1. INTRODUCTION

1.1 Objective

There is a voluminous amount of data in the present world and additionally it is being incremented day by day so to retrieve the most related information among this ample of documents is being complex. Whenever a user requests a query they are being resulted with the documents either relating to the title of the document or to the author of the document, but we propose a system in order to make the user get the result based on the data present among the available vast documents.

1.2 About the project

An application to collect the documents either the .pdf's or the .txt files, in case the files provided are the .pdf's then we would like to convert the pdf's to text files through this application as this is purely the text based, later on forming groups out of the given documents based upon the words matching with the predefined file ,which is useful in order to reduce the time for processing, then retrieve the exact and the most relevant document(s) among the clusters, depending upon the search query from the user.

1.3 Purpose

We are expecting to provide the user with the most relevant document for the given query through this application, store the huge collection of the document in the hadoop file system so that it can support in increasing the data set day by day.

2.ANALYSIS

2.1 EXISTING SYSTEM

The existing system has a facility that is when given a query; it searches only the title of the book and author of the book for the particular word given by the user. This process may not give the accurate result of what the user is expecting and we know that every user expects to retrieve the search result in less time.

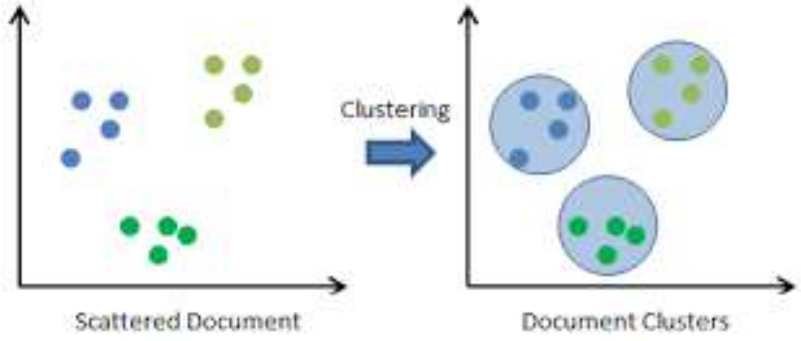
2.2 PROPOSED SYSTEM

This system enables us to maintain the pdf files later convert them to the text files using this application, which helps us in reducing the size to store the documents and return them in ranking order, later on divide them into clusters in order to reduce the time of searching of the all the ample of documents. Retrieving the information as per the user requested queries which are relevant to what the user is expecting based on the ranking is provided to the users. Result is produced in the ranking order of the score of the words present in the document making them as the fields. Scoring considers of many criteria's in order to provide with the apt result.

2.3 DATA MINING WITH THE BIG DATA

Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.[1] It is an essential process where intelligent methods are applied to extract data patterns.It is an interdisciplinary subfield of computer science The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

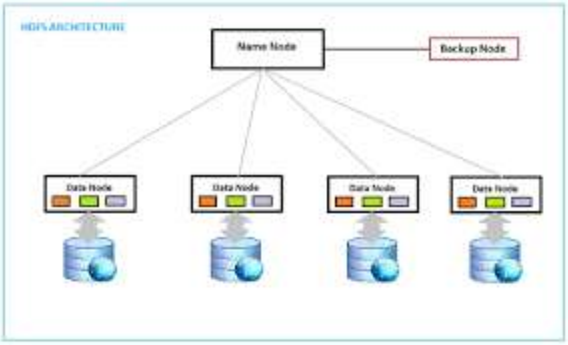
Clustering



- Clustering is the process of partitioning a set of data objects (or observations) into subsets.
- The subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters.
- The set of clusters resulting from a cluster analysis can be referred to as a clustering.
- Different clustering methods may generate different clustering on the same data set.

Big data is a term for a large data set. Big data sets are those that outgrow the simple kind of database and data handling architectures that were used in earlier times, when big data was more expensive and less feasible. For example, sets of data that are too large to be easily handled in a Microsoft Excel spreadsheet could be referred to as big data sets.

Hadoop Distributed File System



- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on the commodity hardware.
- HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

3. SYSTEM REQUIREMENT SPECIFICATION

System analysis is the first technical step in software engineering process. It is at this point that a general statement of software scope is refined into concrete specification that becomes the foundation for all software engineering activities that follow. Analysis must focus on information, functional and behavioural domain of the problem. To better understand what is required, models are created and the problem is partitioned. In many cases it is not possible to completely specify a problem at an early stage.

3.1 System Requirement Specification (SRS)

A software requirement specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.

Software Requirement Specification is the starting point of the software development activity. The Software Requirement Specification is produced at the culmination of the analysis task. The introduction of the software requirement specification states the goals and objectives of the software, describing it in the context of the computer-based system. The SRS includes an information description, functional description, behavioural description, validation criteria.

The purpose of this document is to present the software requirements in a precise and easily understood manner. This document provides the functional, performance, design and verification requirements of the software to be developed.

This is the only document that describes the requirements of the system. This is meant for use by the developers and will also be the basis for validating the final delivered system.

A requirement is a statement about what the proposed system will do that all stakeholders agree must be made true in order for the customer’s problem to be adequately solved.

Requirements can be divided in two major types, functional and non-functional.

Domain analysis is the process by which a software engineer learns background information. He or she has to learn sufficient information so as to be able to understand the problem and make good decisions during requirement analysis and other stages of software engineering process.

To perform domain analysis, you must gather information from whatever sources of information are available: these include the domain experts; and any books about the domain, any existing software and its documentation, and any other documents you can find.

3.2. Functional Requirements

The functional requirements of the system defines a function of software system or its components. A function is described a set of inputs, behaviour of a system and output.
The following are the functional requirements of proposed system

INPUT:

INSERTING MODULE:

When given a pdf file, we extract the pure text from the pdf files, and forward it to the folder depending on some standard predefined keywords, which is clustering and later the file is to be processed to generate the index file for that text file by eliminating the stop words and tokenizing the words and by performing further analysis for the given text file words. Later on the user can provide this application with the query which can be either boolean, phrase, term, or wildcard. This query is accepted and the rank wise files are displayed through this search maintaining the score and the doc id along with the documents. The documents are stored in the hadoop file system as it increases day to day.

.

PROCESS:

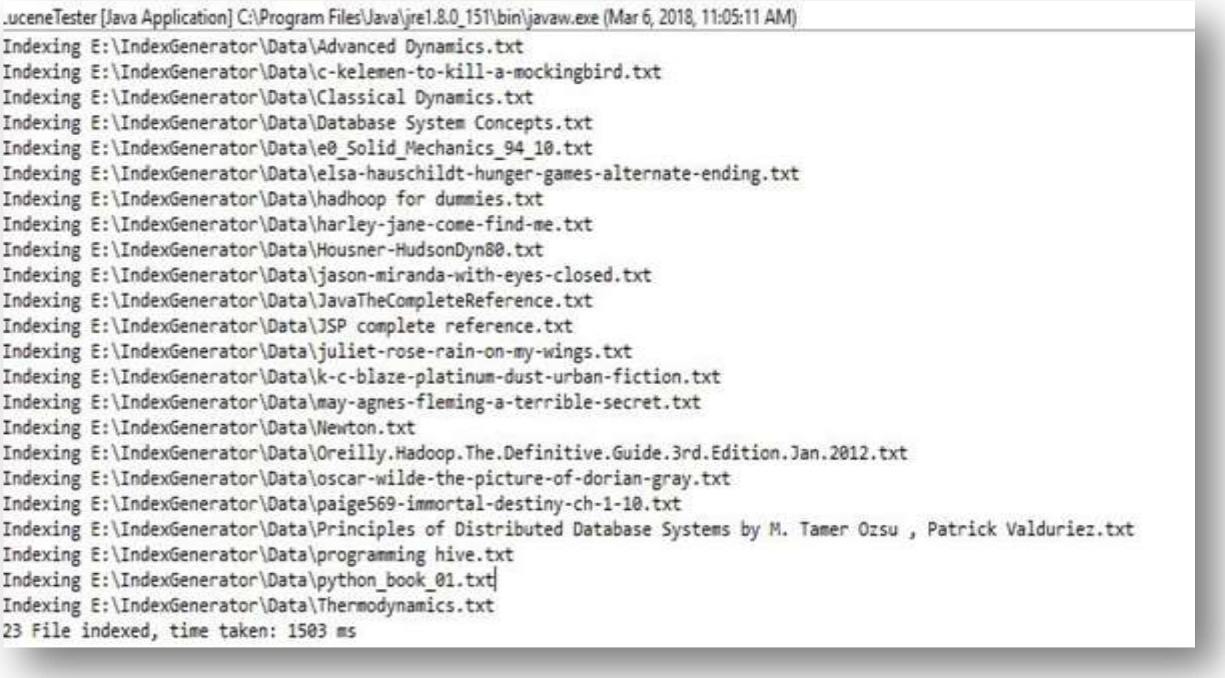
- PDF to TEXT

In the present generation the world is running around the ebooks and its completely pdf format, as we are running this application purely on text we would like to convert the pdf to text completely by eliminating the other contents other than the text.

This is benefiting us to process the query fast and easy maintaining the score of each field in the document.

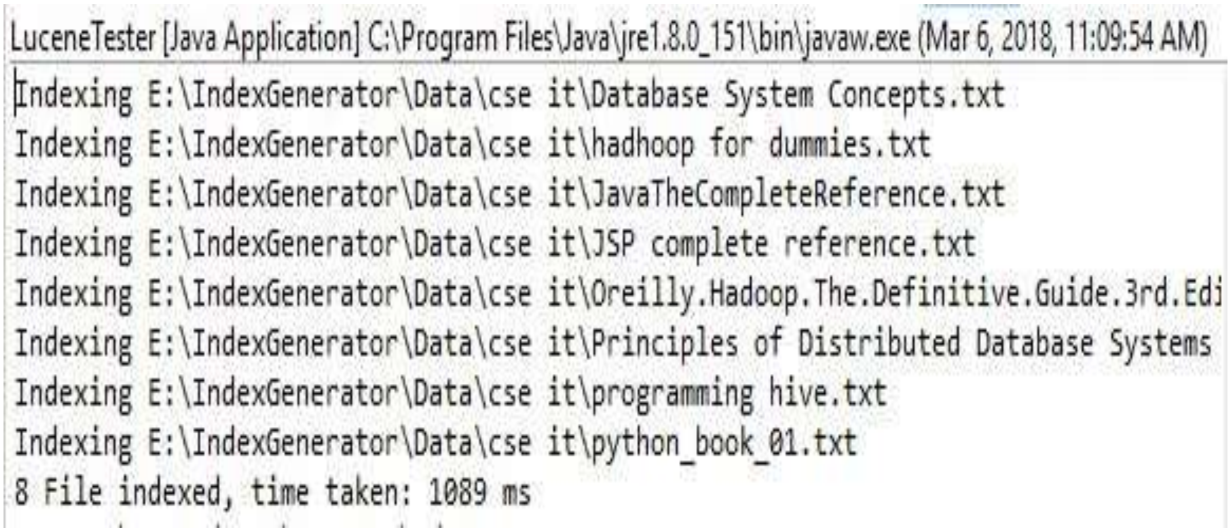
- CLUSTERING

When some text files are uploaded they are processed by splitting the contents as tokens and they are compared to the files of another and split into different folders depending on the comparisons made by the above function. This process helps us in the retrieval of the document to be efficient and reduce the processing time by forming out the clusters.



```
.uceneTester [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (Mar 6, 2018, 11:05:11 AM)
Indexing E:\IndexGenerator\Data\Advanced Dynamics.txt
Indexing E:\IndexGenerator\Data\c-kelemen-to-kill-a-mockingbird.txt
Indexing E:\IndexGenerator\Data\Classical Dynamics.txt
Indexing E:\IndexGenerator\Data\Database System Concepts.txt
Indexing E:\IndexGenerator\Data\e0_Solid_Mechanics_94_10.txt
Indexing E:\IndexGenerator\Data\elsa-hauschildt-hunger-games-alternate-ending.txt
Indexing E:\IndexGenerator\Data\hadhoop for dummies.txt
Indexing E:\IndexGenerator\Data\harley-jane-come-find-me.txt
Indexing E:\IndexGenerator\Data\Housner-HudsonDyn80.txt
Indexing E:\IndexGenerator\Data\jason-miranda-with-eyes-closed.txt
Indexing E:\IndexGenerator\Data\JavaTheCompleteReference.txt
Indexing E:\IndexGenerator\Data\JSP complete reference.txt
Indexing E:\IndexGenerator\Data\juliet-rose-rain-on-my-wings.txt
Indexing E:\IndexGenerator\Data\k-c-blaze-platinum-dust-urban-fiction.txt
Indexing E:\IndexGenerator\Data\may-agnes-fleming-a-terrible-secret.txt
Indexing E:\IndexGenerator\Data\Newton.txt
Indexing E:\IndexGenerator\Data\Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.txt
Indexing E:\IndexGenerator\Data\oscar-wilde-the-picture-of-dorian-gray.txt
Indexing E:\IndexGenerator\Data\paige569-immortal-destiny-ch-1-10.txt
Indexing E:\IndexGenerator\Data\Principles of Distributed Database Systems by M. Tamer Ozsu , Patrick Valduriez.txt
Indexing E:\IndexGenerator\Data\programming hive.txt
Indexing E:\IndexGenerator\Data\python_book_01.txt
Indexing E:\IndexGenerator\Data\Thermodynamics.txt
23 File indexed, time taken: 1503 ms
```

So the time difference is observed clearly for the application with clustering and application without clustering.

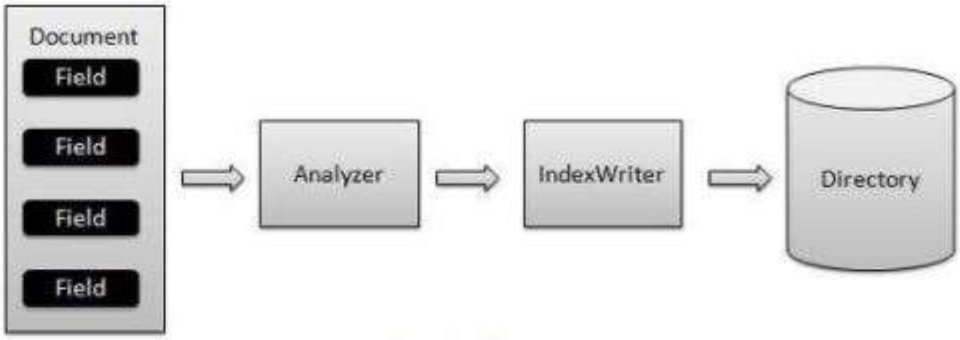


```
LuceneTester [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (Mar 6, 2018, 11:09:54 AM)
Indexing E:\IndexGenerator\Data\cse it\Database System Concepts.txt
Indexing E:\IndexGenerator\Data\cse it\hadhoop for dummies.txt
Indexing E:\IndexGenerator\Data\cse it\JavaTheCompleteReference.txt
Indexing E:\IndexGenerator\Data\cse it\JSP complete reference.txt
Indexing E:\IndexGenerator\Data\cse it\Oreilly.Hadoop.The.Definitive.Guide.3rd.Edi
Indexing E:\IndexGenerator\Data\cse it\Principles of Distributed Database Systems
Indexing E:\IndexGenerator\Data\cse it\programming hive.txt
Indexing E:\IndexGenerator\Data\cse it\python_book_01.txt
8 File indexed, time taken: 1089 ms
```

- INDEXING

This application is able to achieve fast search responses since, time taken for searching the entire text directly among the ample of documents is more, whereas this application searches an index instead. This would be the equivalent of retrieving pages in a book related to a keyword by searching the index at the back of a book i.e,the appendix ,as opposed to searching the words in each page of the book. This type of index is called an **inverted index**, because it inverts a page-centric data structure ta keyword-centric data structure.

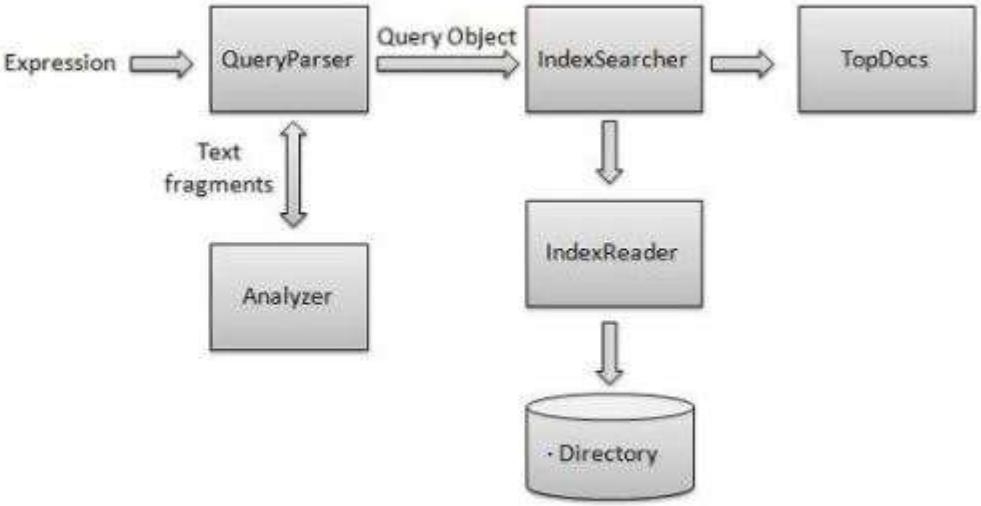
As soon as the text file is been uploaded the index file is been generated for the words present in the document and later on based on the index file been generated we would consider that file and locate the documents being related and rank them in order. The index file keeps on appending as and when the index is generated for the uploaded text file. Index file consists of the field and its properties like time to find the word in the document, frequency count in one document, relevancy of word among the voluminous documents and so on. This step helps us to provide us with the efficient documents for the searching and ranking.



Indexing Process

- SEARCHING

Searching requires an index to be verified for the documents in order to provide the rank wise order for the given query. It involves creating a Query given by the user and handing this Query to an IndexSearcher, which returns a list of Hits.



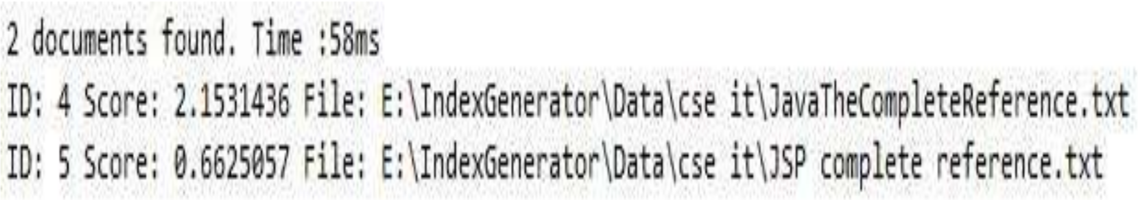
Searching Process

OUTPUT:

RANKING

Ranking of query results is one of the fundamental problems in information retrieval (IR).Given a query *q* and a collection *D* of documents that match the query, the problem is to rank, that is, sort, the documents in *D* according to some criterion so that the "best" results appear early in the result list displayed to the user. Classically, ranking criteria are phrased in terms of relevance of documents with respect to an information need expressed in the query.

Ranking based upon the relavancy by calculating the score and displaying the document path along with the document id.



```
2 documents found. Time :58ms
ID: 4 Score: 2.1531436 File: E:\IndexGenerator\Data\cse it\JavaTheCompleteReference.txt
ID: 5 Score: 0.6625057 File: E:\IndexGenerator\Data\cse it\JSP complete reference.txt
```

3.3. Non-Functional Requirements

Non-functional requirements are constraints that must be adhered to during development. They limit what resources can be used and set bounds on aspects of the software's quality.

One of the most important things about non-functional requirements is to make them verifiable. The verification is normally done by measuring various aspects of the system and seeing if the measurements confirms to the requirements. Non-Functional Requirements are divided into several groups. The first group of categories reflects the five qualities attributes

- Usability
- Efficiency

- Reliability
- Maintainability
- Reusability

The second group of non-functional requirements categories constraints the environment and technology of the system

1. Platform: It is quite important to make it clear on what hardware and operating system of the software must work on. Normally such requirements specify the least powerful platforms and declare that it must work on anything more recent or more powerful.

Hardware Requirements:

- 4GB RAM
- 1.7-2.4 GHz processor speed
- 500 GB HDD

2. Technology to be used: While it is wise to give the designers as much flexibility as possible to choose how to implement the system, sometimes constraints must be imposed. Requirements are normally stated to ensure that all systems in an organization use the same technology – this reduces the need to train people in different technologies.

Software Requirements:

- Ubuntu 16.04
- Java 1.8
- Hadoop 2.5.2

The third group of non-functional requirements categories constraint the project plan and development methods

- **Development process (methodology) to be used**: In order to ensure quality, some requirements documents specify that certain processes be followed; for example, particular approaches to testing. The details of the process should not be included in the requirements; instead a reference should be made to other documents that describe the process.

- **Cost and delivery date:** One of the biggest challenges in software engineering is accurately forecasting how much time and effort it will take either to develop a system or to make a specific set of changes. All software developers have to participate in cost estimation.

3.4 REQUIREMENTS

HADOOP

Apache Hadoop is an open-source software framework used for distributed storage and processing of datasets of big data using the Map-Reduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework. The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a Map-Reduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to.

JAVA

Java is a network-friendly programming language invented by Sun Microsystems. Java is a complete software ecosystem that represents different values to different types of users. Java technology is the first software technology that simply works without a struggle. Users are delighted to see applications run reliably and compatibly on such an incredible array of network products - from PCs, game players, and mobile phones to home appliances and automotive electronics. Java technology is an object-oriented, platform-independent, multithreaded programming environment. It is the foundation for Web and networked services, applications, platform independent desktops, robotics, and other embedded devices.

3.4.1 INSTALLATION PROCEDURES

1. Java

- Open terminal (Ctrl+Alt+T) and run the command:

```
sudo add-apt-repository ppa:webupd8team/java
```

Type in your password when it asks and hit Enter.

- Update and install the installer script:

  Run commands to update system package index and install Java installer script:

  ```
  sudo apt update; sudo apt install oracle-java8-installer
  ```

  You may replace oracle-java8-installer with oracle-java9-installer to install Java 9.

  While the install process, you have to accept Java license to continue downloading & installing Java binaries.

- Check the Java version

  To check the Java version after installing the package, run command:

  ```
  javac -version
  ```

- Set Java environment variables

  The PPA also contains a package to automatically set Java environment variables, just run command:

  ```
  sudo apt install oracle-java8-set-default
  ```

2. HDFS

- $ sudo apt-get update
- $ sudo apt-get install default-jdk
- $ java –version
- $ sudo apt-get install ssh
- $ sudo apt-get install rsync
- $ ssh-keygen -t dsa -P ' ' -f ~/.ssh/id_dsa
- $ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

  $ wget -c http://mirror.olnevhost.net/pub/apache/hadoop/common/current/hadoop-2.6.0.tar.gz
- $ sudo tar -zxvf hadoop-2.6.0.tar.gz
- $ sudo mv hadoop-2.6.0 /usr/local/Hadoop
- $ update-alternatives --config java

- $ sudo gedit ~/.bashrc

```
#Hadoop Variables
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

- $ source ~/.bashrc
- $ cd /usr/local/hadoop/etc/hadoop
- $ sudo gedit hadoop-env.sh

```
#The java implementation to use.
export JAVA_HOME="/usr/lib/jvm/java-7-openjdk-amd64"
```

- $ sudo gedit core-site.xml

```
<configuration>
      <property>
          <name>fs.defaultFS</name>
          <value>hdfs://localhost:9000</value>
      </property>
</configuration>
```

- $ sudo gedit yarn-site.xml

```
<configuration>
      <property>
```

```
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    <property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value> org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
```

- $ sudo cp mapred.site.xml.template mapred-site.xml
- $ sudo gedit mapred-site.xml

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

- $ sudo gedit hdfs-site.xml

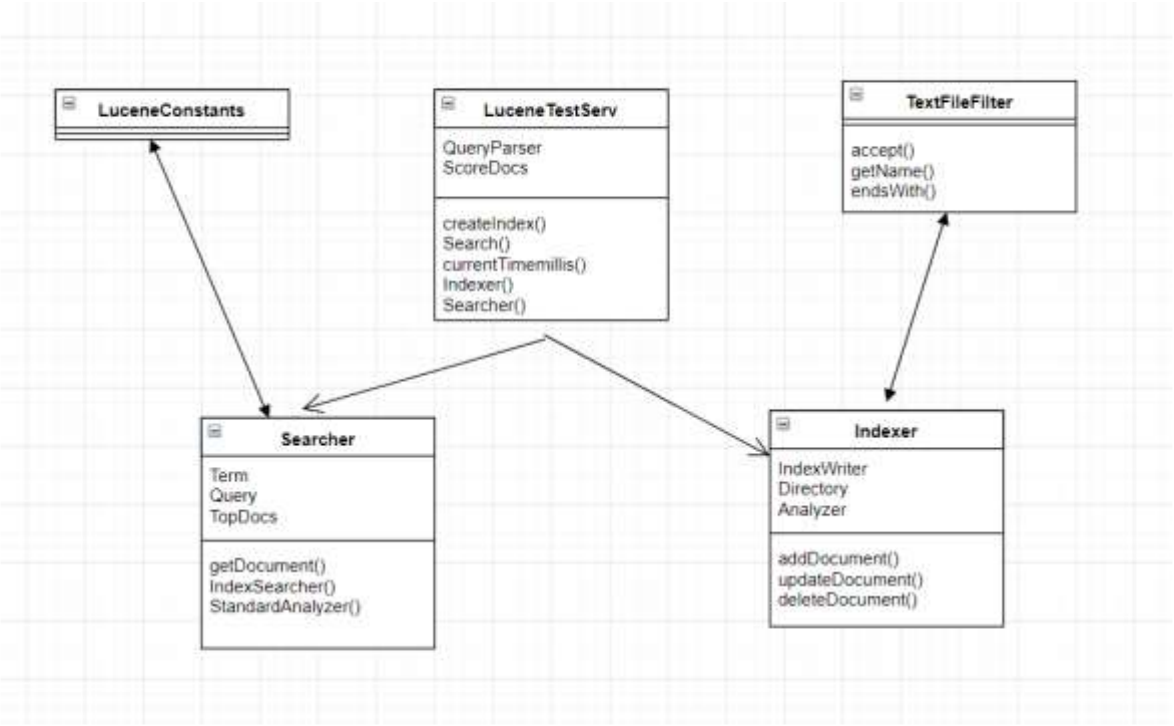
```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/hadoop/hadoop_data/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/hadoop/hadoop_store/hdfs/datanode</value>
    </property>
</configuration>
```

```
$ cd
$ mkdir -p /usr/local/hadoop/hadoop_data/hdfs/namenode
$ mkdir -p /usr/local/hadoop/hadoop_data/hdfs/datanode
$ sudo chown chaalpritam:chaalpritam -R /usr/local/Hadoop
$ hdfs namenode –format
$ start-all.sh
$ jps
```

3.5 METHODOLOGY

3.5.1 LUCENE

Lucene is an open source Java based search library. It is very popular and a fast search library. It is used in Java based applications to add document search capability to any kind of application in a very simple and efficient way. This tutorial will give you a great understanding on Lucene concepts and help you understand the complexity of search requirements in enterprise level applications and need of Lucene search engine.



Setup Lucene Framework Libraries

If the startup of java is successful, then you can proceed to set up your Lucene framework. Following are the simple steps to download and install the framework on your machine.

https://archive.apache.org/dist/lucene/java/3.6.2/

- Make a choice whether you want to install Lucene on Windows, or Unix and then proceed to the next step to download the .zip file for windows and .tz file for Unix.
- Download the suitable version of Lucene framework binaries from https://archive.apache.org/dist/lucene/java/.
- At the time of writing this tutorial, I downloaded lucene-3.6.2.zip on my Windows machine and when you unzip the downloaded file it will give you the directory structure inside C:\lucene-3.6.2

Lucene libraries in the directory **C:\lucene-3.6.2**. Make sure you set your CLASSPATH variable on this directory properly otherwise, you will face problem while running your application. If you are using Eclipse, then it is not required to set CLASSPATH.

$$\mathrm{score}(q,d) = \mathrm{coord}(q,d) \cdot \mathrm{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\mathrm{tf}(t \text{ in } d) \cdot \mathrm{idf}(t)^2 \cdot t.\mathrm{getBoost}() \cdot \mathrm{norm}(t,d) \right)$$

The factors involved in Lucene's scoring algorithm are as follows:

1. **tf** = term frequency in document = measure of how often a term appears in the document
2. **idf** = inverse document frequency = measure of how often the term appears across the index
3. **coord** = number of terms in the query that were found in the document
4. **lengthNorm** = measure of the importance of a term according to the total number of terms in the field
5. **queryNorm** = normalization factor so that queries can be compared
6. **boost (index)** = boost of the field at index-time
7. **boost (query)** = boost of the field at query-time

3.5.2 APACHE iTEXT

Apache iText is an open-source Java library that supports the development and conversion of PDF documents. In this tutorial, we will learn how to use iText to develop Java programs that can create, convert, and manipulate PDF documents.

The Portable Document Format (PDF) is a file format that helps to present data in a manner that is independent of application software, hardware, and operating systems. Each PDF file holds description of a fixed-layout flat document, including text, fonts, graphics, and other information needed to display it.

3.6 Feasibility Study

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

3.6.1Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

1. Analyzes the technical skills and capabilities of the software development team members
2. Determines whether the relevant technology is stable and established
3. Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

3.6.2 Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority

- Determines whether the solution suggested by the software development team is acceptable
- Analyzes whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

3.6.3 Economic Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization
- Cost required conducting full software investigation (such as requirements elicitation and requirements analysis)
- Cost of hardware, software, development team, and training.

4.DESIGN

4.1 PROCESS

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing software. Initially, the blue prints depicts a holistic view of software. That is, the design is represented as a high level of abstraction. As design iteration occur, subsequent refinement leads to design representations at much lower levels of abstractions. These can still be traced to requirements, but connection is more subtle.

Throughout the design process, the quality of the evolving design is assessed with a series of formal technical reviews or design walkthroughs. Three characteristics that serve as a guide for evaluation of good design:

The design must implement all of the explicit requirements contained in the analysis model. Design must be readable, understandable guide for those who generate code for those who test and subsequently support the software. Design should provide a complete picture of the software, addressing the data, functional and behavioural domains from an implementation perspective.

4.2 IMPORTANCE OF UML IN SOFTWARE DEVELOPMENT:

The Unified Modelling Language (UML) provides a standard format via construction of a model and using object oriented paradigm for describing software systems as well as non-software systems, business processes for the enterprise's problem areas and corporate infrastructure.

The model abstracts the essentials details of the underlying problems and provides a simplified view of the problem so as to make easy for the solution architect to work towards building the solution.

In context of the software development, the importance of UML can be comprehended using analogy of a construction process. Normally builders use the designs and maps to construct buildings.

The services of a civil architect are needed to create designs, maps which act as reference point for the builder. The communication between architect and builder becomes critical according to the degree of complexity in the design of the building. Blueprints or Architectural designs are the standard graphical language that both architects and builders must understand for an effective communication.

Software development is a similar process in many ways. UML has emerged as the software blueprint methodology for the business and system analysts, designers, programmers and everyone involved in creating and deploying the software systems in an enterprise.

The UML provides for everyone involved in software development process a common vocabulary to communicate about software design.

4.3. UML DIAGRAMS

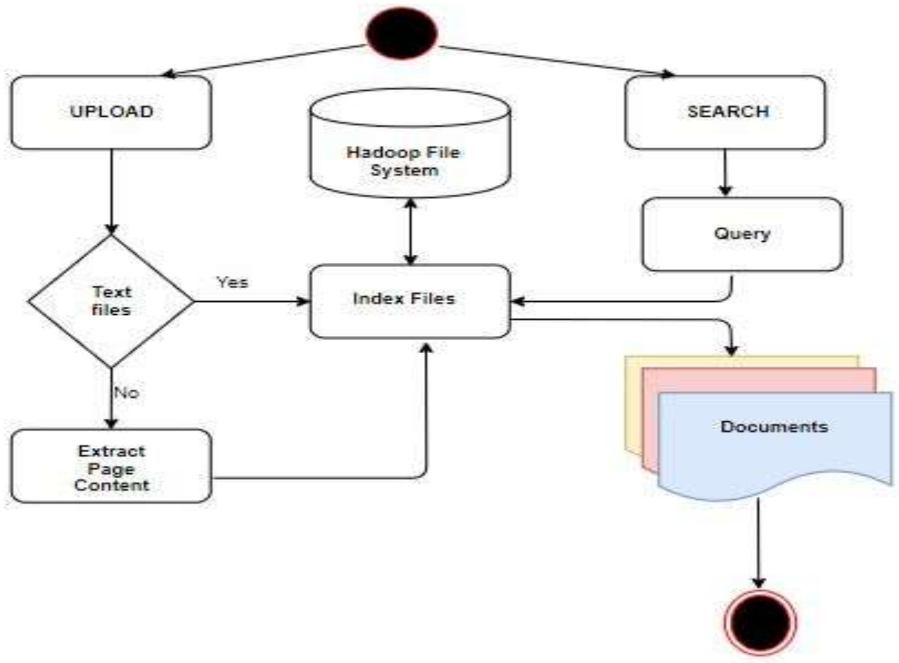
The Unified Modelling Language (UML) is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. It allows software to be visualised in multiple dimension, so that a computer system can be completely understood before construction begins.

Conceptual model of UML can be mastered by learning the following three major elements:

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

4.3.1 ACTIVITY DIAGRAM

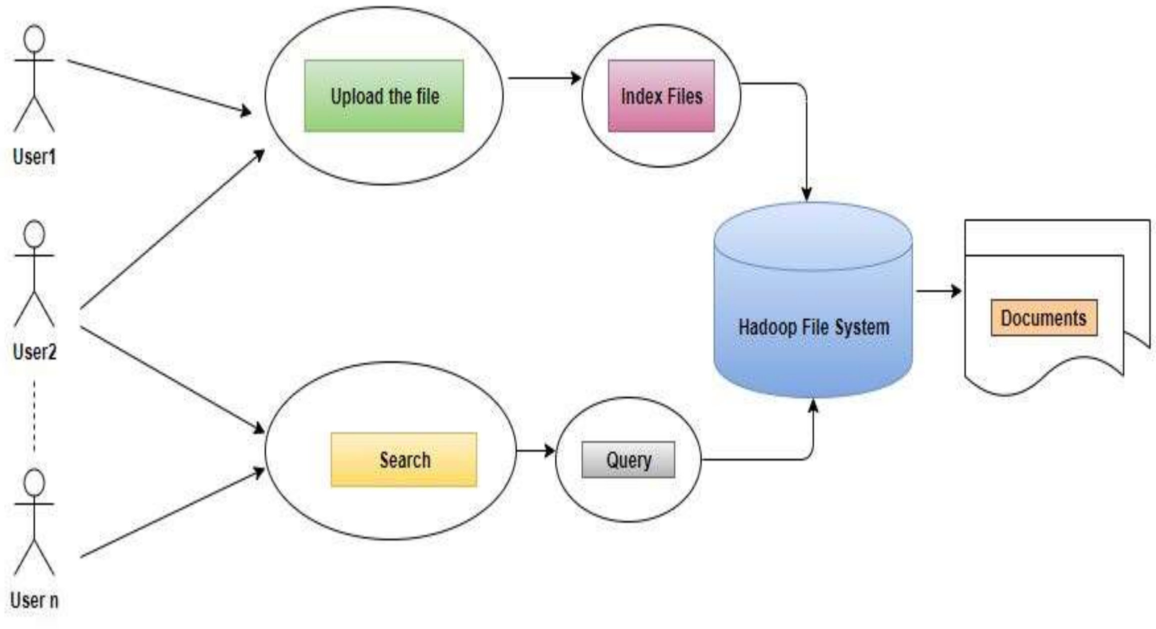
Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system.



4.3.2 USECASE DIAGRAM

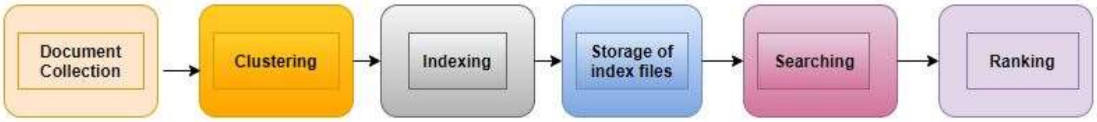
To model a system the most important aspect is to capture the dynamic behavior.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.



4.3.3 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system, modelling its process aspects. It is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. It can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored.



5. DEVELOPMENT

5.1 SAMPLE CODE

Indexer.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.File;
import java.io.FileFilter;
import java.io.FileReader;
import java.io.IOException;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

/**
 *
 * @author akshaya
 */
public class Indexer {
        private IndexWriter writer;

   public Indexer(String indexDirectoryPath) throws IOException {
```

```
   //this directory will contain the indexes
   Directory indexDirectory =
      FSDirectory.open(new File(indexDirectoryPath));

   //create the indexer
   writer = new IndexWriter(indexDirectory,
      new StandardAnalyzer(Version.LUCENE_36),true,
      IndexWriter.MaxFieldLength.UNLIMITED);
}

public void close() throws CorruptIndexException, IOException {
   writer.close();
}

private Document getDocument(File file) throws IOException {
   Document document = new Document();

   //index file contents
   Field contentField = new Field(LuceneConstants.CONTENTS, new FileReader(file));
   //index file name
   Field fileNameField = new Field(LuceneConstants.FILE_NAME,
      file.getName(),Field.Store.YES,Field.Index.NOT_ANALYZED);
   //index file path
   Field filePathField = new Field(LuceneConstants.FILE_PATH,
      file.getCanonicalPath(),Field.Store.YES,Field.Index.NOT_ANALYZED);

   document.add(contentField);
   document.add(fileNameField);
   document.add(filePathField);

   return document;
```

```
    }

    private void indexFile(File file) throws IOException {
      System.out.println("Indexing "+file.getCanonicalPath());
      Document document = getDocument(file);
      writer.addDocument(document);
    }

    public int createIndex(String dataDirPath, FileFilter filter)
      throws IOException {
      //get all files in the data directory
      File[] files = new File(dataDirPath).listFiles();

      for (File file : files) {
        if(!file.isDirectory()
          && !file.isHidden()
          && file.exists()
          && file.canRead()
          && filter.accept(file)
        ){
          indexFile(file);
        }
      }
      return writer.numDocs();
    }

}
```

Searcher.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.File;
import java.io.IOException;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.Sort;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;
```

```
public class Searcher {

IndexSearcher indexSearcher;
  QueryParser queryParser;
  Query query;

  public Searcher(String indexDirectoryPath) throws IOException {
    Directory indexDirectory
      = FSDirectory.open(new File(indexDirectoryPath));
    indexSearcher = new IndexSearcher(indexDirectory);
    queryParser = new QueryParser(Version.LUCENE_36,
      LuceneConstants.CONTENTS,
      new StandardAnalyzer(Version.LUCENE_36));
  }

  public TopDocs search( String searchQuery)
    throws IOException, ParseException {
    query = queryParser.parse(searchQuery);
    return indexSearcher.search(query, LuceneConstants.MAX_SEARCH);
  }

  public TopDocs search(Query query)
    throws IOException, ParseException {
    return indexSearcher.search(query, LuceneConstants.MAX_SEARCH);
```

```
   }

   public TopDocs search(Query query,Sort sort)
      throws IOException, ParseException {
      return indexSearcher.search(query,
         LuceneConstants.MAX_SEARCH,sort);
   }

   public void setDefaultFieldSortScoring(boolean doTrackScores,
      boolean doMaxScores) {
      indexSearcher.setDefaultFieldSortScoring(
         doTrackScores,doMaxScores);
   }

   public Document getDocument(ScoreDoc scoreDoc)
      throws CorruptIndexException, IOException {
      return indexSearcher.doc(scoreDoc.doc);
   }

   public void close() throws IOException {
      indexSearcher.close();
   }
}
```

LuceneConstants.java

```
public class LuceneConstants {

    public static final String CONTENTS = "contents";

   public static final String FILE_NAME = "filename";

   public static final String FILE_PATH = "filepath";

   public static final int MAX_SEARCH = 10;

}
```

LuceneTester.java

```
/*

* To change this license header, choose License Headers in Project Properties.

* To change this template file, choose Tools | Templates

* and open the template in the editor.

*/


import java.io.IOException;


import org.apache.lucene.document.Document;

import org.apache.lucene.index.Term;

import org.apache.lucene.queryParser.ParseException;

import org.apache.lucene.search.FuzzyQuery;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.ScoreDoc;

import org.apache.lucene.search.Sort;

import org.apache.lucene.search.TopDocs;
```

```
public class LuceneTester {


String indexDir = "/home/sampath/eeswar/Desktop/IndexGenerator/Index";

String dataDir = null;

Indexer indexer;

Searcher searcher;


public static void main(String[] args) {

  LuceneTester tester;

  try {

      tester = new LuceneTester();

       tester.createIndex();

        //String st[]=tester.search();


  } catch (IOException e) {

      e.printStackTrace();

  }

}

public void createIndex() throws IOException {

  indexer = new Indexer(indexDir);

  int numIndexed;

  long startTime = System.currentTimeMillis();

  numIndexed = indexer.createIndex(dataDir, new TextFileFilter());

  long endTime = System.currentTimeMillis();
  indexer.close();
  System.out.println(numIndexed+" File indexed, time taken: "
    +(endTime-startTime)+" ms");
}
public String[] search(String searchQuery) throws IOException, ParseException {
      searcher = new Searcher(indexDir);
          String st[]=new String[1000];
          int i=0;
      long startTime = System.currentTimeMillis();
      TopDocs hits = searcher.search(searchQuery);
      long endTime = System.currentTimeMillis();
      System.out.println(hits.totalHits +
        " documents found. Time :" + (endTime - startTime));
      for(ScoreDoc scoreDoc : hits.scoreDocs) {
        Document doc = searcher.getDocument(scoreDoc);
          System.out.println("score=    "+scoreDoc.score+"        id:"+scoreDoc.doc+"       File: "
          + doc.get(LuceneConstants.FILE_PATH));
          st[i]="score=           "+scoreDoc.score+"                              id:          "+scoreDoc.doc+"
File: "
          + doc.get(LuceneConstants.FILE_PATH);
          i++;
      }

      searcher.close();
          return st;
```

```
   }


}
```

LuceneTesterServ.java

```
import java.io.*;

import java.util.Scanner;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


import org.apache.lucene.queryParser.ParseException;


/**

 * Servlet implementation class LuceneTesterServ

 */

@WebServlet("/LuceneTesterServ")

public class LuceneTesterServ extends HttpServlet {

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

```
    /**
     * @see HttpServlet#HttpServlet()
     */
    public LuceneTesterServ() {
        super();
        // TODO Auto-generated constructor stub
    }


            /**
             * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
             */
            protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                        // TODO Auto-generated method stub
                        //response.getWriter().append("Served at: ").append(request.getContextPath());
                        PrintWriter out =response.getWriter();
                        out.println("<html>");
        out.println("<head>");
        out.println("<title>Inserting the file</title>");
        out.println("</head>");
                                                out.print("<body style='background-color:#FFFFFF' > ");
//out.print("<a href="index.html"><font color="white">HOME</font></a>");
                        LuceneTester tester;
                        tester = new LuceneTester();
                        tester.dataDir="/home/sampath/eeswar/Desktop/IndexGenerator/Data";

                        tester.indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";



                        String varData=request.getParameter("dept");

                        switch(varData)
                        {
                          case "IT & CSE":
                                    tester.dataDir                                      =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it";
                              break;
                        case "MECH":
                                    tester.dataDir                                      =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/mech";
                              break;
                        case "NOVELS":
                                    tester.dataDir                                      =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/novel";
                              break;
                                    default :
                              break;
                        }
```

```
        String word =request.getParameter("query");
        out.println("Key word searched: "+word);
                out.print("<br>");
out.println("The documents belongs to: "+varData);


        out.println();




        try {


                tester.createIndex();
                String st[]=tester.search(word);
                for(int k=0;st[k]!=null;k++)
                {
         out.print("<table align=center border=1 width=100%>");


                 out.print("<tr>");
                 out.print("<td>");


                        out.print("rank=            ");
                        out.print(k+1+"     "+st[k]);
```

```
                              out.print("</tr>");
                              out.print("</td>");
out.print("</table>");
                             }

                                  } catch (IOException e) {
                             e.printStackTrace();
                                  }
                             catch(ParseException e){
                             e.printStackTrace();
                                  }


                              HttpSession session=request.getSession(false);
                    String n=(String)session.getAttribute("query");
                    out.print(n);

      out.close();
                                }
         protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                   // TODO Auto-generated method stub
                   doGet(request, response);
         }
```

```
}
```

FileCopyDemo.java

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import org.apache.commons.io.FileUtils;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import org.apache.lucene.queryParser.ParseException;
import java.util.Scanner;
import java.util.*;
import java.util.stream.*;

 public class FileCopyDemo extends HttpServlet {
private static final long serialVersionUID = 1L;
public static ArrayList<String> it_cse=new ArrayList<String>();
```

```
            public static ArrayList<String> mec=new ArrayList<String>();

            public static ArrayList<String> novels=new ArrayList<String>();

            public static ArrayList<String> inut=new ArrayList<String>();


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

 {

                    PrintWriter out =response.getWriter();

                    out.println("<html>");

                    out.print("<body style='background-color:#FFFFFF' > ");

            //out.println("hi    bbk");

    String tofile=null;

            String var=request.getParameter("myFile");

            var="/home/sampath/eeswar/Desktop/IndexGenerator/pdfs/"+var;

            //out.println(var);

           /* out.print(""

                                    + ""

                                    + ""

                                    + ""

                                    + ""

                                    + ""

                                    + ""

                                    + ""

                                    + "");*/

    String indept=request.getParameter("indept");
```

```
            int l=var.lastIndexOf('.')+1;

            if(var.toLowerCase().endsWith(".pdf"))

            {                   ExtractPageContent exp=new ExtractPageContent();

                    //out.println("before"+var);

                     /** The original PDF that will be parsed. */

                     String ch="\\";

                      String var3=var.replace(ch,"//");

                     //out.println("after"+var);

               String pdffile= var3;

                    String PREFACE=pdffile;


                    String var1=var3.replace(".pdf", ".txt");

                    //out.println("textfile"+var1);

                    /*out.println(""

                                        + ""

                                        + ""

                                        + ""

                                        + ""

                                        + ""

                                        + ""

                                        + ""

                                        + "");*/

    //var1="E://shilpa//programming hive.txt";

                    /** The resulting text file. */
```

```
                String txtfile=var1;

            String RESULT =txtfile;


                exp.parsePdf(PREFACE, RESULT);




        }

         var=var.replace(".pdf", ".txt");

         File from = new File(var);

         intialise(var);

                for(String s : it_cse)

                {//System.out.println(s);

                        int t=1;

                        for(String in:inut)

                        if(in.equals(s))

                        {       //PrintWriter out =response.getWriter();

                String ch="/";

                                System.out.print("it"+"  "+s+"   "+"in"+"\n");

                                        File                    to=                   new
File("/home/sampath/eeswar/Desktop/IndexGenerator/Data/cse
it/"+var.substring(var.lastIndexOf(ch)+1));

                                //out.print(var.substring(var.lastIndexOf('/')+1));

                                copy(from,to);

                                System.out.println("in it copied");
            //System.out.println(s);


            //out.println("<html>");
        // out.println("<head>");
            //out.println("<title>PdfToText</title>");
                // out.println("</head>");
                 //out.print("<body style='background-color:#808080' > ");
                 //out.print("<br>");
                 //out.print("<br>");
                //out.print("<br>");
                //out.print("<br>");
            /*out.print(""
                + ""
                + ""
                + ""
                + ""
                + ""
                + ""
                + ""
                + "");*/


                 out.println("Copying file started");
                 LuceneTester tester;
                 String dataDir=null;
```

```
                    String
indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";

                    dataDir                                   =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it";

                    String word =request.getParameter("query");

                    try {

                        tester = new LuceneTester();


tester.indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";

                    tester.dataDir                            =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it";

                    tester.createIndex();

                        out.println("index created");

                        }

                        catch (IOException e)

                        {

                            e.printStackTrace();

                      }

                    //out.print("<h1>");

                    out.print("<table align=center border=1 width=100%>");


                    out.print("<tr>");

                    out.print("<td>");


               out.println("Copying to"+"\n"+to);

              out.print("</tr>");
```

```
                         out.print("</td>");
out.print("</table>");
                   //out.print("</h1>");
                         //out.print("</html>");
             /*out.println(" "
                        + ""
                        + ""
                        + ""
                        + ""
                        + "  ");
             */



                          t=0;
                          System.out.println("internal break");
                         break;


        }
                if(t==0)
                {System.out.println("exterrnral break");
                break;
                }
```

```
                    }
                    for(String s : mec)
                    {//System.out.println(s);
                                   int t=1;String ch="/";
                         for(String in:inut)
                         if(in.equals(s))
                         {//PrintWriter out =response.getWriter();
                              System.out.print("mec"+"  "+s+"   "+in+"\n");
                                   File                    to=                    new
File("/home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/"+var.substring(var.lastIndexO
f(ch)+1));
                              copy(from,to);
                              System.out.println("in mec coped");
                              //out.println("<html>");
                           //out.println("<head>");
                            //out.println("<title>PdfToText</title>");
                          // out.println("</head>");
                              //out.print("<body style='background-color:#808080' > ");
                              //out.print("<br>");
                              //out.print("<br>");
                              //out.print("<br>");
                              //out.print("<br>");
                              // out.println("Copying file started");
                              //System.out.println(s);
                              LuceneTester tester;
```

```
                String dataDir=null;

                String
indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";

                dataDir                                        =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/mech";

                String word =request.getParameter("query");

                try {

                    tester = new LuceneTester();


    tester.indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";

                    tester.dataDir                             =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/mech";

                    tester.createIndex();

                    //out.print("<h1>index created</h1>");

                    }

                catch (IOException e)

                    {

                    e.printStackTrace();

                    }

            //out.print("<h1>");

            //  out.println("Copying "+var+" done to Index Generator as "+to);


            /*out.print(""

                    + ""

                    + ""

                    + ""
```

```
                              + ""

                              + ""

                              + ""

                              + ""

                              + "");*/


                //out.print("</h1>");

                //out.print("</html>");

                t=0;

                 System.out.println("internal break");

                break;

                 }

                 if(t==0)

                 {System.out.println("exterrnral break");

                 break;}

        }




        for(String s : novels)

        {//System.out.println(s);

                        int t=1;

                for(String in:inut)

                if(in.equals(s))

                {String ch="/";
```

```
//out.print(ch);

        PrintWriter outl =response.getWriter();

        System.out.print("nvels"+"  "+s+"   "+"in"+"\n");

        //out.println(var);


            File to= new File("/home/sampath/eeswar/Desktop/IndexGenerator/Data/novel/"+var.substring(var.lastIndexOf(ch)+1));

        copy(from,to);

        System.out.println("in noverls copied");

        //out.println(<html>);

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

        //out.println("<title>PdfToText</title>");

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

        //out.print("<body style='background-color:#808080' > ");

        //out.print("<br>");

        //out.print("<br>");

        //out.print("<br>");

        //out.print("<br>");

        /*out.print(""

            + ""

            + ""

            + ""

            + ""

            + ""
```

```
                                        + ""

                                        + ""

                                        + "");*/

                        out.println("Copying file started");

                                //System.out.println(s);

                                LuceneTester tester;

                                String dataDir=null;

                                //String indexDir="E:\\IndexGenerator\\Index";

                                //dataDir = "E:\\IndexGenerator\\Data\\novel";

                                String word =request.getParameter("query");

                                try {

                                        tester = new LuceneTester();


            tester.indexDir="/home/sampath/eeswar/Desktop/IndexGenerator/Index";

                                        tester.dataDir                                              =
"/home/sampath/eeswar/Desktop/IndexGenerator/Data/novel";

                                        tester.createIndex();

                                        out.println("index created");

                                        }

                                catch (IOException e)

                                        {

                                        e.printStackTrace();

                                        }

                                //out.print("<h1>");

                        //out.println("Copying "+var+" done to Index Generator as "+to);
```

```
                    //out.print("</h1>");

                    //out.print("</html>");

                    t=0;

                    System.out.println("internal break");

                    break;


                    }

                    if(t==0)

                    {System.out.println("exterrnral break");

                    break;}

            }



    }






        public static void intialise(String from)

            {addit(from);

                addmec(from);

                addnov(from);

                addinp(from);

            //it_cse.add();
```

```
                }

public static void  addit(String from)
                {

                        try {//PrintWriter out=new response.PrintWriter();
                                File file = new
File("/home/sampath/eeswar/Desktop/IndexGenerator/Pre/cse it/clusterwo.txt");
                                FileReader fileReader = new FileReader(file);
                                BufferedReader bufferedReader = new
BufferedReader(fileReader);
                                StringBuffer stringBuffer = new StringBuffer();
                                String line;
                                while ((line = bufferedReader.readLine()) != null) {
                                        stringBuffer.append(line);
                                        stringBuffer.append("\n");
                                }
                                fileReader.close();
                                String s=stringBuffer.toString();

                                String words = s.replaceAll("[^a-zA-Z\\s]", "");
                                words.replaceAll("\n"," ");
                                String formatted = words.trim().replaceAll(" +", " ");
                String[] k = Arrays.asList(formatted.split("\n")).stream().filter(str -
> !str.isEmpty()).collect(Collectors.toList()).toArray(new String[0]);

                System.out.println("it----------------------");

                for(String sa : k)

                {

                System.out.println(sa);

                    it_cse.add(sa.toLowerCase());


                }


            } catch (IOException e) {

                e.printStackTrace();

            }





        }

public static void addmec(String from)

{

    try {

        File file = new
File("/home/sampath/eeswar/Desktop/IndexGenerator/Pre/mech/clusterwo.txt");

        FileReader fileReader = new FileReader(file);

        BufferedReader bufferedReader = new BufferedReader(fileReader);

        StringBuffer stringBuffer = new StringBuffer();
        String line;

        while ((line = bufferedReader.readLine()) != null) {

            stringBuffer.append(line);

            stringBuffer.append("\n");

        }

        fileReader.close();

        String s=stringBuffer.toString();


    String k[]=s.split("\n");

    System.out.println("mec-----------------------");

        for(String sa : k)

        {System.out.println(sa);

            mec.add(sa.toLowerCase());


        }


    } catch (IOException e) {

        e.printStackTrace();

    }


}


public static void addnov(String from)

{
    try {
        File file = new
File("/home/sampath/eeswar/Desktop/IndexGenerator/Pre/novel/clusterwo.txt");
        FileReader fileReader = new FileReader(file);
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        StringBuffer stringBuffer = new StringBuffer();
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            stringBuffer.append(line);
            stringBuffer.append("\n");
        }
        fileReader.close();
        String s=stringBuffer.toString();

    String k[]=s.split("\n");
    System.out.println("nov-----------------------");
        for(String sa : k)
        {System.out.println(sa);
            novels.add(sa.toLowerCase());

        }

    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
}
public static void addinp(String from)
{
    try {System.out.println("hi");
        File file = new File(from);
        FileReader fileReader = new FileReader(file);
        BufferedReader bufferedReader = new
                BufferedReader(fileReader);
        StringBuffer stringBuffer = new StringBuffer();
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            stringBuffer.append(line);
            stringBuffer.append("\n");
        }
        fileReader.close();
        String s=stringBuffer.toString();

        String words = s.replaceAll("[^a-zA-Z\\s]", "");
        words.replaceAll("\n"," ");
        String formatted = words.trim().replaceAll(" +", " ");
        String[] k = Arrays.asList(formatted.split(" ")).stream().filter(str ->
!str.isEmpty()).collect(Collectors.toList()).toArray(new String[0]);
        for(String sa : k)
        {if(!sa.equals(" ")||!sa.equals("\r\n")||!sa.isEmpty())
        //System.out.println(sa);
            inut.add(sa.toLowerCase());


        }


    } catch (IOException e) {
        e.printStackTrace();
    }



}



 public static void copy(File src, File dest) throws IOException
 {
    InputStream is = null; OutputStream os = null;
    try {
        is = new FileInputStream(src);
        os = new FileOutputStream(dest);
        // buffer size 1K
        byte[] buf = new byte[1024]; int bytesRead;
        while ((bytesRead = is.read(buf)) > 0)
        { os.write(buf, 0, bytesRead);
```

```
                        }
                        System.out.println("Copying file done");
                        }
              finally {
                        is.close();
                        os.close();
                        }
             }

  }
```

ExtractPageContent.java

```
import java.io.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintWriter;

import com.itextpdf.text.pdf.PdfReader;
import com.itextpdf.text.pdf.parser.PdfReaderContentParser;
import com.itextpdf.text.pdf.parser.SimpleTextExtractionStrategy;
import com.itextpdf.text.pdf.parser.TextExtractionStrategy;
```

```
public class ExtractPageContent{


    /** The original PDF that will be parsed. */
   // public static final String PREFACE = "E:\\programming hive.pdf";
    /** The resulting text file. */
   // public static final String RESULT = "E:\\programming hive.txt";


    /**
     * Parses a PDF to a plain text file.
     * @param pdf the original PDF
     * @param txt the resulting text
     * @throws IOException
     */
    public void parsePdf(String pdf, String txt) throws IOException {
        PdfReader reader = new PdfReader(pdf);
        PdfReaderContentParser parser = new PdfReaderContentParser(reader);
        PrintWriter out = new PrintWriter(new FileOutputStream(txt));
        TextExtractionStrategy strategy;
        for (int i = 1; i <= reader.getNumberOfPages(); i++) {
            strategy = parser.processContent(i, new SimpleTextExtractionStrategy());
            out.println(strategy.getResultantText());
        }
        reader.close();

        out.flush();

        out.close();

    }



    /**

     * Main method.

     * @param    args    no arguments needed

     * @throws IOException

     */

    //public static void main(String[] args) throws IOException {

    //    new ExtractPageContent().parsePdf(PREFACE, RESULT);

     //   System.out.print("Complted");

   // }

}
```

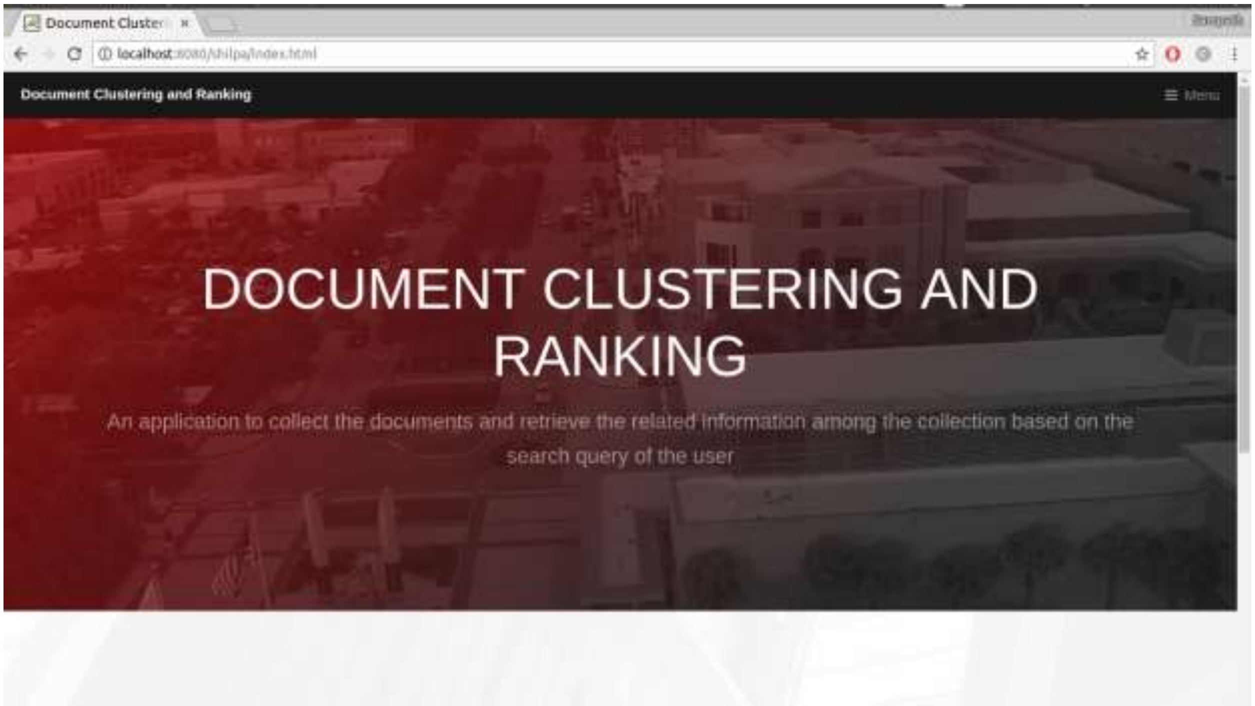
ExtractPageContentServ.java

```
import java.io.*;

import java.util.Scanner;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class ExtractPageContentServ extends HttpServlet {

        private static final long serialVersionUID = 1L;




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

                String var=request.getParameter("myFile");

                var="/home/sampath/eeswar/Desktop/IndexGenerator/pdfs/"+var;

                PrintWriter out =response.getWriter();

           out.println(" Converting "+var+"  file");

                ExtractPageContent exp=new ExtractPageContent();

                out.println("before"+var);

                 /** The original PDF that will be parsed. */

                 //String ch="\\";

                 // var=var.replace(ch,"//");

                 out.println("after"+var);

           String pdffile= var;

                String PREFACE=pdffile;


                String var1=var.replace(".pdf", ".txt");
```

```
                    out.println("textflie"+var1);
//var1="E://shilpa//programming hive.txt";
                    /** The resulting text file. */
                     String txtfile=var1;
              String RESULT =txtfile;


                    exp.parsePdf(PREFACE, RESULT);


                    }



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


                    doGet(request, response);
          }


}
```

5.2 Input and Output Screens

5.2.1 Screenshots:


Document Cluster
localhost:8080/shilpa/index.html
Document Clustering and Ranking
Menu

DOCUMENT CLUSTERING AND RANKING

An application to collect the documents and retrieve the related information among the collection based on the search query of the user

Fig 5.2.1.1

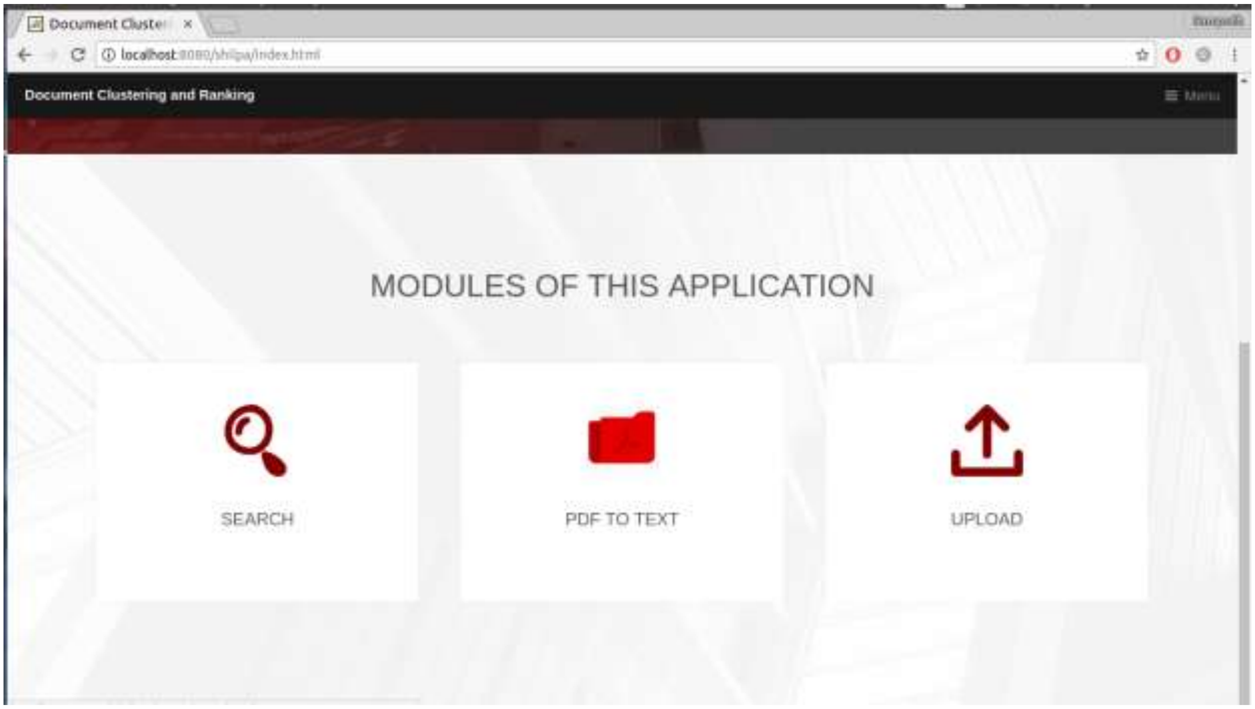
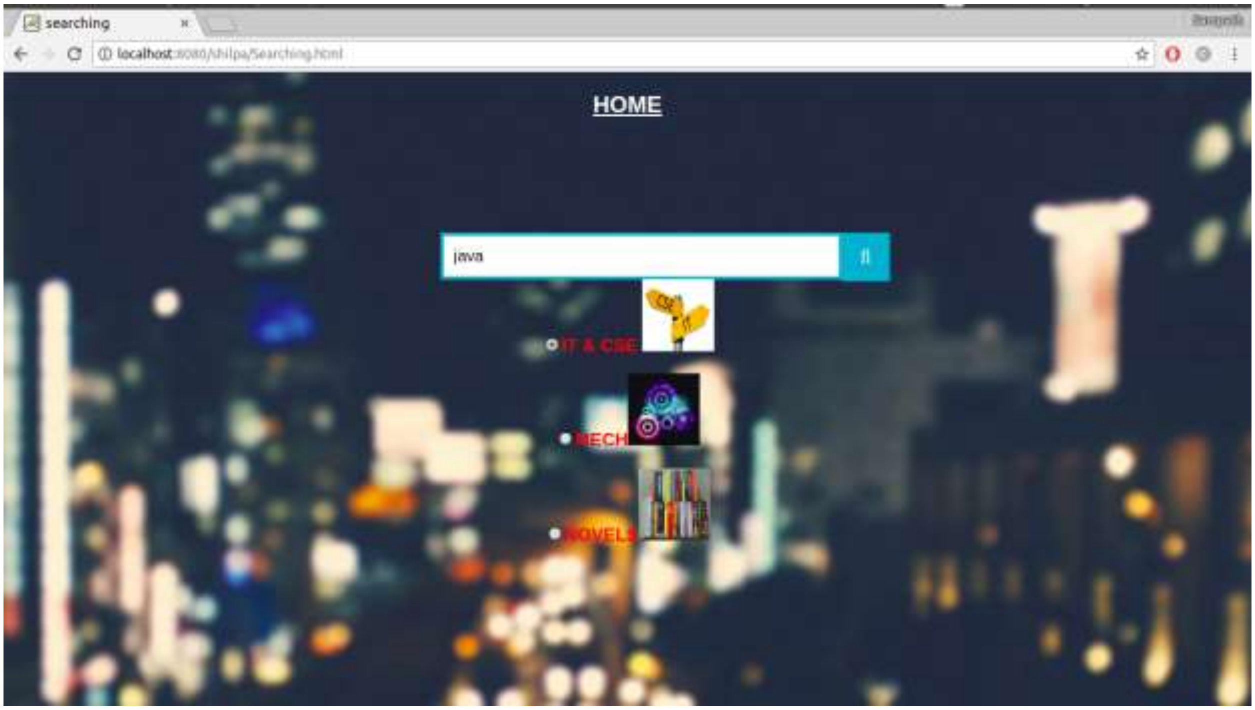
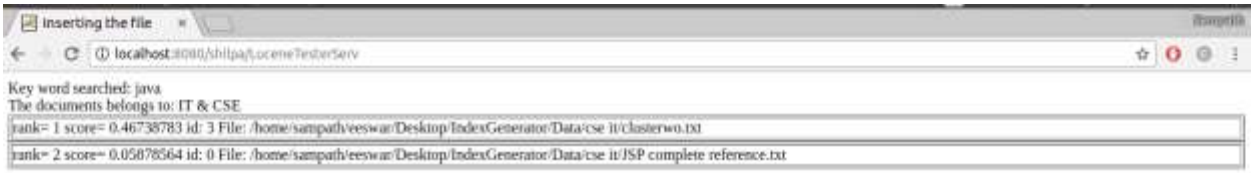

Document Cluster ×
localhost:8080/shilpa/index.html
Document Clustering and Ranking
≡ Menu
MODULES OF THIS APPLICATION
SEARCH
PDF TO TEXT
UPLOAD

Fig 5.2.1.2



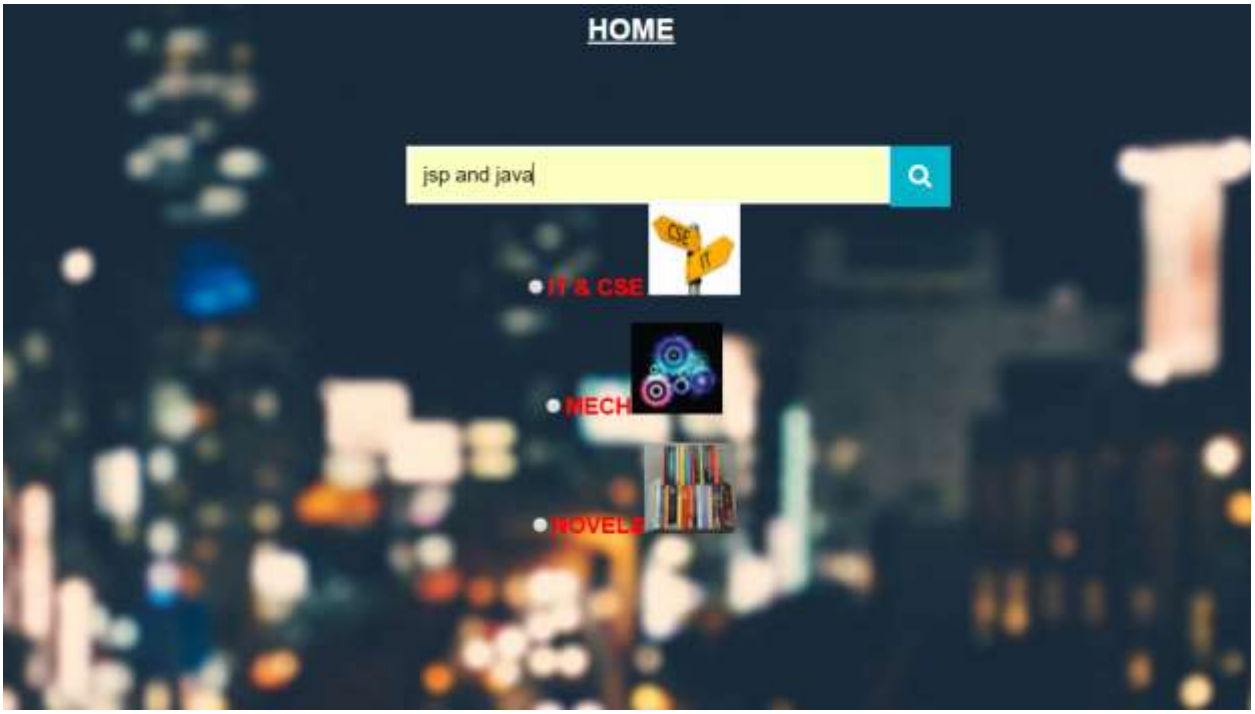
java

Fig 5.2.1.3 Term Query



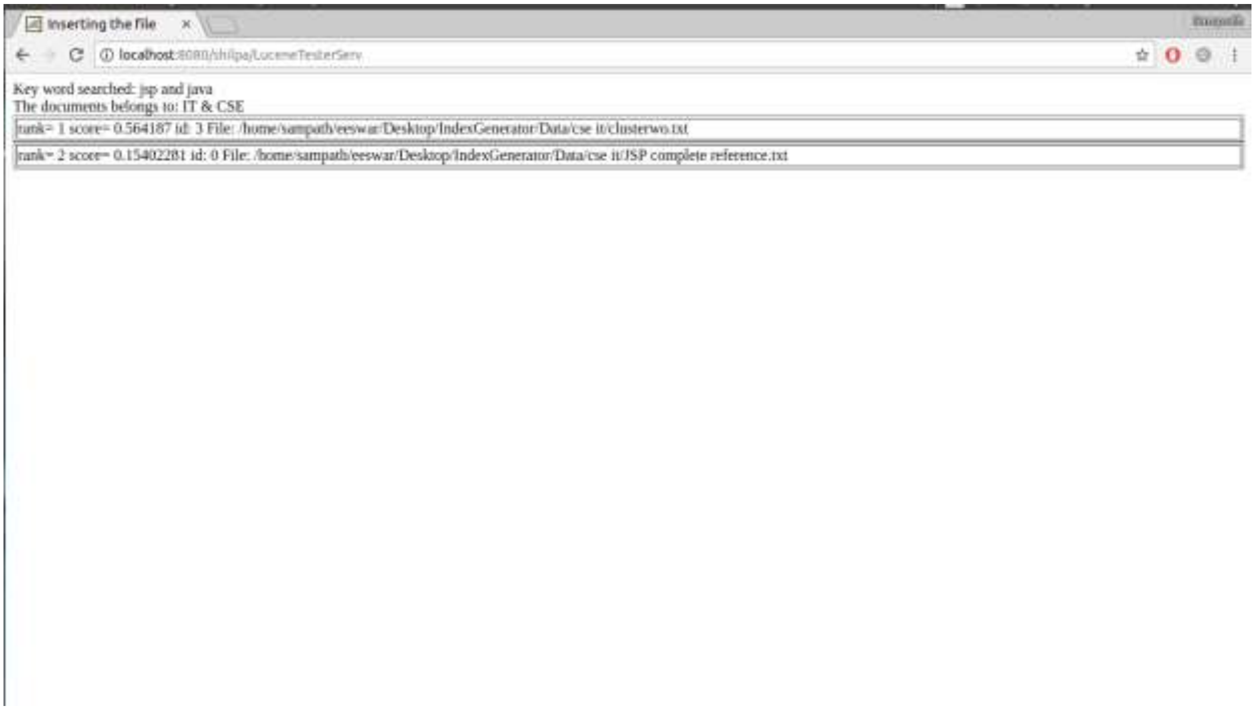
```
Key word searched: java
The documents belongs to: IT & CSE
rank= 1 score= 0.46738783 id: 3 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it/clusterwo.txt
rank= 2 score= 0.05878564 id: 0 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it/JSP complete reference.txt
```

Fig 5.2.1.4



jsp and java

Fig 5.2.1.5 Boolean Query



```
Key word searched: jsp and java
The documents belongs to: IT & CSE
rank= 1 score= 0.564187 id: 3 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it/clusterwo.txt
rank= 2 score= 0.15402281 id: 0 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it/JSP complete reference.txt
```

Fig5.2.1.6

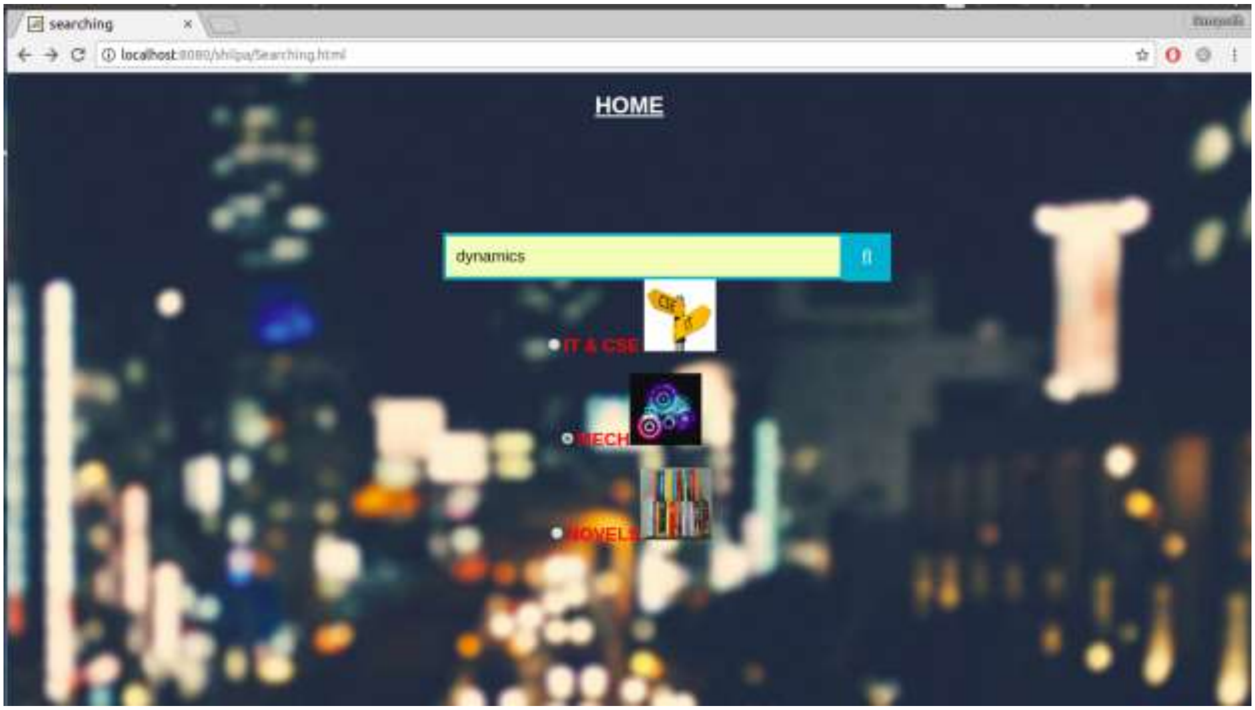
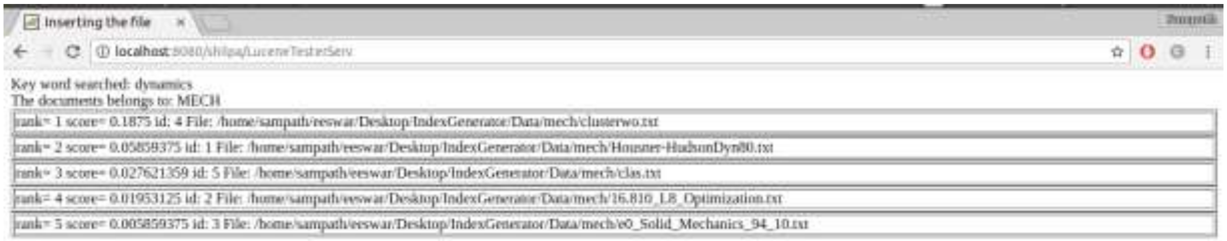


Fig 5.2.1.7



```
Key word searched: dynamics
The documents belongs to: MECH
rank= 1 score= 0.1875 id: 4 File: /home/sampath/reswar/Desktop/IndexGenerator/Data/mech/clusterwo.txt
rank= 2 score= 0.05859375 id: 1 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/Housner-HudsonDyn80.txt
rank= 3 score= 0.027621359 id: 5 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/clas.txt
rank= 4 score= 0.01953125 id: 2 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/16.810_L8_Optimization.txt
rank= 5 score= 0.005859375 id: 3 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/e0_Solid_Mechanics_94_10.txt
```

Fig 5.2.1.8

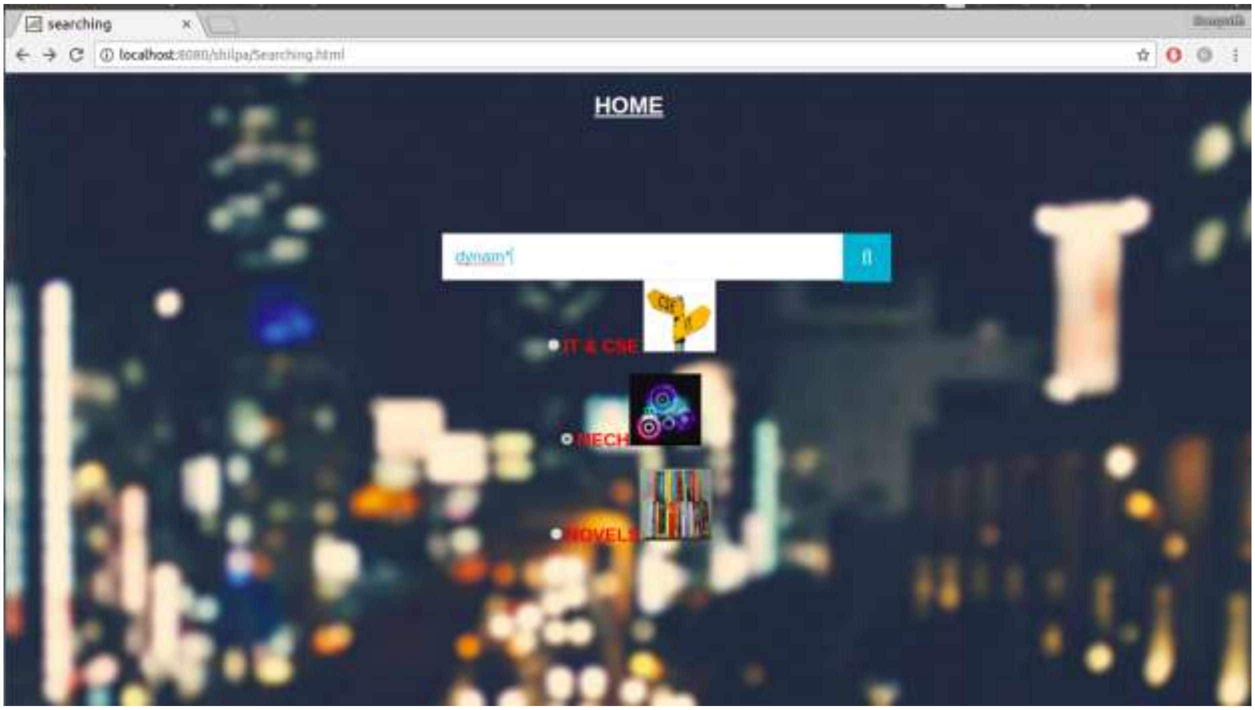
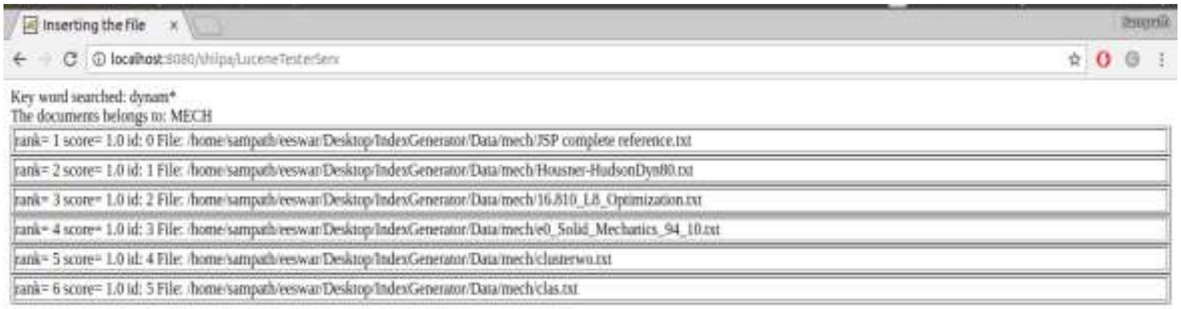


Fig 5.2.1.9 Wildcard Query



```
Key word searched: dynam*
The documents belongs to: MECH
rank= 1 score= 1.0 id: 0 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/JSP complete reference.txt
rank= 2 score= 1.0 id: 1 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/Housner-HudsonDyn80.txt
rank= 3 score= 1.0 id: 2 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/16.810_L8_Optimization.txt
rank= 4 score= 1.0 id: 3 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/e0_Solid_Mechanics_94_10.txt
rank= 5 score= 1.0 id: 4 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/clusterwu.txt
rank= 6 score= 1.0 id: 5 File: /home/sampath/eeswar/Desktop/IndexGenerator/Data/mech/clas.txt
```

Fig 5.2.1.10

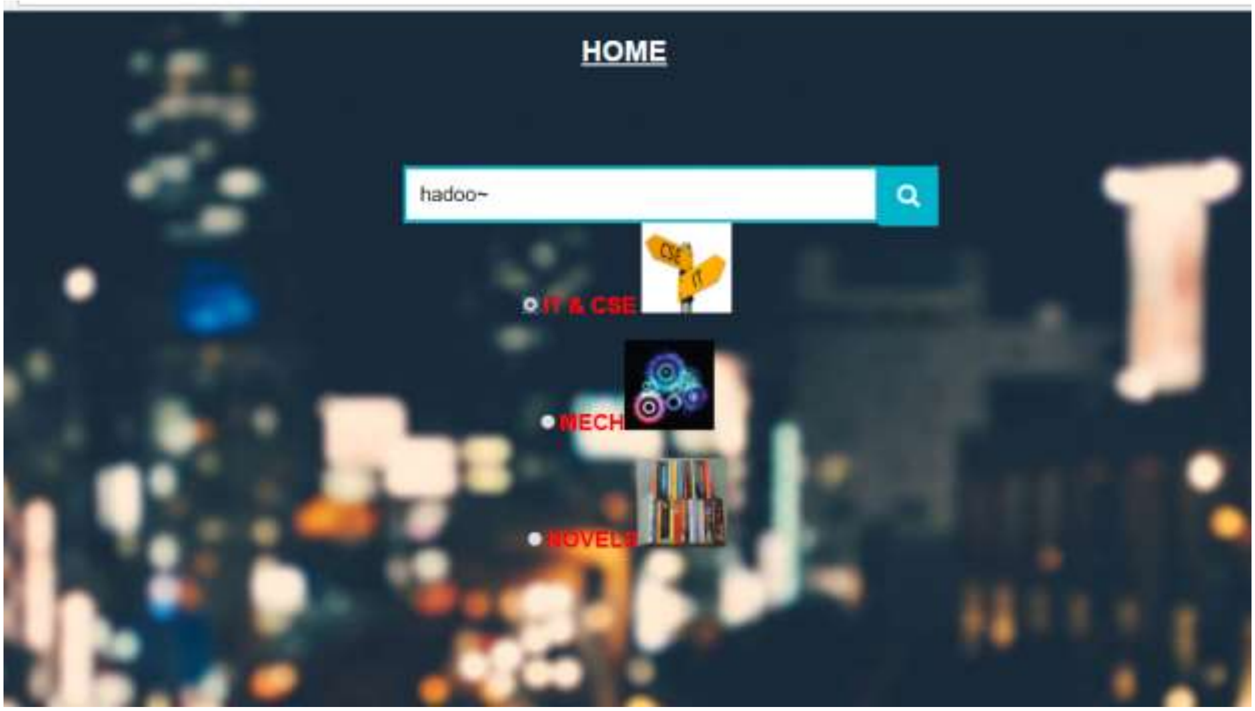
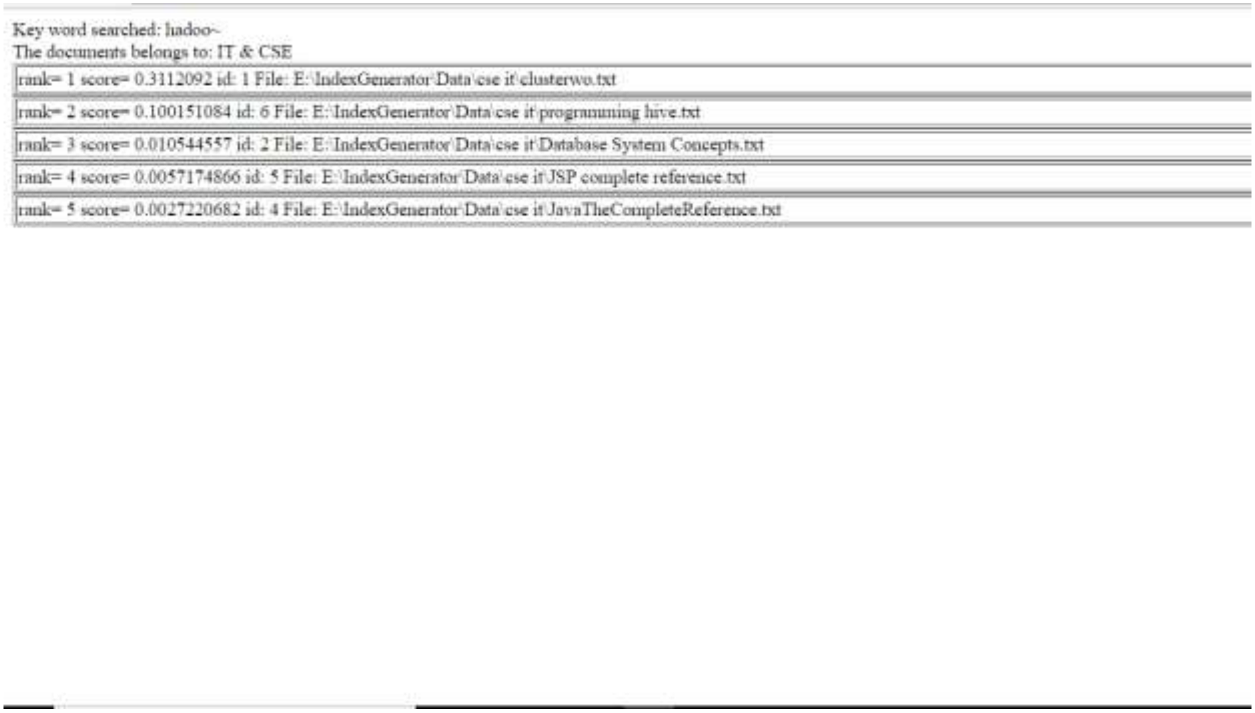
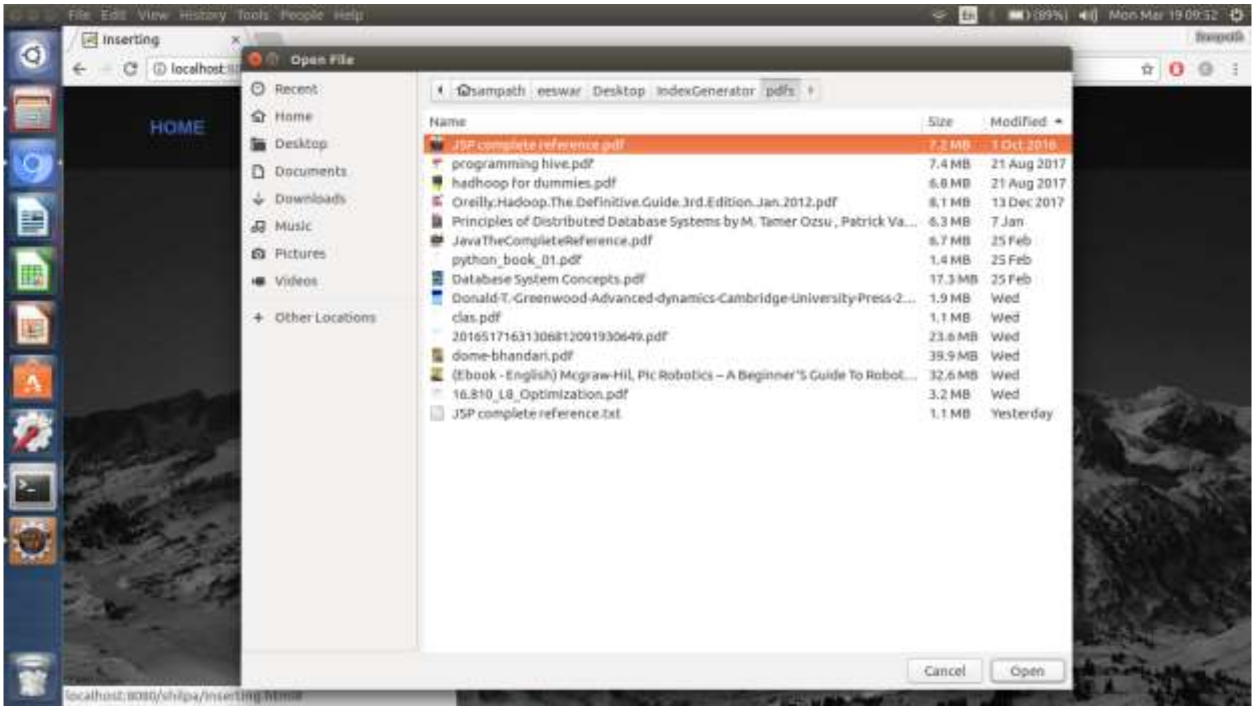


Fig 5.2.1.11



```
Key word searched: hadoo~
The documents belongs to: IT & CSE
rank= 1 score= 0.3112092 id: 1 File: E:\IndexGenerator\Data\cse it\clusterwo.txt
rank= 2 score= 0.100151084 id: 6 File: E:\IndexGenerator\Data\cse it\programming hive.txt
rank= 3 score= 0.010544557 id: 2 File: E:\IndexGenerator\Data\cse it\Database System Concepts.txt
rank= 4 score= 0.0057174866 id: 5 File: E:\IndexGenerator\Data\cse it\JSP complete reference.txt
rank= 5 score= 0.0027220682 id: 4 File: E:\IndexGenerator\Data\cse it\JavaTheCompleteReference.txt
```

Fig 5.2.1.12



Name	Size	Modified
JSP complete reference.pdf	7.2 MB	1 Oct 2016
programming hive.pdf	7.4 MB	21 Aug 2017
hadhoop for dummies.pdf	6.8 MB	21 Aug 2017
Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.pdf	8.1 MB	13 Dec 2017
Principles of Distributed Database Systems by M. Tamer Ozsu , Patrick Va…	6.3 MB	7 Jan
JavaTheCompleteReference.pdf	8.7 MB	25 Feb
python_book_01.pdf	1.4 MB	25 Feb
Database System Concepts.pdf	17.3 MB	25 Feb
Donald-T.-Greenwood-Advanced-dynamics-Cambridge-University-Press-2…	1.9 MB	Wed
clas.pdf	1.1 MB	Wed
20165171631306812091930649.pdf	23.6 MB	Wed
dome-bhandari.pdf	39.9 MB	Wed
(Ebook - English) Mcgraw-Hil, Pic Robotics – A Beginner'S Guide To Robot…	32.6 MB	Wed
16.810_L8_Optimization.pdf	3.2 MB	Wed
JSP complete reference.txt	1.1 MB	Yesterday

Fig 5.2.1.13

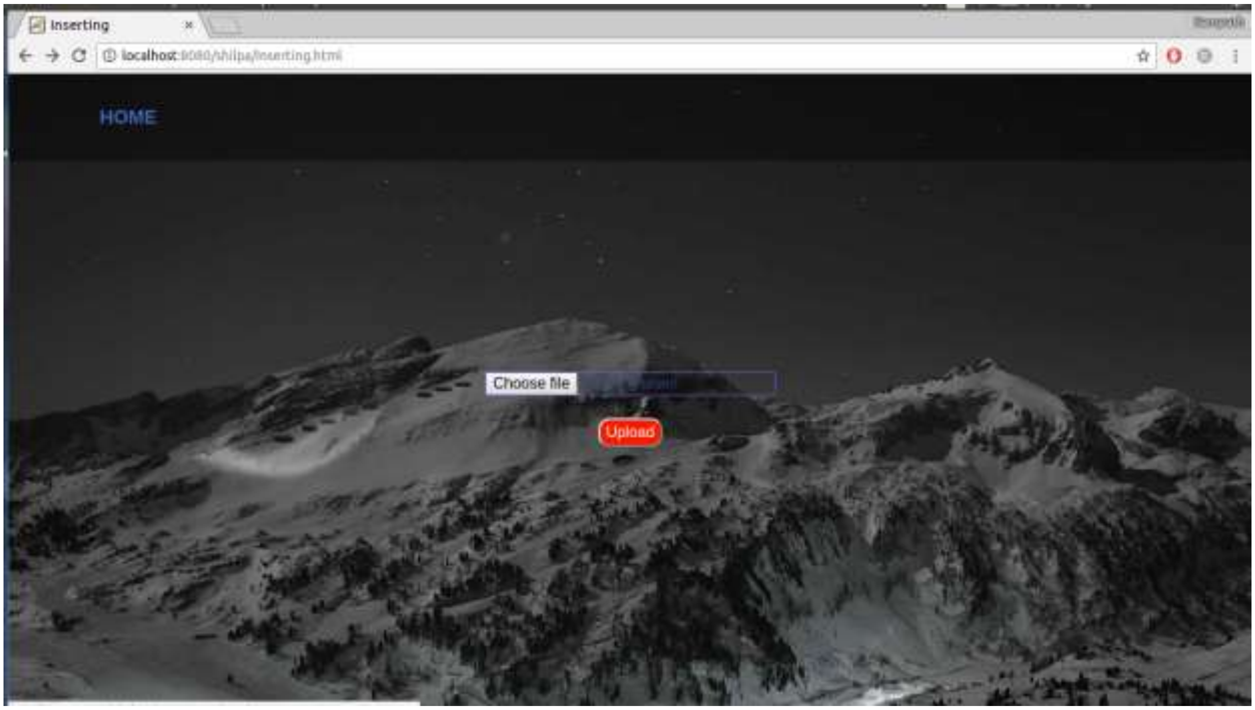


Fig 5.2.1.14

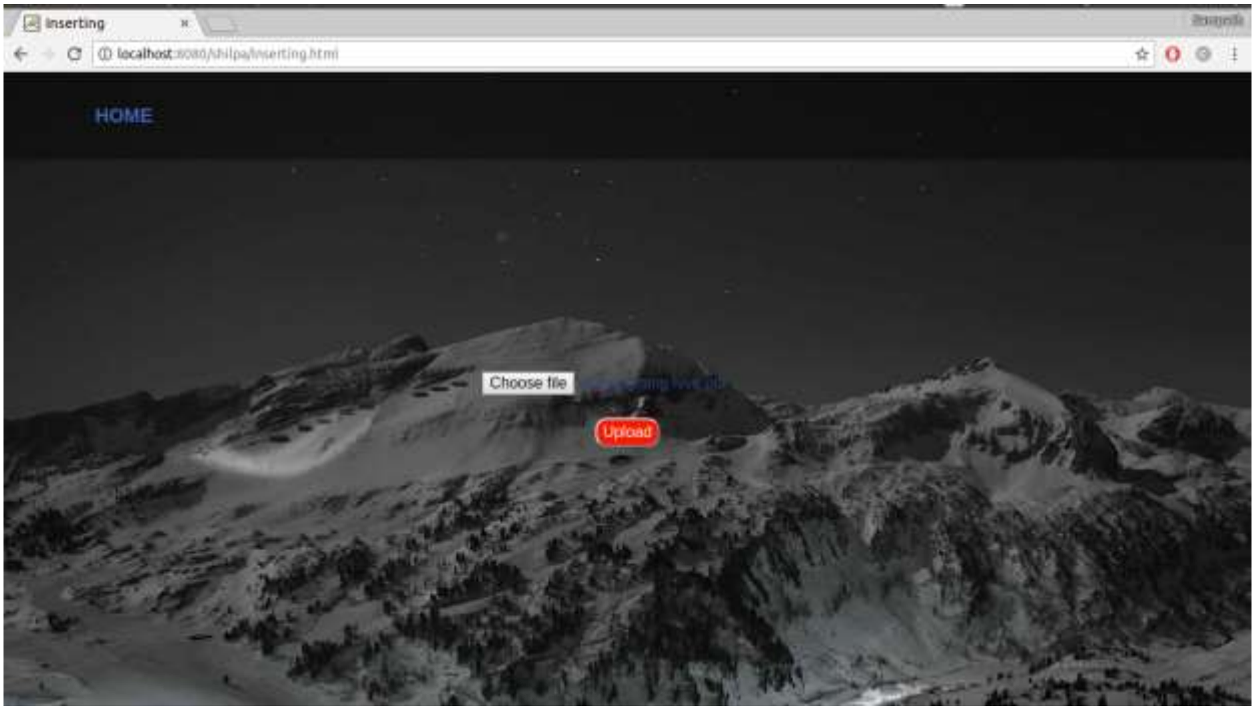
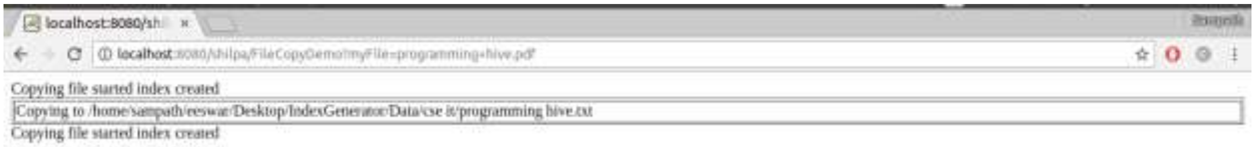
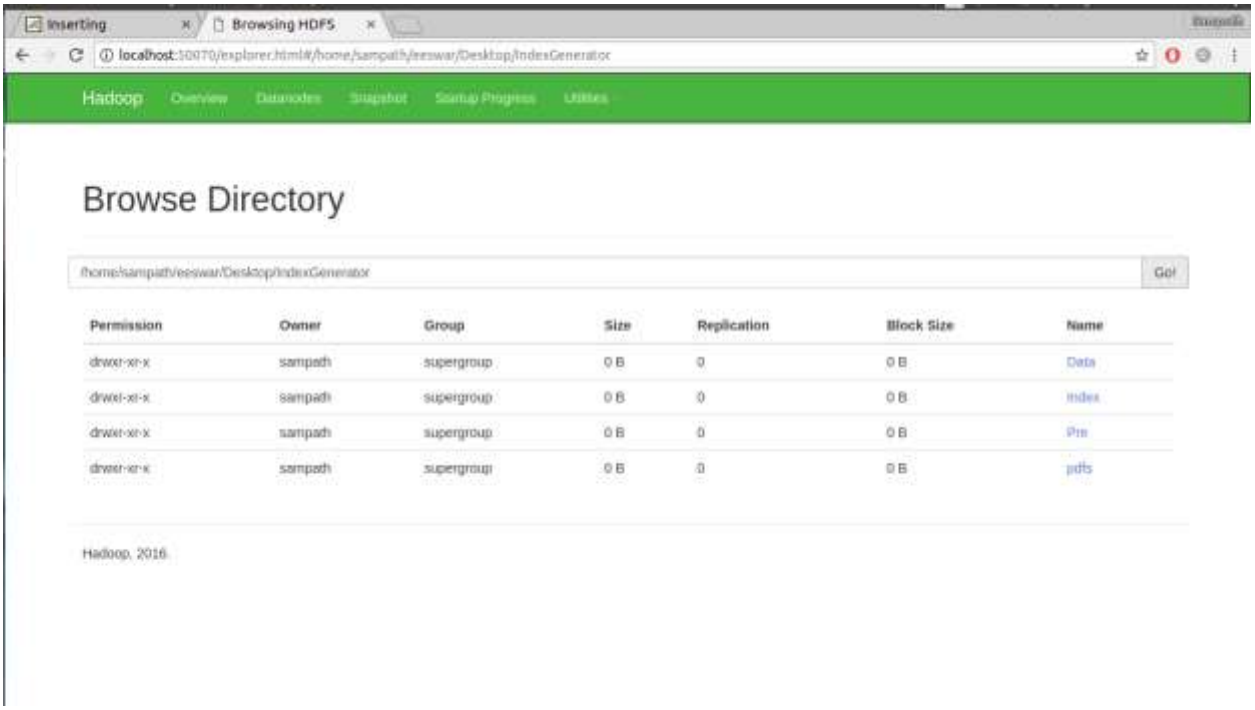


Fig 5.2.1.15


localhost:8080/sh
localhost:8080/shilpa/FileCopyDemo?myFile=programming+hive.pdf

```
Copying file started index created
Copying to /home/sampath/eeswar/Desktop/IndexGenerator/Data/cse it/programming hive.txt
Copying file started index created
```

Fig 5.2.1.16



Browse Directory

/home/sampath/eeswar/Desktop/IndexGenerator

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	sampath	supergroup	0 B	0	0 B	Data
drwxr-xr-x	sampath	supergroup	0 B	0	0 B	Index
drwxr-xr-x	sampath	supergroup	0 B	0	0 B	Pre
drwxr-xr-x	sampath	supergroup	0 B	0	0 B	pdfs

Hadoop, 2016.

Fig 5.2.1.17 Hdfs view of the data

6. TESTING-TEST DATA SET, TEST CASES

Software Testing is a critical element of software quality assurance and represents the ultimate service of specification design and coding. The increasing visibility of software as a system element and the attended costs associated with the software failure and motivating forces for well planned, through testing. It is not unusual for a software development to spend between 30 to 40% of total project effort in testing.

System Testing Strategies for this system integrate test case design techniques into a well-planned series of steps that result in the successful construction of this software. It also provides a road map for the developer, the quality assurance organization and the customer, a roadmap that describes the steps to be conducted as path of testing, when these steps are planned and then undertaken and how much effort, time and resources will be required. The test provisions are follows.

6.1 Testing Objectives

The following are the testing objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers as a yet undiscovered error.

The above objectives imply a dramatic change in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically different clauses of errors and do so with minimum amount of time and effort.

If testing is conducted successfully, it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification and that performance requirements appear to have been met. In addition, data collected as testing is conducted provides a good indication of software. Testing can’t show the absence of defects, it can only show that software errors are present. It is important to keep this stated in mind as testing is being conducted.

6.2 Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing.

- All tests should be traceable to customer requirements.
- Tests should be planned before testing begins.
- Testing should begin “in the small” and progress towards testing “in the large”.
- Exhaustive testing is not possible.

6.3 Testing Strategies

A strategy for software testing indicates software test case design methods in to a well-planned series of steps that results in the successful construction of software. The strategy provides a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time and resources will be required. It must be rigid enough to promote reasonable planning and management tracking as the project progresses. The strategies for testing are envisioned by the following methods. A number of software testing strategies have been proposed. All provide the software developer with a template for testing and all following generic characteristics.

- To perform effective testing. A software team should conduct effective formal technical reviews. By doing this, many errors will be eliminated before testing commences.
- Testing begins at the component level and works “outward” towards the integration of entire computer based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and an independent test group.
- Testing and debugging are different activities, but debugging must accommodate in any testing strategy.

A strategy for software testing must accommodate low level test that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validates major system functions against customer requirements. A strategy must provide guidance for the practitioner and set of milestones for the manager.

6.4 Types of Testing

The primary objective of test case design is to derive a set of tests that have the highest likelihood for uncovering errors in the software. To accomplish this objective two different categories of test case design techniques are used.

1. White-Box Testing

White box testing sometimes called glass box testing is a test case designs that focus on the program control structure. Test cases are derived to ensure that

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical design on their true and false sides.
3. Executes all loops at their boundaries and within their operational boundaries.
4. Exercise internal data structure to ensure their validity.

Several methods are used in the white box testing.

2. Black Box Testing

Tests can be conducted at software interface by knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational, at the same time searching for errors is called black box testing, sometimes called as behavioural testing. Black box testing is not an alternative to white box techniques. Rather it is a complementary approach that is likely to uncover a different class of errors than white box methods. Black box tests are designed to uncover errors in functional requirements without regard to the internal workings of a program. Black box testing techniques focus on the information domain of the software.

Black box testing attempts to find errors in the following categories.

1. Incorrect or missing functions.
2. Interface errors.
3. Errors in the data structures or external database access.

4. Performance errors.

5. Initialization and termination errors.

3. System Testing

System tests are designed to validate a fully developed system with a view to assuming that it meets its requirements. System testing is actually a series of different tests, whose primary purpose is to be fully exercising the computer-based system. In this system, although each test has a different purpose, all the works are verified to ensure that all system elements have been properly integrated and performed allocated functions.

There are three kinds of system testing:

1. Alpha Testing: Alpha testing refers to the system testing that is carried by the customer within the organization along with the developer. The alpha test are conducted in controlled manner.
2. Beta Testing: Beta testing is the system performed by a selected group of customers, the developer is not present at the site and the user will inform the problems that are encountered during testing. The software developer makes the necessary changes and submits to the customer.
3. Acceptance Testing: Acceptance testing is the system testing performed by the customer to whether or not to accept the delivery of the system.

4. Unit Testing

Unit testing focuses verification effort on the smallest unit of the software design, the module. Using the detailed design description as a guide, important control paths are tested to uncover errors with the boundary of the module for the following modules. All the statements in the module are executed at least once. From this we can ensure that all independent paths through the control structures are exercised.

5. Integration Testing

Integration testing is a systematic technique for constructing the program structures and to conduct tests for uncovered errors with interfacing.

6.5 TESTCASES

S.No	Input	Expected Output	Obtained Output	Remarks
1	Term Query	Documents related to the given term in relevant order	Documents related to the given term in relevant order	Pass
2	Phrase Query	Documents related to the given phrase in relevant order	Documents related to the given phrase in relevant order	Pass
3	Boolean Query	Documents related to the given boolean query in relevant order	Documents related to the given boolean query in relevant order	Pass
4	Wildcard Query	Documents related to the given wildcard query in relevant order	Documents related to the given wildcard query in relevant order	Pass
5	Group Query	Documents related to the given group query in relevant order	Documents related to the given group query in relevant order	Pass
6	Fuzzy Query	Documents related to the given fuzzy query in relevant order	Documents related to the given fuzzy query in relevant order	Pass

8. CONCLUSION

This application ensures to collect the documents, form groups out of the given documents based upon the uploaded book consisting of the words in it by placing a file consisting of the predefined words allotted in the file, and later retrieve the exact and the most relevant document among the clusters, depending upon the search query from the user. We are expecting to provide the user with the most relevant document for the given query through this application and pdf to text convertor as this application is purely text based and generally the eBooks are of pdf type.

The existing system has a facility that is when given a query; it searches only the title of the book and author of the book for the particular word given by the user. This process may not give the accurate result of what the user is expecting and we know that every user expects to retrieve the search result in less time.

This system enables us to maintain the pdf files later convert them to the text files using this application, which helps us in reducing the size and later on divide them into clusters in order to reduce the time of searching of the documents. Retrieving the information and documents as per the user requested queries which are relevant to what the user is expecting based on the ranking is provided to the users. Result is produced in the ranking order of the frequency of the word present in the document.

9. BIBLIOGRAPHY

9.1 List of Book References:

- Java Complete Reference, Herbert Schildt
- Database Programming with JDBC and JAVA, Reese and George
- Developing Java Servlets, Goodwill and James
- HTML & CSS: Design and Build Web Sites, John Duckett
- Java servlet programming by Jason Hunter

9.2 List of Web References:

- https://www.javatpoint.com
- http://www.tutorialspoint.com
- https://www.w3schools.com