Academic Search Engine

Software Requirement Specification (SRS)

Abhishek Kumar	Himanshu Meenia	Shashank Verma
2009CS10175	2009CS10192	2009CS10218

Table of Contents

1. Introduction

1.1 Purpose

- The web today provides with a lot of search engines like Google, Bing, Yahoo, etc., with Google dominating the search industry with more than 50% of search queries reaching Google according to the U.S. statistics.
- Originally the search engines were developed for bringing the academic ease. But today the results returned by Google are often not sufficient to satisfy our needs so that we do not need to move any further with searching.
- The reasons for that are many. One of the reasons for this is that there are many useless pages on the web which due to some reason get ranked up by the ranker of the search algorithm. Another one is monetary reason which is evident in the ads seen on the search engine result page. There are some other reasons as well.

1.2 Scope

What we want to build is a academic search engine. There will be many features of this system.

- This system will provide search function allowing user to retrieve relevant research papers.
- This system will also provide the function of finding out the related papers in terms of fields, citations etc.
- The user need not search only by the name of research paper, he can search by author.
- The user sees the papers of his interest area right on his homepage.
- The user browse through his account and see the papers he has downloaded so far
- The user can upload the paper which needs to be approved by the Admin

Once users find their needed research paper, this system will also display the graph representing other papers that cited the searched paper. The results will be ranked according to a ranking algorithm. It will also provide information about total number of search result.

1.3 Definitions, Acronyms and Abbreviations

Terms/Abbreviation	Definition
Authors	Persons who have written the research paper
Crawling	Populating the database with new pages
Database	Collection of all the information monitored by this system.
DFD	Data Flow Diagram
DC	Distributed Computing
DFS	Distributed File System
Homepage	The page the user gets to see after login
Indexing	Hashing the database so as to minimize the search time
SERP	Search Engine Result Page
SRS	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
User	Person viewing the research Paper

1.4 References

- IEEE Std 830-1998
- IEEE Computer Society, 1998, IEEE Recommended Practice for Software Requirements Specifications

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574&userType=inst

- www.citeseer.com

1.5 Overview

This SRS document provides the overall description and the requirements of the academic search engine. There are totally three parts: introduction, overall description or details and specific requirements. In the second part, there will be a more detailed description of the system perspective and features, also the user characteristics and the constraints. In the third part, use case diagram and data flow diagram will be listed in detail.

2. Overall description

This section describes the general factors that affect the product and project requirements. It provides a background for the requirements, which are defined product perspective, its functions, the constraints faced by the system, and who will be using the system. Also, providing project constraints, assumptions and apportioning of requirements.

2.1 Product perspective

Academic Search Engine is a web-based application worked as generic Search Engine systems but more specific on Academic domain. Users can use this application to search what they need to know about their academic interest related to any field like: Computer Networks, Computer graphics, Compilers, Genetics etc. Users can also filter their search referred to the mention above. Moreover, user can see the fields of his interest.

2.1.1 System Interface

Our software will use backend in form of SQL databases. These SQL databases will consist of different types of tables so as to customize the search and provide for other features like displaying the papers of interest to the user. To improve the search performance when the database grows, we will be using Distributed File System and Distributed Computing.

2.1.2 User Interface

User will come to website homepage and he can search as guest or can login. If user logs in and then make the required search than we can track his interests and show the research papers based on the interest of the user.

The homepage will have the following features:

- **Simple Search**
  - **Search by author**
  - **Search by title**
  - **Search by Field**
- **Advanced Search**

  User can use advanced search in Search by Title.

  Eg. Genetics & Computer: This would give results with the title having both “genetics” and “computer”.
- **User Login**
- **Upload Papers**
- **Recent Papers**

  User can see recently added papers of the homepage

- **Citation Graph**

  A research paper can be cited in many other research papers. User can be interested in those papers too. So we provide him with a citation graph in which where each paper is denoted by a node and user can click on the node to open the paper.

2.1.3 Hardware Interface

We would be using php server. Other than that no hardware is required.

2.1.4 Software Interface

When a user enters a query the results are shown on the same page and to facilitate the user to open the right research paper, we will provide **“hover to see abstract”** feature on the very same page. Then he can use the hyperlink to go to the desired research paper.

2.1.5 Communication Interface

We would be implementing our software based on MVC model. MVC is an approach to separating your code depending on what role it plays in your application. In the application flow it starts with a controller that is loaded. That Controller executes a method which retrieves data using Models. Once it is done the controller decides what View to load, which contains the output your visitors get to see.

- **Model**

  Whenever data needs to be retrieved, manipulated or deleted this should always be done by a model. A Model is a representation of some kind of data and has the

methods to change them. For example: you never put **SQL** queries in a Controller, those are put in the Model and the Controller will call upon the Model to execute the queries. This way if your database changes you won't need to change all your Controllers but just the Model that acts upon it.

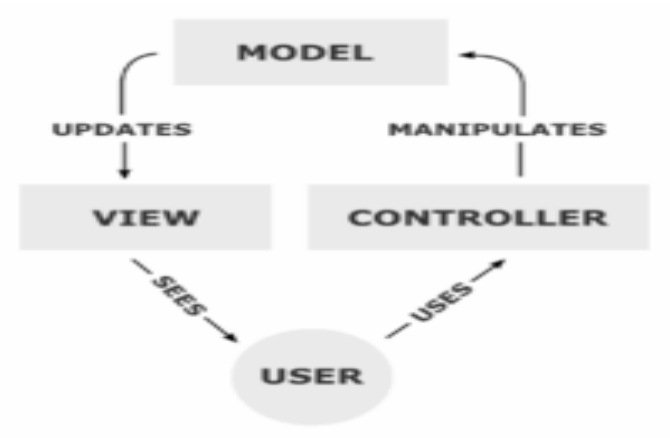
- **View**

  Views contain your HTML, which should never be found in your Controllers or any other class that is not specifically meant to create output. By separating your layout from your logic you ensure that when you decide to change your layout you only have to change the views and won't have to care about the Controllers.

  Views should thus contain little more than echo and for each usage of **PHP**.

- **Control**

  Fuel's routing decides based on the requested URL what controller to load and what method to call upon it. This is where your application starts working. The Controller decides what actions to take, what to do with any user input, what data gets manipulated and which View is shown to the user. The Controller does none of these things itself however; it calls upon Models and Classes to do the work.

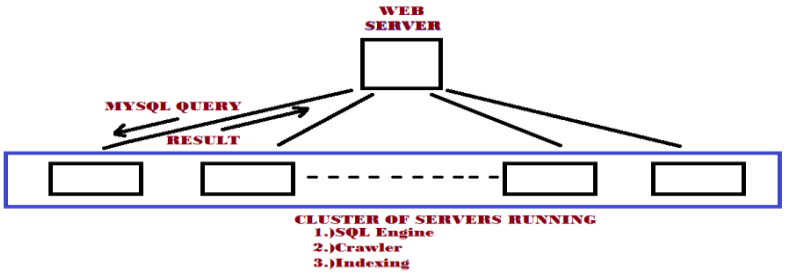


2.1.6 Memory Constraints

To deal with the growing database we will be using Distributed File System. This would also decrease the query time by allowing to implement parallel computation of database search. So the total size of database is actually the hard disk space required. Not much dynamic memory required.

2.1.7 Operations

- User types the search query
- HTTP request sent to the DFS
- SQL query in individual database
- Top 10 result transferred to the central server



2.1.

2.1.8 Site Adaptation Requirements

- **Scalability**

  To account for ever-increasing database we would be using Distributed File System instead of a single file system.

- **Customized Result**

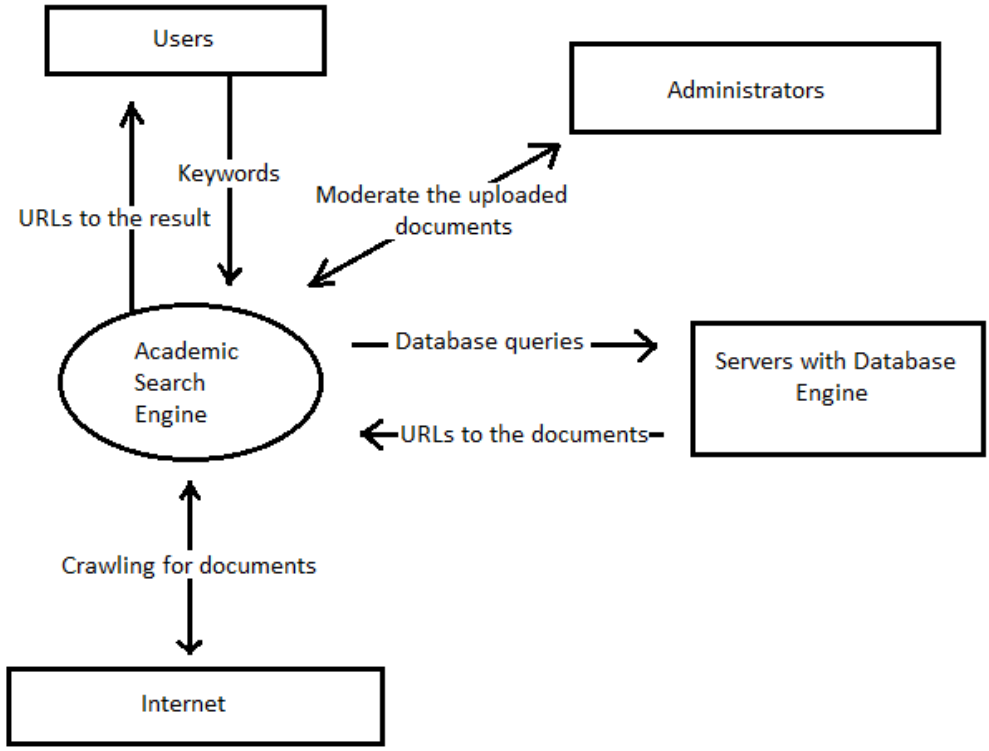
  The search results displayed are based on the interest fields of the user. Also the query is run on the interest field database first so as to improve the performance. If there is no good result from this database then the result of the search on the complete database is outputted.

2.2 Product Functions

Use case and Design diagram is provided in Section 3. The product features and functions are:

- Search engine for finding academic research papers
- User can login or use the search as Guest
- Search can be on title of paper, Author's name
- Machine learning to give custom results to the user
- User can hover over the result link to see the abstract of the corresponding research paper. This would ease user to look for right paper
- New trendy articles on the homepage
- Search based on Fields of research paper
- Citation graph is shown to the user
- Taking everything into account from the perspective of a user the product would suffice his academic needs in a user friendly way

2.2.1 Context Diagram



2.3 User characteristics

- **Language**: Query understanding will be implemented only in English
- **Education**: Users who use this website are expected to be well versed in the field
- **Profession**: A handy website for a person in research field
- **Technical expertise**: Users do not need to have a technical skill but do require the competency to use a computer.

Users who use this website might have an inclination towards research area. They can search the research papers of their choice. The website is well suited to college going students and PhD students.

2.4 Constraints

- **Parallel operation:**
  Implementation of parallel query search (that is simultaneously search two queries) in a

database would require modification in database search engine, which is beyond the scope of this project.

- **Database size:**
  The search result is restricted by the size of database so the results can only be as sufficient as the database backend allows.
- **Query Length and quality:**
  Analysis shows that increase in query length degrades the quality of search result. This is because ambiguity increases in understanding a long query.

2.5 Assumptions and dependencies

- The search result is based on the assumption that user will not make a spelling mistake.
- PDF support for browser to read papers online
- Currently the idea is to use java applet to display Citation graph. So java is needed on the client side
- The size of some papers might be large so a good internet connection is recommended.

3. Specific Requirements

Here we will explain the specific requirements of the project. The topics that have not been discussed before will be given more importance. Some graphical interpretation would be shown as to how does the control flow in our software and what are possible cases that may arise.

3.1 External Interface

- **Web Server –** Apache will be used as our web server. When a query comes to the web server it executes our code with input as the post data in the http request. Then the appropriate response is returned.
- **CGI application –** It is the MVC framework that will actually perform the operations written in PHP.
- **SQL Database –** This is the database to store everything that will be needed and our application will communicate with it to return the results.

3.2 Functional Requirement

Here there will be a graphical description of the design of the product and how control flows would be evident from the use case. The design diagram for the use case below is given in section 3.2.2.

3.2.1 Use Case 1: Academic Paper Search

- **Brief Description**

  This use case describes how the user uses the Academic Search Engine to search for research paper of his interest.

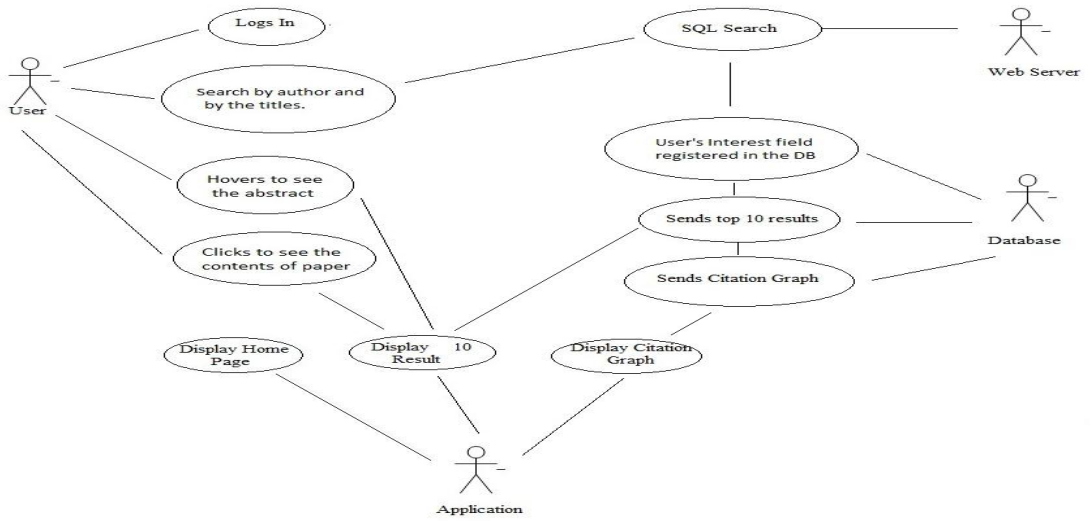
- **Actors**
  - User (Student, Professor, Research Scholar)
  - Database
  - Application
  - Web server
- **Preconditions**
  - Internet connection on the client side
  - Database must contain sufficient papers of all fields
  - The web server is up and running
- **Basic Flow of Events**
  - The use case begins when user opens our website
  - The user enters his login details
  - User will see the research papers of his own interest and new available research papers which are viewable
  - User selects his search criteria like **search by author, search by title**
  - User enters a query for a research paper
  - The application gets the query through post request and queries it to the database through SQL
  - The application decides whether to add the current query's field to the current user's interest field
  - Backend does the searching on its indexed table and returns its top results
  - Application populates the search page with the papers that it got
  - User hovers over them thus seeing their abstract so as to decide which one to view
  - Having decided he clicks on a paper for viewing
  - In order to give more details on the paper he chose he will get to see the **Citation graph**
  - He can click on the nodes of this graph to see the paper that cited it
  - Use case ends successfully

- **Alternate Flows**
  - **Invalid User –** Supplied user name or password was incorrect**.** The user may enter a different id or cancel the operation. Use case ends

- **Skip Login –** User can also search as a guest. In this case we won't track his query searches. User won't get to see the citation graph in this case
- **Query not found** – If a particular query is not found in the database then system will display an error message. The user may enter a new query. Use case ends
- **Paper not available –** This case should not arrive but if it does that matched a query to an paper but we don't have the paper then we will the alternative link that we store with every entry to display it

- **Post Condition**
  - Successful Completion – The user got the required research paper and his interests have been updated
  - Failure Condition – The user has to retry
- **Special Requirement**
  - The page will display 10 top results at a time
- **Use case relationships**
  - None

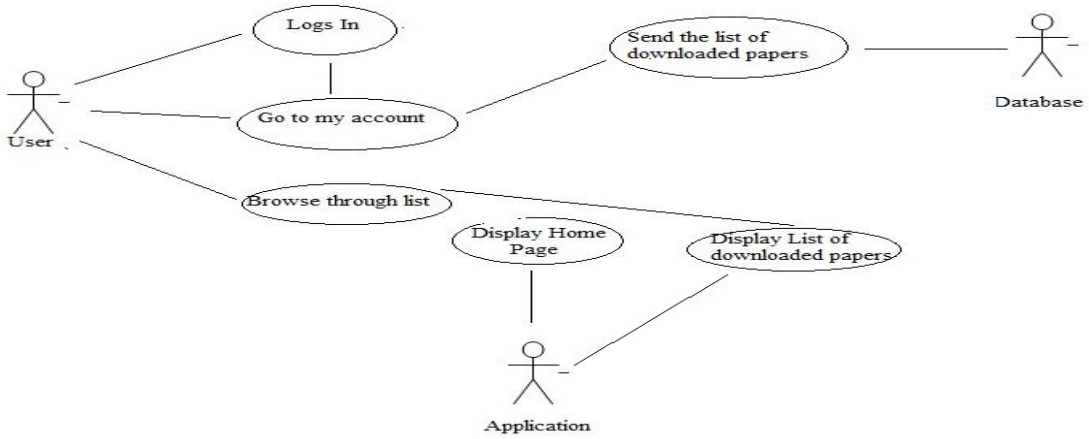
- **Process Flow Diagram**



3.2.2 Use Case 2: User Download History

- **Brief Description**

  The user can login and check his account for the list of papers he has already downloaded
- **Actors**
  - User (Student, Professor, Research Scholar)
  - Database
  - Application
- **Preconditions**
  - Internet connection on the client side
  - The web server is up and running
- **Basic Flow of Events**
  - The use case begins when user opens our website
  - The user enters his login details
  - User goes to his account details
  - The application gets the query through post request and queries it to the database through SQL
  - The database sends back the list of all the papers issued in the name of that particular user
  - The list is displayed to the user's account
  - Use case ends successfully
- **Alternate Flows**
  - **Invalid User –** Supplied user name or password was incorrect**.** The user may enter a different id or cancel the operation. Use case ends
- **Post Condition**
  - Successful Completion – The user gets the downloaded history
  - Failure Condition – The user has to retry
- **Special Requirement**
  - The page will display 10 top results at a time
- **Use case relationships**
  - None
- **Process Flow Diagram**



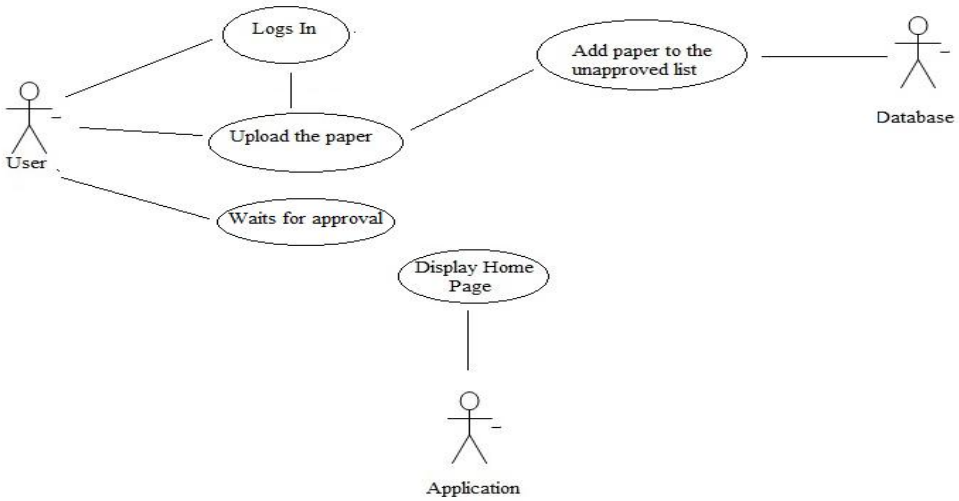
3.2.3 Use Case 3: Upload the paper

- **Brief Description**

  The user can login and upload a new paper from his account. The paper has to be approved by the admin
- **Actors**
  - User (Student, Professor, Research Scholar)
  - Database
  - CGI application
- **Preconditions**
  - Internet connection on the client side
  - The web server is up and running
- **Basic Flow of Events**
  - The use case begins when user opens our website
  - The user enters his login details
  - User uploads the paper and enter details about its tag
  - The database adds the paper to the unapproved list
  - User waits for its approval by the admin
  - The admin authenticates the paper
  - Use case ends successfully
- **Alternate Flows**
  - **Invalid User –** Supplied user name or password was incorrect**.** The user may enter a different id or cancel the operation. Use case ends
  - **Admin disapproves –** The admin disapproves the paper as it is not a research paper. Use case ends
- **Post Condition**
  - Successful Completion – The user gets the poster approved and upload is successful
  - Failure Condition – The user has to retry

- **Special Requirement**
  - None
- **Use case relationships**
  - None

- **Process Flow Diagram**



3.2.4 Use Case 4: Approving the papers

- **Brief Description**

  The admin can login and approve the unapproved paper
- **Actors**
  - Admin
  - Database
  - Application
- **Preconditions**
  - Internet connection on the client side
  - The web server is up and running
- **Basic Flow of Events**
  - The use case begins when admin opens our website
  - Admin enters his login details
  - The database sends the list of unapproved posters
  - Application displays the list
  - Admin sees the list of the unapproved papers

- Admin clicks on the paper to read it's content
- The admin authenticates the paper
- Use case ends successfully

➢ **Alternate Flows**

- **Invalid User –** Supplied user name or password was incorrect**.** The user may enter a different id or cancel the operation. Use case ends

➢ **Post Condition**

- Successful Completion – Admin approves the paper
- Failure Condition – None

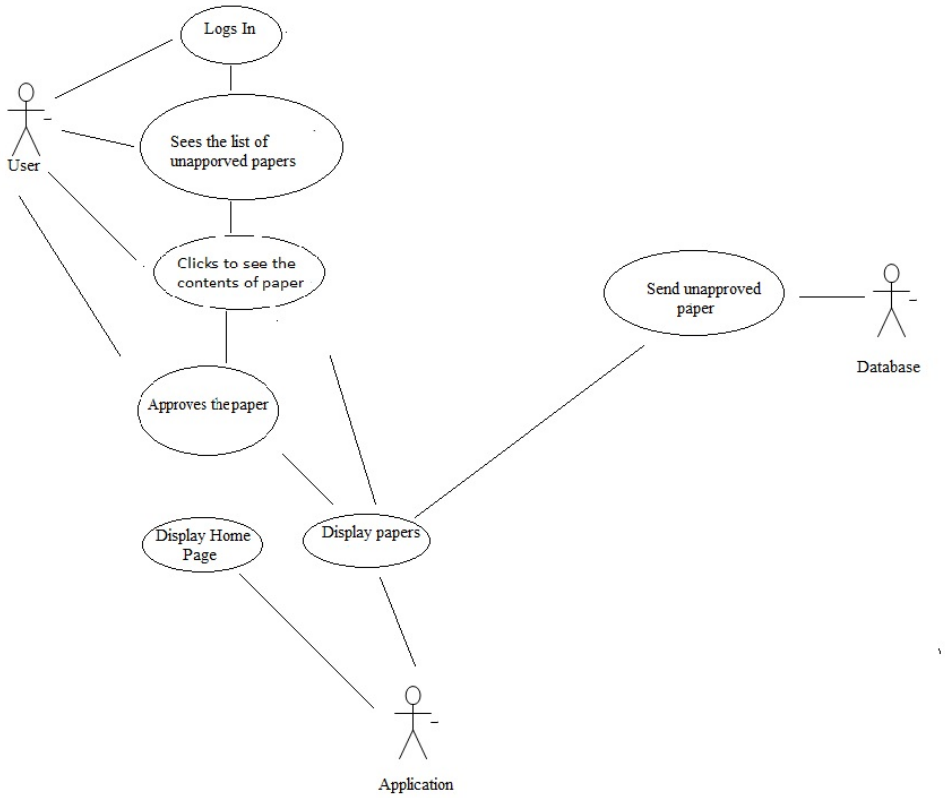
➢ **Special Requirement**

- None

➢ **Use case relationships**

- None

➢ **Process Flow Diagram**



3.3 Performance Requirements

The times specified below obviously do not include network lag. So a good internet connection would be used for meeting the requirements.

- **Query Search Time** – Most important aspect of a search engine is the search time. We would make sure that the query sending to result returning time is less than 5 seconds.
- **Page Load Time** – We will aim to make all the pages of the website to be light weight without compromising quality and attractiveness. This is important so that loading page does not pile on the query search time.
- **Viewing Abstract** – The viewing of abstract feature upon hovering over the papers would not take more than 1.5 seconds. We will try to get the abstract in background so that when hovered, it just has to display and not fetch and display it.
- **Viewing full paper –** This would fetch the paper from our server. Would take time depending on internet connection speed and size of paper.
- **Checking the next page –** This should also take minimal time as when top ten results are shown the next ten would be buffered by server (gathered and ready to send).

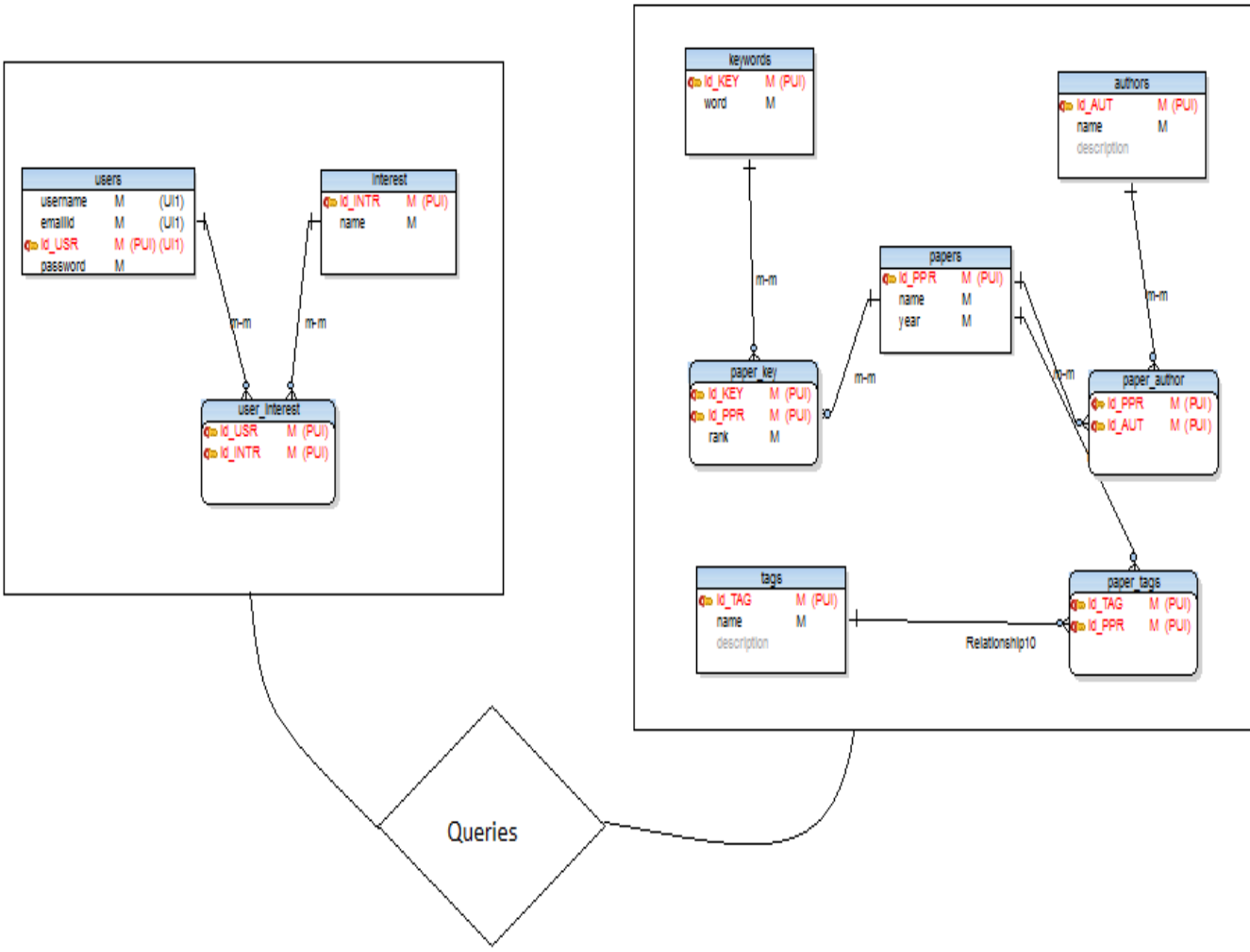
3.4 Logical Database Requirements

We would be using a database engine to manage our backend. SQL is most likely to be the one used. It would be used to not only store the research papers but also index them to allow faster search. The method thought of as of now is matching number of non trivial words that match in any document to rank them. This approach may change later which may need us to change the design of tables described below.

As for this approach we'll require the following tables:

- One would just have the **names of the research paper, their authors and their fields**. Fields are hash-tags like "High Speed Networks", "Approximation Algorithms", "Parallel Computation", etc. Initially we will keep broad category fields and then go into depths.
- Then **we will keep separate tables for each field**. This is for two reasons; one is to get faster search performance by searching in the interest (of user) field table first and then the full table. With proper setting of research interest of the users, this will be able to decrease the search time by at least half on average. The other reason is to implement fast search by field feature in UI.
- Then we will have one **table for mapping authors to their respective papers**. This will be used to implement search by author.
- Another **table would be required having non trivial words mapped to the paper they exist in**. It is a very important table because no one wanting to search some paper will know its full name. So words of query would be matched here and results would be ranked based on the number of hits in its document.

- Then one **table is needed to maintain user information like authentication information, interest of users, etc.** This would be used first while doing login and then when a user queries (fetching of interest field).



3.5 *Design Constraints*

- The product will run on any browser and no specific requirements on OS will be there.
- The product will be written in PHP.
- Communication between product and its backend will be done using SQL queries.
- The output will be compatible with W3C HTML 4.0
- The source code will follow the coding conventions and model design of MVC framework.
- The product would require internet connection to be used.
- No error case would lead to crashing of the product (unexpected behavior).

3.6 Software System Attributes

As discussed earlier, our product will use the elements such as the web server, our CGI application and the SQL database. All the error cases would be handled properly and also the database should remain consistent at all times no matter what case arrives.

3.6.1 Reliability

The reliability of the product would require all individual components to be up and functioning. There are no major issues in this area.

3.6.2 Availability

As long as we have server running with the web site hosted and the database existing, consistent and as required the product site will be available to be accessed from any place with internet connection.

3.6.3 Security

There is not much incentive for hacking other persons account on our web site, so we are not tracking IP of user and not keeping encrypted passwords.

3.6.4 Maintainability

As such the product does not require any maintenance except in case of failure of server when we need to move the hosting and database to another machine to server as server.

3.6.5 Portability

The product consists of web server, CGI application and database. So it does not has any special platform requirements. Any Operating system will be able to run it. All that is required is a server to host the website and database. The end user requires even less, only a browser which supports HTML 4.0 which every machine has.

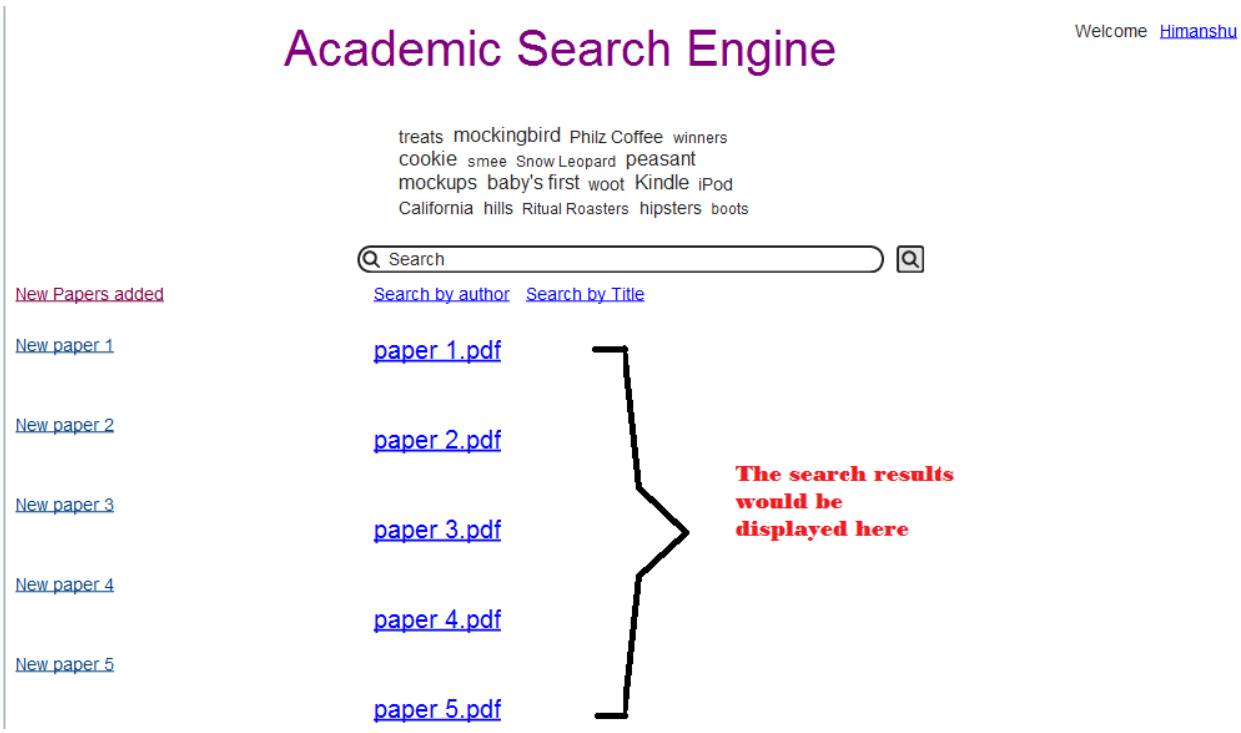
4. Supporting Information

The index and table of contents is given at the starting of this document.

- **Appendix A – Screen Layout**

    - The main page would look like this:

- The user homepage after login would look like this



- **Appendix B – Report Layout**

This report is the Software Requirement Specification (SRS) for project **Academic Search Engine** written following the IEEE 830 Template. It is organized in three major parts. The first

part deals with Introduction and a general Overview of the project. The second part gives Overall Description and third part gives Specific Requirements. All the details needed for implementing the project have been provided in this report.