NDSU SCHOLAR: APPLICATION TO SEARCH ARTICLES RELEVANT TO A RESEARCH PROBLEM

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Rahul Sharma

In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

August 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

NDSU SCHOLAR: APPLICATION TO SEARCH ARTICLES RELEVANT TO A RESEARCH PROBLEM

By

Rahul Sharma

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kenneth Magel
Chair

Dr. Gursimran Walia

Dr. Sudharshan Srinivasan

Approved by Department Chair:

08/19/2015	Kenneth Magel
Date	Signature

ABSTRACT

Because of recent development in the field of computers, there has been substantial growth in the field of online education that varies from online tutors to electronic research papers. Now days, a student can find almost any resource on the internet on any topic they wish to choose from. This platform also assists students in doing their research on a particular topic by providing them a search interface where they can enter their query and get relevant results. Some examples of these search engines are Google Scholar, ASK etc. I have built an online search application that will help assist the students of North Dakota State University (current, or alumni) to search for a research problem in the field of computer science by providing a user friendly interface that will help in finding relevant articles for their query from a large bibliographical database.

ACKNOWLEDGEMENTS

I want to thank my adviser- Dr. Kenneth Magel for guiding me the correct and rational way for this project and also giving me ideas on how to develop and test the application. His support and assistance in helping with this paper has been invaluable. His constant willingness to discuss my application throughout these semesters is greatly appreciated.

I’m grateful to Dr. Gursimran Walia for taking out the time to be a part of my supervisory committee and also providing his feedback on the paper.

I’m grateful to Dr. Sudharshan Srinivasan for taking out the time to be a part of my supervisory committee and also providing his feedback on the paper.

I'm also grateful to my family-Subodh Kumar, Renuka Sharma and Rohan Sharma for giving me moral support in completing this paper. I also dedicate this paper to my two lovely dogs- dollar and junior.

I am also grateful to Mr.Anurag Goswami for his relentless support and motivation in completing this paper.

Finally, I’m grateful to the Computer Science department for all their help and support in my Bachelor's and Master’s program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. PROBLEM STATEMENT AND INTRODUCTION

Problem Statement: Develop a web based search engine that will be used for searching articles relevant to a research problem in the field of computer science. This application will be developed for the North Dakota State University Computer Science Department.

Introduction: This paper provides the explanation and the steps I followed to develop a web based search engine for the NDSU Computer Science (CS) Department. The paper also describes the challenges faced while gathering data and designing the database for this application . The paper will also describe the search algorithm that I derived for this application based on the data set that I retrieved from the database.

There is a need for students who are working on a research problem to have a user friendly interface that will search articles relevant to their problem. The scope of this is very large because of the numerous articles on the internet which includes the NDSU catalog. We need an application that will perform the task of finding these articles related to the field of Computer Science only. The scope of our application would be providing the user with a collection of relevant Computer Science research articles for a particular search request.

Because of the nature of my application, the scope is only limited to Computer Science scholarly articles and the problem is to develop a dynamic interface that would provide an efficient solution to a given search request by a user. I have developed a user friendly interface that provides efficient and fast processing of the request. The performance of my application is solely based on how the data is stored in the database and the indexing of the data columns . My application displays the list of all relevant articles that belong to a particular search request i.e. the title of the paper, the author, link to the article, year the article was published and the number of pages to all the relevant articles for a given request with maximum accuracy and precision.

I have built a system that is similar to a bibliographic database search engine because of the nature and implementation of my application. I was determined to find an efficient solution for finding computer science articles relevant to a research problem. NDSU Scholar has the following functionalities:

1. A user friendly GUI that will be compatible with popular web browsers such as Internet Explorer, Google Chrome, Mozilla Firefox etc.
2. A properly formatted HTML page that will return back the results for a given search query in the correct order.
3. A Login Page that will only allow NDSU students (current, alumni) to use the application. I will be using this login data for creating a cache table to store the most recent keywords for a particular user. This functionality is intended for future releases of my application.
4. A set of articles returned that will be concise to the search query with their proper bibliographical information.
5. The retrieved data set will be free of any duplicates for a search query.
6. My application will allow the user to sort articles by year and date the article was entered in the bibliography.
7. My application will allow users to specify a maximum age for the articles for a search query.
8. My application has proper links for navigating to and fro the search engine.

The main motive of my application is to help computer science students find articles relevant to their research area. My application will also provide users the functionality to sort and filter the articles based on their need. My implemented system replicates most of the functions of a traditional bibliographic search engine.

The database for NDSU Scholar is the backbone of this application. There are two ways through which we can create the database for NDSU Scholar.

1. We can get the bibliographical information from NDSU library on request from the stakeholder.
2. We can create a web crawler that will parse through a huge dataset which will be in the BibTeX format and capture the relevant data and store it in the database.

Step 1 will be a quick and easy solution since we can design our application on top of an already existing database whereas Step 2 will be tedious and complicated to design since we would have to develop a web crawler and a BibTeX parser that will convert the information from the data set links from BibTeX format to a SQL friendly format that my application will use in order to generate formatted HTML tables to display the result set for the given search query. In either ways, the stakeholder has to assume the fact that the database may not always be consistent and errors may arise based on the conversion of the information gathered. NDSU Scholar will be developed to support simultaneous users at a time. This will require a dedicated database server and a web server that will handle and process the requests by users. This server set will be provided by the NDSU CS Department.

1.1. Explanation of Terms

In this section, we explain the terms and abbreviations used in this paper.

1. GUI: Graphical User Interface
2. NDSU: North Dakota State University
3. Backend: This is referred to the database server that processes the requests

4. SQL: Structured Query Language
5. HTML: Hyper Text Markup Language
6. PHP: Hypertext Preprocessor; which is a popular scripting language
7. Web crawler: A program that will download the contents of a given valid URL into a text file
8. BibTeX : It is a popular reference management software which is used for formatting list of references
9. Stakeholder: A person who has a stake in the process. This can be the end user or the person developing the application
10. IEEE: Institute of Electrical and Electronics Engineers
11. ACM: Association for Computer Machinery
12. SIAM: Society for Industrial and Applied Mathematics
13. WWW: World Wide Web
14. CSE: Custom Search Engine which is a service offered by Google that lets a user create and manage their own custom search engine.
15. API: Application Programming Interface
16. NDSU-CAS: North Dakota State University Central Authentication Service

1.2. Organization of the Paper

This paper is organized into 6 chapters, starting with the Problem Statement and Introduction. Chapter 2 discusses the research that I did on the topic, followed by Chapter 3 which explains the how the application was designed by determining the use cases and planning an architecture. Chapter 4 shows the implementation of the application that I developed with detailed state transition diagram, sequence diagram, class diagram and the flow of a user's interaction with the application exemplified by screenshots . Chapter 5 discusses the how the testing of the application was done and the results generated. Chapter 6 gives the Conclusion and Future Work for the application. Chapter 7 and 8 are the References and Bibliography section.

2. BACKGROUND RESEARCH

There are already existing search engines such as Google Scholar, Scirus, CiteSeer, RefSeek, Microsoft Academic Search etc that search for scholarly literature articles across a wide range of topics. The similarities between these search engines is that they have a user friendly GUI and all of them index through a large set of bibliographical entries for their data. The database of these applications consists of bibliographic records which is an organized collection of references to published literature including journal and newspaper articles, reports, government and legal publications, books etc. 95% of students use Google or Yahoo [1] for their search preferences and general search engines are the preferred [2].

Students believe that results from these search engines have less accuracy and therefore are unsuccessful in giving a right solution to their search queries [3]. Research [4] suggested that blogs and wikipedia received the same relevance as academic sources. Hence problem with search results is accuracy as well as irregularity in the output of these search queries. Therefore an alternative (customized) search engine needs to be identified that will solve this purpose for students seeking information on similar interest [5] especially in the field of computer science. This will serve the purpose of limiting sources for computer science students or graduates.

Developing a customized search engine for students will resolve the difficulty of evaluating the results from generalized search engines by automatically narrowing down information which includes more accurate and applicable information. This will enable as well encourage computer science students or graduates more thoughtful use of resources.

There are various search engines [6] which illustrate the strengths and serve the purpose of results that are suitable for students looking for certain information for getting their job done. Following is the description of search engines that was found useful for academic research:

- **Google Scholar**

  Google Scholar(https://scholar.google.com/) is a popular search engine that is used by students for searching research articles. It indexes the full text or metadata of scholarly literature across an array of publishing formats and disciplines. It includes most peer reviewed journals of U.S and Europe scholarly publications, scholarly books and other non peered review journals. Some researchers estimate it contains approximately 160 million documents [7] and approximately 80-90% coverage of all articles published in English [8].

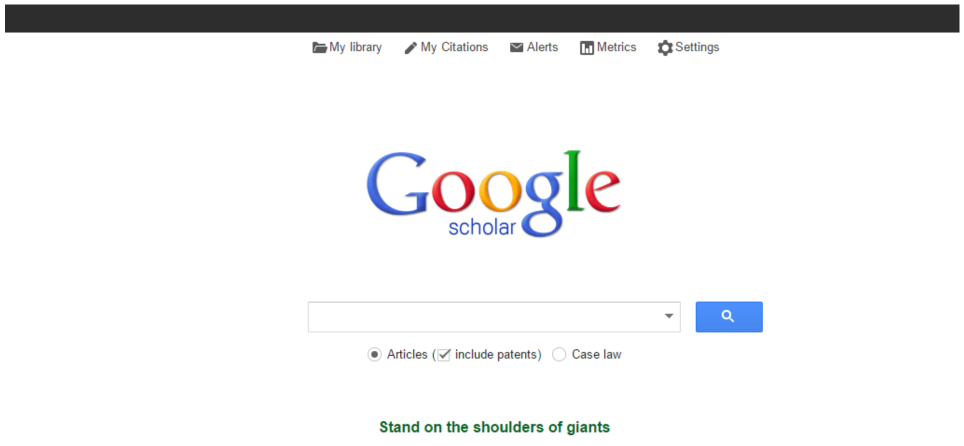
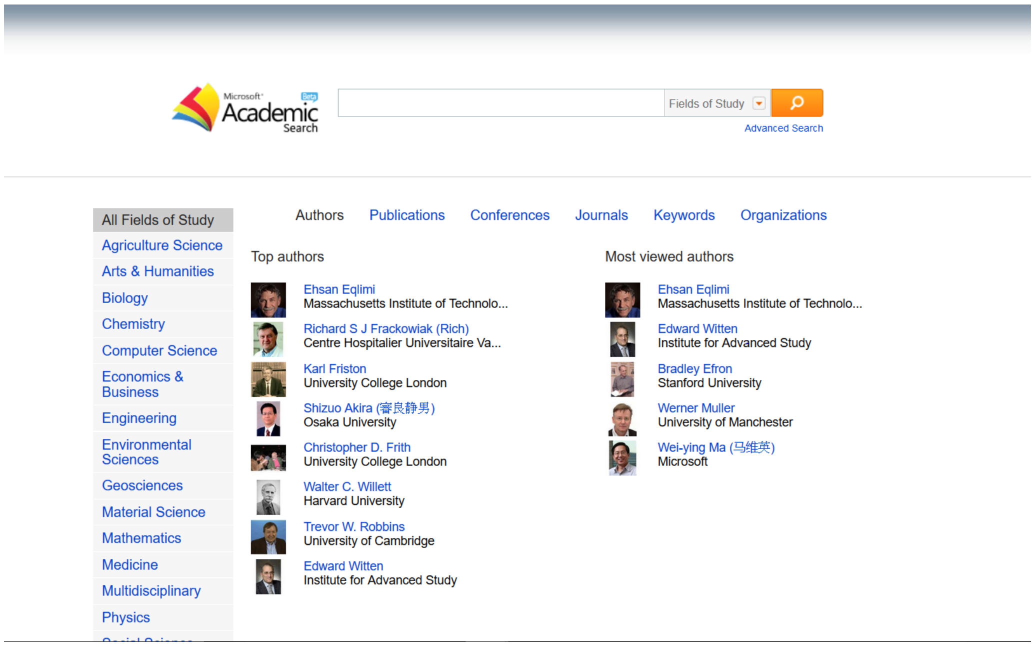


Figure 1: Google Scholar Homepage

- **Microsoft Academic Search**

  Microsoft Academic Search(http://academic.research.microsoft.com/) is a popular free search engine for scholarly articles and literature. It provides a very user friendly interface to students and also provides filtering functionalities based on categories (computer science, social science etc). It is based on a semantic network that has bibliographical information for scholarly papers published in journals, conferences and universities. It is estimated that Microsoft Academic Search indexes over 40 million publications and 20 million authors till date, although their service has not been updated since 2013 and therefore a decline has been seen in the number of indexed documents [9].



Top authors

Ehsan Eqlimi
Massachusetts Institute of Technolo...
Richard S J Frackowiak (Rich)
Centre Hospitalier Universitaire Va...
Karl Friston
University College London
Shizuo Akira (審良静男)
Osaka University
Christopher D. Frith
University College London
Walter C. Willett
Harvard University
Trevor W. Robbins
University of Cambridge
Edward Witten
Institute for Advanced Study

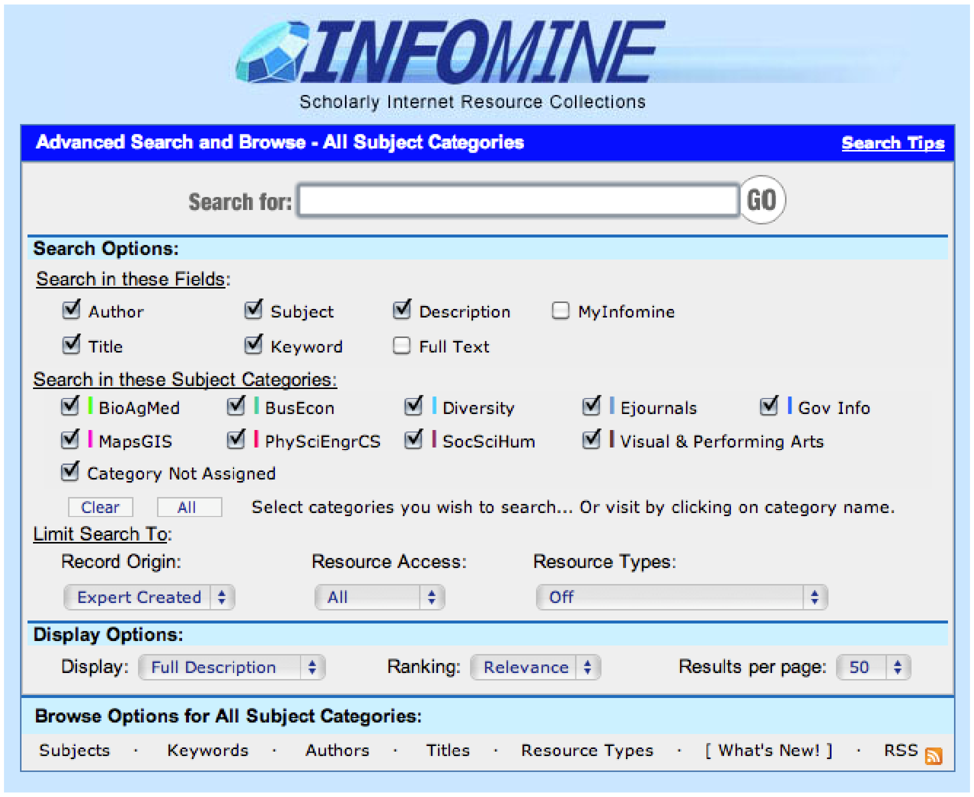
Most viewed authors

Ehsan Eqlimi
Massachusetts Institute of Technolo...
Edward Witten
Institute for Advanced Study
Bradley Efron
Stanford University
Werner Muller
University of Manchester
Wei-ying Ma (马维英)
Microsoft

Figure 2: Microsoft Academic Search Homepage

- **INFOMINE**

INFOMINE(http://infomine.ucr.edu) was a search engine that is utilized for preselected content by real people (2008) created by the university of California, Wake Forest University, California State University, The University of Detroit-Mercy, and other universities. This search engine was a powerful tool for students who are unaware of search terms to be used and how to limit search results. It also consisted of many features such as RSS feeds to help students to be updated of certain topics. However, after 20 years of serving the academic and research community, the University of California Riverside Library ended the INFOMINE service on December 15, 2014.


INFOMINE
Scholarly Internet Resource Collections
Advanced Search and Browse - All Subject Categories
Search Tips
Search for:
GO

Search Options:

Search in these Fields:

☑ Author ☑ Subject ☑ Description ☐ MyInfomine

☑ Title ☑ Keyword ☐ Full Text

Search in these Subject Categories:

☑ BioAgMed ☑ BusEcon ☑ Diversity ☑ Ejournals ☑ Gov Info

☑ MapsGIS ☑ PhySciEngrCS ☑ SocSciHum ☑ Visual & Performing Arts

☑ Category Not Assigned

Clear All Select categories you wish to search... Or visit by clicking on category name.

Limit Search To:

Record Origin: Expert Created

Resource Access: All

Resource Types: Off

Display Options:

Display: Full Description Ranking: Relevance Results per page: 50

Browse Options for All Subject Categories:

Subjects · Keywords · Authors · Titles · Resource Types · [What's New!] · RSS

Figure 3: INFOMINE Homepage

- **RefSeek**

  RefSeek (http://www.refseek.com) is a search engine used by students for subject searching. This search engine has a database of 1 billion documents whose material is not preselected by individuals and is therefore larger than INFOMINE. RefSeek has an ability to narrow down search results by a particular subject which allows the user to search for a specific domain. It returns results for scholarly articles and scholarly websites. The drawbacks of RefSeek include inability to save results, save search history and use advance searching.

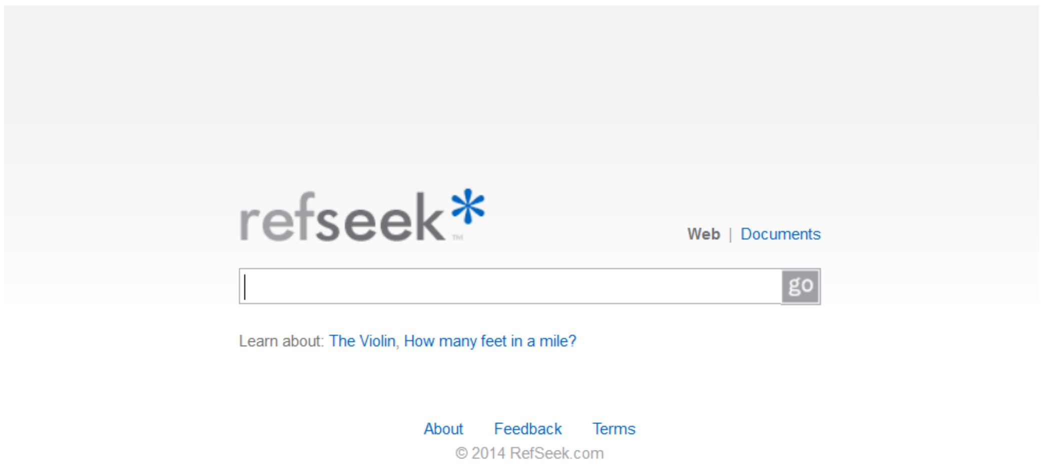


Figure 4: RefSeek Homepage

Therefore motivated from these search engines, I came to a conclusion to develop a customized search engine for computer science students which would provide concise results to search queries and therefore will encourage students to use more of search engines like such.

3. DESIGN

This section discusses how my application was designed by showing the use cases and the functional requirements derived from it. We shall also discuss the architecture of the application in the form of an architecture diagram. We shall conclude this section by discussing the non-functional requirements of the application.

3.1. Search Interface

The front end of my application is the GUI that will be shown to every user that logs in to the system. My main motive was to try out some of the popular search engines discussed in Chapter 2 and get an idea of how their search interface is and also how the result set was generated and shown to the user. I derived that that my application should have display a result page that should contain relevant information about a search query. I also derived that my application should use every usable word from a given search query and that my application should return the maximum number of results with good precision and accuracy.

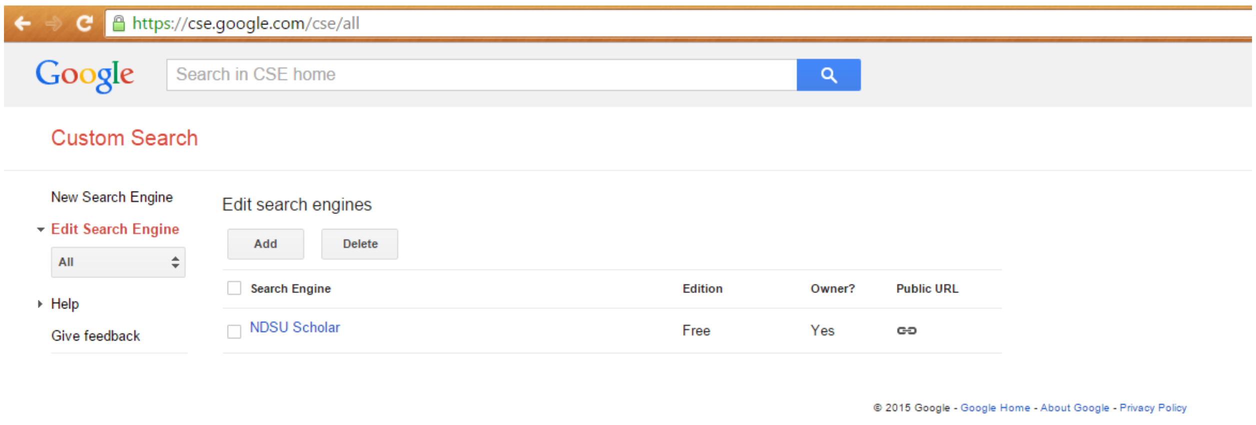
3.2. Backend Database

Since the nature of my application involved research articles, I determined that there was a need of a bibliographical database to be created as the backend of my application. A bibliographical database consists of information about electronic journal articles that can be found in periodicals such as ACM, SIAM etc. Some of the information types that are stored in this database include abstract, author name, title, year etc. Some of the databases even store the links to the full text content of the journal.

Keeping these facts in mind, my next step was to determine how the database for my application was to be designed. I tried to find some free bibliographical database providers like BioOne, Thomson Reuters etc but those providers had their own search interface and therefore I was unable to incorporate their database into my application. Another problem that I encountered was that if I wanted to access their database, I had to pay some amount of money that varied from provider to provider. Therefore I decided to design my own database that would overcome this financial barrier plus I would have full control of my database which means that I could design the database according to my application requirements.

With all these shortcomings, I decided to investigate more into BibTeX format and I tried to find some data online that was in this format. I was able to find a huge dataset of bibliographical entries that were only related to computer science articles [10] which consisted of 272 links that have data in BibTeX format with some in UNIX bib format. All these links point to a number of Internet bibliography collections from ACM, SIAM, IEEE and other magazines and journals.

Another interesting thing that I found during was that Google offers its own search engine API called CSE-Google which lets you create and manage your own search engine for your website. The only thing that is required to incorporate this search engine in your website is that Google requires a JavaScript to be embedded in your program with the search interface and your own custom search engine is ready to be used. Please see below screenshots that illustrate this service provided by Google:


https://cse.google.com/cse/all

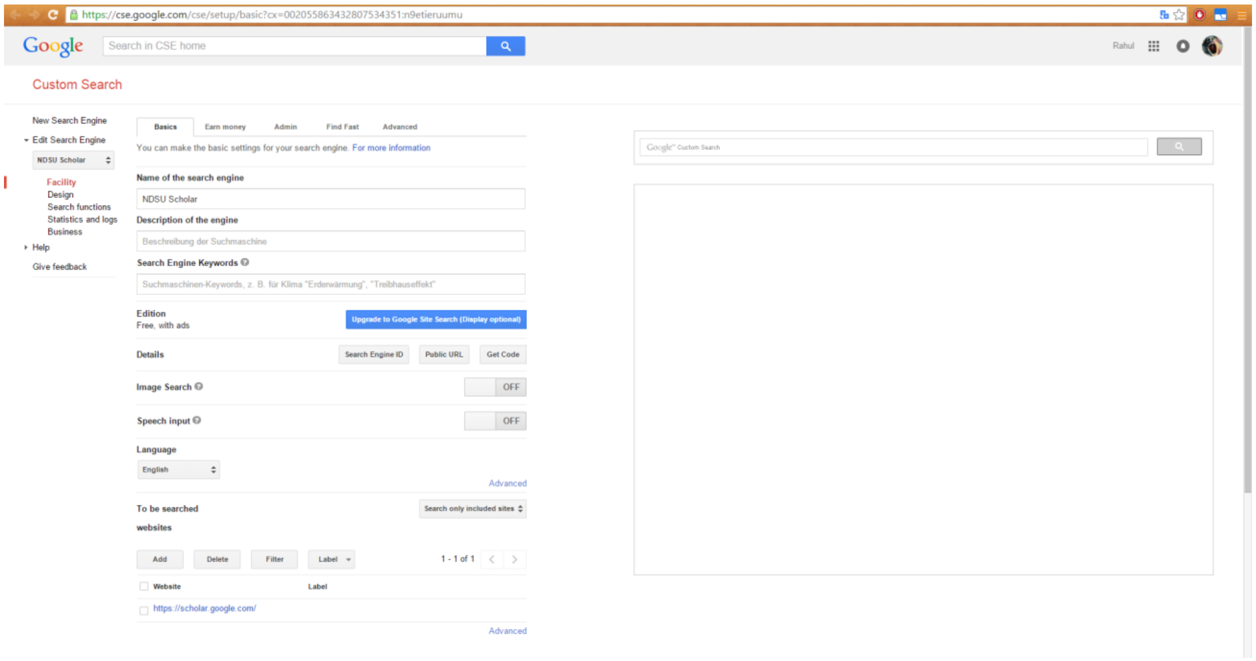
Custom Search

Edit search engines

	Search Engine	Edition	Owner?	Public URL
☐	NDSU Scholar	Free	Yes	

© 2015 Google - Google Home - About Google - Privacy Policy

Figure 5: Google CSE Main Page


https://cse.google.com/cse/setup/basic?cx=002055863432807534351:n9etieruumu

Custom Search

You can make the basic settings for your search engine. For more information

Name of the search engine: NDSU Scholar

Description of the engine: Beschreibung der Suchmaschine

Search Engine Keywords: Suchmaschinen-Keywords, z. B. für Klima "Erderwärmung", "Treibhauseffekt"

Edition: Free, with ads

Image Search: OFF

Speech input: OFF

Language: English

To be searched websites: Search only included sites

	Website	Label
☐	https://scholar.google.com/	

Figure 6: Google CSE Preferences Page

Using these two interfaces, you can create and manage your search engine. Google offers a variety of search interface themes to choose from. Once you choose a them, you can specify the different options to be incorporated in your search engine like speech input, image search etc.

Once you choose your options, Google will generate a JavaScript code which you can incorporate in your website and your search engine is ready to be used.

After evaluating the pros and cons of this service, I got a general idea on how to build my application and therefore after a careful background research on how to design my application, I managed to minimize the functional constraints on my application by taking control of the front and the back end and therefore was able to create a fully functioning search engine.

3.3. Use Case Diagram

The purpose of this diagram is to provide a diagrammatic overview of the functions NDSU Scholar shall provide to a user.

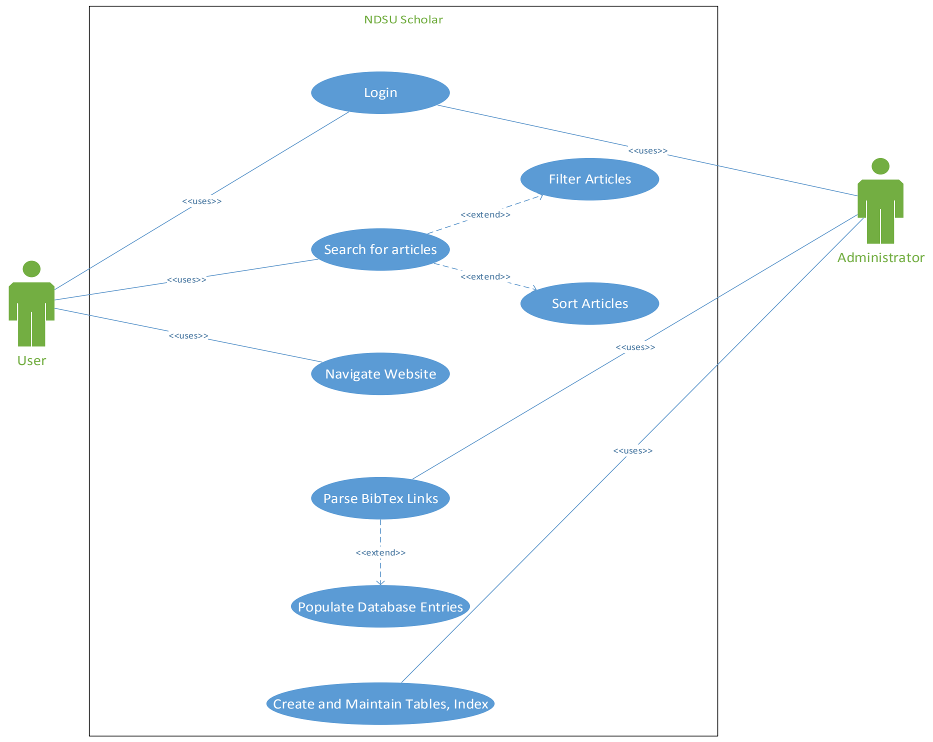


Figure 7: NDSU Scholar Use Case Diagram

Figure 7 models the possible use cases that a user/administrator can perform on the system. The application will authenticate the user first and determine if it is an administrator or not. Once the role of the user is determined, the appropriate interface will be displayed. The user can search for articles using the application which only provides links to articles that are being retrieved from the database. The user is also given the functionality to sort the articles based on newest to oldest and filter the articles based on the year. The application will be user friendly and will allow the user to navigate to and fro the site.

The administrator has the functionality to input a link that points to BibTeX entries and if the validation succeeds, those entries are populated in the database. The administrator can also see the statistics on how many entries were inserted in the database. The administrator is also in charge of the database which involves maintaining, creating, deleting tables, indices on columns, adding/removing new columns in tables based on user needs.

3.4. Functional Requirements

This section identifies the functional requirements that are required to build the system. The functional requirements are derived from the use case diagram and the technical specifications that are provided to us on what the system should do.

Table 1: NDSU Scholar Functional Requirements

Req. Number	**Requirement Description**
FR-1	The application shall display a standard login page to every user accessing the URL
FR-2	The application shall display a tabular form asking the user to fill in their credentials
FR-3	The application shall alert the user if they missed a required field
FR-4	The application shall check against the database if the correct credentials are provided

Table 1: NDSU Scholar Functional Requirements (continued)

Req. Number	Requirement Description
FR-5	Upon successful submission, the application shall display a standard search bar that will allow the logged in user to search for articles
FR-6	The application shall allow every user to search for new queries by providing the search bar on top
FR-7	The application shall display the results of a search query in a formatted HTML page to the logged in user
FR-8	The application shall allow the user to display the results based on most recent articles to oldest by providing a sort button
FR-9	The application shall allow the user to specify a maximum age for the articles to be searched via text input
FR-10	The application shall then filter the articles based on this filter and display it to the user
FR-11	The application shall successfully logout an user when the user signs out
FR-12	The application shall allow the administrator to input a link and based on correct validation, parse the BibTeX data from it.
FR-13	Based on the parsing of the link, the application shall either shown an error or the statistics of the parsed data
FR-14	The application shall provide the administrator with a database interface where the admin can add/delete/modify tables, indices, columns
FR-15	The application shall pass in the parameters to the database for retrieval of articles.

Table 1: NDSU Scholar Functional Requirements (continued)

Req. Number	**Requirement Description**
FR-16	The application shall use regular expressions to extract out the data from the BibTeX format and store in the database.
FR-17	The application shall use Natural Language Full Text search for searching the user query against the database.

FR-1 to FR-14 actually demonstrates what the application will do when a user invokes an event on it. The next set of functional requirements are more specific to the development of the application. FR-15 to FR-17 is the process that the application shall use to refine the search query to give more accurate results from the database.

3.5. Architecture Diagram

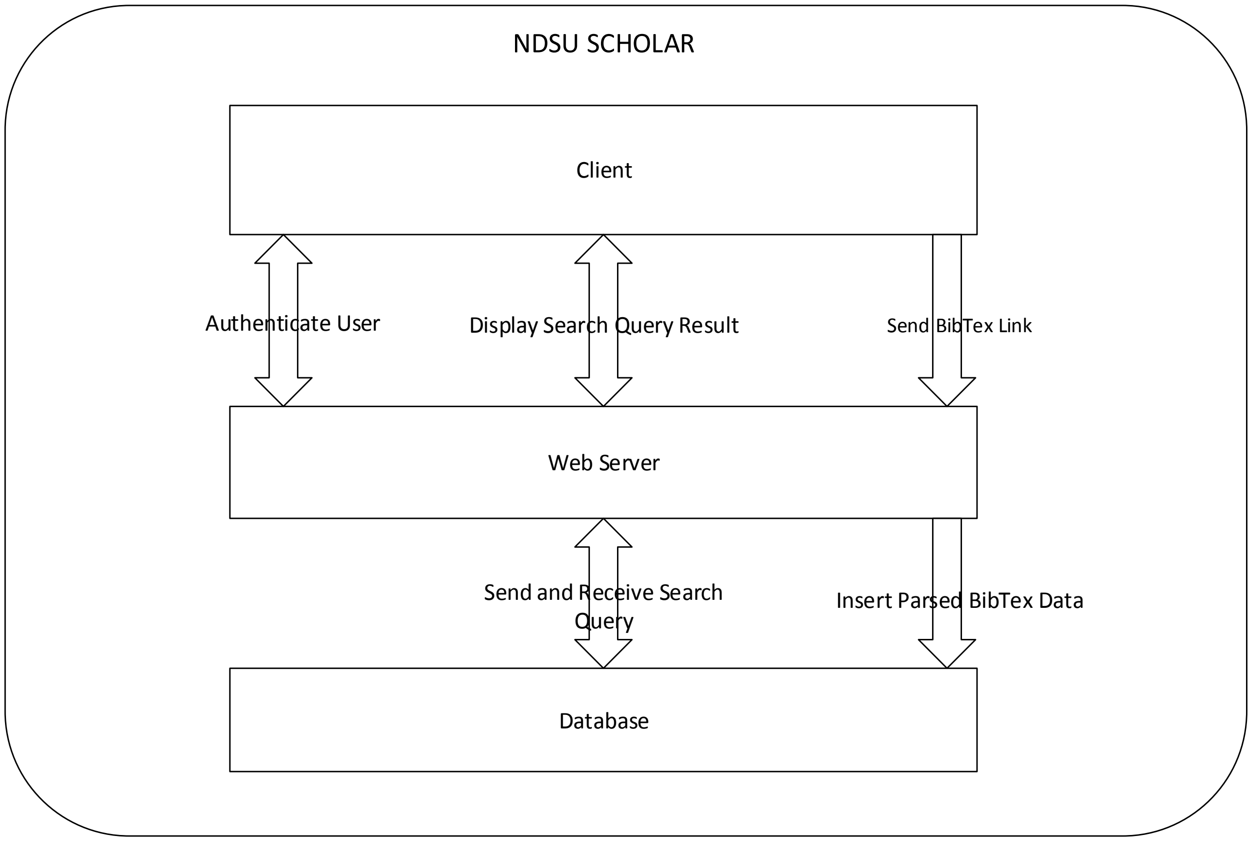

NDSU SCHOLAR
Client
Authenticate User
Display Search Query Result
Send BibTex Link
Web Server
Send and Receive Search Query
Insert Parsed BibTex Data
Database

Figure 8: NDSU Scholar Architecture Diagram

The architecture of my application is very simple to understand. It is a 2 tier architecture with the web server acting as the middle layer between the client and the database. The client is continuously interacting with the web server which handles every user request and processes it. The critical layer for this application is the web server because it handles the client request and also the transactions on the database. The web server will call the database on every user query to retrieve data from the database and also will insert new entries into the database when the administrator inputs a correct BibTeX link. The database has its own tables and index on the relevant columns that will help in the search process. The database has no knowledge of the client and the client has no knowledge of what data is stored in the database thereby maintaining total abstraction.

3.6. Non Functional Requirements

This section discusses the identified non functional requirements for the application. These requirements indirectly affect the overall working of the application and are more general to the performance of the application.

- **NFR-1: Integrity**-: The application should support different login access from different users and should give the correct result for every search query provided to it.
- **NFR-2: Correctness**-: The application should properly manage the transactions and retrieve the results from the database and confirm transactions to the users with 100 % accuracy.
- **NFR-3: Availability**-: The application must be available to user whenever it is needed. The maintenance phase for the application should not be more than 2 hours and should ensure that the application is not offline after the maintenance.
- **NFR-4: Robustness**-: The application should run on any compatible browser that fulfills the minimum requirements. Also the application database should have a nightly backup in case of any failure.
- **NFR-5: Flexibility**-: The application should be dynamic enough in its design and implementation to be able to cope up with the new requirements from stakeholders.

4. IMPLEMENTATION

This section discusses the tools and the programming languages that was used to build the application into a fully functional working entity as proposed in the design phase. The application consists of two main parts which is the front end search interface and the backend database. Therefore it is absolutely essential that the correct tools are used to build both the components separately and then integrate them into one entity. The points below will illustrate how my application was implemented.

4.1. Tools and Techniques

NDSU Scholar was developed using PHP and HTML-5 with CSS and JavaScript on a dedicated server hosted on BITNAMI WAMP Stack 5.4.22-0. The database for NDSU Scholar was created using MySQL hosted on phpmyadmin which is bundled with BITNAMI WAMP Stack.

The implementation of NDSU Scholar was divided into two parts and then integrated into one functioning entity. The front end search interface was built using HTML-5 with CSS style sheets and form validation was done using JavaScript. This technique was used because HTML-5 is compatible with most modern browsers that are available. The CSS style sheet was also designed to give an appealing outlook to the search interface. JavaScript was used to enforce form validation on the form data to ensure that correct parameters are passed to the web server. The database was created using MySQL queries but the data inserted into the database was parsed using a web crawler that is written in PHP scripting language.

For parsing out the BibTeX format links, I used the technique of creating a web crawler that downloads the contents of the link on to a .bib file and then I used a BibTeX parser to parse out the data from the link. This BibTeX parser package is available at:

http://pear.php.net/package/Structures_BibTeX

The parser was tailored according to the data that I got from the BibTeX link and new functions and variables were incorporated so that correct and concise data would be inserted into the database. Extensive error handling was done in the web crawler to ensure that there are no logical errors in the code and correct data is inserted into the database. This was done because this process is the absolute backbone of the application. Since my application is data oriented i.e. the functioning of the application depends solely on the data that is being sent and retrieved, therefore I have used the concept of defensive programming in order to ensure smooth and efficient searching of articles. It is noted that this process is only invoked when the administrator is using the interface to input a BibTeX link.

Once both the front end and the back were developed, they were integrated using PHP pages to show the end result. Since PHP is a dynamic scripting language that allows HTML, CSS and JavaScript to be incorporated with its syntax, it was easy to integrate the client with the web server and display the final result set in a tabular form.

4.2. Classes

This section will illustrate the classes used to develop my application in the form a UML class diagram. This will show the variables and methods used in these classes/pages for my application.

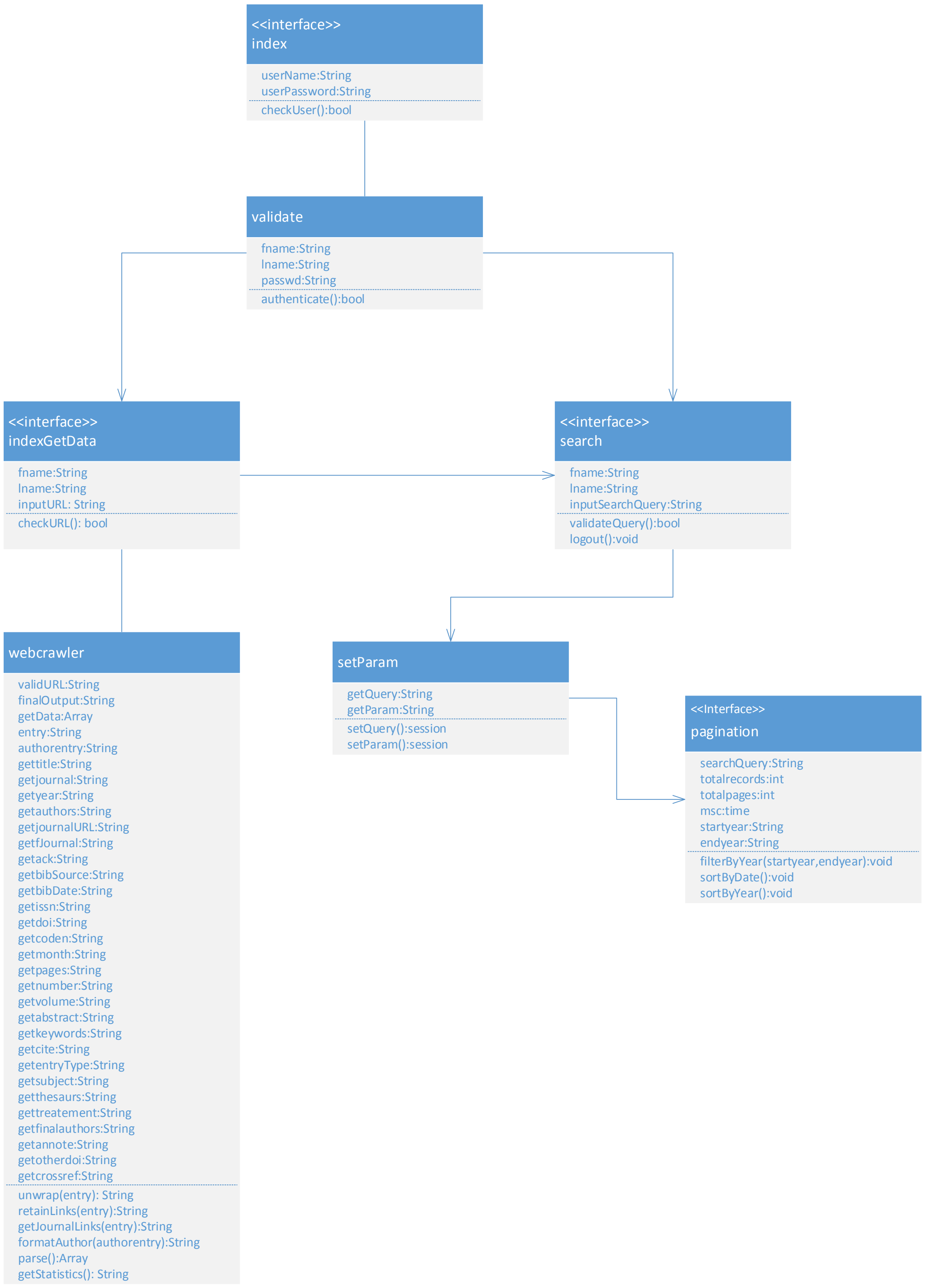


Figure 9: NDSU Scholar Class Diagram

Brief explanations of the classes used in the application are as follows:

Table 2: NDSU Scholar Classes Explanation

Class Name	**Type**	**Purpose**
Index	Interface	This is an interface that is implemented by the validate class. This interface will basically contain all the controls needed to display the login form.
Validate	Class	This class will process the user input and authenticate against the database if the correct credentials are provided. Based on the credentials provided, this class will redirect to admin page or user page.
indexGetData	Interface	This interface can only be accessed by the administrator who gives a valid URL that will be parsed into proper contents.

Table 2: NDSU Scholar Classes Explanation (continued)

Class Name	Type	Purpose
webcrawler	Class	This is the core class that will parse the URL link from the indexGetData interface and parse the data into its respective contents and store them in the database.
Search	Interface	This interface is shown to the user and the admin which takes in their search query.
SetParam	Class	This class takes in every search query and stores it in a session to be used throughout the search process. This class is very essential as it retains the search query and parameters for filtering
pagination	Interface	This interface displays the search results of every user and also provides functionalities to sort and filter the search query.

Design Patterns identified in the class diagram as follows:

Table 3: NDSU Scholar Class Design Patterns

Design Name	**Class using this pattern**	**Purpose**
Singleton	Validate	This pattern is being used in this class because it always ensures that a single instance of this class is created and used for authentication
State	Pagination	This pattern basically allows an object to alter its behavior when internal state changes. Here this class state changes based on the input from setParam class.

4.3. Sequence Diagram

This section will illustrate the sequential logic of my application in the form of a UML sequence diagram. The sequence diagram below will show the sequence of steps that each user will take in order to search for an article using NDSU Scholar. Note that the objects are the classes that we identified in the class diagram.

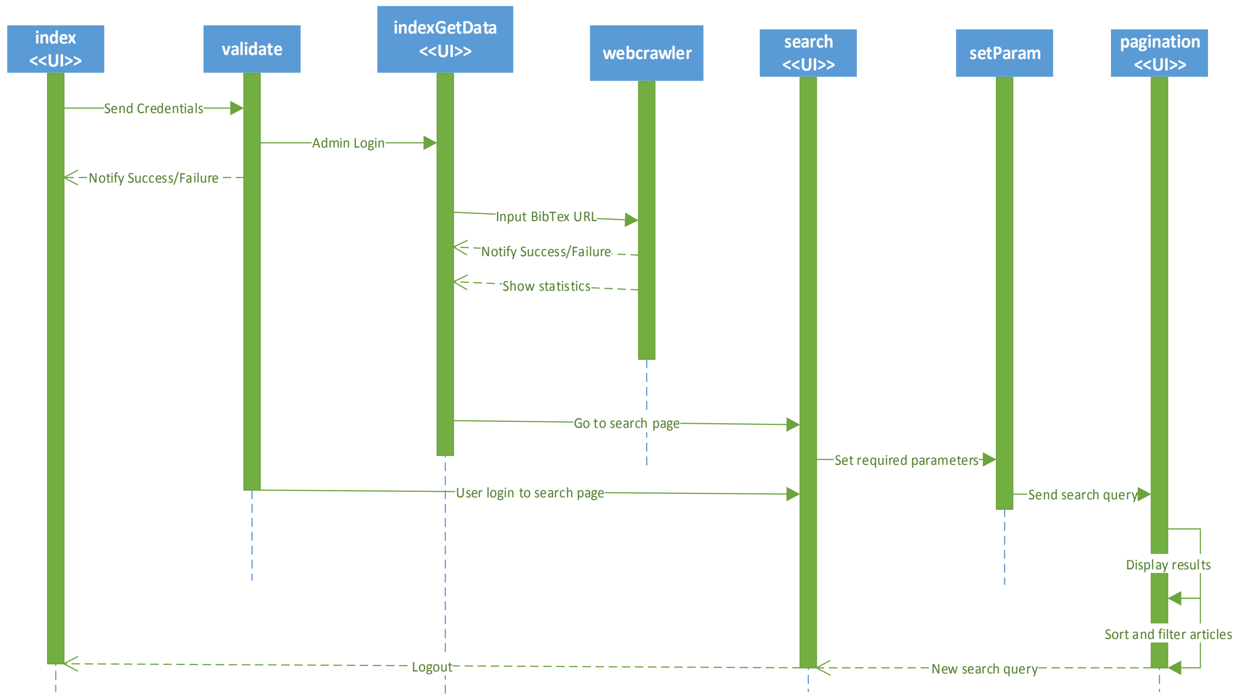


Figure 10: NDSU Scholar Sequence Diagram

4.4. State Transition Diagram

This section will illustrate the different states reached on different events invoked by a user in the form of a UML State Transition Diagram.

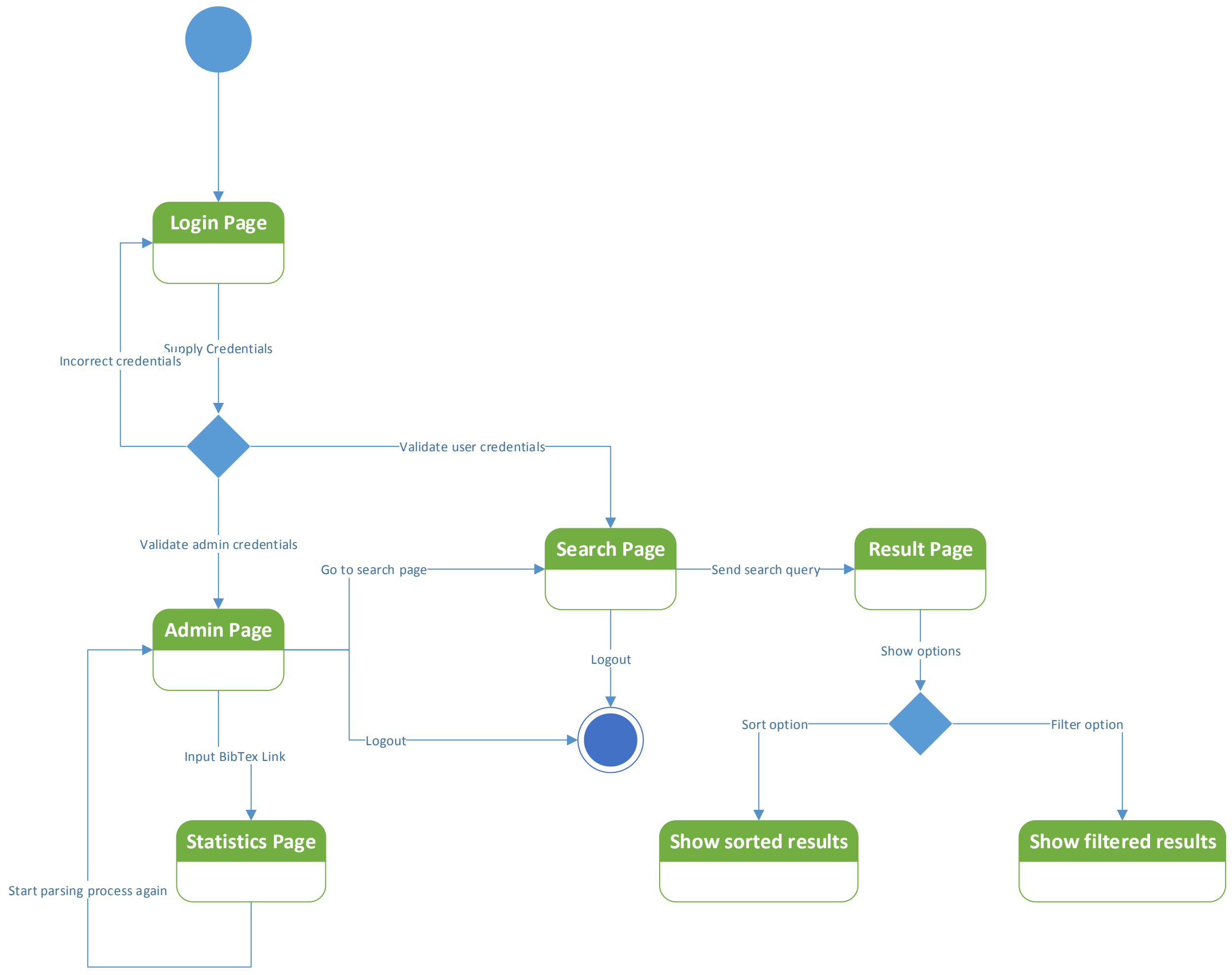
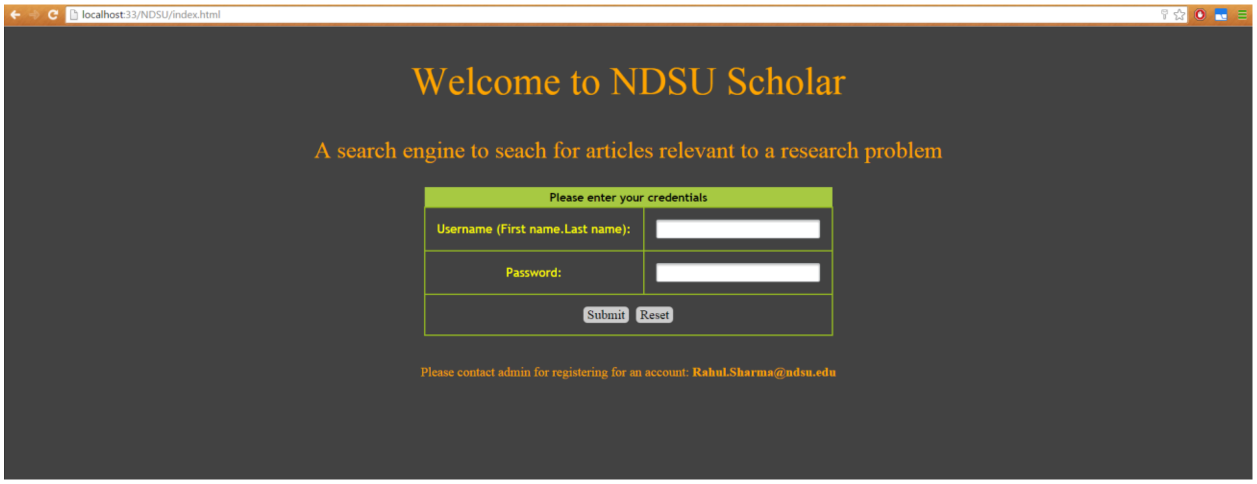


Figure 11: NDSU Scholar State Transition Diagram

4.5. Screenshots of NDSU Scholar

After following the design architecture and UML modeling diagrams, NDSU Scholar was constructed into a physical application that is ready to be tested. Please see below some screen shots of my application:


localhost:33/NDSU/index.html

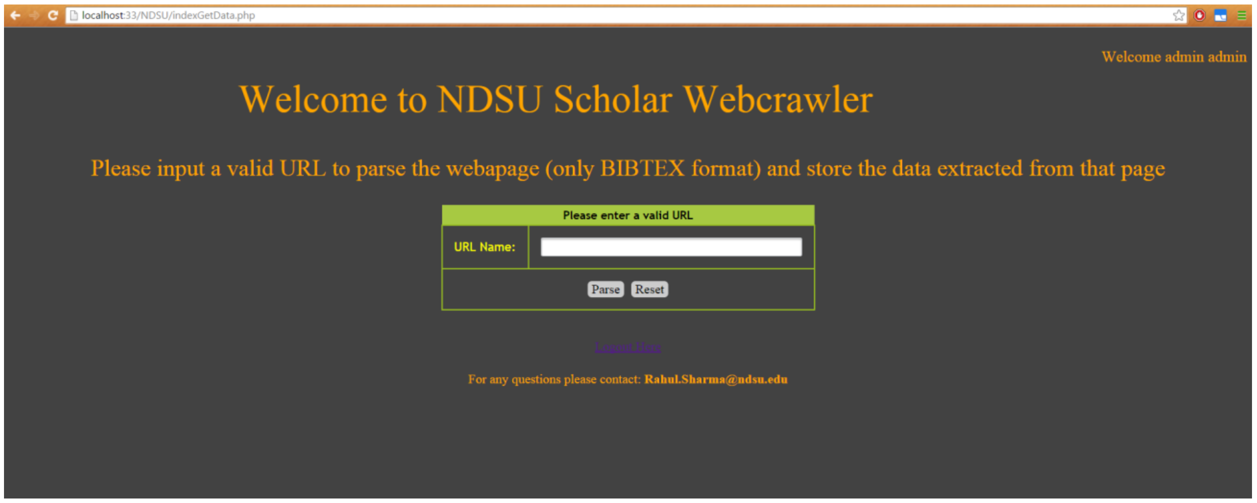
Welcome to NDSU Scholar

A search engine to seach for articles relevant to a research problem

Please enter your credentials	
Username (First name.Last name):	
Password:	

Please contact admin for registering for an account: **Rahul.Sharma@ndsu.edu**

Figure 12: NDSU Scholar Login Page


localhost:33/NDSU/indexGetData.php

Welcome admin admin

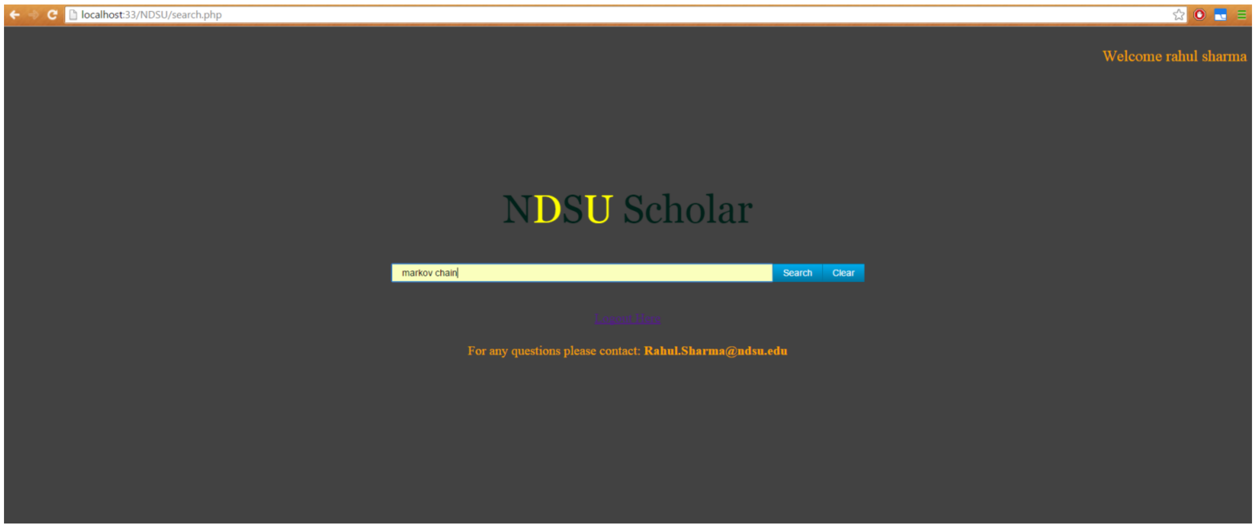
Welcome to NDSU Scholar Webcrawler

Please input a valid URL to parse the webpage (only BIBTEX format) and store the data extracted from that page

Please enter a valid URL	
URL Name:	

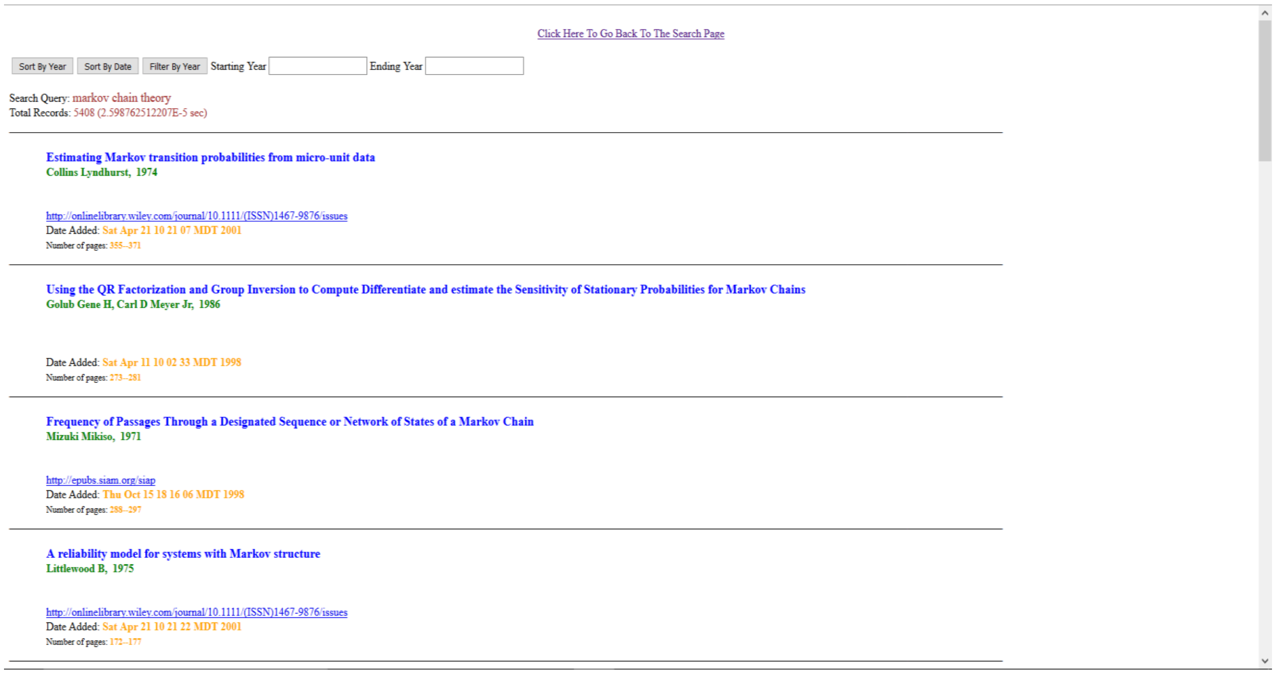
For any questions please contact: **Rahul.Sharma@ndsu.edu**

Figure 13: NDSU Scholar Admin Page



For any questions please contact: **Rahul.Sharma@ndsu.edu**

Figure 14: NDSU Scholar Home Page: Input String- markov chain theory



Search Query: markov chain theory
Total Records: 5408 (2.598762512207E-5 sec)

Estimating Markov transition probabilities from micro-unit data
Collins Lyndhurst, 1974

http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1467-9876/issues
Date Added: **Sat Apr 21 10 21 07 MDT 2001**
Number of pages: 355--371

Using the QR Factorization and Group Inversion to Compute Differentiate and estimate the Sensitivity of Stationary Probabilities for Markov Chains
Golub Gene H, Carl D Meyer Jr, 1986

Date Added: **Sat Apr 11 10 02 33 MDT 1998**
Number of pages: 273--281

Frequency of Passages Through a Designated Sequence or Network of States of a Markov Chain
Mizuki Mikiso, 1971

http://epubs.siam.org/siap
Date Added: **Thu Oct 15 18 16 06 MDT 1998**
Number of pages: 288--297

A reliability model for systems with Markov structure
Littlewood B, 1975

http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1467-9876/issues
Date Added: **Sat Apr 21 10 21 22 MDT 2001**
Number of pages: 172--177

Figure 15: NDSU Scholar Result Page

Figures 14 and 15 show the working of NDSU Scholar by a user. It is noted that links to research articles are being retrieved and shown to the user.

5. TESTING

This section discusses the testing strategies that were performed on my application in order to ensure its correct functioning. Black box testing was mostly performed to check for different states that are explained in below section. Some unit testing was also performed on the classes which mostly pertained to the parsing process of the application. This was done because this process is very essential to the working of the application. The last step in this section is the comparison of NDSU Scholar to two popular search engines: Google Scholar and Microsoft Academic Search.

5.1. Black Box Testing

Table 4 below lists the different possible states that were deduced.

Table 4: Black Box Testing States

State	**Description**
S1	Application loads correctly on a web browser.
S2	Application validates user credentials correctly.
S3	Application displays search engine interface to user.
S4	Application displays parsing link interface to admin.
S5	Application shows correct result set to the user based on the search query.
S6	Application sorts the search results based on newest to oldest.
S7	Application filters the search results based on the year of the article.
S8	Application parses the contents of the link and shows the statistics to the admin.
S9	Application successfully logouts a user.

After finding the different states for my application, I performed black box testing by subjecting different states to different events and if the states changed as expected and reached the desired state, the test was concluded to have been passed. Table 5 illustrates the results of these tests.

Table 5: Black Box Test Cases

S.No	**Initial State**	**Event**	**Expected State**	**Test Result**
1	S1	Launch Application	S2	Pass
2	S2	Input User Credentials	S3	Pass
3	S2	Input Admin Credentials	S4	Pass
4	S3	Input Search Query	S5	Pass
5	S5	Sort Query	S6	Pass
6	S5	Filter Query	S7	Pass
7	S4	Input BibTeX link	S8	Pass
8	S3	Logout User	S9	Pass
9	S4	Logout Admin	S9	Pass

After performing these core tests, additional black box testing was done to check for some of the major functionalities of NDSU Scholar which include searching for articles, sorting the result set, filtering the result set, GUI compatibility etc. Please see table 6 that summarizes the different test cases deduced:

Table 6: Black Box Test Cases for Functionalities

S.No	Test Case	Preconditions	Steps	Expected Result	Test Result
1	Sort By Year	User has to be signed in to NDSU Scholar	• Input search query • Click on sort by year button	The result set will be sorted in ascending order according to the year of the article.	Pass
2	Sort By Date	User has to be signed in to NDSU Scholar	• Input search query • Click on sort by date button	The result set will be sorted in ascending order according to the date of the article.	Pass

Table 6: Black Box Test Cases for Functionalities (continued)

S.No	Test Case	Preconditions	Steps	Expected Result	Test Result
3	Filter By Year	User has to be signed in to NDSU Scholar	• Input search query • Input year range to be searched • Submit the query	The result set will only display those articles that are in the year range inputted by the user.	Pass
4	Search for articles	User has to be signed in to NDSU Scholar	• Input search query	A dynamic result set pertaining to the input query will be displayed to the user.	Pass

Table 6: Black Box Test Cases for Functionalities (continued)

S.No	Test Case	Preconditions	Steps	Expected Result	Test Result
5	Check for duplicates	User has to be signed in to NDSU Scholar	• Input search query • Check for the search query result set in the database • Check if any duplicates exist • If yes, then check if the duplicate entry is shown to user	A result set consisting of only unique entries will be displayed to the user.	Pass
6	Browser Compatibility	User has to be signed in to NDSU Scholar	• Run NDSU Scholar on Google Chrome, Mozilla Firefox, Internet Explorer and Microsoft Edge	NDSU Scholar should display the search interface and the results correctly.	Pass

Table 6: Black Box Test Cases for Functionalities (continued)

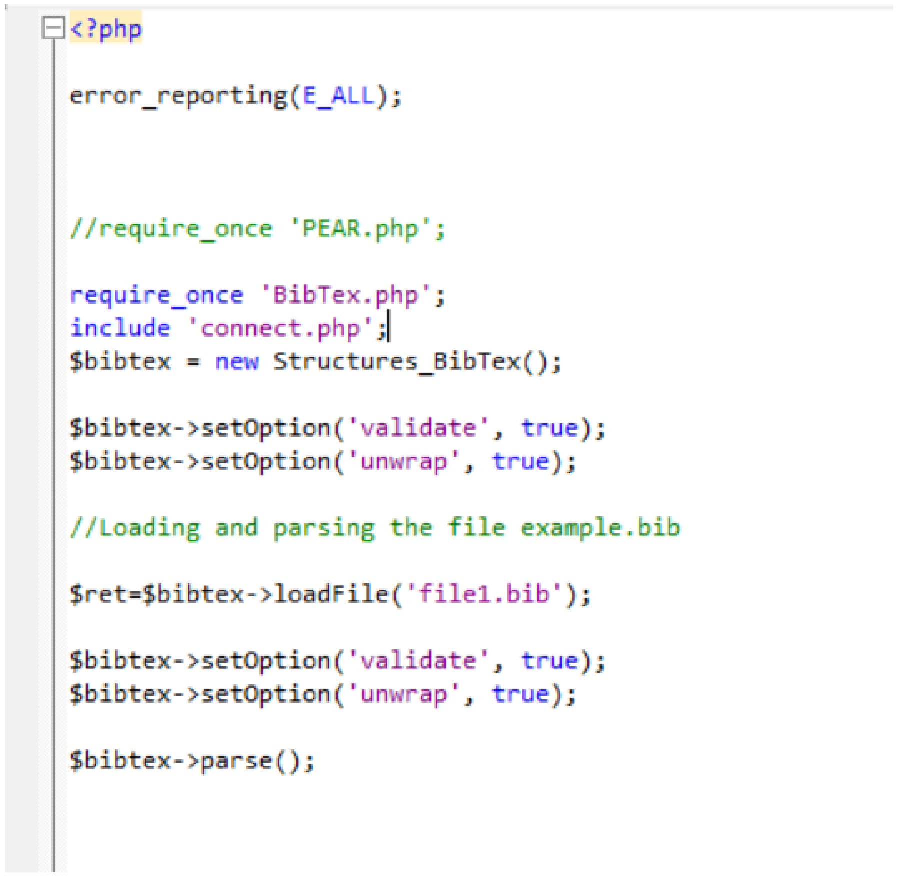
S.No	Test Case	Preconditions	Steps	Expected Result	Test Result
7	Check for stop words	User has to be signed in to NDSU Scholar	• Input search query consisting of only stop words	No result set will be displayed to the user.	Pass
8	Check for format of user credentials	NDSU Scholar should be running on the browser	• Input user credentials with special characters to NDSU Scholar • Input incorrect user credentials to NDSU Scholar	NDSU Scholar should raise an error stating that incorrect credentials have been provided.	Pass

5.2. Unit Tests

This section discusses the unit tests performed on the **Structures_BibTeX** class which is the core class that handles the parsing of the BibTeX link that is acquired via the admin interface. The critical functions of this class are the **parse** and **_parseEntry** functions that perform the task of parsing the data array that comes in from the BibTeX link. The parse function performs the initial task of splitting the huge data array and then it sends each parsed entry to the _parseEntry function which then further splits each part into its corresponding bibliographical information such as author name, abstract, volume, number of pages etc.

Unit tests were performed on these two functions by including a test script that I created to test the inner logic of these functions and check if correct data is being returned by this class. This step is very essential as we are inserting the parsed data from this class into the database that I have created.

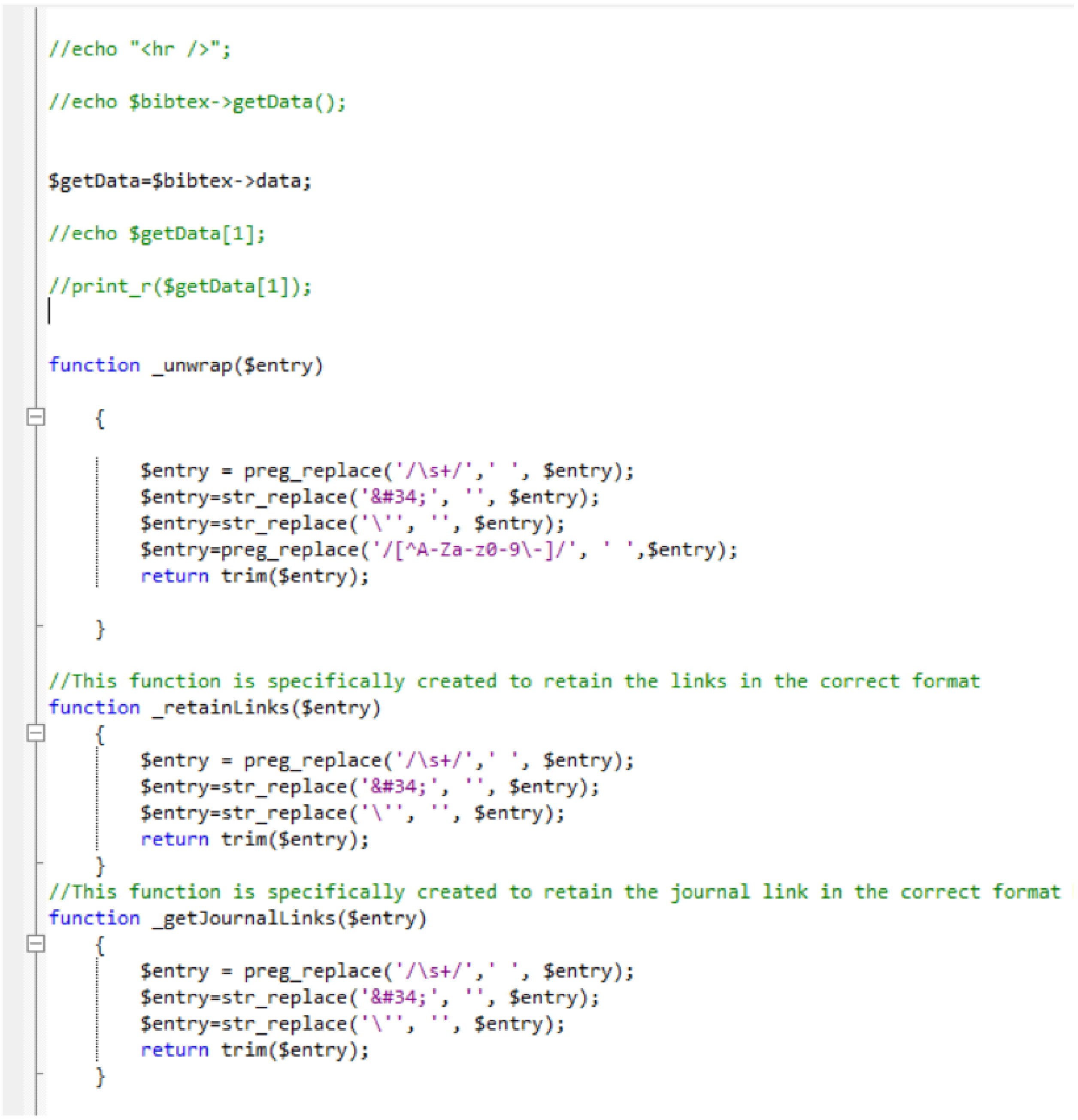
Figures 16 and 17 explain some parts of the code where the critical testing of the application is being performed:



```
<?php

error_reporting(E_ALL);



//require_once 'PEAR.php';

require_once 'BibTex.php';
include 'connect.php';
$bibtex = new Structures_BibTex();

$bibtex->setOption('validate', true);
$bibtex->setOption('unwrap', true);

//Loading and parsing the file example.bib

$ret=$bibtex->loadFile('file1.bib');

$bibtex->setOption('validate', true);
$bibtex->setOption('unwrap', true);

$bibtex->parse();
```

Figure 16: Loading .bib file- Unit Test

The above code abstract shows the basic loading of the .bib file which consists of the initial data gathered from the BibTeX link. For this test script, I manually inserted data into the file according to my own testing strategy. Once the file is read, the parse function is called which then parses the contents of the file into corresponding bibliographical information.



```
//echo "<hr />";

//echo $bibtex->getData();


$getData=$bibtex->data;

//echo $getData[1];

//print_r($getData[1]);


function _unwrap($entry)

    {

        $entry = preg_replace('/\s+/',' ', $entry);
        $entry=str_replace('&#34;', '', $entry);
        $entry=str_replace('\'', '', $entry);
        $entry=preg_replace('/[^A-Za-z0-9\-]/', ' ',$entry);
        return trim($entry);

    }

//This function is specifically created to retain the links in the correct format
function _retainLinks($entry)
    {
        $entry = preg_replace('/\s+/',' ', $entry);
        $entry=str_replace('&#34;', '', $entry);
        $entry=str_replace('\'', '', $entry);
        return trim($entry);
    }
//This function is specifically created to retain the journal link in the correct format
function _getJournalLinks($entry)
    {
        $entry = preg_replace('/\s+/',' ', $entry);
        $entry=str_replace('&#34;', '', $entry);
        $entry=str_replace('\'', '', $entry);
        return trim($entry);
    }
```

Figure 17: Object creation and parsing data- Unit Test

Figure 17 shows how an object of Structures_BibTeX class was created that got parsed data array from the two functions mentioned above. I have used regular expressions throughout my program to test if correct data is being returned from the parse function. After performing a variety of tests, I was able to get correct and concise data. The last step included inserting this retrieved data into the database.

5.3. Comparison to Google Scholar and Microsoft Academic Search

Google Scholar and Microsoft Academic Search are two major search engines that are frequently used by students to search for articles relevant to their research problem. NDSU Scholar aims at achieving the same desired functionality to help students out with their research problem. The database size of Google Scholar and Microsoft Academic Search are considerably huge as compared to NDSU Scholar. We are only concerned with the database size of computer science articles stored in each search engine. Figure 18 shows the proportion of documents gathered by Google Scholar and Microsoft Academic Search:

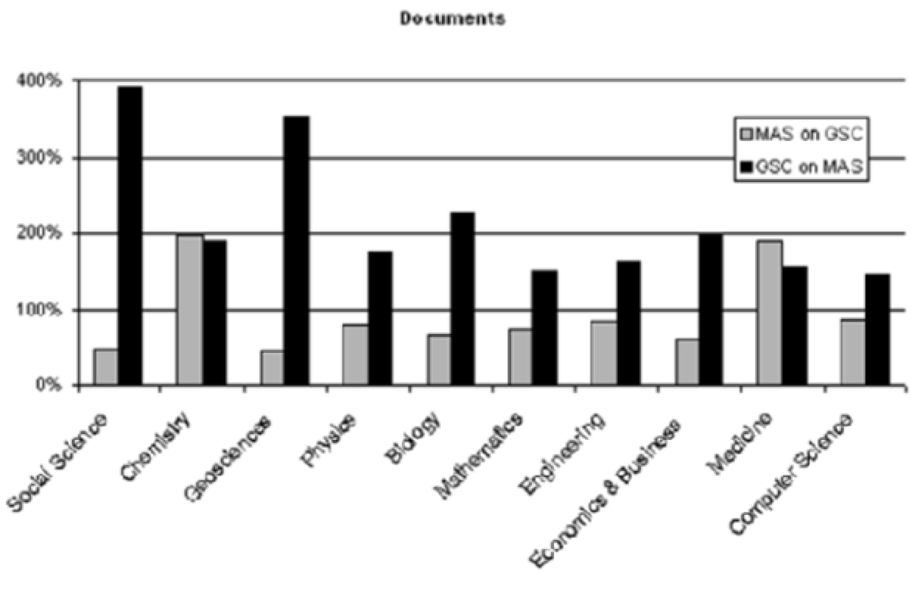


Figure 18: Proportion of documents gathered by Google Scholar and Microsoft Academic Search according to disciplines [11]

As we can see from Figure 18, Google Scholar has more proportion of documents than Microsoft Academic Search in the field of computer science which implies that Google Scholar gives better search results for computer science articles. The number of citations stored by each search engine is shown in Figure 19 below:

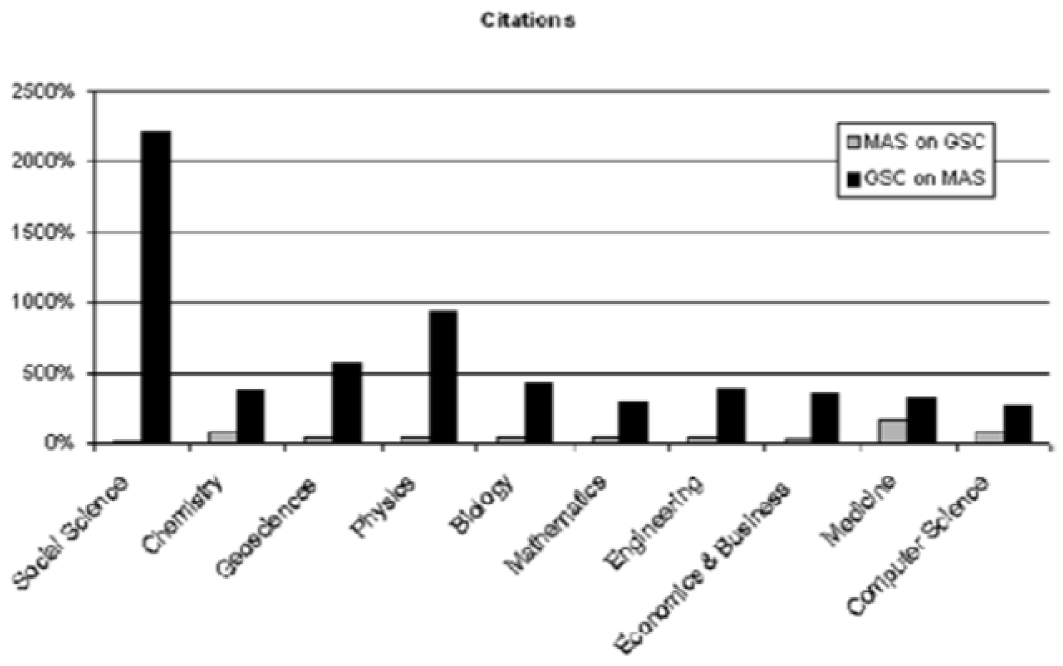


Figure 19: Proportion of citations gathered by Google Scholar and Microsoft Academic Search according to disciplines [11]

NDSU Scholar database size is smaller as compared to these two search engines because the implementation is still in the preliminary stages. My application only provides basic functionality for searching, sorting and filtering articles . The database only consists of articles from the field of computer science. The core information of my application consists of the parsed data that we got from the BibTeX links. The current size of NDSU Scholar database is 123793 documents and 123792 citations with the possibility of expanding the database.

Once I was successfully able to insert the data into the database, my next step included writing test SQL scripts to search for queries from the table that I created using the parsing process. In order to compare NDSU Scholar with Google Scholar and Microsoft Academic

Search, I ran 14 different queries on different functionalities and compared my results to Google Scholar and Microsoft Academic Search. My sole motive for this was to evaluate the result set and functionalities of NDSU Scholar and compare them with Google Scholar and Microsoft Academic Search. Since NDSU Scholar only returns articles relevant to the field of computer science, I was able to compare the result set and the results are summarized in Table 7:

Table 7: Comparison of NDSU Scholar with Google Scholar and Microsoft Academic Search

S.No	**Task**	**NDSU Scholar**	**Google Scholar**	**Microsoft Academic Search**
1	Sort By Date	✓	✓	✗
2	Sort By Year	✓	✓	✗
3	Filter By Category (computer science)	✗	✗	✓
4	Filter By Year	✓	✓	✓
5	Filter By Year and Sort by Date	✓	✗	✗
6	Search By Author	✓	✓	✓
7	Search By Title	✓	✓	✓

Table 7: Comparison of NDSU Scholar with Google Scholar and Microsoft Academic Search (continued)

S.No	Task	NDSU Scholar	Google Scholar	Microsoft Academic Search
8	Search By Keywords	✓	✓	✓
9	Search By Citations	✓	✗	✗
10	Search By Link	✗	✓	✗
11	Removal of duplicate records	✓	✓	✓
12	User Sign In Required	✓	✗	✗
13	Result Set Accuracy (keyword: software testing)	100%	75%	100%
14	Stop word checking	✓	✗	✗

As we can see from above, NDSU Scholar gives fairly good results and also provides with most of the functionalities similar to Google Scholar and Microsoft Academic Search. My application also returns relevant articles to a search query similar to the result of Google Scholar and Microsoft Academic Search.

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

Developing a search engine is a very challenging task because of the complexity of the data available on the internet and how to incorporate this data into a fully functioning application that returns concise results based on a user input. There are other services provided by various organizations that let you search for articles from their database but it is very difficult to reverse engineer such applications because of lack of documentation of their implementation. Google offers its own custom search engine API (discussed in Chapter 2) that lets a user incorporate this API into their code. But that is still not what I was trying to achieve in this research, therefore I had to create my own search interface and design a process to fill this database with correct bibliographical information.

NDSU Scholar is a dynamic search engine that searches for articles relevant to a research problem in the field of computer science. I was successfully able to create a user friendly GUI for searching computer science articles with maximum accuracy to the user query. The GUI is compatible on most modern web browsers (Firefox, Chrome, Internet Explorer) with easy navigation for better usability for the user. After a detailed comparison with Google Scholar and Microsoft Academic Search, it was concluded that NDSU Scholar resembles closely to Google Scholar GUI. The output of search results are formatted for easy readability (**Title of the Paper**, **Author Names**, **Link to the paper**, **Year** and **Number of pages**) for the users and is paginated to provide the users with relevant results based on different filters and sorting choices or options.

The entire search engine was designed from scratch that included the database. The entire process is very efficient in returning the result set to the user. The database utilizes FULL TEXT index columns to perform faster searches. The parser of NDSU Scholar uses defensive programming technique which ensures that all aspects of error handling and exceptions are covered. This provides concrete and concise data for the database which in turn gives better results to the user. The GUI provides flexibility for the administrator to input a BibTeX link and parse the required data out to the database.

6.2. Future Work

NDSU Scholar is currently running on localhost and hence future work for the application would be deploying it on NDSU servers. The deployment has to be successfully done in order to make the application live for real time users who can then search for articles. We need to get the required approvals for this step from the NDSU CS-Department. Once we get approvals, we need a PHP server and MySQL database server in order to deploy NDSU Scholar.

To make the searching faster and relevant to a user, the concept of cache can be introduced to the database side which would include a cache table that would consist of the username and their 20 most recent searches. Since we are capturing each user that logs in to the system, we can build this table and then store that first 20 searches based on a count that will be incremented on every search. In this way, the program can be designed to retrieve the result set based on this table for the most recent search queries.

Other changes to the application can be incorporating JavaScript for the search interface to give suggestions to users as they type their search query. Some basic GUI changes can be made to the search interface in order to enhance the outlook and style of the elements placed. The authentication portion of the application can be removed and be replaced with NDSU-CAS to authenticate NDSU students, faculty and staff in more efficient manner.

Currently, the system does not check for the link that is already parsed in the system. This probes a problem to the database because any duplicate links parsed can add duplicate data to the database table which can decrease the accuracy of the search results. Therefore this functionality needs to be incorporated in the administrator end for future work.

The administrator has the ability to input one link at a time to parse the data to the database. Another future work will include automating this parsing process so that instead of the administrator manually importing data, a dynamic web crawler can parse the entire links on a particular webpage.

The current parsed data is taken from BibTeX format which is a limitation to the database because the parser can only process data if the format is correct. Another future work will include additions of different formats such as amsref, LaTeX which will increase the size of the database and therefore return more results to the user.

7. REFERENCES

1 Jones, S., Johnson-yale, C., Pérez, F.S., and Schuler, J.: ‘The internet landscape in college’, Yearbook of the National Society for the Study of Education, 2007, 106, (2), pp. 39-51

2 Griffiths, J.R., and Brophy, P.: ‘Student searching behavior and the Web: Use of academic resources and Google’, 2005, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.172.8640

3 Interactive, H.: ‘How academic librarians can influence students' web-based information choices: OCLC white paper on the information habits of college students’ (OCLC, 2002. 2002)

4 Brabazon, T.: ‘The Google effect: Googling, blogging, wikis and the flattening of expertise’, Libri, 2006, 56, (3), pp. 157-167

5 Lorence, D., and Abraham, J.: ‘Comparative analysis of medical web search using generalized vs. niche technologies’, Journal of medical systems, 2006, 30, (3), pp. 211-219

6 Winder, D.: ‘Online tools: Blogs and wikis’, Information World Review, 2006, 229, pp. 28-31

7 Orduña-Malea, E., Ayllón, J.M., Martín-Martín, A., and López-Cózar, E.D.: ‘About the size of Google Scholar: playing the numbers’, arXiv preprint arXiv:1407.6239, 2014

8 Khabsa, M., and Giles, C.L.: ‘The number of scholarly documents on the public web’, 2014, journals.plos.org/plosone/article?id=10.1371/journal.pone.0093949

9 Orduna-Malea, E., Ayllon, J.M., Martin-Martin, A., and Lopez-Cozar, E.D.: 'Empirical evidences in citation-based search engines: is Microsoft Academic Search dead?', arXiv preprint arXiv:1404.7045, 2014

10 Beebe, N.: 'The authors collaboration network in computational geometry was produced from the BibTeX bibliography available at http://www. math. utah. edu/~ beebe/bibliographies. html', The network data is available at http://vlado. fmf. uni-lj. si/pub/networks/data/collab/geom. htm, 2002

11 Ortega, J.L., and Aguillo, I.F.: 'Microsoft academic search and Google scholar citations: Comparative analysis of author profiles', Journal of the Association for Information Science and Technology, 2014, 65, (6), pp. 1149-1156

8. BIBLIOGRAPHY

- http://www.math.utah.edu/~beebe/bibliographies.html
- https://cse.google.com/cse/
- http://www.coderanch.com/t/100567/patterns/UML-Sequence-Diagram-database
- http://www.uic.edu/classes/bhis/bhis510/lim3/orgknow.htm
- http://www.BibTeX .org/
- http://websearch.about.com/od/enginesanddirectories/tp/custom-search-engines.htm
- http://pear.php.net/