

## ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our esteemed institute “Gayatri Vidya Parishad College of Engineering (Autonomous)”, which has provided us an opportunity to fulfill our cherished desire.

We express our sincere thanks to our Principal Dr. **A.B.Koteswara Rao** for his encouragement to us during the course of this project.

We express our heartfelt thanks and acknowledge our indebtedness to **Prof. Dr. K. B. MADHURI**, Head of the Department, Department of Information Technology.

We express our profound gratitude and our deep indebtedness to our guide and project coordinator **Mr. D.NAGA TEJ**, Assistant Prof. whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing our present project “WEB SEARCH ENGINE”.

We would also like to thank all the members of teaching and non-teaching staff of the Information Technology Department for all their support in completion of our project.

### **Project Members:**

**Ch. Ravi (10131A1219)**  
**D. Bharat (10131A1221)**  
**D.S.M. Kumar (10131A1224)**  
**K.Chaitanya(10131A1236)**

## **ABSTRACT**

A Web Search Engine is a software system that is designed to search for information on the World Wide Web.

The web search engine system allows us to search the sample HTML pages based on exact keyword match and retrieve them in the order of keyword specific like count and view count.

It allows the user to specify the relative importance to their search keywords through which the problem of differentiating the importance of one keyword from that of another is solved.

It maintains a database that stores the sample HTML pages according to specific keywords to implement domain based search to minimise the search space. Here a user specifies the domains in which he was interested like mobiles, cars e.t.c. In this instead of considering the view count of the page as a whole we are considering keyword specific view count. In this along with the view count we are taking feedback from the users in to account asking him whether he got the information he was searching for or not.

# CONTENTS

<b>1. INTRODUCTION.....</b>	<b>5</b>
<b>1.1 FUNCTIONALITY.....</b>	<b>5</b>
<b>1.2 EXISTING SYSTEM AND PROPOSED SYSTEM.....</b>	<b>5</b>
<b>2. ANALYSIS AND SRS DOCUMENT.....</b>	<b>8</b>
<b>3. REQUIREMENTS.....</b>	<b>11</b>
<b>3.1 HARDWARE REQUIREMENTS.....</b>	<b>11</b>
<b>3.2 SOFTWARE REQUIREMENTS.....</b>	<b>11</b>
<b>3.3 TECHNOLOGY--.....</b>	<b>13</b>
<b>4. DESIGN ..... </b>	<b>19</b>
<b>4.1 IMPORTANCE OF UML IN SOFTWARE DEVELOPMENT .....</b>	<b>19</b>
<b>4.2 UML DIAGRAMS:.....</b>	<b>29</b>
<b>5. DEVELOPMENT .....</b>	<b>35</b>
<b>6. IMPLEMENTATION .....</b>	<b>37</b>
<b>6.1 SAMPLE CODE:.....</b>	<b>37</b>
<b>6.2 OUTPUT.....</b>	<b>60</b>
<b>7. TESTING.....</b>	<b>65</b>
<b>7.1 INTRODUCTION.....</b>	<b>65</b>
<b>7.2 TYPES OF TESTING.....</b>	<b>68</b>
<b>8. CONCLUSION.....</b>	<b>74</b>

<b>9. REFERENCES.....</b>	<b>75</b>
<b>10. BIBLIOGRAPHY.....</b>	<b>76</b>

# 1. INTRODUCTION

## 1.1 Functionality:

Search engines are programs that search documents for specified keywords and returns a list of the documents where the keywords were found. A search engine is really a general class of programs. However, the term is often used to specifically describe systems like Google, Bing and Yahoo! Search that enable users to search for documents on the World Wide Web.

The main motive of this application is to search the sample HTML pages based on exact keyword match and retrieve them in the order of keyword specific like count and view count. This allows the user to specify the relative importance to their search keywords through which the problem of differentiating the importance of one keyword from that of another is solved. The provision of domain based search give benefit to the user as it reduces the search space and reduces the unnecessary results.

Here considering the keyword specific count instead of the view count of the page gives advantage to the user as the increment will be proceed on the keyword instead of the whole page.

In this along with the view count taking feedback from the users in to account asking him whether he got the information he was searching for or not. It is useful to calculate the rank of the keyword according to the specific page.

Also it provides registration to the website administrators to provide the information on the web. The registration contains the administrator organization's information, domain, keywords he want to provide for the site, importantly the URL of the page.

## 1.2 Existing and proposed system:

### Existing System:

- ▶ Traditional search engines treat all keywords as the same importance and cannot differentiate the importance of one keyword from that of another.

- ▶ Some of the traditional search engines lack an applicable classification mechanism to reduce the search space and improve the search results.
- ▶ Traditional search engines retrieve the web pages based on the view count.
- ▶ The problem with view count is that everyone will tend to open the web page which was displayed first leading to the increase of the view count. But the page may or may not contain the user required information.
- ▶ In some traditional systems when user open a webpage for desired keywords the view count will increment as a whole for the web page. But the web page contain various other keywords.

### **Proposed system:**

- ▶ The proposed system searches the sample HTML pages based on exact keyword match and retrieve them in the order of keyword specific like count and view count.
- ▶ In this we are allowing the user to specify the relative importance to their search keywords through which the problem of differentiating the importance of one keyword from that of another is solved.
- ▶ In this we are implementing a domain based search to minimise the search space.
- ▶ Here a user specifies the domains in which he was interested like mobiles, cars e.t.c.
- ▶ In this instead of considering the view count of the page as a whole we are considering keyword specific view count.
- ▶ In this along with the view count we are taking feedback from the users in to account asking him whether he got the information he was searching for or not.

**Advantages of proposed system:**

- ▶ Do not treat all keywords as the same importance and can differentiate the importance of one keyword from that of another.
- ▶ It reduces the search space and improves the search results.
- ▶ The view count will be incremented to given keyword rather than the whole page.

## **2. SYSTEM ANALYSIS**

### **PURPOSE:**

Search Engine is a web application, it searches the web pages for keywords imposed by the user and it gives the results back to the user. This search engine is more effective and gives results faster because of domain based search, it reduces the search space. The problems encountered with page count in traditional search engines are corrected in this system.

### **Feasibility Study:**

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses Of the existing business or proposed venture, opportunities and threats as presented by the Environment, the resources required to carry through, and ultimately the prospects for success. In Its simplest term, the two criteria to judge feasibility is cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations And management, marketing research and policies, financial data, legal requirements and tax Obligations. Generally, feasibility studies precede technical development and project Implementation.

### **Technical Feasibility:**

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not
- Whether the required resources are available - Manpower- programmers, testers & debuggers- Software and hardware.

### **Economic feasibility:**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the



benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

➤ **Cost-based study:**

It is important to identify cost and benefit factors, which can be categorized as follows: 1. Development costs; and 2. Operating costs. This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system.

➤ **Time-based study:**

This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

**Operational feasibility:**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

## **SOFTWARE REQUIREMENT SPECIFICATION:**

The Software Requirement Specification document itself states in precise and explicit language those functions and capabilities a software system (i.e. a software application, an ecommerce website, etc) must provide, as well as states any required constraints by which the system must abide. The SRS functions are blue print for completing a project with as little cost growth as possible. The SRS is often referred to the document because all subsequent project management documents, such as design specifications, statements of work software architecture specifications, testing, validation plans and documentation plans are related to it Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

## **3. REQUIREMENTS**

### **3.1 Functional Requirements:**

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

- User interface for registering websites.
- User interface for searching the information
- Selecting domain and keyword importance
- Collecting feedback
- Display the result

### **3.2 Non- Functional Requirements:**

This report refers to the non-functional requirements of the application. To develop the application, we must consider the following non-functional requirements:

- ☐ User interface
- ☐ Performance
- ☐ Modifiability
- ☐ Error Handling
- ☐ Security issues

## **System Specification:**

The system on which the project developed has the following configuration.

### **SERVER SIDE:**

#### **SOFTWARE REQUIREMENTS:-**

OPERATING SYSTEM :	WINDOWS XP AND ABOVE/LINUX
PROGRAMMING LANGUAGE :	JAVA (JDK1.5)
DATABASE :	DB2
IDE :	ECLIPSE INDIGO
PROCESSOR :	INTEL 2 <sup>nd</sup> GEN. I3 and ABOVE
SERVER :	APACHE TOMCAT 7.0

#### **HARDWARE REQUIREMENTS:-**

RAM :	8GB
HARDDISK :	500GB

### **CLIENT SIDE:**

#### **SOFTWARE REQUIREMENTS:-**

OPERATING SYSTEM:	ANY O.S that supports following browsers
BROWSER COMPATABILITY:	IE 7 AND ABOVE, ALL OTHER
BROWSERS	(Google Chrome, Firefox)

#### **HARDWARE REQUIREMENTS:-**

RAM :	128MB (Min.)
-------	--------------

### **3.3 TECHNOLOGIES:**

#### **ABOUT WEBSERVICES:**

Web services are open standard ( XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data Web Services can convert your existing applications into Web-applications. After analyzing the system's requirements we decided to implement the main server using a web service. This web service is the system's main service, which means that will provide an Application Programming Interface (API) or Web API for the rest of the applications in this system, such as Android application and Web site. This API has to implement all the functions the Android program and the Web site requires that have to do with the database. It is decided to use a web service because it will be called from different environments. The web services provide a high compatibility with any system. The reason for this is that these remote calls are serialized in XML, transferred over HTTP using SOAP protocol. The client accesses a Web Services Description Language (WSDL) file that specifies where is the web service, the web service's available functions, the parameters of these functions and the return types. Using this information the client creates an XML file (SOAP message) where function's name and parameters are serialized following the SOAP protocol. This message is sent via HTTP to the server, which is going to de-serialize it, run the right method and answer the client, using the same system. In our case these methods are going to work on the database.

#### **Web Services:**

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, and then waits for a corresponding XML response. Because all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with UNIX applications. Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be

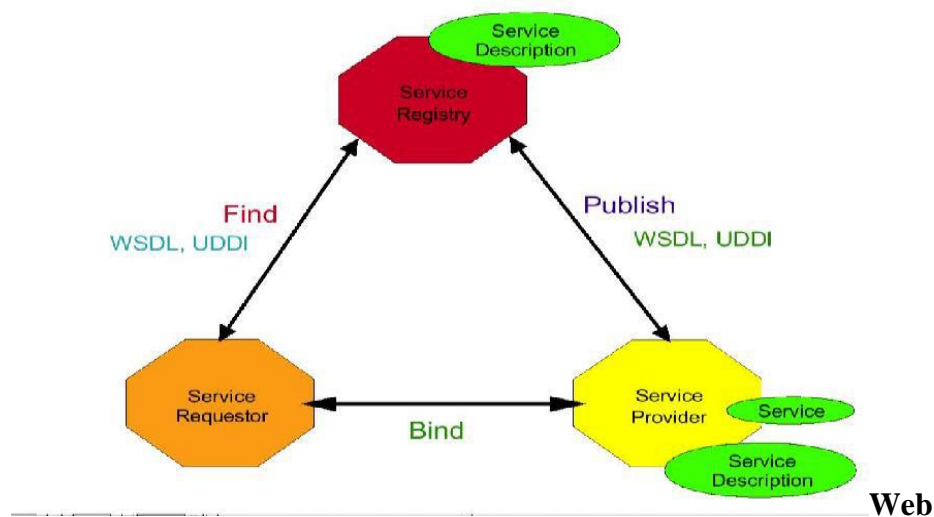
local, distributed, or Web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML. Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents. A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

### **Components of Web Services:**

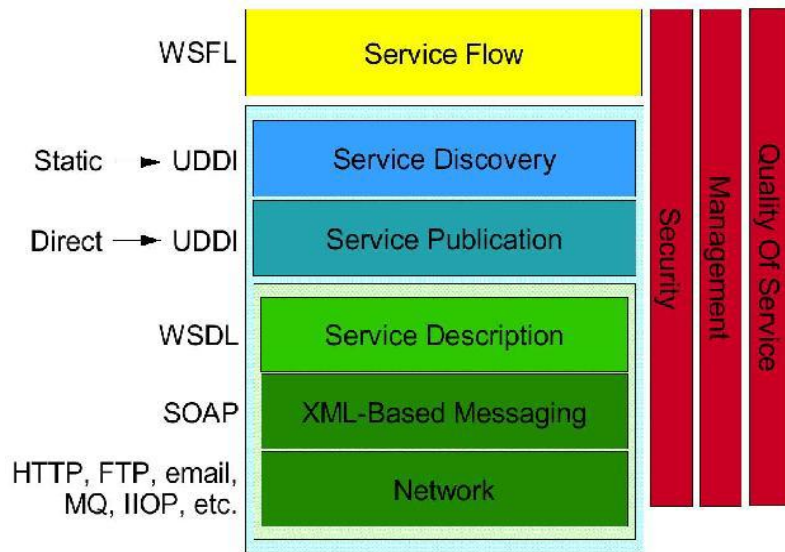
The basic Web services platform is XML + HTTP. All the standard Web Services works using following components

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

### **Web Services Architecture:**



## **Service Protocol Stack:**



A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers.

### **• Service transport:**

This layer is responsible for transporting messages between applications. Currently, this layer includes hypertext transfer protocol (HTTP), Simple Mail Transfer Protocol (SMTP), file transfer protocol (FTP), and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP).

### **• XML messaging:**

This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.

### **• Service description:**

This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

- **Service discovery:**

This layer is responsible for centralizing services into a common registry, and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI). As web services evolve, additional layers may be added, and additional technologies may be added to each layer. The bottom of the web service protocol stack is service transport. This layer is responsible for actually transporting XML messages between two computers.

- **Hyper Text Transfer Protocol (HTTP):**

Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic. This allows XMLRPC or SOAP messages to masquerade as HTTP messages. This is good if you want to easily integrate remote applications, but it does raise a number of security concerns.

- **Blocks Extensible Exchange Protocol (BEEP):**

One promising alternative to HTTP is the Blocks Extensible Exchange Protocol (BEEP). BEEP is a new IETF framework of best practices for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management. SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.



## 4. DESIGN

### **4.1 IMPORTANCE OF UML IN SOFTWARE DEVELOPMENT:**

#### **UML (Unified Modeling Language):**

UML stands for Unified Modeling Language [5]. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities involved in the project which later need to be built.

➤ The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly perspective. Each view is defined by a set of diagrams, which is as follows.

- Use case view
- Logical view
- Implementation view
- Process view
- Deployment view

#### **Diagrams:**

The diagrams contain the graphical elements arranged to illustrate a particular part or aspect of the system. A system model typically has several diagrams of varying types, depending on the goal for the model.

Software design is a process that gradually changes as various new, better and more complex methods with a broader understanding of the whole problem in general come into existence. There are various kinds of diagrams used in software design.

**Mainly these are as follows:**

- Use case diagrams
- Class diagram
- Object diagram
- Sequence diagrams
- Collaboration diagrams
- Activity diagrams
- State chart diagram
- Component diagram
- Deployment diagram

**Use case diagram:**

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system.

**The primary goals of Use Case diagrams include:**

- Providing a high-level view of what the system does
- Identifying the users ("actors") of the system
- Determining areas needing human-computer interfaces

Use Cases extend beyond pictorial diagrams. In fact, text-based use case descriptions are often used to supplement diagrams, and explore use case functionality in more detail.

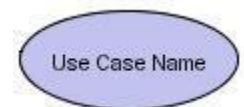
## **Graphical Notation:**

The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

**Actor :** An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.



**Use Case :** A Use Case is functionality provided by the system, typically described as verb object (e.g. Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.

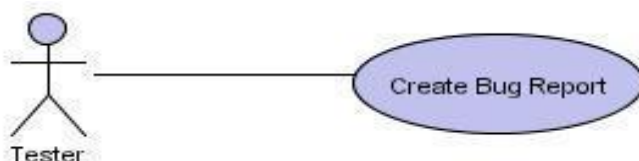


**Association:** Associations are used to link Actors with Use Cases, and indicate that an Actor participates in the Use Case in some form. Associations are depicted by a line connecting the



**Actor and the Use Case:**

The following image shows how these three basic elements work together to form a use case diagram.



### **Class Diagram:**

Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship. The Class diagram is one of the most widely used diagrams from the UML specification. Part of the popularity of Class diagrams stems from the fact that many CASE tools, such as Rational XDE, will auto-generate code in a variety of languages, including Java, C++, and C#, from these models. These tools can synchronize models and code, reducing your workload, and can also generate Class diagrams from object-oriented code. for those “code-then-design” maintenance projects.

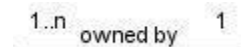
The elements on a Class diagram are classes and the relationships between them

**Class :** Classes are the building blocks in object-oriented

programming. A Class is depicted using a rectangle divided into three sections. The top section is the name of the Class. The middle section defines the properties of the Class. The bottom section lists the methods of the class.



**Association:** An Association is a generic relationship between two classes, and is modeled by a line connecting the two classes. This line can be qualified with the type of relationship, and can also feature multiplicity rules (e.g. one-to-one, one-to-many, many-to-many) for the relationship.



**Composition:** If a class cannot exist by itself, and instead must be a member of another class, then that class has a Composition relationship with the containing class. A Composition relationship is indicated by a line with a filled diamond.



**Dependency:** When a class uses another class, perhaps as a member variable or a parameter, and so "depends" on that class, a Dependency relationship is formed. A Dependency relationship is indicated by a dotted arrow.



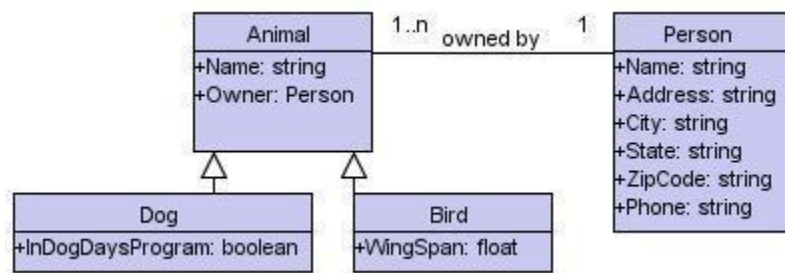
**Aggregation:** Aggregations indicate a whole-part relationship, and are known as "has-a" relationships. An Aggregation relationship is indicated by a line with a hollow diamond.



Generalization: A Generalization relationship is the equivalent of an *inheritance* relationship in object-oriented terms (an "is-a" relationship). A Generalization relationship is indicated by an arrow with a hollow arrowhead pointing to the base, or "parent", class.



Consider the example of a veterinary system. Animals served, such as dogs and birds, are tracked along with their owners. The following diagram models a potential solution. Since dogs and birds are "a kind of" animal, we use a Generalization relationship.



To validate your model, you can apply real-world data into instances of the classes. In fact, there is a diagram for precisely this task, the [Object Diagram](#)

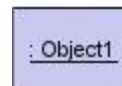
### **Sequence diagram:-**

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time is used, with two exceptions: Objects are written their names underlined and all instances in a relationship are shown.

Object diagrams are not as important as class diagrams but they can be used to exemplify a complex class diagram by showing what the actual instances and the relationships look like. Objects are also used as part of interaction diagrams that show the dynamic collaboration between a set of objects.

### **Notation:**

In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.



Object: Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

Actor: Actors can also communicate with objects, so they too can be listed as a column. An Actor is modeled using the ubiquitous symbol, the stick figure.



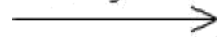
Lifeline: The Lifeline identifies the existence of the object over time.

The notation for a Lifeline is a vertical dotted line extending from an object.



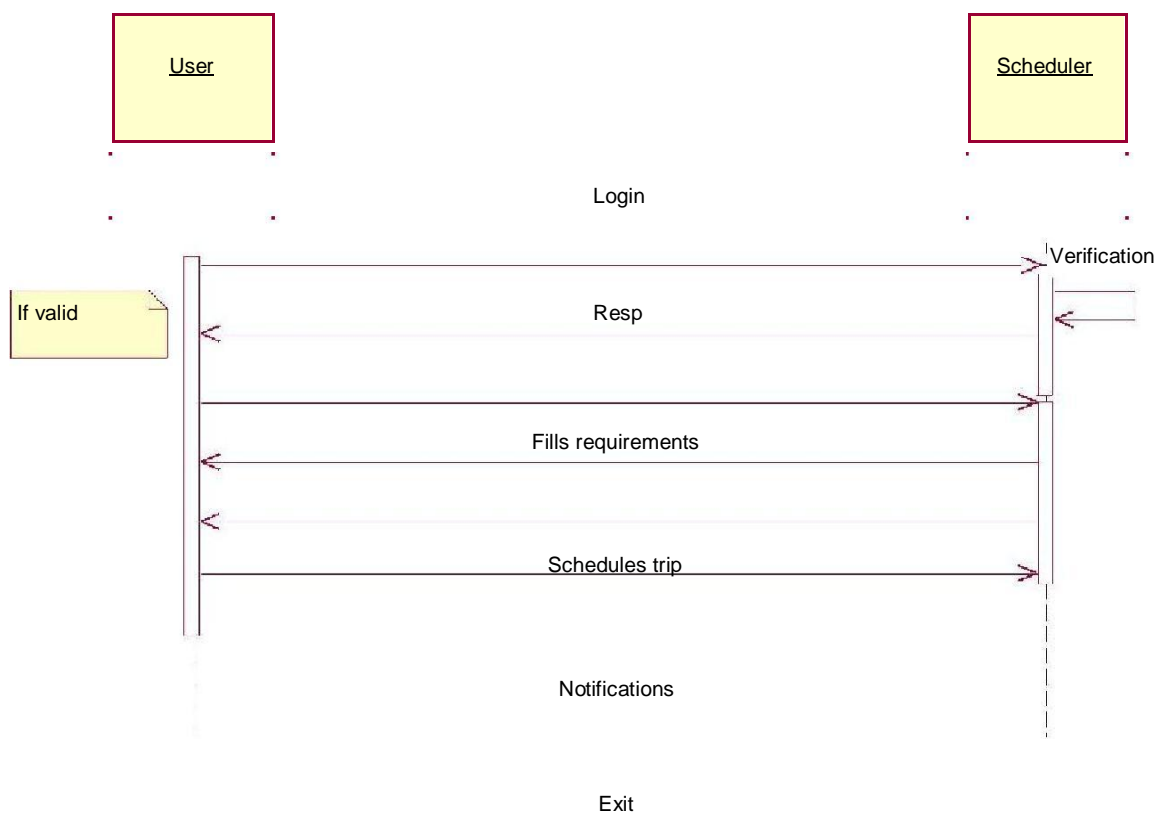
Activation: Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.

Message



Message: Messages, modeled as horizontal arrows between Activations, indicate the communications between objects.

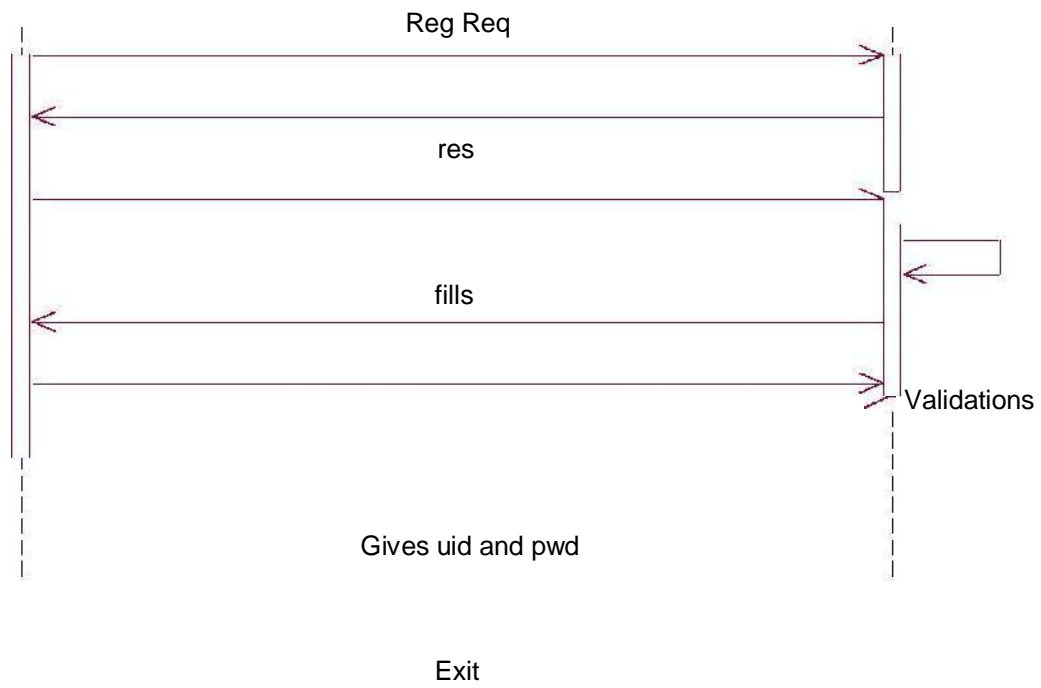
Following is an example of a Sequence diagram, using the default named objects. You can imagine many instances where a user performs an action in the user interface, and the system in turn calls another object for processing.





User

Adm in



### **Collaboration Diagram:**

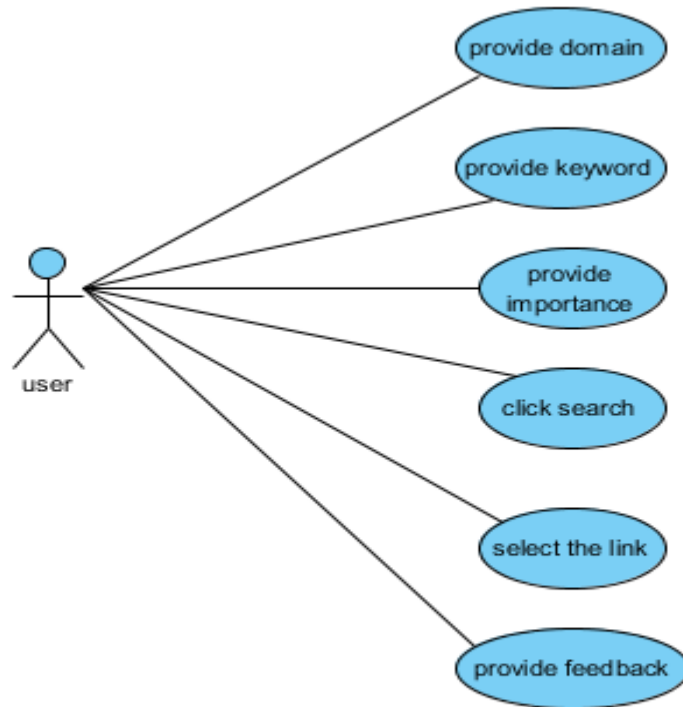
Like the other Behavioral diagrams, Collaboration diagrams model the interactions between objects. This type of diagram is a cross between an object diagram and a sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free-form arrangement of objects as found in an Object diagram. This makes it easier to see all integrations involving a particular object. In order to maintain the ordering of messages in such a free-form diagram, messages are labeled with a chronological number. Reading a Collaboration diagram involves starting at message 1.0, and following the messages from object to object.

### **Activity Diagram:**

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. When looking at an Activity diagram, you'll notice elements from [State diagrams](#). In fact, the Activity diagram is a variation of the state diagram where the "states" represent operations, and the transitions represent the activities that happen when the operation is complete. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

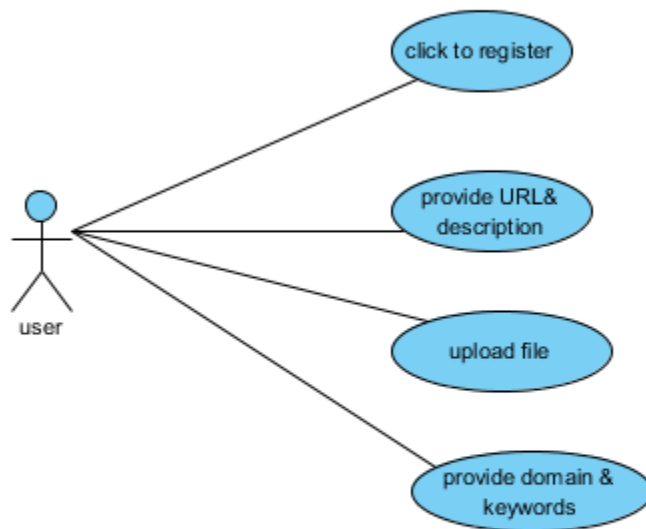
## 4.2 UML DIAGRAMS

### Search Use-case Report:

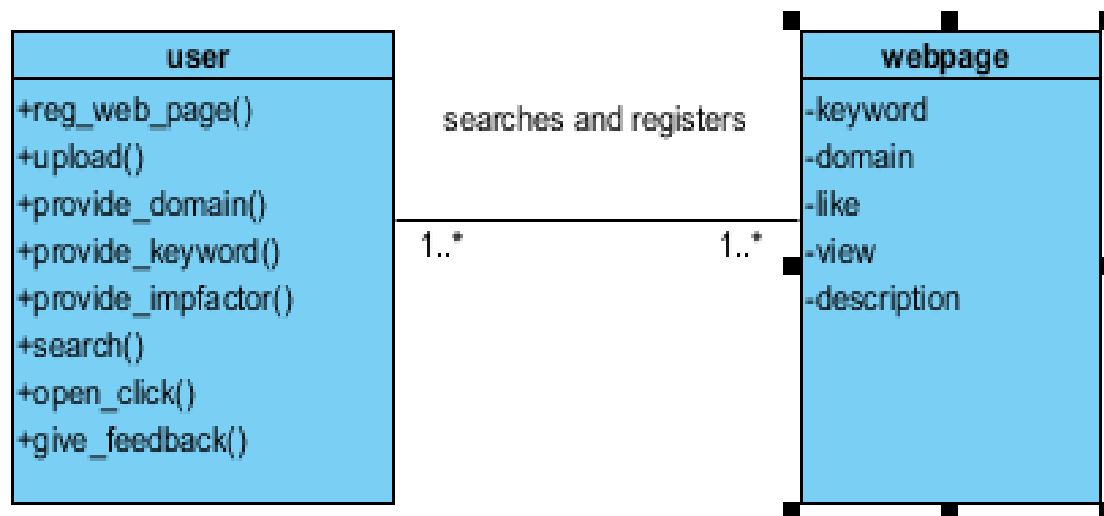


### Web Page Registration Use-case Report:

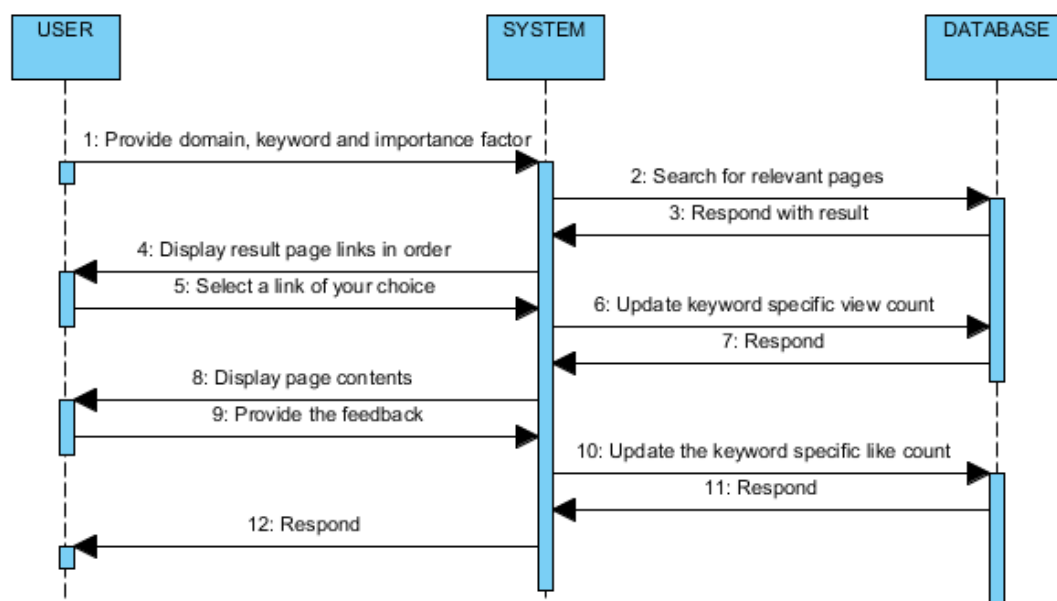
---



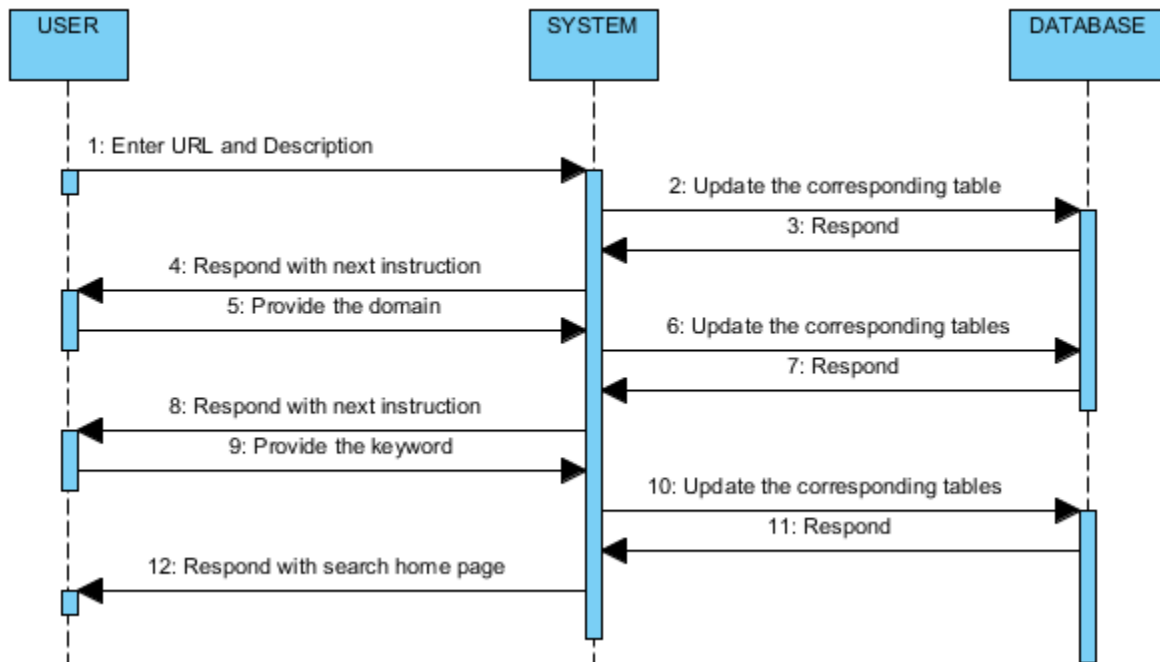
## Class Diagram



## Search Sequence Diagram

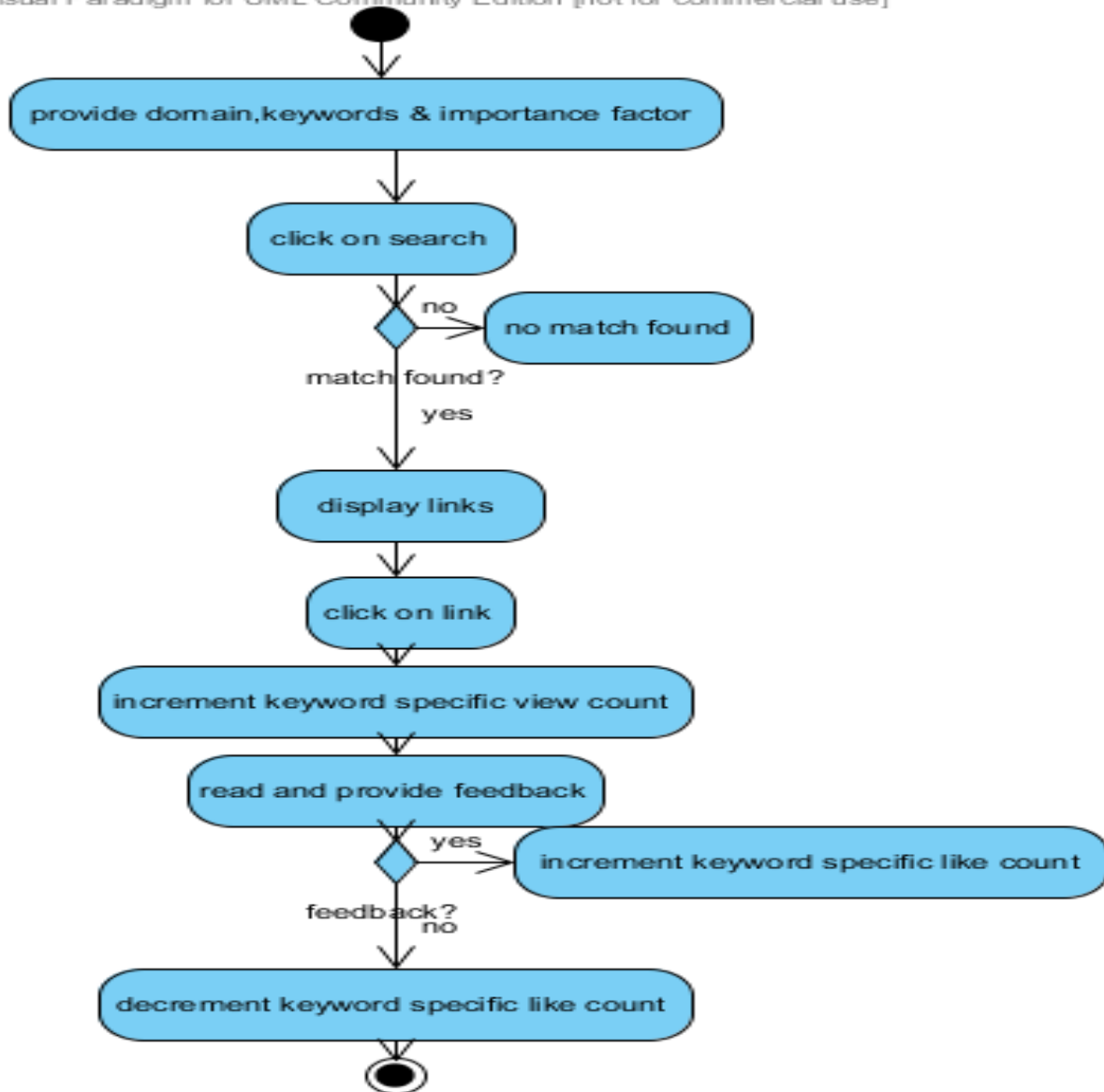


## Registration Sequence Diagram



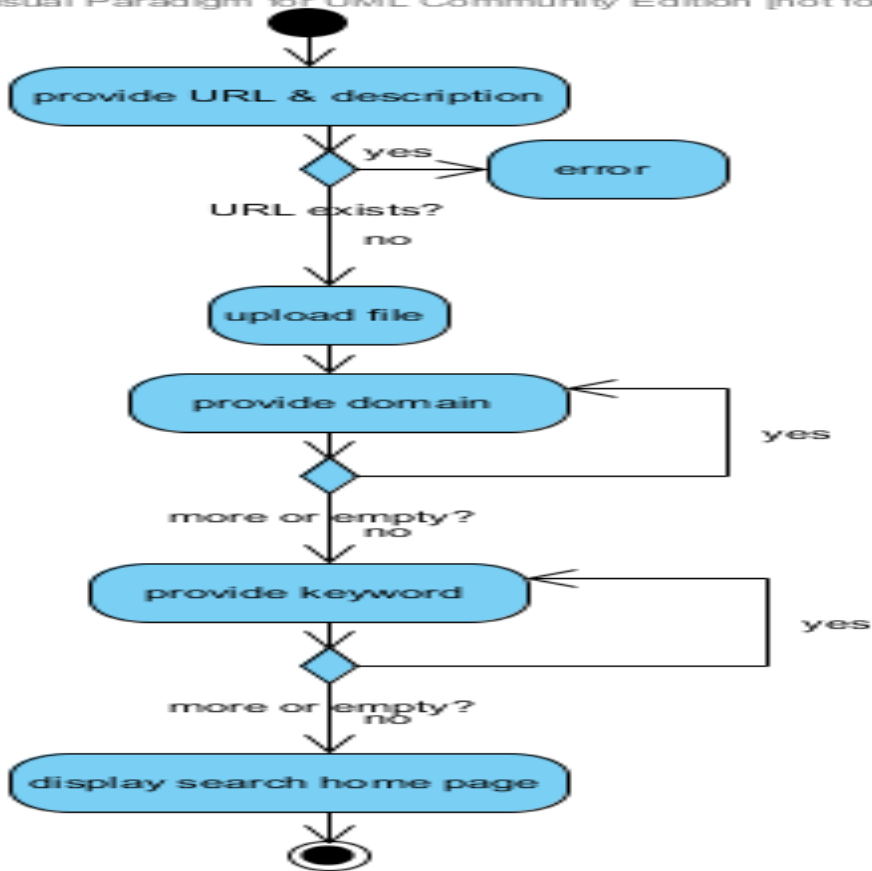
## Search Activity

Visual Paradigm for UML Community Edition [not for commercial use]



## Web Page Registration Activity

Visual Paradigm for UML Community Edition [not for cc]

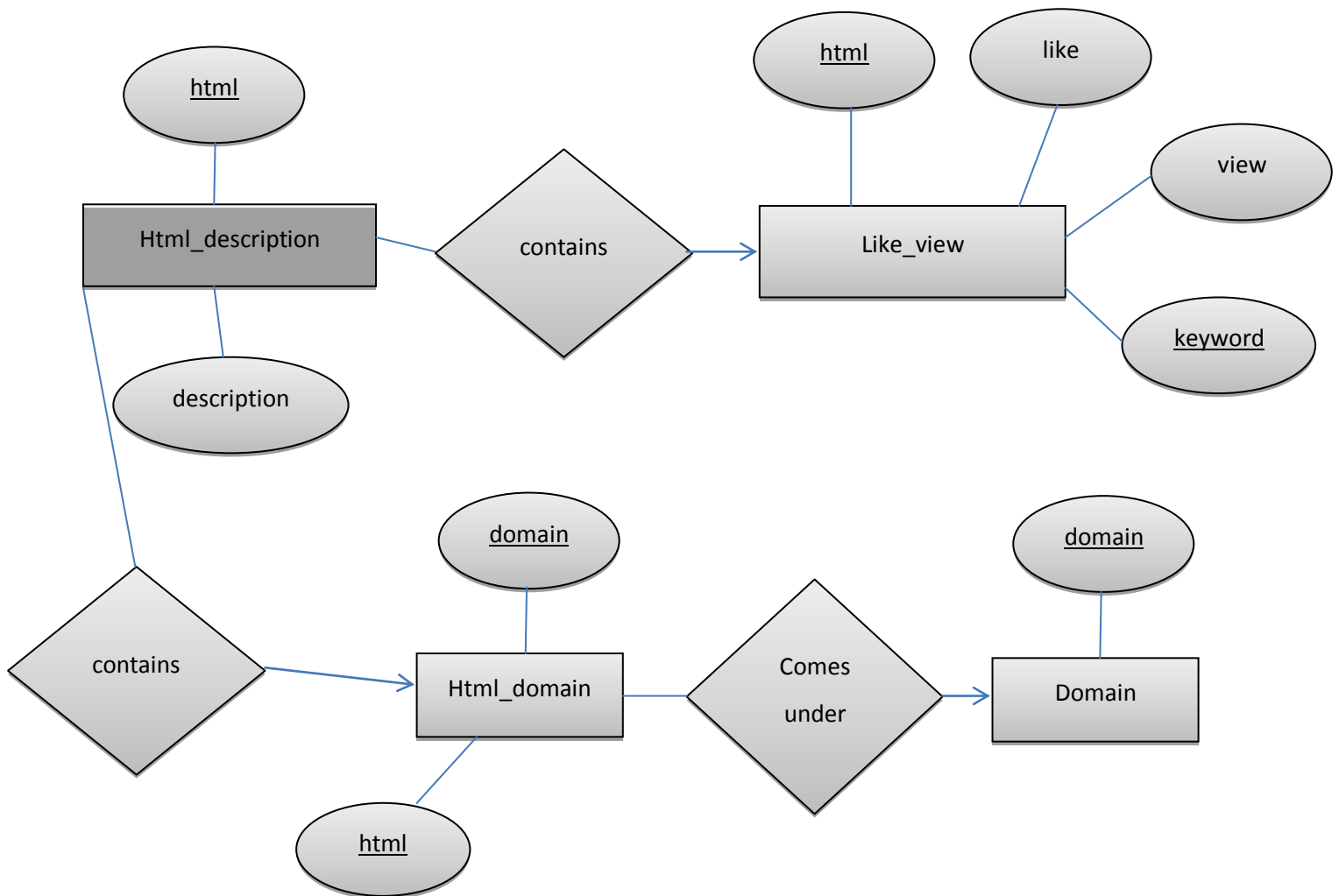




## 5. DEVELOPMENT


### DATABASE DESIGN

#### ER Diagram:-





### Domain:

#### Columns

Key	Name	Data type	Length	Nullable
	DOMAIN	VARCHAR	20	No


### Html domain:

#### Columns

Key	Name	Data type	Length	Nullable
	HTML	VARCHAR	30	No
	DOMAIN	VARCHAR	30	No



### Html description:

#### Columns

Key	Name	Data type	Length	Nullable
	HTML	VARCHAR	30	No
	NAME	VARCHAR	30	No

### Like view:

#### Columns

Key	Name	Data type	Length	Nullable
	HTML	VARCHAR	30	No
	KEYWORD	VARCHAR	30	No
	"LIKE"	INTEGER	4	No
	"VIEW"	INTEGER	4	No

## 6. IMPLEMENTATION

### 6.1 Sample Code

**index.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%response.sendRedirect("s1.jsp"); %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
</body>
</html>
```

**dm.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

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

<form name="f" action="S4" method="post">

<table>

    <%
        Class.forName("com.ibm.db2.jcc.DB2Driver");

        Connection
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");

        String v="";

        //out.println("connection established");

        Statement stmt=con.createStatement();

        ResultSet rs1=stmt.executeQuery("select * from domain");%>

    <tr><td>Domain</td><td><select name="dm">

        <%
            while(rs1.next())

            {

                v=rs1.getString(1);%>

                <option value="<%=v %>"><%=v %></option>

            <% }

        %>

    </select></td>    <td>If      new      domain:      </td><td><input      type="text"
name="dm1"/></td></tr></table>

    Want to add another domain?<input type="radio" value="yes" name="r1"/>yes<input
type="radio" value="no" name="r1" checked="checked"/>no <br/><input type="submit"
value="next" name="next"/>

```

```
</form></body>
```

```
</html>
```

### **key.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
    pageEncoding="ISO-8859-1"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Insert title here</title>
```

```
</head>
```

```
<body>
```

```
<form name="f" action="S5" method="post">
```

```
Enter the keyword: <input type="text" name="k"/>
```

```
<br/>
```

```
Want to add another keyword?<input type="radio" value="yes" name="r1"/>yes<input
type="radio" value="no" name="r1" checked="checked"/>no <br/><input type="submit"
value="next" name="next"/>
```

```
</form>
```

```
</body>
```

```
</html>
```

### **like.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<form name="f1" action="Sample1" method="post">

    Did you got the information you are looking for? <input type="radio" onclick="s1()"
    name="r" value="yes">yes<input onclick="s1()" type="radio" name="r" value="no">no

</form>

<script type='text/javascript'>

function s1(){document.f1.submit();}

</script>

</body>

</html>

```

### **reg.jsp**

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<form name="f" action="S3" method="post">

```

```

<table border="0">

<tr><td>URL</td><td><input type="text" name="url"/></td></tr>

<tr><td>Some Description About Web Page</td><td><input type="text" maxlength="50"
name="name"/></td></tr>


</table>

<input type="submit" value="next"/>


</form>

</body>

</html>

```

### **S1.jsp**

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

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

```

```

<form name="f" action="S1" method="post">

    <table>

        <%

            Class.forName("com.ibm.db2.jcc.DB2Driver");

            Connection
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");

            String v="";

            //out.println("connection established");

            Statement stmt=con.createStatement();

            ResultSet rs1=stmt.executeQuery("select * from domain");%>

            <tr><td>Domain</td><td><select          name="dm"><option          value="All"
selected>All</option>

            <%

                while(rs1.next())

                {

                    v=rs1.getString(1);%>

                    <option value="<%=v %>"><%=v %></option>

                <% }

            %>

            </select></td></tr></table>

            <br/><input  type="text"  name="k1"  /><select  name="p1"><option  value="0"
selected>None</option><option          value="1">Less          Important</option><option
value="2">Important</option><option          value="3">Very          Important</option><option
value="4">Very Very Important</option></select>

            <br/><input  type="text"  name="k2"/><select  name="p2"><option  value="0"
selected>None</option><option          value="1">Less          Important</option><option
value="2">Important</option><option          value="3">Very          Important</option><option

```



```
value="4">Very Very Important</option></select>
```

```
    <br/><input type="text" name="k3"/><select name="p3"><option value="0"
selected>None</option><option value="1">Less Important</option><option
value="2">Important</option><option value="3">Very Important</option><option
value="4">Very Very Important</option></select>
```

```
    <br/><input type="text" name="k4"/><select name="p4"><option value="0"
selected>None</option><option value="1">Less Important</option><option
value="2">Important</option><option value="3">Very Important</option><option
value="4">Very Very Important</option></select>
```

```
    <br/><input type="submit" name="s" value=" search " />
```

```
    <br/><br/>
```

```
    Want to Register a new link with us?<a href="reg.jsp">click here</a>
```

```
</form>
```

```
</body>
```

```
</html>
```

## **upload.jsp**

```
<% @ page import="java.io.*,java.util.*, javax.servlet.*" %>
```

```
<% @ page import="javax.servlet.http.*" %>
```

```
<% @ page import="org.apache.commons.fileupload.*" %>
```

```
<% @ page import="org.apache.commons.fileupload.disk.*" %>
```

```
<% @ page import="org.apache.commons.fileupload.servlet.*" %>
```

```
<% @ page import="org.apache.commons.io.output.*" %>
```

```
<%
```

```
File file ;
```

```
int maxFileSize = 5000 * 1024;
```

```
int maxMemSize = 5000 * 1024;
```

```

ServletContext context = pageContext.getServletContext();

String filePath = context.getInitParameter("file-upload");

// Verify the content type

String contentType = request.getContentType();

if ((contentType.indexOf("multipart/form-data") >= 0)) {

    DiskFileItemFactory factory = new DiskFileItemFactory();

    // maximum size that will be stored in memory

    factory.setSizeThreshold(maxMemSize);

    // Location to save data that is larger than maxMemSize.

    factory.setRepository(new File("c:\\temp"));

    // Create a new file upload handler

    ServletFileUpload upload = new ServletFileUpload(factory);

    // maximum file size to be uploaded.

    upload.setSizeMax( maxFileSize );

    try{

        // Parse the request to get file items.

        List fileItems = upload.parseRequest(request);

        // Process the uploaded file items

        Iterator i = fileItems.iterator();

        out.println("<html>");
    }
}

```

```

out.println("<head>");

out.println("<title>JSP File upload</title>");

out.println("</head>");

out.println("<body>");

while ( i.hasNext () )

{

    FileItem fi = (FileItem)i.next();

    if ( !fi.isFormField () )

    {

        // Get the uploaded file parameters

        String fieldName = fi.getFieldName();

        String fileName = fi.getName();

        boolean isInMemory = fi.isInMemory();

        long sizeInBytes = fi.getSize();

        // Write the file

        if( fileName.lastIndexOf("\\") >= 0 ){

            file = new File( filePath +

            fileName.substring( fileName.lastIndexOf("\\"))) ;

        }else{

            file = new File( filePath +

            fileName.substring(fileName.lastIndexOf("\\")+1)) ;

        }

        fi.write( file ) ;

        out.println("Uploaded Filename: " + filePath +

        fileName + "<br>");
    }
}

```

```

        }
    }
    out.println("</body>");
    out.println("</html>");
    response.sendRedirect("dm.jsp");
} catch (Exception ex) {
    System.out.println(ex);
}
} else {
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet upload</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<p>No file uploaded</p>");
    out.println("</body>");
    out.println("</html>");
}
%>

```

## **S2.java**

```

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

```

```
import java.sql.ResultSet;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
```

```
/**
```

```
 * Servlet implementation class for Servlet: S2
```

```
 *
```

```
 */
```

```
public class S2 extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {  
    static final long serialVersionUID = 1L;
```

```
    /* (non-Java-doc)
```

```
        * @see javax.servlet.http.HttpServlet#HttpServlet()
```

```
        */
```

```
    public S2() {  
        super();  
    }
```

```
    /* (non-Java-doc)
```

```
        *      @see      javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,  
        HttpServletRequest response)
```

```
        */
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
    // TODO Auto-generated method stub
```

```
}
```

```
/* (non-Java-doc)
```

```
 *      @see      javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
HttpServletRequest response)
```

```
 */
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
    // TODO Auto-generated method stub
```

```
    PrintWriter pw=response.getWriter();
```

```
    String html=request.getParameter("r");
```

```
    //pw.print(html);
```

```
    HttpSession hs=request.getSession(true);
```

```
    hs.setAttribute("html", html);
```

```
    String k1=(String) hs.getAttribute("k1");
```

```
    String k2=(String) hs.getAttribute("k2");
```

```
    String k3=(String) hs.getAttribute("k3");
```

```
    String k4=(String) hs.getAttribute("k4");
```

```
    try {
```

```
        Class.forName("com.ibm.db2.jcc.DB2Driver");
```

```
        Connection
```

```
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");
```

```
html=?");
PreparedStatement ps=con.prepareStatement("select * from lk where
```

```
String mid1=(String) hs.getAttribute("html");
```

```
//int mid=Integer.parseInt(mid1);
```

```
ps.setString(1, mid1);
```

```
ResultSet rs=ps.executeQuery();
```

```
while(rs.next()){
```

```
int q=rs.getInt(4);
```

```
if(rs.getString(2).equals(k1)||rs.getString(2).equals(k2)||rs.getString(2).equals(k3)||rs.getString(2).equals(k4)){
```

```
q++;
```

```
PreparedStatement ps2=con.prepareStatement("delete lk
where html=? and keyword=?");
```

```
ps2.setString(1, rs.getString(1));
```

```
ps2.setString(2, rs.getString(2));
```

```
ps2.executeUpdate();
```

```
PreparedStatement ps1=con.prepareStatement("insert into lk
values(?,?,?,?)");
```

```
ps1.setString(1,rs.getString(1));
```

```
ps1.setString(2,rs.getString(2));
```

```
ps1.setInt(3,rs.getInt(3));
```

```

        ps1.setInt(4, q);

        int chck=ps1.executeUpdate();

        if(chck>0){

            //response.sendRedirect("");

            pw.print("<html><head></head><frameset
rows=\"350,*\"><frame  src=\""+rs.getString(1)+"\"  name=\"top\"/><frame  src=\"like.html\"
name=\"bottom\"/></frameset><body>");

            //pw.print("<script    type='text/javascript'>function
hai(){ document.write(\"hai\");}</script></body></html>");

            }

        }

    }

    catch(Exception e){pw.print(e);}

}
}

```

### **S3.java**

```

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

```



```

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/**

 * Servlet implementation class for Servlet: S3

 *

 */

public class S3 extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {

    static final long serialVersionUID = 1L;

    /** (non-Java-doc)

        * @see javax.servlet.http.HttpServlet#HttpServlet()

        */

    public S3() {

        super();

    }

    /** (non-Java-doc)

        * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,

        HttpServletResponse response)

        */

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

    ServletException, IOException {

```

```

        // TODO Auto-generated method stub

    }

    /* (non-Java-doc)

    *      @see      javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
    HttpServletRequest response)

    */

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        // TODO Auto-generated method stub

        PrintWriter pw=response.getWriter();

        String html=request.getParameter("url");

        String name=request.getParameter("name");

        if(html.equals("")||name.equals("")){

            response.sendRedirect("rege.jsp");

        }

        else{

            HttpSession hs=request.getSession(true);

            hs.setAttribute("htmlreg", html);

            try {

                Class.forName("com.ibm.db2.jcc.DB2Driver");

                Connection
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");

                //pw.print("3");

                PreparedStatement      ps=con.prepareStatement("insert      into      name

```

```

values(?,?)");

        //pw.print("2");

        //pw.print(did1);

        ps.setString(1,html);

        ps.setString(2,name);

        int n=ps.executeUpdate();

        if(n>0){

            response.sendRedirect("idea.html");

        }

    }

    catch(Exception e){pw.print(e);}

}

}

```

#### **S4.java**

```

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;


import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

```

```
/**
```

```
 * Servlet implementation class for Servlet: S4
```

```
 *
```

```
 */
```

```
public class S4 extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {
```

```
    static final long serialVersionUID = 1L;
```

```
    /** (non-Java-doc)
```

```
        * @see javax.servlet.http.HttpServlet#HttpServlet()
```

```
    */
```

```
    public S4() {
```

```
        super();
```

```
    }
```

```
    /** (non-Java-doc)
```

```
        * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
        HttpServletResponse response)
```

```
    */
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
```

```
        // TODO Auto-generated method stub
```

```
    }
```

```
    /** (non-Java-doc)
```

```
        * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
        HttpServletResponse response)
```

```

*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    // TODO Auto-generated method stub

    PrintWriter pw=response.getWriter();

    String dm=request.getParameter("dm");

    String new1=request.getParameter("dm1");

    String se=request.getParameter("r1");


    HttpSession hs=request.getSession(true);

    String html=(String) hs.getAttribute("htmlreg");

    try {

        Class.forName("com.ibm.db2.jcc.DB2Driver");


        Connection
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");

        //pw.print("3");

        if(!new1.equals("")){

            dm=new1;

            PreparedStatement ps1=con.prepareStatement("insert into domain
values(?)");

            ps1.setString(1,dm);

            ps1.executeUpdate();

        }

        PreparedStatement ps=con.prepareStatement("insert into dm values(?,?)");

        //pw.print("2");

```

```

        //pw.print(did1);

        ps.setString(1,html);

        ps.setString(2,dm);

        int n=ps.executeUpdate();

        if(n>0){

                if(se.equals("yes")){

                        response.sendRedirect("dm.jsp");

                }

                else if(se.equals("no")){

                        response.sendRedirect("key.jsp");

                }

        }

    }

    catch(Exception e){pw.print(e);}

}
}

```

### **S5.java**

```

import java.io.IOException;

//import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

```

```

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class for Servlet: S5
 *
 */
public class S5 extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {
    static final long serialVersionUID = 1L;

    /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#HttpServlet()
        */
    public S5() {
        super();
    }

    /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
        HttpServletResponse response)
        */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        // TODO Auto-generated method stub

```

```

    }

    /* (non-Java-doc)
    *      @see      javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
    HttpServletRequest response)
    */

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        // TODO Auto-generated method stub

        //PrintWriter pw=response.getWriter();

        String key=request.getParameter("k");

        if(key.equals("")){

            response.sendRedirect("key.jsp");

        }

        else{

            String se=request.getParameter("r1");

            HttpSession hs=request.getSession(true);

            String html=(String) hs.getAttribute("htmlreg");

            try {

                Class.forName("com.ibm.db2.jcc.DB2Driver");

                Connection
con=DriverManager.getConnection("jdbc:db2://localhost:50000/main","Chaitanya","welcome");

                //pw.print("3");

                PreparedStatement      ps=con.prepareStatement("insert      into      lk

```



```

values(?,?,?,?");

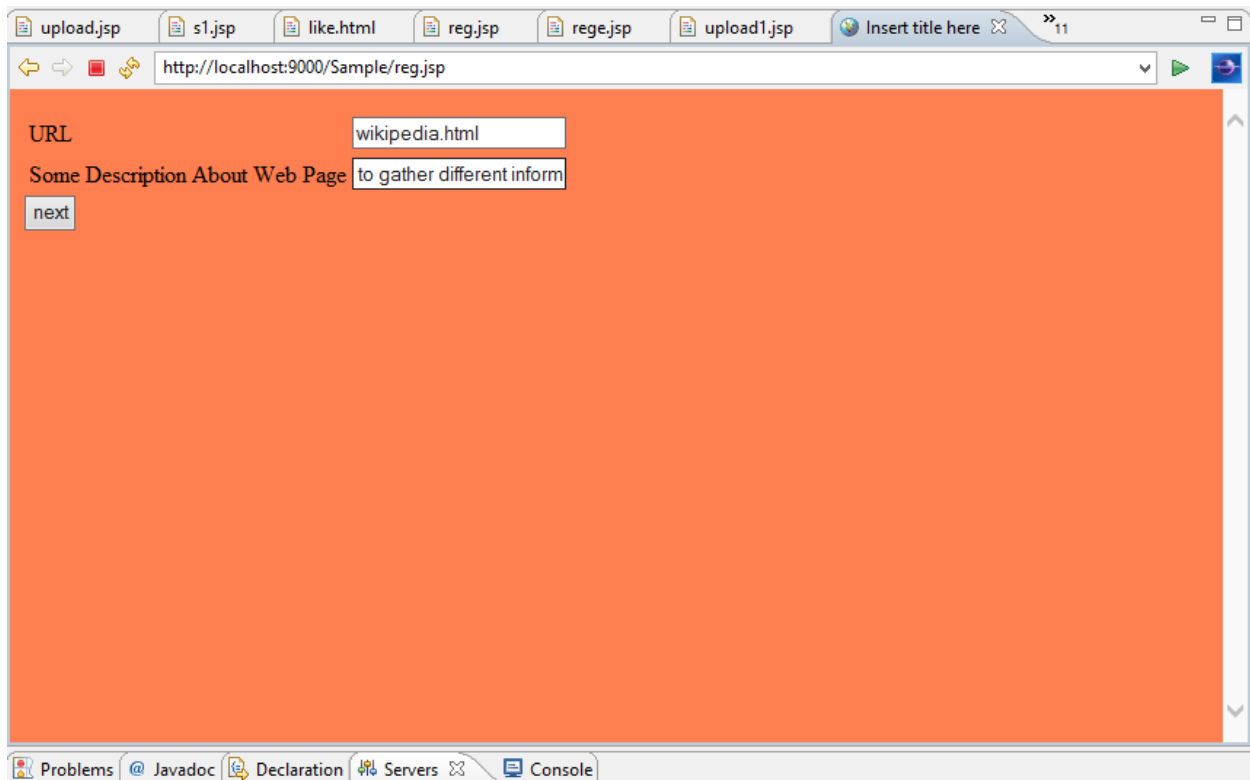
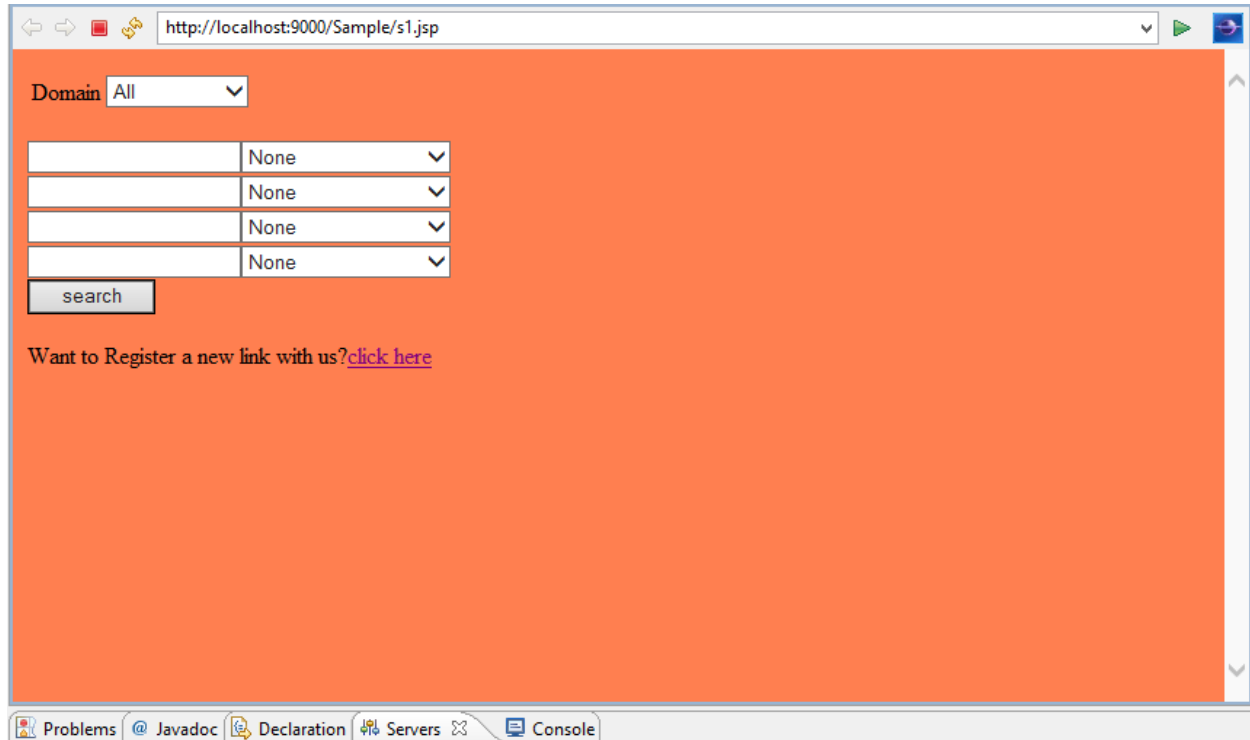
        //pw.print("2");

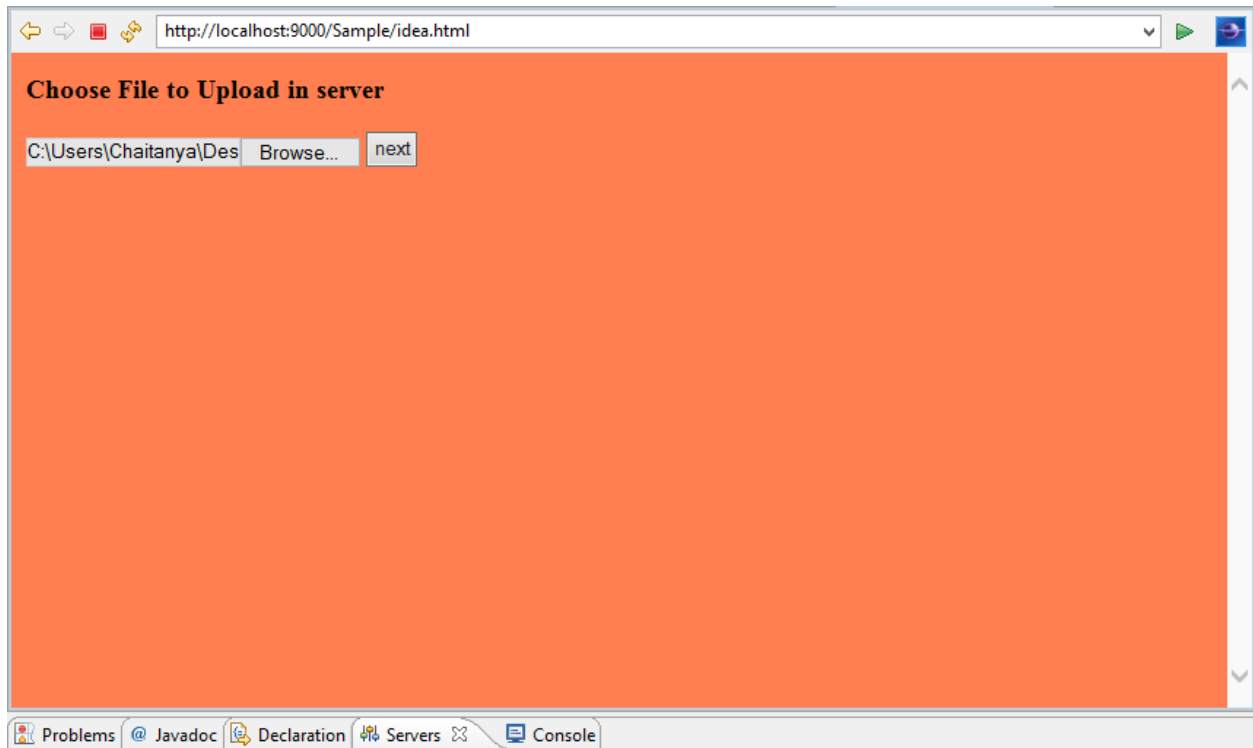
        //pw.print(did1);

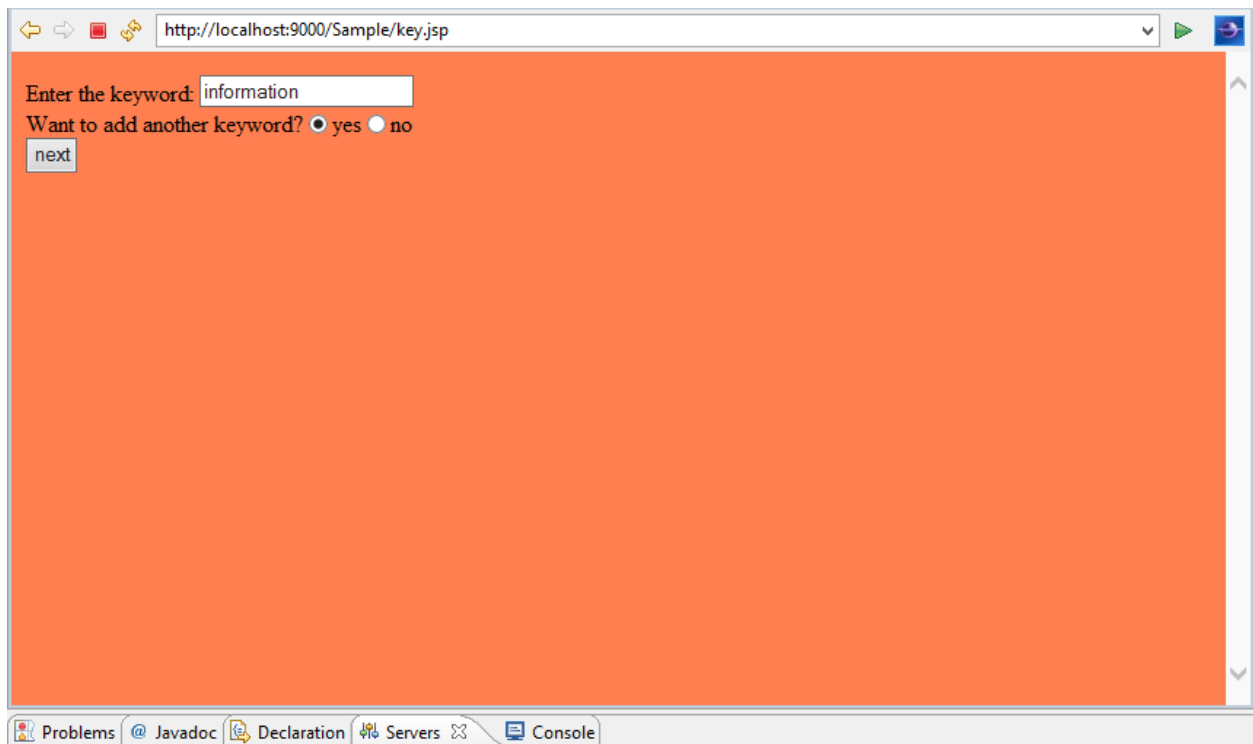
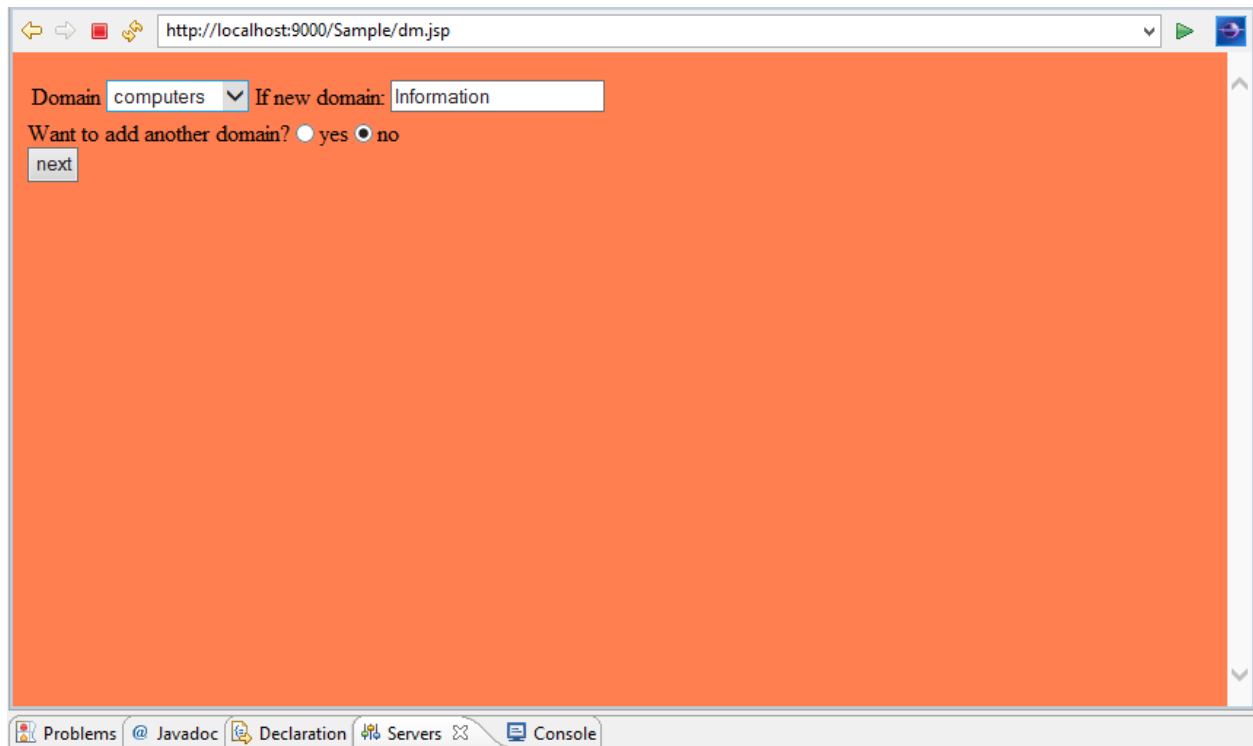
        ps.setString(1,html);
        ps.setString(2,key);
        ps.setInt(3,0);
        ps.setInt(4,0);
        int n=ps.executeUpdate();
        if(n>0){
            if(se.equals("yes")){
                response.sendRedirect("key.jsp");
            }
            else if(se.equals("no")){
                response.sendRedirect("s1.jsp");
            }
        }
    }
    catch(Exception e){ }
}
}

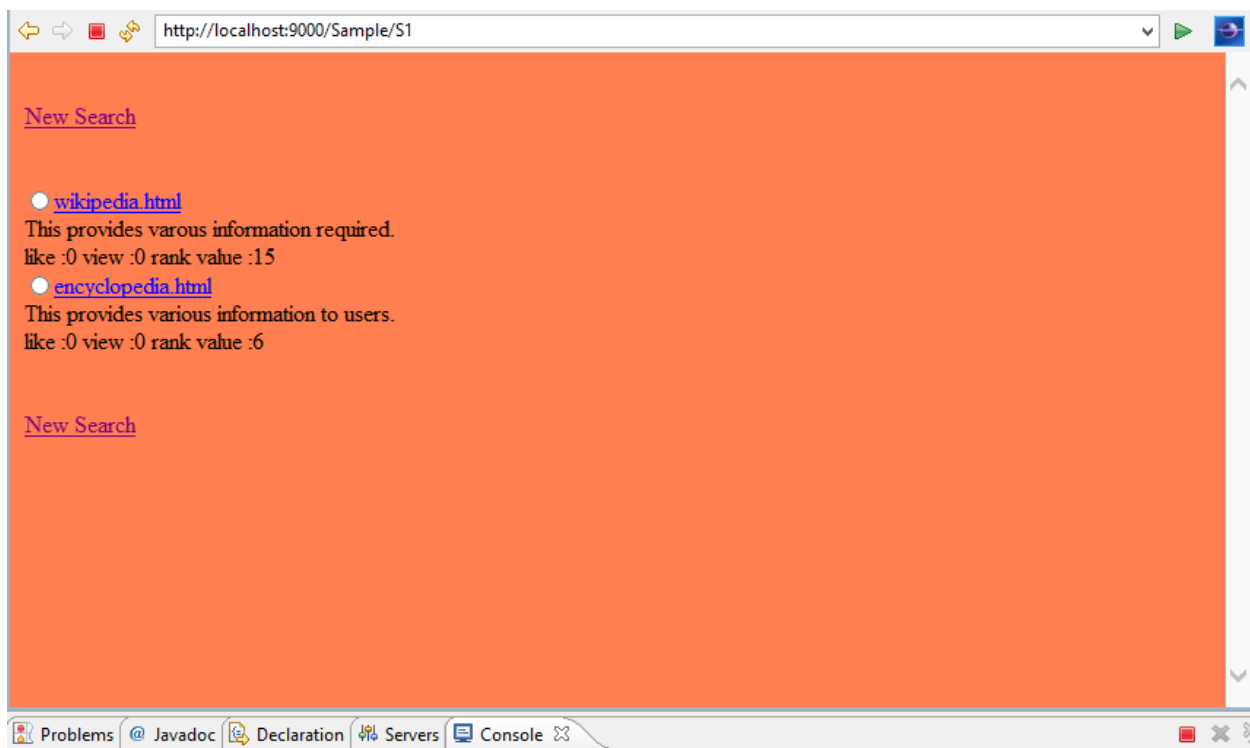
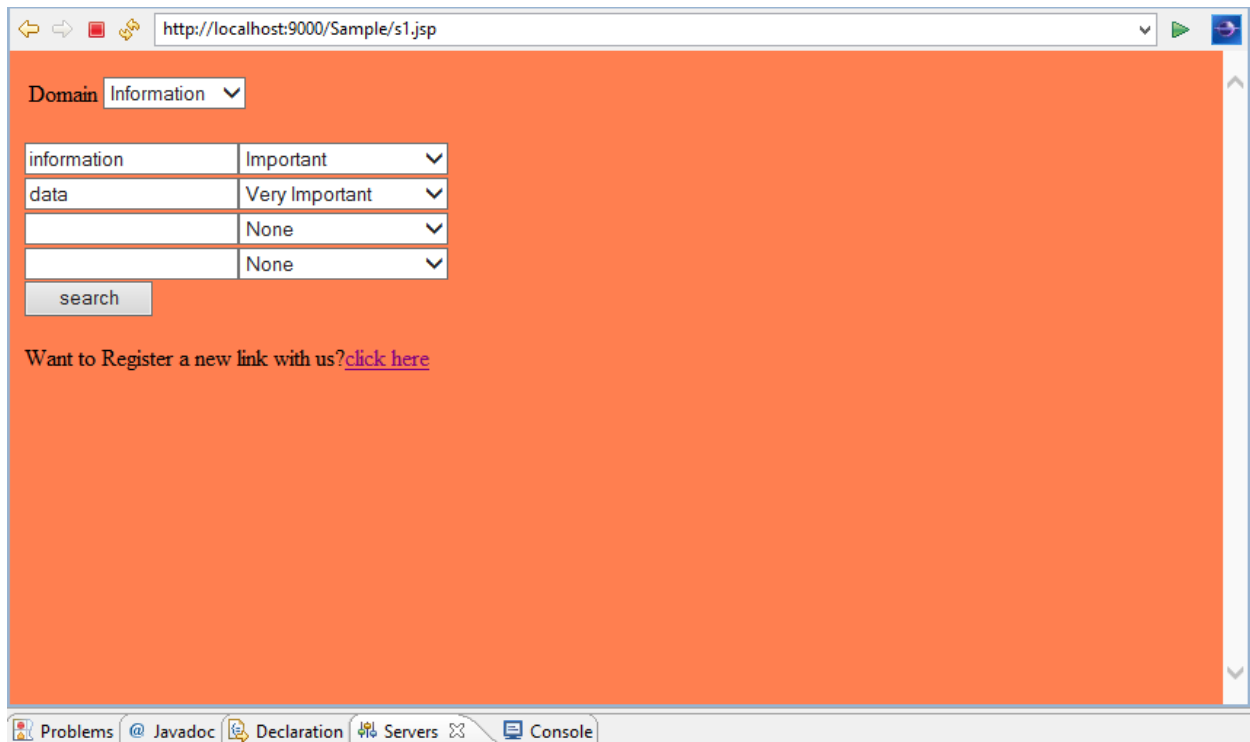
```

## 6.2 Output











## **7. TESTING**

### **7.1 Introduction**

#### **TESTING**

##### **Testing Activities:**

Testing is the process of analyzing a system or system component for finding differences between the expected behavior specified by system models and the observed behavior of the system.

It is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

##### **Psychology of Testing**

The aim of the testing is often to demonstrate that a program works by showing that it has no errors. This is the opposite of what testing should be viewed as. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence, one should not start testing with the intent of showing that a program works; but the intent should be to show that a program does not work. With this in mind we define testing as follows. Testing is the process of executing a program with the intent of finding errors.

## **OVERVIEW**

### **Reliability**

It is a measure of success with which the observed behavior of a system confirms to the specification of its behavior.

### **Software Reliability**

The probability that a software system will not cause the failure of the system for a specified time, under specified conditions.

### **Failure**

It is the deviation of the observed behavior from the existing behavior. The system is in a state such that further processing by the system will lead to a failure, which causes the system to deviate from its intended behavior.

### **Fault**

Mechanical or algorithmic causes of an error. There are three important techniques for increasing the reliability of a software system.

- Fault avoidance techniques
- Fault detection techniques
- Fault tolerance techniques

### **Fault avoidance techniques**

These detect errors statically that is without relying on the execution of any of the system models (code).



### **Fault detection techniques**

These are used during development process to identify errors and find the underlying faults before releasing the system.

### **Fault tolerance techniques**

By using this, system can be released with errors and failures those can be recovered at run time.

There are two types of fault detection techniques.

- Debugging
- Testing

### **Debugging**

The faults can be found by starting from an unplanned failure. The developer moves the system through a succession of states, ultimately arriving at and identifying the erroneous state. Once this state is identified, the algorithmic or mechanical fault causing this state needs to be determined.

There are two types of debugging

### **Correctness Debugging**

It addresses the deviation between the observed and specified functional requirements.

### **Performance Debugging**

It addresses the deviation between the observed and specified nonfunctional requirements such as response time.

## 7.2 Types of Testing

### **Testing**

Testing is a process of analyzing a system. The goal of testing is to design tests that exercise defects in the system and to reveal problems

There are three types of testing.

- Unit testing
- Integration testing
- System testing

### **Testing Strategies**

#### **White Box Testing**

This testing is also called as “Structural Testing” because it uses the internal structure of the program to derive the test data. The software has been thoroughly tested using the “White Box Testing” Strategy and it is confirmed that it is working according to the standards and the requirement specifications.

#### **Black Box Testing**

This testing is also called as “Functional Testing” because it is based on the definition of what Programming is intended to do, i.e., on the programs specified, rather than of the internal Structure of the program. The software has been tested using the “Black Box Testing” Strategy and confirmed that it is working according to the standards and the requirements specified.

## **Test plan**

### **➤ Unit Testing**

Unit testing focuses on testing individual modules within a program .There are two types of unit testing

- Static analysis tests
- Dynamic analysis tests

### **Static analysis tests**

These evaluate the quality of a module through a direct examination of a source code without execution on the machine..

### **Dynamic analysis tests**

Dynamic tests require the module to be executed on a machine.

There are two types of dynamic analysis tests.

- **Black-box test**

In black-box testing internal logic of a module is not examined. Test cases are designed based on the requirements specification of the module and are executed to find the deviations from requirements.

- **White-box test**

White-box testing reveals the internal workings of a module. Test cases are designed after the examination of internal logic of a module and traverse the different execution paths built into the system.

## ➤ **Integration Testing**

It detects faults by focusing on small groups of components. Two or more components are integrated and tested, and once tests do not reveal any new faults, additional components are added to the group

There are three types of integration testing.

### **1. Big-bang testing**

All components are tested individually and tested together as a single system.

### **2. Top-down testing**

This strategy unit tests the components of the top layer first and then integrates the components of the next layer down. When all the components of the new layer have been tested together, the next layer is selected. Again the tests incrementally add one component after the other to the test.

### **3. Bottom-up testing**

This strategy unit tests the components of the bottom layer first and then integrates the components of the next layer up. When all the components of the new layer have been tested together, the next up-layer is selected. This is repeated until all components from all layers are combined together.

## ➤ **System Testing**

It tests all the components together, seen as a single system to identify errors from the problem statement, requirement analysis and system design respectively. System testing is of three types.

### **1. Functional testing**

Tests the requirements specified as in requirement analysis document and user manual in applicable cases and are done by the developer.

### **2. Performance testing**

Tests the nonfunctional requirements specified as in requirement analysis document and are done by the developer.

### **3. Acceptance testing**

Usability, functional and performance tests that are performed by client in the development environment against acceptance criteria i.e. from project agreement.

### **➤ Installation testing**

Usability, functional and performance tests that are performed by client in the target environment.

A test case is a set of input data and expected results that exercises the component with the purpose of causing failures and detecting faults. Test cases are classified into black box test and white box test.

System testing is of three types.

### **2. Functional testing**

Tests the requirements specified as in requirement analysis document and user manual in applicable cases and are done by the developer.

### **3. Performance testing**

Tests the nonfunctional requirements specified as in requirement analysis document and are done by the developer.

### **4. Acceptance testing**

Usability, functional and performance tests that are performed by client in the development environment against acceptance criteria i.e. from project agreement.

## **TEST CASES & RESULTS**

TEST NO	TEST CASE DESCRIPTION/ACTION	INPUT	EXPECTED OUTPUT	OBSERVED OUTPUT
1.	Provide URL during registration	Enter an url that already exists	An error message "url already exists"	An error message "url already exists"
2.	Provide URL during registration	Enter a new url	Update the database	Update the database
3.	Search	Enter the fields that match to nothing	Display "no mach found"	Display "no mach found"
4.	Search	Enter the fields that match to some pages	Display the matching pages in order	Display the matching pages in order

## **8. Conclusion**

- The web search engine system allows us to search the sample HTML pages based on exact keyword match and retrieve them in the order of keyword specific like count and view count. It allows the user to specify the relative importance to their search keywords through which the problem of differentiating the importance of one keyword from that of another is solved. Thus using this we can search for the required information easily.



## 9. References

### **References:**

- 2011 IEEE International Conference on Fuzzy Systems June 27-30, 2011, Taiwan
- Keyword Density, [http://en.wikipedia.org/wiki/Keyword\\_density](http://en.wikipedia.org/wiki/Keyword_density)
- Page Rank Check, [http://www.prchecker.info/check\\_page\\_rank.php1](http://www.prchecker.info/check_page_rank.php1)
- <http://www.w3schools.com>

## 10. Bibliography

### **Bibliography**

- Software Engineering, 7th, Ian Somerville.
- Object Oriented Modeling and Design with UML-Michael Blaha, James Rumbaugh. 1<sup>st</sup> Edition.
- The complete Reference Java 5<sup>th</sup> Edition, TMH, 2007
- Hans Bergsten: Java Server Pages, 3<sup>rd</sup> Edition, o'Riley publication, 2008