***CHAPTER - 1***

***INTRODUCTION***

*The chapter gives brief introduction of the project.*

## 1. INTRODUCTION

### 1.1 Introduction

The deep (or hidden) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. Based on extrapolations from a study done at University of California, Berkeley, it is estimated that the deep web contains approximately 91,850 terabytes and the surface web is only about 167 terabytes in 2003. More recent studies estimated that 1.9 zettabytes were reached and 0.3 zettabytes were consumed worldwide in 2007. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 zettabytes in 2014. A significant portion of this huge amount of data is estimated to be stored as structured or relational data in web databases — deep web makes up about 96% of all the content on the Internet, which is 500-550 times larger than the surface web. These data contain a vast amount of valuable information and entities such as Infomine, Clusty, BooksInPrint may be interested in building an index of the deep web sources in a given domain (such as book). Because these entities cannot access the proprietary web indices of search engines (e.g., Google and Baidu), there is a need for anefficient crawler that is able to accurately and quickly explore the deep web databases.

It is challenging to locate the deep web databases because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, *genericcrawlers* and *focused crawlers*. Generic crawlers, fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) and Adaptive Crawler for Hidden-web Entries (ACHE) can automatically search online databases on a specific topic. FFC is designed with link, page, and form classifiers for focused crawling of web forms, andis extended by ACHE with additional components for form filtering and adaptive link learner. The link classifiers in these crawlers play a pivotal role in achieving higher crawling efficiency than the best-first crawler. However, these link classifiers are usedto predict the distance to the page containing searchable forms, which is difficult to estimate, especially for the delayed benefit links (links eventually lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms.

### 1.2 Literature Survey

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

* Survey on building applications on J2EE.
* Implementation of a framework scripted in JSP.
* Different online examples of data synchronization.
* Research on web server connectivity with Bing Search engine.
* Approached different Solutions for the requirement.

### 

### 1.2 Project Background

### Besides efficiency, quality and coverage on relevant deep web sources are also challenging. Crawler must produce a large quantity of high-quality results from the most relevant content sources. For assessing source quality, SourceRank ranks the results from the selected sources by computing the agreement between them. Furthermore, little work has been done on the source selection problem when crawling more content sources. Thus it is crucial to develop smart crawling strategies that are able to quickly discover relevant content sources from the deep web as much as possible.

### 1.3 Objective

### In this paper, we propose an effective deep web harvesting framework, namely *SmartCrawler*, for achieving both wide coverage and high efficiency for a focused crawler. Based on the observation that deep websites usually contain a few searchable forms and most of them are within a depth of three, our crawler is divided into two stages: *site locating* and *in-site exploring*. The site locating stage helps achieve wide coverage of sites for a focused crawler, and the in-site exploring stage can efficiently perform searches for web forms within a site. Our main contributions are:

* We propose a novel two-stage framework to address the problem of searching for hidden-web resources. Our site locating technique employs a *reverse searching* technique (e.g., using Google’s”link:” facility to get pages pointing to a given link) and incremental two-level site prioritizing technique for unearthing relevant sites, achieving more data sources. During the in-site exploring stage, we design a link tree for balanced link prioritizing, eliminating bias toward webpages in popular directories.

### We propose an adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on a topic using the contents of the root page of sites, achieving more accurate results. During the insite exploring stage, relevant links are prioritized for fast in-site searching.

### 

***CHAPTER - 2***

***SOFTWARE REQUIREMENT SPECIFICATION***

*Gives the details of platform specifications, Hardware and Software specifications.*

## 2. SOFTWARE REQUIREMENTS SPECIFICATION

### 2.1 Requirement Analysis

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports.

1. Problem statement
2. Data flow diagrams
3. Use case diagram
4. Other UML diagrams.

The above mentioned documents gives us diagrammatical view of the system what we are going to develop.

### 2.2 Problem Statement

The problem statement system only requires a semi-trusted third party, responsible for carrying out simple matching operations correctly.

### 2.3 Functional Requirements

* The application should results from a search engine (Bing).
* The application must have a user friendly interface that stores user details.
* The application must have a view of the listing of links present in Bing by using Jsoup API.
* The application will crawl all internal links particular url.
* The application ranks the top 10 links by using adaptive learning.
* Search results user can get in accurate.

### 2.4 Software Requirement Specification

The project is developed in Java Programming Language by using the Eclipse Juno Integrated Development Environment (IDE). We use the Java Development Kit (JDK) which includes a variety of custom tools that help us to develop web applications on the Apache Platform. At the Server side Apache Tomcat Server is used. For Data Storage Purpose we use MySql as a database server. And We can run our application in Windows and Linux any version.

**2.4.1 Purpose**

The purpose of this document is to present a detailed description of **“Smart Crawler Two Stage Crawler Efficiently Harvesting Deep Web Interfaces”** application. It will explain the purpose and features of the system that it will provide, constraints under which it must operate and how the system will react. The document also describes the non functional requirements of the system.

**2.4.2 Technologies Used**

**JAVA PLATFORM**

The programmer writes Java source code in a text editor which supports plain text. Normally the programmer uses an *Integrated Development Environment* (IDE) for programming. An IDE supports the programmer in the task of writing code, e.g. it provides auto-formatting of the source code, highlighting of the important keywords, etc.

At some point the programmer (or the IDE) calls the Java compiler (javac). The Java compiler creates the *byte code* instructions. . These instructions are stored in .class files and can be executed by the Java Virtual Machine.

**J2SE**

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming language known as "JAVA"

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming language known as "JAVA" and one of its three basic editions of Java known as Java standard edition being used for writing Applets and other web based applications.

J2SE platform has been developed under the Java umbrella and primarily used for writing applets and other Java-based applications. It is mostly used for individual computers. Applet is a type of fast-working subroutine of Java that is platform-independent but  work within other frameworks. It is a mini application that perform a variety of functions, large and small, ordinary and dynamic, within the framework of larger applications.

J2SE provide the facility to users to see Flash movies or hear audio files by clicking on a Web page link. As the user clicks, page goes into the browser environment and begin the process of launching application-within-an-application to play the requested video or sound application. So many online games are being developed on J2SE. JavaBeans can also be developed by using J2SE.

**J2EE**

Java ***2*** Platform ***E***nterprise ***E***dition. J2EE is a [platform](http://www.webopedia.com/TERM/P/platform.html)-independent, [Java](http://www.webopedia.com/TERM/J/Java.html)-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, [APIs](http://www.webopedia.com/TERM/A/API.html), and [protocols](http://www.webopedia.com/TERM/P/protocol.html) that provide the functionality for developing multitier, Web-based applications.

**Key features and services of J2EE:**

* At the [client](http://www.webopedia.com/TERM/C/client.html) tier, J2EE supports pure [HTML](http://www.webopedia.com/TERM/H/HTML.html), as well as Java [applets](http://www.webopedia.com/TERM/A/applet.html) or applications. It relies on [Java Server Pages](http://www.webopedia.com/TERM/J/JSP.html) and [servlet](http://www.webopedia.com/TERM/S/servlet.html) code to create HTML or other formatted data for the client.
* [Enterprise JavaBeans](http://www.webopedia.com/TERM/E/Enterprise_JavaBeans.html) (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
* [Java Database Connectivity](http://www.webopedia.com/TERM/J/JDBC.html) (JDBC), which is the Java equivalent to [ODBC](http://www.webopedia.com/TERM/O/ODBC.html), is the standard interface for Java databases.
* The Java servlet API enhances consistency for developers without requiring a [graphical user interface](http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html).
  1. **JAVA DATA BASE CONNECTION**

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

* Making a connection to a database
* Creating SQL or MySQL statements
* Executing that SQL or MySQL queries in the database
* Viewing & Modifying the resulting records

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as:

* Java Applications
* Java Applets
* Java Servlets
* Java ServerPages (JSPs)
* Enterprise JavaBeans (EJBs)

All of these different executables are able to use a JDBC driver to access a database and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

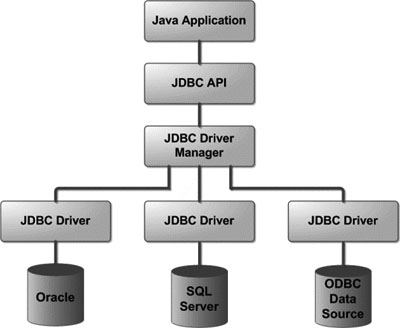
**JDBC Architecture:**

The JDBC API supports both two-tier and three-tier processing models for database access but in general JDBC Architecture consists of two layers:

* **JDBC API:** This provides the application-to-JDBC Manager connection.
* **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application.



### 2.5 Software Requirements

The software interface is the operating system, and application programming interface used for the development of the software.

Operating System: Windows XP or higher / Linux

Platform: JDK

Application Server: Apache Tomcat 7 or higher

Database: MySQL

Technologies used: Java, J2EE.

### 2.6 Hardware Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CLIENT** | | | | |
| OPERATING SYSTEM | SOFTWARE | PROCESSOR | RAM | Hard disk |
| Windows/Linux | Any Advanced Browser. Chrome/Opera | Intel/AMD processor | 256 Mb | 160 GB |

**Table 2:1 Client Requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SERVER** | | | | |
| OPERATING SYSTEM | SOFTWARE | PROCESSOR | RAM | HARD DISK |
| Windows/Linux | Any Browser.  JDK 1.7 or above  Apache 7 or above  MySQL 5.0 or above | Intel/AMD processor | 256Mb | 160 GB |

Table 2:2 Server Requirements

### 2.7 Functional Requirements (Modules)

The project having its own respective specific functionalites.

**2.7.1. Two-stage crawler:**

We propose a two-stage framework, namely Smart Crawler , for efficient harvesting deep web interfaces. In the first stage, Smart Crawler performs site-based searching for center pages with the help of search engines, avoiding visiting a large number of pages. To achieve more accurate results for a focused crawl, Smart Crawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking. To eliminate bias on visiting some highly relevant links in hidden web directories, we design a link tree data structure to achieve wider coverage for a website.

**2.7.2. Adaptive learning**

Adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on atopic using the contents of the root page of sites, achieving more accurate results. During insite exploring stage, relevant links are prioritized for fast in-site searching. We have performed an extensive performance evaluation of Smart Crawler over real web data in representative domains.

**2.7.3. Admin**

In our proposed architecture admin is a data owner, and perform site ranker and adaptive link ranker. He search site links from the google search engine according to some topics, and choose links for smart crawling. He maintains the site database.

**2.7.4. Service Provider**

### In our proposed architecture user is end user of our application, and data user. Whenever he wants the data he can search from our application, data retrieve from the Bing search engine, but when sites are matched with our seed sites then smart crawl results he can get according to ranking.

### 2.8 Non-Functional Requirements

**2.8.1 Flexibility & Scalability**

Oracle itself has given a set of applications with JDK but the whole developer community can develop their own applications and they have access to same resources and public API which are accessible to core applications.

**2.8.2 Robust**

The application is fault tolerant with respect to illegal user/receiver inputs. Error checking has been built in the system to prevent system failure.

**2.8.3 Fragmentation**

Java gave the same environment which is open; the entire API’s which is open to all the devices which reduces fragmentation. If you develop an java application, it will run on all the devices.

**2.8.4 Open Source:**

Java open source is free and easy to download. Java is a platform in depended based programming language and The Java virtual machine (JVM) is a software implementation of a computer that executes programs like a real machine.

**2.8.5 Scalability:**

The system can be extended to integrate the modifications done in the present application to improve the quality of the product. This is meant for the future works that is to be done on the application.

**2.8.6 Reliability:**

Since the application is being developed through java, the most famous, efficient and reliable language, so it is reliable in every aspect until and unless there is an error in the programming side. Thus the application can be a compatible and reliable one.

**2.8.7 Portability:**

This System must be intuitive enough such that user with average background in using mobile phones can quickly experiment with the system and learn how to use the project. The system has user friendly interface.

### 2.9 Feasibility study

A key part of the preliminary investigation that reviews anticipated costs and benefits and recommends a course of action based on operational, technical, economic, and time factors. The purpose of the study is to determine if the systems request should proceed further.

**2.9.1 Organisational Feasibility**

The application would contribute to the overall objectives of the organization. It would provide a quick, error free and cost effective solution to the current process CRM marketing. It would provide a solution to many issues in the current system. As the new system is flexible and scalable it can also be upgraded and extended to meet other complex requirements which may be raised in the future. However it is up to the organization to upgrade or extend it.

**2.9.2 Economic Feasibility**

The project is economically feasible as it only requires a system with window or linux operating system. The application is free to get once released into realtime host server. The users should be able to connect to internet through mechine and this would be the only cost incurred on the project.

**2.9.3 Technical Feasibility**

To develop this application, a high speed internet connection, a database server, a web server and software are required. The current project is technically feasible as the application was successfully deployed on Lenovo Thinkpad L60 having Windows 7 operating system and also Intel I5 Processor.

**2.9.4 Behavioural Feasibility**

The application is behaviourally feasible since it requires no technical guidance, all the modules are user friendly and execute in a manner they were designed to.

***CHAPTER - 3***

***ANALYSIS & DESIGN***

*This chapter gives the details of the system and data design.*

## 3. ANALYSIS & DESIGN

### 3.1 Introduction

**3.1.1 Purpose**

In this section the purpose of the document and the project is described.

**3.1.1.1 Document Purpose**

An SDD (Software design description) is a representation of a software system that is used as a medium for communicating software design information.

**3.1. Project Purpose**

The prime purpose of this “Smart Crawler Two Stage Crawler  
 Efficiently Harvesting Deep Web Interfaces” is to create a fully fledged web application which would communicate with the remote server to send and retrieve data as per requirement. This application works when there is internet connectivity

The application retrieves user details from the server and syncs it to the user machines. These details can be stored in the Mysql database. The retrieved information can be updated in the User Machine and synched to the server. We propose a two-stage framework, namely Smart Crawler, for efficient harvesting deep web interfaces. In the first stage, Smart Crawler performs site-based searching for center pages with the help of search engines, avoiding visiting a large number of pages. To achieve more accurate results for a focused crawl, SmartCrawlerranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking..

**3.1.2 Scope**

In this section the scope of the document and the project is explained in brief.

**3.1.2.1 Document Scope**

This document contains a thorough description of the high level architecture that will be used in developing the system. Communicating at a purposefully high level, it will only form the basis for the Software Detailed Design and implementation. However, the SDD itself will not be in sufficient detail to implement the code. It will convey the overall system design of the system, the user interface design and higher level module design and the architecture working of the Java Virtual Machine. Design details that will not be included in the SDD are:

* Low level classes that will be used in the implementation. The full description of the implementation of each module is not needed, but the public modules that will be interfaced will be described.
* Exact detailed description of interactions within each module

### 3.2 System Overview

**3.2.1 Development Tools**

Java framework uses certain development tools which are as follows:

**3.2.1.1 JDK**

The Java Development Kit (JDK) is provided by Sun Microsystems as a basic development environment for Java. The JDK provides similar facilities to the cc compiler for C programs, plus a JVM simulator and some additional facilities such as a debugger. To use the JDK, programs are constructed as ascii text files (by using an editor, for example). The program files are compiled, which translates the Java code to JVM bytecode in .class files.

Each public class must be in a file having the class name (case sensitive on Unix) followed by a .java suffix. There may be any number of classes defined in a .java file, but the compiler produces a separate .class file for each class. A file is compiled with the **javac** command, which is similar to the cc (or gcc) command. A class is executed (or more precisely, the method **main** in a class is executed) by the command **java** with the class name (not the .class file) as the parameter. Thus, for example, to compile the program in file Hi.java, we would use the command

***javac Hi.java***

and then to execute the program we would use the command

***java Hi***

Both compile-time and execution-time (exceptions) error messages include the file name and line where the error occurred. No .class file is produced if there is a compile-time error.

**3.2.1.2 TOMCAT 7.0 WEB SERVERS**

Apache Tomcat is a web container developed at the Apache Software Foundation (ASF). Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Tomcat includes its own HTTP server internally.

**Environment:**

Tomcat is a web server that supports servlets and JSPs. The accompanying Tomcat Jasper compiler compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache HTTP Server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Tomcat is cross-platform, running on any operating system that has a Java Runtime Environment. The following properties can be specified, either as system properties, or by using a deployer. Properties file located in the root folder of the deployer package:

* Build: The build folder used will be, by default, ${build}/webapp${path}. After the end of the execution of the compile target, the web application WAR will be located at ${build}/webapp${path}.war.
* Webapp: Folder containing the expanded web application which will be compiled and validated. By default, the folder is myapp.
* Path: Deployed context path of the web application, by default /myapp.
* URL: Absolute URL to the manager web application of a running Tomcat server, which will be used to deploy and undeploy the web application. By default, the deployer will attempt to access a Tomcat instance running on local host, at http://localhost:8080/manager.
* Username: Username to be used to connect to the Tomcat manager.
* Password: Password to be used to connect to the Tomcat manager

A web application which is programmatically developed by the developer is stored in the webapps folder and WEB-INF folder also saved in that location which consists of folder named classes which supports to run application automatically.

* The deployer package includes a ready to use Ant script, with the following targets:
* compile (default): Compile and validate the web application. This can be used standalone, and does not need a running Tomcat server. The compiled application will only run on the associated Tomcat 5.0.x server release, and is not guaranteed to work on another Tomcat release, as the code generated by Jasper depends on its runtime component. It should also be noted that this target will also compile automatically any Java source file located in the /WEB-INF/classes folder of the web application.
* deploy: Deploy a web application (compiled or not) to a Tomcat server
* undeploy: Undeploy a web application
* Start: Start web application
* reload: Reload web application
* Stop: Stop web application

**3.2.1.3 MySQL**

The MySQL (TM) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for missioncritical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a trademark of MySQL AB. The MySQL software has Dual Licensing, which means you can use the

MySQL software free of charge under the GNU General Public License (http://www.gnu.org/licenses/). You can also purchase commercial MySQL licenses from MySQL AB if you do not wish to be bound by the terms of the GPL. The MySQL web site (http://www.mysql.com/) provides the latest information about the MySQL software.

**What Is MySQL?**

MySQL, the most popular Open Source SQL database, is developed and provided by MySQL AB. MySQL AB is a commercial company that builds its business providing services around the MySQL database. The MySQL web site (http://www.mysql.com/) provides the latest information about MySQL software and MySQL AB.

**MySQL is a relational database management system.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. The tables are linked by defined relations making it possible to combine data from several tables on request. The SQL part of “MySQL” stands for “Structured Query Language”the most common standardised language used to access databases.

**Why use the MySQL Database Server?**

The MySQL Database Server is very fast, reliable, and easy to use. If that is what you are looking for, you should give it a try. MySQL Server also has a practical set of features developed in close cooperation with our users. You can find a performance comparison of MySQL Server to some other database managers on our benchmark page.

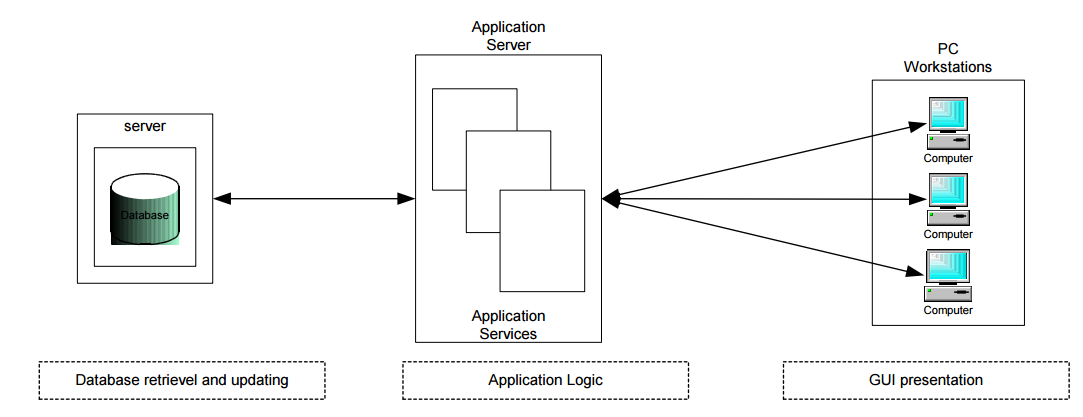
**The technical features of MySQL Server**

For advanced technical information, The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of programming interfaces (APIs).

### 3.3 System Architecture

**3.3.1 Architectural Design**

### The J2EE platform uses a multitier distributed application model. This means application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on which tier in the multitier J2EE environment the application component belongs. Figure shows multitier J2EE applications divided into the tiers described in the bullet list below. The J2EE application parts shown in Figure 1 are presented in J2EE Application Components



Application Processing provided by multiple tiers

1. Database Server

2. Application Server

3. PC Workstation

### J2EE applications use a thin client. A thin client is a lightweight interface to the application that does not do things like query databases, execute complex business rules, or connect to legacy applications. Heavyweight operations like these are off-loaded to web or enterprise beans executing on the J2EE server where they can leverage the security, speed, services, and reliability of J2EE server-side technologies.

### Web Components

### J2EE web components can be either JSP pages or servlets. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that contain static content and snippets of Java programming language code to generate dynamic content. When a JSP page loads, a background servlet executes the code snippets and returns a response.

### Static HTML pages and applets are bundled with web components during application assembly, but are not considered web components by the J2EE specification. Server-side utility classes can also be bundled with web components, and like HTML pages, are not considered web components.

**Benefits**

* Divides Application Processing across multiple machines:
* Non-critical data and functions are processed on the client
* Critical functions are processed on the server.
* Optimizes Client Workstations for data input and presentation (e.g., graphics and mouse support)
* Optimizes the Server for data processing and storage (e.g., large amount of memory and disk space)
* Scales Horizontally – Multiple servers, each server having capabilities and processing power, can be added to distribute processing load.
* Scales Vertically - Can be moved to more powerful machines, such as minicomputer or a mainframe to take advantage of the larger system’s performance
* Reduces Data Replication - Data stored on the servers instead of each client, reducing the amount of data replication for the application.

### 3.4 Data Design

**3.4.1 Databases**

**MySQL RDBMS**

|  |
| --- |
| Name |
| **Crawler** |

Table 3:1 MySQL Database

**3.4.2.1** Table**:** FLOGIN

**Columns**

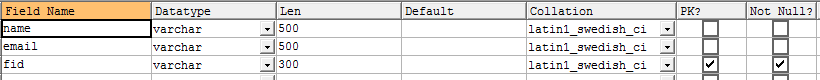
****

Table 3:2 Structure of FLOGIN table of MySQL Database

**Definition**

CREATE TABLE `flogin` (

`name` varchar(500) DEFAULT NULL,

`email` varchar(500) DEFAULT NULL,

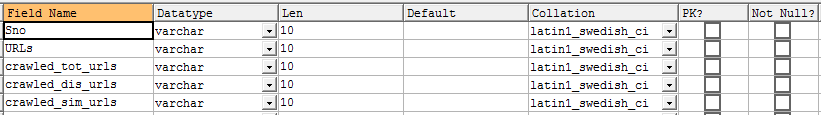
`fid` varchar(300) NOT NULL,

PRIMARY KEY (`fid`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

**3.4.2.2 Table:** Graph

**Columns:**

****

**Table 3:3 Structure of Graph table MySql Database**

Definition:

CREATE TABLE `graph` (

`Sno` varchar(10) DEFAULT NULL,

`URLs` varchar(10) DEFAULT NULL,

`crawled\_tot\_urls` varchar(10) DEFAULT NULL,

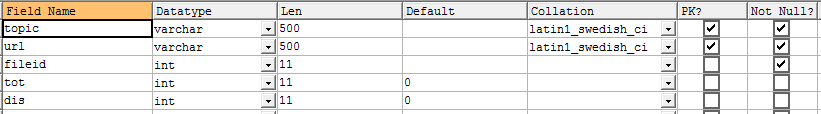
`crawled\_dis\_urls` varchar(10) DEFAULT NULL,

`crawled\_sim\_urls` varchar(10) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1

**3.4.2.3** **Table:** METADATA

**Columns:**

****

**Table 3:4 Structure of METADATA table of MySQL Database**

**Definition:**

CREATE TABLE `metadata` (

`topic` varchar(500) NOT NULL,

`url` varchar(500) NOT NULL,

`fileid` int(11) NOT NULL,

`tot` int(11) DEFAULT '0',

`dis` int(11) DEFAULT '0',

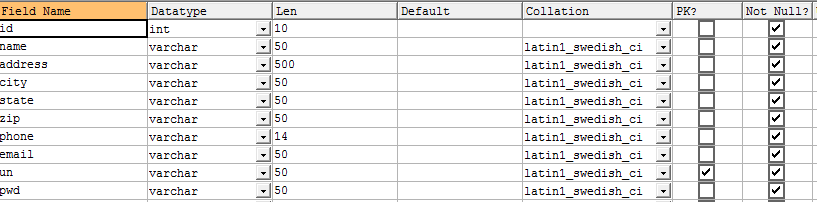
PRIMARY KEY (`topic`,`url`),

UNIQUE KEY `fileid` (`fileid`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

**3.4.2.4** **Table:** REGISTER

**Columns:**

****

**Table 3:5 Structure of REGISER table of MySQL Database**

**Definition:**

CREATE TABLE `register` (

`id` int(10) NOT NULL,

`name` varchar(50) NOT NULL,

`address` varchar(500) NOT NULL,

`city` varchar(50) NOT NULL,

`state` varchar(50) NOT NULL,

`zip` varchar(50) NOT NULL,

`phone` varchar(14) NOT NULL,

`email` varchar(50) NOT NULL,

`un` varchar(50) NOT NULL,

`pwd` varchar(50) NOT NULL,

PRIMARY KEY (`un`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

***CHAPTER - 4***

***MODELING***

*This chapter gives the unified modeling language diagrams.*

## 4. MODELING

### 4.1 Design

Requirements gathering followed by careful analysis leads to a systematic Object Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams. UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development.

**4.1.1. Use Case Diagram**

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical over view of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application there is only actor – soldier and below is the use case diagram of this application.



**Figure 4-1 Use Case Diagram for System**

**4.1.2 Sequence Diagram**

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals later).

A popular use for them is to document the dynamics in an object-oriented system. For each key, collaboration diagrams are created that show how objects interact in various representative scenarios for that collaboration.

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a numbers of lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, combined fragment, interaction use, state invariant, continuation, destruction occurrence.

****

**Figure 4-4 Sequence Diagram for User**

****

**Figure 4-5 Sequence Diagram for Admin**

**4.1.3 Activity Diagram**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.Activity is a particular operation of the system.



**Fig 4-6 Activity Diagram for user**



**4.1.4 Class Diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object oriented odeling. It is used both for general conceptual odeling of the odelingc of the application, and for detailed odeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

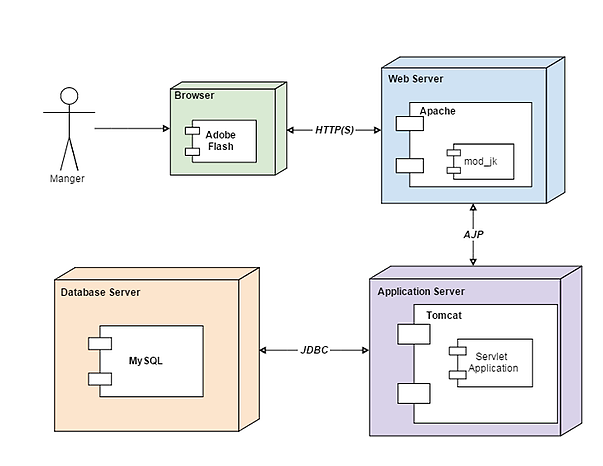
****

**Figure 4-8 Class Diagram**

**4.1.5 Deployment Diagram**

Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes).

Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.

****

**Figure 4-9 Deployment Diagram of the system**

***CHAPTER - 5***

***IMPLEMENTATION***

*The chapter gives the details of the implementation.*

## 5. IMPLEMENTATION

### 5.1 Sample Code

**5.1.1 Code for databasecon.java**

package databaseconnection;

import java.sql.\*;

public class databasecon

{

static Connection co;

public static Connection getconnection()

{

try

{

Class.forName("com.mysql.jdbc.Driver");

co = DriverManager.getConnection("jdbc:mysql://localhost:3306/crawler","root","root");

}

catch(Exception e)

{

System.out.println("Database Error"+e);

}

return co;

}

}

**5.1.2 Code for CountWords.java**

package Crawl;

import org.apache.commons.lang.StringUtils;

public class CountWords

{

public static int main(String input, String compare)

{

input=input.toLowerCase();

compare=compare.toLowerCase();

int number = StringUtils.countMatches(input, compare);

return number;

}

}

**5.1.3 Code for CrawlText.java**

package Crawl;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

public class CrawlText

{

public static String main(String urlString) throws Exception

{

Document doc;

//urlString = "http://db.apache.org/derby/releases/release-10.11.1.1.cgi";

doc = Jsoup.connect(urlString).get();

String text = doc.body().text();

return text;

}

public static void main(String[] args) throws Exception

{

CrawlText.main("http://db.apache.org/derby/releases/release- 10.11.1.1.cgi");

}

}

**5.1.4 Code for googleSearch.java**

package google;

import java.util.Vector;

import java.io.File;

import java.io.PrintWriter;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

public class googleSearch {

public static Vector ae=new Vector();

Vector result=new Vector();

public static Vector main(String args) {

Vector result=new Vector();

int count=0;

String key="";

Document doc;

try {

// need http protocol

boolean term=false;

//System.out.println("BING SEARCH"+args);

String[] words=null;

words=args.split(" ");

//System.out.println("words l = "+words.length);

for(String s:words)

{

if(term==true)

{

key=key+"+";

key=key+s;

}else{

key=key+s;

term=true;

}

}

//System.out.println(key);

File f=new File("crawlerdata/googleResults.txt");

System.out.println(f);

if(f.exists())

{

f.delete();

}

f.createNewFile();

PrintWriter pw=new PrintWriter(f);

String first="11";

String urlString = "http://www.bing.com/search?q="

+ key;

doc = Jsoup.connect(urlString).get();

//System.out.println(urlString);

// get page title

String title = doc.title();

//System.out.println("title : " + title);

// get all links

Elements links = doc.select("a[href]");

for (Element link : links) {

// get the value from href attribute

String lk=link.attr("href");

if((lk.contains("http://")||lk.contains("https://"))&&!lk.contains("microsoft")&&!lk.contains("msn"))

{

lk=lk.replaceAll("https://","");

lk=lk.replaceAll("http://","");

//System.out.println(link.attr("href")+"------"+count++); ;

pw.println("<a href=search3.jsp?link="+lk+" target=\_blank>http://"+lk+"</a>");

pw.println(title);

pw.println("");

result.add(lk);

result.add(link.text());

//result.add("");

}

}

//}

pw.flush();

pw.close();

} catch (Exception e) {

e.printStackTrace();

}

return result;

}

public static void main(String[] args)

{

}

}

**5.1.4 Code for ana2.js**

### <%@ page import="databaseconnection.\*" %>

### <%@ page import="java.sql.\*" %>

### <%@ page import="java.io.\*" %>

### <%@ page import="java.awt.\*" %>

### <%@ page import="org.jfree.chart.ChartFactory" %>

### <%@ page import="org.jfree.chart.ChartUtilities" %>

### <%@ page import="org.jfree.chart.JFreeChart" %>

### <%@ page import="org.jfree.chart.plot.PlotOrientation"%>

### <%@ page import="org.jfree.data.\*" %>

### <%@ page import="org.jfree.data.jdbc.JDBCCategoryDataset"%>

### <%

### String topic=request.getParameter("id");

### int c1=0, c2=0, c3=0,c4=0;

### try

### {

### Connection con = databasecon.getconnection();

### Statement st=con.createStatement();

### st.executeUpdate("delete from graph");

### ResultSet rs=st.executeQuery("select count(\*), SUM(TOT),SUM(DIS) from metadata where topic='"+topic+"' ");

### if(rs.next())

### {

### c1=rs.getInt(1);

### c2=rs.getInt(2);

### c3=rs.getInt(3);

### }

### c4=c2-c3;

### st.executeUpdate("insert into graph value('0','"+c1+"','"+c2+"','"+c3+"','"+c4+"')");

### String query="SELECT \* from graph";

### JDBCCategoryDataset dataset=new JDBCCategoryDataset("jdbc:mysql://localhost:3306/crawler","com.mysql.jdbc.Driver","root","root");

### dataset.executeQuery( query);

### JFreeChart chart = ChartFactory .createBarChart3D("EVALUATION GRAPH", "", "",dataset, PlotOrientation.VERTICAL, true, true, true);

### 

### ChartUtilities.saveChartAsJPEG(new File("D:/Apache Tomcat/webapps/SmartCrawler/images/logotype.jpg"), chart, 700, 300);

### response.sendRedirect("graph2.jsp");

### }

### catch (Exception e)

### {

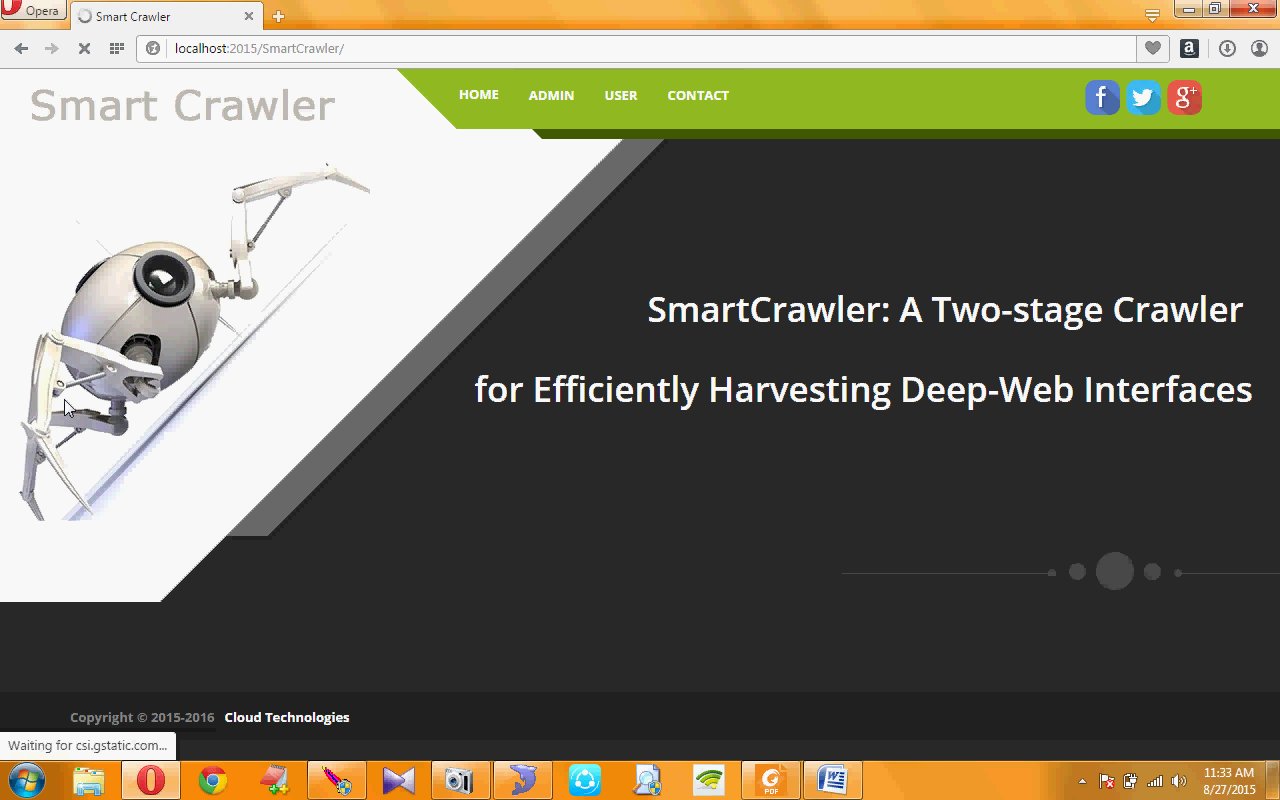
### System.out.println("Problem in creating chart."+e);

### }

### %>

### 5.2 Screen Captures

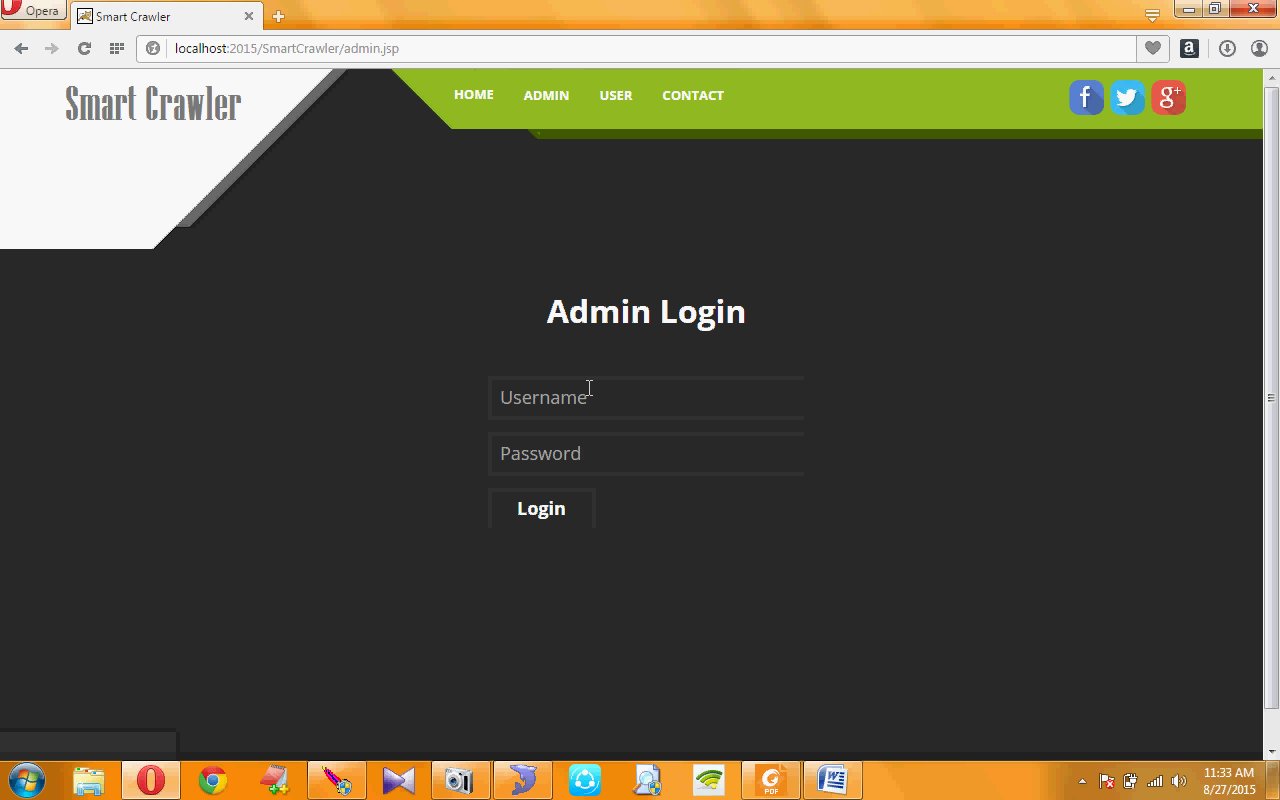
**5.2.1 Welcome Screen**

****

**Figure 5-1 Welcome Activity**

**Description:** After launching the application a Welcome message will come into view.

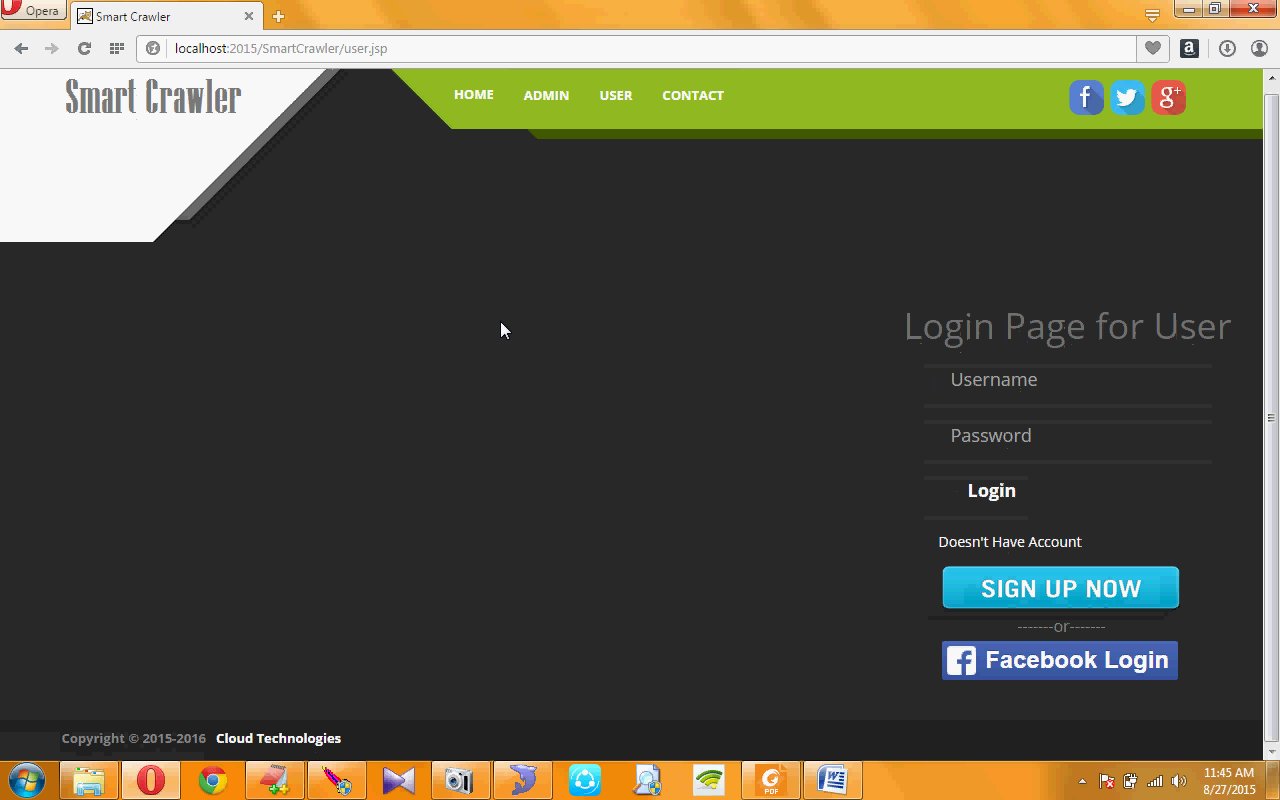
**5.2.2 Login Screen for Admin**

****

**Figure 5-2 Login Activity**

**Description:** The application has an admin login which is used to start the application to prevent authenticity infringement.

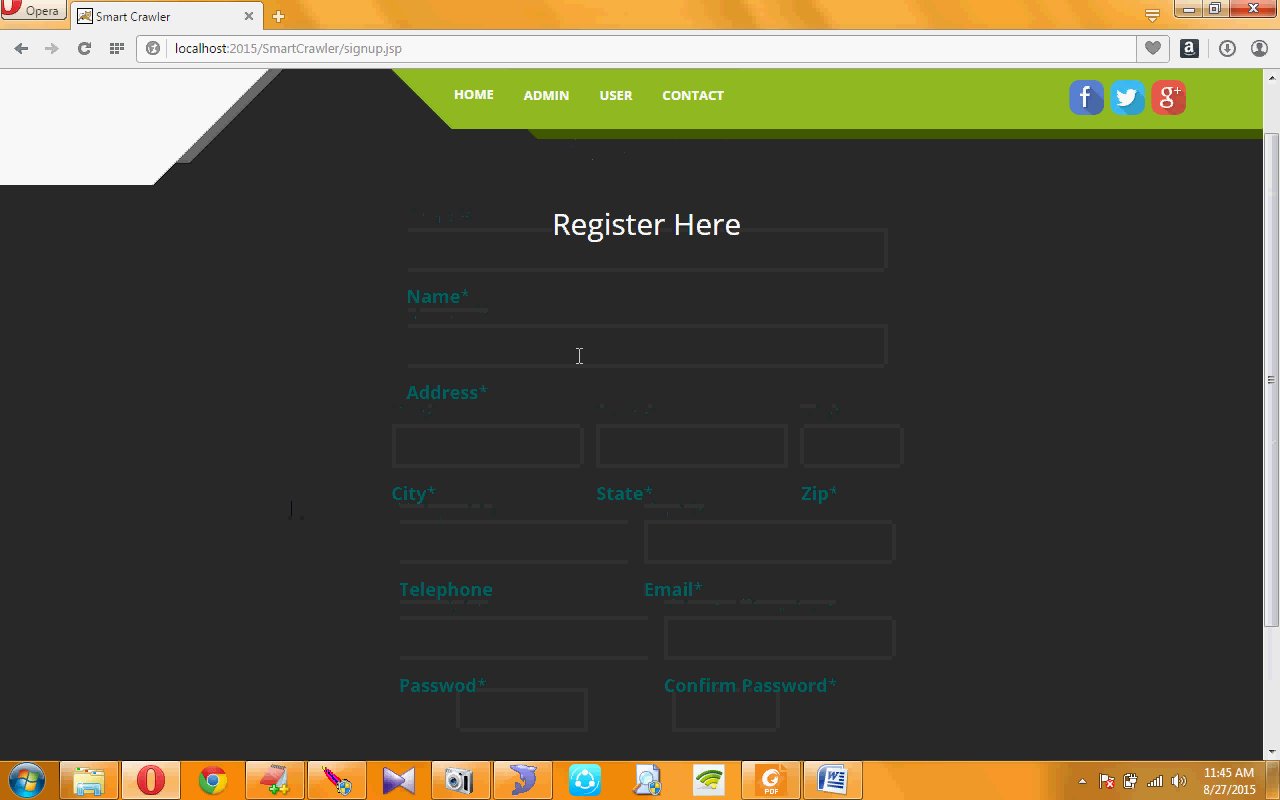
**5.2.3 Login Screen for User**

****

**Figure 5-3 Login Activity**

**Description:** The application has an User login which is used to start the application to prevent authenticity infringement.

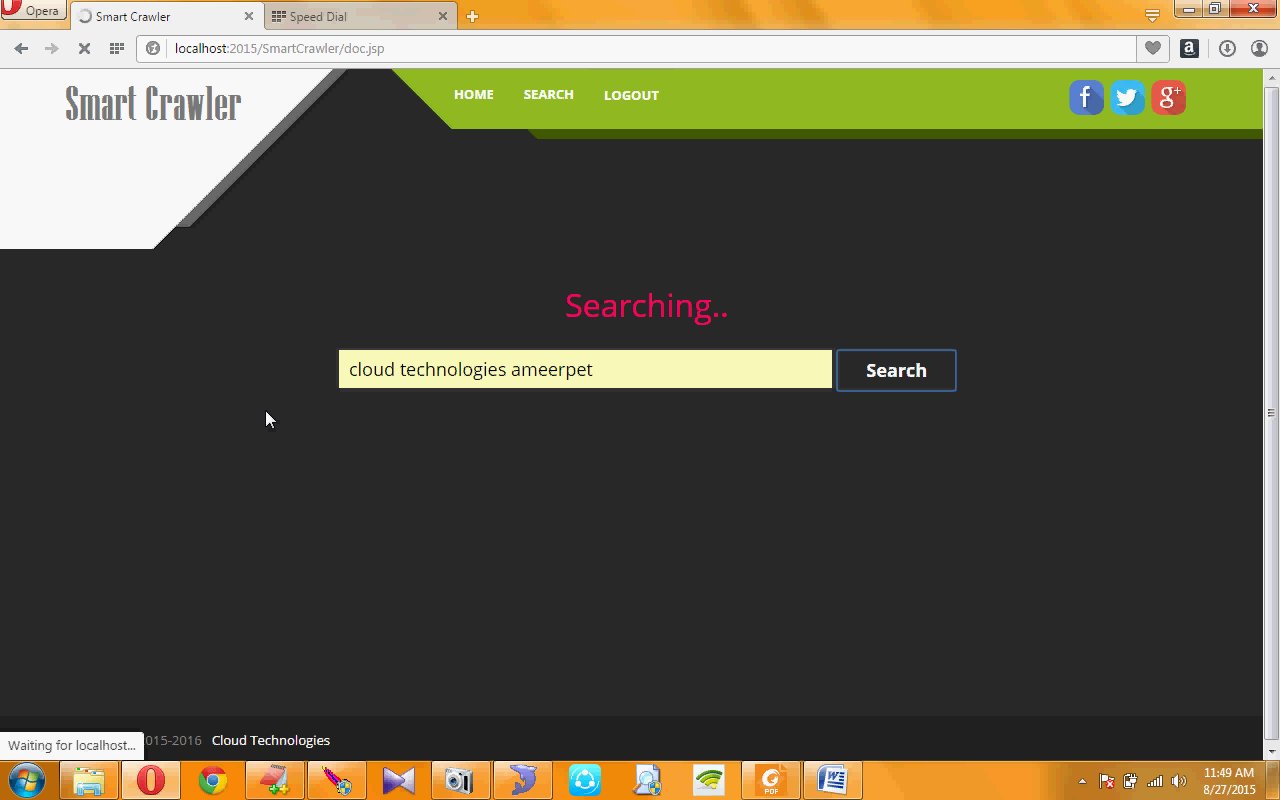
**5.2.4 User Registration**

****

**Figure 5-4 User Registration**

**Description:** In this activity is useful for the new user to register themselves by giving their valid details such as email id, user name, Phone number, etc. The user has to fill all the details else message is displayed to the user.

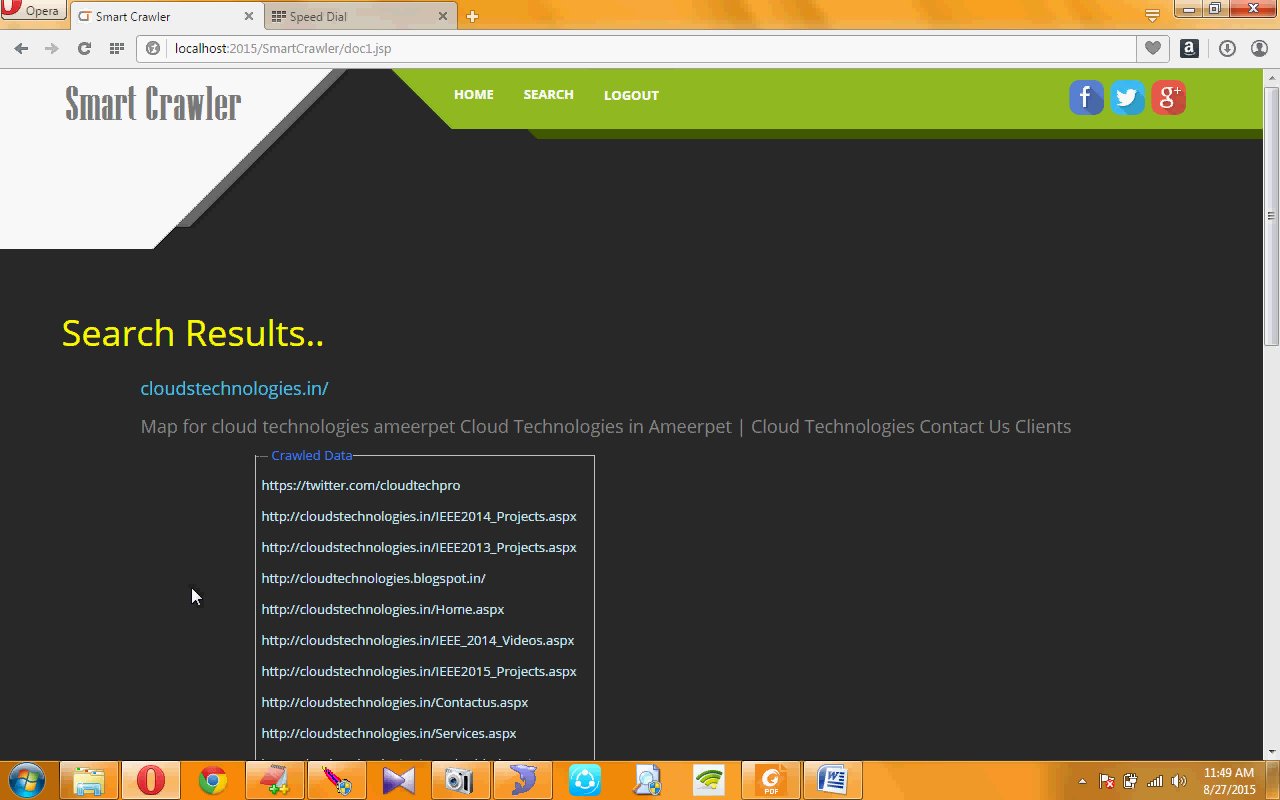
**5.2.5 User Search**

****

**Figure 5-5 User Search**

**Description:** This activity used to user search a query based on his requirement.

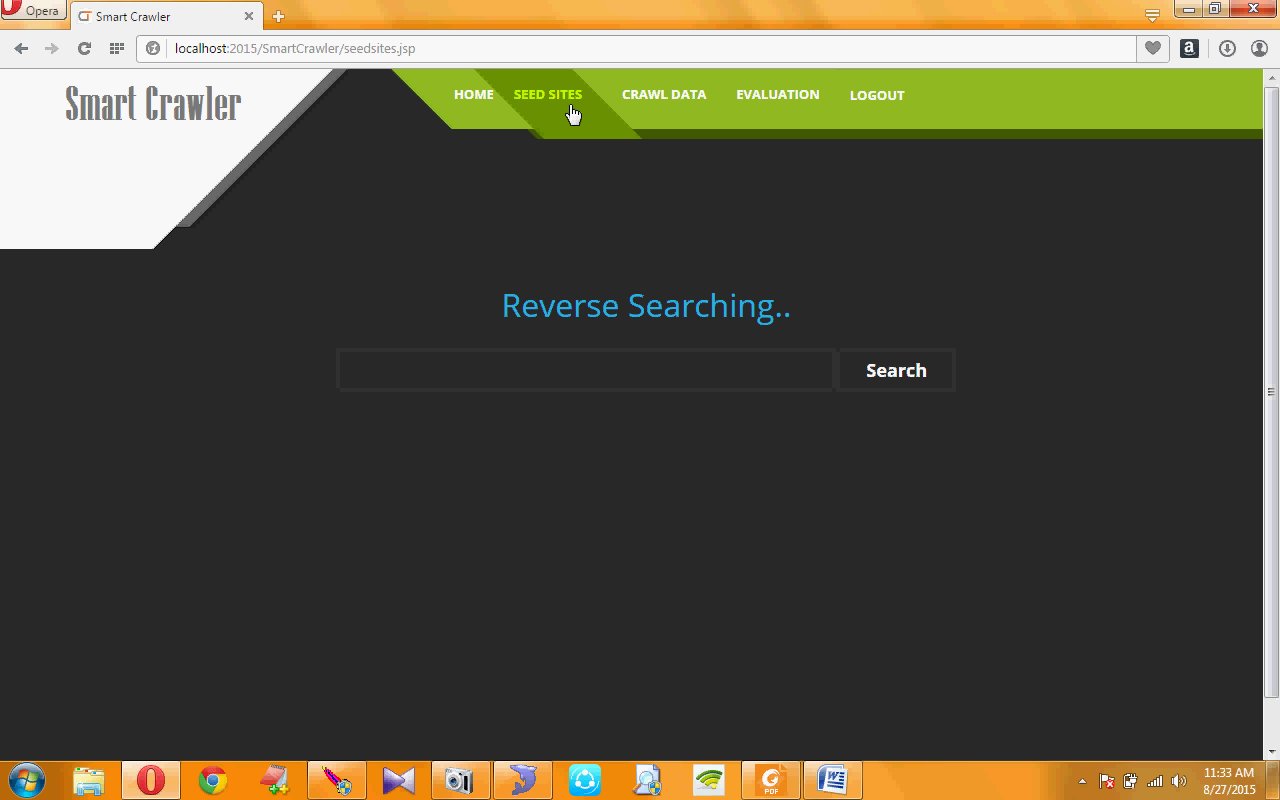
**5.2.6 Search Results**

****

**Figure 5-6 Search Results**

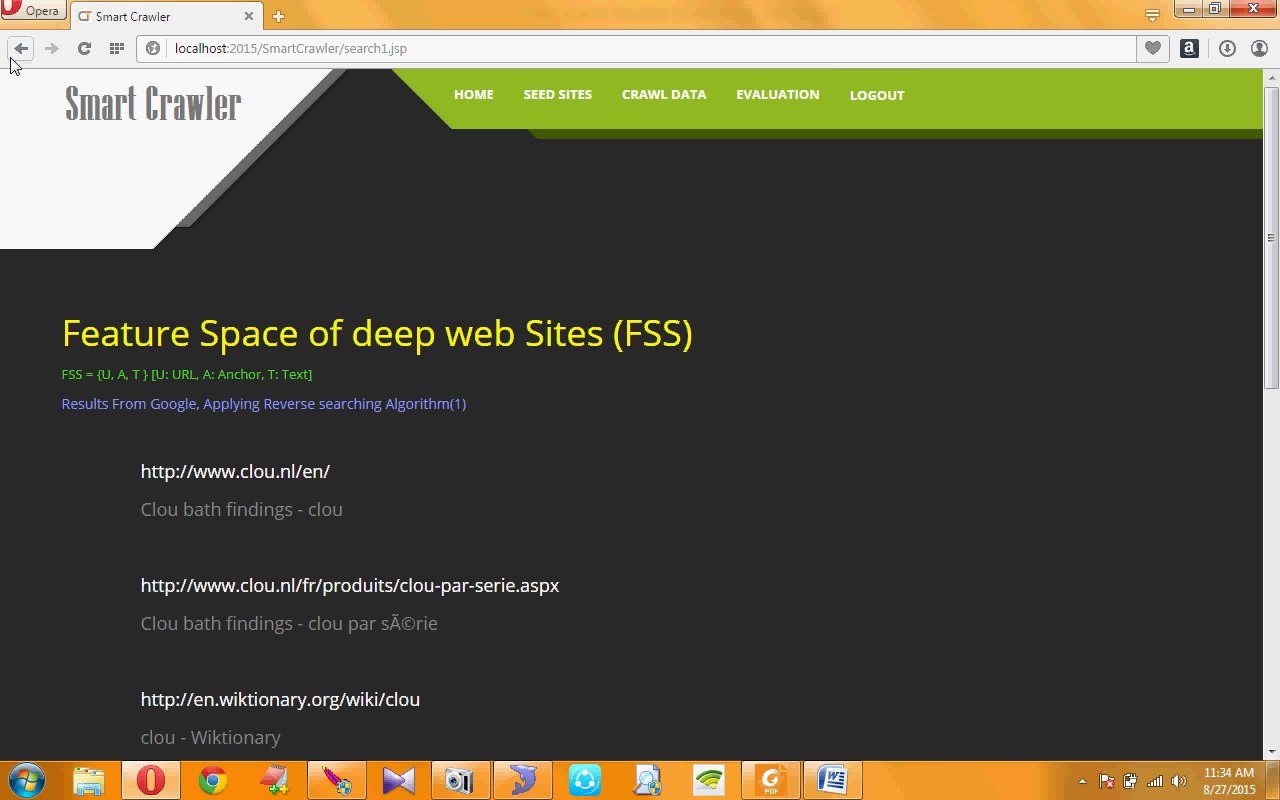
**Description:** When user select to search query results get from bing search engine and crawl results from form database.

**5.2.7 Seed Sites**

**Figure 5-7 Seed Sited**

**Description:** Search engine for get links to crawl from bing.

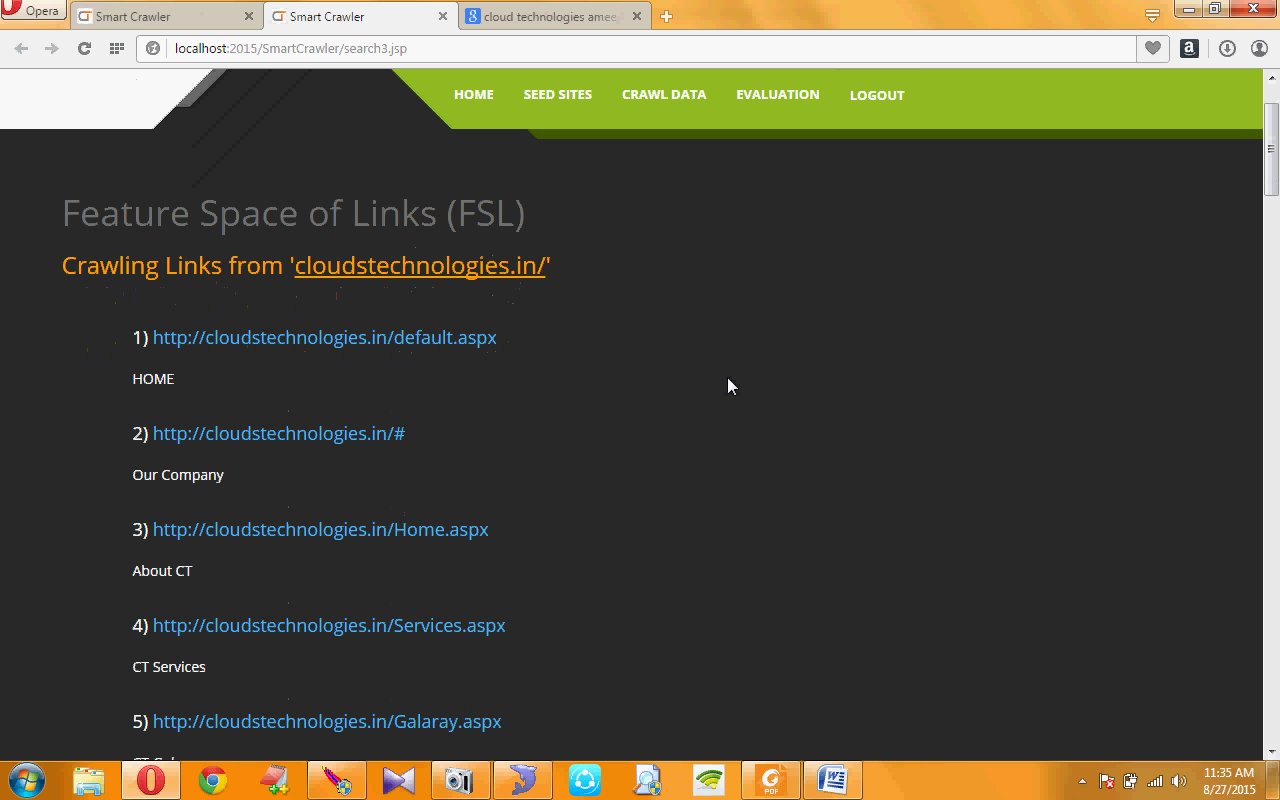
**5.2.8 Results from Bing**

****

**Figure 5-8 Results from Bing**

**Description:** Results got from bing search engine.

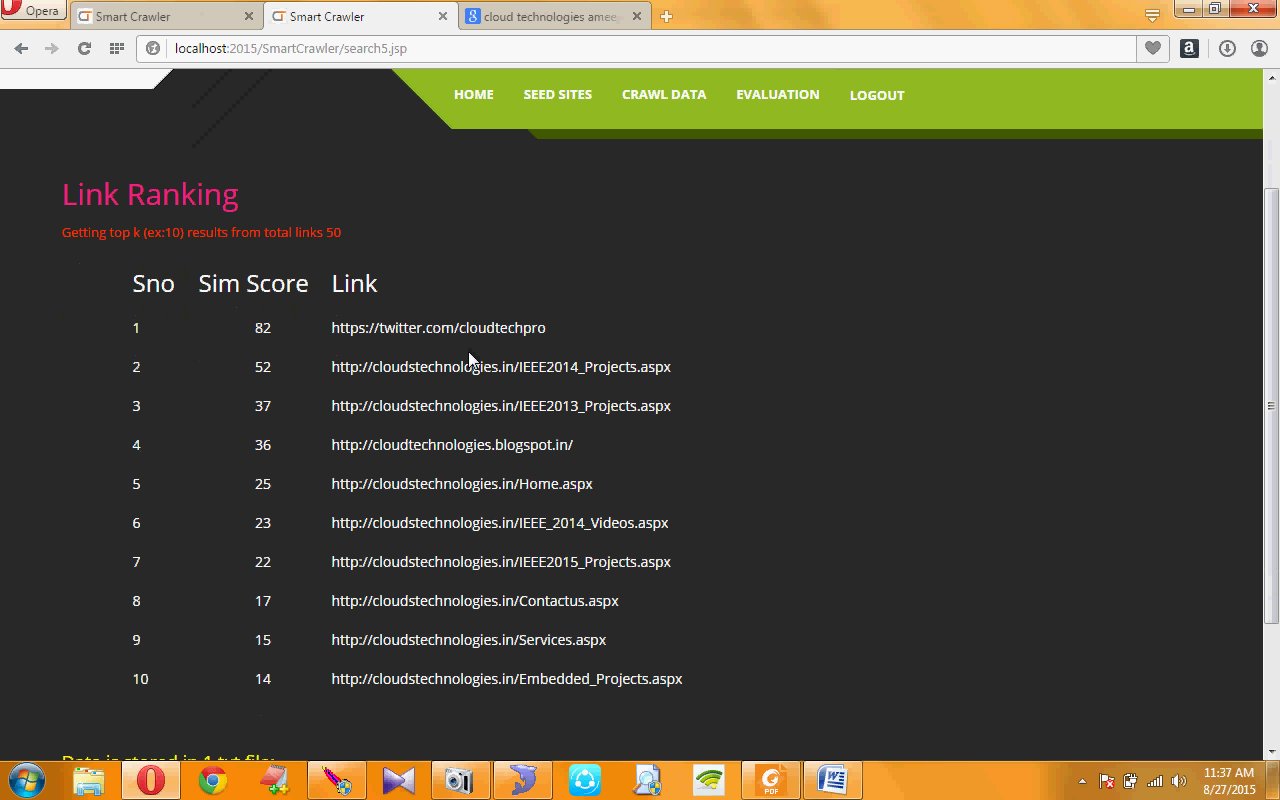
**5.2.9 Crawl Links**

****

**Figure 5-9 Crawl Links**

**Description:** These are internal links from a link.

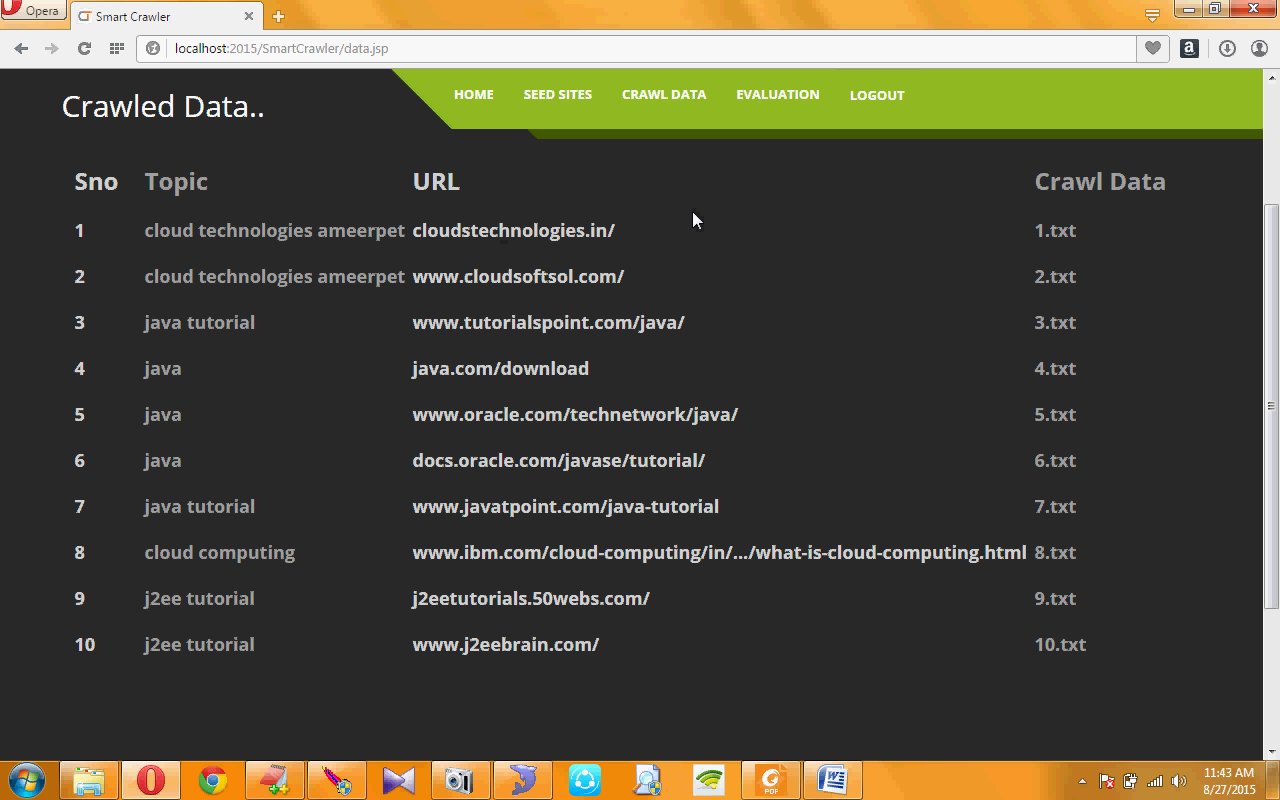
**5.2.10 Link Ranking**

****

**Figure 5-10 Link Ranking**

**Description:** Link ranking performed in crawl results.

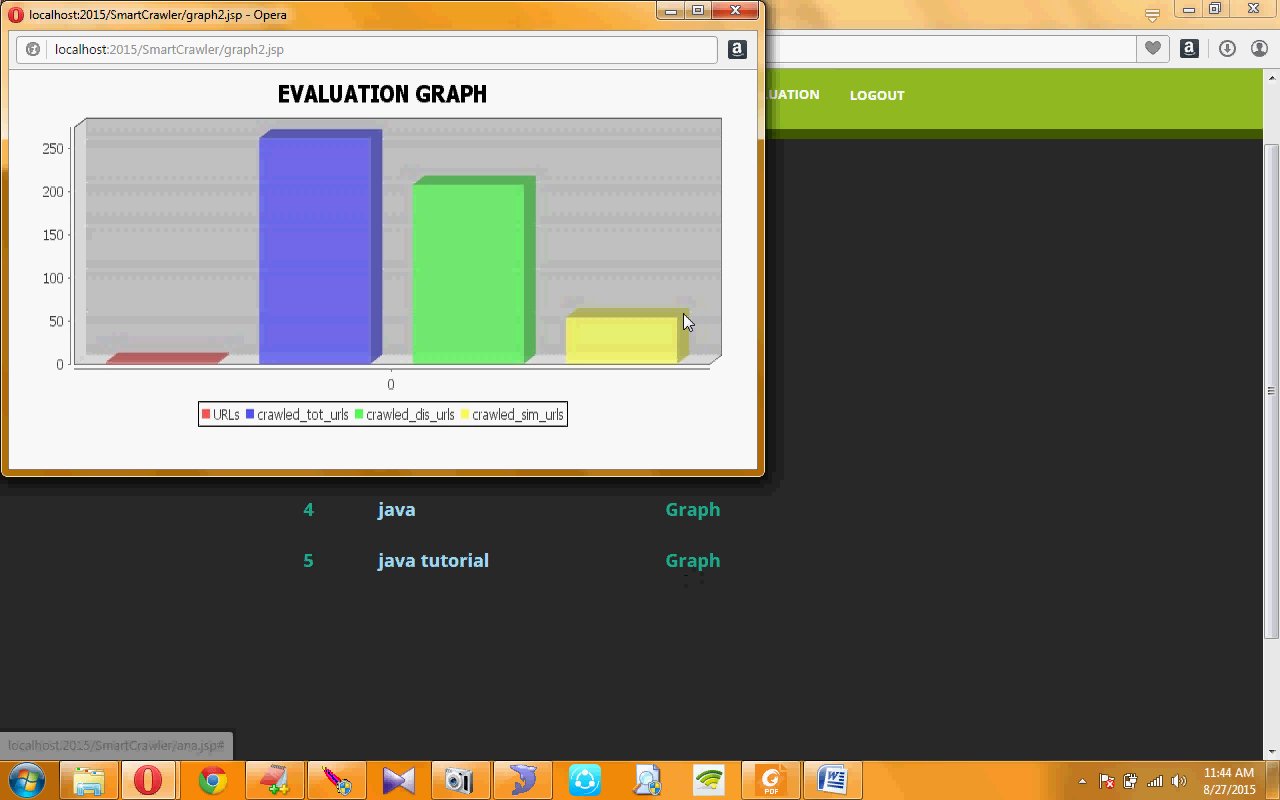
**5.2.11 Form Data**

****

**Figure 5-11 Form Data**

**Description:** Admin can list out total form data.

**5.2.12 Evaluations**

****

**Figure 5-12 Evaluation**

**Description:** Admin can see evaluation of crawl results

***CHAPTER - 6***

***TESTING***

*The chapter shows the various test cases.*

## 6. TESTING

### 6.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

### 6.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

### 6.3 White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

### 6.4 Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

### 

### 6.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

### 6.6 Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

|  |  |
| --- | --- |
| **Test Case 1** | |
| Test Case Name | Empty login fields testing |
| Description | In the login screen if the username and password fields are empty |
| Output | Login fails showing an alert box asking to enter username and password. |

**Figure 6-1 Test Case for Empty Login Fields**

|  |  |
| --- | --- |
| **Test Case 2** | |
| Test Case Name | Wrong login fields testing |
| Description | A unique username and password are set by administrator. On entering wrong username or password gives. |
| Output | Login fails showing an alert box username or password incorrect. |

|  |  |
| --- | --- |
| **Test Case 3** | |
| Test Case Name | Add to MySql testing |
| Description | Entity code and Name are mandatory fields. If left empty. |
| Output | Insert Fails and a toast message appears asking to enter entity code and name. |

|  |  |
| --- | --- |
| **Test Case 4** | |
| Test Case Name | Data Storing Testing |
| Description | Testing at database server side weather data is stored or not . |
| Output | n rows effeced |

***CHAPTER - 7***

* 1. ***CONCLUSIONS & BIBLIOGRAPHY***

*The chapter gives brief introduction of the project.*

## 7. CONCLUSION

### 8.1 Conclusions

In this paper, we propose an effective harvesting framework for deep-web interfaces, namely *Smart-Crawler*. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. *SmartCrawler* is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. *SmartCrawler* performs site-based locating by reversely searching the known deep web sites for center pages, which can effectively find many data sources for sparse domains. By ranking collected sites and by focusing the crawling on a topic, *SmartCrawler* achieves more accurate results. The in-site exploring stage uses adaptive link-ranking to search within a site; and we design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories. Our experimental results on a representative set of domains show the effectiveness of the proposed two-stage crawler, which achieves higher harvest rates than other crawlers. In future work, we plan to combine pre-query and post-query approaches for classifying deep-web forms to further improve the accuracy of the form classifier.

## 

## BIBLIOGRAPHY

[1] Peter Lyman and Hal R. Varian. How much information? 2003. Technical report, UC Berkeley, 2003.

[2] Roger E. Bohn and James E. Short. How much information? 2009 report on american consumers. Technical report, University of California, San Diego, 2009.

[3] Martin Hilbert. How much information is there in the ”information society”? *Significance*, 9(4):8–12, 2012.

[4] Idc worldwide predictions 2014: Battles for dominance – and survival – on the 3rd platform. <http://www.idc.com/> research/Predictions14/index.jsp, 2014.

[5] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. *Journal of electronic publishing*, 7(1), 2001.

[6] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In *Proceedingsof the sixth ACM international conference on Web search and datamining*, pages 355–364. ACM, 2013.

[7] Infomine. UC Riverside library. <http://lib-www.ucr.edu/>, 2014.

[8] Clusty’s searchable database dirctory. <http://www.clusty>. com/, 2009.

[9] Booksinprint. Books in print and global books in print access.http://booksinprint.com/, 2015.

[10] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR*, pages 44–55, 2005.

[11] Denis Shestakov. Databases on the web: national web domainsurvey. In *Proceedings of the 15th Symposium on InternationalDatabase Engineering & Applications*, pages 179–184. ACM, 2011.

[12] Denis Shestakov and Tapio Salakoski. Host-ip clustering technique for deep web characterization. In *Proceedings of the12th International Asia-Pacific Web Conference (APWEB)*, pages 378–380. IEEE, 2010.

[13] Denis Shestakov and Tapio Salakoski. On estimating the scale of national deep web. In *Database and Expert SystemsApplications*, pages 780–789. Springer, 2007.

[14] Shestakov Denis. On building a search interface discoverysystem. In *Proceedings of the 2nd international conference onResource discovery*, pages 81–93, Lyon France, 2010. Springer.

[15] Luciano Barbosa and Juliana Freire. Searching for hidden-web databases. In *WebDB*, pages 1–6, 2005.

[16] Luciano Barbosa and Juliana Freire. An adaptive crawler for locating hidden-web entry points. In *Proceedings of the16th international conference on World Wide Web*, pages 441–450. ACM, 2007.

[17] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640, 1999.

[18] Jayant Madhavan, David Ko, Łucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy. Google’s deep web crawl. *Proceedings of the VLDB Endowment*, 1(2):1241–1252, 2008.

[19] Olston Christopher and Najork Marc. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

[20] Balakrishnan Raju and Kambhampati Subbarao. Sourcerank: Relevance and trust assessment for deep web sources based on inter-source agreement. In *Proceedings of the 20th internationalconference on World Wide Web*, pages 227–236, 2011.

[21] Balakrishnan Raju, Kambhampati Subbarao, and Jha Manishkumar. Assessing relevance and trust of the deep web sources and results based on inter-source agreement. *ACMTransactions on the Web*, 7(2):Article 11, 1–32, 2013.

[22] Mustafa Emmre Dincturk, Guy vincent Jourdan, Gregor V.

**Websites Referred:**

http://java.sun.com

http://www.sourcefordgde.com

http://www.networkcomputing.com/

<http://www.roseindia.com/>