Final Year Project Report

BS (COMPUTER SCIENCE)

Intelligent Web Spider for Efficient Search Engine

Submitted by

Abdullah Rather	**(Group Leader)**	**REG. # 35472**
Bilal Ahmed	**(Group Member)**	**REG. # 35708**
Muhammad Hassaan	**(Group Member)**	**REG. # 35387**
Syed Farid Uddin	**(Group Member)**	**REG. # 35485**

Project Supervisor

Sir Israr Ali

Project Coordinators

Dr. Mansoor Ebrahim & Dr. Atiya Masood

FACULTY OF ENGINEERING, SCIENCE and TECHNOLOGY

IQRA UNIVERSITY, KARACHI

2021

ABSTRACT

In today's modern world, where social media and other information sources are producing humongous data every moment, it is the need of the hour to build smart and efficient searching tool that should be able to collect keywords from websites, perform some filtration to purify repetitive / non-required data and then store collected keywords in a database so that searching becomes fast and storage space may be reduced. To achieve this aim, we decided to study and develop an application with the name of **Intelligent Web Spider for Efficient Search Engine** in order to realize our concept into reality.

This application has huge benefits across versatile industries ranging from government organization to defense forces and also for corporate world., The application follows a process by navigating various websites in order to fetch their contents, collect keywords from any website around the world, apply linguistic algorithm to filter the keywords, test links present on the website to identify its valid structure/hierarchy, means if some website has large number of pages, then it is a headache to check the correctness of hierarchy of that site. Whereas our application makes it easier for the user to test the correctness of hierarchy. While crawling through the website, it can also identify bad links (the links which are invalid or are without target pages), stores Clustered based keyword in database for efficient retrieval followed by maintaining ratings against each keyword.

DECLARATION

I hereby declare that the work has been done by myself to fulfill the requirement of the BS (Computer Science) and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

I hereby further declare that in the event of any infringement of the provision of the Act whether knowingly or unknowingly the university shall not be liable for the same in any manner whatsoever and undertake to indemnify and keep the university indemnified against all such claims and actions.

< student's signature>

© STUDENT NAMES [ID]

Abdullah Rather [35472]

ACKNOWLEDGEMENT

First, we thank Almighty Allah who praise us with the ability to think, work and deliver what we are assigned to do. Secondly, we are grateful to our Project Coordinators
"**Dr. Mansoor Ebrahim**" & "**Dr. Atiya Masood**" who helped us and guided us to complete the project successfully, and for the valuable effort they made during this course, to help us achieve a great experience of outcome as a successful software project at the end. For the support, they had given towards the success of this project and for giving us the domain knowledge and providing necessary information and documents regarding the project. We also acknowledge our teachers that throughout our studies helps us and guides us, departmental staff, university staff or other then this. We wish to express our gratitude to our project supervisor "**Sir Israr Ali",** who gave constant supervision despite his busy schedule, for the technical guidance provided to help us achieve the correct path, and without him this would not have been possible. We are also grateful to our family and friends, for supporting and encouraging us to complete this project successfully. Finally, we would like to thank all the colleagues of Iqra University who have been with us in all difficult times with suggestions and support which carry us to make this project a reality.

TABLE OF CONTENTS

LIST OF TABLES

Name **Page #**

LIST OF FIGURES

Name **Page #**

LIST OF ACRONYMS

1. WWW	World Wide Web
2. SE	Search Engine
3. WS	Web Spider
4. INT	Intelligent
5. ALGO	Algorithm
6. DB	Database
7. IWS	Intelligent Web Spider
8. HTTP	Hypertext Transfer Protocol
9. UI	User Interface
10. C	Consonant
11. V	Vowel

CHAPTER 1: INTRODUCTION

1.1 Introduction

Now-a-days, the databases for search engines are being populated by using automated tools called web crawlers. These tools search the keywords on each site and maintain the database based on search results.

These tools can also be used to see the statistics and content of any site for search purposes. But these tools are costly and inaccessible for students and small level professionals to use, and these tools do not provide the complete access and authentication for users, e.g., User can't restrict the level of depth of search, or any specific domain to search from.

Secondly, the keywords generated by such tools are very large in number from each website and a huge repetition of similar keywords are also generated like normal websites are based on a single business domain and it is very common that similar words are used on multiple pages. Therefore, the keywords generated on each page are also repeated. [1]

Another big challenge is the storage of generated keywords in a way that is efficient for future searching.

1.2 Project Objectives:

1.2.1 Motivation

Efficient and accurate searching from a huge data set of websites available on the web nowadays is a gigantic challenge. Number of universities and various organizations worldwide are doing extensive research to find out ways to make systems that can search the desired information from the widespread information available on the internet in minimum possible time with accurate results. In the recent past if one looks behind, the technology giant Google started its journey just by providing a revolutionary search engine that became so much popular that today it is the top ranked IT organization of the world beating behind many technology giants of that era. [2]

While keeping the importance of efficient searching in mind, we started exploring ways to provide raw material that search engines can utilize to enhance their efficacy and generate better results while maintaining the lowest possible footprint and storage size.

1.2.2 Goal:

To build a fast and reliable web crawler by utilizing linguistic algorithms for efficient storage of keywords.

1.2.3 Objectives:

i. To identify a viable linguistic algorithm that could simplify and transform keywords so that storage could be minimized without compromising search efficiency of a search engine.

ii. To do the research in exploring linguistic algorithms that are best suited for our needs.

iii. To explore efficient storage mechanisms for keywords in database clusters.

1.2.4 Problem:

Search engines use keywords to find results from the web. These keywords are collected by Web Crawlers. The duty of a crawler is to crawl through websites automatically, analyze the data written in text format and save this data into manageable hierarchies by applying clustering techniques. This saved data will be accessible to Search Engines to search results based on user's query and list related web pages.[3]

The databases for search engines are being populated by using automated tools. These tools search the keywords on each site and maintain a database based on search results. These tools can also be used to analyze statistics and content of any site for searching purposes. But the problem is their cost, these tools are costly and inaccessible for students and small level professionals. Moreover, these tools do not provide the flexibility to define level of depth for searching or specific domain level searching / keyword generation.[4]

So, a tool to answer these problems is needed that will provide fast and accurate search results for general users based on their own interest by using some intelligent searching techniques.

1.3 Structure of the Report:

Remaining Chapters of this report includes:

- **Chapter 2: <u>Background</u>**
  - This chapter consists of the background information necessary to understand the concepts presented in this report. And consist of methodology used and requirements to solve the problems described in the previous chapter.
- **Chapter 3: <u>Project Plan & Initial Design</u>**
  - This chapter consists of initial design or Prototype of UI of the system, and also describes how to complete the project in the given timeframe.
- **Chapter 4: <u>Design & Specification</u>**
  - This chapter contains an explanation of the actual project designs including DFDs, ERDs, Data Dictionary, Use Cases, and implementation of the system.
- **Chapter 5: <u>SYSTEM PROTOTYPE and DEVELOPMENT</u>**
  - This chapter contains Algorithm used in development of the system, protype design, frontend, and backend designs, database queries, and screenshots of the system.
- **Chapter 6: <u>RESULT ANALYSIS and TESTING</u>**
  - This chapter will contain unit testing of the system in terms of the functional requirement of the system.
- **Chapter 7: <u>Conclusion</u>**
  - This chapter will conclude the report also describing the problems/ challenges faced, and the future work or improvements that could be implemented on the system.

CHAPTER 2: BACKGROUND

2.1 Introduction:

In this chapter we introduced the implementation phase of Search Engine with Intelligent Web Spider. This includes Tools and Technologies we have chosen for the development of this software system and the operating system we used for development environment all of this information is mentioned in this chapter.

2.2 Overview of Basic Technologies

Our project has two distinct areas which are as follows:

a. **Desktop Application (Spider/Crawler)** that extract keywords from the websites and maintains a Database of keywords along with their frequencies

b. **Web Search Engine**, to search from keyword database

For the development of our **Desktop Crawler**, we will use the following tools and technologies:

1. **Java 1.8.0**

   Java is a very powerful language that is available since 1995 and serving a community base of more than 3 billion. It is extremely popular for the development of Desktop Applications, Mobile applications (especially Android apps), Games, Database driven Applications and many other domains. [2.1]

   It is one of the most popular programming language in the world, due to the facts that it is Open Source, Platform Independent (Windows, Mac, Linux, Android etc.). It is easy to learn and simple to use. Moreover, it is Object Oriented, Secure, Fast, Powerful and has a huge community support (tens of millions of developers) [2.2]

2. **Apache NetBeans IDE 12.0**

   Apache NetBeans is an Open-Source Integrated Development Environment. That is used to develop Java Based Applications of different flavors. It is very powerful and

fully support various domains of Java development like Desktop Applications Development, Mobile Apps, and Database Driven etc.

3. **MySQL Database**

MySQL is an open-source, fast reliable, and flexible relational database management system, typically used with PHP. Commonly used for developing Desktop and Web-based software for both small and large applications. IT supports standard SQL (Structured Query Language). It is also available in the market since 1994. Websites like Facebook, Wikipedia, Google (not for search), YouTube, Flickr, WordPress, Drupal, Joomla etc. uses MySQL DB.

For the development of our **Web Based Search Engine**, we will use the following tools and technologies:

1. **PHP 5.3**

PHP is an open-source HTML-embedded server-side scripting language that is used to develop dynamic and interactive web applications and also used as a general-purpose programming language.

PHP is extremely easy to learn and use and also has many advanced features for a professional programmer. It runs efficiently on the server-side and works on many operating systems such as Linux, Windows, Mac etc. [2.3]

2. **Adobe Dreamweaver**

Adobe Dreamweaver is a proprietary web development tool from Adobe Inc. It was created by Macromedia in 1997 and developed by them until Macromedia was acquired by Adobe Systems in 2005. It is a web designing and an Integrated Development Environment (IDE) application that is used to design and develop websites.

3. **WAMP Server 2.0**

WAMP "Windows, Apache, MySQL, and PHP." WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and

PHP). It is often used for web development and internal testing, but may also be used to host live websites. It is a Web development platform on Windows that allows you to create dynamic Web applications

4. **Web Browser**

   Web Browser (IE, Firefox, Chrome etc.) are used to run the web application and also test PHP based source code.

2.3 Background Research

The process is called web crawling or spidering. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a website, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for spam).
A web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies.

A. Architecture of Web Crawler

A crawler must not only have a good crawling strategy, but it should also have a highly optimized architecture. As suggested in [2.5] "While it is fairly easy to build a slow crawler that downloads a few pages per second for a short period of time, building a high-performance system that can download hundreds of millions of pages over several weeks presents a number of challenges in system design, I/O and network efficiency, and robustness and manageability."

Generally, the architecture of common Web Crawler has two major processors i.e. Scheduler and Multi-threaded Downloader. The job of downloader is to initiate and

send HTTP request to server (or Internet) for downloading content of Web pages. Then the URLs and Text/metadata is separated from the downloaded content. This separated text/metadata is to be stored on Storage location (i.e. database). And the list of URLs will be passed to Scheduler for recursively download.

B. How it Works

The process start by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and so on, recursively. Web Crawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. It resides on a single machine and simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links.

C. Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The algorithm has been a long-standing problem in computer science; the first paper on the subject was published in 1968. The process of stemming, often called conflation, is useful in search engines for query expansion or indexing and other natural language processing problems.[2.6]
Stemming programs are commonly referred to as **stemming algorithms** or **stemmers**.

A stemmer for English, for example, should identify the string "cats" (and possibly "catlike", "catty" etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". A stemming algorithm reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish".

D. History of Stemmers

The first ever published stemmer was written by Julie Beth Lovins in 1968 [2.7]. This paper was remarkable for its early date and had great influence on later work in this area. A later stemmer was written by Martin Porter and was published in the July 1980 issue of the journal Program. This stemmer was very widely used and became the de-facto standard algorithm used for English stemming. Dr. Porter received the Tony Kent Strix award in 2000 for his work on stemming and information retrieval.

Many implementations of this algorithm were written and freely distributed; however, many of these implementations contained subtle flaws. As a result, these stemmers did not match their potential. To eliminate this source of error, Martin Porter released an official free-software implementation of the algorithm around the year 2000. He extended this work over the next few years by building Snowball, a framework for writing stemming algorithms, and implemented an improved English stemmer together with stemmers for several other languages. [2.6]

E. Stemming Algorithms

There are several types of stemming algorithms which differ in respect to performance and accuracy, some of these are listed here:

- Brute Force Algorithms
- Porter's Stemmer
- Suffix Stripping Algorithms
- Lemmatisation Algorithms

2.4 Summary

Above is the technological overview of our project, covering tools and technologies that we are using along with Stemming Algorithms that we have identified to incorporate in our project in order to enhance the storage capabilities of keywords generated by our crawler.

CHAPTER 3: PROJECT PLAN AND INITIAL DESIGN

3.1 Introduction:

In this chapter we introduced the detailed timeframe for the completion of the Project and Hardware Requirements to run the system. Discussing all modules mentioned in the activity schedule and time required to complete each one of them. Also including Functional and Non-Functional requirements of the system all this information is mentioned in this chapter.

3.2 Summary of Activity Schedule

Based on the estimated time, the project requires 14 weeks for completion. In the 1st two weeks we complete design specification milestone. In the 3rd week we do environment preparation Installation of Java, PHP and MySQL in the system, in 4th week we cover Technical Design which consist of Overview of basic Technologies and then from 5th to 10th week we start Application Development, initially starting with studying and finalization of Linguistic Algorithm after moving on to Development of Crawler Module and then start Development of User Interface, implementation of business logic and finally integration between Crawler and Linguistic Algorithm. After all previous tasks are completed, then in 11th week we start Application Testing in it doing Unit Testing and complete application testing and bugs fixing, finalizing it till end of 12th week. Then in 13th we start User Documentation in it providing User manual and installation guide. And finally, in the 14th week we give Application Demonstration covering Project conclusion, Future Work and Application Finalization.

3.2.1 Gantt Chart

ID	Task Name	Duration	Start	Finish	Predecessors
1	**Intelligent Web Spider for Efficient Search Engine**	**70 days**	**Mon 01-03-21**	**Fri 04-06-21**	
2	Design Specification	2 wks	Mon 01-03-21	Fri 12-03-21	
3	**Environment Preperation**	**5 days**	**Mon 15-03-21**	**Fri 19-03-21**	
4	Configuring Dev Environment (Java, PHP, MySQL)	1 wk	Mon 15-03-21	Fri 19-03-21	2
5	Preparation of Project Skeleton	1 wk	Mon 15-03-21	Fri 19-03-21	2
6	**Designing**	**5 days**	**Mon 22-03-21**	**Fri 26-03-21**	
7	Application Architecture & Design	1 wk	Mon 22-03-21	Fri 26-03-21	5
8	Database Design & Development	1 wk	Mon 22-03-21	Fri 26-03-21	5
9	**Application Development**	6 wks	Mon 29-03-21	Fri 07-05-21	8
10	Testing	2 wks	Mon 10-05-21	Fri 21-05-21	9
11	Beta Release	2 wks	Mon 10-05-21	Fri 21-05-21	9
12	Making of User Guide	1 wk	Mon 24-05-21	Fri 28-05-21	11
13	Demonstration to Project Committee	1 wk	Mon 31-05-21	Fri 04-06-21	12

[Fig. 3.1] Gantt Chart

3.3 Functional Requirements

1. Sign up

1.1 Description

The user will be asked to provide their information for registration.

1.2 Functional Requirements

- User shall provide a valid email address/phone number to receive a one-time code for verification of their account.
- User shall create a password to be able to sign in to their account after completing the registration.
- If the password created by the user don't match in password/confirm password section, system shall display a message to the user indicating that 'password do not match'.

- If the user provides invalid code send to their email address, the system shall display a message to the user stating, ‘invalid code’.
- After successful sign up, system shall redirect user to sign in page.

2. Sign in

2.1 Description

The user will be asked to enter their account credentials to get access to the application.

2.2 Functional Requirements

- User shall only be able to Sign in with a valid registered account.
- If the user enters invalid sign in credentials, system shall display a message, ‘Account not found’
- After successful login, system shall redirect user to Dashboard page.

3. Dashboard

3.1 Description

From Dashboard, user can access main features of the application.

3.2 Functional Requirements

- System shall display a welcome message on the top right corner of the page, mentioning the logged in user, e.g. ‘Welcome, Abdullah’

- If the user clicks on ‘Profile’ button, system shall redirect the user to Profile page.
- If the user clicks on “List of users” button, system shall redirect the user to list of user’s page.
- If the user clicks on ‘Crawler’ button, system shall redirect the user to Crawler page.
- If the user clicks on ‘Account Security’ button, system shall redirect the user to Account security page.
- If the user clicks on Logout button, system shall terminate their session.

4. List of Users

4.1 Description

List of users feature stores all the registered users on system.

4.2 Functional Requirements

- Only the authorized user shall have access to “list of users” functionality.
- The authorized admin user shall have the privileges to give/take access of certain features from other non-admin users.

5. Profile

5.1 Description

User shall be able to access their profile details in the profile feature.

5.2 Functional Requirements

- If the user clicks on profile button from the dashboard, system shall redirect the user to the profile page.
- The profile page shall contain the details of the user, e.g. Name, Phone No, Email, etc.

6. Account Security

6.1 Description

The user shall be able to access and modify their credentials in account security feature.

6.2 Functional Requirements

- If the user clicks on account security button from the dashboard, system shall redirect the user to account security page.
- The account security page shall contain an option for the user to modify their credentials.
- If the user selects change profile option, system will allow the user to modify his/her profile. This will include public profile. However, email address which will be the unique user name in our system will not be allowed to change.
- If the user clicks on change password button, system must first ask the user their current password then procced on to the new password setup.

7. Crawler

7.1 Description

In crawler option System shall allow the user to perform in depth crawling on websites. It is the High priority or primary feature of the system, because without crawling the raw data can't be collected for maintaining Keyword Database

7.2 Functional Requirements

- In Crawler page, under URL option user shall provide the website on which they want crawling to be performed.
- User shall specify the level of depth of search.
- System shall display crawling progress to the user
- Total number of URLs found at the current visiting web page should be visible to administrator.
- If the user clicks stop crawl button, system shall stop crawl.
- System should transform keywords generated into simplified words by processing the keywords through Linguistic Algorithm.
- If the user clicks save crawl button, system shall save the Keywords of crawled pages within their respective Clusters.
- If the user clicks back button, system shall take the user back to the dashboard screen.
- After the crawling process is completed, system shall display the list of crawled keywords to the user.

8. Stemming the Key Words

8.1 Description

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

Stemming the keywords is useful in search engines for query expansion or indexing and other natural language processing problems. So it has High priority in our system.

8.2 Functional Requirements

- Remove the plurals, -ed and –ing from words.
- Replace terminal y to i when there is another vowel in the word.
- Maps double suffices to single ones, e.g. -ization (= -ize plus -ation) maps to -ize etc
- Remove characters like: -ic-, -full, -ness, -icate, -alize, -ative, -ical, from the word.
- Takes off -ant, -ence etc., in context <c>vcvc<v>, (where c=consonant, v=vowel).
- Remove a final –e from end of word, if required.

9. Clustering of Keywords

9.1 Description

Clustering is the classification of objects/keywords into different groups, or more precisely, the partitioning of a keywords set into categories (or clusters) for better storage and efficient searching.

Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, search engines, pattern recognition, image analysis and bioinformatics.

9.2 Functional Requirements

- Minimum frequency of keyword at a specific page should be greater than the number defined by administrator.
- Categories (i.e. cluster types) for keywords should be predefined on the basis of distances.

.

10. Logout

10.1 Description

Logout feature is for the user to terminate their session on the application.

10.2 Functional Requirements

- The system shall provide users with an option to terminate their session by logging out of the system.

- The system shall remove session data if the user logs out of the system.

11. Search Engine

11.1 Description

Through Search Engine users can search the Keywords from the local Data Bank generated by the Crawler while crawling different websites.

11.2 Functional Requirements

- User shall enter the desired search term into the search field.
- The search engine shall look through its Data Bank for relevant websites and display them in the form of list to the user.
- System should transform keywords generated into simplified words.
- System should Search keywords from respective Database Clusters.
- System should display Search Results on a Search Page.

3.4 Non-Functional Requirements

a. Operational Requirements

- Software should be available and workable as and when required to the user.
- System should be cost effective.
- System should be flexible to accommodate future releases / enhancements

b. Performance Requirements

- A reliable internet connection to fetch keywords form website as the performance is dependent on bandwidth.
- System should be efficient to Crawl multiple webpages efficiently and without errors
- System should utilize available bandwidth without breaking the internet connection

c. Security Requirements

- Only authorized persons/administrators should be able to run this system.
- Keywords extracted by Crawlers should be securely stored within Database Clusters

d. Cultural and Political Requirements

- None

3.5 Hardware Requirements

- Internet Connection Minimum 4 Mbps; Recommended 8 Mbps or above.
- Memory (RAM): Minimum 4 GB; Recommended 8 GB or above.
- PC Core i5 Quad Core Processor, 1 Terabyte Hard disk

3.6 Summary

Above is the Project plan and initial design of the project covering overview of the timeline, Gantt chart, complete Functional and Non-Functional requirements of project and system's hardware requirements.

CHAPTER 4: DESIGN AND SPECIFICATION:

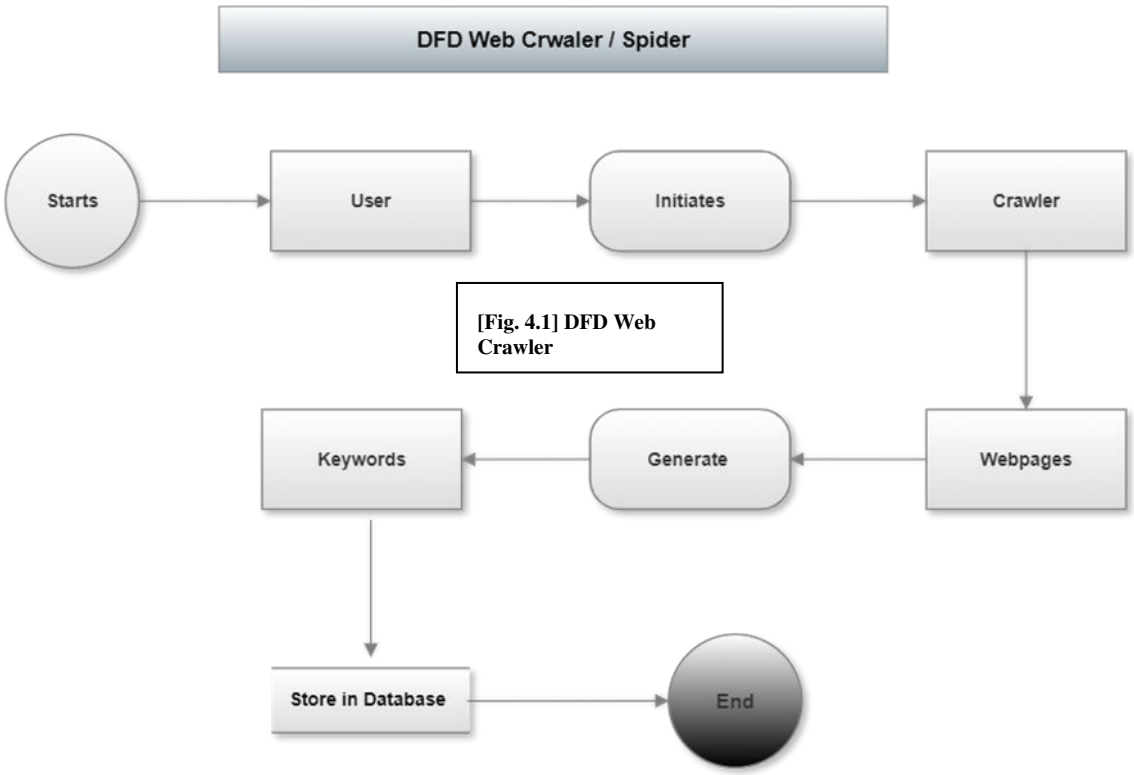
4.1 Introduction:

In this chapter we introduced the design and specification of Intelligent Web Spider for Efficient Search Engine. It includes artifacts that are produced in the design phase and how different system requirements were analyzed and mapped to the ultimate system. All these details are mentioned in this chapter. The purpose of this phase is to design the system specifications based on functional, non-functional requirements, our study based on literature review and the proposed system we have suggested based on all these artifacts.

4.2 Data Flow Diagram (DFD)

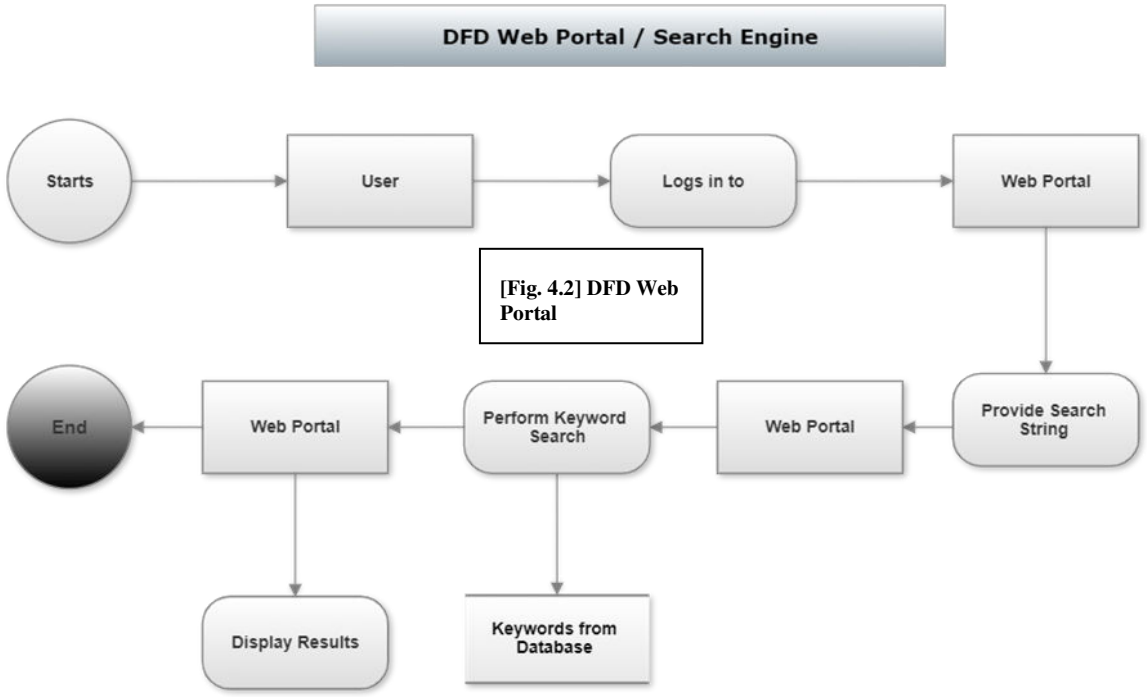
Entity Relationship Diagram of our project is shown below:

- **DFD Web Crawler / Spider**



[Fig. 4.1] DFD Web Crawler

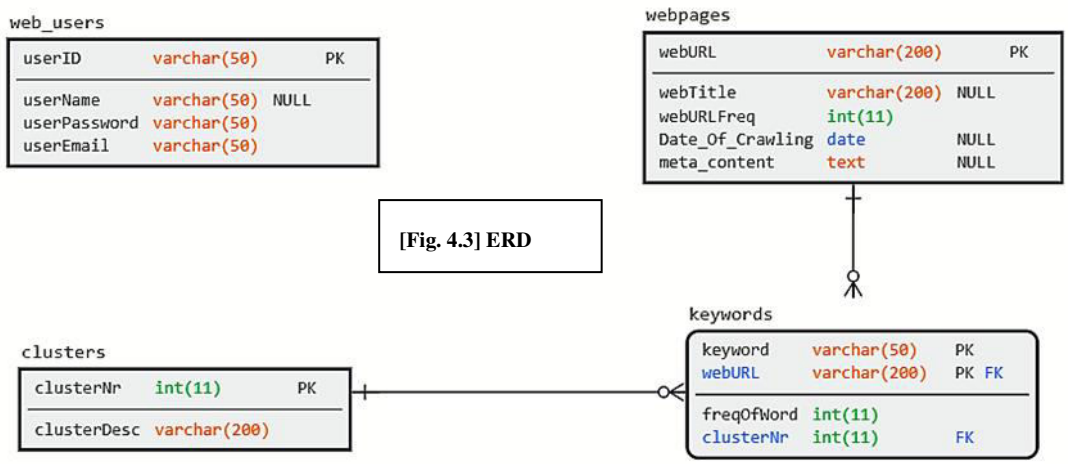
- **DFD of Web Portal / Search Engine**



[Fig. 4.2] DFD Web Portal

4.3 Entity Relationship Diagram (ERD)

Entity Relationship Diagram of our project is shown below:



[Fig. 4.3] ERD

Database of our project contains four main tables details of each table is as follows:

- **web_user**

  This table contains all registered users that will have the privilege to use our crawler. userID is the primary key in this table.

- **clusters**

  Our crawler will maintain multiple clusters to store keywords in order to maintain efficient storage and fast retrieval. These clusters will be stored in this table.
  ClusterNr is the primary key in this table. It has one to many relationship with keywords table

- **webpages**

  When our crawler starts crawling, all the webpages it will crawl will be stored in this page. If a webpage is present as an external link in the website, our crawler will also crawl that web page and identify keywords. e.g., a website www.abc.com has a link to www.xyz.com our crawler will crawl both the links and generate keywords.
  This table will maintain the URL, name of webpage it has crawled along with its frequency in the website and date when the crawling is actually happen.
  WebURL is the primary key in this table. It has one to many relationship with keywords table

- **keywords**

  All the keywords generated as a result of crawling will be stored in this table.
  It has a composite primary key based on keyword and webURL.

4.4 Normalized Tables

Our database is normalized to the best possible scenario.

- **First Normalization Form 1NF**

First Normalization Form 1NF is achieved by introducing primary key in all tables. This each records in uniquely identified through the primary key.

- **Second Normalization Form 2NF**

Second Normalization Form 2NF is achieved by complying to 1NF and all repeating records are stored in a separate table while maintaining a Foreign Key.

- **Third Normalization Form 3NF**

Third Normalization Form 3NF is achieved by complying 2NF and removing all Transitive Functional Dependencies.

4.5 Data Dictionary

Data Dictionary of the project is as follows:

- **Clusters**

Field	Type	Null	Default	Remarks
clusterNr	int(11)	No		
clusterDesc	varchar(200)	No		

Data Dictionary
Clusters [Table 4.1]

- **Keywords**

Field	Type	Null	Default	Remarks
Keyword	varchar(50)	No		
webURL	varchar(200)	No		
freqOfWord	int(11)	No		
clusterNr	int(11)	No		

Data Dictionary Keywords [Table 4.2]

- **web_users**

Field	Type	Null	Default	Remarks
userID	varchar(50)	No		
Username	varchar(50)	Yes	Null	
userPassword	varchar(50)	No		
userEmail	varchar(50)	No		

Data Dictionary web_users [Table 4.3]

- **webpages**

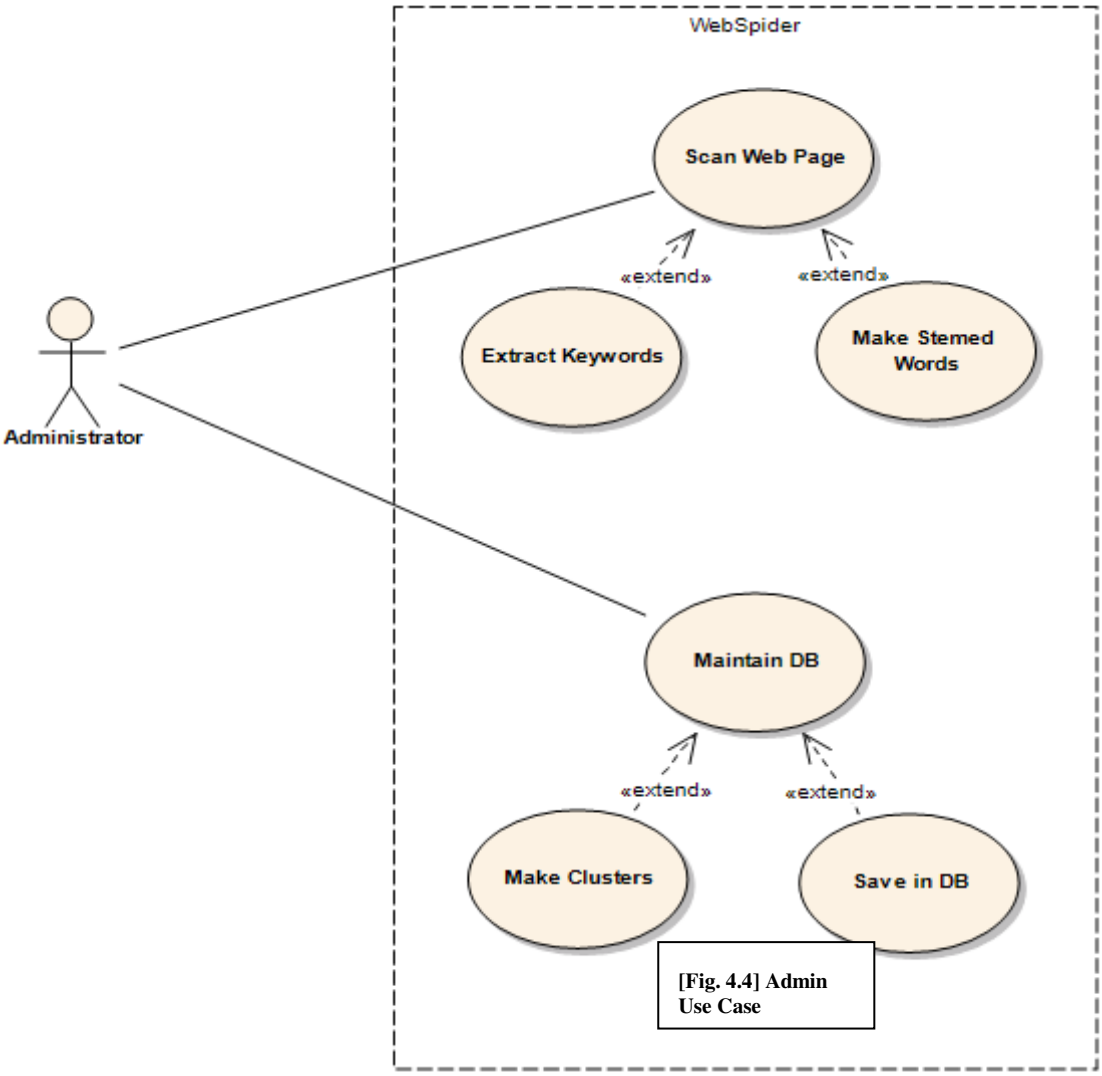
Field	Type	Null	Default	Remarks
webURL	varchar(200)	No		
webTitle	varchar(200)	Yes	Null	
webURLFreq	int(11)	No		
Date_Of_Crawling	Date	Yes	Null	
meta_content	Text	Yes	Null	

Data Dictionary webpages [Table 4.4]

4.6 Use Cases

Below are the use cases of our project

- **Administrative Use Case**



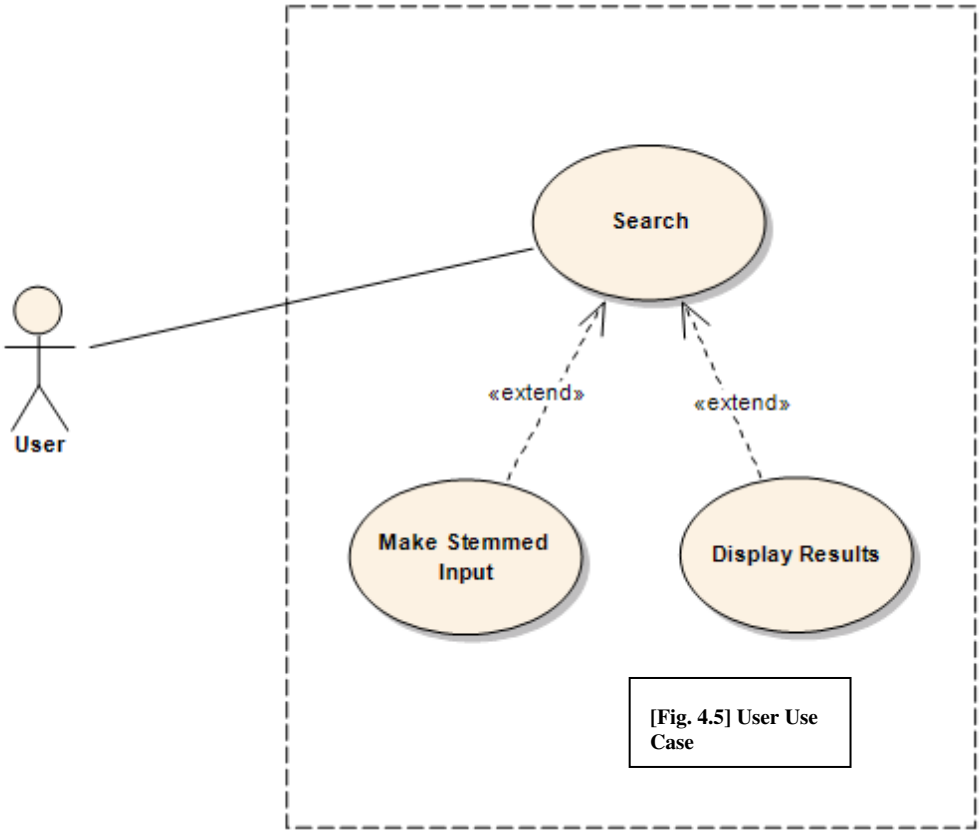
[Fig. 4.4] Admin Use Case

Use Case Name:	Web Spider
ID:	UC-LHD-01
Actors Involved:	Administrator
Brief Description	Administrator initiates web spider to start crawling the web pages and generate keywords. Then after stemming these keywords, store keywords in respective clustered database

Pre-Conditions	System is available for use	
Post-Conditions	Keywords stored in database	
Normal Flow of Events:	**Actor Action**	**System Response**
	1. Administrator starts the application by clicking a button. 2. Application starts crawling web pages 3. Keywords are granted and stored in respective clustered database	1. Application starts crawling web pages 2. Keywords are granted and stored in respective clustered database.

Admin Use Case [Table 4.5]

- **User Use Case**



[Fig. 4.5] User Use Case

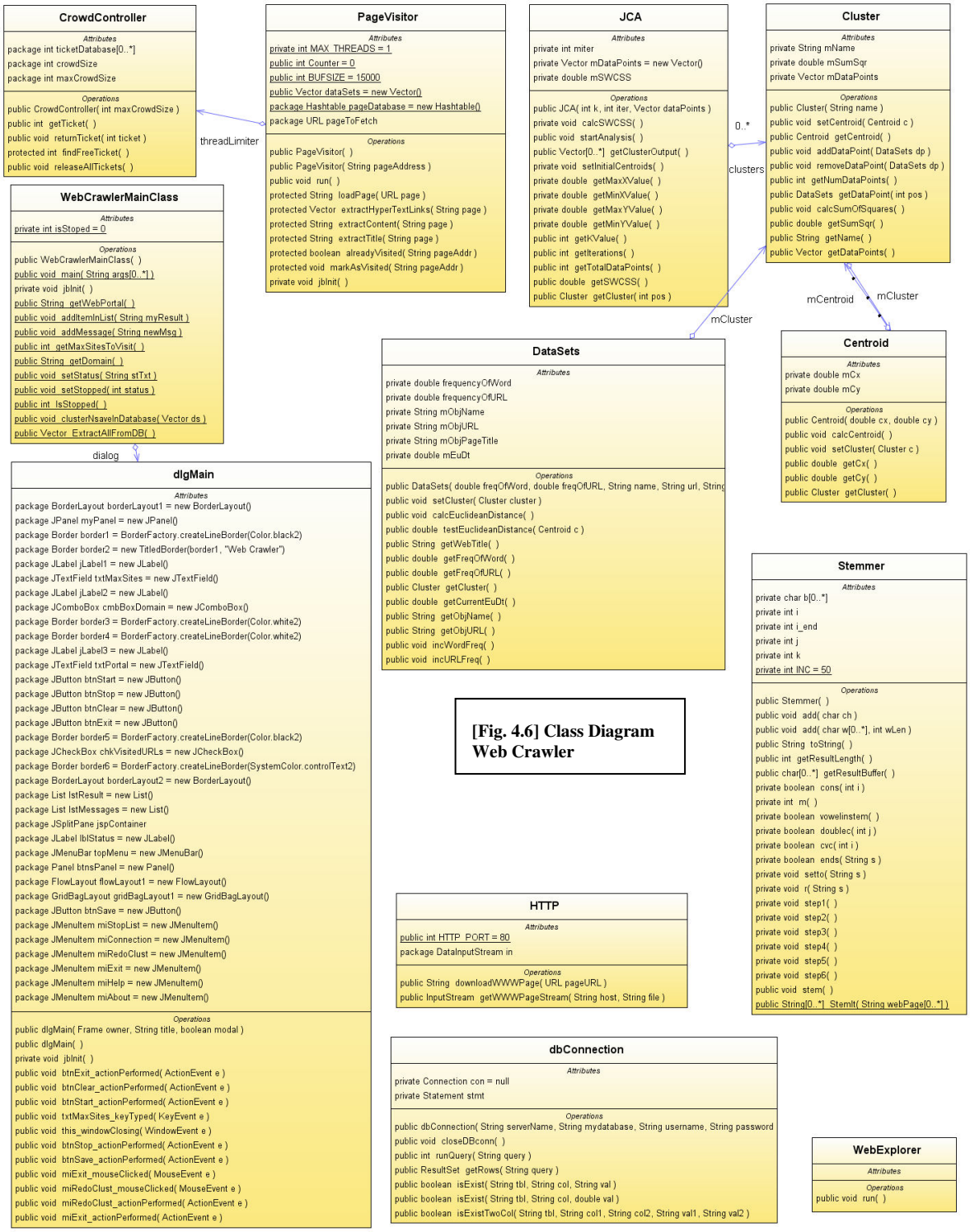
Use Case Name:	Search from keywords	
ID:	UC-CNT-002	
Actors Involved:	User	
Brief Description	Convert user input search criteria into stemmed input and apply searching from generated keywords	
Pre-Conditions	Use Case ID: UC-LHD-01	
Post-Conditions	Results are displayed.	
Normal Flow of Events:	**Actor Action**	**System Response**
	1. User provide search string 2. Press Search button	1. System will convert input into stemmed input 2. Apply search from keyword database 3. Display results

User Use Case [Table 4.6]

4.7 Class Diagram

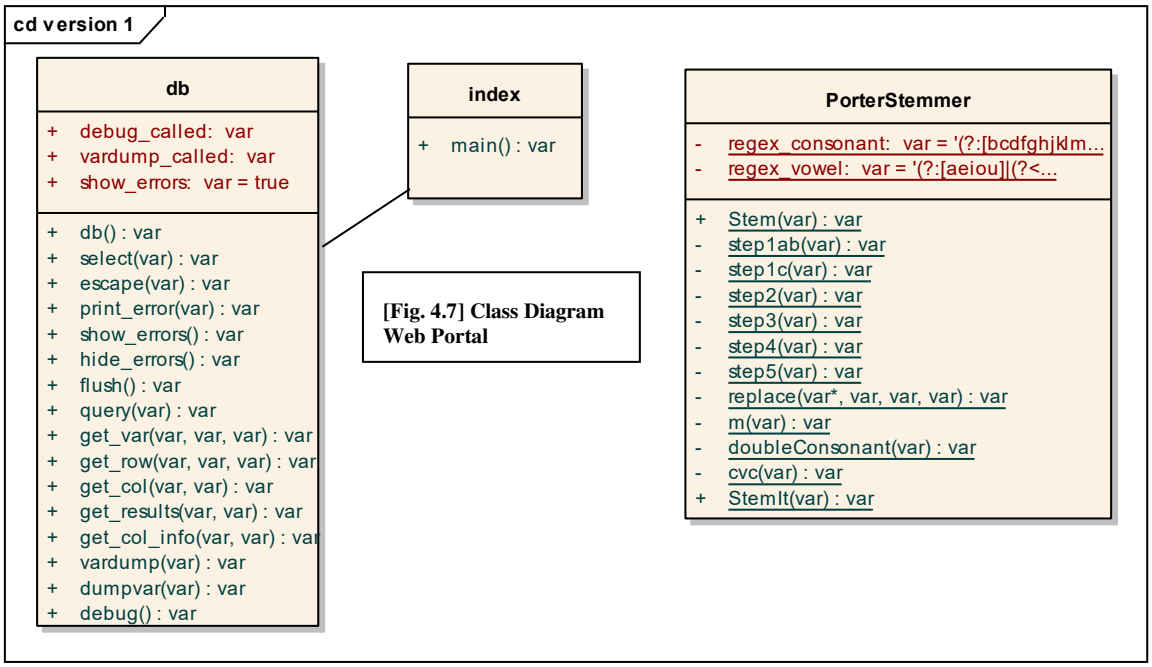
As the project has two main modules a **Web Crawler / Spider** and **a Web Search Engine / Web Portal**. Class diagrams of both modules are appended below:

- **Class Diagram of Web Crawler / Spider**



[Fig. 4.6] Class Diagram Web Crawler

- **Class Diagram of Web Portal**



[Fig. 4.7] Class Diagram Web Portal

4.8 Summary

Above are the design specifications of our project. These design specifications are based on the user requirements mentioned in previous chapters. We are developing our system based on these design specifications.

CHAPTER 5: SYSTEM PROTOTYPE AND DEVELOPMENT:

5.1 Introduction:

Crawler behave like a client to the web servers. It sends an HTTP request for any specified web URL. The resultant web page contains the data to be analyzed. This data is managed by HTML tags. So, our crawling application should have ability to identify HTML tags, and to fetch textual information written in those tags. As the fetched textual information is helpful to keep the record of keywords, and the associated URLs.

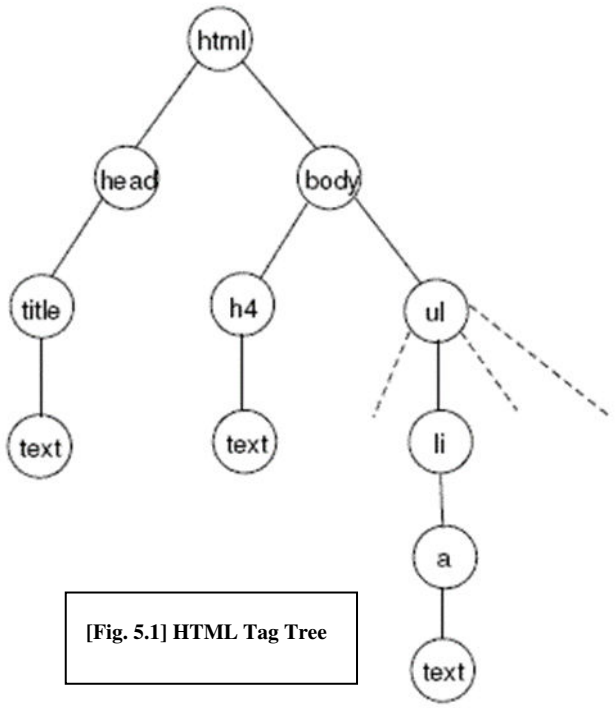
5.1.1 HTML Tag Tree

Crawlers assess the value of a URL or a content word by examining the HTML tag context in which it resides. For this, a crawler may need to utilize the tag tree or DOM structure of the HTML. Figure 5.1 shows a tag tree corresponding to an HTML source. The <html> tag forms the root of the tree and various tags and texts form nodes of the tree. Unfortunately, many Web pages contain badly written HTML. For example, a start tag may not have an end tag (it may not be required by the HTML specification), or the tags may not be properly nested. In many cases, the <html> tag or the <body> tag is all-together missing from the HTML page. Thus structure-based criteria often require the prior step of converting a dirty HTML document into a well-formed one, a process that is called tidying an HTML page. This includes both insertion of missing tags and the reordering of tags in the page. Tidying an HTML page is necessary for mapping the content of a page onto a tree structure with integrity, where each node has a single parent. Hence, it is an essential precursor to analyzing an HTML page as a tag tree. Note that analyzing the DOM structure is only necessary if the topical crawler intends to use the HTML document structure in a non-trivial manner. For example, if the crawler only needs the links within a page, and the text or portions of the text in the page, one can use simpler HTML parsers.

An Example of HTML Page Structure:

```
<html>
    <head>
        <title>Projects</title>
```

```
    </head>
    <body>
        <h4>Current Projects</h4>
        <ul>
            <li> <a href="lamp.html">LAMP</a> Linkage analysis with MP.</li>
            <li> <a href="ni.html">NI</a> Network Infrastructure.</li>
            <li> <a href="adna.html">ADNA</a> A DNA Algorithm.</li>
            <li> <a href="dlt.html">DLT</a> A distributed, first-order logic
            theorem.</li>
        </ul>
    </body>
</html>
```



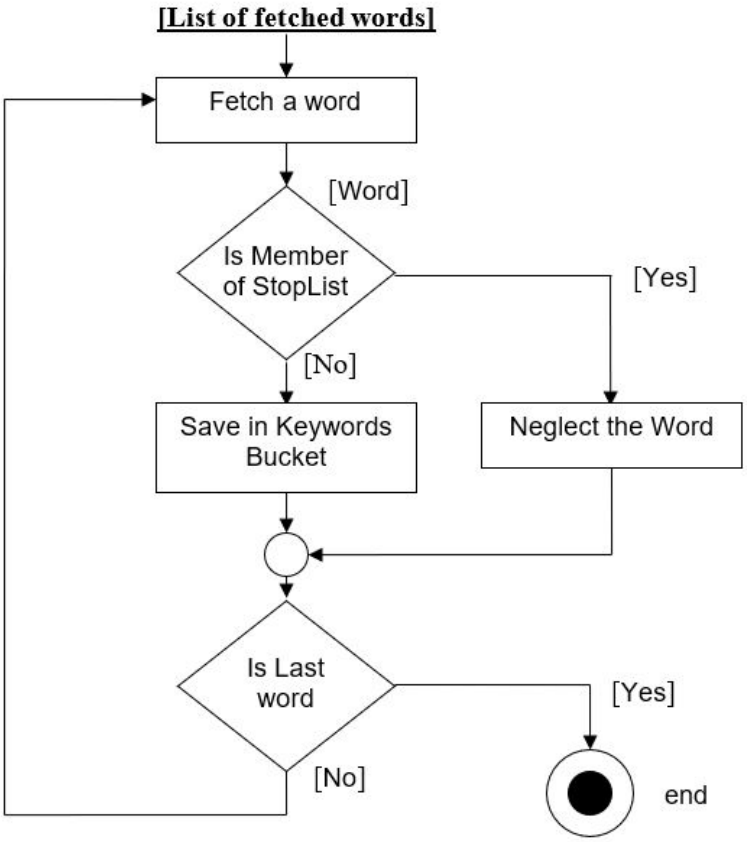
[Fig. 5.1] HTML Tag Tree

5.1.2 Words Filtering

The textual data fetched from the web page contains huge amount of text containing many keywords as well useless words. Our application has to filter the keywords by avoiding useless words (like A, Is, The, also, although, most, mostly etc.).

In our case the useless words can be defined as “All words, those are not derived from a Noun”. So, we have built a list of such words, called “Stop List”. This list is helpful for dropping common/useless words.

Our crawler parses all the words in textual information and perform search operation in the Stop List, if the word exists in Stop List, then Drop the word otherwise save the word in the bucket of Keywords.



[Fig. 5.2] Flow of Filtering Words

5.2 Stemming Algorithm

5.2.1 Introduction

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base, or root form — generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The algorithm has been a long-standing problem in computer science; the first

paper on the subject was published in 1968. The process of stemming, often called conflation, is useful in search engines for query expansion or indexing and other natural language processing problems. [5.1]

Stemming programs are commonly referred to as **stemming algorithms** or **stemmers**.

5.2.2 Stemming Example

A stemmer for English, for example, should identify the string "cats" (and possibly "catlike", "catty" etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". A stemming algorithm reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish".

5.2.3 Stemming History

The first ever published stemmer was written by Julie Beth Lovins in 1968 [5.2]. This paper was remarkable for its early date and had great influence on later work in this area.

A later stemmer was written by Martin Porter and was published in the July 1980 issue of the journal Program. This stemmer was very widely used and became the de-facto standard algorithm used for English stemming. Dr. Porter received the Tony Kent Strix award in 2000 for his work on stemming and information retrieval.

Many implementations of this algorithm were written and freely distributed; however, many of these implementations contained subtle flaws. As a result, these stemmers did not match their potential. To eliminate this source of error, Martin Porter released an official free-software implementation of the algorithm around the year 2000. He extended this work over the next few years by building Snowball, a framework for writing stemming algorithms, and implemented an improved English stemmer together with stemmers for several other languages [5.1]

5.2.4 Porter Stemming Algorithm

The Porter stemming algorithm (or ‘Porter stemmer’) is a process for removing the commoner morphological and inflexional endings from words in English. Its

main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems [5.3]

A consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. (The fact that the term 'consonant' is defined to some extent in terms of itself does not make it ambiguous.) So in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A **consonant** will be denoted by **C**, a **vowel** by **V**. A list ccc... of length greater than 0 will be denoted by C, and a list vvv... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

These may all be represented by the single form

[C]VCVC ... [V]

Where the square brackets denote arbitrary presence of their contents. Using (VC)m to denote VC repeated m times, this may again be written as

[C](VC)m[V].

m will be called the measure of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples:

m=0	TR, EE, TREE, Y, BY.
m=1	TROUBLE, OATS, TREES, IVY.
m=2	TROUBLES, PRIVATE, OATEN, ORRERY.

The rules for removing a suffix will be given in the form:

(condition) S1 -> S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m, e.g.

(m > 1) EMENT ->

Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

The 'condition' part may also contain the following:

*S	- the stem ends with S (and similarly for the other letters).
v	- the stem contains a vowel.
*d	- the stem ends with a double consonant (e.g. -TT, -SS).
*o	- the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

And the condition part may also contain expressions with and, or and not, so that

(m>1 and (*S or *T))

tests for a stem with m>1 ending in S or T, while

(*d and not (*L or *S or *Z))

tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely.

In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching S1 for the given word. For example, with

SSES	->	SS
IES	->	I
SS	->	SS
S	->	

(Here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally CARESS maps to CARESS (S1='SS') and CARES to CARE (S1='S').

In the rules below, examples of their application, successful or otherwise, are given on the right in lower case. The algorithm now follows:

Step 1a

SSES	->	SS	caresses	->	caress
IES	->	I	ponies	->	poni
			ties	->	ti
SS	->	SS	caress	->	caress
S	->		cats	->	cat

Step 1b

(m>0) EED	->	EE	feed	->	feed
			agreed	->	agree
(*v*) ED	->		plastered	->	plaster
			bled	->	bled
(*v*) ING	->		motoring	->	motor
			sing	->	sing

If the second or third of the rules in Step 1b is successful, the following is done:

AT	->	ATE	conflate(ed)	->	conflate
BL	->	BLE	troubl(ed)	->	trouble
IZ	->	IZE	siz(ed)	->	size
(*d and not (*L or *S or *Z))	->	single letter	hopp(ing)	->	hop
			tann(ed)	->	tan
			fall(ing)	->	fall
			hiss(ing)	->	hiss
			fizz(ed)	->	fizz
(m=1 and *o)	->	E	fail(ing)	->	fail
			fil(ing)	->	file

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognized later. This E may be removed in step 4.

Step 1c

(*v*) Y	->	I	happy	->	happi
			sky	->	Sky

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

Step 2

(m>0) ATIONAL	->	ATE	relational	->	relate
(m>0) TIONAL	->	TION	conditional	->	condition
			rational	->	rational
(m>0) ENCI	->	ENCE	valenci	->	valence
(m>0) ANCI	->	ANCE	hesitanci	->	hesitance
(m>0) IZER	->	IZE	digitizer	->	digitize
(m>0) ABLI	->	ABLE	conformabli	->	conformable
(m>0) ALLI	->	AL	radicalli	->	radical
(m>0) ENTLI	->	ENT	differentli	->	different
(m>0) ELI	->	E	vileli	->	vile
(m>0) OUSLI	->	OUS	analogousli	->	analogous
(m>0) IZATION	->	IZE	vietnamization	->	vietnamize
(m>0) ATION	->	ATE	predication	->	predicate
(m>0) ATOR	->	ATE	operator	->	operate
(m>0) ALISM	->	AL	feudalism	->	feudal
(m>0) IVENESS	->	IVE	decisiveness	->	decisive
(m>0) FULNESS	->	FUL	hopefulness	->	hopeful
(m>0) OUSNESS	->	OUS	callousness	->	callous
(m>0) ALITI	->	AL	formaliti	->	formal
(m>0) IVITI	->	IVE	sensitiviti	->	sensitive
(m>0) BILITI	->	BLE	sensibiliti	->	sensible

The test for the string S1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives a fairly even breakdown of the possible values of the string S1. It will be seen in fact that the S1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3

(m>0) ICATE	->	IC	triplicate	->	triplic
(m>0) ATIVE	->		formative	->	form
(m>0) ALIZE	->	AL	formalize	->	formal
(m>0) ICITI	->	IC	electriciti	->	electric
(m>0) ICAL	->	IC	electrical	->	electric
(m>0) FUL	->		hopeful	->	hope
(m>0) NESS	->		goodness	->	good

Step 4

(m>1) AL	->	revival	->	Reviv
(m>1) ANCE	->	allowance	->	Allow
(m>1) ENCE	->	inference	->	Infer
(m>1) ER	->	airliner	->	Airlin
(m>1) IC	->	gyroscopic	->	gyrosco p
(m>1) ABLE	->	adjustable	->	adjust
(m>1) IBLE	->	defensible	->	defens
(m>1) ANT	->	irritant	->	irrit
(m>1) EMENT	->	replacement	->	replac
(m>1) MENT	->	adjustment	->	adjust
(m>1) ENT	->	dependent	->	depend
(m>1 and (*S or *T)) ION	->	adoption	->	adopt
(m>1) OU	->	homologou	->	homolog
(m>1) ISM	->	communism	->	commun
(m>1) ATE	->	activate	->	active
(m>1) ITI	->	angulariti	->	angular
(m>1) OUS	->	homologous	->	homolog
(m>1) IVE	->	effective	->	effect
(m>1) IZE	->	bowdlerize	->	bowdler

The suffixes are now removed. All that remains is a little tidying up.

Step 5a

(m>1) E	->	probate	->	probat
		rate	->	rate
(m=1 and not *o) E	->	cease	->	ceas

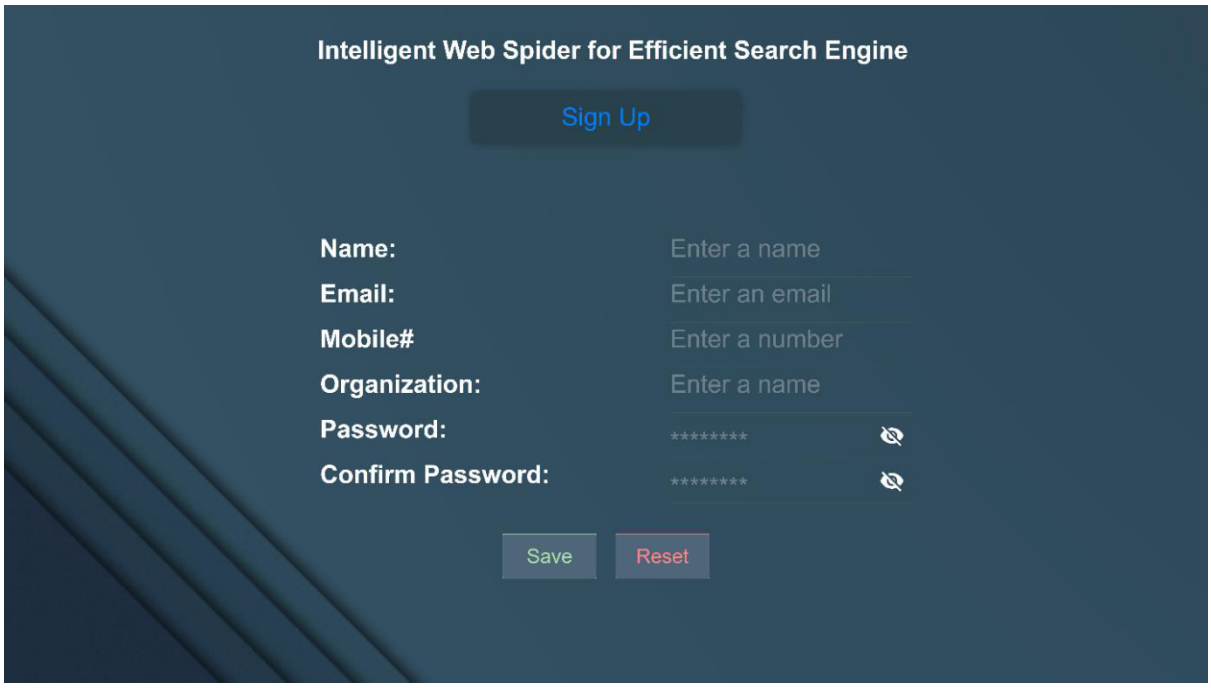
Step 5b

(m > 1 and *d and *L)	->	single letter	controll	->	control
			roll	->	roll

Above mentioned Algorithm and its Steps are taken from www.snowbal.tartarus.org [5.4]

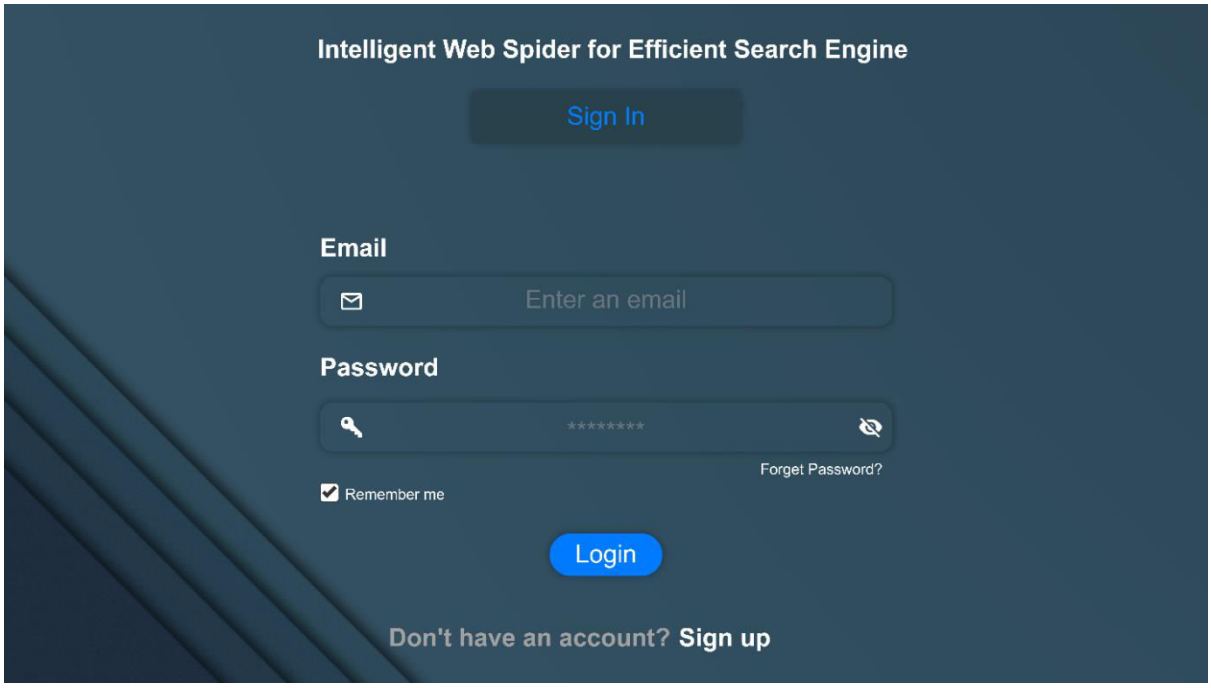
5.3 Prototype Design

5.3.1 Sign up Screen



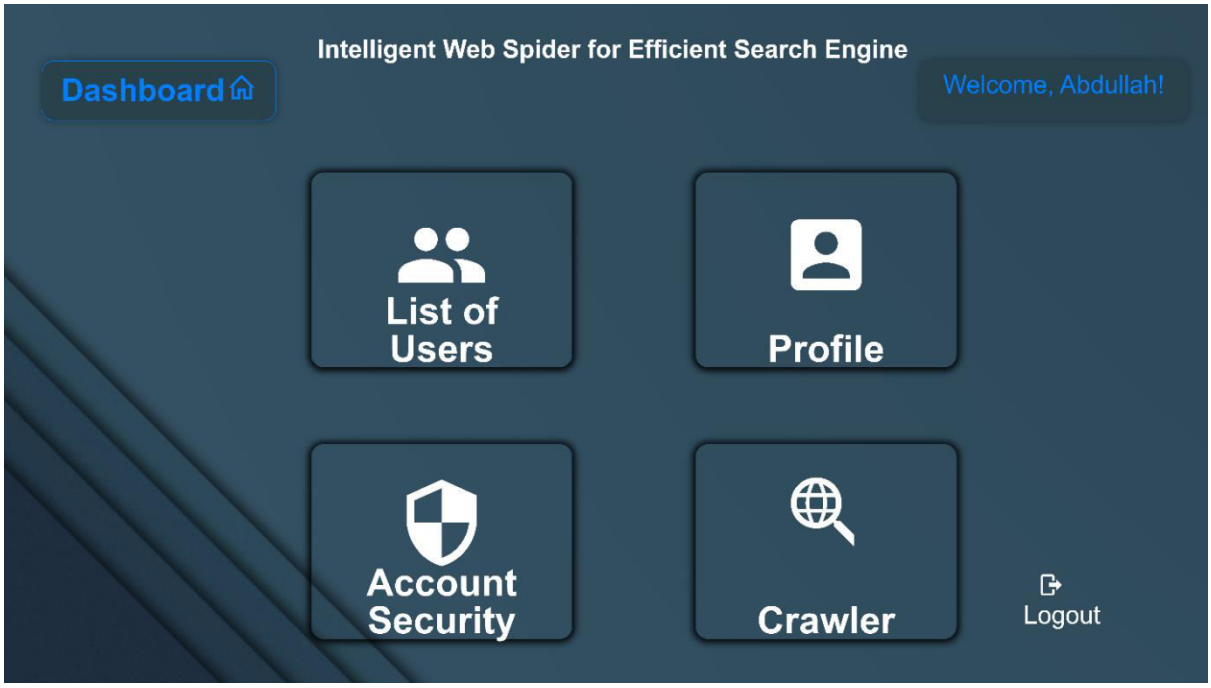
[Fig. 5.3] Prototype Sign up Screen

5.3.2 Sign in Screen



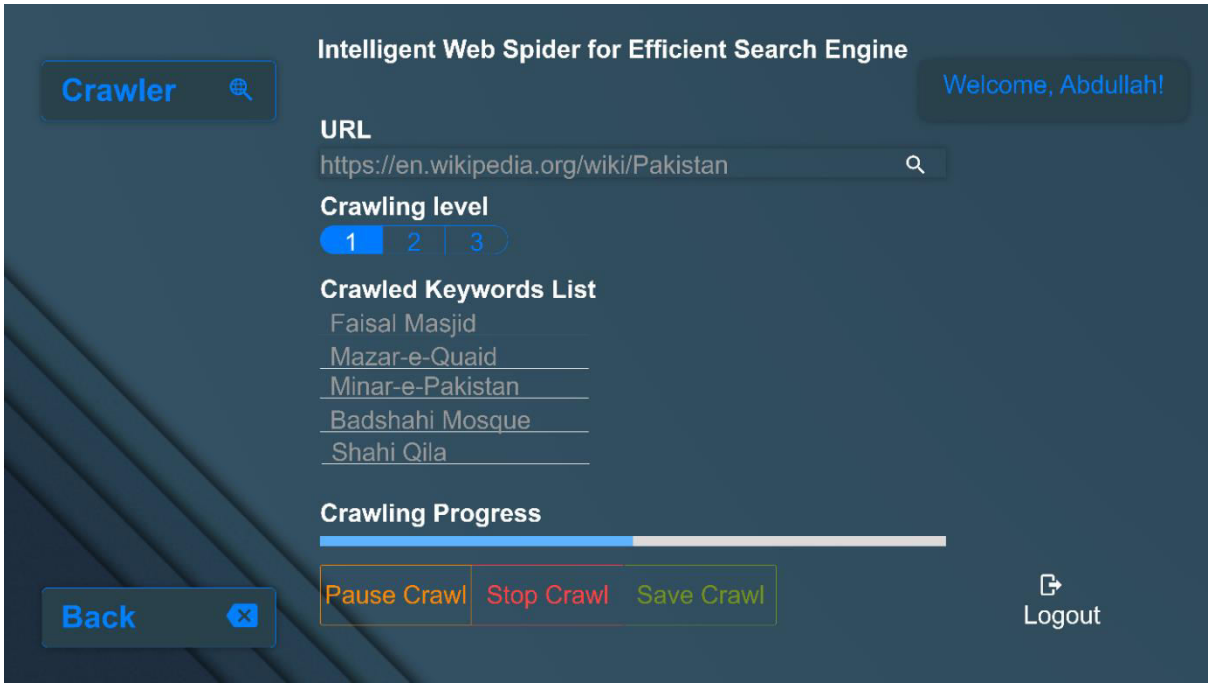
[Fig. 5.4] Prototype Sign in Screen

5.3.3 Dashboard Screen



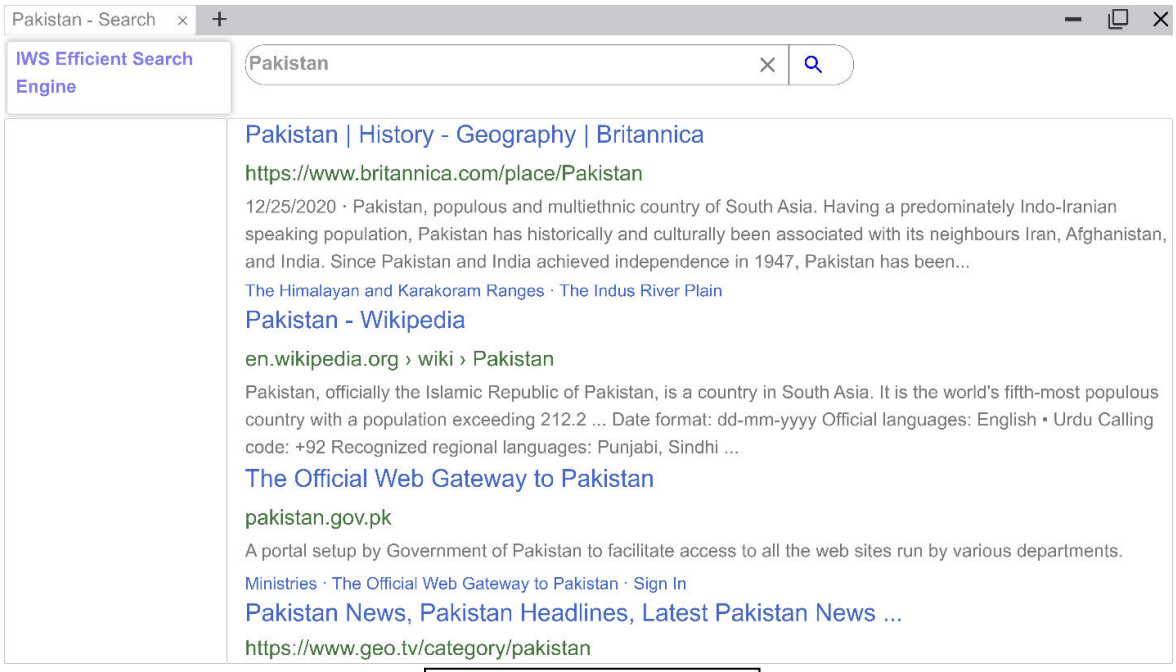
[Fig. 5.5] Prototype Dashboard Screen

5.3.4 Crawler Screen



[Fig. 5.6] Prototype Crawler Screen

5.3.5 Search Engine Screen

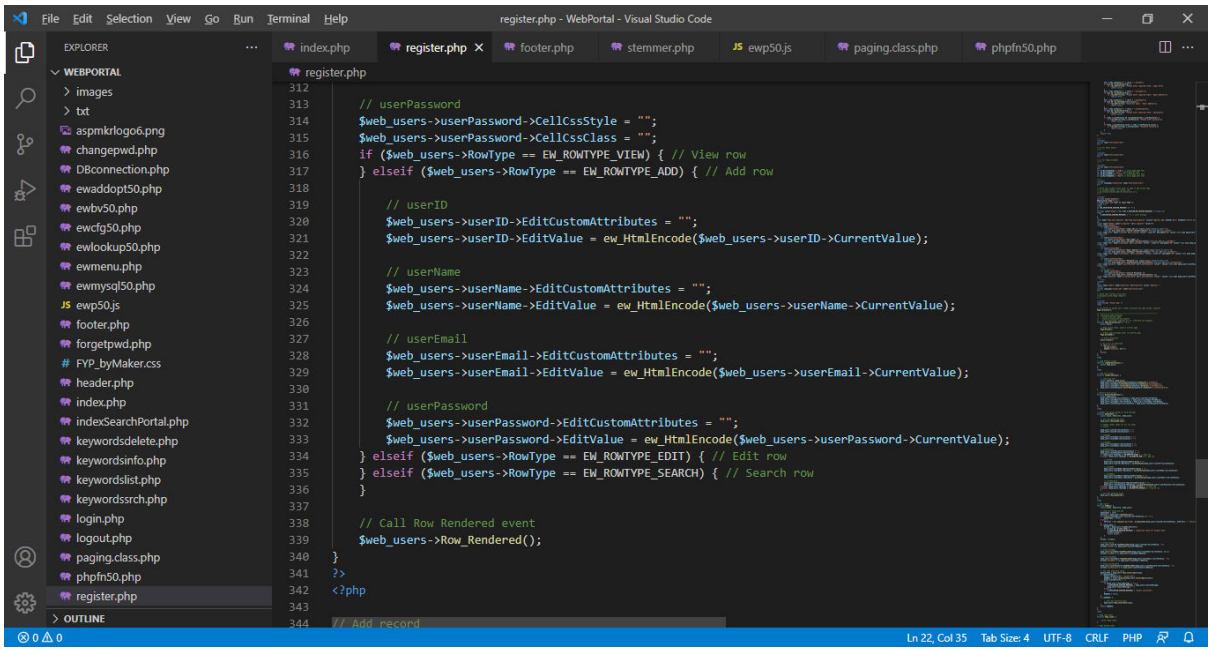


[Fig. 5.7] Prototype Search Engine Screen

5.4 Frontend Design

5.4.1 User Registration Screen

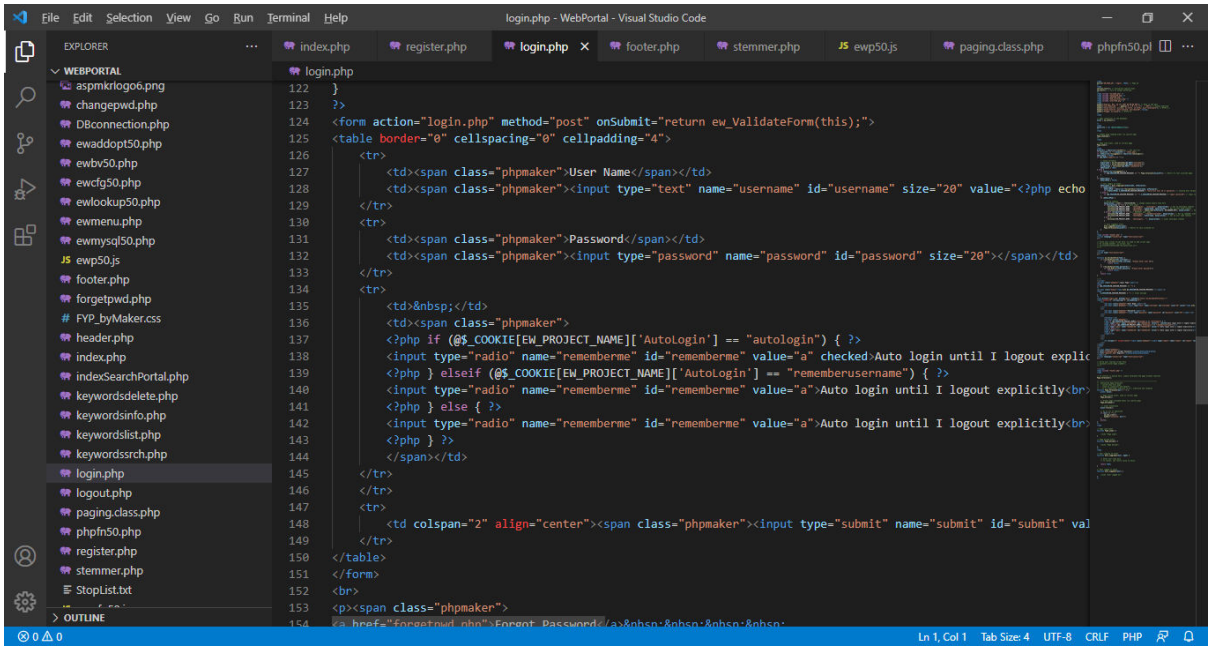
This screen allows a user to register himself to use our search engine. It asks for Login ID, User Name, Email, Password and Confirm Password and a Registration Button. User ID, Email, Password and Confirm Password are mandatory fields.



[Fig. 5.8] User Registration Screen

5.4.2 User Login Screen

This screen allows a user to login using Login Name and Password assigned. A radio button asking Do you Want to Save Password allows user to save password for next time.

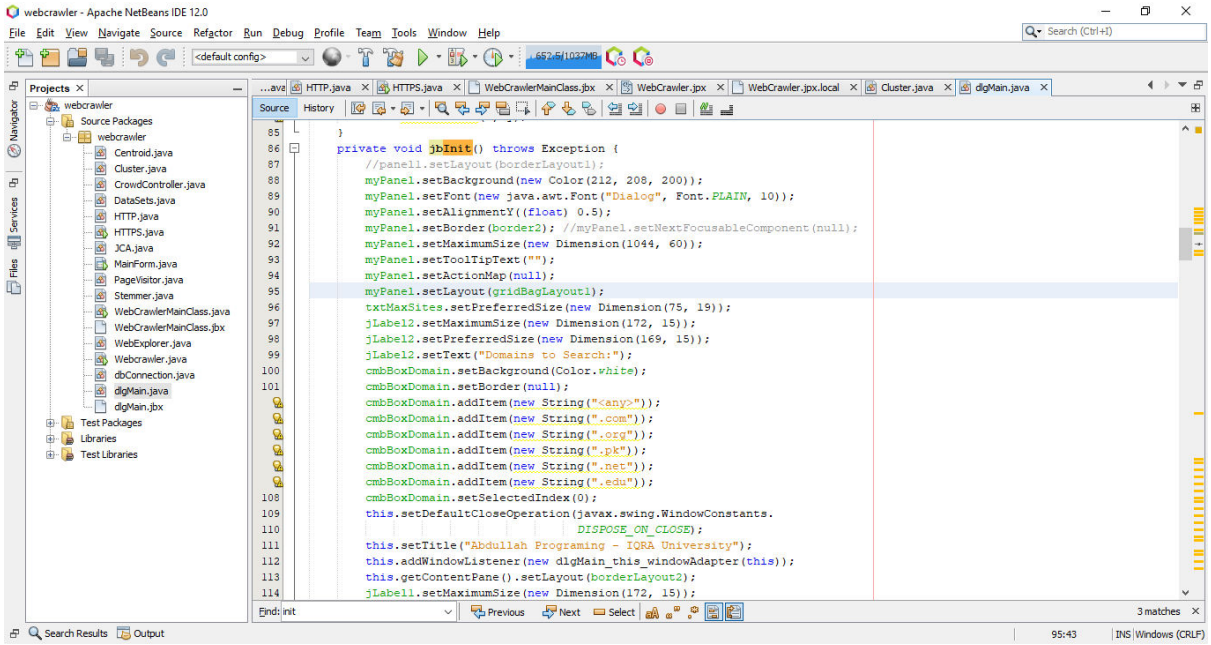


[Fig. 5.9] User Login Screen

5.4.3 Crawler Screen

This screen allows a user to crawl a website by providing following information:

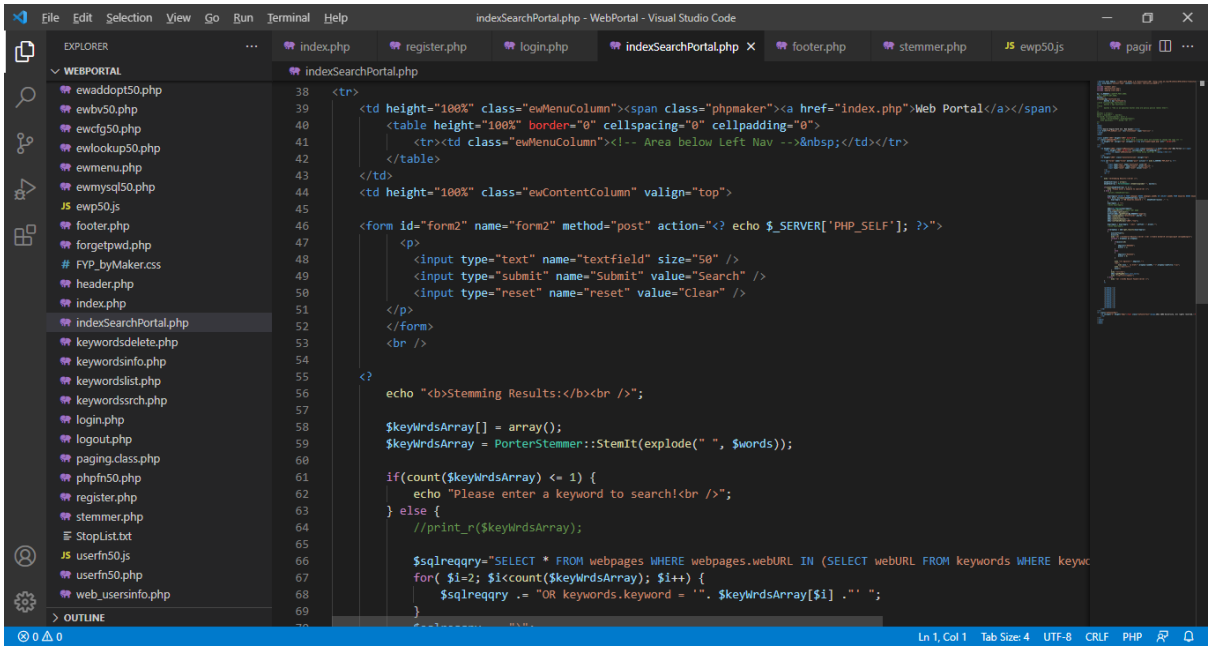
Site URL and maximum sites to visit. It has a start button to start crawling and Stop button to stop crawling. Save button to save keywords in the database. Clear button to clear all contents and Exit Button to close the application.



[Fig. 5.10] Crawler Screen

5.4.4 Search Engine Screen

This screen allows user to type a keyword and by pressing search button, system will search keywords from the database and display the results along with URLs where this keyword is found.

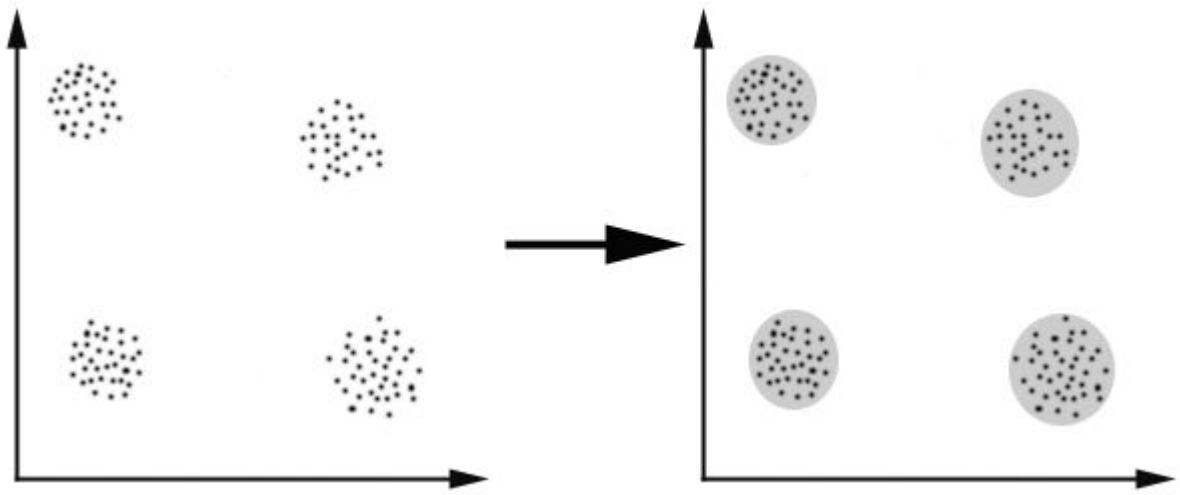


[Fig. 5.11] Search Engine Screen

5.5 Backend Design

5.5.1 Clustering

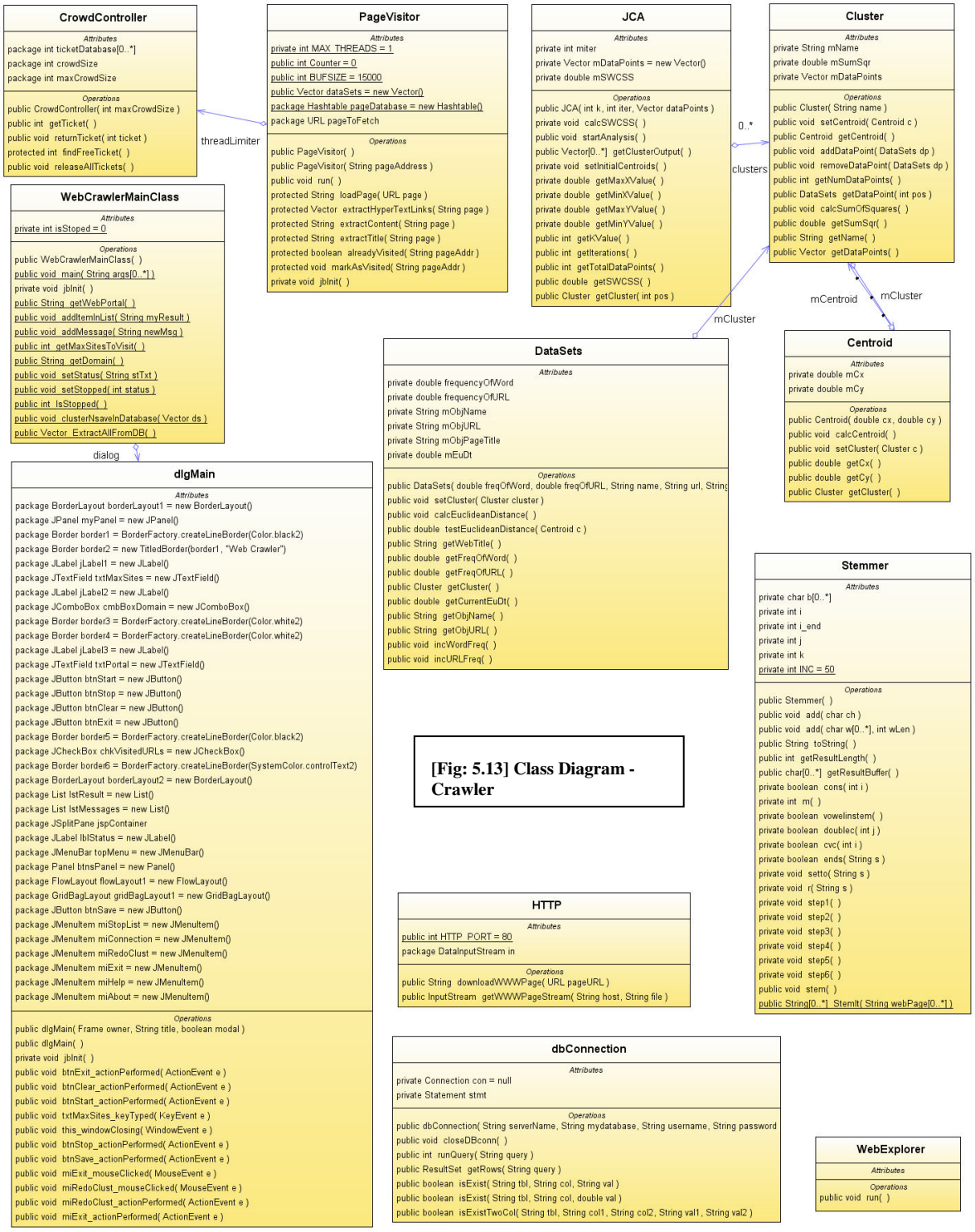
Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics



[Fig: 5.12] Grouping of different points into four clusters

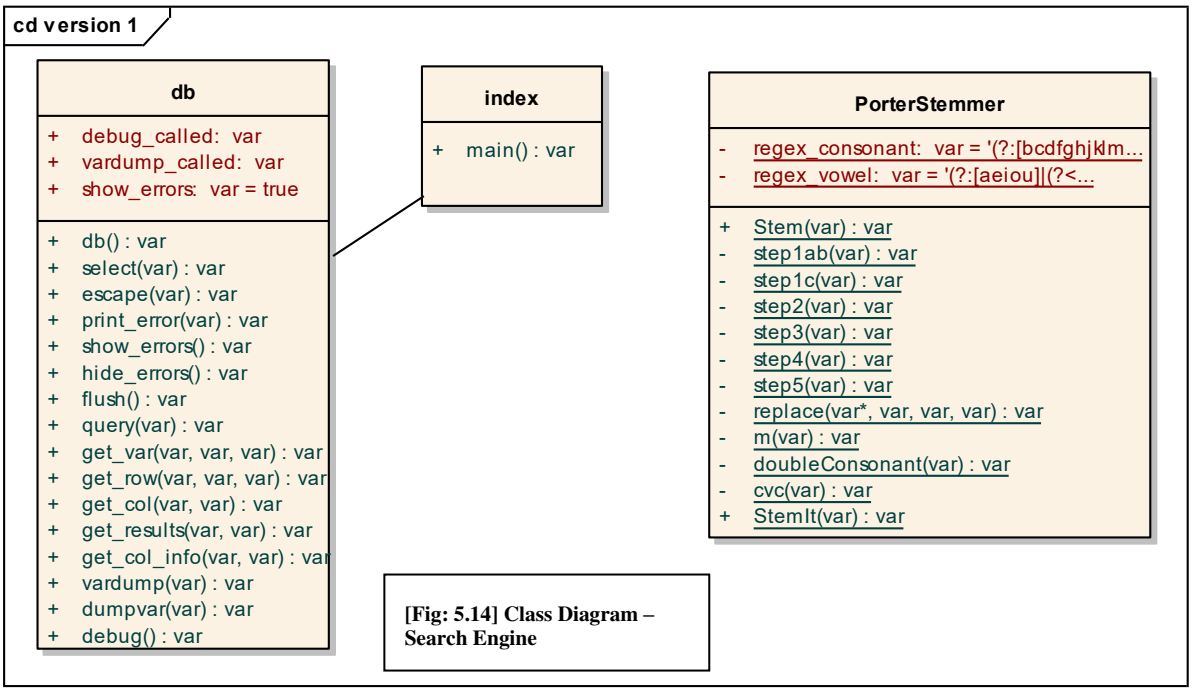
-level application flow chart depicting starting point, processing nodes and ending phase of the project.

5.5.2 Class Diagram – Crawler



[Fig: 5.13] Class Diagram - Crawler

5.5.3 Class Diagram – Search Engine



[Fig: 5.14] Class Diagram – Search Engine

5.6 Database Queries

Following are the main database queries used in the system.

Insert webpages in the webpages table of database:

INSERT INTO webpages(webURL, webTitle, webURLFreq, meta_content, Date_Of_Crawling) VALUES ('"+ dpTemp.getObjURL() + "','" + dpTemp.getWebTitle() + "', " + (int) dpTemp.getFreqOfURL() +", '"+ dpTemp.getMetaContent() + "', CURDATE());

Insert keywords in keywords table of database:

INSERT INTO keywords(keyword, webURL, freqOfWord, clusterNr) VALUES ('" + dpTemp.getObjName() + "', '" + dpTemp.getObjURL() + "'," + (int) dpTemp.getFreqOfWord() + "," + i + ");

Select all results from webpages table:

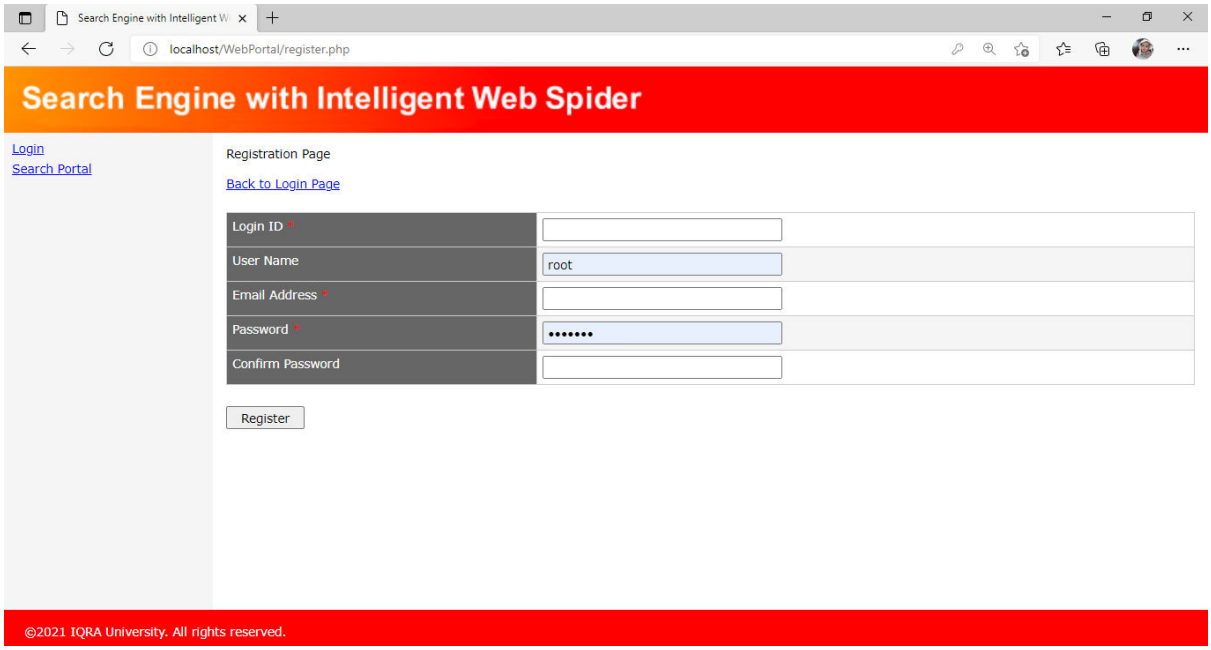
SELECT keyword, freqOfWord, webURLFreq, keywords.webURL, meta_content FROM webpages INNER JOIN keywords WHERE webpages.webURL = keywords.webURL")

5.7 External Libraries

No external libraries are used.

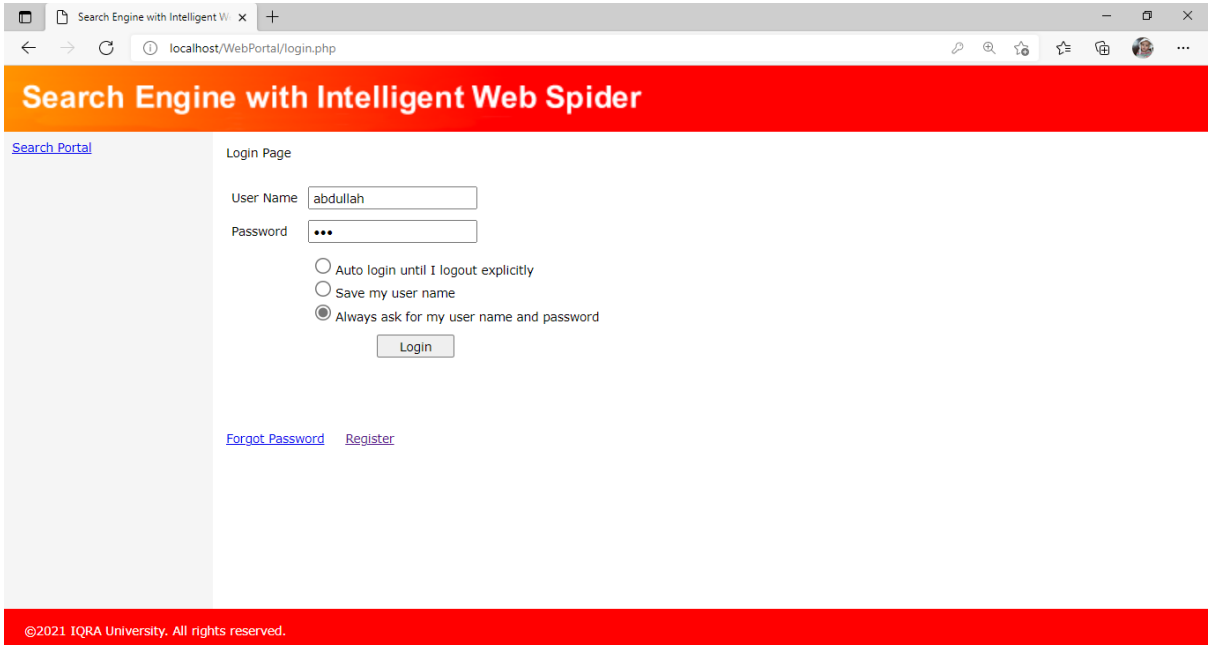
5.8 Screenshots

5.8.1 User Registration Screen



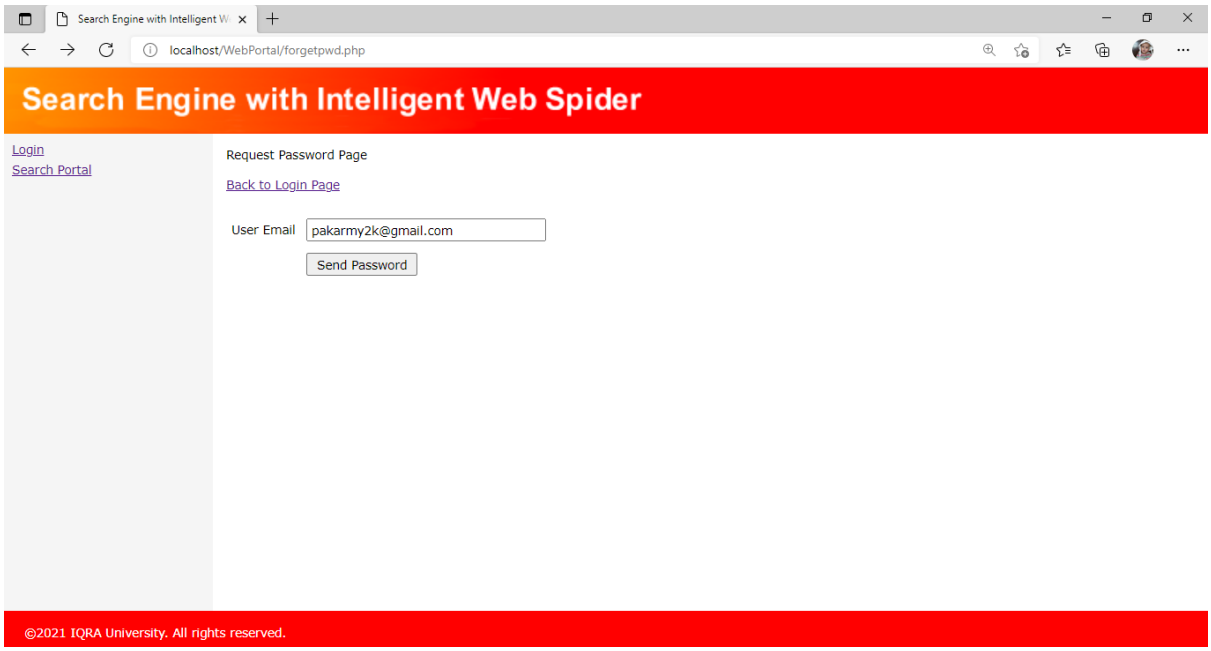
[Fig. 5.15] User Registration Screen Screenshot

5.8.2 Login Screen



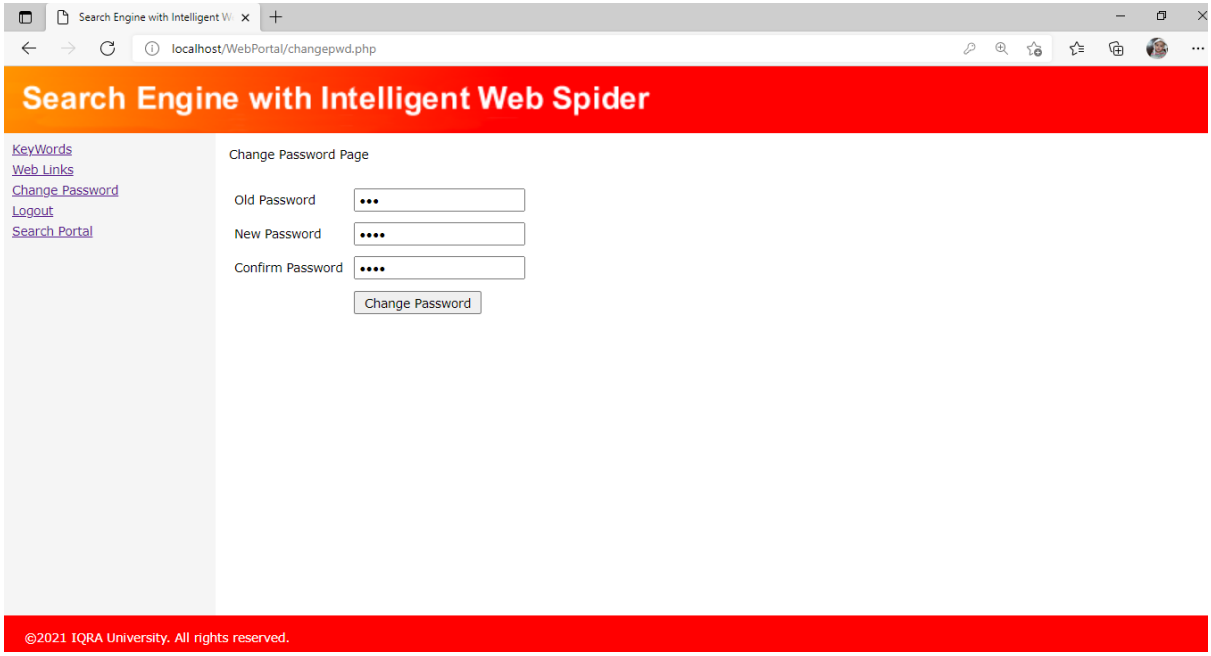
[Fig. 5.16] Login Screen Screenshot

5.8.3 Forgot Password Screen



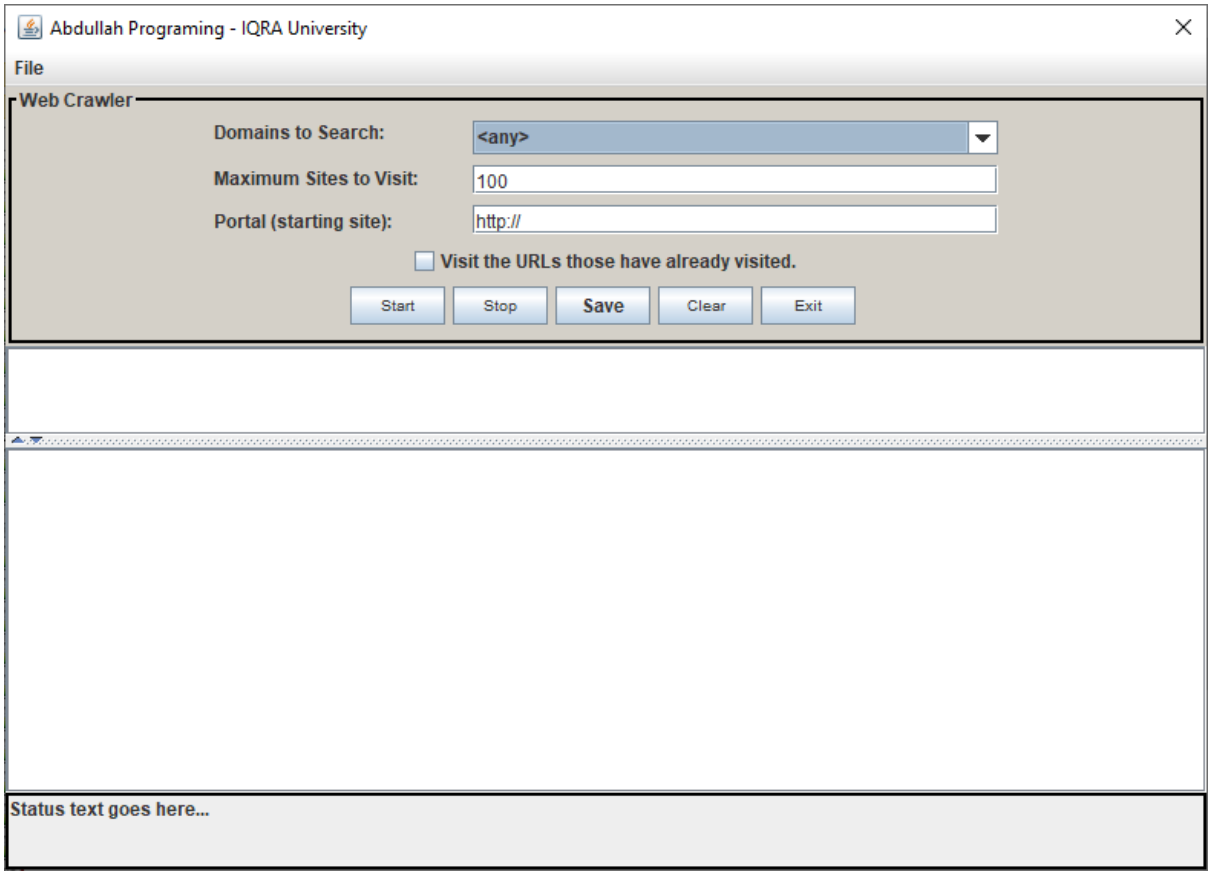
[Fig. 5.17] Forgot Password Screen Screenshot

5.8.4 Change Password Screen



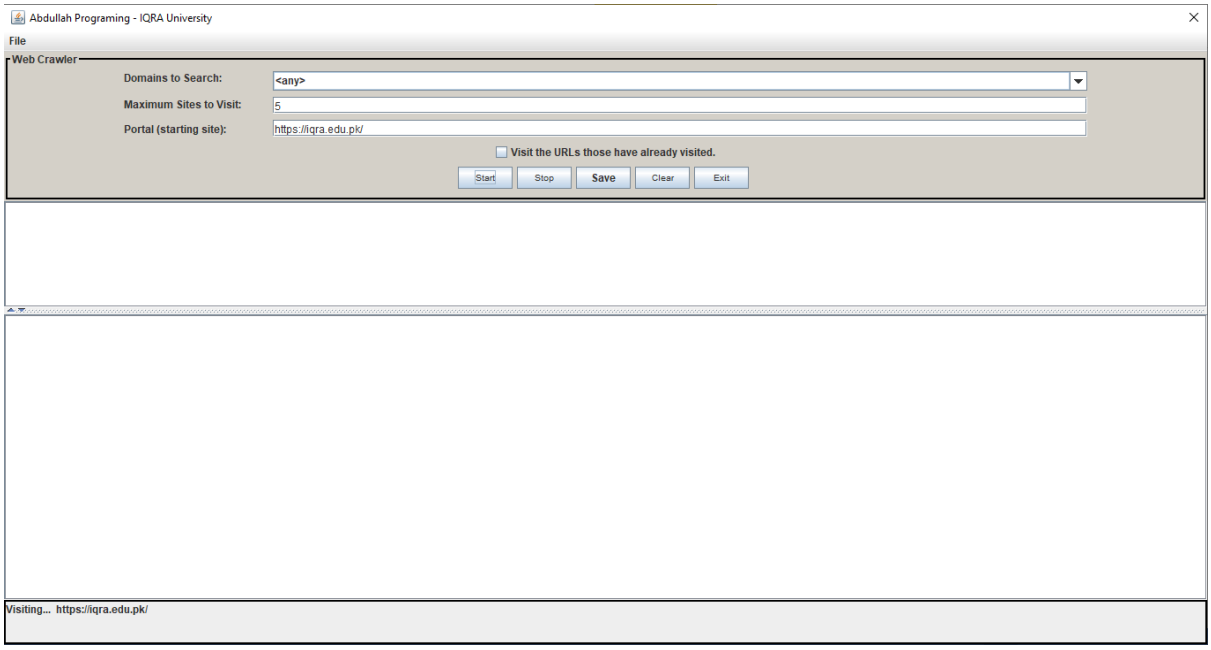
[Fig. 5.18] Change Password Screen Screenshot

5.8.5 Crawler Main Screen



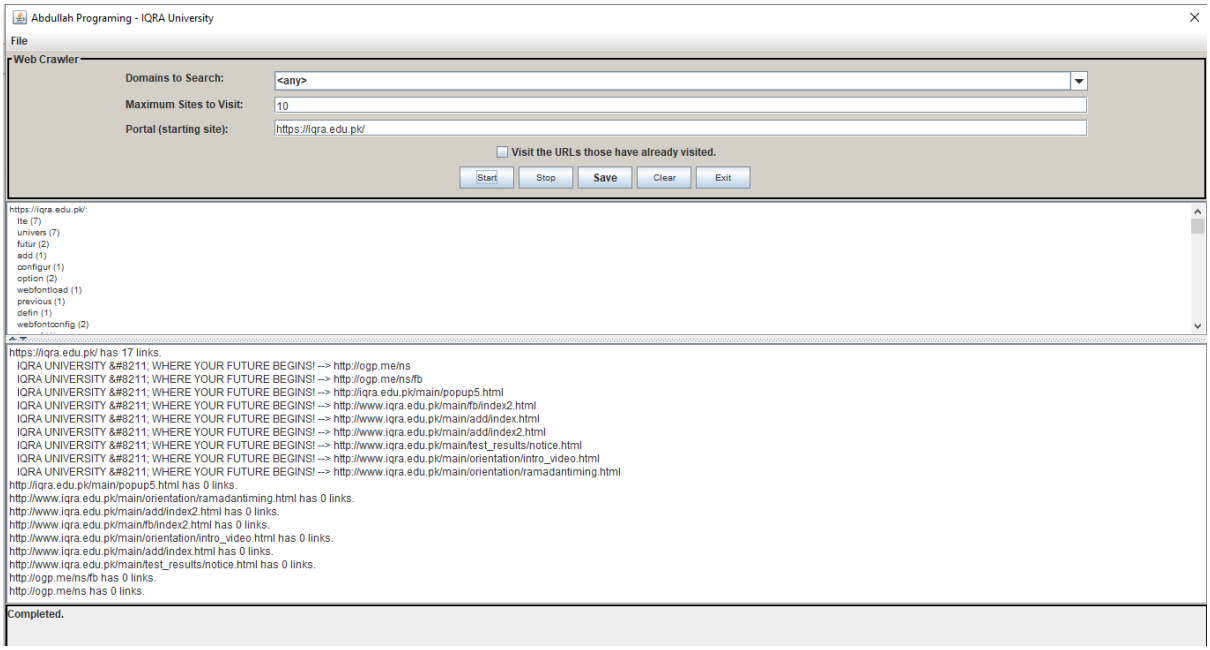
[Fig. 5.19] Crawler Main Screen Screenshot

5.8.6 Crawling Screen



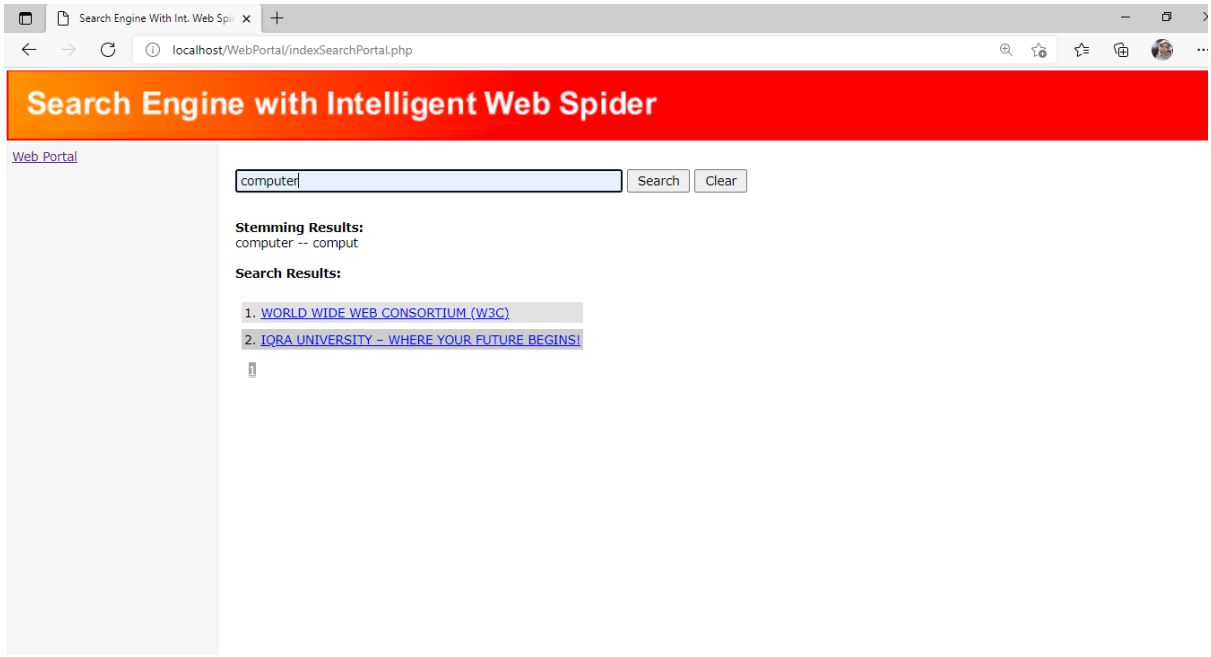
[Fig. 5.20] Crawling Screen Screenshot

5.8.7 Crawling Complete Displaying Keywords



[Fig. 5.21] Crawling Complete Displaying Keywords Screenshot

5.8.8 Search Engine



[Fig. 5.22] Search Engine Screenshot

5.9 Summary

Web Spider is the member of a software family that is responsible to crawling web pages and populates databases for search engines, using fast and intelligent techniques. Our project also includes a web-based module that will use the database maintained by spider.

CHAPTER 6: RESULT ANALYSIS AND TESTING:

6.1 Introduction:

This chapter explains the test cases based on functional requirements, Test Results including functional testing, frontend, backend testing and usability testing at the end summary of testing activity is presented.

6.2 Test Cases

Following are the Test Cases designed for our system:

Requirement Reference	**FR-1**	**Project Name**	**Intelligent Web Spider for Efficient Search Engine**
Test Case Id	TC-1	Test Type	Functionality
Test Case Description	System should have a user management mechanism that will allow access to the user to use the system		
Test Steps	1. Open the application 2. Click on User Registration 3. Fill the Registration Form by providing login id, name, email, password, confirm password 4. Click on Register Button		
Expected Result	1. System should check if mandatory fields are present 2. System should check if password and confirm password are the same 3. If the data is valid, system should display success message and create a new user		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 1 [Table 6.1]

Requirement Reference	**FR-2**	**Project Name**	**Intelligent Web Spider for**

			Efficient Search Engine
Test Case Id	TC-2	Test Type	Functionality
Test Case Description	The crawling depth should be a positive integer, and can't be infinity		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button		
Expected Result	1. System should check if mandatory fields are present 2. System should check if Crawler depth is a valid positive integer 3. If the data is valid, system should start crawling the URL		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 2 [Table 6.2]

Requirement Reference	**FR-3**	**Project Name**	**Intelligent Web Spider for Efficient Search Engine**
Test Case Id	TC-3	Test Type	Functionality
Test Case Description	System should fetch keywords from the contents of URLs provided by the user.		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button		
Expected Result	1. System should download contents of website and generate Keywords from the content downloaded		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 3 [Table 6.3]

Requirement Reference	FR-4	Project Name	Intelligent Web Spider for Efficient Search Engine
Test Case Id	TC-4	Test Type	Functionality
Test Case Description	System should transform keywords generated into simplified words		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button		
Expected Result	1. System should convert download keywords into simplified form by applying Stemming Algorithm		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 4 [Table 6.4]

Requirement Reference	**FR-5**	**Project Name**	**Intelligent Web Spider for Efficient Search Engine**
Test Case Id	TC-5	Test Type	Functionality
Test Case Description	System should keep user posted about the results		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button		
Expected Result	2. After completing the process of Crawling, system should display list of simplified keywords generated from the URL provided along with additional details about sub URLs etc on the specified areas of the screen		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 5 [Table 6.5]

Requirement Reference	FR-6	Project Name	Intelligent Web Spider for Efficient Search Engine
Test Case Id	TC-6	Test Type	Functionality
Test Case Description	System should generate Frequency of keywords		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button		
Expected Result	1. All crawled keywords should be assigned Frequencies based on their occurrences		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 6 [Table 6.6]

Requirement Reference	FR-7	Project Name	Intelligent Web Spider for Efficient Search Engine
Test Case Id	TC-7	Test Type	Functionality
Test Case Description	Store keywords in respective database clusters		
Test Steps	1. Open the Crawler Application 2. Specify Type of Domains to Crawl 3. Specify Crawling Depth as an integer value 4. Specify URL 5. Click on Start Button 6. After keywords are generated and system completes processing. A completion message along with keywords generated will be displayed 7. Click on Save Button		
Expected Result	1. All keywords generated should be saved in respective clusters as per their frequencies.		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 7 [Table 6.7]

Requirement Reference	FR-8	Project Name	Intelligent Web Spider for Efficient Search Engine
Test Case Id	TC-8	Test Type	Functionality
Test Case Description	Login to the system		
Test Steps	1. Open the Application 2. Specify Login Name and Password 3. Click on Login Button		
Expected Result	1. System should validate user name and password, if both are correct system should take the user to search engine's main page		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 8 [Table 6.8]

Requirement Reference	**FR-9**	**Project Name**	**Intelligent Web Spider for Efficient Search Engine**
Test Case Id	TC-9	Test Type	Functionality
Test Case Description	Perform Search		
Test Steps	1. Open the Application 2. On the Search field, type your search criteria 3. Click on Search Button		
Expected Result	2. System should display all keywords matching the search criteria. 3. Keywords should be clickable and by clicking the keyword, it should take the user to concerned website.		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 9 [Table 6.9]

Requirement Reference	FR-10	Project Name	Intelligent Web Spider for Efficient Search Engine
Test Case Id	TC-10	Test Type	Functionality
Test Case Description	Change Password		
Test Steps	1. Open the Application 2. Click on Change Password link 3. System should ask for old password, new password and confirm password 4. Click on Change Password Button		
Expected Result	1. System should check if the old password is correct 2. System should check if new password and confirm password are the same 3. In case both are correct, system should change the password and display a message to the user		
Actual Result	It is working according to the requirements		
Pass/Fail	Pass		
Date Prepared	May 2021		
Date Run	Jun 2021		
Prepared By	Abdullah Rather		
Tested By			

Test Case 10 [Table 6.10]

6.3 Summary

All the test cases were successfully run on the system has passed all test cases.

CHAPTER 7: CONCLUSION:

7.1 Introduction:

In this chapter we have explained the limitation of our system, challenges, and future work we think that will further enhance the features and scope of our project.

7.2 System Limitation and Challenges:

Following are the limitations and challenges of our system

- Messages generated by the system needs to be more informative
- User interface needs to be more attractive

7.3 Future Work:

Keywords generated by the crawler are based on the text it captures from the website. There are some useless words that should be skipped while adding the keywords. We applied this logic by adding skipping words in a text file with our project. This text file (we are calling it as “Stop List”) currently is based on the words we populated in the text file; however, we suggest that the Stop List should be auto generated based on some Artificial Intelligence (AI) Techniques so that more and more accurate results should be populated, and crawler should learn with the passage of time to filter useless words.

7.4 Summary:

By doing research as suggested under Future Work section, we can further improve the performance and efficiency of our crawler and search engine.

REFERENCES

1. http://en.wikipedia.org/wiki/Web_crawler
2. "Programming Spiders, Bots and Aggregators in Java", by Sybex.
3. IEEE Std 830-1993, *Recommended Practice for Software Requirements Specifications*, Software Engineering Standards Committee of the IEEE Computer Society, New York (1993)
4. Search Engines and Legal Issues, available at: http://searchenginewatch.com/searchday/article.php/2161041

2.1 "Introduction to Java" by Oracle Inc. https://www.oracle.com/java/technologies/java-ee-glance.html

2.2 https://www.w3schools.com/java/java_intro.asp

2.3 "Introduction to PHP by W3Schools", by W3Schools https://www.w3schools.in/php/intro/

2.4 "Dreamweaver Overview & Features" by Wikipedia https://en.wikipedia.org/wiki/Adobe_Dreamweaver

2.5 "Architecture of Crawler" by Shkapenyuk and Suel (Shkapenyuk and Suel, 2002)

2.6 "Introduction to Stemmers" by Wikipedia http://en.wikipedia.org/wiki/Stemming

2.7 "Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11:22–31" by Julie Beth Lovins (1968)

5.1 http://en.wikipedia.org/wiki/Stemming

5.2 Julie Beth Lovins (1968). Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11:22–31

5.3 http://tartarus.org/~martin/PorterStemmer/

5.4 http://snowball.tartarus.org/algorithms/porter/stemmer.html