SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

Hadoop add-on API for Advanced Content Based Search & Retrieval

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Computer Engineering)

Kshama Jain	B120334280
Aditya Kamble	B120334297
Siddhesh Palande	B120334356
Rahul Rao	B120334381

Under The Guidance of

Prof. Shailesh Hule

DEPARTMENT OF COMPUTER ENGINEERING
Pimpri Chinchwad College of Engineering
Sector -26, Pradhikaran, Nigdi Pune, Maharashtra 411044

Pimpri Chinchwad College of Engineering
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project Entitled

Hadoop add-on API for Advanced Content Based Search & Retrieval

Submitted By

Kshama Jain	B120334280
Aditya Kamble	B120334297
Siddhesh Palande	B120334356
Rahul Rao	B120334381

is a bonafide work carried out by Students under the supervision of Prof. Guide Name and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering) Project.

Prof. Shailesh Hule
Internal Guide
Dept. of Computer Engg.

Prof. Dr. K. Rajeswari
H.O.D
Dept. of Computer Engg.

Abstract

Unstructured data like doc, pdf, accdb is lengthy to search and filter for desired information. We need to go through every file manually for finding information. It is very time consuming and frustrating. It doesnt need to be done this way if we can use high computing power to achieve much faster content retrieval.

We can use features of big data management system like Hadoop to organize unstructured data dynamically and return desired information. Hadoop provides features like Map Reduce, HDFS, HBase to filter data as per user input. Finally we can develop Hadoop Addon for content search and filtering on unstructured data. This addon will be able to provide APIs for different search results and able to download full file, part of files which are actually related to that topic. It will also provide API for context aware search results like most visited documents and much more relevant documents placed first so user work get simplified.

This Addon can be used by other industries and government authorities to use Hadoop for their data retrieval as per their requirement.

After this addon, we are also planning to add more API features like content retrieval from scanned documents and image based documents

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***Hadoop add-on API for Advanced Content Based Search & Retrieval****.*

I would like to take this opportunity to thank my internal guide ***Prof. Shailesh Hule*** *for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.*

I am also grateful to ***Prof. Dr. K. Rajeswari****, Head of Computer Engineering Department, Pimpri Chinchwad College of Engineering for his indispensable support, suggestions.*

In the end our special thanks to ***teaching and non teaching staff*** *for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.*

Kshama Jain
Aditya Kamble
Siddhesh Palande
Rahul Rao
(B.E. Computer Engg.)

INDEX

List of Figures

List of Tables

CHAPTER 1

SYNOPSIS

1.1 PROJECT TITLE

Hadoop add-on API for Advanced Content Based Search & Retrieval

1.2 PROJECT OPTION

Persistent Systems Sponsored Project

1.3 INTERNAL GUIDE

Prof. Shailesh Hule

1.4 SPONSORSHIP AND EXTERNAL GUIDE

Mr. Atul Shimpi (Persistent Systems)

1.5 TECHNICAL KEYWORDS (AS PER ACM KEYWORDS)

- Hadoop
- HDFS
- MapReduce
- HBase
- Content Based System

1.6 PROBLEM STATEMENT

Hadoop add-on API for Advanced Content Based Search & Retrieval

1.7 ABSTRACT

Please Write here One Page Abstract. It should mainly include introduction, motivation,outcome and innovation if any. Unstructured data like doc, pdf, accdb is lengthy to search and filter for desired information. We need to go through every file manually for finding information. It is very time consuming and frustrating. It doesnt

need to be done this way if we can use high computing power to achieve much faster content retrieval.

We can use features of big data management system like Hadoop to organize unstructured data dynamically and return desired information. Hadoop provides features like Map Reduce, HDFS, HBase to filter data as per user input. Finally we can develop Hadoop Addon for content search and filtering on unstructured data. This addon will be able to provide APIs for different search results and able to download full file, part of files which are actually related to that topic. It will also provide API for context aware search results like most visited documents and much more relevant documents placed first so user work get simplified.

This Addon can be used by other industries and government authorities to use Hadoop for their data retrieval as per their requirement.

After this addon, we are also planning to add more API features like content retrieval from scanned documents and image based documents

1.8 GOALS AND OBJECTIVES

Goals:

Current Systems Focus on Search by Title,Author,etc which Is time consuming and finding relevant content from those documents is tedious task. So there is a need of such a system which shall find the relevant contents to the end user

Objective:

To find the relevant content from the huge number of PDF files present on Hadoop Distributed File System

1.9 RELEVANT MATHEMATICS ASSOCIATED WITH THE PROJECT

S = {s,e,x,y,DD,NDD,Mem-shared}

s= start state $->$ Taking input from the user as search query

e= End State $->$ return the output to the user in the form text based content

x = Input $->$ Search Query

y = Output $->$ Text Based Result

DD = Deterministic Data

1) Number of PDF Files

2) Keyword Tokenisation and Filteration

3) Number of DataNodes

4) Search Progress

5) Number of Results Obtained

NDD = Non Deterministic Data

1) Failure of Cluster Nodes

2) Communication Failure

Mem-Shared = Storage Space

1) HDFS will be distributed among a number of nodes in Hadoop Cluster and will share common FileSystem which will be managed by Hadoop

1.10 NAMES OF CONFERENCES / JOURNALS WHERE PAPERS CAN BE PUBLISHED

IFERP - International Conference Institute for Engineering Research and Publication

1.11 REVIEW OF CONFERENCE/JOURNAL PAPERS SUPPORTING PROJECT IDEA

The reviewers committee of NIER congratulate you for acceptance of your research paper for International Conference. You are cordially invited to convene the event by presenting your research paper through PowerPoint presentation.The Conference is being organized by NIER-India in association with ”Technoarete”

1.12 PLAN OF PROJECT EXECUTION

Activity	Weeks to Spend	Deliverables	Priority
Analysis of Existing System	2 weeks	-	Normal
Requirement Gathering	2 weeks	Requirements	Normal
Literature Survey	3 Week	-	Normal
Designing and Planning	5 weeks	Modules	High
Implementation	10 weeks	API	High
Testing	3 weeks	Test Report	High
Documentation	4 week	Project Report	Normal

Table 1.1: Plan of Project Execution

CHAPTER 2

TECHNICAL KEYWORDS

2.1 AREA OF PROJECT

This system shall retrieve the required contents of files which are in an unstructured format containing huge amount of Data like e-books in a digital library where the number of books are present in thousands. The scope here is initially limited to PDF files which may be expanded to other unstructured formats like ePUB, mobi.

The input is provided to the system API in the form of search query which will be firstly filtered to find the important expression as a query to the API which will return the important content to the user in the form of paragraph or text highlighted using the power of distributed computing.

The particular page or the Entire book itself can be downloaded by the Library users if the content is satisfied else the search continues for finding relevant content

2.2 TECHNICAL KEYWORDS

1. Hadoop: o Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models.
    - Distributed
    - Scalable
    - Fault-tolerant
    - Open source
2. HDFS:
    - HDFS, the Hadoop Distributed File System, is responsible for storing data on the cluster.
    - Data is split into blocks and distributed across multiple nodes in the cluster.

- HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers.

3. MapReduce

   - MapReduce is the system used to process data in the Hadoop Cluster.
   - Consist of to phases:Map and the Reduce
   - Each map task operates on discrete portion of the overall dataset
   - After all Maps are compete, the mapReduce system distributes the intermediate data to nodes which perform the Reduce phase

4. HBase

   - HBase is an open source, non-relational, distributed database
   - The data is partitioned based on the RowKeys into Regions.
   - Each Region contains a range of RowKeys based on their binary order.
   - A RegionServer can contain several Regions.

5. Content based:

   - Using information manually entered or included in the table design, such as titles, descriptive keywords from a limited vocabulary, and predetermined classification schemes. The primary benefit of using content-based retrieval is reduced time and effort required to obtain image-based information.

CHAPTER 3

INTRODUCTION

3.1 PROJECT IDEA

Basic project idea is to reduce manual efforts for content based searching in large set of documents using Hadoop Big Data Management framework to automate content based searching and retrieving the relevant content.

3.2 MOTIVATION OF THE PROJECT

Taks like assignment completion, taking notes from text books and reference books on particular topic, topics for presentation need deep reading and need to go through every document manually just to find relevant content on given topic.

Currently present systems are only searching based on document title, author, size, and time but not on content. So to do content based search on big data documents and large text data Haddop framework can be used.

So using Hadoop Big Data management framework consist of HDFS, MapReduce, and HBase, we are developing content based search on PDF documents to solve real life problem. So this is basic motivation for the project.

3.3 LITERATURE SURVEY

Title	Keyword	Content	Author
Hadoop-HBase for Large- Scale Data (2011)	large-scale data; distributed storage; Hadoop; HDFS; Map Reduce; HBase; noSQL database	The paper aims at evaluating the performance of random reads and random writes of data storage location information to HBase and retrieving and storing data in HDFS respectively.	Mehul Nalin Vora
High Performance and Fault Tolerant Distributed File System for Big Data Storage and Processing using Hadoop(2014)	Big data; Analytics; Hadoop; Hadoop Distributed File System (HDFS); Hype cycle; MapReduce; Replication; Faulttolerance; Unstructured data	In this paper they have highlighted the evolution and rise of big data and discussed how HDFS produces multiple replicas of data.	E.Sivaraman, Dr.R.Manickachezian
Handling Big Data Efficiently by using Map Reduce Technique(2014)	Data Mining; Clustering; DBMS; Parallel processing; Hadoop; MapReduce.	In this paper, They discussed work around MapReduce, its advantages, disadvantages and how it can be used in integration with other technology.	Seema Maitrey, C.K. Jha

The Dawn of Big Data - Hbase	HBase, Hadoop Distributed File System (HDFS), HBase column oriented table.	This paper includes the step by step introduction to the HBase,IdentifY differences between apache HBase and a traditional RDBMS.	Vijayalakshmi Bhupathirajul, Ravi Prasad Ravud

CHAPTER 4

PROBLEM DEFINITION AND SCOPE

4.1 PROBLEM STATEMENT

Hadoop add-on API for Advanced Content Based Search & Retrieval

4.1.1 Goals and objectives

Goals:

Current Systems Focus on Search by Title,Author,etc which Is time consuming and finding relevant content from those documents is tedious task. So there is a need of such a system which shall find the relevant contents to the end user

Objective:

To find the relevant content from the huge number of PDF files present on Hadoop Distributed File System

4.1.2 Statement of scope

This system shall retrieve the required contents of files which are in an unstructured format containing huge amount of Data like e-books in a digital library where the number of books are present in thousands. The scope here is initially limited to PDF files which may be expanded to other unstructured formats like ePUB, mobi.

The input is provided to the system API in the form of search query which will be firstly filtered to find the important expression as a query to the API which will return the important content to the user in the form of paragraph or text highlighted using the power of distributed computing.

The particular page or the Entire book itself can be downloaded by the Library users if the content is satisfied else the search continues for finding relevant content

4.2 SOFTWARE CONTEXT

The end product of this project is the API for the distributed system having hadoop distributed system which would have enhanced abilities in searching the relevant contents inside big data of documents in a very rapid manner by completely harnessing the power of distributed computing and searching algorithms.This API shall be provided as an AddOn feature on top of Hadoop specially for content based retrieval purposes which would mainly deal with unstructured big data like PDF files.

4.3 MAJOR CONSTRAINTS

1. The first constraint is the that API used on a standalone system would not provide good performance required for content based retrieval system developed by the developer.

2. A pseudo distributed system deployed using different number of virtual machines would almost provide the similar performance as that of standalone systems depending on the number of virtual machine instances running simultaneously and the amount of memory allocated to each instance.Here care must be taken that guest nodes should not become a bottleneck to the hosted master node.

3. The best and suggested recomendation is that the API should be used and deployed on a fully-distributed system of nodes each connected to each other having independent memory ,disk and processor which will perform the best and provide expected results.

4.4 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

1. Single Mode When a cluster with only a single node is used for implementing the content based retrieval system,slower performance is expected due to the less amount of memory and processing power.

2. Pseudo Distributed Mode When a system is implemented either with number of virtual machines the different daemons as different process then the system is expected to perform better

3. Fully Distributed Mode When a cluster has number of physical nodes and hadoop daemons as different java processes with greater than 10 or more nodes in a single rack then the expected results are expected to be more better due to the presence of large storage and processing power.

4.5 SCENARIO IN WHICH MULTI-CORE, EMBEDDED AND DISTRIBUTED COMPUTING USED

The trends are nowadays shifting from standalone single node applications to distributed applications where in Java Frameworks like Hadoop help the applications to utlize the power of distributed computing wherein each and every node is assigned a task and the result is obtained to the master node which assigns the tasks to the slave nodes.

4.6 OUTCOME

1. The system is expected to provide the content based on the keywords entered in the form of keywords and the output shall be in the form of any of the following formats:

2. Entire Document

3. Content and the required paragraphs surrounding it

4. Only Paragraph itself

5. Only Particular Page itself and many such options shall be provided in the API.

4.7 APPLICATIONS

- Content based search and retrieval on

    1. Digital library books

2. IEEE Papers
3. Private Authorities
4. Government Authorities
5. Unstructured text files

- This API can be used to develop above functionality on platforms like

1. Web Application
2. Android Application
3. Standalone Application
4. Command Line Interface Application

4.8 HARDWARE RESOURCES REQUIRED

- Memory require for Hadoop installation and HBase and Map Reduce components.
- API requires minimum 100 MB space as it contains core components for content based retrieval
- 4 GB Primary Memory / RAM
- Intel i3/i5/i7 64bit or AMD Processors
- SSH Authentication to communicate with Namenode and Datanodes

4.9 SOFTWARE RESOURCES REQUIRED

- Any Open Source Linux Distribution.(Ubuntu Server version 14.04.2 Preferred)
- OPENSSH installed on each machine with public key of each node in authorized_keys directory along with the localhost.
- JDK Version 1.7 or above

- JAVA_HOME to be appended in the $PATH environment variable
- Hostname to be initialized for each node in the /etc/hosts file having a mastern-ode (namenode), secondary namenode and various slave nodes to be added.
- $HADOOP_HOME environment variable path to added in the /.bashrc
- Following Files to be configured in $HADOOP_HOME/etc/hadoop directory

CHAPTER 5

PROJECT PLAN

5.1 PROJECT ESTIMATES

5.1.1 Reconciled Estimates

Reconciliation is the method of bringing together all of the data and analyses into one final estimate of value. Hence following is the total data which we have reconciled and given approx metric of all the factors.

5.1.1.1 Cost Estimate

The cost of project includes both hardware and software.

Hardware:

The total hardware cost includes the price of systems with following configuration:

- Intel Core i3 processor
- 4 GB RAM
- 200 GB of Hard Disk
- NIC Card
- Ethernet Cable
- Wireless Router

So , the total cost will be around Rs. 80,000/-

Software:

There is no software cost as such as all the software used for this project is open source.

5.1.1.2 Time Estimates

The total time required is 3 months for developing the Hadoop API.

5.1.2 Project Resources

Hardware Resources

- There shall be N number of nodes each having the following hardware configuration in a fully distributed system connected environment together in a wired manner using Ethernet interface.
- Intel i3/i5/i7 processor
- Ethernet Interface on NIC
- Virtualization of upto 2 or 3 nodes on a single machine for testing purposes
- 2 or 4 GB RAM (and upto 512 MB for each node for pseudo disrtibuted)

Software Resources

- Any Open Source Linux Distribution.(Ubuntu Server version 14.04.2 Preferred)
- OPENSSH installed on each machine with public key of each node in authorized_keys directory along with the localhost.
- JDK Version 1.7 or above
- JAVA_HOME to be appended in the $PATH environment variable
- Hostname to be initialized for each node in the /etc/hosts file having a masternode (namenode), secondary namenode and various slave nodes to be added.
- $HADOOP_HOME environment variable path to added in the /.bashrc
- Following Files to be configured in $HADOOP_HOME/etc/hadoop directory

1. core-site.xml
2. hadoop-env.sh
3. yarn-site.xml
4. mapred-site.xml
5. master
6. slave

- Master node with Ubuntu Workstation version preferred for supporting Eclipse IDE with Hadoop plugin for development and Server having Ubuntu Server version Preferred.

Tools

- Eclipse IDE
- Hadoop Eclipse Plugin
- Java 1.7
- Hadoop Java Libraries.

5.2 PROJECT SCHEDULE

5.2.1 Project task set

Major Tasks in the Project stages are:

- Create distributed system with Hadoop namenodes and datanodes
- Develop API methods for extracting keywords from input query
- Test API methods for finding keywords from input query
- Develop API methods to perform keyword search on HDFS data using Map Reduce operations

- Test API methods to perform keyword search on HDFS data using Map Reduce operations
- Design database schema for indexing each and every document in HDFS
- Schema validation for HBase indexing database
- Develop background service for indexing newly added data to HBase database automatically
- Test background service for indexing newly added data to HBase database is working automatically or not
- Integrating all modules
- Integration testing

5.2.2 Task network

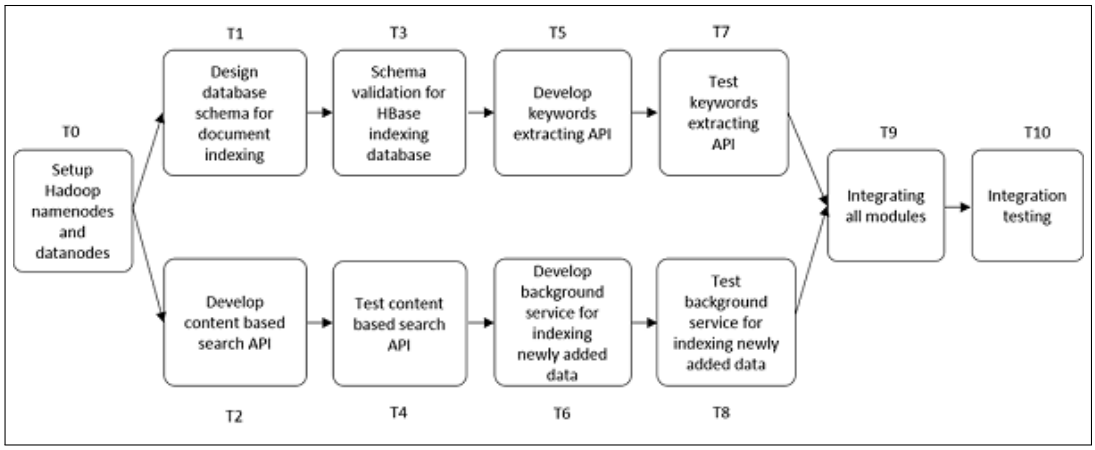


Figure 5.1: Task network

5.2.3 Timeline Chart

5.3 TEAM ORGANIZATION

For this project team is of 4 developers.

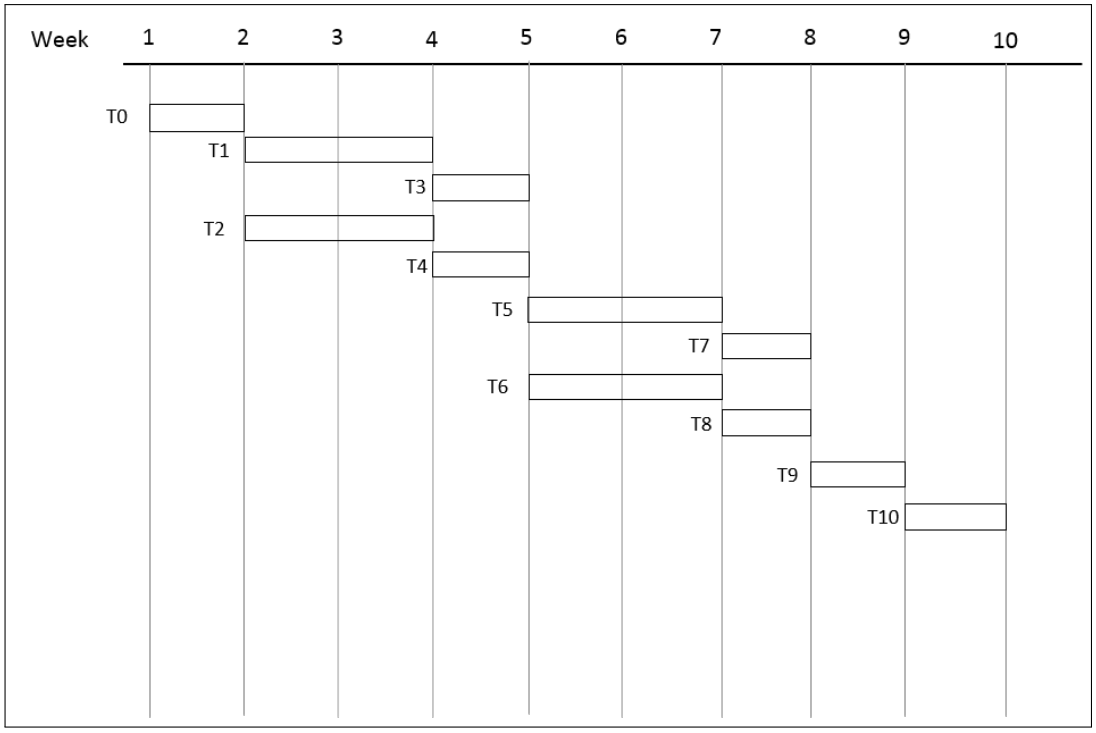


Figure 5.2: Timeline Chart

Team Member	Roles
Kshama Jain	Literature Survey and Solution Analysis
Aditya Kamble	Software API Development and Testing
Rahul Rao	System Management and API Development
Siddhesh Palande	Integration and Testing

Table 5.1: Team Structure

5.3.1 Team structure

The team structure for the project is of 4 people . Roles are defined as mentioned above

5.3.2 Management reporting and communication

The communication and reporting of work is done on weekly basis depending upon the time which is around 4hrs for a week and since they are sharing the same premises the communication is also excellent

CHAPTER 6

SOFTWARE REQUIREMENT SPECIFICATION (SRS IS TO BE PREPARED USING RELEVANT MATHEMATICS DERIVED AND SOFTWARE ENGG. INDICATORS IN ANNEX A AND B)

6.1 INTRODUCTION

6.1.1 Purpose and Scope of Document

This system shall retrieve the required contents of files which are in an unstructured format containing huge amount of Data like e-books in a digital library where the number of books are present in thousands. The scope here is initially limited to PDF files which may be expanded to other unstructured formats like ePUB, mobi.

The input is provided to the system API in the form of search query which will be firstly filtered to find the important expression as a query to the API which will return the important content to the user in the form of paragraph or text highlighted using the power of distributed computing.

The particular page or the Entire book itself can be downloaded by the Library users if the content is satisfied else the search continues for finding relevant content.

6.1.2 Overview of responsibilities of Developer

1. The developer should have the knowledge of Hadoop.
2. The developer should have the knowledge of Mapreduce and HBase.
3. The developer should have the knowledge of Java.

6.2 USAGE SCENARIO

This section provides various usage scenarios for the system to be developed.

6.2.1 User profiles

There will be number of users who will use the same application which will be provided by the developer. As the user search for query in the HDFS of Distributed Hadoop System, system will return the result with expected outcome.

6.2.2 Use-cases

All use-cases for the software are presented.

Sr No.	Use Case	Description	Actors	Assumptions
1	Use case 1	User	Add Document	User add document to HDFS system using API Insert user added document to HDFS and scan for keywords to store in metadata.
2	Use case 2	Administrator	Setup Hadoop cluster.	Hadoop cluster will work with our addon API for content based retrieval.

Table 6.1: Use Cases

6.2.3 Use Case View

Use Case Diagram. Example is given below

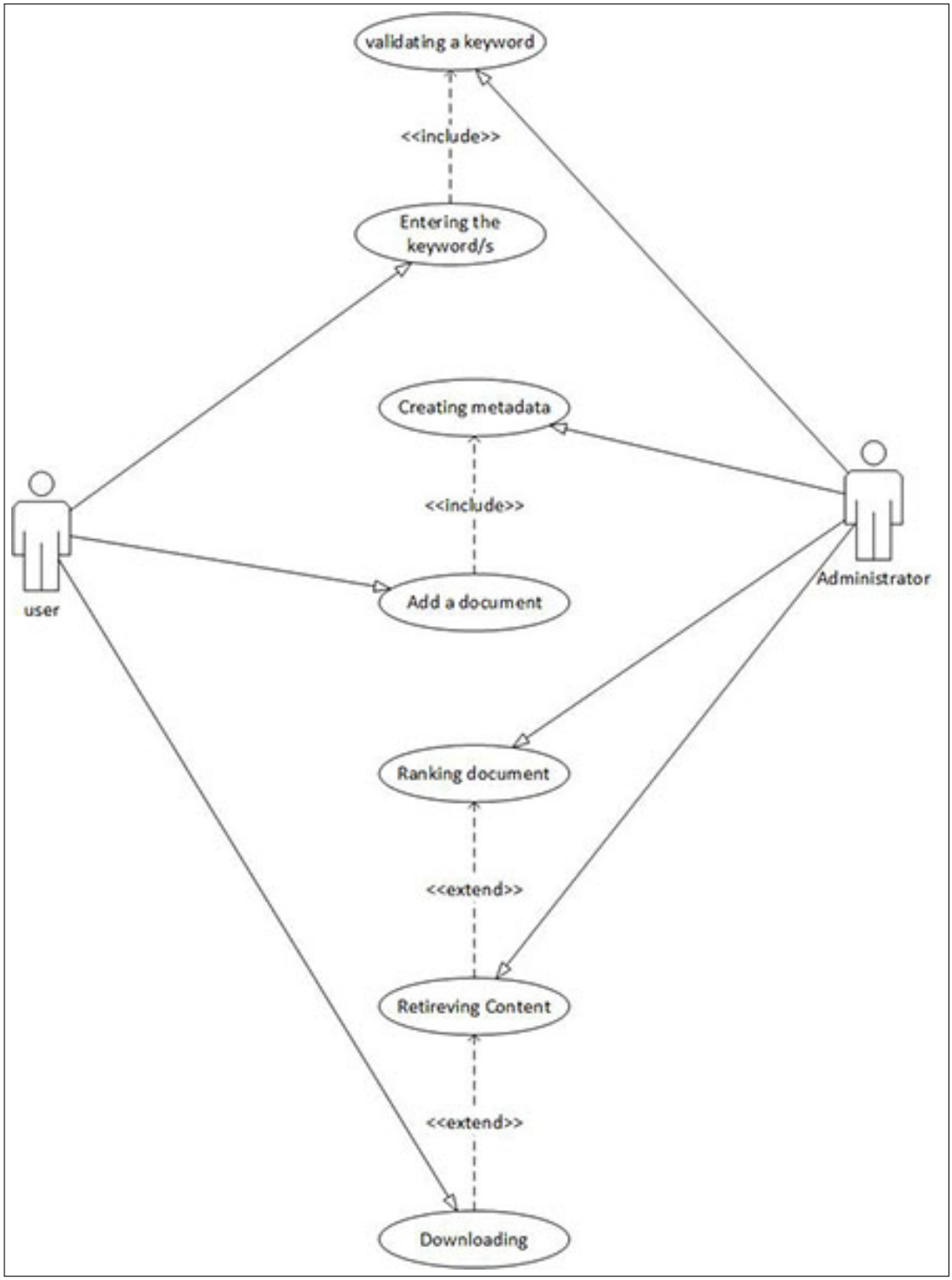


Figure 6.1: Use case diagram

6.3 DATA MODEL AND DESCRIPTION

6.3.1 Data Description

In this project we are searching and retrieving information from data inside HDFS. Hadoop Map Reduce operations will be used to perform key value pair generation and depend upon result information is searched.

Here, data is documents in PDF (Portable Document Format) format. These documents are stored on HDFS - Hadoop Distributed File System. These documents are considered as raw data. These documents can have properties like

- Name
- Size
- Date
- Author

Document meta data is also maintain in HBase distributed database. It has attributes like

- Content Keywords
- Index Keyword
- Document Keywords

This information will be useful to filter documents before performing content based searching operations.

6.3.2 Data objects and Relationships

Use Case Diagram. Example is given below

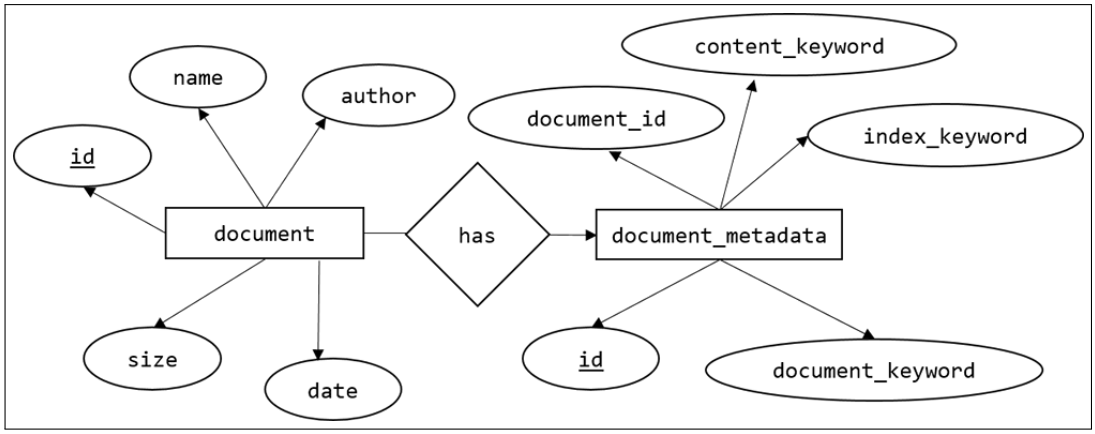


Figure 6.2: Data Model

6.4 FUNCTIONAL MODEL AND DESCRIPTION

A description of each major software function, along with data flow (structured analysis) or class hierarchy (Analysis Class diagram with class description for object oriented system) is presented.

6.4.1 Data Flow Diagram

Use Case Diagram. Example is given below

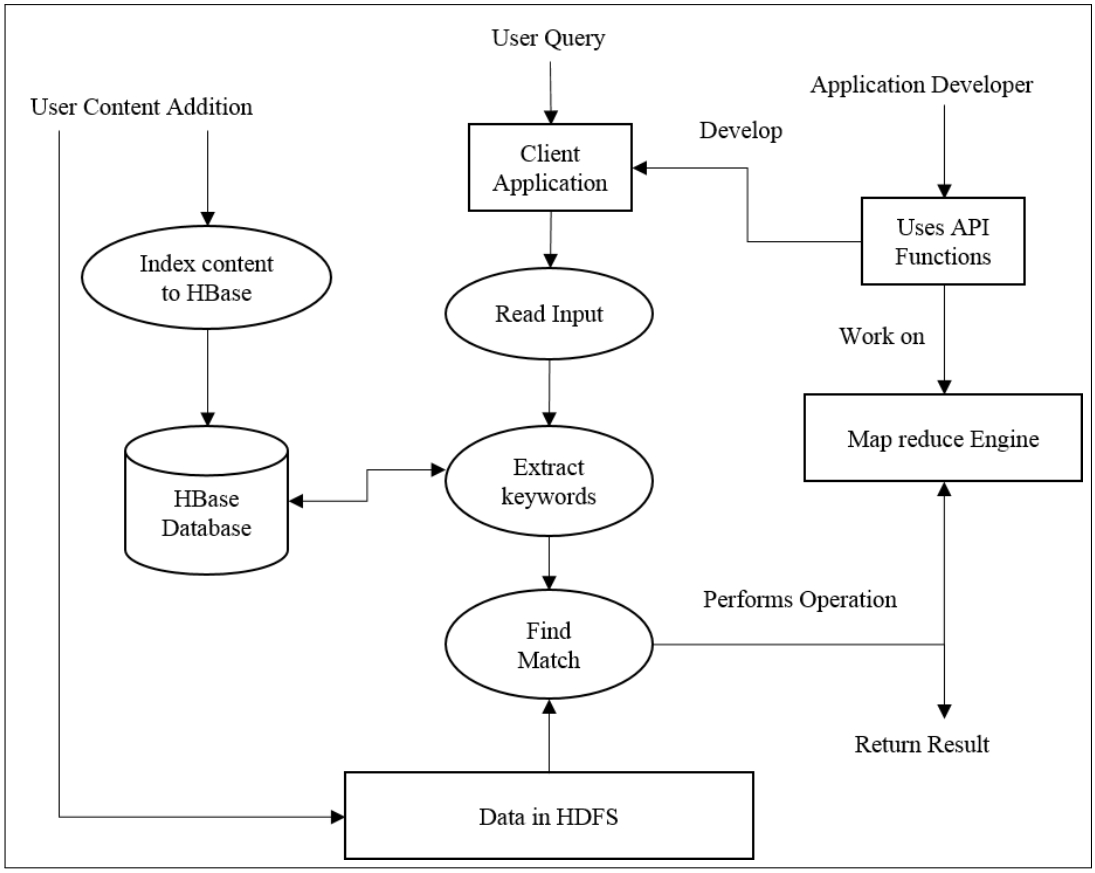


Figure 6.3: Data Flow Diagram

6.4.2 Description of functions

A description of each software function is presented. A processing narrative for function n is presented

6.4.3 Activity Diagram:

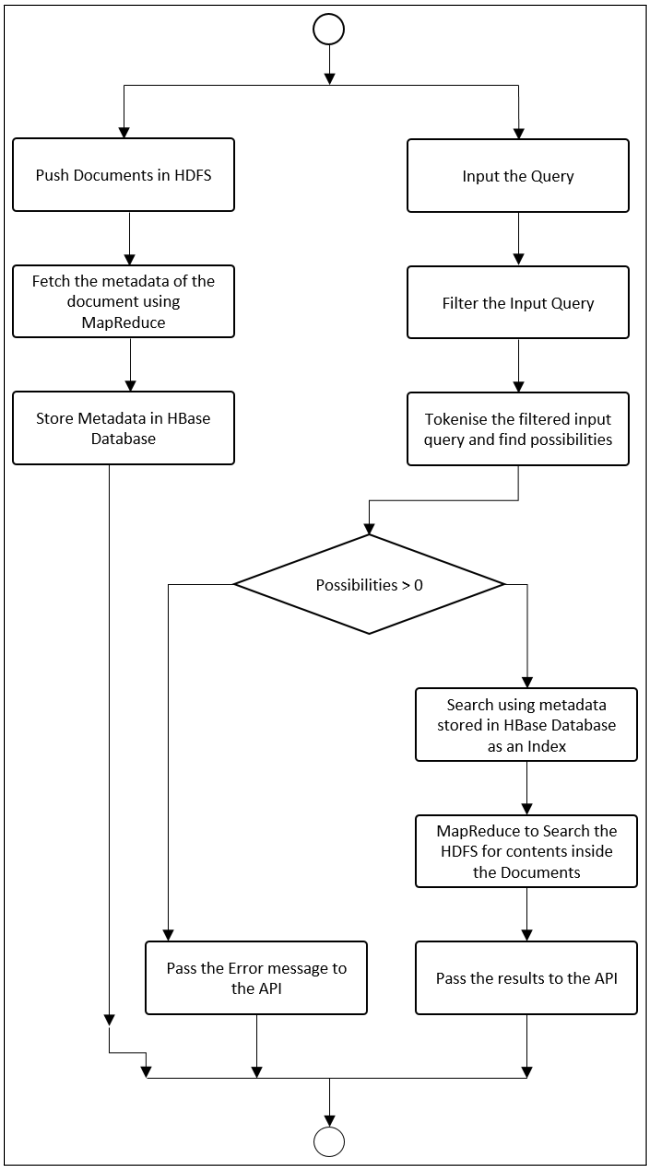


Figure 6.4: Activity diagram

6.4.4 Non Functional Requirements:

Performance Requirements

Memory Requirements:

- Memory require for Hadoop installation and HBase and Map Reduce components.

- API requires minimum 100 MB space as it contains core components for content based retrieval
- 4 GB Primary Memory / RAM

Speed Requirements:

- Intel i3/i5/i7 64bit or AMD Processors

Security Requirements

- SSH Authentication to communicate with Namenode and Datanodes

Software Quality Attributes

Software Quality can be defined as the conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

Software Quality Attributes are

1. **Functionality**

   This is an ability by which the software satisfies the needs of the software denoted by suitability, accuracy, interoperability, compliance and security.

2. **Reliability**

   Due to wired connectivity, reliability can be guaranteed.

3. **Availability**

   The system should be available during their respected hours.

4. **Usability**

   This ability indicates that the usefulness of the software.

5. **Efficiency**

   This indicates the measure of computing resources and time required by the program to perform.

6. **Maintainability**

   The ability required to locate or fix bugs in software. There should be facility to add or delete or update documents.

7. **Portability**

   The software works properly even if the environment gets changed (i.e. change in hardware or software).

8. **Reusability**

   With new versions of Hadoop this API can be improved using new features of Hadoop

6.4.5 State Diagram:

State diagram represents the various states that an object attains during its lifecycle in response to events. A state depicts a condition of an object and the activities of an object during its lifecycle.
We model the dynamic aspects of a system by using State diagrams. Therefore we create a state diagram for the objects that respond to events. The various constituents of a state diagram are:

- State machines
- Events
- Transitions

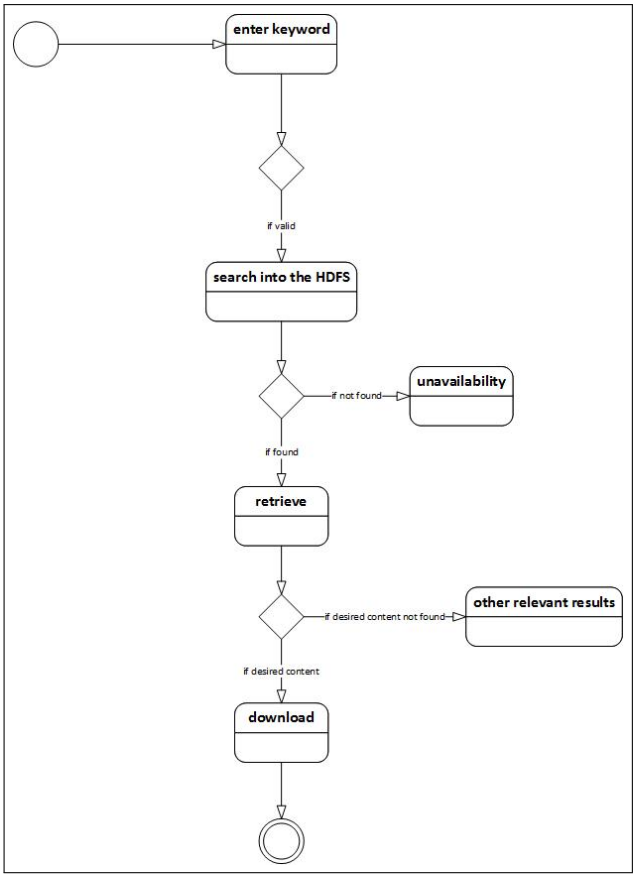


Figure 6.5: State transition diagram

6.4.6 Design Constraints

The primary design constraint is the distributed platform. Since the application is designated for distributed systems. Creating a Application Programming Interface which is both effective and easily usable will pose a difficult challenge. Other constraints such as limited memory and processing power are also worth considering.

6.4.7 Software Interface Description

- Any Open Source Linux Distribution.(Ubuntu Server version 14.04.2 Preferred)
- OPENSSH installed on each machine with public key of each node in authorized_keys directory along with the localhost.
- JDK Version 1.7 or above

- JAVA_HOME to be appended in the $PATH environment variable
- Hostname to be initialized for each node in the /etc/hosts file having a mastern-ode (namenode), secondary namenode and various slave nodes to be added.
- $HADOOP_HOME environment variable path to added in the /.bashrc
- Following Files to be configured in $HADOOP_HOME/etc/hadoop directory

    1. core-site.xml
    2. hadoop-env.sh
    3. yarn-site.xml
    4. mapred-site.xml
    5. master
    6. slave

- Master node with Ubuntu Workstation version preferred for supporting Eclipse IDE with Hadoop plugin for development and Server having Ubuntu Server version Preferred.

CHAPTER 7

DETAILED DESIGN DOCUMENT USING APPENDIX A AND B

7.1 INTRODUCTION

This document specifies the design that is used to solve the problem of Product.

7.2 ARCHITECTURAL DESIGN

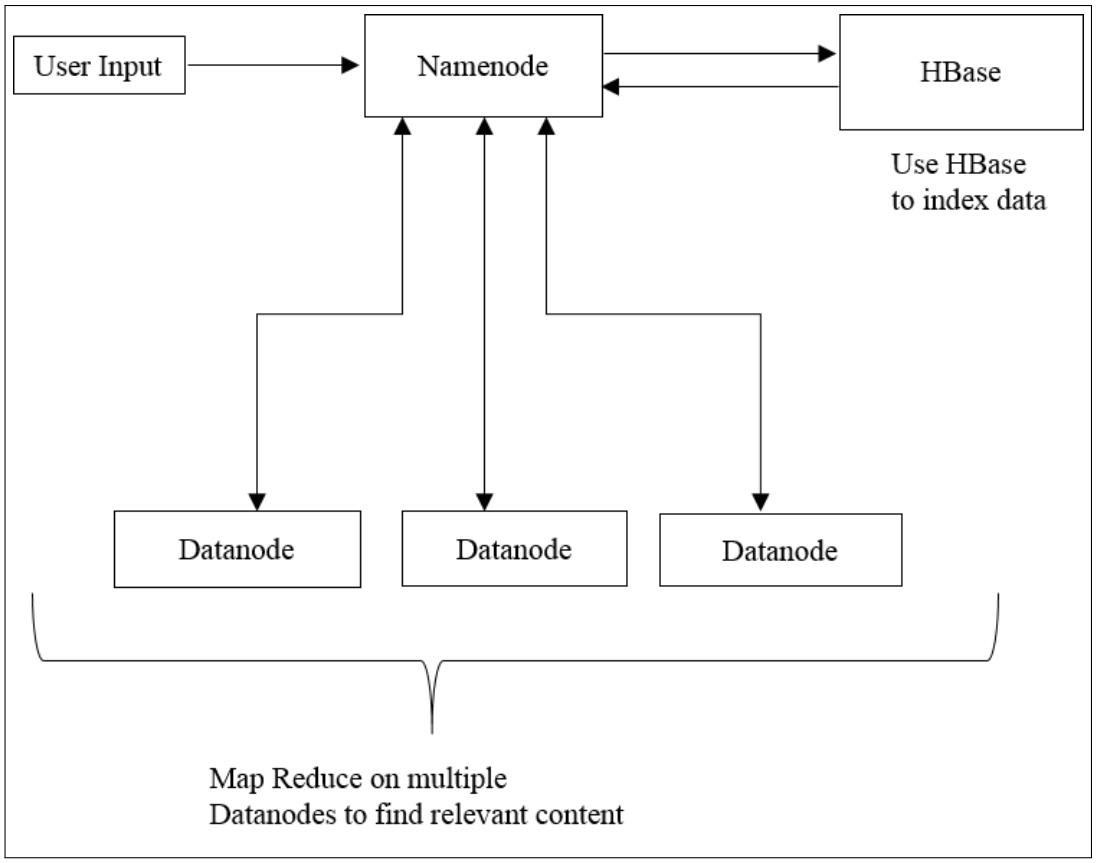


Figure 7.1: Architecture diagram

7.3 DATA DESIGN (USING APPENDICES A AND B)

7.3.1 Internal software data structure

We are using HashMap as internal datastructure since MapReduce uses it.
A HashMap is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value. It models the mathematical function abstraction. The Map interface includes methods for basic operations (such as put, get, remove, containsKey, containsValue, size, and empty), bulk operations (such as

putAll and clear), and collection views (such as keySet, entrySet, and values).
Creating Hashmap datastructure: Map¡String, Integer¿ m = new HashMap¡String, Integer¿();

The second argument is a conditional expression that has the effect of setting the frequency to one if the word has never been seen before or one more than its current value if the word has already been seen. Try running this program with the command: java Freq if it is to be it is up to me to delegate The program yields the following output. 8 distinct words: to=3, delegate=1, be=1, it=2, up=1, if=1, me=1, is=2 Suppose we prefer to see the frequency table in alphabetical order. All you have to do is change the implementation type of the Map from HashMap to TreeMap. Making this four-character change causes the program to generate the following output from the same command line. 8 distinct words: be=1, delegate=1, if=1, is=2, it=2, me=1, to=3, up=1 Similarly, we make the program print the frequency table in the order the words first appear on the command line simply by changing the implementation type of the map to LinkedHashMap. Doing so results in the following output. 8 distinct words: if=1, it=2, is=2, to=3, be=1, up=1, me=1, delegate=1

7.3.2 Global data structure

We are using Set interface as Global datastructure.
A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited. Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ. Two Set instances are equal if they contain the same elements.
The Java platform contains three general-purpose Set implementations: HashSet, TreeSet, and LinkedHashSet. HashSet, which stores its elements in a hash table, is the best-performing implementation; however it makes no guarantees concerning the order of iteration. TreeSet, which stores its elements in a red-black tree, orders

its elements based on their values; it is substantially slower than HashSet. LinkedHashSet, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set (insertion-order). LinkedHashSet spares its clients from the unspecified, generally chaotic ordering provided by HashSet at a cost that is only slightly higher.

7.3.3 Temporary data structure

Arrays as temporary datastructures:
An array is an object of reference type which contains a fixed number of components of the same type; the length of an array is immutable. Creating an instance of an array requires knowledge of the length and component type. Each component may be a primitive type (e.g. byte, int, or double), a reference type (e.g. String, Object, or java.nio.CharBuffer), or an array. Multi-dimensional arrays are really just arrays which contain components of array type.
Arrays are implemented in the Java virtual machine. The only methods on arrays are those inherited from Object. The length of an array is not part of its type; arrays have a length field which is accessible via java.lang.reflect.Array.getLength(). Reflection provides methods for accessing array types and array component types, creating new arrays, and retrieving and setting array component values.

7.3.4 Database description

Fields	Type
id	int
name	varchar(50)
author	varchar(50)
size	int
date	date

Table 7.1: Table - Document

Fields	Type
document_id	int
content_keyword	varchar(50)
index_keyword	varchar(50)
document_keyword	varchar(50)

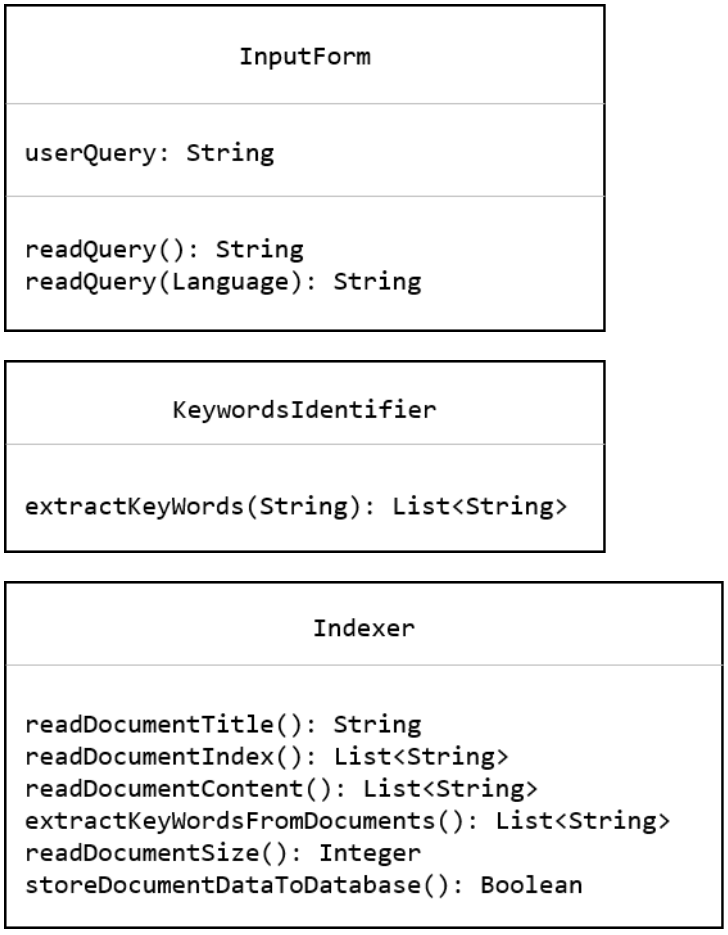
Table 7.2: Table - Document Metadata

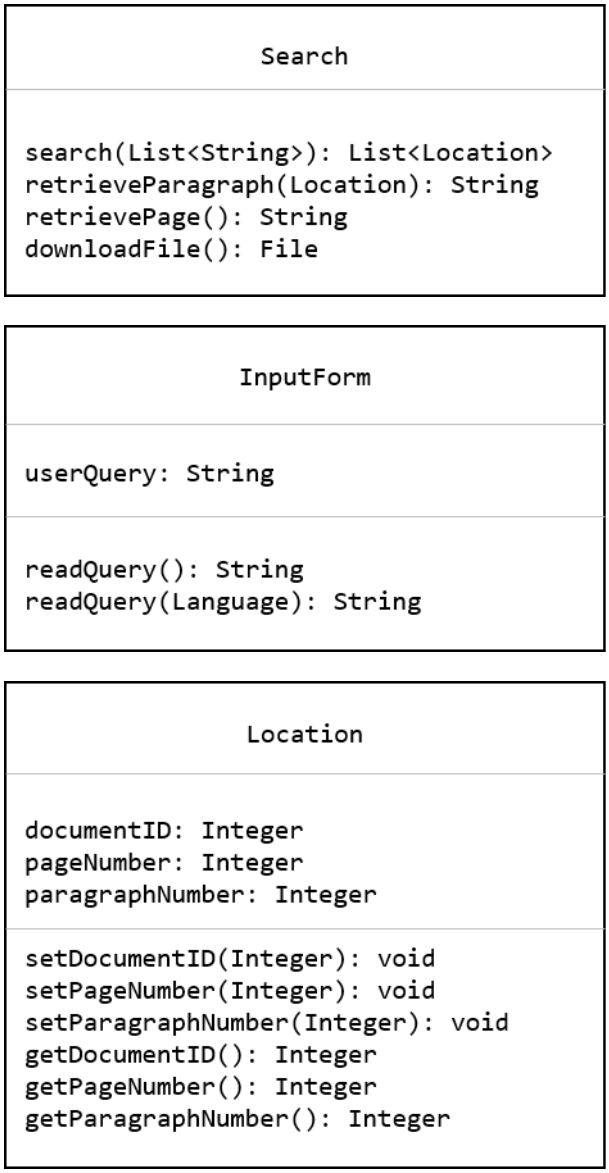
7.3.5 Database description

7.4 COMPOENT DESIGN

Class diagrams, Interaction Diagrams, Algorithms. Description of each component description required.

7.4.1 Class Diagram




Search
search(List<String>): List<Location>
retrieveParagraph(Location): String
retrievePage(): String
downloadFile(): File
InputForm
userQuery: String
readQuery(): String
readQuery(Language): String
Location
documentID: Integer
pageNumber: Integer
paragraphNumber: Integer
setDocumentID(Integer): void
setPageNumber(Integer): void
setParagraphNumber(Integer): void
getDocumentID(): Integer
getPageNumber(): Integer
getParagraphNumber(): Integer

CHAPTER 8

SUMMARY AND CONCLUSION

8.1 SUMMERY

Unstructured data like doc, pdf, accdb is lengthy to search and filter for desired information. We need to go through every file manually for finding information. It is very time consuming and frustrating. It doesnt need to be done this way if we can use high computing power to achieve much faster content retrieval. This addon will be able to provide APIs for different search results and able to download full file, part of files which are actually related to that topic. It will also provide API for context aware search results like most visited documents and much more relevant documents placed first so user work get simplified.

8.2 CONCLUSION

Thus we can use this API in Hadoop to reduce manual efforts and bring advance content based search and retrieval

CHAPTER 9

REFERENCES

[1] Lars George, ”HBase: The Definitive Guide”, 1st edition, O’Reilly Media, September 2011, ISBN 9781449396107

[2] Tom White, ”Hadoop: The Definitive Guide”, 1st edition, O’Reilly Media, June 2009, ISBN 9780596521974

[3] Apache Hadoop HDFS homepage http://hadoop.apache.org/hdfs/

[4] Mehul Nalin Vora ,”Hadoop-HBase for Large-Scale Data”,Innovation Labs,PERC,ICCNT Conference

[5] Yijun Bei,Zhen Lin,Chen Zhao,Xiaojun Zhu ,”HBase System-based Distributed Framework for Searching Large Graph Databases”,ICCNT Conference

[6] Seema Maitrey,C.K.”Handling Big Data Efficiently by using Map Reduce Technique”,ICCICT

[7] Maitrey S, Jha. An Integrated Approach for CURE Clustering using Map-Reduce Technique. In Proceedings of Elsevier,ISBN 978-81- 910691-6-3,2 nd August 2013.

ANNEXURE A

LABORATORY ASSIGNMENTS ON PROJECT ANALYSIS OF ALGORITHMIC DESIGN

To develop the problem under consideration and justify feasibilty using concepts of knowledge canvas and IDEA Matrix.

I	D	E	A
INCREASE: Accuracy of the data retrieved and processed	**DRIVE:** Interaction between developer and hadoop addon API	**EDUCATE:** The API can be used by libraries ,industries and government authorities to use hadoop for their data retrieval	**ACCELERATE:** Content based searching and Retrieval
IMPROVE: Relevant and speed	**DELIVER:** High Performance	**EVALUATE:** The collected information or data accurately	**ASSOCIATE:** More than one client
IGNORE: Media files	**DECREASE:** Manual Efforts	**ELIMINATE:** Go through every file manually for finding information.	**AVOID:** System failure

Figure A.1: IDEA Matrix

Project problem statement feasibility assessment using NP Hard NP complete or satisfiability issues using modern algebra and/or relevant mathematical models.

Input: A set of input keywords taken as a search query

Output: Question: Whether given recommendation is satisfiable or not?

P Class Problem :

Since the Application Programming Interface has a searching algorithm which works on map-reduce engine to retrieve the content from HDFS and return the result in the form of paragraph and with relevant contents with the link to download whole document or only a part of it.

NP Class :

The problem gets converted into NP Class when there is a verifier process to verify the content retrieved by the API .

ANNEXURE B

LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN

Use of divide and conquer strategies to exploit distributed/parallel/concurrent processing of the above to identify objects, morphisms, overloading in functions (if any), and functional relations and any other dependencies (as per requirements

Divide and Conquer: Divide and conquer (D and C) is an algorithm design paradigm based on multi-branched recursion. A divide and con- quer algorithm works by recursively breaking down a problem into two or more sub-problems of the same (or related) type (divide), until these become simple enough to be solved directly (conquer). The solutions to the sub-problems are then combined to give a solution to the original problem

Greedy Approach: A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of nding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

Dynamic Programming: Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler sub problems. It is applicable to problems exhibiting the properties of over- lapping sub problems and optimal sub structure. Dynamic programming algorithms are used for optimization (for example, nding the shortest path between two points, or the fastest way to multiply many matrices). A dynamic programming algorithm will examine the previously solved sub problems and will combine their solutions to give the best solution for the given problem. The alternatives are many, such as using a greedy algorithm, which picks the locally optimal choice at each branch in the road.

Brute Force: Brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each can-

didate satises the problems statement. A brute-force algorithm to nd the divisors of a natural number n would enumerate all integers from 1 to the square root of n, and check whether each of them divides n without remainder.

Branch and Bound: Branch and bound is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as general. A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root. The algorithm explores branches of this tree, which represent sub- sets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better so- lution than the best one found so far by the algorithm.

Conclusion: In our project we are using the Divide and Conquer Approach for solving the problem.

Use of above to draw functional dependency graphs and relevant Soft- ware modeling methods, techniques including UML diagrams or other necessities using appropriate tools.

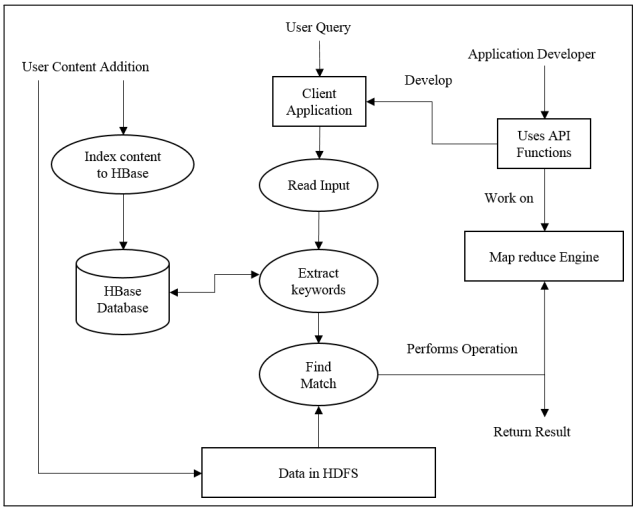


Figure B.1: Data Flow Diagram

Testing of project problem statement using generated test data (using mathematical models, GUI, Function testing principles, if any) selection and appropriate use of testing tools, testing of UML diagrams reliability

B.1 MATHEMATICAL MODEL

S = {s,e,x,y,DD,NDD,Mem-shared}

s= start state $->$ Taking input from the user as search query

e= End State $->$ return the output to the user in the form text based content

x = Input $->$ Search Query

y = Output $->$ Text Based Result

DD = Deterministic Data

1) Number of PDF Files

2) Keyword Tokenisation and Filteration

3) Number of DataNodes

4) Search Progress

5) Number of Results Obtained

NDD = Non Deterministic Data

1) Failure of Cluster Nodes

2) Communication Failure

Mem-Shared = Storage Space

1) HDFS will be distributed among a number of nodes in Hadoop Cluster and will share common FileSystem which will be managed by Hadoop

B.2 SCENARIOS TO BE TESTED

Instance number	Originating flow	Alternate flow	Expected result	Execution condition	State
1.	User enters keyword API checks the keyword for spelling errors, extra unwanted letters, unwanted words User entered a valid keyword API validates the keyword	None	User gets the desired content	User enters a valid keyword	Valid
2.	User enters keyword API checks the keyword for spelling errors, extra unwanted letters, unwanted words	User enters an invalid keyword API prompts to enter a valid keyword	User does not get the desired content	User enters a valid keyword	Invalid

Figure B.2: Test Case

Instance number	Originating flow	Alternate flow	Expected result	Execution condition	State
1.	User adds a document to the DB API checks for the .pdf extension file and corrupt file API creates a metadata for that	None	API successfully creates a metadata	User adds a .pdf extension file	Valid
2.	User adds a document to the HDFS API checks for the .pdf extension file	User adds a document other than .pdf extension API prompts to add a .pdf document to the HDFS	API fails to create a metadata for that document	User adds a .pdf extension file	Invalid

Figure B.3: Test Case

Instance number	Originating flow	Alternate flow	Expected result	Execution condition	State
1.	User downloads the desired content API search the keywords in documents stored in HBase and HDFS	None	User can download the content	API finds relevant content in the HDFS	Valid
2.	User downloads the desired content API search the keywords in documents stored in HBase and HDFS	API prompts to enter more keywords	API fails to find relevant document	API finds relevant content in the HDFS	Invalid

Figure B.4: Test Case

ANNEXURE C

PROJECT PLANNER

Major Tasks in the Project stages are:

- Create distributed system with Hadoop namenodes and datanodes
- Develop API methods for extracting keywords from input query
- Test API methods for finding keywords from input query
- Develop API methods to perform keyword search on HDFS data using Map Reduce operations
- Test API methods to perform keyword search on HDFS data using Map Reduce operations
- Design database schema for indexing each and every document in HDFS
- Schema validation for HBase indexing database
- Develop background service for indexing newly added data to HBase database automatically
- Test background service for indexing newly added data to HBase database is working automatically or not
- Integrating all modules
- Integration testing

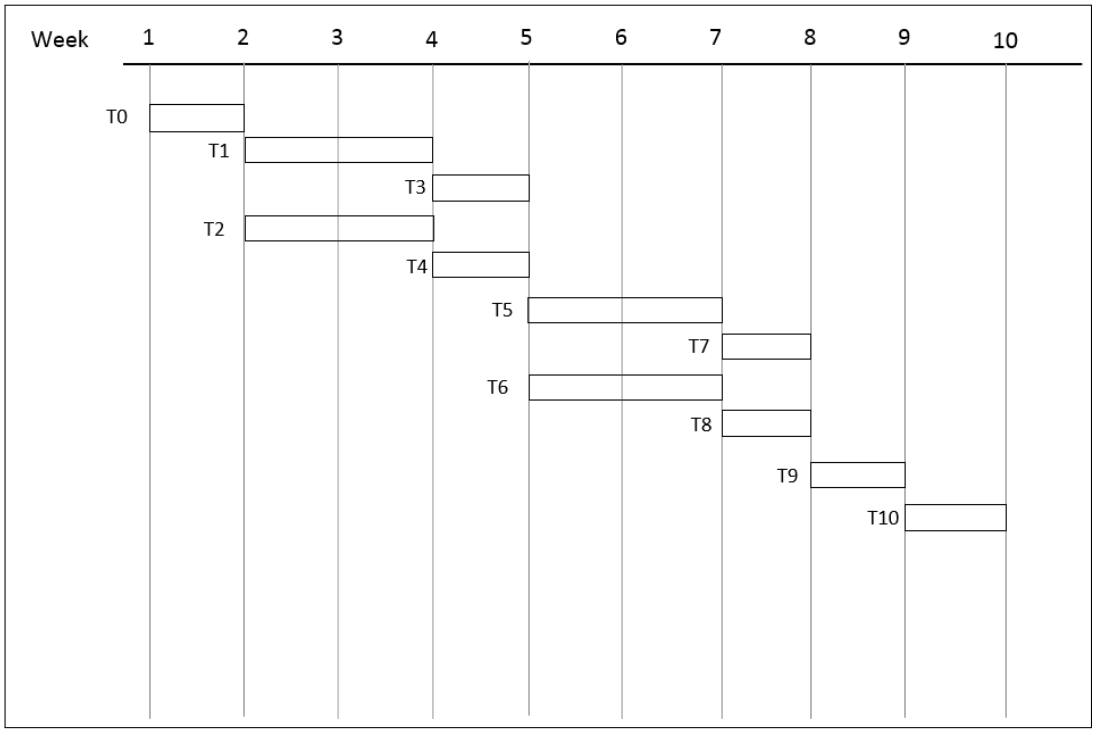


Figure C.1: Planning Chart

ANNEXURE D

REVIEWERS COMMENTS OF PAPER SUBMITTED

1. Paper Title: Hadoop add-on API for Advanced Content Based Search & Retrieval

2. Name of the Conference/Journal where paper submitted : IFERP - International Conference Institute for Engineering Research and Publication

3. Paper accepted/rejected : Accepted

4. Review comments by reviewer : The reviewers committee of NIER congratulate you for acceptance of your research paper for International Conference. You are cordially invited to convene the event by presenting your research paper through PowerPoint presentation.The Conference is being organized by NIER-India in association with ”Technoarete”

ANNEXURE E

PLAGIARISM REPORT

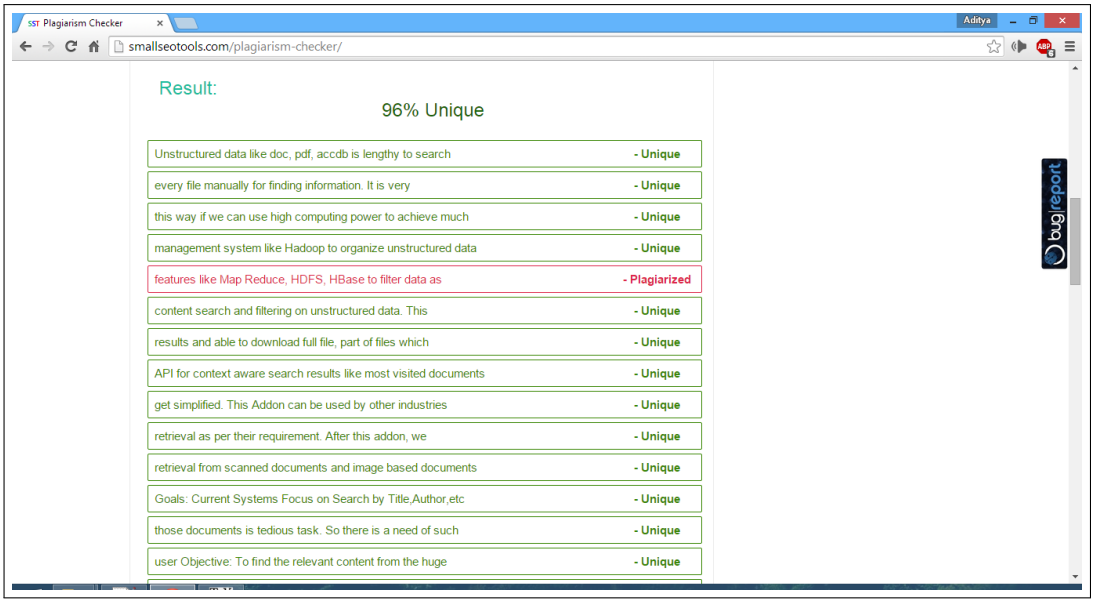


Figure E.1: Plagiarism Report