classic 3-tier architecture. The software is divided into a presentation layer, logic layer and a persistence layer. [1, 2]

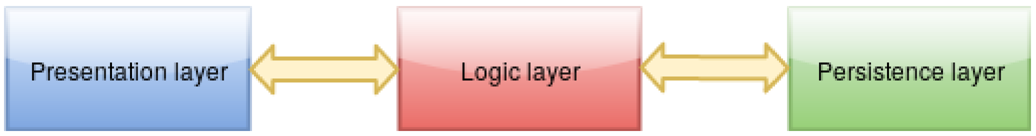


Figure 12: 3-tier Architecture of ShowMe

The presentation layer is where all the user interactions take place. The presentation layer communicates with the logic layer. Our logic layer is a REST API that provides URL endpoints for the presentation layer to communicate. Through the logic layer user inputs are processed and information are returned to the client. The logic layer communicates with the persistence layer to store and retrieve data from the system.

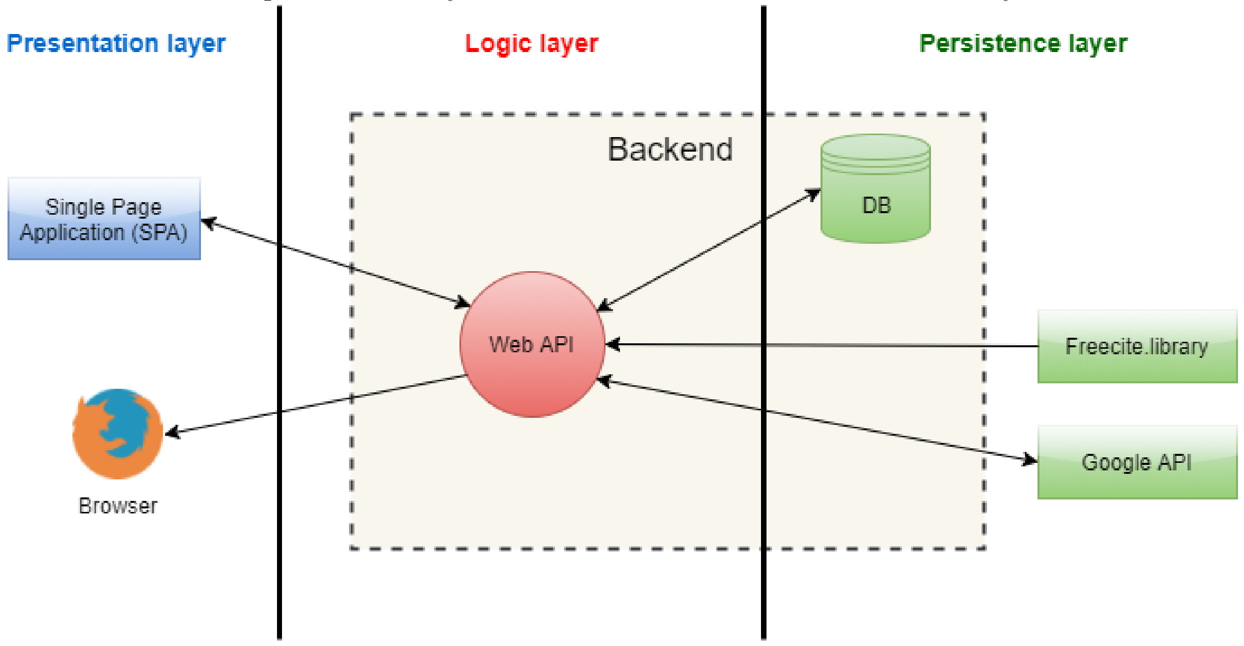


Figure 13: Architectural design of ShowMe

When the client requests the server for the application a single page application will be loaded. Later on all requests to the server will be AJAX calls. The server will reply in JSON. The server stores and retrieves data from the database server. We will also need some external data provided as a service by the freecite.library API and the Google

API. The freecite library provides with parsed data of a citation and the Google API helps logging in with a Google account.

3.1 Choices and tradeoffs

Initially we thought of scraping google scholar for a paper's metadata and citation data. But using this approach would mean we had to make huge number of requests to google scholar. As google scholar disallows bots we would eventually end up being blocked. To minimize the number of requests sent to Google Scholar we changed our data extraction approach by using both Google Scholar and parsing PDFs. Search results are made using Google Scholar and citation graph is made parsing PDFs.

4 Management Plan

4.1 Team Member

Serial No	Member Name	Email Address	Role	Area of Expertise
1	Kishan Kmar Gaguly	bsse0505@iit.du.ac.bd	Scrum Master	Software Architecture, machine learning and self-adaptive systems
2	Eusha Kadir	bsse0708@iit.du.ac.bd	Developer	Machine Learning and Artificial Intelligence
3	Aquib Azmain	bsse0718@iit.du.ac.bd	Designer	Designing and Front-end Development
4	Moumita Asad	bsse0731@iit.du.ac.bd	Developer	Software Requirement Analysis
5	Rafed Muhammad Yasir	bsse0733@iit.du.ac.bd	Developer	Web Development and Networking

4.2 Team Coordination

For team management, we followed Scrum. For agile software development, Scrum is a popular methodology. It is chosen because it is lightweight and simple to understand.

The roles are chosen according to the knowledge of the members. Our Scrum Master is Kishan Kumar Ganguly. In our development team, there are four members: Rafed Muhammad Yasir, Moumita Asad, Eusha Kadir and Aquib Azmain.

For working together on a document we used Google Docs [3]. This is a collaborative platform to share and edit our document among team members. Our diagrams are made with draw.io [4]. In draw.io diagrams can be made collaboratively.

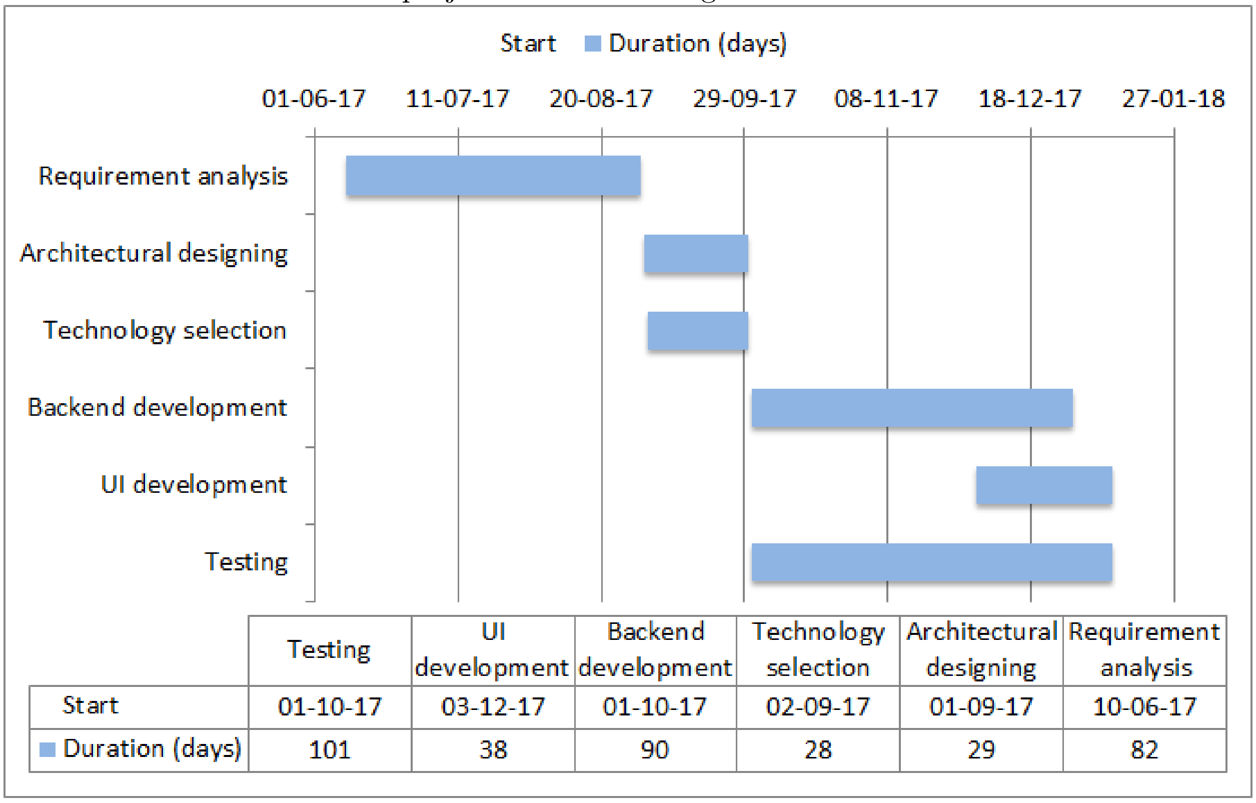
We have used Trello [5] project management application to assign tasks to member. In every scrum meeting, progress of the tasks were discussed. When tasks were completed, they were marked as complete and new tasks were assigned.

5 Source Code Management

To work collaboratively we used Git to manage our project. Our project is hosted at Github [6]. We worked in a distributed manner and each of us pushed our code to the remote to keep everyone updated. Git also helped managing versions and different branches of our project.

6 Time Management

The time distribution of our project is shown in Figure 14.



	Testing	UI development	Backend development	Technology selection	Architectural designing	Requirement analysis
Start	01-10-17	03-12-17	01-10-17	02-09-17	01-09-17	10-06-17
■ Duration (days)	101	38	90	28	29	82

Figure 14: Time distribution of ShowMe

7 Implementation

The whole implementation has three parts- the database for storing data, the web api for data communication and the UI for presentation.

For the database we used MySQL RDBMS (Relational Database Management System). MySQL was chosen for its simplicity and ease of use. It is easy to install, use, as well as scalable and manageable. Apart from our own database we have used the freecite API which is a free citation parser that parses document citations into fielded data. For parsing, it uses the conditional random fields model.

For serving the client we made a web API that receives client requests and serves JSON data (search results, citation data). This web API is written in Python. The reason we chose Python is that during the initial phase of our project we had to try out many prototypes before actually selecting one for the project. Python is very suitable for designing prototypes as a working implementation can be produced in minimal code. Not only is it concise it's also easy for others to read.

We also used the Flask [7] framework for making the web API. Flask is a web micro-framework for Python. We considered Django but realized it would be too heavy for our project. Flask is a lot more lightweight framework and is easier to use.

Our front end is built using Angular 4 [8]. Angular is a framework for making single page applications. It also allows the testing of front end components. For building the citation graph we used cytoscape.js [9] library. There were quite a handful of graph libraries to choose from but this library showed very good backward compatibility with older browsers and hence it was preferred.

8 User Interface

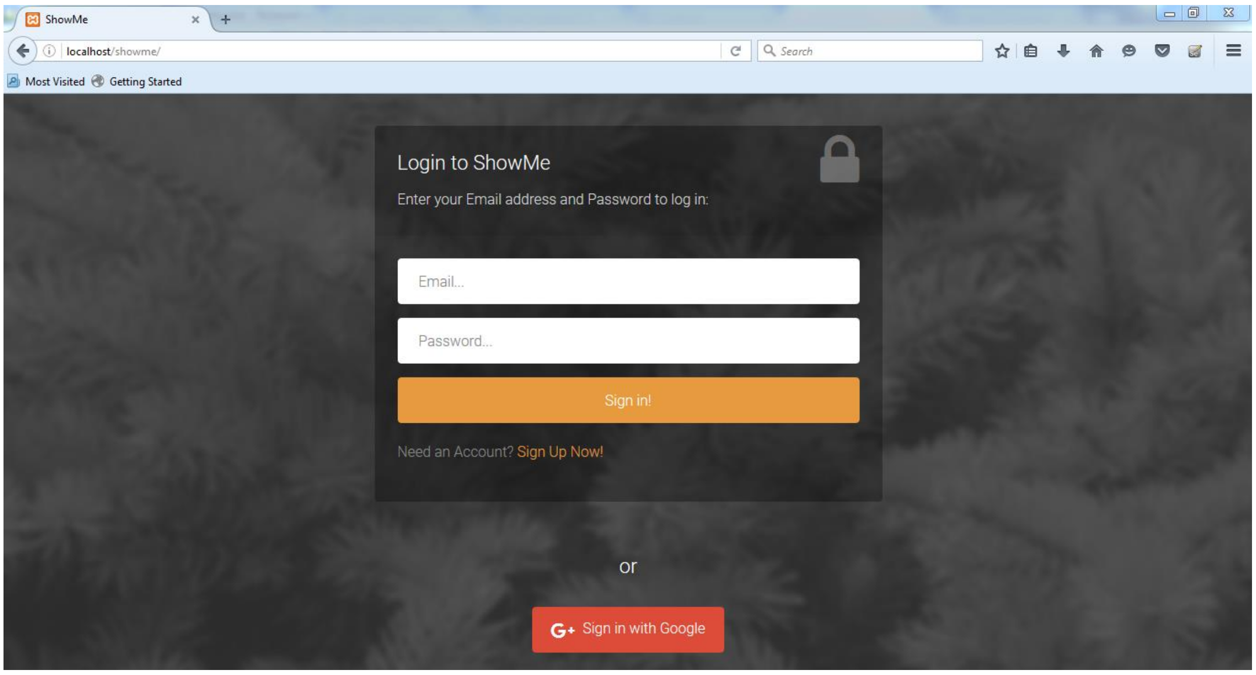
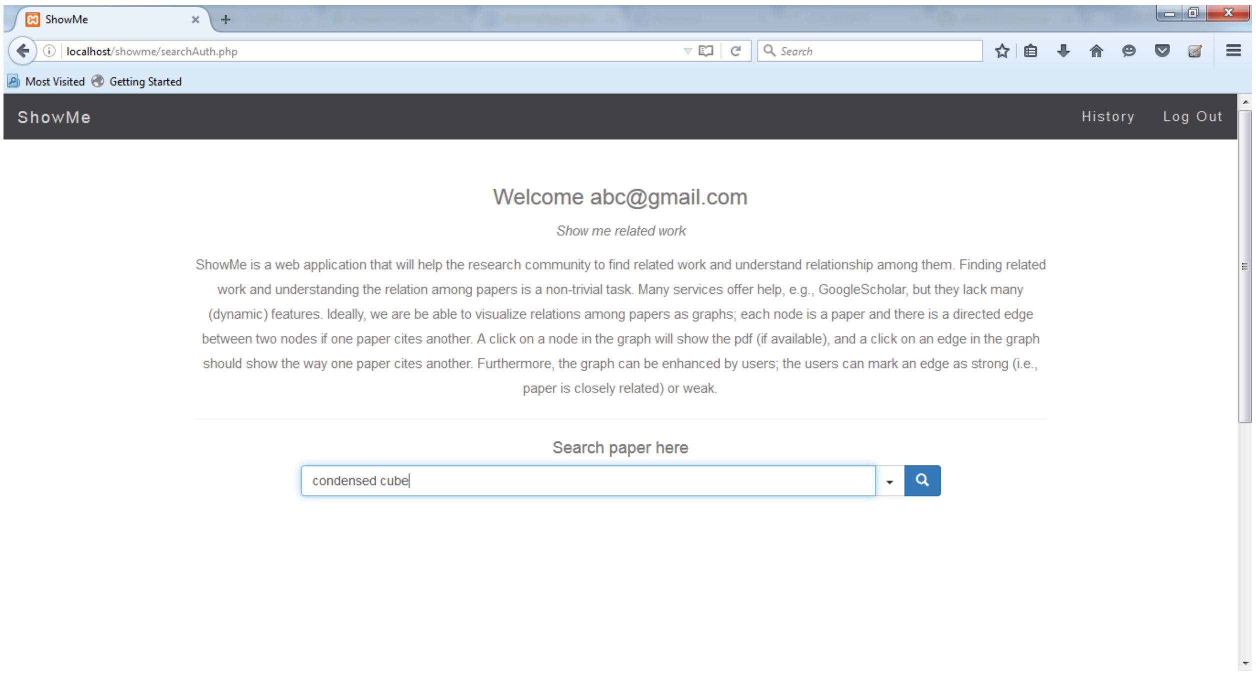


Figure 15: Screenshot of Log in page



Welcome abc@gmail.com

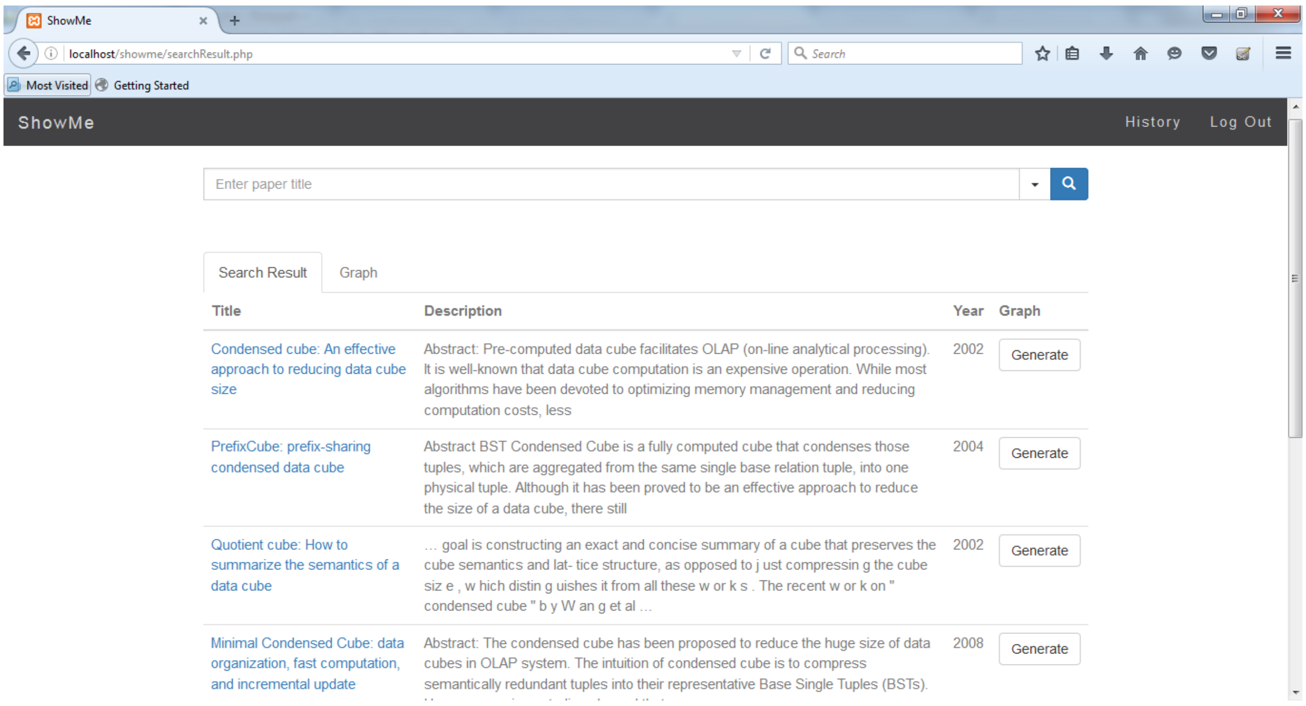
Show me related work

ShowMe is a web application that will help the research community to find related work and understand relationship among them. Finding related work and understanding the relation among papers is a non-trivial task. Many services offer help, e.g., GoogleScholar, but they lack many (dynamic) features. Ideally, we are be able to visualize relations among papers as graphs; each node is a paper and there is a directed edge between two nodes if one paper cites another. A click on a node in the graph will show the pdf (if available), and a click on an edge in the graph should show the way one paper cites another. Furthermore, the graph can be enhanced by users; the users can mark an edge as strong (i.e., paper is closely related) or weak.

Search paper here

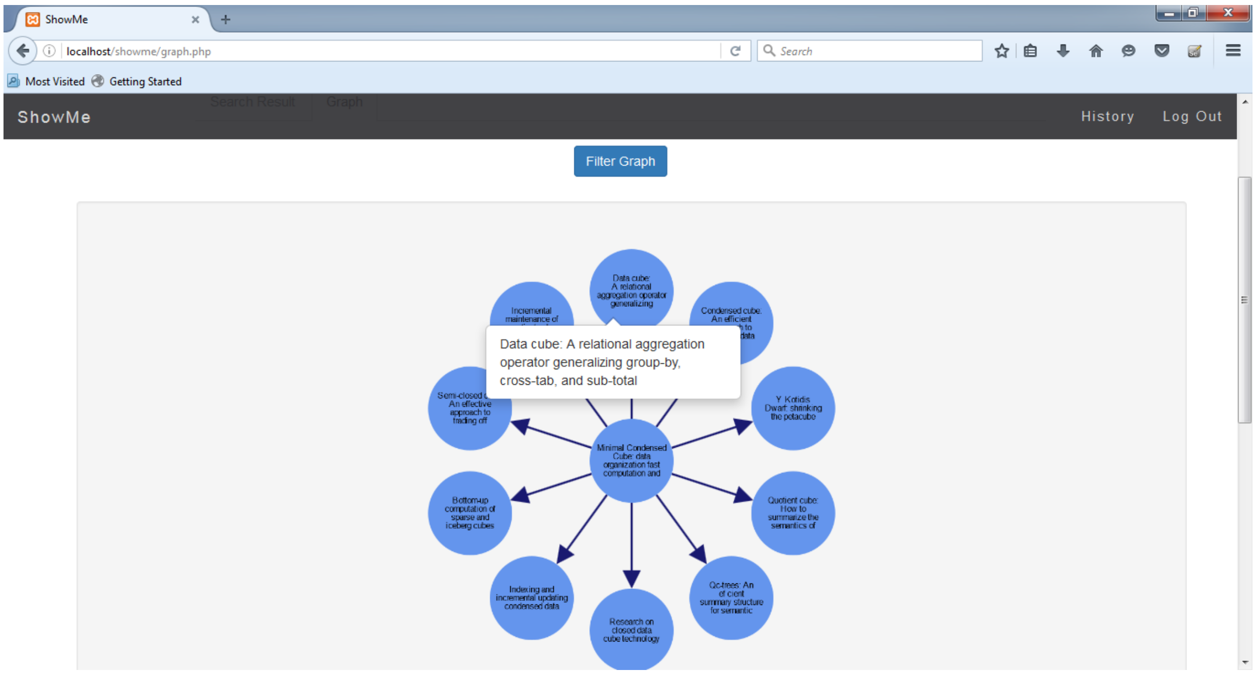
condensed cube

Figure 16: Screenshot of Search Paper page



Title	Description	Year	Graph
Condensed cube: An effective approach to reducing data cube size	Abstract: Pre-computed data cube facilitates OLAP (on-line analytical processing). It is well-known that data cube computation is an expensive operation. While most algorithms have been devoted to optimizing memory management and reducing computation costs, less	2002	Generate
PrefixCube: prefix-sharing condensed data cube	Abstract BST Condensed Cube is a fully computed cube that condenses those tuples, which are aggregated from the same single base relation tuple, into one physical tuple. Although it has been proved to be an effective approach to reduce the size of a data cube, there still	2004	Generate
Quotient cube: How to summarize the semantics of a data cube	… goal is constructing an exact and concise summary of a cube that preserves the cube semantics and lat- tice structure, as opposed to j ust compressin g the cube siz e , w hich distin g uishes it from all these w or k s . The recent w or k on " condensed cube " b y W an g et al …	2002	Generate
Minimal Condensed Cube: data organization, fast computation, and incremental update	Abstract: The condensed cube has been proposed to reduce the huge size of data cubes in OLAP system. The intuition of condensed cube is to compress semantically redundant tuples into their representative Base Single Tuples (BSTs).	2008	Generate

Figure 17: Screenshot of Search Result page


ShowMe
localhost/showme/graph.php
Search
Most Visited
Getting Started
ShowMe
Search Result
Graph
History
Log Out
Filter Graph
Data cube: A relational aggregation operator generalizing
Incremental maintenance of
Condensed cube: An efficient
Semi-closed An effective approach to trading off
Y. Kotidis Dwarf: shrinking the petacube
Minimal Condensed Cube: data organization fast computation and
Bottom-up computation of sparse and iceberg cubes
Quotient cube: How to summarize the semantics of
Indexing and incremental updating condensed data
Research on closed data cube technology
Qc-trees: An of cient summary structure for semantic

Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total

Figure 18: Screenshot of Graph page

9 Verification and Validation Activities

Verification checks whether we are building the product right where validation checks whether we are building the right product. We first developed a verification and validation plan which consisted of standards, schedules, resource summary, techniques and methods etc. As part of the verification, we conducted reviews and manual inspection on each of the work products. The requirement, design documents, codes and test cases were thoroughly reviewed. In the documents, we checked for inconsistencies, logical errors, tractability, understandability, completeness and the level of detail. For code reviews, we performed informal walkthroughs. To perform code reviews, apart from manual inspection, we also took assistance from a static analysis tool called Pylint.

We further performed three levels of validation such as unit, integration and system testing. We wrote unit test case plans where we mentioned the test case id, method/ function, input, expected output and priority of the test case. We generated most of the test cases by generating control flow graphs from the source code and considering inputs that lead to traversing a valid independent path. Some of the unit test cases are generated by manually inspecting the code. For unit testing, we used unittest for python and for AngularJS. We integrated Travis CI to our project repository at Github and as a result test cases are run at each integration to our codebase. This ensures that each new piece of code added does not break the codebase. Thus healthier software is developed. We further performed black box testing where we mostly considered boundary value analysis and equivalence class testing techniques. We also planned for load testing which we will perform using Apache JMeter. We are also preparing acceptance test cases according to the requirements which we will use to test whether the product fits the user. We have also primarily collected feedback on the Unser Interface (UI) from the researchers of the Institute of Information Technology, University of Dhaka and working to furnish the UI.

10 Challenges and Lessons Learned

10.1 Challenges

The project was full of challenges. Some of those challenges are:

- Google Scholar disallows bots. As a result, limited queries has to be made to avoid blocking. So extracting data from Google Scholar was challenging.
- Because of the rate limit of google scholar, designing was also very challenging. We had to try out many different ways before actually selecting one that would be suitable for our project (see Choices and tradeoffs for details).
- Extracting reference list from pdf was also challenging as there is no fixed format of referencing.
- Extracting individual reference was also difficult as we could not locate the starting and ending point of a reference with high accuracy. We had to design an algorithm for proper detection.

10.2 Lessons Learned

- We were inexperienced regarding research papers as most of us have not done any research work yet. Through this project we came to know a lot about research paper which will help us in future.
- Working with pdfs was a new experience for us.
- We have performed software testing before but integrating with Travis CI was something we had not done before.
- We have performed project management using Trello and using Trello was completely new for us.
- This was our first big project in python. Through this project we have learnt much about python itself and the huge number of libraries it offers including the testing frameworks.

11 Conclusion

We are very happy working with this project and it was a new learning experience for us. We have tried to put the best software engineering practices into play and implemented as many features as we could. Before we started writing the code we thoroughly went through the design phases. Our design phases is reflected in the requirements engineering and architecture design. There is still much room for expansion of this project.

- At present we are generating citation graph by extracting reference list from pdfs only. Our coverage can be expanded by supporting extracting reference list from websites like IEEE, Research Gate, SpringerLink and others.
- Currently we are showing relationship between two papers based on user rating and measuring textual similarity. Following relationships can be developed also:
    - A particular university cites which university the most.
    - Which authors work more together.
- PDF can be added to drive if google authentication is used.

We would like to thank Professor Milos Gligoric who helped us time to time by replying to our queries regarding this project. We would also like to thank Professor Christine Julien who guided us during the time of registration.

References

[1] »Defenition of 3-tier-application,« [Online]. Available: searchsoftwarequality.techtarget.com/definition/3-tier-application. [Last accessed on 5 January 2018].

[2] »Benefits of 3-tier architecture,« [Online]. Available: www.izenda.com/blog/5-benefits-3-tier-architecture. [Last accessed on 10 January 2018].

[3] »Google docs,« [Online]. Available: docs.google.com.

[4] »Draw.io,« [Online]. Available: github.com/jgraph/drawio.

[5] »Trello,« [Online]. Available: www.trello.com.

[6] »Github,« [Online]. Available: www.github.com.

[7] »Flask,« [Online]. Available: flask.pocoo.org.

[8] »Angular 4,« [Online]. Available: angular.io.

[9] »Cytoscape,« [Online]. Available: js.cytoscape.org.