Data Analysis Using MapReduce Programming Model on the Cloud

Final Senior Project Report Submitted to

The Department of Computer Science and Engineering

Faculty of Engineering

Qatar University

In Partial Fulfillment of the Requirements for the Degree of

Bachelors of Engineering in

Computer Engineering

By

Amira Salah Eddine Ghenai	**(200657271)**
Farah AbdulMutaleb El-Qawasmi	**(200652758)**
Nadia Rashid Al-Okka	**(200653782)**

Spring 2010

©Copyright by Amira Ghenai, Farah El-Qawasmi, Nadia Al-Okka, 2010

DEDICATION

This work is dedicated to my parents who has been the wind beneath my wings until I completed this work. A big thank you to my sister, brother and aunt , words are never enough to express how much you have done for me. To my brilliant team mates who occupy a special place in my heart.   To all my friends for their patience, humor and advice. To my beloved country , university and last but not least, special thanks to all my respected professors.

Amira

This work is dedicated to my parents, brothers, and sister for their continues love , and for making me the person I am today. To my gorgeous team mates, my best friends in Doha and Gaza and to all my respected professors for their support and help.

Farah

This work is dedicated to my parents, for their continuous love, patience and support, for my lovely brothers for their help and support in every step, to my sisters for their love and care, to my best friend for being always around, to my nice and lovely team mates, to my all special friends, to all my wonderful professors for their never-ending support and to my great university for everything that it gave me.

Nadia

DECLARATION

This report has not been submitted for any other degree at this or any other University. It is solely the work of us except where cited in the text or the Acknowledgements page. It describes work carried out by us for the project described in the submitted senior Project Proposal which in available in the appendices of this report. We are aware of the penalties for plagiarism fabrications and unacknowledged syndication and declare that this report is free of these.

Farah El-Qawasmi	*signature*
Amira Ghenai	*signature*
Nadia Al-Okka	*signature*

Signed on

June. 10th .2010

ACKNOWLEDGEMENTS

First of all, we would like to thank Almighty GOD for giving us the strength to sustain all the stress and study pressure during our last senior year. Our sincere gratitude to the supervisor of this project Prof. Qutaibah Malluhi, for his guidance, help and motivation that helped us through the course of our journey towards producing this thesis. Apart from the subject of the project, We learnt a lot from him, which will be surely useful in different stages of our lives.

We wish to extend our heartfelt gratitude to all our Qatar University professors and instructors and especially to Dr. Sayed ElSayed, Dr. Abdelkarim Erradi, Dr. Ryan Riley and Mr. Zeyad Ali whose encouragement, guidance and support from the initial to the final level enabled us to develop an understanding in different aspects of the project.

We take this opportunity to convey our sincere thanks to Dr. Osama Shata, the Senior Project Coordinator, for guiding us during the two semesters and helping us in submitting the project in an organized, well presented and professional manner.

We would like also to thank the IT team of Qatar University: Mr. Sajeer Thavot, Mr. Ali Zahid and Mr. Shuja Ashfaq, for their assistance, criticisms and useful insights in debugging and solving encountered QU cloud issues.

Furthermore, our deepest appreciation to Mr. Alfredo Cappariello, the cloud computing software engineer from IBM Innovation Centre in Dublin, for his never-ending technical support regarding cloud concerns during the whole project.

In addition to that, we would like to thank Qatar National Research Fund (QNRF) for funding the project[1].

Lastly, we offer our heartiest regards and thanks to the Carnegie Mellon University staff for their help and great support and especially to Dr. Majd Sakr, Mr. Brian Gallew, previous system administrator, and Eng. Suhail Rehman.

We offer our deepest regards and blessings to all of those who supported us in any respect during the completion of the project.

[1] This publication was made possible by a grant from Qatar National Research Fund under its Undergraduate Research Experience Program. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of Qatar National Research Fund.

ABSTRACT

Cloud computing is an emerging paradigm of computing where virtualized information technology resources are dynamically provided as a scalable on-demand service delivered over Internet technologies. Qatar University (QU) has partnered with IBM, Carnegie Mellon University in Qatar (CMU-Q), and Texas A&M University at Qatar (TAMUQ) on an initiative that brings the first could computing system to the Middle East. This initiative leverages the IBM Blue Cloud solutions to provide a cloud infrastructure (called QLoud) that is open for local businesses and industries to implement and test relevant applications

In our project, this new computing model was leveraged to solve a real problem that is relevant to the education and research fields in Qatar. We took advantage of Qatar University cloud and we focused on analyzing a large data set.

Considering the limited scope of the project, the project focused on designing and implementing only one useful problem, which is indexing. Data indexing was processed efficiently by employing the MapReduce programming model on a large virtual cluster provisioned on the Qatar University cloud infrastructure. MapReduce enables fast distributed processing of results by employing two steps. The "Map" step divides the gigantic data into smaller chunks and distributes these chunks over a large number of worker nodes. Each worker node does processing on its chunk and passes the result to a master node. The master node performs the “Reduce” step by combining the pieces of results into a final answer. In our case, the MapReduce indexing MapReduce algorithm produces textual files, which are processed and used in a search application. A simple user friendly interface is designed to demonstrate this concept.

This project showed the effectiveness of the cloud computing model in improving search applications using the MapReduce indexing algorithm. In addition to that, MapReduce programming model has proven to be a powerful, clean abstraction for programmers. For all those reasons, the cloud technology can be used effectively for further research at QU. However, our experience in this project has demonstrated that this technology, in its current state, is not as easy and as seamless as it is advertised.

Furthermore, the project promotes collaboration between Qatar University, CMU-Q, IBM and other universities in Qatar. It has provided a unique educational experience for us by allowing us to interact with other institutions, and learn about an emerging leading-edge technology.

TABLE OF CONTENTS

ABBREVIATION

CMU-Q	Carnegie Mellon University at Qatar
GFS	Google File system
HDFS	Hadoop Data File System
IaaS	Infrastructure as a Service
OGPS	Oil and Gas Production Systems
PaaS	Platform as a Service
QNRF	Qatar National Research Fund
QP	Qatar Petroleum
QU	Qatar University
SANs	Storage Area Networks
SaaS	Software as a Service
SSH	Secure Shell
TAMUQ	Texas A&M University at Qatar
VPN	Virtual Private Network

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

This chapter is considered as an introduction to the whole project report. In this section, we discuss the project overview and the fields investigated in the project. Then, the detailed problem statement is elaborated on by focusing on the most important aspects of the problem and how it is interpreted and solved. Also the project targeted environment is explained and the scope of study is described.

1.2 Overview

If we look deeply around us, we will quickly notice that we are living is a "data age". The International Data Corporation IDC estimated the total universe storage to be 0.18 zettabytes in 2006 which let us think of how we are going to manage this huge amount of data and how are we going to process it. This problem will affect both individuals and organizations. To solve this issue, combining multiple hardware units has been proposed. However, two major points needed to be in consideration: first is to solve the hardware multiple points of failure and second is to combine the distributed tasks for processing the data. [1]

The above described issue was the main concern in our project. We elaborated the situation by a combination of different technologies from various fields:

First, the cloud computing concept is motivated by data demands. The infrastructure of cloud computing can automatically scale up to meet the requests of users by its virtualization and distributed system technology. In addition to that, it can provide redundancy and backup features to solve the hardware failure problem. For that, the cloud computing field is strongly involved in our project. The Qatar University Blue Cloud is employed to store the project large datasets (0.5 Terabyte is available to be used for the project purposes).

Second, the cloud is used as a proper distributed system platform to apply a parallel programming model which is utilized in our project; that is the "***MapReduce***" programming model. The ***MapReduce*** programming model is a model that solves the task distribution problem by the computation of the map and reduce methods and the in-between interface. [1]. In addition to that, ***MapReduce*** makes it easier for programmers to develop parallel processing algorithms and reduces the programming efforts. In our project a specific ***MapReduce*** algorithm is used, which is indexing.

Indexing is a common operation performed in web search engines for three main purposes: HTML parsing, morphological and language normalized analysis and large-scale indexing. In our project, this concept in used to build an index from a large set of documents. Using the ***MapReduce***-generate index makes it much faster to locate an item in a list of documents. For that purpose, another algorithm is appended and applied on the indexed file which is searching. The searching algorithm enables the user to search within the index file for a specific word.

The ***MapReduce*** model is implemented in the cloud using the ***Hadoop*** implementation developed by Apache. ***Hadoop*** includes a distributed file system, HDFS and a system for provisioning virtual ***Hadoop*** clusters over a large physical cluster called ***Hadoop*** On Demand (HOD). [2]. Implementing the ***MapReduce*** model on ***Hadoop*** enables one to exploit the massive parallelism provided by the cloud and provides a simple interface to a very complex and distributed computing infrastructure.[3].

The proposed project will demonstrate the effectiveness of the cloud computing model in dealing with large scale data. Our initial plan was serving the oil and gas industry by analyzing historical databases generated by Oil and Gas Production Systems. The alternative plan, after the first plan failed, was analyzing data generate through web crawling. Web crawling is following link pages collected already to pages that have not been collected yet and getting their files. Good open-source web crawlers are available and can be used to meet the needs of the project. The one used in our project to collect data is crawler4j. Although crawler4j was a fast web crawler, the network connection was not fast enough to meet the project time constraints and requirements (half terabyte was planned to be gathered). Finally, to get the project data, file/data generation code was designed and applied.

1.3 Problem Statement

1.3.1 Critical and Important Aspects of the Problem

Dealing with large sets of data was and still is a huge concern that developers and programmers need to pay attention to, to succeed in producing flourishing projects. A huge list of considerations should be in mind to process large datasets in a minimum amount of time and within a specific budget, which is a real challenge! Processing data within a specific time and budget is very critical for lots of applications: in scientific and engineering problems (geology, physics, molecular science…) , commercial applications (datamining, network video and multi-national corporations…) and user applications (image and video editing, games, 3D-animations…) [4]. The limitations for such demanding applications is seen from two different perspectives: hardware and software. First in hardware, processors hit a frequency wall of 4GHz in the 2000's (Figure 1.1) because of terminal Silicon limit (Figure 1.2).

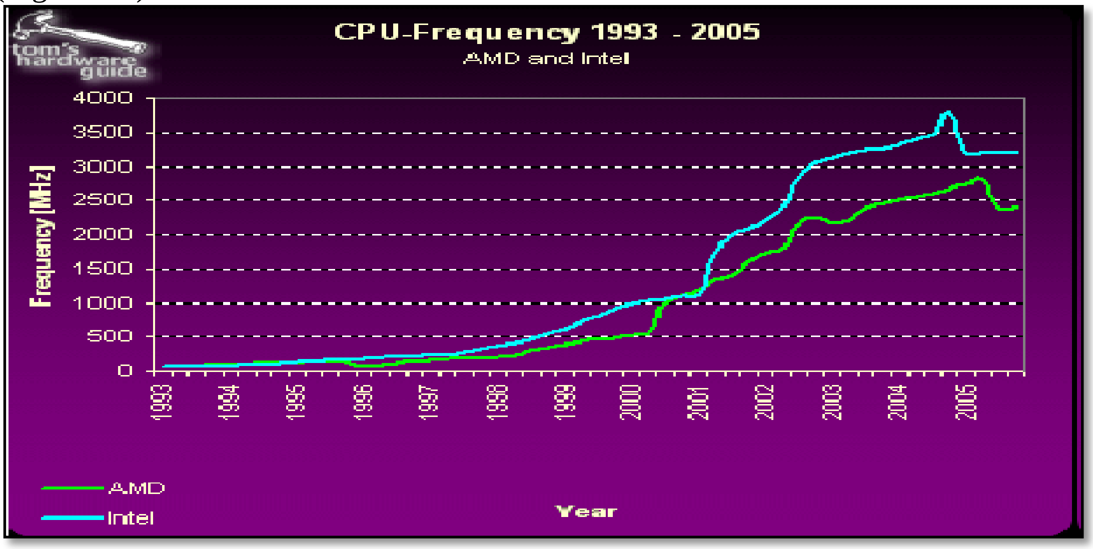


Figure 1.1: CPU Frequency with Respect to Time[4]

Figure 1. 2:CPU Limitations [4]

Second, the sequential programming model that executes instruction after the other is not a good solution in case of large scale data given the time limitation constraint.

In our project, we examine this problem and investigate one of the solutions which was a combination of various technologies that, at the end, solve the large dataset processing problem.

To overcome the hardware limits, a distributed system is used which is, in our case, provided through the cloud computing system: this way, we can enhance the performance through load distributing on multiple virtual machines and by this way we will eliminate the one physical processor limitations. In addition to that, we will gain additional features provided by the cloud system; backup, no single point of failure, add computing power on existing infrastructure, share applications over multiple machines, etc.[5].

Moving to the software side: the major key for improving the programming model is using the MapReduce functional programming model. The main goal of designing this specific model is to process large amounts of data using thousands of processors which would perfectly mach to our hardware design as we are using a distributed system “the cloud”. In addition, MapReduce would provide fault-tolerance, status and monitoring tools and clean abstraction for programmers. Even though Google was the one to develop the MapReduce model for processing web data, its implementation is proprietary. For that reason, The Apache project “Hadoop” is commonly used as an alternative in, for example, Facebook, Yahoo and also in our project [6].

In our study, we focus on providing an indexing application (as an example of MapReduce algorithm) for a user to search for a specific word from the indexed document that was produced by running a MapReduce program on the cloud for a large set of data. This data is located in the Hadoop distributed file system provided by the Hadoop system installed in the cloud. . [7].

In our study, we focus on providing an indexing application (as an example of MapReduce algorithm) for the client to search for a specific word from the indexed document that was produced by running a MapReduce program on the cloud for a large set of data. This data is located in the Hadoop distributed file system provided by the Hadoop system installed in the cloud.

1.3.2 The Targeted Environment

The project was first designed to develop and evaluate new techniques for applying pattern recognition and data mining algorithms on historical databases generated by Oil and Gas Production Systems (OGPS) to answer useful queries about trends and patterns of gas reservoirs and oil fields. This plan could not be achieved successfully due to the confidential nature of the target OGPS data.

The second plan was dealing with large set of text data and applying a MapReduce algorithm to generate and index of this data using cloud computing. With the new plan, the targeted environment of the project is conceived as a simple searching process supported by the indexing algorithm, which consists of a set of innovative and advanced tools and services (MapReduce, Cloud Computing, Hadoop…). Here, a user is presented with a simple interface in the form of search window: interacting with it results into finding out the files were the word most frequently occurs in a short period.

1.4 Goal and Main Objectives

The project goals can be summarized in six main points:

1. Understand the cloud computing technology and its environment, structure, platforms, applications and services.
2. Explore the MapReduce model, to be able to understand how the map and reduce functions are implemented.
3. Merge the knowledge about the cloud computing, MapReduce program model and Hadoop platform. This is considered a major project goal, since it allows applying data analysis techniques using MapReduce model on the cloud.
4. Investigate one of the most important applications of MapReduce model which is indexing. In addition to that, the project aims at designing the needed data structures, so optimal conditions are reached.
5. Apply the designed indexing algorithm over an intensive input dataset.
6. Design a search application with a friendly user interface. In this interface a user can enter a certain word to be searched, and it will return back a list of names of files where the word has highest occurrence. Considering that the result will be delivered using the generated index.

1.5 Scope

The original project scope was directed towards efficiently analyzing historical data repositories of oil and gas production systems using cloud computing. Data was planned to be offered from Qatar Petroleum (QP) to develop and evaluate new techniques for applying pattern recognition and data mining algorithms on historical databases generated by Oil and Gas Production Systems (OGPS) to answer useful queries about trends and patterns of gas reservoirs and oil fields.

However, this plan was not achieved successfully as it was not possible to get this confidential QP data within the timeframe of this project. Therefore, the project was redirected to a new scope that focuses on applying a MapReduce algorithm using cloud computing technique on Web-scale data. To be more specific, apply indexing and use MapReduce on a large set of data generated using a simple java code and perform a search operation on the indexed documents to let the user search for a specific word.

CHAPTER 2

FEASIBILITY STUDY & REVIEW OF RELATED LITERATURE

2.1 Introduction

The objective of this chapter is to give an overview about the main components of our project and to give an idea about the relation between our project and previous projects done in the same field. A literature review of the main components are discussed from different aspects such as history, structure, hardware, software, platforms and other related points needed to be clarified to understand the idea and the work flow of the project. The following sections in this chapter are going to discuss these main points.

2.2 Cloud Computing

2.2.1 Cloud Computing Concept

Cloud computing is an emerging Internet cloud based development with central remote servers to maintain data and applications. In other words, it is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet[7] (Figure 2.1). The resources may represent storage area networks (SANs), network equipment, firewall and other security devices. Cloud applications use large data centers and powerful servers that host Web applications and Web services. Anyone with a suitable Internet connection and a standard browser can access a cloud application.

Large corporation executives get benefit of cloud computing because it makes their life much more easier: instead of taking care of a large amount of computer devices and installing all the needed applications suite and licensed software, a better alternative is to install one application into all client end-devices which will link them to web-services where they can find all needed programs [8].

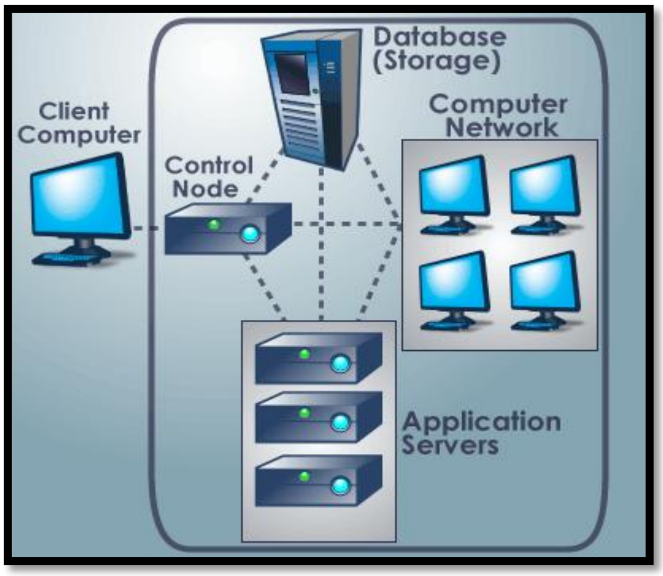
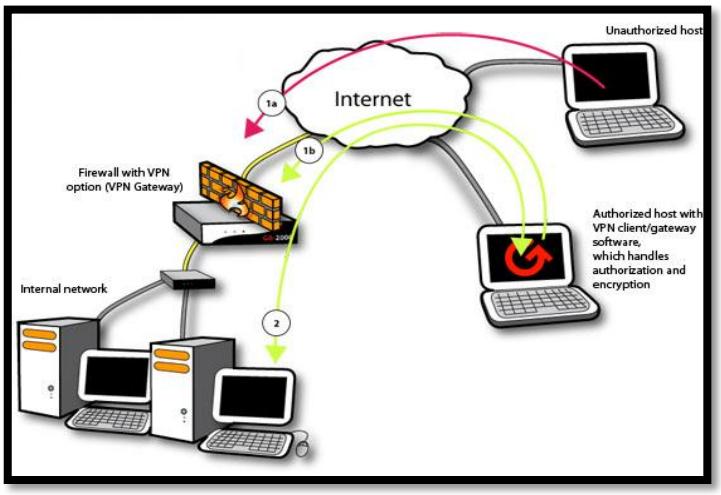


Figure 2.1: Cloud Computing Concept[8]

Cloud computing is a concept used in e-mail applications like Hotmail, Yahoo! Mail or Gmail: you log in to a Web e-mail account remotely. The software and storage for your account doesn't exist on your computer, it's on the service's computer cloud [8]

2.2.2 Cloud Computing History

The Cloud is a term borrowed from telephony field in the 1990s. Telephone companies where able to change wired fixed circuits which represented virtual private network (VPN) with virtual private network (cloud) keeping the same bandwidth which results in utilizing there bandwidth more efficiently (Figure 2.2)



Unauthorized host
Internet
1a
1b
Firewall with VPN option (VPN Gateway)
Authorized host with VPN client/gateway software, which handles authorization and encryption
Internal network
2

Figure 2.2: Virtual Private Network[9]

In 1999, Marc Andreessen was one of the first to attempt to commercialize cloud computing with an Infrastructure as a Service model. Then in 2000, Microsoft extended the concept of SaaS through the development of web services.Next in 2001, IBM detailed these concepts: it described advanced automation techniques such as self-monitoring, self-healing, self-configuring, and self-optimizing in the management of complex IT systems.

After that; in 2005, Amazon used cloud in its infrastructure which resulted in new features faster and easier and as a result, the concept of cloud computing was getting developed (Figure 2.3)

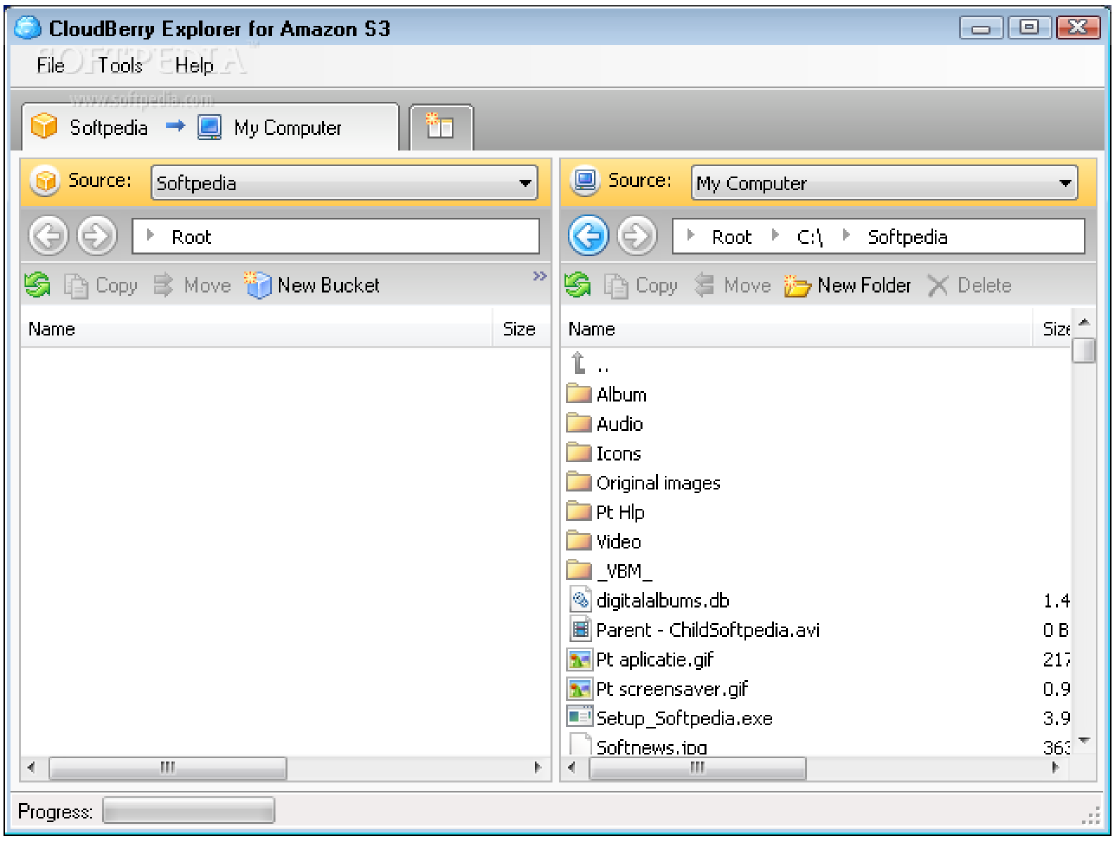


Figure 2.3: Amazon Cloud Explorer[10]

In 2007, Google, IBM, and a number of universities embarked on a large scale cloud computing research projects [7]

2.2.3 Cloud Computing Advantage

The unique cloud architecture could give the cloud a large number of unique benefits that other technologies could not possess. One of its advantages is that clients could access their data anytime, anywhere if only having net access. Data is not resided in the user's device or even an internal network which makes it much easier for the user.

Switching to the financial point of view, cloud computing brings hardware cost lower and saves IT support expenses: instead of having expensive large hardware requirements of each client including large memory and fast CPUs, a cheap terminal with a keyboard and a mouse would be sufficient to perform the same tasks using the cloud computing system. In addition to that, streamlined hardware would have fewer problems than a network of heterogeneous machines and operating systems and by this way, IT team do not have to worry about software issues as updates and will be free to concentrate more on innovation. Cloud computing will also provide faster time to market: companies will have the ability to deploy and scale apps in hours without changing the code ultimately which enables them to begin making a profit sooner.

Usually, servers and digital storage devices take huge space from the company residence which is considered a real trouble for small companies but with cloud computing, no more worries about the location. Cloud computing is letting the company decide of either locate the storage devices in its location or rent someone else's hardware and by this way, location problem is solved![8]

On the other hand, scientists and researchers work with calculations so complex that it would take years for individual computers to complete them. On cloud computing system, the client could send the calculation to the cloud for processing. The cloud system would tap into the processing power of all available computers on the back end, significantly speeding up the calculation which will result the output in much less time with additional services that include more security, redundancy and bandwidth.

2.2.4 Cloud Types

We have different ways of categorizing the cloud concept. One reasonable way is to divide it into public, private and hybrid cloud (Figure 2.5):

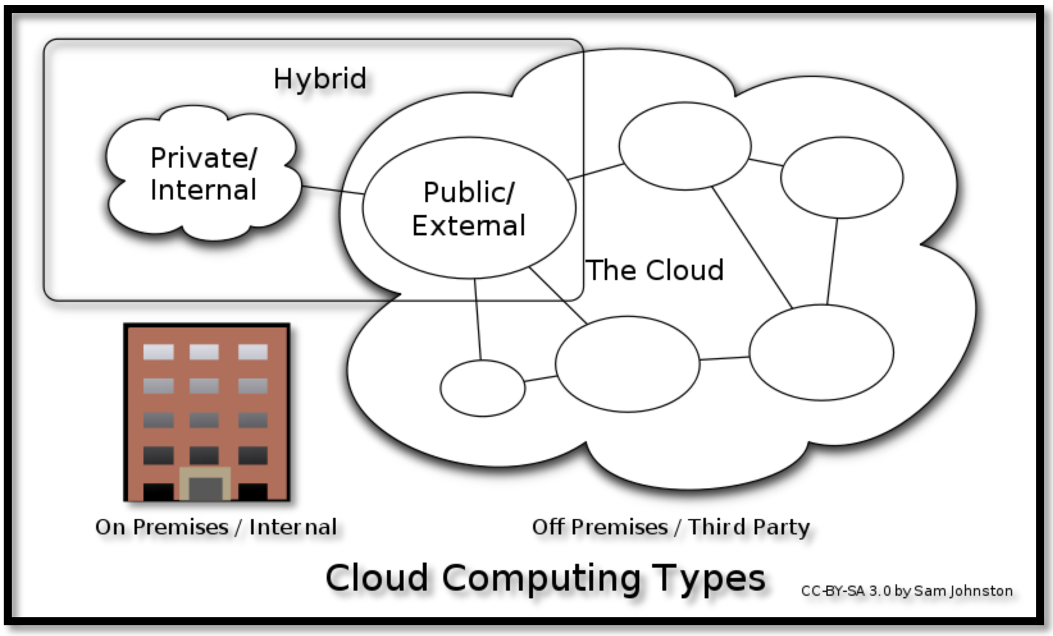


Figure 2.4: Cloud Computing Types[7]

Public cloud: *(external cloud*) It's the most traditional cloud environment that is located outside the company' boundaries. This service is offered as a 3rd party vendor and is provisioned on the Internet using web applications services (eg. Amazon EC2, Sun OCP, Google AppEngine).

Private cloud: (*internal cloud*) A cloud environment which creates a pool of resources within a company's firewall and includes resource management and dynamic allocation, chargeback and support for virtualization.

Hybrid cloud: (*mixed cloud*) A mixture of both private and public cloud. It is an environment in which external services are leveraged to extend or supplement the internal cloud. For instance, one of the virtualization environments which require servers is firewalls and spam filters. [8], [11]

2.2.5 High-level Architecture

The new cloud computing technique evaluated in 2009 [7] provides its unique services, delivered through data centers, relying on its architecture. The high virtualization technology architecture is mainly composed of three traditional layers from the hardware till the applications (Figure 2.4). The cloud layers are represented as follows: the infrastructure layer, the platform layer and the application layer.

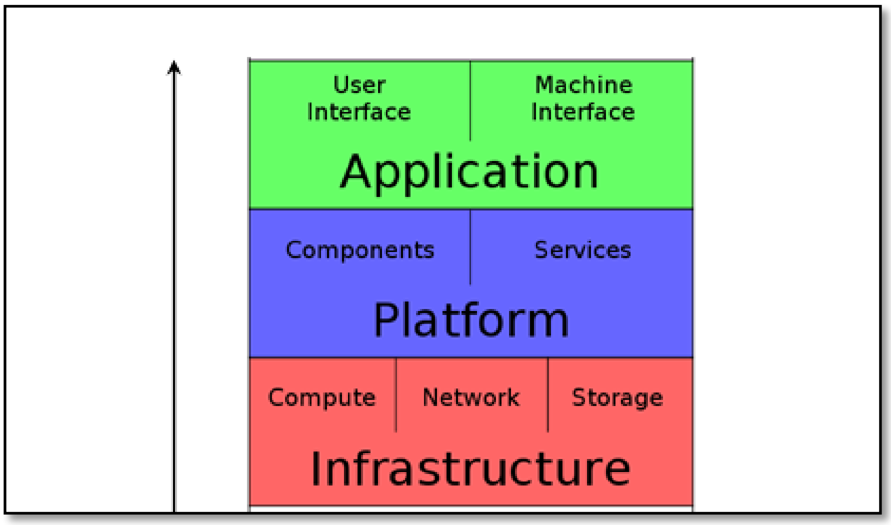


Figure 2.5: Cloud Computing Architecture[7]

2.2.5.1 Infrastructure layer (IaaS)

Starting with the infrastructure layer, it is considered as a platform virtualization environment and it delivers infrastructure as a service (IaaS). The IaaS service, in addition to the ability to scale, reduces the costs because you only need to pay for what you use: rather than purchasing servers, software, data center space or network equipment, clients instead buy those resources as a fully outsourced service in the cloud[12] .IaaS is divided into three main categories:

The first category is the compute category which consists of physical machines and virtual machines such as Amazon EC2, GoGrid and the OS-level virtualization [7]

The second category is the network category which provides network services that may be firewall or load balancing techniques. One of the network service examples provided by the network category is offering a private virtual network where customers can access the cloud over the network internet protocol security [13].

The third category is the storage category which identifies the amount of storage available and can be manipulated and managed by the clients.

2.2.5.2 Platform Layer (PaaS)

Moving now to the platform layer which is considered as the primary key for consuming the cloud infrastructure to support cloud applications [7]. The platform layer offers platform as a service (PaaS) which allows clients to run their own applications on the provided infrastructure delivered via Internet from the provider's servers. PaaS service offers workflow which helps in applications design, develop, and test stages. It is also beneficial in application services such as database integration, state management, team collaboration and much more [14]. We can divide the PaaS in two different perspectives: the PaaS producer and the PaaS consumer. First, the PaaS producer deals with integrating the OS, application software and service environment provided to the client. Second, the PaaS consumer how interacts with the offered services using API or GUI components [7].

2.2.5.3 Application Layer (SaaS)

Finally, the application layer that delivers software as a service is considered as a multitenant architecture model where a client can browse a single application provided by the cloud owner. The provider takes full responsibility of the application for the client on demand so the client does not need to alleviate the burden of software maintenance, ongoing operation, and support [7], [14]. Example of SaaS well known applications are: YouTube (web application) and DropBox (for storage purposes) [7]

2.3 Hadoop

2.3.1 Hadoop History

In 2000, Hadoop was created by Doug Cutting, who named it after his child's stuffed elephant. It was used as an open source to support distribution for the Nutch web search engine project which is a part of the Lucene Apache project.. Although he made a great amount of improvement , after indexing a few hundred million web pages, he realized he was a long way off from indexing the quickly growing billions of web pages on the internet.

In December 2004, Google File system (GFS) and MapReduce papers were published by Google Labs, which allows very huge amount of computations to be trivially parallelized across large clusters of servers. Cutting used that information from the paper and added the GFS and MapReduce implementation to Nutch using twenty nodes to run on.

In years 2006-2007, Cutting got a position in Yahoo company after seeing the Hadoop code, then a team of engineers worked on the software so tens of thousands of computers could be used to run them simultaneously, and researchers used that software as data mining tool.

As any new good developed program, word spread about it and by the beginning of year 2008, Amazon, Intel and Facebook were using Hadoop for many issues like log analysis and other things. Even Google got involves, initiating a project with IBM to offer major universities with clusters of some hundred computers so students could improve their techniques for parallel programming.[15]

2.3.2 Hadoop Definition

Hadoop is a Java software framework for running distributed applications on large clusters of commodity hardware. In the process application is divided into a number of small chunks of work because Hadoop implements a computational model called MapReduce, and each of the fragments may be executed or re-executed on any node in the cluster. In addition, Hadoop has its own distributed file system (HDFS) which stores data on the compute nodes and replicates data to multiple nodes to ensure if failure happened for data in a node, there are at least two other nodes from which to recover that piece of information. [11]

2.3.3 Why Hadoop is used

Having a huge unstructured data that comes from many sources and takes many types such as web logs, text files, sensor readings, text messages, audio, video and more. Dealing with this data needs many things as huge storage, reliability, tools to deal and analyze this data and supervise any failure could be occurred .All of these requires can be managed by the inexpensive Hadoop open source framework which is used on cross-platform operating system[16]

2.3.4 HDFS Architecture

HDFS has a master/slave architecture as presented in (Figure 2.6). An HDFS cluster contains only one NameNode that is a master server controls the file system namespace and regulate access to files by clients. Moreover, there are some DataNodes, usually divided one per node in the cluster. These DataNodes organize the storage space related to the nodes that they run on. HDFS represents a file system namespace and allows user data to be stored in files. Internally, a file is divided into one or more blocks which are stored in a set of DataNodes. The NameNode settles on the mapping of blocks to DataNodes , and executes file system namespace operations such as opening and closing files and directories. The DataNodes read and write requests from the file system’s clients, also perform block creation, deletion, and replication after getting the instruction from the NameNode.

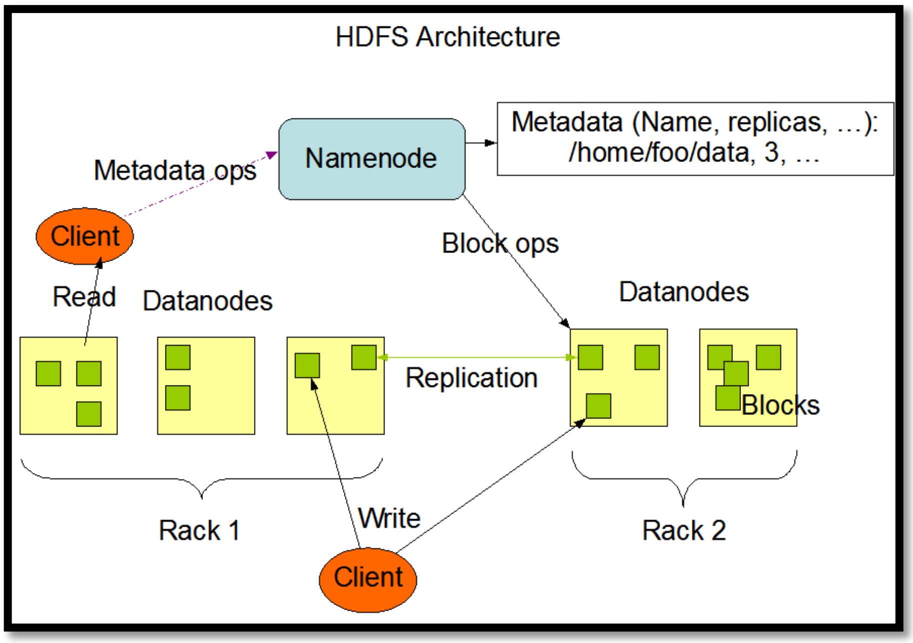


Figure 2.6: HDFS Architecture[17]

HDFS is built using the Java language. As a result of that, any machine that supports Java can run the NameNode or DataNode software. Whenever highly portable Java language is used ,then HDFS can be arranged on a wide range of machines. A usual deployment has a certain machine to run only the NameNode software, while other each machine in the cluster runs one instance of the DataNode software. The architecture does not prevent running multiple DataNodes on the same machine but it is unusual to run like this case. Having only one NameNode in a cluster simplifies the architecture of the system very much. The NameNode is the arbitrator and repository for all HDFS metadata, and all the system is designed so user data never flows through the NameNode[17]

2.4 MapReduce

2.4.1 MapReduce Overview

In the beginning of the twenty first century, many computations that are specialized to process large amounts of raw data as crawled documents, web request logs, etc… These computations were applied by authors and many others at Google. They computed different types of derived data, like inverted indices, summaries of the number of pages per host, various representations of the graph structure of web documents. The input data was usually large in the computations, so it has to be distributed among hundreds or thousands of machines to finish the process in a reasonable time. Although the computations were straightforward, the matters of how to parallelize the computation, distribute the data, and deal with the expected failures, made the original simple computation to be very complex specially with the huge amount of code to handle these issues.

As a result to this complexity a new model was designed to allow expression of the simple computation with hiding the complex details of data distribution, parallelization and fault-tolerance in a library. This model is inspired by the map and reduce primitive in Lisp which is the oldest high-level programming language. The use of this functional abstraction with user specified map and reduce operations enables automatic parallelization and distribution of huge computations, and achieves high performance on large clusters of PCs.[15]

2.4.2 MapReduce Programming Model

MapReduce is a programming model and a linked implementation for dealing out with many terabytes of data on thousands of machines. Computation obtains a set of input key/value pairs, and generates a set of output key/value pairs, so the user of the MapReduce library states the computation as two functions: Map and Reduce.

In the map reduce function the user gets the data from data sources like lines out of files, rows of a database, etc)and feeds them to the function as an input key/value pair (e.g.: filename, line). Then it generate a set of intermediate key/value pairs as shown in Figure 2.7. After that the library combined together these intermediate values related with the same intermediate key, and pass them to the Reduce method which accepts an intermediate key I and a set of values for that key and tries to merges together these values to form a possibly smaller set of values. From practice user can visualize that usually only zero or one final value will be produced per key as presented in (Figure 2.7)

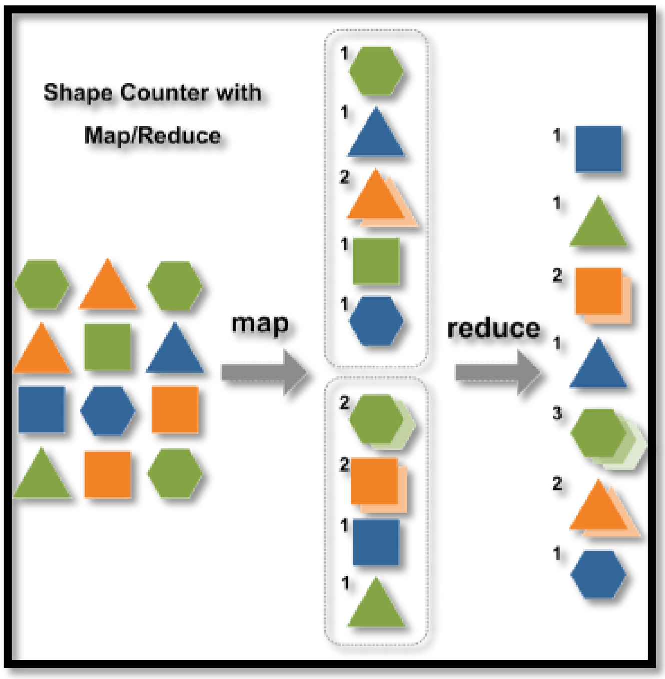
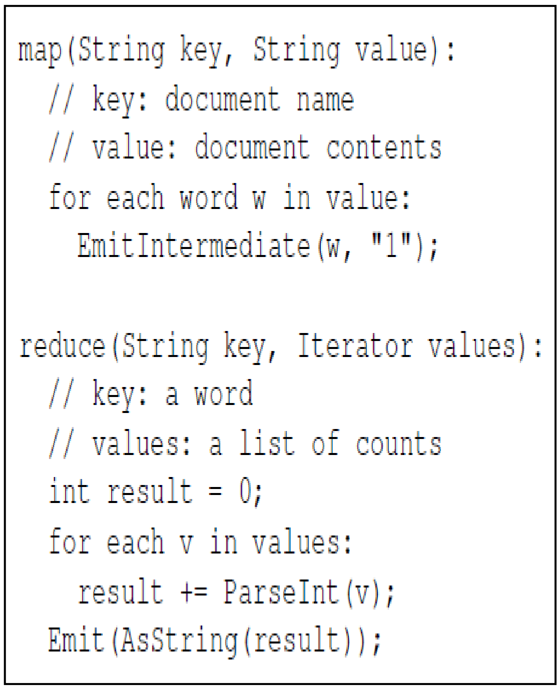


Figure 2.7: MapReduce[18]

Important thing which should be noticed that all the map( ) functions work in parallel to create different intermediate values from different input data sets, and the same for the reduce( ) functions which run in parallel so each one work on different output key.

Here is a simple example that could explain the functionality of MapReduce Model. In this problem the number of occurrences of each word will be accounted from a large collection of documents.

The map function emits each word and an related count of occurrences just like 1 in the simple pseudo-code shown in (Figure 2.8) .On the other hand the reduce function sums together all counts emitted for a certain word. In addition, another code is used by the user to fill in a MapReduce specification object with the names of the input and output files, the user then invokes the MapReduce function, passing it the specification object.



```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Figure 2.8: Simple Code of MapReduce[19]

As said before when MapReduce process is used, the data must be distributed among the different nodes, so the master program divvies up tasks depending on location of data where it tries to have map() tasks on same node as physical file data, or at least same rack.

Another thing that MapReduce module deals with the expected failures, where the master detects worker failures in the re-executes completed, in-progress map() tasks, and re-executes in-progress reduce() tasks. In addition, master notices particular input key/values that cause crashes in map(), and skips those values on re-execution.[19]

2.4.3 MapReduce Algorithms

There are many problems that can be mapped to a MapReduce program, such as: sorting, searching, indexing and classification. These programs must fit the features of the MapReduce algorithm. For any MapReduce algorithm, processing data must go through a map phase and a reduce phase. With consideration that the output of the map phase is the input to the reduce phase. [20]

2.4.4 Indexing using MapReduce

Indexing is the process of classifying and arranging a collection of data in such way to make it retrieved easily.[21]

Using the MapReduce programming model, usually, inverted indexing algorithms is used. Inverted indexing is referred to an index data structure that maps from content, a key, may be word, number or phrase, to a location in specific storage space.[22]

For map function, pairs of (file_name, content) are used as the input. Each word is emitted with the file name, so the output of this map function is pairs of (key_word, file_name). on the other hand, the reduce function takes(key_word, file_name) pairs as an input. Then, for each certain key(word), the reduce function make a list of files that this word is in. Finally, the output of this function is a pair of (key_word, List_of_files) [20]

2.5 Previous Work on Cloud

Amazon Elastic MapReduce: This is considered as a web service application which can be implemented in various fields such as business, research, and data analysis to process huge amount of data easily and cost effectively. Hadoop platform is used as a framework and is running on the web scale infrastructure of the Amazon cloud in which the infrastructure layer is composed of the Amazon EC2 virtual machines [23] and the platform layer consists of Amazon S3 which represents the simple storage service[24].

With Amazon Elastic MapReduce, one can perform data intensive tasks whatever the data size is to achieve a wide variety of applications such as web indexing, data mining, machine learning, log file analysis, financial analysis, bioinformatics research and scientific simulation. The benefit of choosing the Hadoop platform and implementing MapReduce Model on the Amazon cloud is: letting one interact with crunching or analyzing the data without worrying about time, management of Hadoop clusters, and cloud infrastructure which is provided by the virtualization technique.[23]

There are lots of similarities between the Amazon Elastic MapReduce application and the designed system in this project. First, the same main goal is targeted which is data analysis using MapReduce programming model on the cloud. Another point is the service type which is platform as a server PaaS: Amazon Elastic MapReduce uses Amazon S3 and the project's Paas is BD2 Server. They both use the infrastructure as a server IaaS also: Amazon Elastic MapReduce uses EC2 while the project's IassS is VMWare (XEN).

Even though both projects are very similar; there is one obvious dissimilarity that is the application query: in the Amazon Elastic MapReduce, we have more than one query (indexing, sorting…). However, the project uses one unique query which is indexing.

2.6 Crawler

2.6.1 Web Crawler Definition

A web crawler is a simple program that crawl to wide world web in a systematically way. It automatically passes through the Web. It is also called spider, robot, or wanderers. [25][26][27]

2.6.2 Why Web Crawlers are Used

Web crawlers or web spiders are used for many purposes. Mainly, they are used by the search engines to provide up-to-date information about the web pages, so they can process this data for faster search. Many other things can be accomplished by using web crawlers such:

1. Web site maintenance: they can be used to check the validation of the links related to this site. [25]
2. Gathering data: they can be programmed to download pages from the web. These pages can also be with some specifications. [25]
3. Searching for copyrights: for some companies, they can use crawlers to search for copyrights violations or infringements. [27]
4. Performing textual analysis: crawlers can be programmed in such a way to help in textual analysis such finding the most common words on the web, etc. [28]

2.6.3 How Web Crawlers Work

A crawler starts with the seeds, which are the URLs to visit. When the crawler visits a URL, it lists all the hyperlinks in the page and adds them to the crawl frontier, which is a list of URLs to visit. URLs from the frontier are recursively visited according to a set of policies. [25]

2.6.4 Open Source Web Crawlers

There are many open source web crawlers, each has its features. Such as the language used in programming, number of machines used to run, and data type needed to crawled. Choosing a crawler is depending on the purpose of using it. Some of the open source crawlers are:

1. Heritrix: It is an extensible, web-scale and distributed internet archive’s crawler. This crawler is designed for archiving periodic snapshots of a large portion of the Web. It was written in Java.[28]
2. Nutch: It is an Apache’s Open Source Search Engine. It is distributed and tested with 100M pages. This crawler is written in Java and released under an Apache License. It can be used in conjunction with the Lucene text-indexing package. [29]
3. WebSphinx: Originally, it is developed by Carnegie Mellon University. Now it is a web crawler Java class library. It can work on a single machine. This crawler has lots of problems and it is reported to be very slow. [29][30]

4. Crawler4j: is a fast crawler written in Java and released under an Apache License. It can be configured in a few minutes and is suitable for educational purpose. It can work on a single machine and it should easily scale to 20M pages. The best advantage is that it is very fast; it can crawl and process the whole English Wikipedia in 10 hours. [29]

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 Introduction

The goal of this chapter is to demonstrate the analysis and design of the system. You can find the system specification, hardware and software requirements. Moreover, system conceptual model, proposed solutions are explored. In addition to that, this chapter shows a full description of the methodology of hardware and software design. Also Design goals influenced by system specification, realistic constraints and evaluation of the effect of design choices are investigated. At the end of this chapter, work break down structure, and project schedule are illustrated.

3.2 System Specification

Referring to the architecture of the designed system shown in Figure 3.1, there are three main layers which are: storage layer presented by HDFS in the cloud, system applications layer which consists of indexing huge sets of data, saving the result of indexed data into hashtable, and serializing the hashtable and search in it. Finally the interface layer which shows the search interface designed in this project. To be able to implement this system, some functional and nonfunctional requirements are desired to be specified.

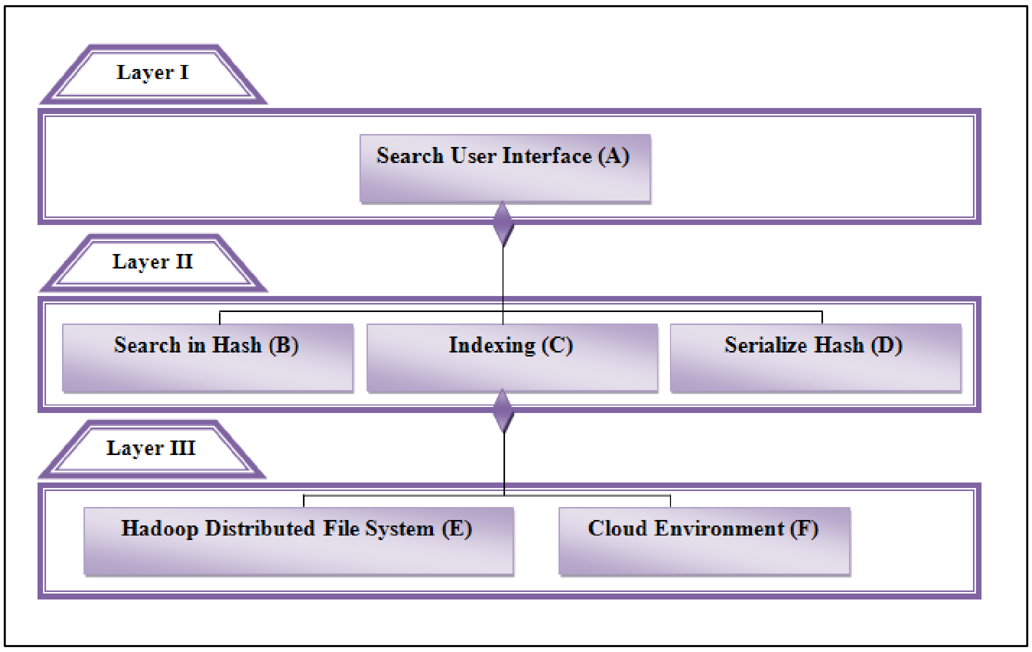


Figure3.1: System Architecture-Three Tier Architecture Style

3.2.1 Functional Requirements

The functional requirements illustrate the relation between the system and its environment independent of its implementation [31]. In this system, the environment consists of hardware and software resources and environment like the cloud, HDFS, Indexing using MapReduce, the inputted data itself, also the user of the system, and its applications. One of the system functional requirements is index a huge sets of data (0.5 TB) using the MapReduce algorithm. This data must be inputted as *.txt files to suit the Indexing algorithm. In addition to that, from the functional requirement is to convert the *.txt indexed output file into hashtable and serialize it.

Exploring the system from the user side, he/she can input a word that he/she looking for through the user interface, if the word exists in the indexed *.ser file (serialized object saved in a file) then the system outputs a result as a list of ten files where the word has highest occurrence. On the other hands, if the word doesn't exist, a message shown to the user as "This word doesn't exist". All those functional requirements are mapped together to produce a system that can analyze the processing of large scale intense data on the cloud using MapReduce programming model.

3.2.2 Non-Functional Requirements

The non-functional requirements express features of the system that are not directly related to the system functional behavior. Regarding the system described in this paper, there are many different requirements that fit in this field. One of them is usability. The user can address usability issues using the friendly user interface where he/she can use it without login or registration. Moreover to ease the dealing with the system, a "read me" file is given in the system, so the user can follow the guidelines to do his/her search. Looking to the system from performance perspective, the system should handle 0.5 TB, and the performance should be improved using MapReduce algorithm on the cloud that provides parallelism.

3.3 Hardware and Software Resources

3.3.1 Hardware Requirements

3.3.1.1 IBM blue Cloud 1.6

One of the most important hardware requirements for this project is: first, IBM blue cloud 1.6 which provides flexibility and scalability. Also it increases the ability for customizing hardware and software in a simple way and reducing the cost, installation and maintenance operations. The project takes advantage of the cloud infrastructure available at Qatar University (QU) and Carnegie Mellon University in Qatar (CMU-Q). The used cloud has some specifications :

1. Physical machines which are HS22 14 blades
2. A number of Virtual machines (at least 6 VMWares)
3. OS-level virtualization is Xen RedHat Linux 5.2
4. The IBM blue cloud in QU offers a private and separate network and composed of VLANs of range from 10.160.0.0 to 10.160.255.255 and the host range is 10.160.255.0 to 10.160.255.255. The network configuration is well demonstrated in Figure3.2

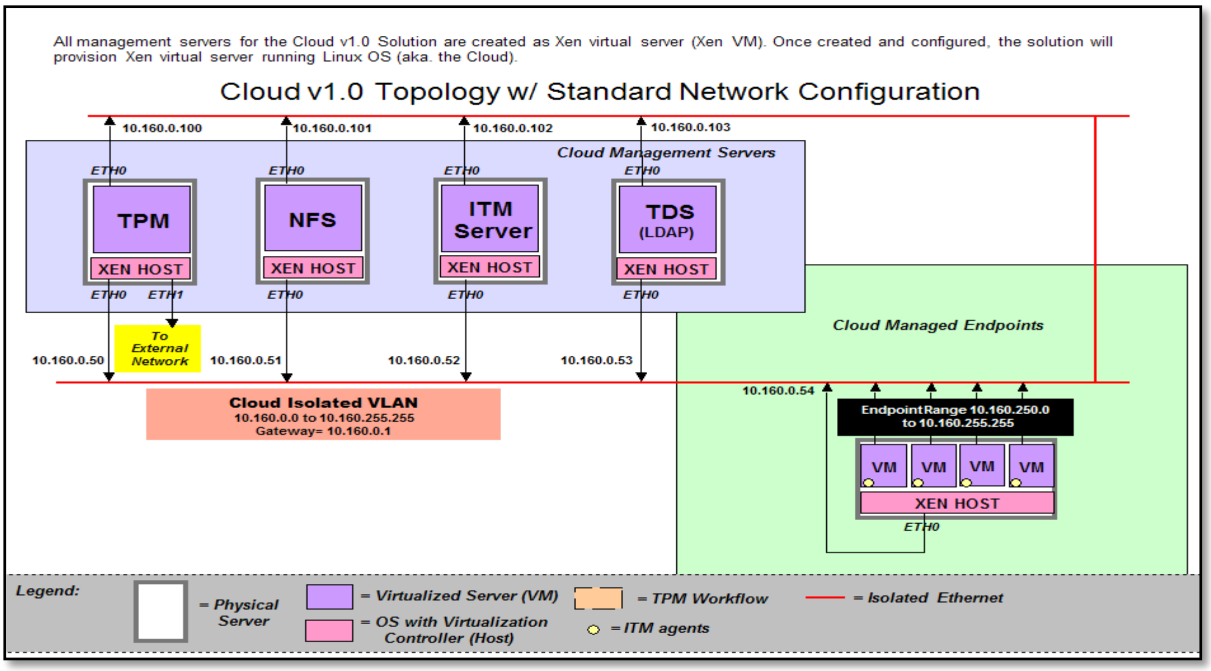
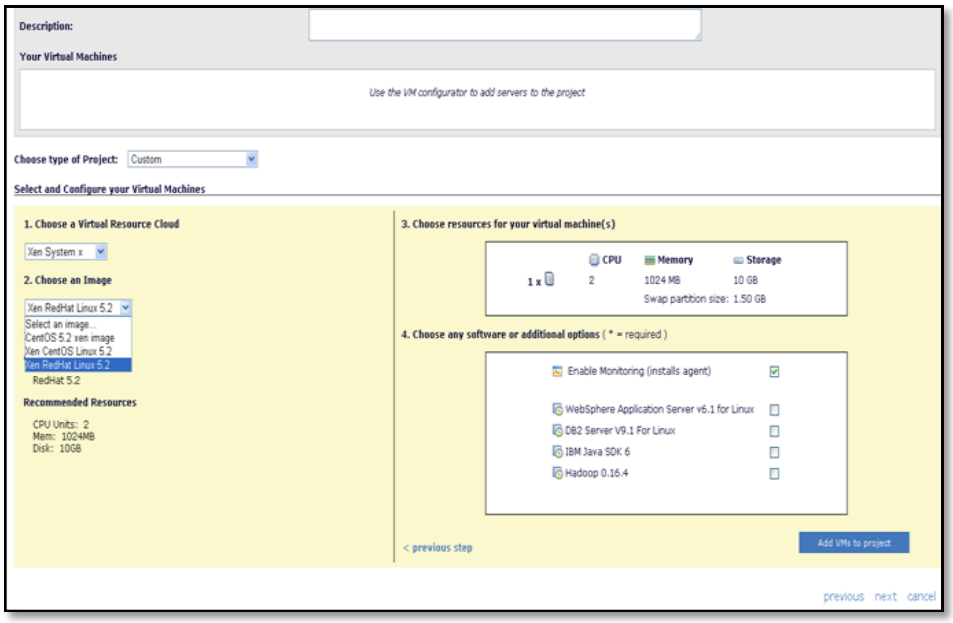

All management servers for the Cloud v1.0 Solution are created as Xen virtual server (Xen VM). Once created and configured, the solution will provision Xen virtual server running Linux OS (aka. the Cloud).
Cloud v1.0 Topology w/ Standard Network Configuration
10.160.0.100
10.160.0.101
10.160.0.102
10.160.0.103
Cloud Management Servers
ETH0
TPM
NFS
ITM Server
TDS (LDAP)
XEN HOST
ETH1
To External Network
10.160.0.50
10.160.0.51
10.160.0.52
10.160.0.53
Cloud Managed Endpoints
10.160.0.54
Cloud Isolated VLAN
10.160.0.0 to 10.160.255.255
Gateway= 10.160.0.1
EndpointRange 10.160.250.0 to 10.160.255.255
VM
Legend:
= Physical Server
= Virtualized Server (VM)
= OS with Virtualization Controller (Host)
= TPM Workflow
= ITM agents
= Isolated Ethernet

Figure 3.2: Blue Cloud 1.6 Network Configuration at QU[32]

5. Blue Cloud 1.6 at Qatar University supports the repository architectural style with at least 2 CPUs ,2.6 GB of RAM and 250 GB disk space and is designed to store persistent data on DB2 relational database management system. To implement those previously described features, DS3400 Storage for IBM is engaged [32].
6. Looking to the PaaS perspective of the Blue Cloud 1.6 at Qatar University (see section 2.2.5.2), we may find a wide range of platforms including: WebSphere Application server, BD2 Server, IBM Java SDK 6 and Hadoop 0.16.4 or Hadoop 0.20.1 (Figure 3.3) [32].


Description:

Your Virtual Machines

Use the VM configurator to add servers to the project

Choose type of Project: Custom

Select and Configure your Virtual Machines

1. Choose a Virtual Resource Cloud

Xen System x

2. Choose an Image

Xen RedHat Linux 5.2
Select an image...
CentOS 5.2 xen image
Xen CentOS Linux 5.2
Xen RedHat Linux 5.2
RedHat 5.2

Recommended Resources

CPU Units: 2
Mem: 1024MB
Disk: 10GB

3. Choose resources for your virtual machine(s)

	CPU	Memory	Storage
1 x	2	1024 MB	10 GB
		Swap partition size: 1.50 GB	

4. Choose any software or additional options (* = required)

Enable Monitoring (installs agent) ☑
WebSphere Application Server v6.1 for Linux ☐
DB2 Server V9.1 For Linux ☐
IBM Java SDK 6 ☐
Hadoop 0.16.4 ☐

< previous step

Add VMs to project

previous next cancel

Figure 3.3: Possible Platforms Available in IBM Blue Cloud 1.6

3.3.1.2 External Disk Drive

Second, external disk drives that can hold up to 250 GB - 1 TB of data. This external storage is needed to hold and move the data to cloud storage infrastructure.

3.3.2 Software Requirements

There are many software requirements related to the cloud which are:

3.3.2.1 Java Language

Java is a programming language which was designed for general purposes. It has many characteristics which make it appropriate to use in the designed system. Java programming language was created with specific goals and benefits. One of them is that the java is designed to be simple, familiar, and object-oriented language so it can be used to develop applications. Another thing that it should be robust and secure and executes with high performance. In addition to that, java is platform independent, and using a java virtual machine, java programs can be run on any platform. Moreover, Java has automatic memory management using the garbage collection in the object lifecycle. Using java syntax assisted in building our codes to apply the whole system, also Java libraries like SWING library help in creating the user interface in this system .[33]

3.3.2.2 Java Virtual Machine

In this project, Java Development Kit (SunJDK 1.6.0_06) or Java Runtime Environment(JRE 1.6)virtual machines can be used and set in JAVA_HOME in the variable environment. These virtual machines are Java software development environment which contains everything that a user needs to create a java program, for example it contains compiler, debugger, .jar Packages and other tools for developing applications. [33]

3.3.2.3 Hadoop Platform

Hadoop is a Java software powerful framework for applying automatic parallelization on many computing distributed applications on huge number of nodes in a commodity hardware. Hadoop implements MapReduce computational model, so the application is divided into a number of small chunks of work. In addition, Hadoop has its own distributed file system (HDFS) which stores data on the compute nodes and replicates data to multiple nodes to ensure if failure happened for data in a node, there are at least two other nodes from which to recover that piece of information. [11] .In this project, Hadoop version 0.20.1 is used ,it was the considered version choice after testing for many other versions which had many problems and bugs, like Hadoop 0.16.4,Hadoop 0.18.3, Hadoop 0.19.1.,Hadoop 0.20.0,

3.3.2.4 Eclipse

Eclipse is a Java environment which is available for windows and Linux platforms. In the applied system, it is used for windows. Using this environment let the user run his java codes easily. There are many versions of Eclipse, and Eclipse 3.2

EUROPA is the used version in this project because it suits the used version of Hadoop 0.20.1.

3.3.2.5 MapReduce Plugins

Those plugins are added to the ones of the Eclipse, and are required to allow running MapReduce programs on Eclipse, also to permit the interaction between Hadoop Distributed File System on the cloud and Eclipse, where MapReduce locations can be created, and MapReduce projects can be used.

3.3.2.6 Cygwin

Cygwin is a Linux like environment that allows Unix or Linux applications to be compiled and run on a Windows Operating system. It is used in this project to help in installation of Hadoop 0.20.1 because it can run the scripts supplied with Hadoop which are written for the Unix platform only. [35]

3.3.2.7 Web Crawler

Web crawler is a small program that crawl to wide world web. It automatically passes through the Web. It is also called spider, robot, or wanderers. It is used in this project to collect huge sets of data to be processed and indexed later. In this project it was supposed to use Crawler4j because it is good for education purposes, and it is a fast crawler written in Java. [29]

3.4 The Conceptual Model

The previously discussed hardware and software requirements are going to be applied in an efficient manner to achieve the project functional and non-functional requirements. The bellow model (Figure 3.4) demonstrates the conceptual general methodology followed during the project development. The system stages are composed of three phases:

- First phase: the LineIndexer code is applied on a large set of data (generated data in our case) stored in the HDFS and the output is stored there.
- Second phase: the CovertTextToHash code is applied on the output of the previous phase , the output.txt is stored in a hash table and then it will be serialized.
- Third phase: In the user interface, the user enters a word, then the serialized file from the previous phase is deserialized and the code will perform search in the deserialized hash and outputs the results to the user: outputs the 10 most frequent files.

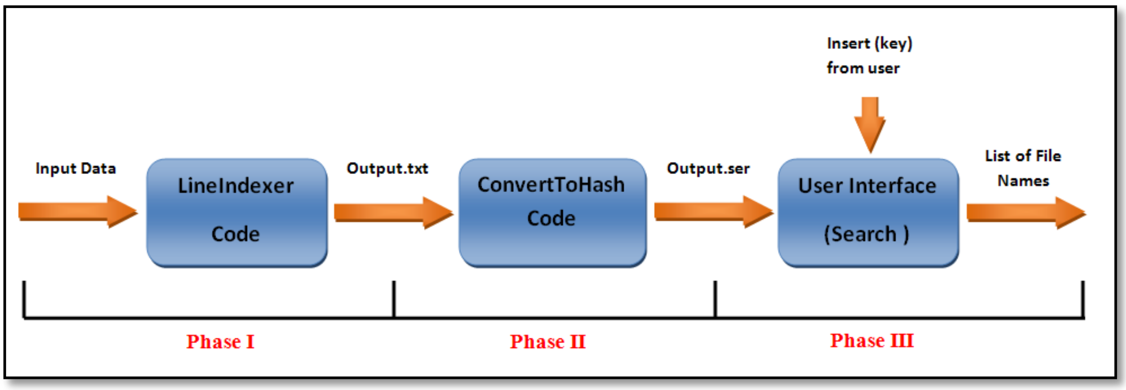


Figure 3.4: System Conceptual Model

Because our project is based on applying MapReduce algorithm, which is indexing in our case, on the cloud, the most important and critical phase is the first one: LineIndexer code. The second following conceptual model (Figure 3.5) describes how the LineIndexer works. First the data is passed to the map step then the output of the map is entered in the reduce step to produce the needed indexed file

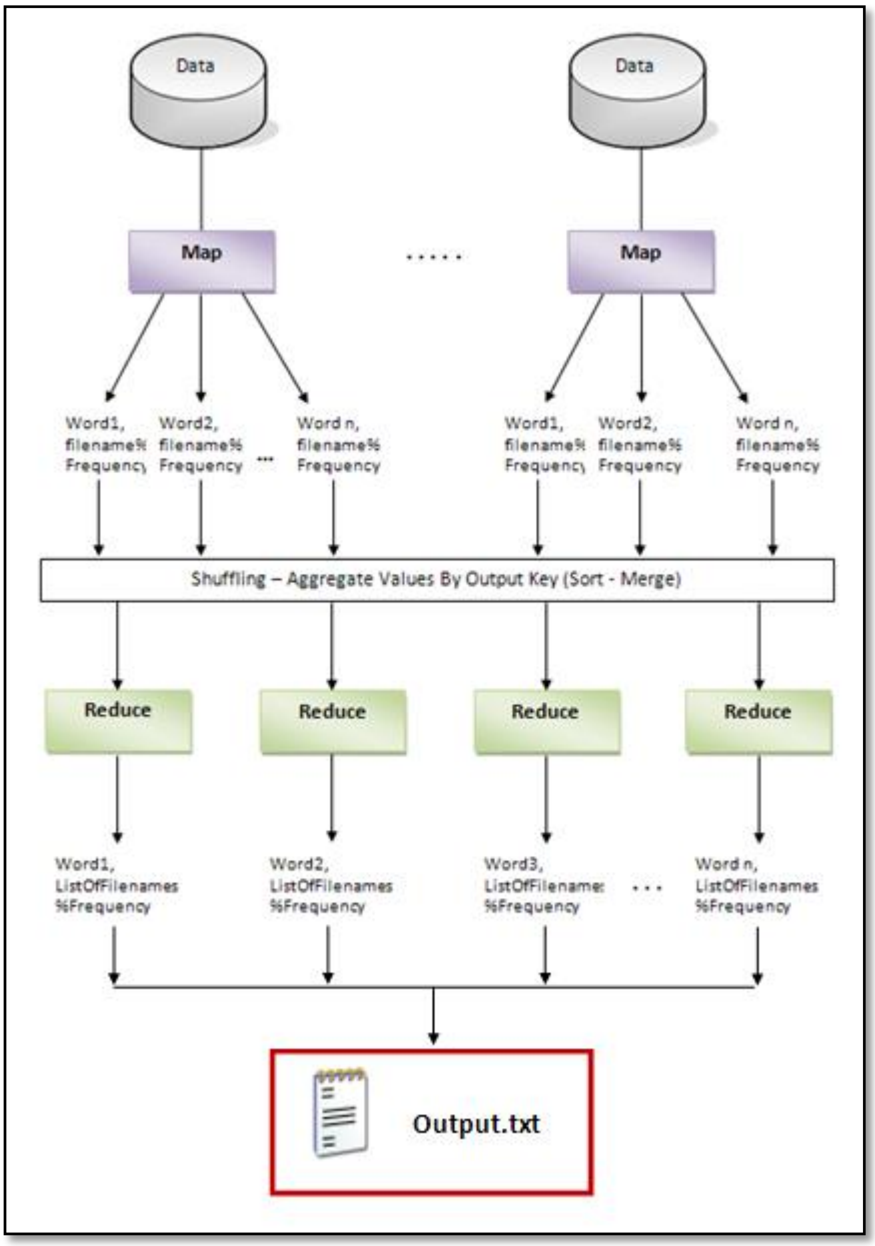

Data
Data
Map
.....
Map
Word1, filename% Frequency
Word2, filename% Frequency
...
Word n, filename% Frequency
Word1, filename% Frequency
Word2, filename% Frequency
Word n, filename% Frequency
Shuffling – Aggregate Values By Output Key (Sort - Merge)
Reduce
Reduce
Reduce
Reduce
Word1, ListOfFilenames %Frequency
Word2, ListOfFilenames %Frequency
Word3, ListOfFilenames %Frequency
...
Word n, ListOfFilenames %Frequency
Output.txt

Figure 3.5: MapReduce Phase Conceptual Model

3.5 Proposed Solutions

3.5.1 Utilization of Design Related Standards and Recognition of Professional Design Codes

3.5.1.1 Cloud Computing Standards:

In our project, because the cloud is chosen to be the most important hardware resource (as mentioned in section 3.3.1.1), cloud standards are applied. Cloud computing open standards are influenced with the cloud computing technology growth. As a result of that, each cloud provider has its own unique API which is not interoperable with others [7]. One of the organizations working on developing cloud computing standards is the Distributed Management Task Force (DMTF) and one of the members of this organization is IBM. One of the standards developed by this corporation is the Open Cloud Standards Incubator [”37]. The DMTF standards focus on standardizing interactions between different cloud environment by developing cloud resource management protocols, packaging formats and security mechanisms to facilitate interoperability [38]. In addition this standard, IBM cloud has standards like:

- Open Cloud Computing Interface.
- Federated security across Clouds.
- Standards for moving applications between Cloud platforms.
- Standards for machine-readable Service Level Agreements.
- Standardized outputs for monitoring, auditing, billing, reports and notification for Cloud applications and Services.[39]

3.5.1.2 Software Standards:

3.5.1.2.1 Eclipse Standards:

Because we chose eclipse to be our software development environment, we had to utilize the standards provided by the eclipse. The runtime system of Eclipse is based on Equinox which is an OSGi standard compliant implementation [40].

3.5.1.2.2 Hadoop Standards:

Hadoop is the platform used to run a MapReduce program on the cloud. To achieve its goals Hadoop is built on different standards. One of them is Hadoop streaming: it is a Unix standard streaming used as an interface between the Hadoop and any software environment like Eclipse which is in our project [1].

Another standard used in the Hadoop environment is the input/output standard built by Python programming language and which says that the natural input format is a text file. This standard is supported by all programming languages and also java which is the language used in our project [1].

3.6 Hardware and Software Design Methodology

3.6.1 Hardware Design Methodology

As mentioned in the Hardware Requirements section (3.3.1), we are going to use ,in this project, the IBM blue cloud 1.6. In addition to its features described in 3.3.1 section, the IBM blue cloud was considered as one of the project hardware needs because this type is the one provided by QU university. Furthermore, all the project group members attended a 3 day training about this specific cloud type so dealing with this cloud would be easier. The last thing in this regards is that this cloud is supported by IBM and any faced problems would be directed to them to be fixed.

3.6.2 Software Design Methodology

3.6.2.1 Java Language and Eclipse

Fist starting with the language, java is used as our project language because it is platform independent: because of using a java virtual machine, java programs can be run on any platform. This is very important in our project since we are working on different platforms (Linux, Microsoft). In addition to all what was said, GUI components are provided in the swing library of the java language and by that, we can develop our interface in a simple and fast manner. Besides that, java language is one of our fundamental subjects studied in the computer engineering curriculum, it was easy to work with it and use it as our programming language.

The used development environment for java is Eclipse. Eclipse is suitable for both windows and Linux platforms. So, it is a good choice to have it as our java environment. Also, the eclipse is already tested to work with the Hadoop platform to write MapReduce algorithms.

3.6.2.2 Code design Methodology

As we are developing a MapReduce algorithm model and as we are using java language, we designed java codes to fulfill this purpose. The java codes are well explained in section 4.1 and the decisions made in those codes and what was exactly used are explained as follows:

3.6.2.2.1 Set Number of mappers and reducers:

The MapReduce programming model is applied in the project for a specific application which is indexing. The part responsible for the indexing phase is demonstrated in the LineIndexer code explained in section 4.2.2. For the main() method, we can find that the number of mappers is not set (default one) and the number of reducers is set to 11.

Starting with the number of mappers: it is set to be the default one which means that the number of mappers is calculated by dividing the total input size over the block size which is by default 64MB. [1] We choose the default method because it provides a high level of parallelism.

Moving to the reduce numbers: we chose 11 to be our reduce task number because referring to statistical calculations: the best number of reduces is the number of used nodes × 0.95 ($6VMs \times 0.95$=5). With 1.75, the faster nodes finish their job and switch to the second wave which provides good load balancing. ($6VMs \times 1.75 = 11$). As a result of that, increasing the number of reduces will increase the number of failure. [41].

3.6.2.2.2 SortedList

The sorted list is used for the LineIndexer (map section) code in two manners: first to store the indexed words, with their location and frequency and then to store the unused list of words, which is a list of words that is usually are not searched for such as: the, a, an and that. These words are chosen to be removed from the indexing input files. Each sorted list has its individual link. We chose to the data structure of sorted list because it has a high speed in inserting the elements (no need to move any element) and its complexity is O($\log_2 n$) , where n is the number of elements, which is considered very fast . The second advantage is that it does not have a specific size: as long as there is a space in the memory, the sorted list will be stored [42]. Another important thing to be mentioned in this regards is the methodology followed to select the unused words that should be deleted from the index output file. We looked for the most frequent words in English and took them from the following website http://www.world-english.org/english500.htm and then we chose the unused ones for a search purpose to make the searching process faster.

3.6.2.2.3 HashTable

The hash table is one of the data structures that offers very fast insertion and searching. In addition to that, hash tables are relatively easy to program. For those reasons, we chose this data structure to be the storage space for our indexed output file. The main reason for choosing hash table to save the indexed files is its speed and high performance. In general, the hash function used in the hash has the most important impact. For the best function choice, and with a hash of n elements and k keys, the number of collisions would be (0,k-n) and the number of lookups (1+ k/n) [43]. A point to be mentioned here is that the hash used in our code is the one developed by the JAVA langue and we only needed to override the JAVA hash methods that are not available and the ones we need like word_search().

There is also another possibility for saving the indexed file in a database. Using a database is another alternative but we chose hash because of the size of the value which is not stable and fix: if we use database we have to fix the size if the second row but our value is not fixed: (the word could be in one file, no files or hundreds of files) so by using database, we may come to memory waste.

3.6.2.2.4 Serializable File

Because we are using a hash table, we can get benefit from the serializable feature which means converting a specific data structure into a sequence of bits stored in a file. In our project, we used this technique to store the hash table (where the indexed file is stored) in a ser file.

We had two ways to perform the search application. Either search for the word directly from the hash table created for each search or to deserialize the ser file and search in it. We chose the second method because we tested both methods and find out that searching directly from hash table takes 187 ms. On the other side, searching from a deserialized file to hash takes 109 ms. This test was performed for an indexed file of 643Kb size. With this results, we can assume that using a serialized file instead of a direct hash table would be faster and more efficient and the suggested reason is that file serializing uses data streaming but hash table doesn't.

3.7 Design Goals Influenced by System Specifications and Realistic Constraints

In this project, the system resources and the architectural design were chosen depending on some important realistic constrains. One of them was the economical factor, and this is shown in the hardware used in the project, where we didn't have the choice to buy a cloud of some higher specifications, but we used the IBM blue cloud1.6 that is designed for educational purposes, and recently available in Qatar University (QU).

Another limitation was considered in this project, was the organizational aspect which also affect the system hardware resources: Although there was collaboration between some organizations such as IBM organization and universities like (QU) and (CMU-Qatar), There was no cloud system that enabled us to work across the cloud, so there was no chance to use a hardware that consisted of multiple clouds represented as one cloud.

In addition to those aspects, this project was limited by time. For example, in the design of the applied system, there was a number of improvements that can be added as discussed in section 6.5, on the other hands, that couldn't happen because there was no enough time to explore new fields.

From the other constrains on the designed system, is the bandwidth restriction. A high bandwidth was needed when data gathering plan was collecting data using web crawler, even it was needed in uploading the huge set of data from an external hard drive into HDFS in the cloud, also it was required when we tried to run any code we had created, specially the code responsible for doing the indexing job. But within the rate of the bandwidth provided, gathering data plan was changed, and running codes should be done in the university campus, so we could at least get good bandwidth rate.

Furthermore, the user interface in this project is designed depending on the social aspect, where it was considered to provide usability, and enable the user to utilize it in a easy way. Looking for the political, ethical, health, security and safety fields, those fields were out of the scope of the designed system.

3.8 Evaluation of The Effect of Design Choices

This system is designed depending on the professional design code. Furthermore, it does not break any related standards. This provides usability where it eases the employment of the system for any user. Moreover, this offers flexibility of adding new features, applications and functionalities to the system. Following these standards helps us as engineers to evaluate our movements in the project.

3.9 Work Breakdown Structure

This section discusses the work breakdown structure of this project as shown in table 3.1. The project is divided into nine main tasks and all of these tasks are performed by all of the team members.

Task 1: Literature Survey and Background: this task is to build a background about the related topics and technologies needed in the project. Examples of the activities that contributed to the fulfillment of this task are: gathering information and reading about cloud computing, MapReduce, indexing, web crawling and Hadoop.

Task 2: Access CMU-Qatar cloud: this task was considered because we didn't get access to (QU) cloud immediately when we started the project. The task helped us to know more about the cloud environment. This task includes having accounts on the CMU-Q cloud. At the same time a Bitvise Tunnelier, SSH Terminal and File Transfer Client to access the cloud. Then Learn how to create new projects and explore clusters.

Task 3: Dealing with Hadoop: this task is divided into two main steps: installing and using Hadoop on the local machine and using Hadoop on Qatar University cloud. A number of versions of Hadoop were explored in this task in order to find a version without problems and suits the used environment.

Task 4: Access QU cloud: In this stage, QU cloud was finally set, so accounts were created and we could access the cloud using VPN client. And IBM training was provided which offered a brief explanation about the hardware of Qatar University cloud. Furthermore, illustration of the roles and capabilities of the cloud is provided. The last day of the training is assigned to deal with cloud and Hadoop projects.

Task 5: Gathering Data: In gathering data task, we considered four different plans:

- Plan A is to get the data from Qatar Petroleum Company(QP). Many meetings are arranged between Dr. Qutaibah Malluhi and Dr. Khalid Shaban with QP employees to convince them of the effectiveness of the new cloud computing technology.
- Plan B is to get data using web crawler, but it was found that the used crawler inefficient to collect needed set of data within the rest of time we have to submit the project.
- Plan C is to get the data from al-Jazeera Networks. But also looking for the time constrain, the idea wasn't doable.
- Plan D is to generate files using a simple java code, so we can gather 0.5 TB. After getting the data, it is moved to the cloud storage space.

Task 6: Query selection: For this project the indexing query was selected.

Task 7: Demo design and implementation: We have already completed the design and implementation of the following components:

- MapReduce algorithm for generating an index for the huge input consisting of large number of documents and a module to convert the MapReduce-generated index (a text file) into a hash table for faster search operation;
- Data generation code to supplement the input data collected through the Web crawler,
- A simple graphical user interface (GUI) to issue a search query and present search results.

Task 8: Testing and Evaluation: We have done some preliminary unit testing to test the correctness of the individual modules, and integration testing to validate that the modules can work well with each other (e.g. the search algorithm can work with the output of the index converter output). More testing is needed to evaluate the performance of our algorithm and to compare the performance of different scenarios for running the algorithm (e.g. different input sizes, or platforms with different number of virtual machines). These performance evaluation experiments have been designed but not yet executed. this task is to test and evaluate the system. It is expected that these performance evaluation experiments will be conducted in June 2010.

Task9: which is the last task is the project documentation.

3.9.1 Role of Team Members

Based on the below tasks description each student is responsible for completing these tasks

Amira:1.1,1.2,1.3,1.4,2,3,4,5,6,7.1,7.2,7.3,8.1,9

Farah: 1.1,1.2,1.31,2,3,4,5,6, 7.1,7.2,7.3,8.1,9

Nadia:1.1,1.2,1.3,1.4,1.5,2,3,4,5,6, 7.1,7.2,8.1,9

3.9.2 Interdependence of Individual Role on the Team Goals

In this project, all the tasks given to the team members ,are relying on each other. First, they need to explore the needed fields which are cloud computing, Hadoop platform, indexing MapReduce algorithm. A second step is to gather the data using files generation. After that, the team should implement the indexing algorithm on the data sets using the power of Hadoop on QU cloud, and finally evaluation of the performance can be done.

3.10 Project Schedule

Table(3.1) shows the Gantt Chart Activity of the project. It illustrates each task with its corresponding time schedule.

Table 3.1: Gantt Chart Activity

Gantt Chart Activity	Months								
🟥 = Activity 🟥X = Completed Activity	1 Oct	2 Nov	3 Dec	4 Jan	5 Feb	6 Mar	7 Apr	8 May	9 Jun
Task1: Literature Survey and Background	🟥	🟥	🟥	🟥	🟥	🟥	🟥X		
1.1 Reading about Cloud Computing	🟥	🟥	🟥X						
1.2 Reading about MapReduce	🟥	🟥	🟥X						
1.3 Reading about Hadoop and Hadoop Distributed File System	🟥	🟥	🟥X						
1.4 Reading about Crawling					🟥	🟥X			

1.5 Reading about Generating Files						▨	▨ X		
Task 2: Access CMU-Q Cloud	🟥	🟥	🟥 X						
2.1 Having Accounts on CMU-Q Cloud	▨ X								
2.2 Installing Tunnelier (a free ssh client for windows	▨ X								
2.3 Configuring the Cloud Access		▨	▨ X						
2.4 Using the Web Browser to Access Qloud URL and exploring the clusters		▨	▨ X						
2.5 Learning how to create a project on the cloud with specific criteria		▨	▨ X						
Task 3: Dealing with Hadoop	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥 X	
3.1 Configuring Hadoop on Local Machine	▨	▨	▨					▨ X	
3.2 Configuring Hadoop on QU cloud	▨	▨	▨	▨	▨	▨ X			
Task 4: Access QU Cloud			🟥	🟥	🟥	🟥	🟥	🟥 X	
4.1 IBM Training			▨ X						
4.1.1 Briefly Reviewing Hardware of Qatar University Cloud			▨ X						
4.1.2 Understanding Cloud Roles and Capability			▨ X						
4.1.3 Dealing with Hadoop on Qatar University Cloud			▨	▨	▨	▨	▨	▨ X	
4.2 Having Accounts on QUCloud						▨ X			
4.3 Installing VPN Client			▨ X						
4.4 Configuring the Cloud Access and Using the Web Browser to Access Qloud URL and exploring the clusters			▨ X						
4.5 Creating a New Hadoop Project with Special Specifications						▨	▨	▨ X	
Task 5: Gathering Data	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥 X	
5.1 Plan A: Getting Data from Qatar Petroleum Company (QP)	▨	▨	▨	▨	▨ X				
5.2 Plan B: Use web crawler to gather data					▨	▨	▨	▨ X	

5.3Plan C: Getting Data from Al-Jazeera Networks							▨	▨ X	
5.4 Plan D: Generate files with some Specifications for Testing							▨	▨ X	
5.5 Moving Data to the QU Cloud Storage Space								▨ X	
Task 6: Query Selection					▨ X				
Task 7: Demo Design and Implementation						🟥	🟥	🟥 X	
7.1 Design Effective Algorithm							▨	▨ X	
7.2 Implement Indexing Using MapReduce on The Cloud						▨	▨	▨ X	
7.3 Build the System Interface							▨	▨ X	
Task 8: Testing and Evaluation						🟥	🟥	🟥	🟥 X
8.1 Execute on QU Cloud						▨	▨	▨ X	
8.2 Evaluate Performance								▨	▨ X
8.3 Optimize Performance								▨	▨ X
Task 9: Research Documentation			🟥	🟥	🟥	🟥	🟥	🟥	🟥 X

CHAPTER 4

IMPLEMENTATION AND DEPLOYMENT

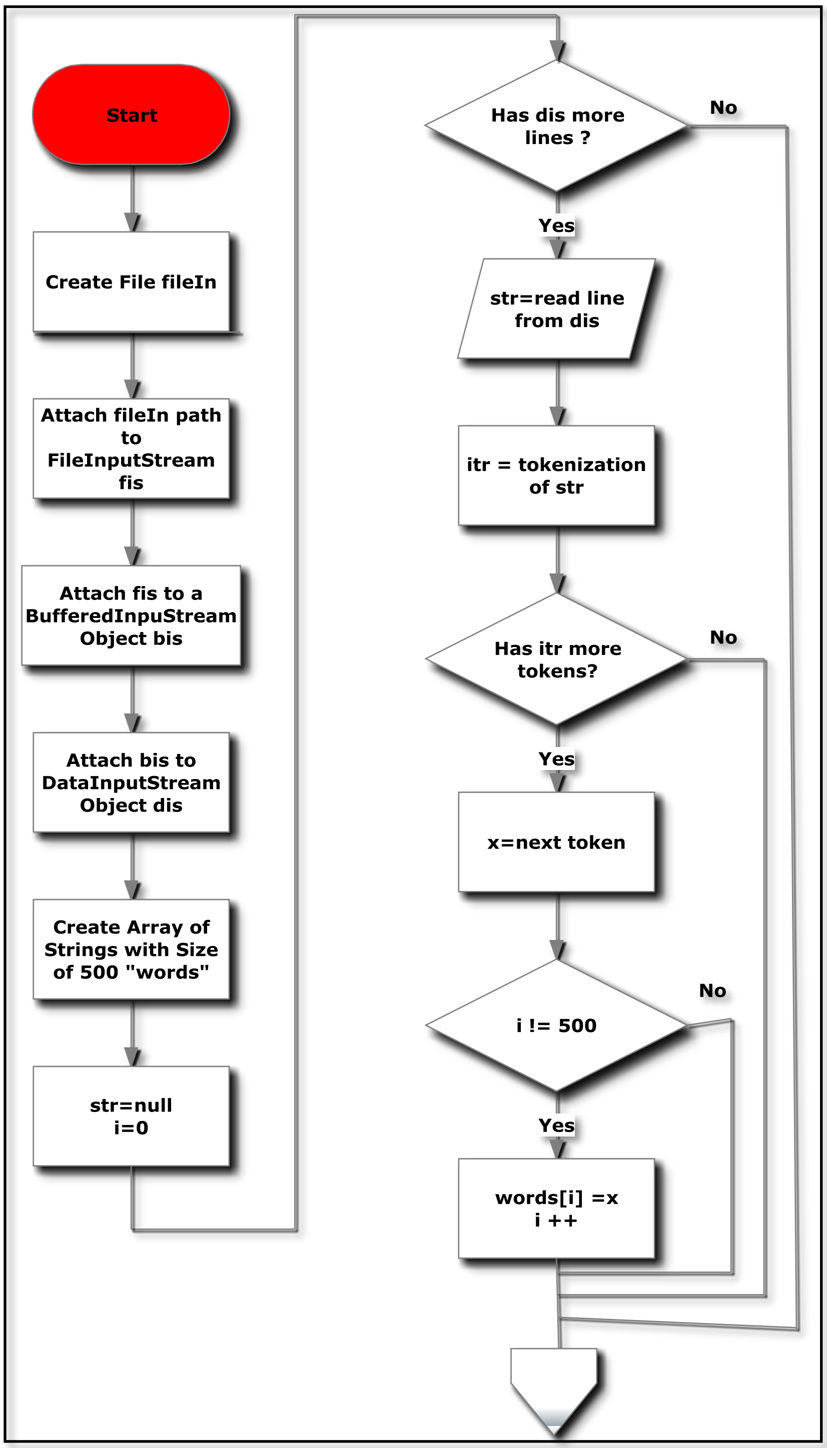
4.1 Introduction

This chapter illustrates the implementation and deployment of our design. Logical flowcharts for basic classes are drawn and explained. Also, you can find a fully description of the completed integrated system. In addition, we point out to the computer environment deployment. At the end of this chapter, software installation and usage steps are listed.

4.2 Logic Flowcharts

4.2.1 Generate Data Flowchart

Generating data was the solution for many problems in getting real data. Figure 4.1 and Figure 4.2 show the whole process



Start
Create File fileIn
Attach fileIn path to FileInputStream fis
Attach fis to a BufferedInpuStream Object bis
Attach bis to DataInputStream Object dis
Create Array of Strings with Size of 500 "words"
str=null
i=0
Has dis more lines ?
No
Yes
str=read line from dis
itr = tokenization of str
Has itr more tokens?
No
Yes
x=next token
i != 500
No
Yes
words[i] =x
i ++

Figure 4.1: Generate- Part 1 Flowchart

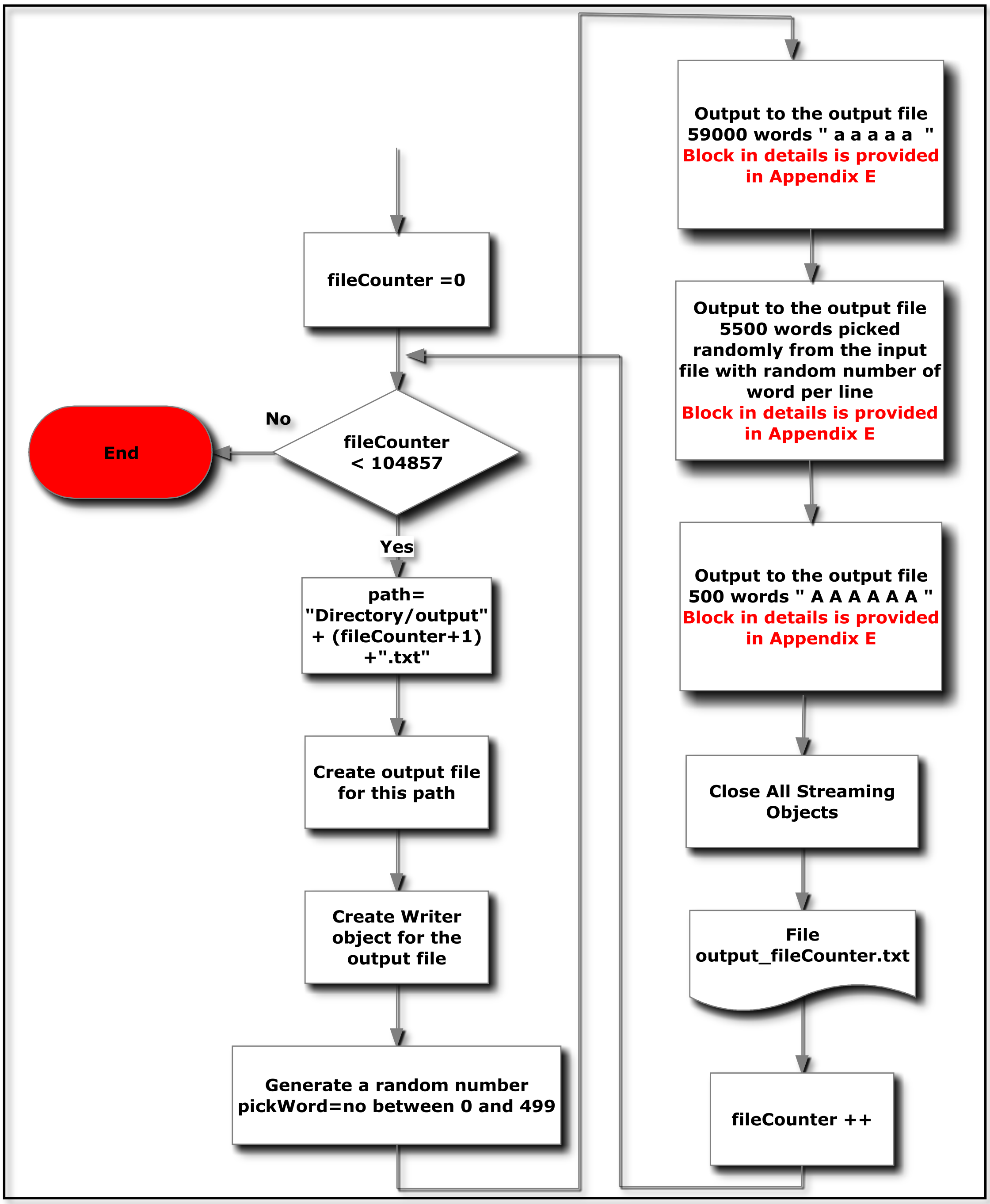

fileCounter =0
fileCounter < 104857
No
End
Yes
path= "Directory/output" + (fileCounter+1) +".txt"
Create output file for this path
Create Writer object for the output file
Generate a random number pickWord=no between 0 and 499
Output to the output file 59000 words " a a a a a "
Block in details is provided in Appendix E
Output to the output file 5500 words picked randomly from the input file with random number of word per line
Block in details is provided in Appendix E
Output to the output file 500 words " A A A A A A "
Block in details is provided in Appendix E
Close All Streaming Objects
File output_fileCounter.txt
fileCounter ++

Figure 4.2: Generate- Part 2 Flowchart

4.2.1.1 Generate Class

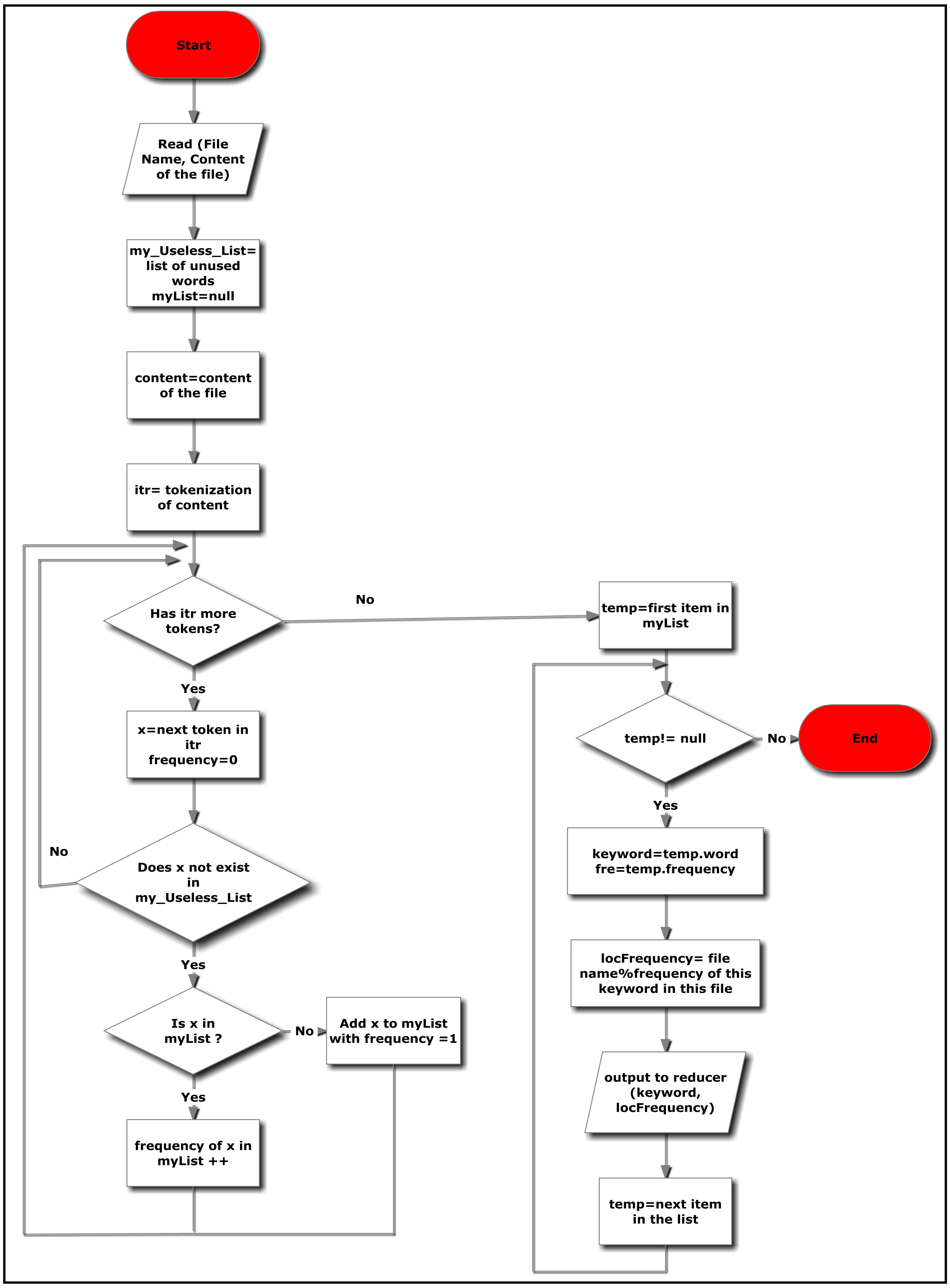
The main method starts with declaring some required objects such as the input file, number of files to be generated, ..etc. Then the code is divided into two main sections:

- Section 1: lines 44-59 in the code are used to save words from the input file to array of strings, words[]. This makes getting random word easier by generating a random number as an index to a random word.
- Section 2: lines 65 to 102 are used to generate the text files. An outer for loop is used for numbering and naming the output files. Words in each output file are collected using 3 for loops. The first loop is used to generate 59000 " a a a a a " at the beginning of the output file. The second loop is used to pick 550000 random words from words[] array and put them in the output file. The last loop is for putting 58000 " A A A A A " at the end of the output file. By the end of the third loop, generating one file is completed and the outer for loop will take care of switching to generate the next file.

Code is provided in Appendix D section 2. Generate.java.

4.2.2 LineIndexer Class Flowcharts

4.2.2.1 Mapper Flowchart



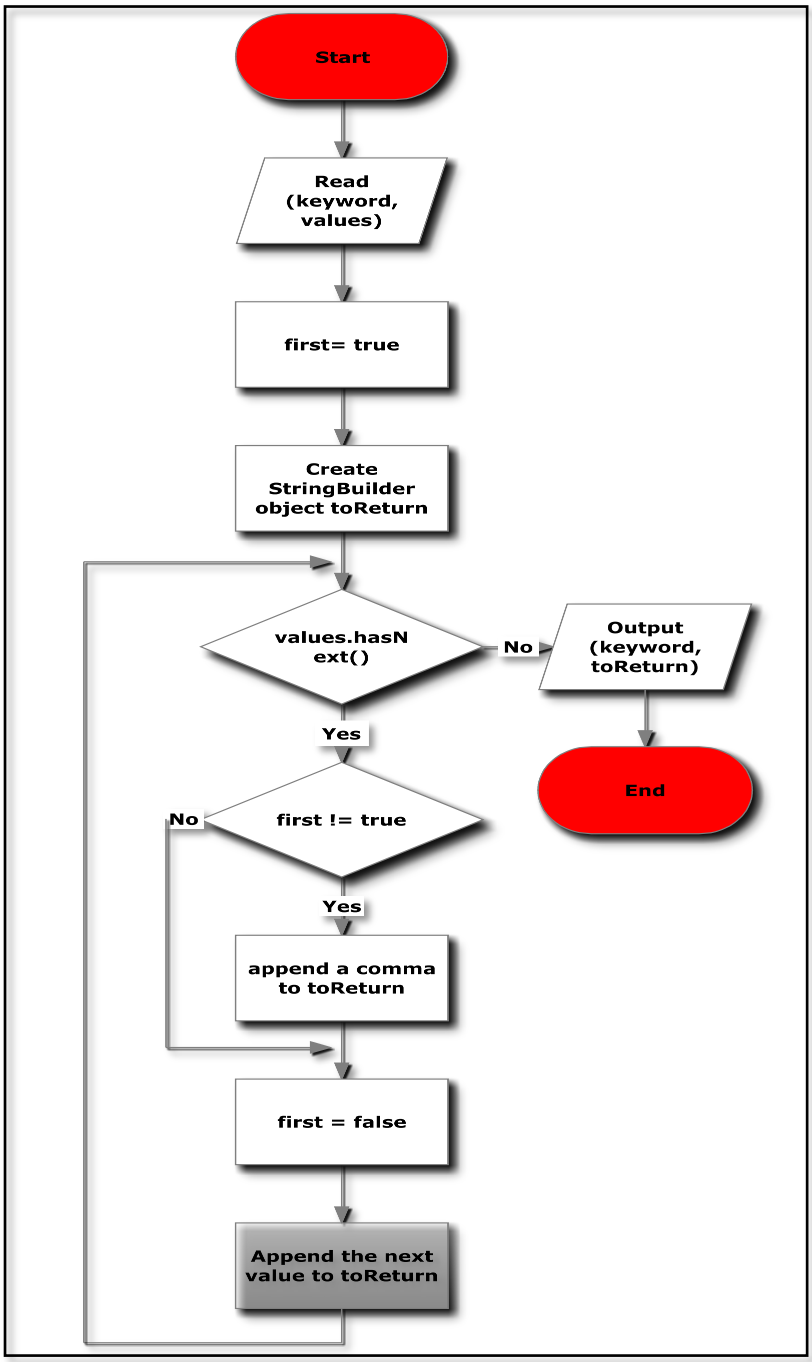
Start
Read (File Name, Content of the file)
my_Useless_List= list of unused words myList=null
content=content of the file
itr= tokenization of content
Has itr more tokens?
No
Yes
x=next token in itr frequency=0
Does x not exist in my_Useless_List
No
Yes
Is x in myList ?
No
Add x to myList with frequency =1
Yes
frequency of x in myList ++
temp=first item in myList
temp!= null
No
End
Yes
keyword=temp.word fre=temp.frequency
locFrequency= file name%frequency of this keyword in this file
output to reducer (keyword, locFrequency)
temp=next item in the list

Figure 4.3: Mapper Flowchart

4.2.2.1.1 Map Method

Figure 4.3 shows how the map works. For each file, it makes a list of the words in the file with their frequencies, and finally, output them all to the reducer as (word, filename%frequency) pairs. Code is provided in Appendix D, 3. LineIndexer.java

4.2.2.2 Reducer Flowchart



Start
Read (keyword, values)
first= true
Create StringBuilder object toReturn
values.hasNext()
No
Output (keyword, toReturn)
End
Yes
first != true
No
Yes
append a comma to toReturn
first = false
Append the next value to toReturn

Figure 4.4: Reducer Flowchart

4.2.2.2.1 Reducer Method

Each reduce task handles a word(the key) to collect all the file names and the frequencies related to it. Then, it passes the (word, List of files with frequencies) pair to the final output file. Figure 4.4 demonstrates the process of the reducer. Also, code is provided in Appendix D, 3. LineIndexer.java.

In LineIndexer class, the map and reduce tasks are configured and called in the main class. Many configurations can be set. This main is the driver of this MapReduce project. The list of configurations, used in our code, are as follows (refer to code listed in Appendix D:3 . LineIndexer.java):

- Line no. 122: set the name of the job for the client to be the class name.
- Line no. 127: set the input path used by the HDFS.
- Line no. 128: set the output path used by the HDFS to store the results.
- Line no. 126: set the number of reducers (refer to section in 3.6.2.2.1 the methodology to know how this number is chosen).
- Line no. 129 and 130 is used to set the mapper and reducer class.

4.2.3 Convert Text to Hash Table Flowchart

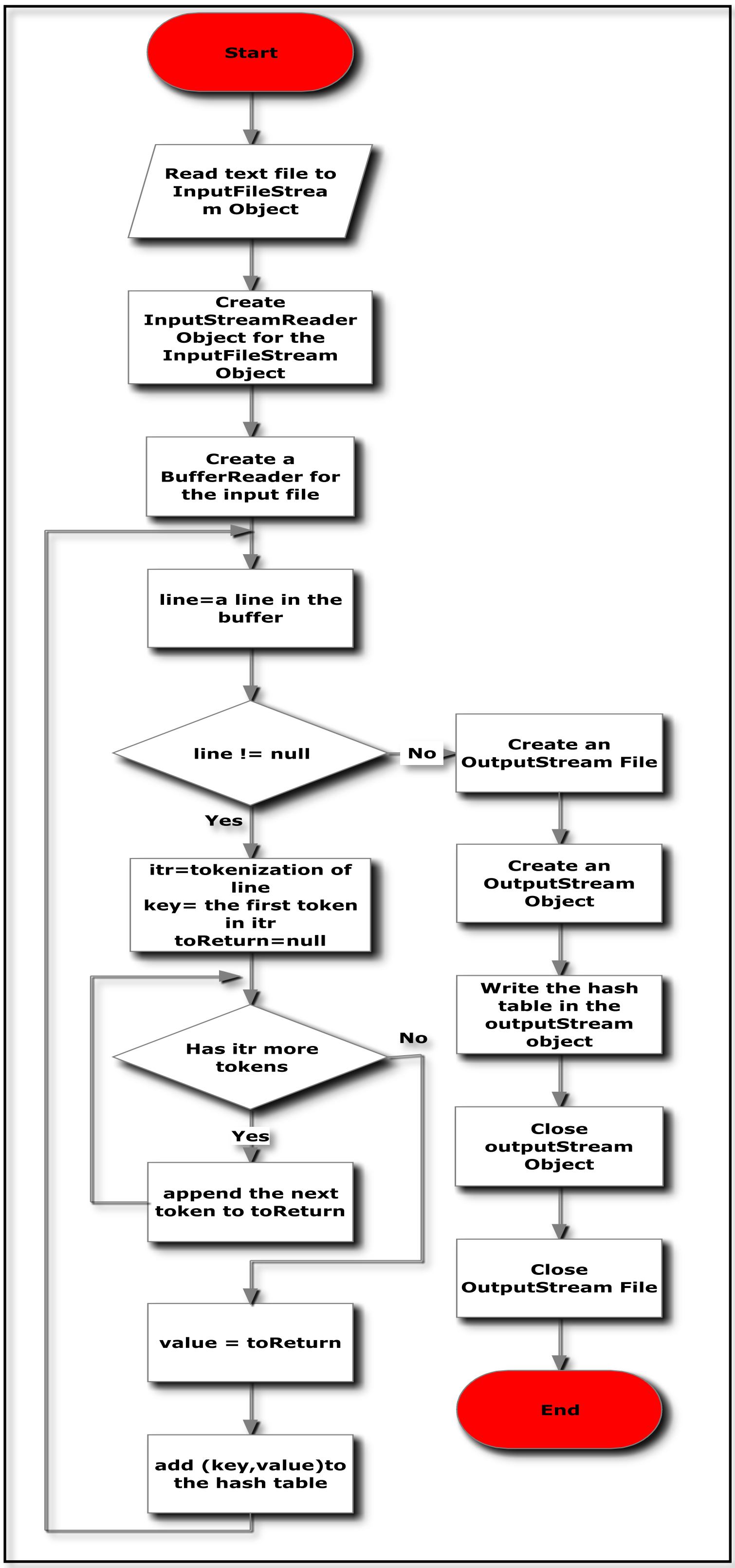

Start
Read text file to InputFileStream Object
Create InputStreamReader Object for the InputFileStream Object
Create a BufferReader for the input file
line=a line in the buffer
line != null
No
Yes
itr=tokenization of line
key= the first token in itr
toReturn=null
Has itr more tokens
No
Yes
append the next token to toReturn
value = toReturn
add (key,value)to the hash table
Create an OutputStream File
Create an OutputStream Object
Write the hash table in the outputStream object
Close outputStream Object
Close OutputStream File
End

Figure 4.5: Convert Text To Hash Table Flowchart

4.2.3.1 Convert Text to Hash Class

Figure 4.5 shows the flowchart of ConvertTextToHash class. This class is used to convert the text file, which is the output of the LineIndexer, to a hash table (object of class Hashtable). Then it serializes it and saves it into a .ser file. This file is used in SearchHash Class to be loaded and used in the search application. The code is provided in Appendix D, 1.ConvertTextToHash.java

4.2.4 Search in a Hash Table Flowchart

Figure 4.6 illustrates the flow of the searching application. The user enters the keyword to search for it. Then, if this word exists in any file that indexed using LineIndexer code, the list of the top 10 files that have highest frequencies is printed to the user. If this word has a list of files contains less than 10 file, then the whole files are printed to the user.

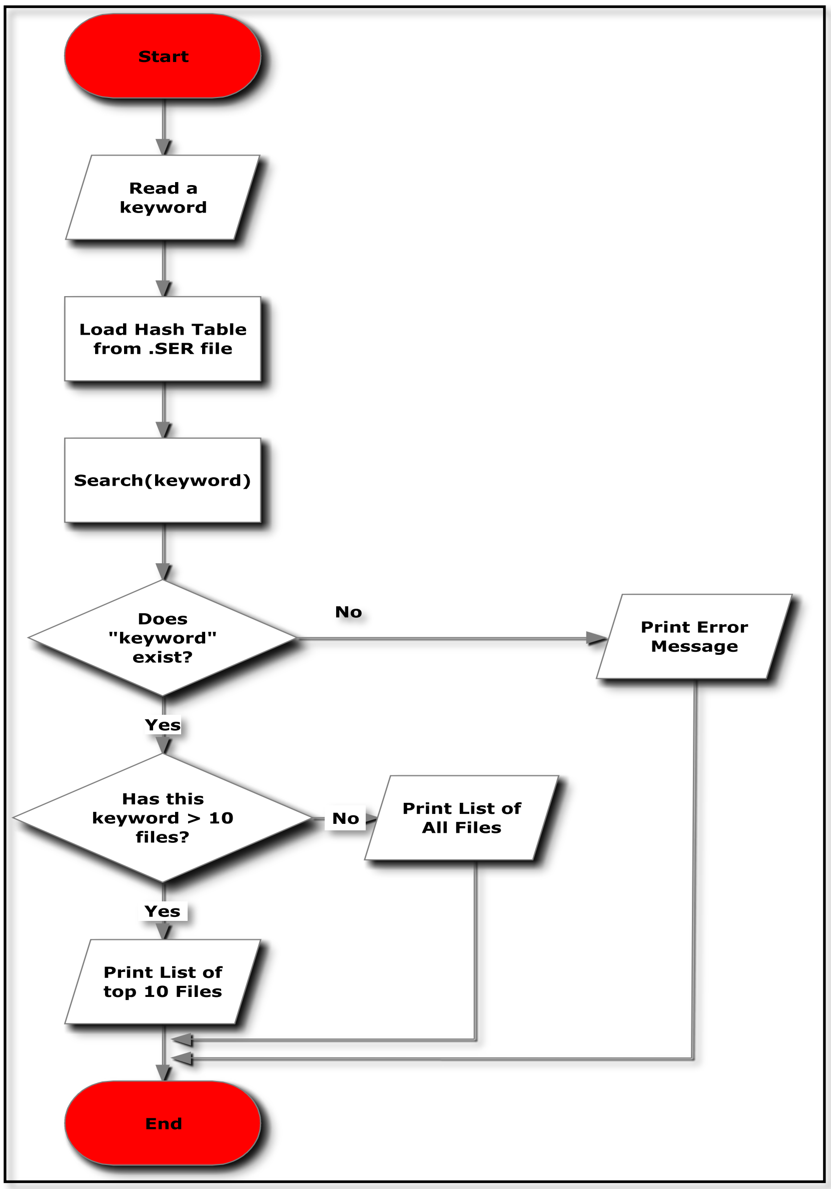


Figure 4.6: Search in a Hash Table Flowchart

SearchInterface class and SearchHash class are used for searching in the hash table. The user uses the interface to enter the keyword and to get the list of files (the result of the searching). However, the SearchInterface class uses SearchHash class for the actual searching. Codes are provided in Appendix D, 6.SearchHash.java and 7.SearchIntrface.java.

4.2.5 Other Classes

There are many other classes used in coding such as Link class, Link1 class, UnusedList class and SortedList class. All commented source code is available in Appendix D.

4.3 Completed Integrated System Deployment

The below figure summarizes the previously discussed classes in section 2.4. It shows the complete integration of whole system. Starting from generating the testing input data, passing through the indexing phase, and ending with the user interface.

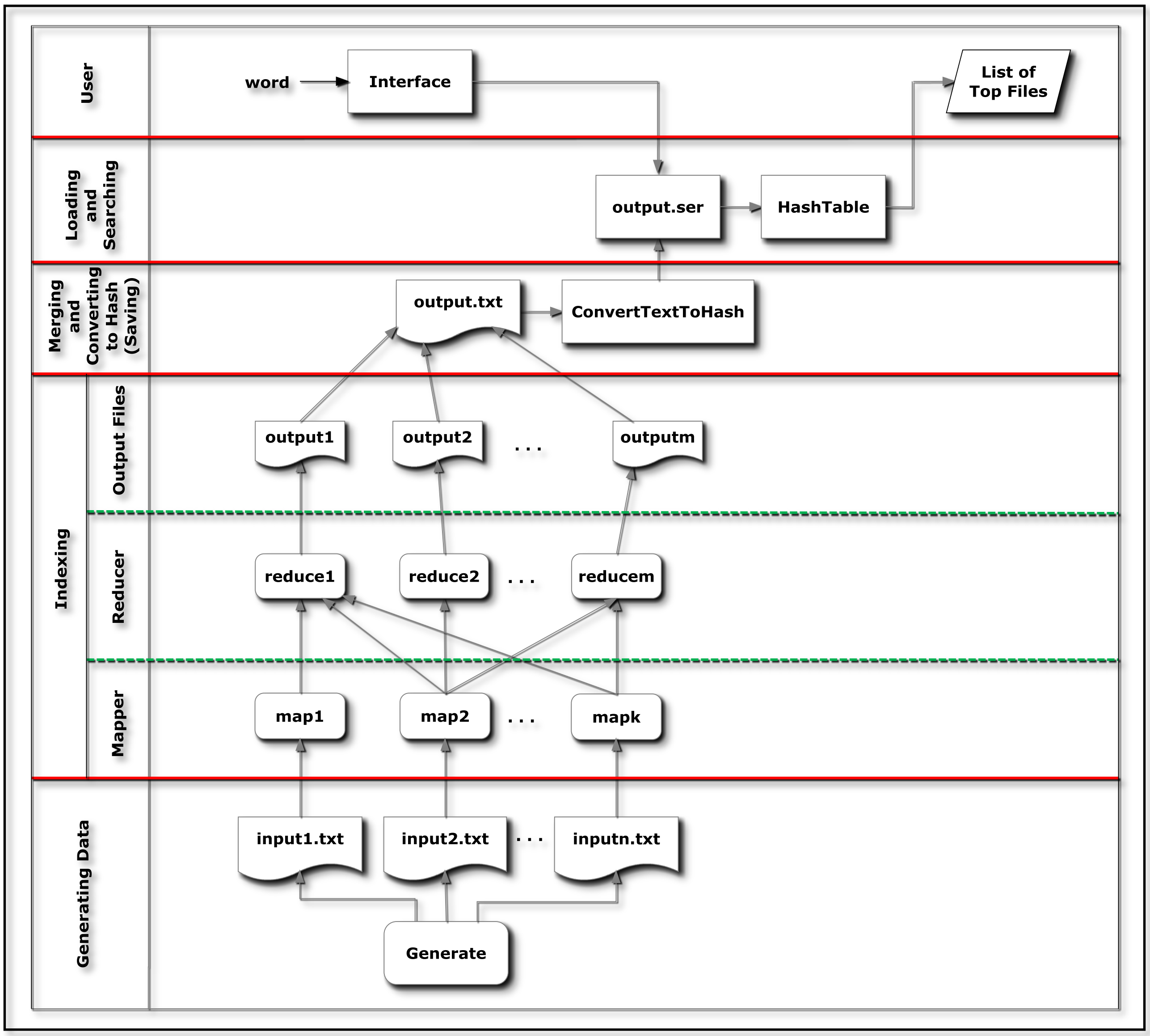


Figure 4.7: Completed Integrated System

4.4 Computer Environment Deployment

This section discuses the needed platform for running the system. First of all, the virtualization system, which is the blue cloud 1.6. As Qatar University students , we used the QU cloud. Second, the Hadoop platform, which should be installed and configured on all the VMs (nodes) in the project on the cloud. In addition, Cygwin, which is a Linux like environment, is needed to run some commands on the Hadoop platform. Also, Java Development Kit (SunJDK 1.6.0_06) or Java Runtime Environment(JRE 1.6) virtual machines are needed in order to run the system. For further information, refer to section 3.3.1.1, 3.3.2.2, 3.3.2.3 and 3.3.2.6 .

4.5 Software Installation and Usage

This section provides the methodology of software installation and usage of the system. The following steps are required to run the system correctly:

1. Be sure to have a custom project on the cloud with 8 VMs each with size at least 80 GB. Each VM should have 3 CPUs for better performance.
2. Download and install on of Java VM (JDK1.6 or JRE1.6)
3. Download Eclipse 3.2 EUROPA.
4. Download and install Cygwin.
5. Download a copy of the used Hadoop on the cloud to your local machine.
6. Configure your local machine by adding the required environment variables and the needed hosts to the defined hosts in your machine.
7. Add the Hadoop plugins to Eclipse.
8. Create a new Hadoop location.
9. Test the connectivity between your machine and HDFS on the cloud.
10. Create a new MapReduce project and extract the given project(saved on the CD, called “FinalSeniorProject”) to it.
11. Create a new directory on the HDFS for the input files and upload them on this directory.
12. Run the LineIndexer.java on the Hadoop location that you have created. Be sure to include the path of the input folder in the main of the line indexer, and specify the output path.
13. The output is 11 text files, each reducer produces a separate output file.
14. Download these files to a new directory in your local machine.
15. Merge them by using any merging software such as TXTcollector, which is a free merging tool.
16. The output of this program should be your input to the ConvertTestToHash class.
17. Run ConvertTestToHash class. Do not forget to put the path of the input file in the main method. This running should output a output.ser file.
18. Put output.ser file in “C:/ “.
19. To run the search application, you have two methods.
    a. Run the SearchInterface.java. Then, the interface will appear.
    b. Copy Searching.jar file from the given CD to “C:/ ” on your local machine. Then go to your command line and change the directory to “C:/”. Next, write this command “java –jar Seaching.jar”. Then, the interface will appear.
20. Now, you can search for any word included in the input files.

For a clearer and more detailed instruction of installation and usage, see Appendix A and Appendix B.

CHAPTER 5
TESTING AND EVALUATION

5.1 Introduction

This chapter is concerned with a very critical step software engineering designers do after analyzing, designing, and implementing their system. This step is testing. Testing is finding out the difference between the expected system's behavior and the implemented system's one. The purpose is to find out faults in the implemented software in a panned manner. [31] In this chapter, the testing plan is presented, then the different testing stages are elaborated including unit testing, integrated testing system testing, and performance testing. Finally, an evaluation of the system and its impact on both computing and economy and society is stated.

5.2 Test Planning

One of the success keys for testing is planning ahead what should be tested and when. Test planning should occur early, i.e. in the development phase so that we would have sufficient time and skills for testing [31]. From that perspective, we had our own testing plan and for each project piece, testing was scheduled. First, the generated code was tested individually, then testing the whole system was the next step and it was scheduled to be tested as figure5.1 shows:

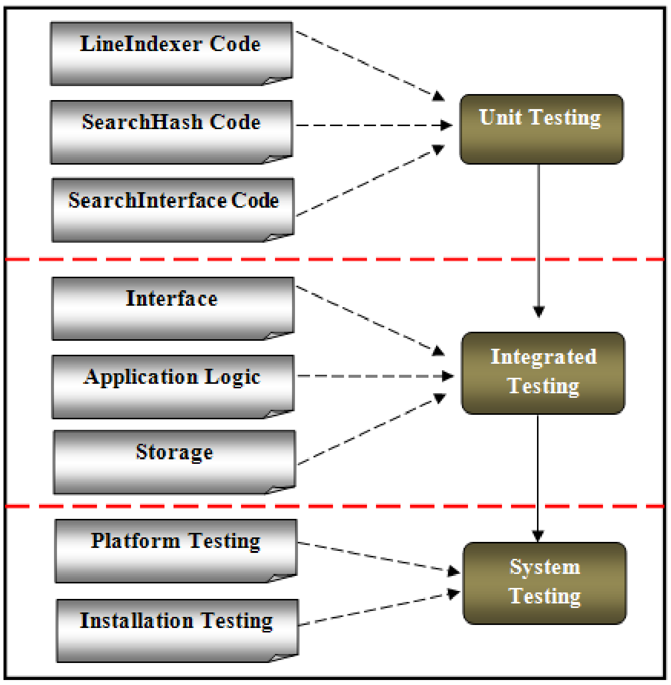


Figure 5.1: Test Planning

5.3 Generating Data Testing

Generating data code had to be tested in the applied system (Refer to Appendix D part 2.Generate.java for the code). Because it is a separate part, it should be tested using unit testing. The generation code is tested with state-based testing as the figure5.2 shows:

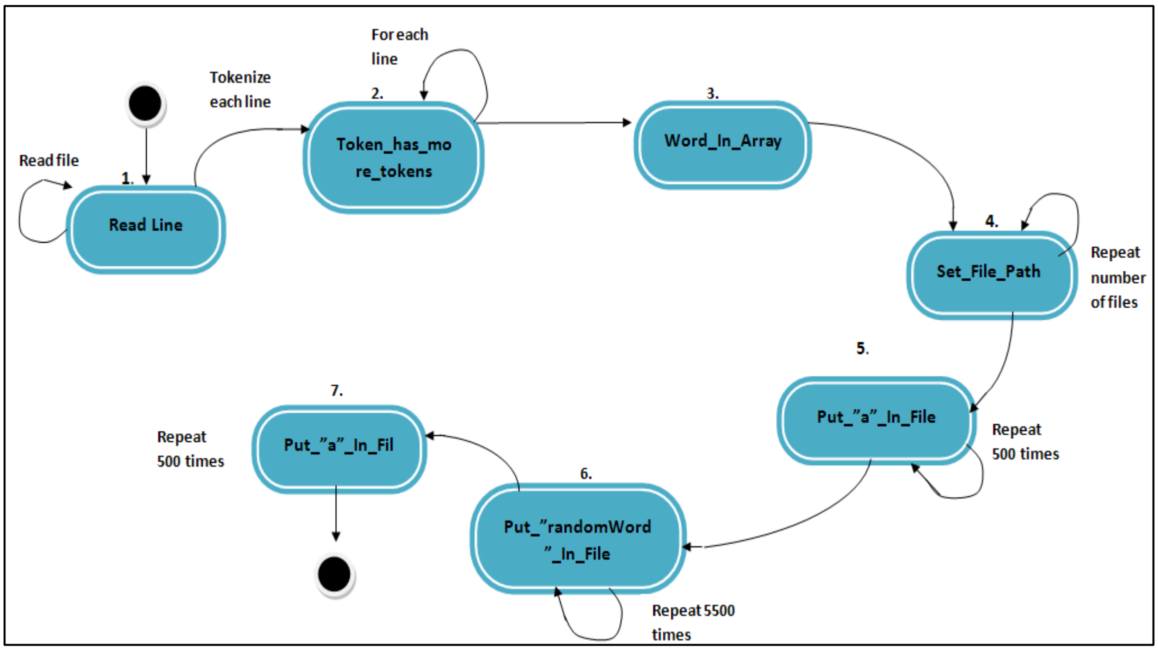


Figure5.2: State Chart Diagram for File Generation

After testing the Generate.java code. We decided to use it as a source to generate the needed data for our project after trying all possible planes previously mentioned in the report: crawler, QP, Al-Jazeera (refer to section 3.10 to look for possible plans for generating the data)

5.4 Unit Testing

Unit testing is a concept of decomposing the whole system into individual blocks. Each one of them is tested independently of the others. This type of testing is very powerful for lots of reasons: first, it reduces the system complexity. Second, it makes it easier to find out the faults for each individual block and third, it allows the designer to test lots of blocks in parallel without the need to wait for each piece to be individually tested [31].

Unit testing was utilized in our project mostly for code testing because the code was easily dividable. The code was decomposed into 3 main tasks. The first one is the LineIndexer code which performs the indexing MapReduce algorithm, the second task is the searchHash where we create a hash from a file and we serialize it and search in it. Finally, the interface code where we use the Java Swing library to provide a simple search interface for the user.

5.4.1 LineIndexer Code

Starting with the LineIndexer code, and looking deeply into its details, we can notice that we can split it into two parts; the map and the reduce parts, and test each one separately using different methods.

The map was tested using the path testing (white box testing) where all possible paths in the code where investigated to define the faults in the map implementation. The bellow activity diagram describes how this test is performed on the map code and how it was implemented to interpret the map faults:

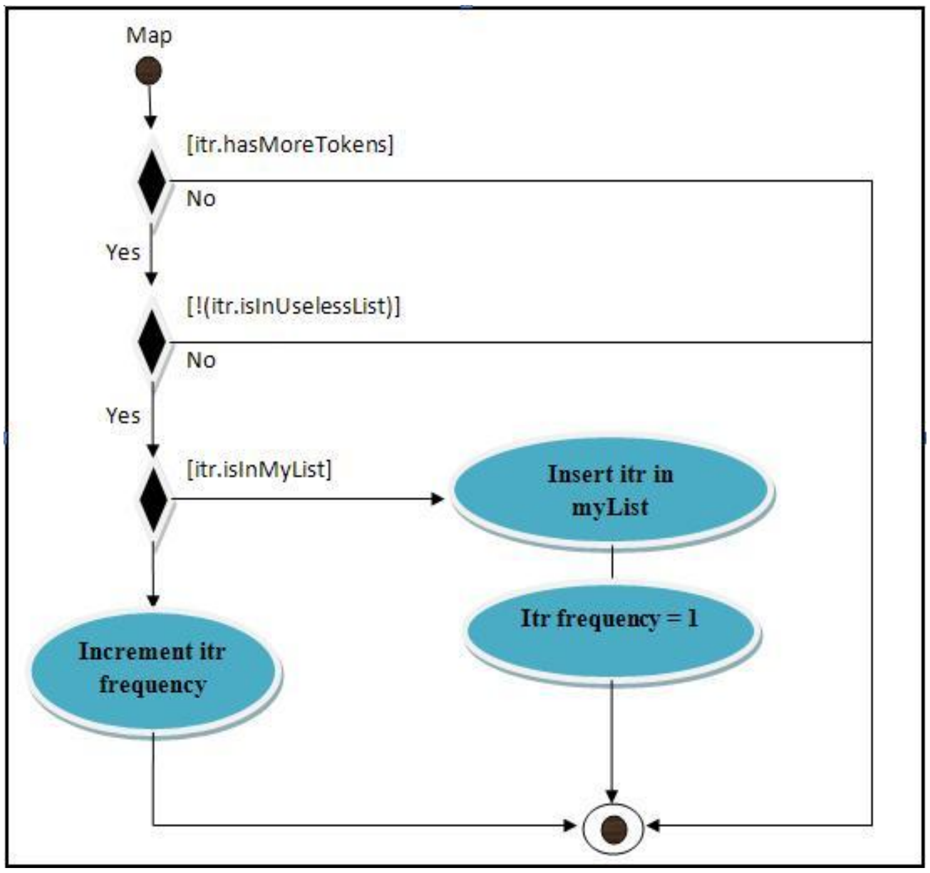


Figure 5.3: Equivalent flow graph for the Map implementation

From the previously shown figure5.3, we can notice that all the Map code paths are elaborated. Using it, we can determine a list of different cases and determine their corresponding paths referring each time to the activity diagram. Refer to table 5.1 to see all possible tested cases in the system for Map implementation.

Table 5.1: Test Cases and Their Corresponding Path for the Map Activity Diagram

Test Case	**Path**
(itr = cloud) // supposed cloud word only appear once	{cloud Filename%1}
(itr = is)	Go to next iteration (is is from unused list)
(itr = project) // supposed project word appeared 3 times in the file	{project Filename%3}
(itr = null) // supposed no words are left in the file	Go out of the map implementation

Another point to be mentioned in testing the map code is testing each part of the map individually. There are three concepts involved in the map: the link class, the SortedList class and the UnusedList class. For each class, some input samples were provided and the output was interpreted, so blackbox testing was exploited for each class used in the map class. One of the faults discovered during the UnusedList blackbox testing is that files are not supported to be used in the MapReduce programming model: we tried to let the unused words list from a file so it is easier to change those words but this did not work.

Moving to the reduce phase of the LineIndexer code, the method used to test this portion of the LineIndexer code is also blackbox testing. Blackbox testing's advantage is that it reduces the number of test cases: all possible inputs are partitioned into equivalent classes and a test case is used for each class [31]. In testing the reduce, we tried to put as input for reduce and output of the map a string instead of a text type but this did not work: reduce method only accepts text input format. Because the input for reduce is text type, special characters and numbers are accepted to be in the file name.

After testing the whole LineIndexer code (refer to Appendix D part 3.LineIndexer.java) and after verifying that it worked correctly and as expected, this code was used to index the set of input data that will be manipulated to apply a simple search interface.

5.4.2 SearchHash Code

In the searchHash code, different testing techniques are applied on the individual methods in the class. The put, containsKey and get methods are implemented from the ones provided by the Java HashTable library so we didn't have to spend time testing them. The ones that were tested are the wordSearch and the serializable methods. Looking at the wordSearch method: state-based testing is applied. It is a recent testing method that depends on comparing the resulted system states to the expected ones. For this purpose, the statechart diagram is used [31]. This technique is applied in our project as the next figure shows:

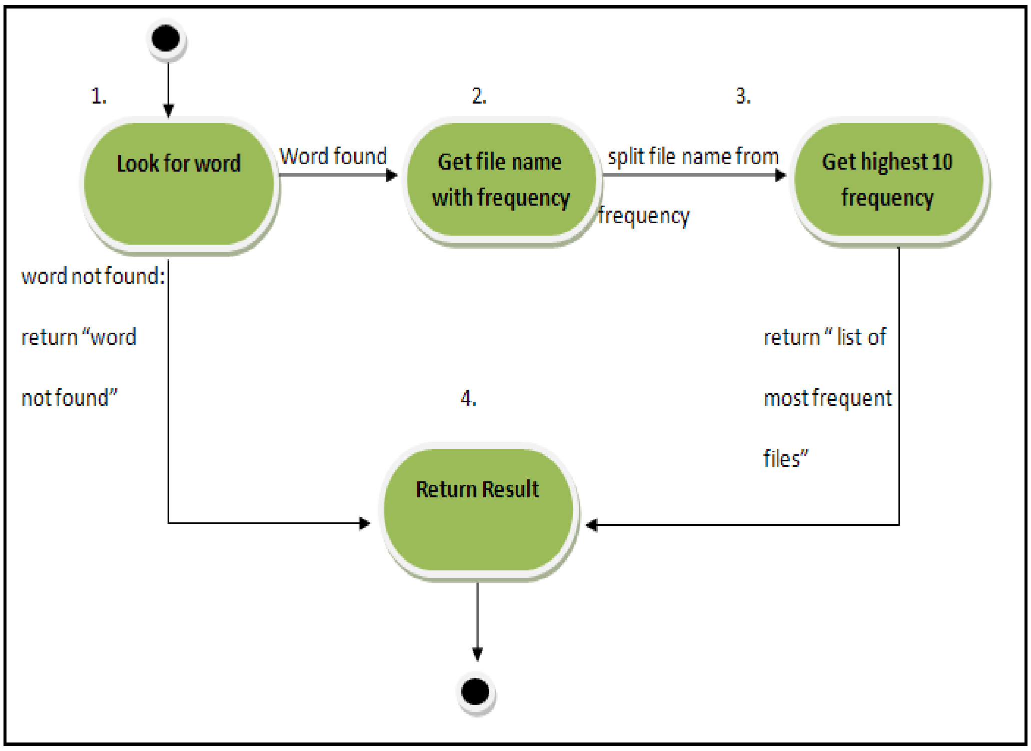

1.
Look for word
Word found
2.
Get file name with frequency
split file name from frequency
3.
Get highest 10 frequency
word not found: return "word not found"
return " list of most frequent files"
4.
Return Result

Figure 5.4: Statechart Diagram for wordSearch Function

Moving from the wordSearch method to the serializable method, this method was tested using blackbox testing. A set of possible inputs are partitioned into equivalent classes and each class was tested. For example a wrong path is put in the serializable output stream and the code could detect the error. Another example of testing is using a hash of different sizes and all of them were successfully converted into a serializable file so there is no hash size limit as long as we have a space in the disk to save the serializable file.

The previously tested part ,after verifying that it worked correctly, is used in the project to serialize the output of the index code. We will use precisely the code for serialization and for searching in hash tables.

5.4.3 System Interface

The system interface testing may be viewed from two diverse perspectives. First point to be tested in the interface may be following the unit testing technique. Another interesting point to be tested regarding the interface is the usability testing.

Starting with the unit testing. The interface code was tested this way and more precisely, it was tested using the blackbox testing. In this test, all possible input cases are tested. An example of that is trying to enter to the search window the same word in capital and small (where Note (1) is stated in figure 5.5) and the result was not the same. From that we could conclude that there was a fault in the code and we had to fix it, so when the input is entered, it is not case sensitive as shown in figures 5.6.

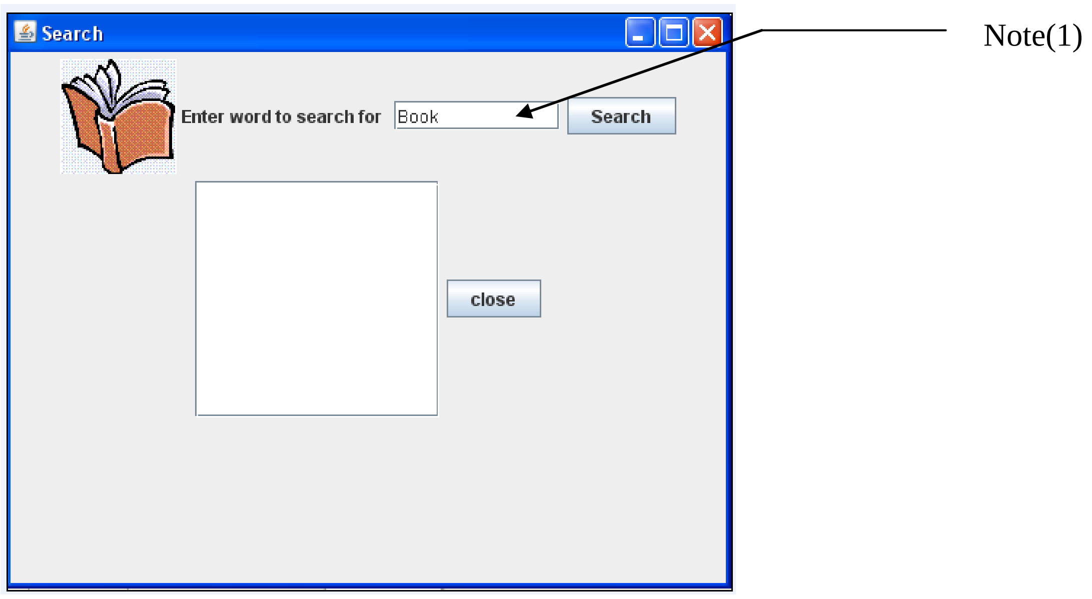
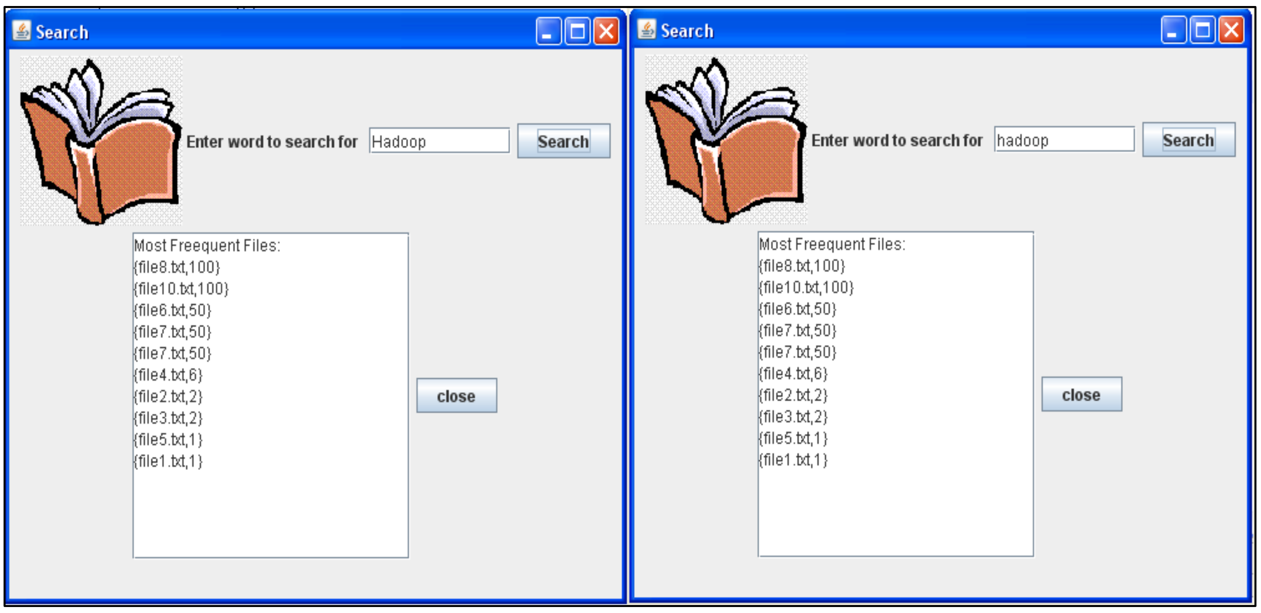


Figure 5.5: Sample of Screen Shot Interface

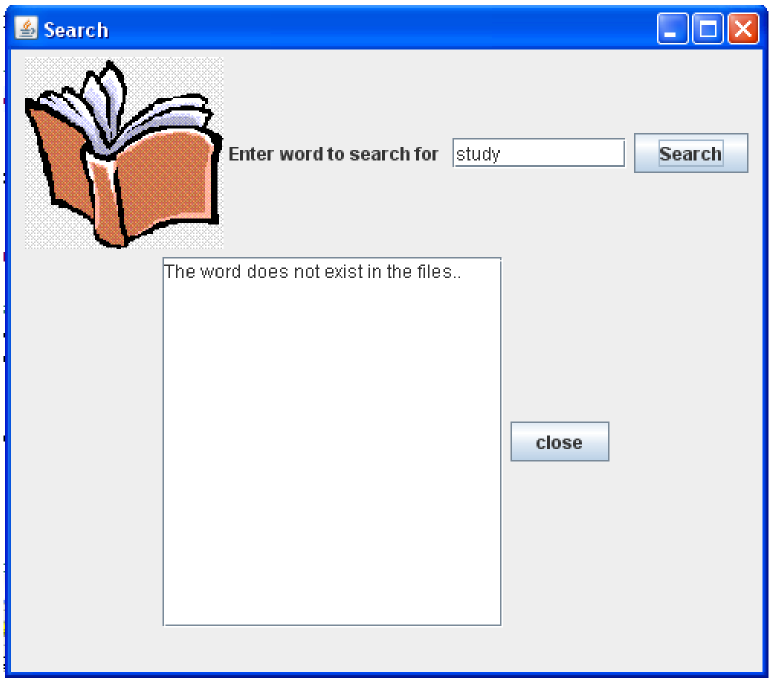


```
Most Freequent Files:
{file8.txt,100}
{file10.txt,100}
{file6.txt,50}
{file7.txt,50}
{file7.txt,50}
{file4.txt,6}
{file2.txt,2}
{file3.txt,2}
{file5.txt,1}
{file1.txt,1}
```

```
Most Freequent Files:
{file8.txt,100}
{file10.txt,100}
{file6.txt,50}
{file7.txt,50}
{file7.txt,50}
{file4.txt,6}
{file2.txt,2}
{file3.txt,2}
{file5.txt,1}
{file1.txt,1}
```

Figure 5.6: Sample of The Interface that Shows Not Case Sensitive Input

Another example about black box testing for the system interface , is entering a word that does not exist in the used indexed data, and the result was having an error. So problem was solved by printing a sentence " the word does not exist in the files" when the user enters unavailable word as shown in figure 5.7



The word does not exist in the files..

Figure 5.7: Output for unavailable word in the files.

Talking about the interface usability test, scenario testing method was chosen to be the type of usability test we are performing. Scenario test is a test where one or more users are presented to the system. This test allow designers to determine how much the system is usable and how the user deals with the system description [31]. In our project, we decided to apply this test by providing a read me brochure to a random user and observe how he interacts with the system. As a result, we got significant feedbacks about the system and those feedbacks were considered and were used to improve the system.

5.5 Integrated Testing

The integrated testing is a test type that focuses on figuring out system faults that are not well tested in the unit testing by focusing on small components of the whole system. It's methodology is very simple: first 2 components are integrated and tested and when no faults are found, additional components are added [31]. There are four types dedicated for integrated testing: big bang testing, bottom-up testing , top-down testing and finally sandwich testing. In our project (figure 5.8), the bottom-up testing was considered as the best testing technique to fit our hierarchal system design. In this specific strategy, each bottom layer component is tested individually then integrated with up layer components. In layer III, first the cloud environment (subsystem F) was exploited then the Hadoop distributed file system (subsystem E) is tested (refer to platform testing in section 5.5.1). Moving to the layer II, triple test was applied on the indexing application (subsystem C) integrated with the subsystem E and F. In the same time, search in hash (subsystem B) and serialize hash (subsystem D) is tested (refer to unit testing section 5.3.2). To finish, layer I is tested using quadruple test is applied where four subsystem were tested and integrated together: B, C, D and search user interface (subsystem A).

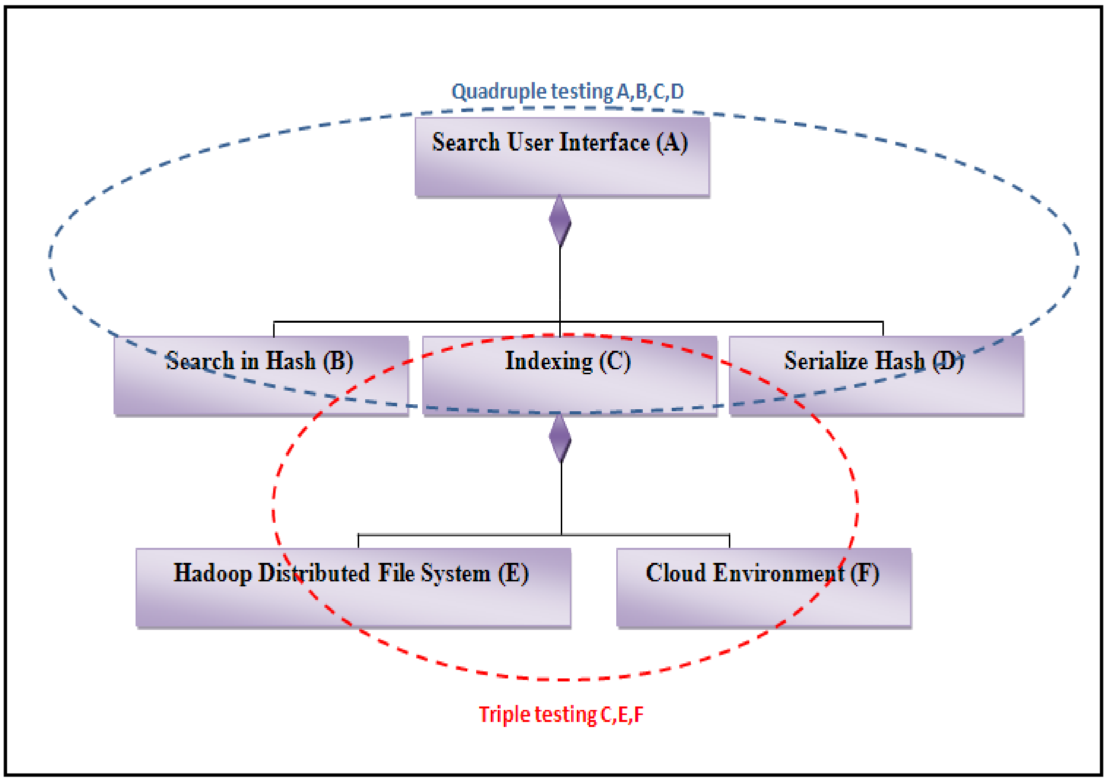


Figure 5.8: Bottom-up test strategy

5.6 System Testing

Unit and integrated testing focus on individual components and the interfaces between the components. After this step is performed, system testing is applied to ensure that the complete system complies with the specified functional and non-functional requirements [31]. The actions to be performed in the system testing in our project is composed from: platform testing, installation testing and performance testing.

5.6.1 Platform Testing

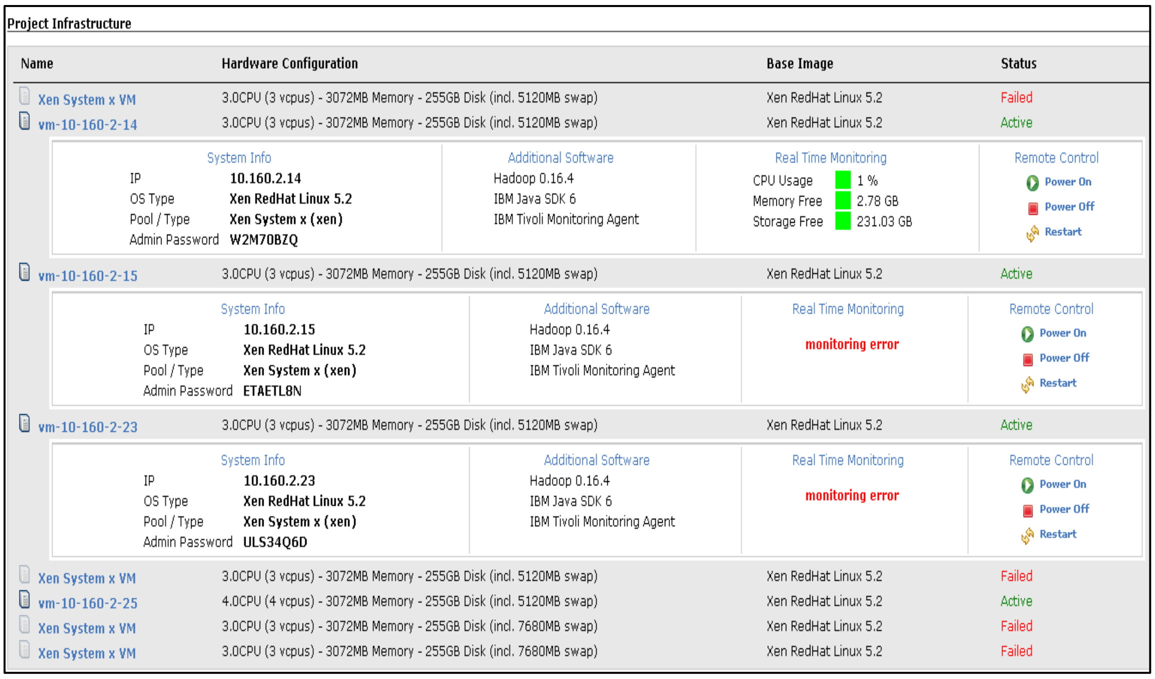
Platform testing is a method for testing used to test the used system platform. In our case, the platform testing is divided into two major sections: the Hadoop platform test and the cloud infrastructure testing.

5.6.1.1 Hadoop Testing

Because Hadoop was chosen to be our MapReduce programming platform, we had to make sure it is working as expected. The first thing done in this regards is installing the local Hadoop and test it on our machines. Starting with Hadoop version 0.16.4 till 0.20.1, all those versions were tested to be working fine. The problem appeared when trying to connect the eclipse software development environment with the different Hadoop versions. To solve this problem, online searches were performed (https://issues.apache.org/jira/browse/HADOOP-5225 is an example of the most useful websites to debug the Hadoop bugs). From the search done about Hadoop development, we figure out that all Hadoop versions need to be reconfigured and some bugs need to be fixed so Hadoop runs properly. The solution for this problem was getting a ready tested version to work on. CMU could provided for us a debugged version by the CMU system administrator. This was Hadoop 0.20.1 and it was tested on our cloud and the eclipse connection was also elaborated. To make sure that everything was working fine, WordCount code and the original LineIndexer code provided by yahoo was implemented as a MapReduce project and we ran it on our cloud location and it worked fine.

5.6.1.2 Cloud Testing

After testing the Hadoop platform and running the LineIndexer code on a small set of data, the same code was tested on larger set of data (1.6 MB). While observing the cloud status and the VMs usage when the code was running, we found out that we were running out of recourses and we need more CPUs and more storage space for each VM. First, we had to change the cloud project system specifications to support at least 250 GBs as shown in figure5.9 (6 VMs each VM with 255 MB). From this point, cloud problems started to showing up: using the cloud, we find out that the cloud status was not stable and the Hadoop was wrongly configured in our cloud. We also discovered some cloud limitations , as an example of them that the cloud interface can't support a project with 8 VMs each with 255GB. The solution of this problem was getting help from IBM corporation.


Project Infrastructure

Name	Hardware Configuration	Base Image	Status
Xen System x VM	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Failed
vm-10-160-2-14	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-15	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-23	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
Xen System x VM	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Failed
vm-10-160-2-25	4.0CPU (4 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
Xen System x VM	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 7680MB swap)	Xen RedHat Linux 5.2	Failed
Xen System x VM	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 7680MB swap)	Xen RedHat Linux 5.2	Failed

vm-10-160-2-14

System Info		Additional Software	Real Time Monitoring		Remote Control
IP	10.160.2.14	Hadoop 0.16.4	CPU Usage	🟩 1 %	Power On
OS Type	Xen RedHat Linux 5.2	IBM Java SDK 6	Memory Free	🟩 2.78 GB	Power Off
Pool / Type	Xen System x (xen)	IBM Tivoli Monitoring Agent	Storage Free	🟩 231.03 GB	Restart
Admin Password	W2M70BZQ				

vm-10-160-2-15

System Info		Additional Software	Real Time Monitoring	Remote Control
IP	10.160.2.15	Hadoop 0.16.4	monitoring error	Power On
OS Type	Xen RedHat Linux 5.2	IBM Java SDK 6		Power Off
Pool / Type	Xen System x (xen)	IBM Tivoli Monitoring Agent		Restart
Admin Password	ETAETL8N			

vm-10-160-2-23

System Info		Additional Software	Real Time Monitoring	Remote Control
IP	10.160.2.23	Hadoop 0.16.4	monitoring error	Power On
OS Type	Xen RedHat Linux 5.2	IBM Java SDK 6		Power Off
Pool / Type	Xen System x (xen)	IBM Tivoli Monitoring Agent		Restart
Admin Password	ULS34Q6D			

Figure 5.9: State of Nodes in the Cloud

5.6.2 Performance Testing

The following two sets of experiments are designed to study the of MapReduce parallelism in many aspects.

5.6.2.1 Correctness

Experiment 1: Testing the correctness of the indexing code

This experiment tests if the output of the LineIndexer is as expected or not. Testing Environment is shown in Table 5.2.

Table 5. 2: Testing Environment for Experiment 1.1

Constraints	Specifications
Input Data	Small, about 2 MB – 1 GB
No of Virtual Machines	6 VMs
Disk Size per VM	Could be from 120 GB - 255 GB
Memory (RAM) Size per VM	3072 MB
No of CPU for Data Nodes	3 CPUs
No of CPU for Name Node	4 CPUs

5.6.2.2 Response Time

In our case, we define response time to be the time that the MapReduce job takes to be terminated successfully.

Experiment 2.1: Study the impact of the number of VMs

For this experiment, three different cloud projects should be created (assuming the total available HDFS storage is 512 GB):

1. Two VMs (each of 256 GB)
2. Four VMs (each of 128 GB)
3. Eight VMs (each of 64GB)

Number of reducers tasks is 5 in this experiment. Other Specifications are shown in Testing Table 5.3

Table 5. 1: Testing Environment for Experiment 2.1

Constraints	Specifications
Input Data	256 GB
No of Virtual Machines	Vary
Disk Size per VM	Vary as shown in the described above steps
Memory (RAM) Size per VM	3072 MB
No of CPU for Data Nodes	3 CPUs
No of CPU for Name Node	4 CPUs

Experiment 2.2: Study the impact of the number of reducers

For this experiment, the same job should be run three times with three different no of reducers and the same other specifications, which shown in Table 5.4.

1. default (one reducer)
2. one wave (number of VMs * 0.95)
3. two waves(number of VMs * 1.75)[1]

Table 5. 2: Testing Environment for Experiment 2.2

Constraints	Specifications
Input Data	256 GB
No of Virtual Machines	6 VMs
Disk Size per VM	Could be from 120 GB - 255 GB
Memory (RAM) Size per VM	3072 MB
No of CPU for Data Nodes	3 CPUs
No of CPU for Name Node	4 CPUs

<u>Experiment 2.3: Study the impact of the input size</u>

In this experiment, the throughput is reported. The same job should be run three times with three different sizes of the input with the same other specifications, which shown in Table 5.5.

1. One GB
2. 256 GB
3. Half TB

Table 5. 3: Testing Environment for Experiment 2.3

Constraints	Specifications
Input Data	Varies
No of Virtual Machines	6 VMs
Disk Size per VM	Could be from 120 GB - 255 GB
Memory (RAM) Size per VM	3072 MB
No of CPU for Data Nodes	3 CPUs
No of CPU for Name Node	4 CPUs

<u>Experiment 2.4: Study the impact of the number of input files</u>

For this experiment, the same job should be run three times with three different no of files. The total input size is equal for trials. Table 5.6 shows the relation between number of files and the size for each file. Other specifications are shown in Table 5.7.

Table 5.4: Relation between Number of Files and Size of Each Size

Size of Each File	**No of Files**
04 MB	65,536

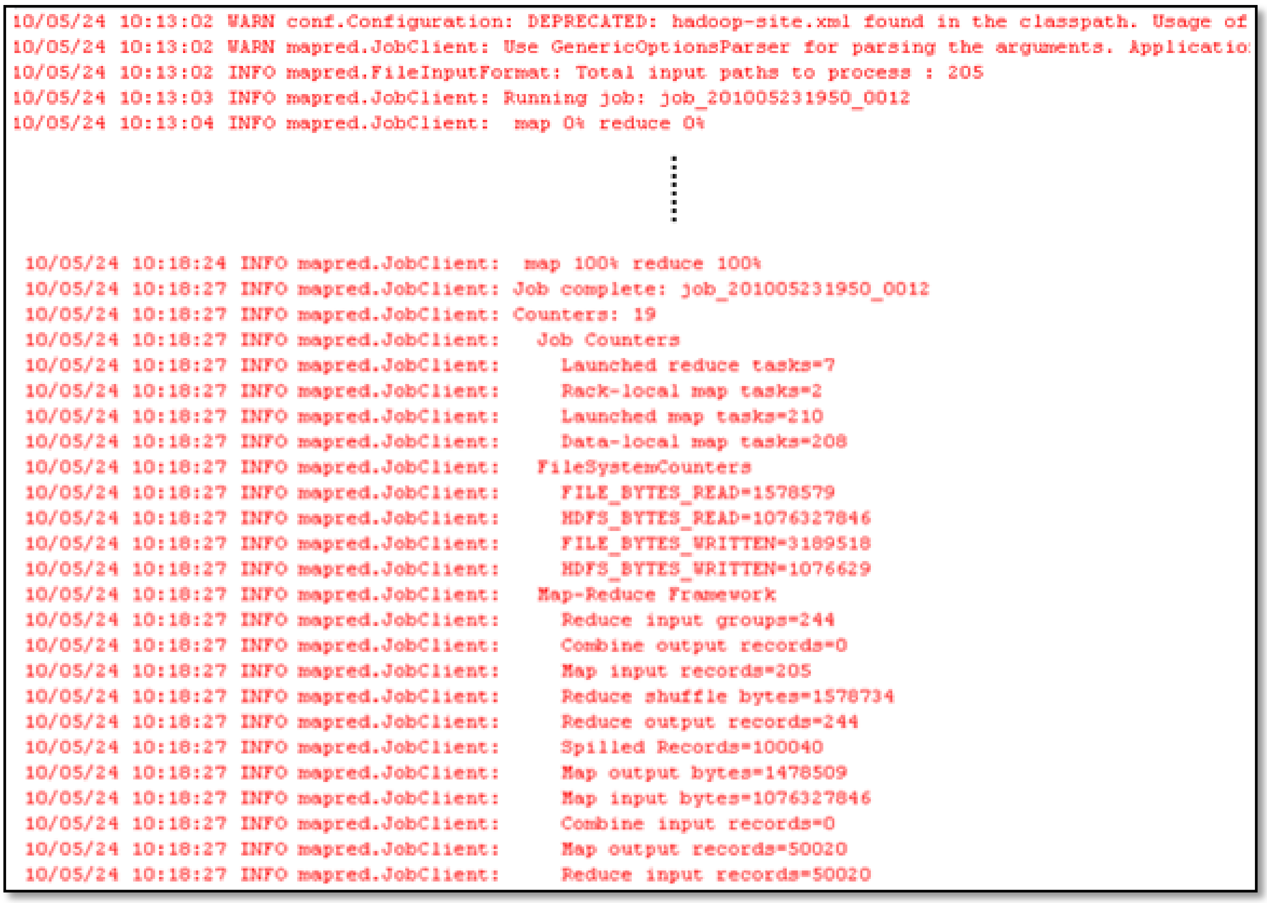
[1] Refer to section 3.6.2.2.1

16 MB	16,384
64 MB	4096

Table 5.5: Testing Environment for Experiment 2.4

Constraints	Specifications
Input Data	Small, about 2 MB – 1 GB per file
No of Virtual Machines	6 VMs
Disk Size per VM	Could be from 120 GB - 255 GB
Memory (RAM) Size per VM	3072 MB
No of CPU for Data Nodes	3 CPUs
No of CPU for Name Node	4 CPUs

With the time limitation and problems faced with the cloud, see section 5.6.1.2, we could only do Experiment1 and a part of Experiment2.2. The expected output from experiment 1 was produced after the job was terminated successfully.



```
10/05/24 10:13:02 WARN conf.Configuration: DEPRECATED: hadoop-site.xml found in the classpath. Usage of
10/05/24 10:13:02 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applicatio
10/05/24 10:13:02 INFO mapred.FileInputFormat: Total input paths to process : 205
10/05/24 10:13:03 INFO mapred.JobClient: Running job: job_201005231950_0012
10/05/24 10:13:04 INFO mapred.JobClient:  map 0% reduce 0%

                                     ⋮

 10/05/24 10:18:24 INFO mapred.JobClient:  map 100% reduce 100%
 10/05/24 10:18:27 INFO mapred.JobClient: Job complete: job_201005231950_0012
 10/05/24 10:18:27 INFO mapred.JobClient: Counters: 19
 10/05/24 10:18:27 INFO mapred.JobClient:   Job Counters
 10/05/24 10:18:27 INFO mapred.JobClient:     Launched reduce tasks=7
 10/05/24 10:18:27 INFO mapred.JobClient:     Rack-local map tasks=2
 10/05/24 10:18:27 INFO mapred.JobClient:     Launched map tasks=210
 10/05/24 10:18:27 INFO mapred.JobClient:     Data-local map tasks=208
 10/05/24 10:18:27 INFO mapred.JobClient:   FileSystemCounters
 10/05/24 10:18:27 INFO mapred.JobClient:     FILE_BYTES_READ=1578579
 10/05/24 10:18:27 INFO mapred.JobClient:     HDFS_BYTES_READ=1076327846
 10/05/24 10:18:27 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=3189518
 10/05/24 10:18:27 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=1076629
 10/05/24 10:18:27 INFO mapred.JobClient:   Map-Reduce Framework
 10/05/24 10:18:27 INFO mapred.JobClient:     Reduce input groups=244
 10/05/24 10:18:27 INFO mapred.JobClient:     Combine output records=0
 10/05/24 10:18:27 INFO mapred.JobClient:     Map input records=205
 10/05/24 10:18:27 INFO mapred.JobClient:     Reduce shuffle bytes=1578734
 10/05/24 10:18:27 INFO mapred.JobClient:     Reduce output records=244
 10/05/24 10:18:27 INFO mapred.JobClient:     Spilled Records=100040
 10/05/24 10:18:27 INFO mapred.JobClient:     Map output bytes=1478509
 10/05/24 10:18:27 INFO mapred.JobClient:     Map input bytes=1076327846
 10/05/24 10:18:27 INFO mapred.JobClient:     Combine input records=0
 10/05/24 10:18:27 INFO mapred.JobClient:     Map output records=50020
 10/05/24 10:18:27 INFO mapred.JobClient:     Reduce input records=50020
```

Figure 5.10: Snapshot of the Console for Running a MapReduce Job (1 GB- 5 Reducers)

Also, the output was examined by complete the whole process until using the interface and it succeeded.

In Experiment 2.2, we could not use 256 GB as input data due to the same limitations. Input data of 1GB, 5 GB and 25 GB could be tested. Regarding 1 GB input data, when 1 reducer task was tested, the job was finished after 7 minutes, resulting with 1 output indexed file. Using wave 1 that applied 5 reducer tasks resulted with 5 output indexed files within 5 minutes and 25 seconds. But when 11 reducer tasks representing the wave 2 were used, the output was 11 indexed files for the inputted 5G , and the output resulted within 37 minutes and 48 seconds. The results for 1G input are shown in figure 5.11.

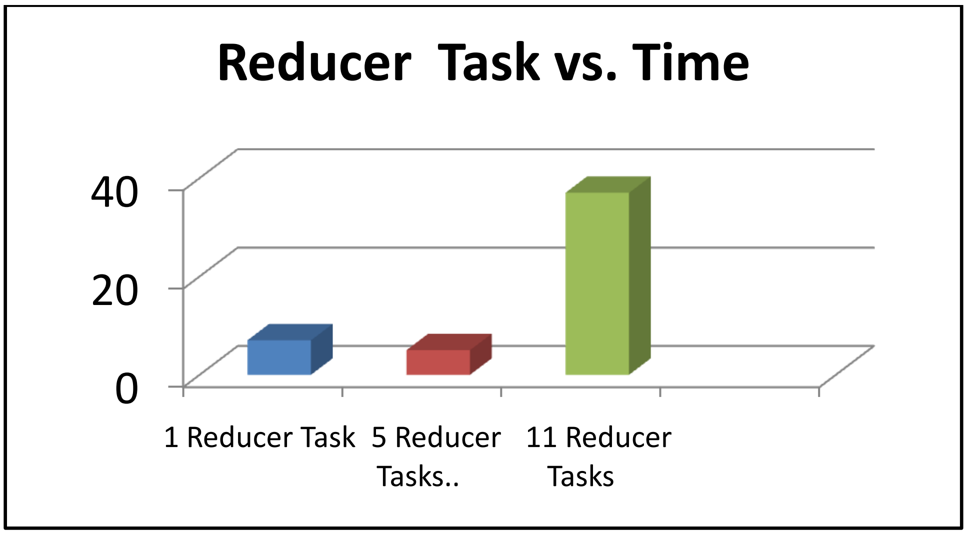


Figure 5.11: Reducer Task vs. Time when data input is 1GB

While 5 GB was used as input data, when 1 reducer task was tested, the job was terminated after 35 minutes. Using wave 1 that applied 5 reducer tasks finished within 32 minutes. On the other hand, when 11 reducer tasks representing the wave 2 were used, the output resulted within 270 minutes. The results for 5G input are shown in figure 5.12.

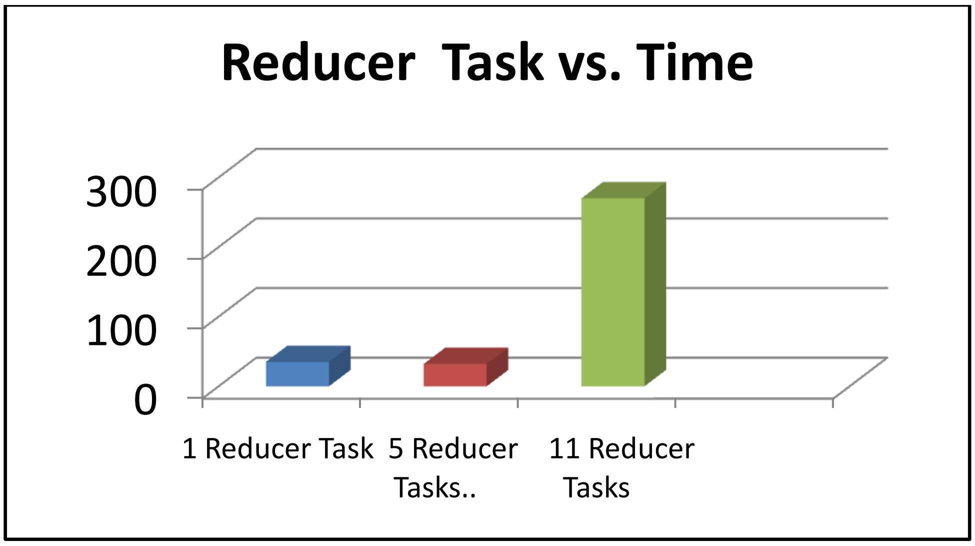


Figure 5.12: Reducer Task vs. Time when data input is 5GB

Using 25 GB as input data, the indexing program using 1 reducer was terminated after 12 hours, 23 minutes and 40 seconds, In addition to that, it was ended within 2 hours,6 minutes and 4 seconds when 5 reducer tasks were used. But when 11 reducers were applied, the job finished after 4 hours and 4 minutes. The results for 25G input are shown in figure 5.13.

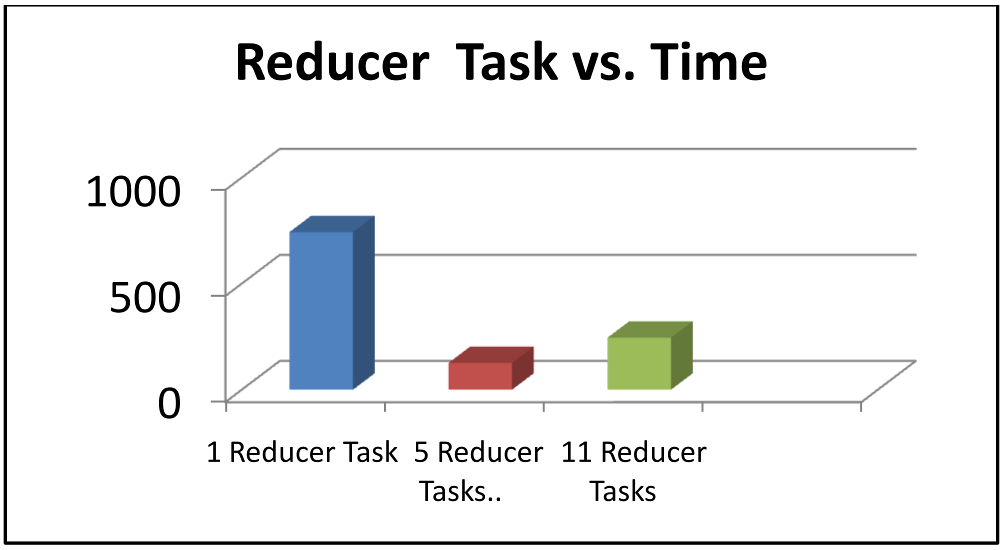


Figure 5.13: Reducer Task vs. Time when data input is 25GB

Depending on the previous results, and referring to section 3.6.2.2.1 ,we think that the used input data is not that enough to test the efficiency of using wave 2 compared to the efficiency of using wave1. So when we are going to complete the testing plan, huge sets of data as described before will be used.

5.7 System Evaluation with Respect to its Impact on Computing Environment and Society

5.7.1 Computing Environment

This project provides interactions between too many fields since it deals with analyzing huge sets of data using a recent technology presented by the cloud hardware. The designed project cloud be improved to work on oil and gas data sets instead of using generated data. The suggested plan would contribute very much in developing the abilities of relevant and important areas in search such as oil and gas production environment using data mining in the designed cloud computing model.

The previously suggested idea could be applied if QP plan was achieved but since the plan changed because Qatar Petroleum (QP) didn’t accept to provide their sensitive data to the project, and the project become data analysis using MapReduce Model on the cloud. This plan has a role in the field of cloud computing in Qatar and specially Qatar University where it is a new implemented technology. Although there are some limitations in using the cloud, over time it’s going to be more mature, so specialized people can use more features and create new applications which can contribute in many other fields in Qatar and the middle east. Furthermore, making a small search interface from the indexed data contributes in the search field, particularly if the project is improved to handle Arabic Language content.

5.7.2 Economy and Society

Since this project is implemented within the scope of some important areas like cloud computing and searching, then it takes the impact and influence form those areas in economical and social fields. Looking for the economical impact of the used hardware in the project, blue cloud 1.6 brings hardware cost lower. A cheap terminal with a keyboard and a mouse would be sufficient to perform the same tasks that expensive large hardware requirements for each client including large memory and fast CPUs can achieve. Cloud computing will also offer faster time to market: companies will have the ability to deploy applications in small period of time without changing the code ultimately which enables them to begin making a profit quicker. Although cloud computing has these positive impacts, local industry is not ready yet to adapt this kind of technology. Hopefully over time, it is expected to improve and organizations will realize how much it can provide benefits and profits to their work. In addition to that, this project has impacts on the society because it provides a small user interface for searching which enables any user to search easily for certain word, and it would provide higher impact if the system improved to deal with Arabic Language.

Another benefit of this project is that it increases the collaboration between multiple organizations in and outside Qatar (e.g., QU,CMU-Qatar, IBM organization). As a result of that, efforts can be gathered to create new valuable projects. Finally, this project is considered an important first step in exploring the cloud computing area in Qatar University, Qatar, the gulf, and middle east.

CHAPTER 6
CONCLUSIONS AND FARTHER WORK

6.1 Introduction

This chapter presents project conclusions. Also, the chapter highlights the challenges faced strengths, and weaknesses in the work done for the project. Finally recommended improvements and further work are identified to add more value to the project.

6.2 Main Conclusion

Data analysis using MapReduce programming model on the cloud project highlighted how the cloud computing technology can influence the performance of useful queries. In this project we were able to show the effectiveness of cloud computing model on search applications using parallel indexing. Motivated by the need to process large datasets, MapReduce programming model incorporates very easy and clean concepts that programmers don’t have to worry about the complex problem of designing, implementing and managing parallel code. Capitalizing on these ideas, the project designs an affective and useful document search application that employs an index file produced through parallel (MapReduce- and cloud- based) processing of documents. A remarkable result that was experienced during the system architecture is that even small design choices may significantly effect the application performance. Accomplishing the previously described goals opened the opportunity for effective collaboration between different universities (QU and CMU) which was a great experience and a change for exchanging knowledge.

Although the project could achieve most of the its desired goals, extensive performance testing did not have the chance to be completed and this was because of the various problems described in the challenges section (6.3). However, as this senior project is a part of a Qatar National Research Fund project supported under the Undergraduate Research Experiences Program (UREP), we will have the chance to continue this work and complete the performance testing. In addition to that, further improvements can be put in consideration to improve the system outcomes in the coming future.

6.3 Challenges and Anticipating Approach for each Challenge

This has been a very challenging open-ended but (at the same time) very rewarding project. As part of our ongoing project and with a strong intention to achieve the project objectives, a list of challenges from different perspectives was confronted. Some problematic issues showed up during disparate project progress stages. Nevertheless, in each and every challenging step, an alternative viable approach was taken to proceed further in the project schedule.

The first major challenge associated with our project was exploring the cloud computing model. In essence, the major concern was the project's main topic "cloud computing" which is very recent field the world is exploring. In fact, our project is considered as one of the first university experiences, in the gulf region, dealing with cloud computing. Qatar University cloud was installed on the 13th of December 2009 and that leaded to another challenge: configuring the IBM cloud was at the end of the semester and, as a result, we had to find and depend on an alternative resource, which was the Carnegie Mellon University (CMUQ) cloud. Using CMUQ cloud was loaded with several and long configuration steps to access. In addition to that, it handled numerous unforeseen problems which consumed a lot of time to solve and was, as a matter of fact, accomplished after more than one month of hard work.

After successfully accessing the cloud, the next challenge was exploring the cloud environment and how it works including some technical issues and configuration steps. Help and guideline resources were very limited and the online documentations were the only possible and available help we could get. However, following the online instructions was full of ambiguous parts and a lot of time was spent trying to understand and figure out those parts.

The trouble with cloud computing is that it encompasses such a huge range of technology offerings and one of them is Hadoop. After exploring the cloud environment, the next task was to deal with the Hadoop platform. The aim was to connect Hadoop with the Eclipse software development environment installed in our machines and run a MapReduce project to experience the use of Hadoop in a MapReduce model. This task was first accomplished in the CMU cloud because of some technical issues with the QU cloud but also a large amount of time was spent with Mr. Brian Gallew, the CMU system administrator, to make this thing works. Switching to QU cloud, the greatest help of this part was from Mr. Alfredo Cappariello, IBM cloud computing IT Specialist, who provided valuable support in solving this problem by using IBM plug-ins instead of Apache plug-ins of the Eclipse for the Hadoop. After this step was accomplished, configuring the installed Hadoop for our purposes was the next challenge. Qatar University IT team did a great effort in this regards, yet this also took a long period of time to be fixed.

The next point to be mentioned regarding this section is the source of data and data collection part. First, historical databases generated by Oil and Gas Production Systems (OGPS) by Qatar Petroleum (QP), since 30 years, was planned to be used to answer useful queries about trends and patterns of gas reservoirs and oil fields using the IBM Blue Cloud 1.6 at Qatar University. Unfortunately, this part of the project did not complete successfully. The reason is the long time process it took to convince QP Company to trust providing their sensitive data to the project. For that, it would be a brave step from QP to offer their OGPS data to the private IBM cloud at QU. Our alternative plan was to use a crawler code and run it to gather at least half terabyte of data size. The crawler that we used was not properly documented, which was the main reason for us to take a long period understanding it. However, with the help of Dr. Sayed Ahmed Hussein, instructor at QU, we could fix the code and run it. The other thing about the crawler is that it was very slow(400 megabyte per day at most), so we thought of another alternative. Al-Jazeera Networks was the third option we had. This company is rich of Arabic txt documents that can be beneficial to our project. Again this process took longer than expected! Our last plan was to generate data by writing a java code.

After generating the data, we decided to run it on our cloud (at least1GB). With this large set of data, IT needed to generate a new project with new 8 VMs that support at least 40GB but they did know how to set this new project because of lost of cloud problems and miss-configuration.

6.4 Strengths and Weakness

Regardless of all the faced challenges during system development, there is a list of strengths and weaknesses points we can spot the light on. Starting with the strengths ones:

- One of the most important strengths in the project, which gives it a distinguishable value, is that it collects a wide range of technologies and standards. Looking deeply to the produced system, we can see that it includes: cloud computing and virtualization technologies, MapReduce programming model, Hadoop platform, data structure concepts, web crawling field, Linux OS commands and various secured remote access software (SSH, VPN…)
- The other strength point to be mentioned is applying a MapReduce programming model on the QU cloud to produce a useful and effective query which is indexing. With the help of MapReduce, processes are distributed using Hadoop platform and as a result, they will be processed in a very fast manner.

Moving now to the weakness points that the system suffer from is:

- The first point to be considered as a weakness one when designing the system is the way of generating the data. After a long process of looking for real data, we decided to work on generated data from a JAVA code. Testing on those produced data was fine, but it would be more real life relevant and realistic to test on actual data.
- As we did the system interface GUI components because of time limitations, the produced system can't be accessed only when it is downloaded on the device. This access limitation can be overcome if designing the system as a web application: the access can be from even the web.

6.5 Suggested Improvement and Further Work

Even though data analysis of large set of data using a MapReduce model on the cloud project achieved the desired goals and objectives, we have to point out that there is some further work that can be done to improve the project. Our developed application opens the door for future improvements that can be directed to three different fields.

First, the improvements dedicated for us as the project developers:

- The data source would be replaced from generated txt files using JAVA code to actual data from institutions or companies and change the application to fed their needs and interests. A good idea is applying the application on Arabic data sets and use it to enrich the Arabic research field.
- The hash table used to store the indexed file use linear probing as a collision solution. A better idea is to use hash table that solves the

collision using the double hashing to increase the application performance.

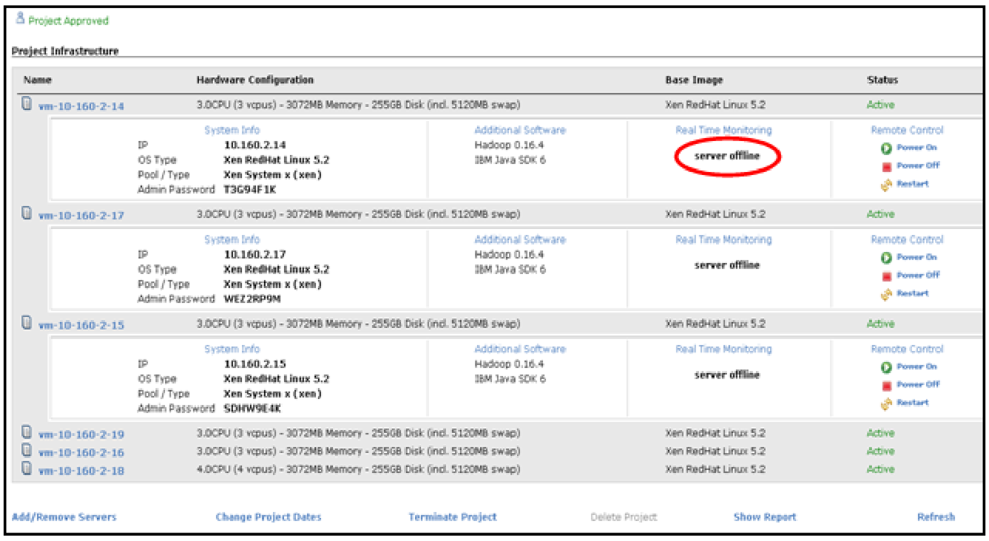
- In the LineIndexer code, a good practice would be changing the code so the user can enter the number of mappers and reducers and input and output paths. We can make it using GUI components or entering the arguments from the main () method.
- Another improvement in the search interface would be letting the user enter the path of the serializable file from the search window.
- The search interface is implemented using the GUI components. A remarkable improvement would be changing the search interface to be a web interface so the application can be accessed and shared within the web.

Second, the improvements dedicated for the MapReduce developers:

- In the map class inside the LineIndexer, the unused list words are taken from a method. An improvement in this regards would be letting the unused list take the unused words from a file so the user can change it easily without the need to go back to the code. This idea can’t be implemented in the map because reading from a file is not supported by map class. It would be very helpful for programmers if reading from file is supported by MapReduce algorithm.
- The reduce method stores the output in a Text file. It would positively affect the performance if we can store the output directly to a hash instead of saving it in a txt file then change it to hash. Also this feature is not supported by the MapReduce algorithm and it would give programmers more freedom when programming applications.

Third, the improvements dedicated for the Qatar University cloud developers:

- The QU cloud should be configured properly with stable state for improving and encouraging future cloud researches.
- The cloud features should be improved and be more usable. For example, the cloud interface does not support 8 VMs each of 255GB as Mr.Sajeer Thavot, Senior System Administrator in IT service, stated. Another thing needed to be fixed is the error message that appears in the cloud interface when there is no actual error which is “server offline” message as shown in the figure 6.1


Project Approved

Project Infrastructure

Name	Hardware Configuration	Base Image	Status
vm-10-160-2-14	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active

System Info		Additional Software	Real Time Monitoring	Remote Control
IP	**10.160.2.14**	Hadoop 0.16.4	**server offline**	Power On
OS Type	**Xen RedHat Linux 5.2**	IBM Java SDK 6		Power Off
Pool / Type	**Xen System x (xen)**			Restart
Admin Password	**T3G94F1K**			

Name	Hardware Configuration	Base Image	Status
vm-10-160-2-17	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active

System Info		Additional Software	Real Time Monitoring	Remote Control
IP	**10.160.2.17**	Hadoop 0.16.4	**server offline**	Power On
OS Type	**Xen RedHat Linux 5.2**	IBM Java SDK 6		Power Off
Pool / Type	**Xen System x (xen)**			Restart
Admin Password	**WEZ2RP9M**			

Name	Hardware Configuration	Base Image	Status
vm-10-160-2-15	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active

System Info		Additional Software	Real Time Monitoring	Remote Control
IP	**10.160.2.15**	Hadoop 0.16.4	**server offline**	Power On
OS Type	**Xen RedHat Linux 5.2**	IBM Java SDK 6		Power Off
Pool / Type	**Xen System x (xen)**			Restart
Admin Password	**SDHW9E4K**			

Name	Hardware Configuration	Base Image	Status
vm-10-160-2-19	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-16	3.0CPU (3 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-18	4.0CPU (4 vcpus) - 3072MB Memory - 255GB Disk (incl. 5120MB swap)	Xen RedHat Linux 5.2	Active

Add/Remove Servers | Change Project Dates | Terminate Project | Delete Project | Show Report | Refresh

Figure 6.1: The error message in the QU cloud interface

References

[1] T.White, *Hadoop The Definition Guide*. United States of America: O'Reilly Media ,2009,pp. 1-4,28 32,35

[2] "Hadoop on Demand," [Online document], 2008 Aug 20, [cited 2009 Oct],Available: http://Hadoop.apache.org/common/docs/r0.17.1/hod.html

[3] "Map/Reduce Tutorial," [Online document], 2009 Sep 1, [cited 2009 Oct],Available:http://Hadoop.apache.org/common/docs/current/mapred_tutorial.html

[4] Carnegie Mellon University, Computer Science department, Power point presentation for Dr.Majed,Lec-07,slide: 11-12,2010. Available: http://www.qatar.cmu.edu/~msakr/15319-s10/lectures/lecture17.pdf. [Accessed May 15, 2010

[5] Carnegie Mellon University, Computer Science department, Power point presentation for Dr.Majed,Lec-11,slide: 14-15,2010. Available: http://www.qatar.cmu.edu/~msakr/15319-s10/lectures/lecture11.pdf. [Accessed May 15, 2010

[6] Carnegie Mellon University, Computer Science department, Power point presentation for Dr.Majed,Lec-17,slide: 07,2010. Available: http://www.qatar.cmu.edu/~msakr/15319-s10/lectures/lecture17.pdf. [Accessed May 15, 2010

[7] "Cloud Computing," [Online document], , [2009 Oct-2010 May], Available: http://en.wikipedia.org/wiki/Cloud_computing

[8] Jonathan Strickland, "How Cloud Computing Works," [Online document], , [cited 2009 Dec 29], Available: http://communication.howstuffworks.com/cloud-computing.htm

[9] ["VPN Web Access", [Online document], , [cited 2009 Dec 17], Available HTTP: http://www.alpha-apr.com/vpn/

[10] "CloudBerry Explorer for Amazon S3 Screenshots" ,[Online document],,[cited 2009 Dec 29], Available:http://www.softpedia.com/progScreenshots/CloudBerry-Explorer-for-Amazon-S3-Screenshot-113427.html

[11] Tony Bain, "What is Hadoop", [Online document], 2008 Oct 15, [cited 2009 Dec 28], Available : http://blog.tonybain.com/tony_bain/2008/10/what-is-Hadoop.html

[12] "Introduction to Cloud Computing Architecture", [Online document],2009 Jun , [cited 2009 Dec 16], Available: http://www.sun.com/featured-articles/CloudComputing.pdf

[13] "Amazon_VPC," [Online document], , [cited 2009 Dec 28], Available: http://en.wikipedia.org/wiki/Amazon_VPC

[14] Platform as a Service,” [Online document], , [cited 2009 Dec 18], Available HTTP: http://en.wikipedia.org/wiki/Platform_as_a_service

[15] ”Hadoop,” [Online document], , [cited 2009 Oct 10], Available: http://en.wikipedia.org/wiki/Hadoop

[16] ” What is Hadoop? Big Data in the Enterprise”, [Online document], , [cited 2009

Oct 20], Available: http://www.vmware.com/appliances/directory/uploaded_files/What%20is%20Hadoop.pdf

[17] “HDFS Architecture,” [Online document],2009 Sep 1 , [cited 2009 Oct 13], Available : http://Hadoop.apache.org/common/docs/current/hdfs_design.html

[18] ”MapReduce,” [Online document],2009 Sep 1 , [cited 2009 Oct 13], Available : http://m.blog.hu/dw/dwbi/image/2009/Q4/MapReduce_small.png

[19] ”MapReduce,” [Online document], , [cited 2009 Dec 20], Available:

http://en.wikipedia.org/wiki/MapReduce

[20] ” 5-MapReduce Algorithms,” [Online document], , [cited 2010 May. 18], Available: http://Hadoop.apache.org/common/docs/current/mapred_tutorial.html

[21] [“Indexing definition,” [Online document],, [cited 2010 May. 18], Available: http://www.google.com.qa/search?hl=en&safe=active&defl=en&q=define:indexing&ei=BzjzS8PFKYjGrAfxtfnUDQ&sa=X&oi=glossary_definition&ct=title&ved=0CBMQkAE&safe=active

[22] [“Inverted Index,” [Online document],, [cited 2010 May. 18], Available:

http://en.wikipedia.org/wiki/Inverted_index

[23] ” Amazon Elastic MapReduce”, [Online document], , [cited 2009 Dec 28], Available HTTP: http://aws.amazon.com/elasticMapReduce/

[24] [”Amazon S3”, [Online document], , [cited 2009 Dec 28], Available HTTP: http://en.wikipedia.org/wiki/Amazon_S3

[25] ”Web Crawler”, [Online document], , [cited 2010 Apr 29], Available: http://en.wikipedia.org/wiki/Web_crawler

[26] ”What is a Web Crawler”, [Online document], , [cited 2010 Apr 29], Available: http://www.wisegeek.com/what-is-a-web-crawler.htm

[27] “WebCrawler”, [Online document], , [cited 2010 Apr 29], Available(http://java.sun.com/developer/technicalArticles/ThirdParty/WebCrawler/

[28] ”Overview”, [Online document], 2010 May 15 , [cited 2010 May 18], Available: http://crawler.archive.org/

[29] ”Web Crawling”, [Online document], , [cited 2010 May 18], Available: http://www.ics.uci.edu/~djp3/classes/2009_01_02_INF141/Lectures/Discussion02.pdf

[30] ”WebSPHINX: Project Web Hosting - Open Source Software”, [Online document], , [cited 2010 May 18], Available: http://websphinx.sourceforge.net/

[31] B.Bruegge and A.H Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, 2nd ed . United States of America: Pearson Education ,2004,pp. 125, 435-438,440,452,453,459,463,469

[32] Mr. Alfredo Cappariello, the cloud computing software engineer from IBM Innovation Centre in Dublin,”IBM Training “. [.ppt]. Qatar University – IBM Training , 2009

[33] ”Java Programming Language,” [Online document], 2010 May,[cited 2010 May 21], Available: http://en.wikipedia.org/wiki/Java_(programming_language)

[34] “JDK”,[Online document],2001 June,[cited 2010 May 21], Available:http://isp.webopedia.com/TERM/J/JDK.html

[35] “Cygwin”,[Online document], 2003 Sep, [cited 2010 May 21], Available:http://searchenterpriselinux.techtarget.com/sDefinition/0,,sid39_gci922130,00.html [36]

[37] Cloud Standards Effort Cloud Turn into a Dustup,” [Online document] , 30 April 2009 at 8:18, [cited:2010 may 21], Available :http://blogs.wsj.com/digits/2009/04/30/cloud-standards-effort-could-turn-into-a-dustup/

[38] ”DMTF Open Cloud Standards Incubator ,” [Online document] , 2010, [cited:2010 may 21], Available http://www.dmtf.org/about/cloud-incubator

[39] ”Cloud standards overview ,” [Online document] , 17 May 2010 at 14:09 , [cited:2010 may 21], Available: http://cloud-standards.org/wiki/index.php?title=Cloud_standards_overview

[40] ”Eclipse (software),” [Online document] , 12 May 2010 at 18:28, [cited:2010 may 21], Available :http://en.wikipedia.org/wiki/Eclipse_(software)

[41] ”Class JobConf,” [Online document] , 2009, [cited:2010 may 08], Available:http://Hadoop.apache.org/common/docs/current/api/org/apache/Hadoop/mapred/JobConf.html#setNumMapTasks(int)

[42] R.Lafore, *Data Structures & Algorithms in JAVA*, 2nd ed . United States of America: Sams Publishing ,2003,pp. 212-213

[43] ”Hash Table,” [Online document] , 10 May 2010 at 10:21, [cited:2010 may 23], Available: http://en.wikipedia.org/wiki/Hash_table

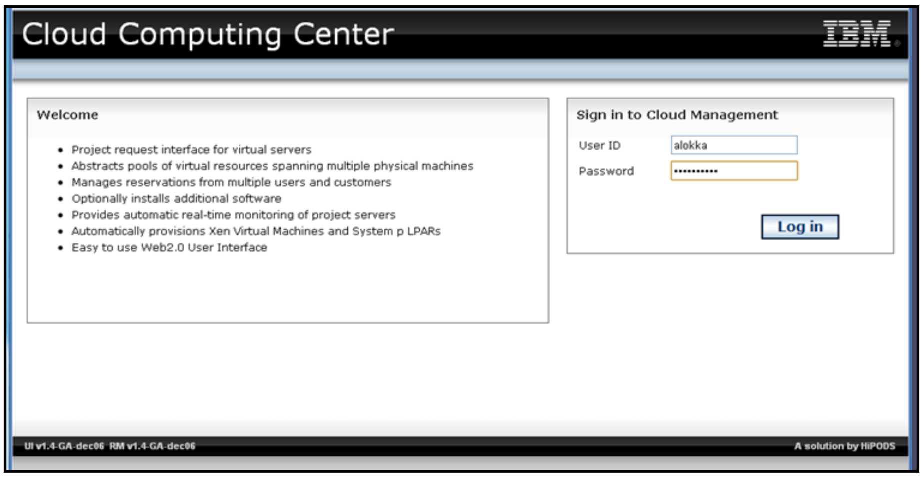
APPENDIX A

Installation and Deployment Guide

This manual gives you a good idea about how to install and use the whole system. Many preparing steps should be done before running the code.

Prerequisites

1. You have to have an account on the cloud. This account could be a customer account or a an administrator/customer account.
2. Enter to the cloud interface using the user name and password given by the cloud administrator (See figure A.1).



Cloud Computing Center

Welcome

- Project request interface for virtual servers
- Abstracts pools of virtual resources spanning multiple physical machines
- Manages reservations from multiple users and customers
- Optionally installs additional software
- Provides automatic real-time monitoring of project servers
- Automatically provisions Xen Virtual Machines and System p LPARs
- Easy to use Web2.0 User Interface

Sign in to Cloud Management

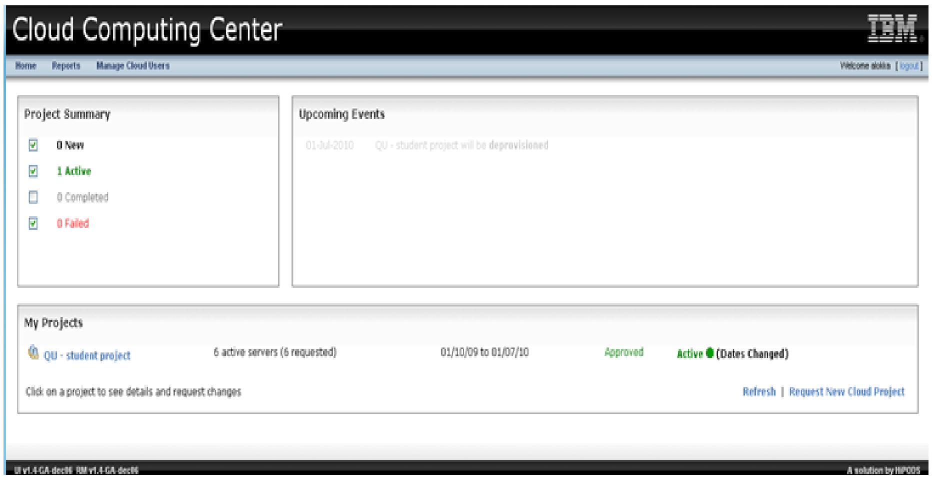
User ID: alokka

Password: ••••••••••

Log in

UI v1.4-GA-dec06 RM v1.4-GA-dec06 — A solution by HiPODS

Figure A.1: Login to the Cloud Interface



Cloud Computing Center

Home | Reports | Manage Cloud Users — Welcome alokka [logout]

Project Summary

- ✓ 0 New
- ✓ 1 Active
- ☐ 0 Completed
- ✓ 0 Failed

Upcoming Events

01-Jul-2010 QU - student project will be deprovisioned

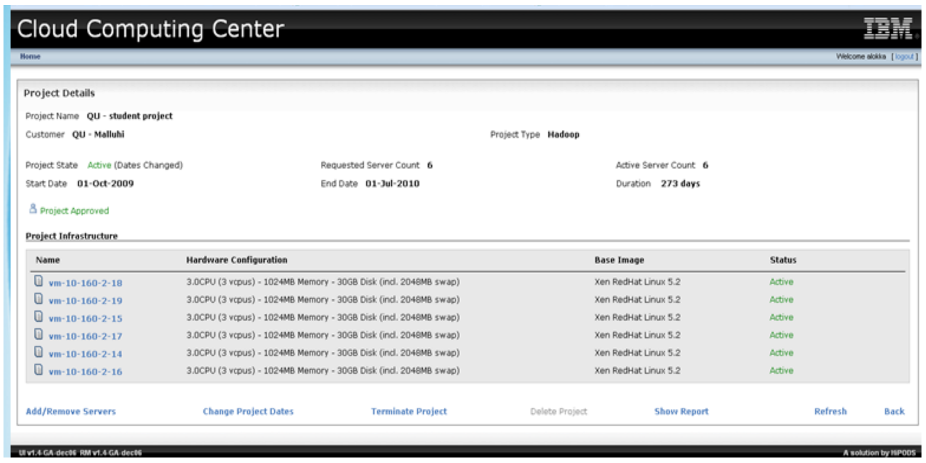
My Projects

QU - student project | 6 active servers (6 requested) | 01/10/09 to 01/07/10 | Approved | Active ● (Dates Changed)

Click on a project to see details and request changes — Refresh | Request New Cloud Project

UI v1.4-GA-dec06 RM v1.4-GA-dec06 — A solution by HiPODS

Figure A.2: Example of an Available Project in the Cloud Interface



Project Details

Project Name **QU - student project**

Customer **QU - Malluhi**

Project Type **Hadoop**

Project State Active (Dates Changed)	Requested Server Count **6**	Active Server Count **6**
Start Date **01-Oct-2009**	End Date **01-Jul-2010**	Duration **273 days**

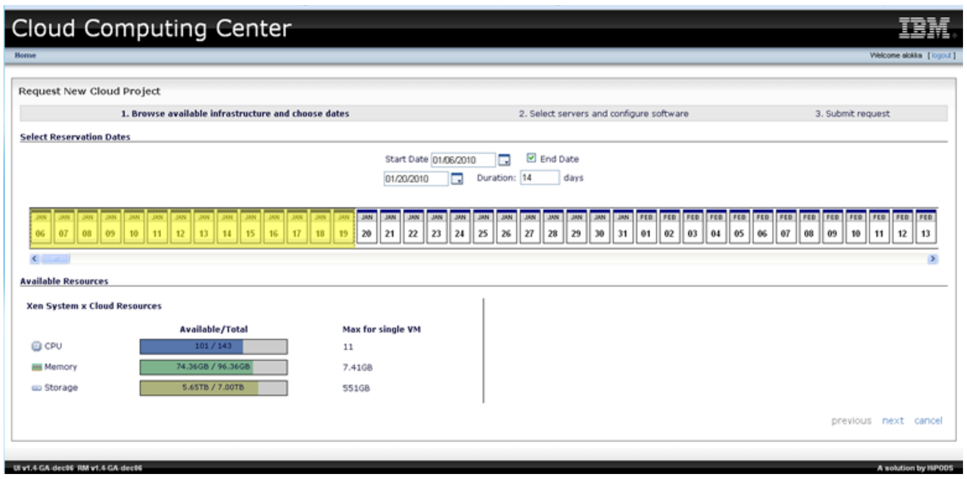
Project Approved

Project Infrastructure

Name	Hardware Configuration	Base Image	Status
vm-10-160-2-18	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-19	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-15	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-17	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-14	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active
vm-10-160-2-16	3.0CPU (3 vcpus) - 1024MB Memory - 30GB Disk (incl. 2048MB swap)	Xen RedHat Linux 5.2	Active

Figure A.3: Available Project Recourses in the Cloud

3. If you have an administrator customer account, you can then request a new project with the needed number of VMs and it should be custom hadoop, see figures below (figure A.4, A.5, A.6)as an example of creating a project on the cloud. Then the cloud administrator should approve the project.



Request New Cloud Project

1. Browse available infrastructure and choose dates 2. Select servers and configure software 3. Submit request

Select Reservation Dates

Start Date 01/06/2010 ☑ End Date 01/20/2010 Duration: 14 days

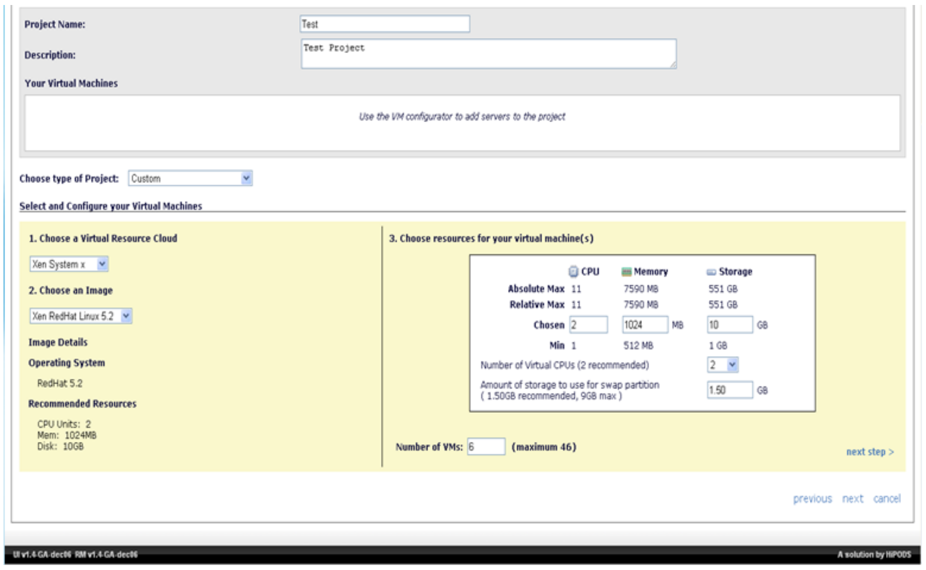
Available Resources

Xen System x Cloud Resources

	Available/Total	Max for single VM
CPU	101 / 143	11
Memory	74.36GB / 96.36GB	7.41GB
Storage	5.65TB / 7.00TB	551GB

Cloud Computing Center
IBM
Home
Welcome alokka [logout]
JAN 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
FEB 01 02 03 04 05 06 07 08 09 10 11 12 13
previous
next
cancel
UI v1.4-GA-dec06 RM v1.4-GA-dec06
A solution by HiPODS

Figure A.4: Create a New Project (step 1)



Project Name: Test

Description: Test Project

Your Virtual Machines

Use the VM configurator to add servers to the project

Choose type of Project: Custom

Select and Configure your Virtual Machines

1. Choose a Virtual Resource Cloud

Xen System x

2. Choose an Image

Xen RedHat Linux 5.2

Image Details

Operating System

RedHat 5.2

Recommended Resources

CPU Units: 2
Mem: 1024MB
Disk: 10GB

3. Choose resources for your virtual machine(s)

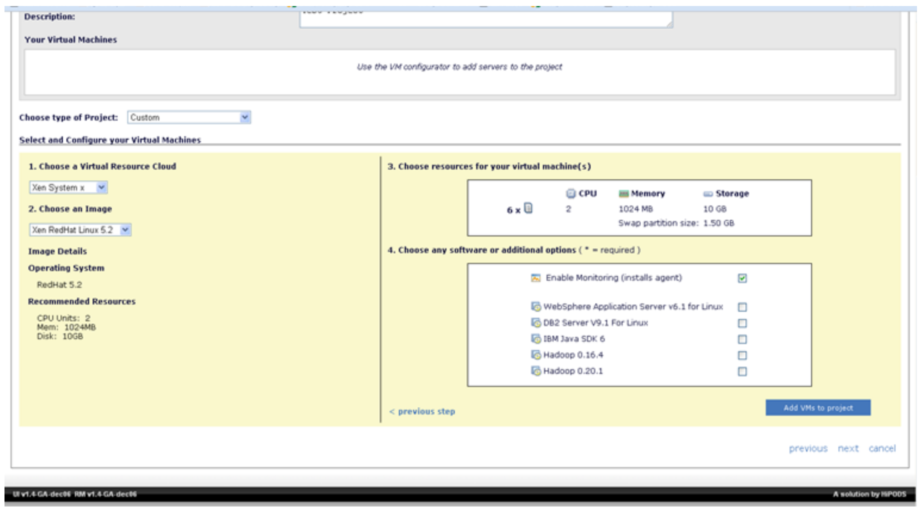
	CPU	Memory	Storage
Absolute Max	11	7590 MB	551 GB
Relative Max	11	7590 MB	551 GB
Chosen	2	1024 MB	10 GB
Min	1	512 MB	1 GB

Number of Virtual CPUs (2 recommended) 2

Amount of storage to use for swap partition (1.50GB recommended, 9GB max) 1.50 GB

Number of VMs: 6 **(maximum 46)**

Figure A.5: Create a New Project (step 2)



Description:

Your Virtual Machines

Use the VM configurator to add servers to the project

Choose type of Project: Custom

Select and Configure your Virtual Machines

1. Choose a Virtual Resource Cloud

Xen System x

2. Choose an Image

Xen RedHat Linux 5.2

Image Details

Operating System

RedHat 5.2

Recommended Resources

CPU Units: 2
Mem: 1024MB
Disk: 10GB

3. Choose resources for your virtual machine(s)

	CPU	Memory	Storage
6 x	2	1024 MB	10 GB
		Swap partition size:	1.50 GB

4. Choose any software or additional options (* = required)

Enable Monitoring (installs agent)	☑
WebSphere Application Server v6.1 for Linux	☐
DB2 Server V9.1 For Linux	☐
IBM Java SDK 6	☐
Hadoop 0.16.4	☐
Hadoop 0.20.1	☐

< previous step

Add VMs to project

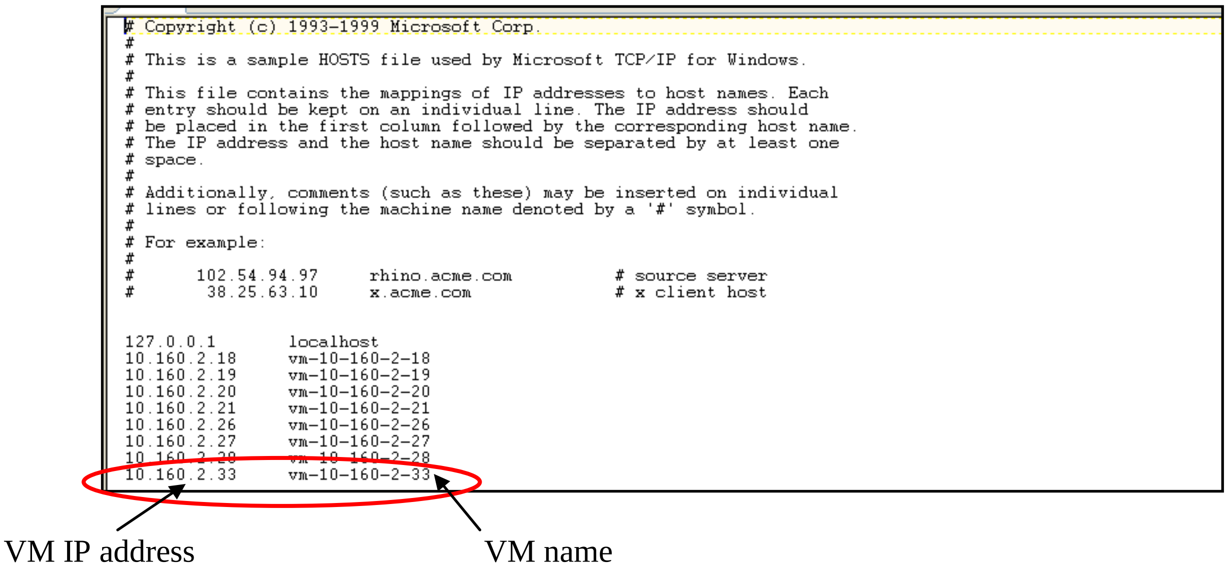
previous next cancel

UI v1.4-GA-dec06 RM v1.4-GA-dec06

A solution by HiPODS

Figure A.6: Create a New Project (step 3)

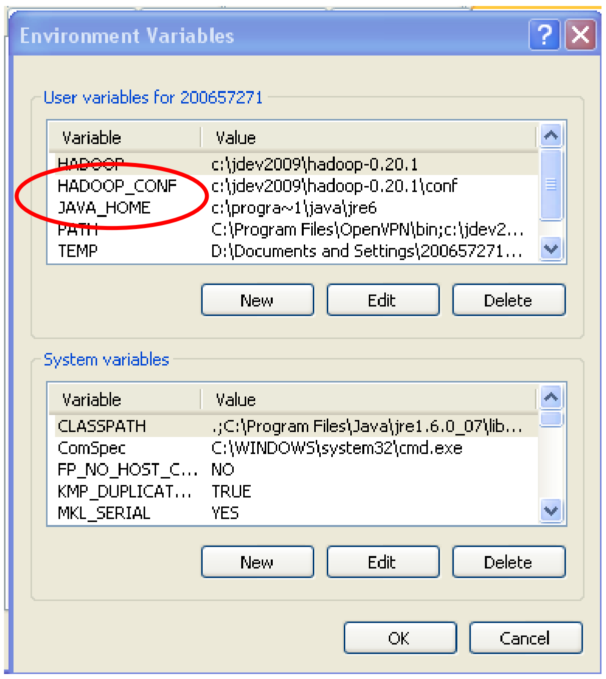
4. I f you do not have a customer account, you need to contact your cloud administrator to create a new project.
5. At this point , the user should have a list of VMs each with an IP address that can be subtracted from figure A.3. The user needs to initiate those VMs in the host file found in C:\WINDOWS\system32\drivers\etc directory. (Figure A.7)



```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host


127.0.0.1       localhost
10.160.2.18     vm-10-160-2-18
10.160.2.19     vm-10-160-2-19
10.160.2.20     vm-10-160-2-20
10.160.2.21     vm-10-160-2-21
10.160.2.26     vm-10-160-2-26
10.160.2.27     vm-10-160-2-27
10.160.2.28     vm-10-160-2-28
10.160.2.33     vm-10-160-2-33
```

Figure A.7: Host file Sample

6. The user now needs to copy the hadoop directory of the cloud in his machine , then he needs to paste them in a known directory. After that, he needs to copy this file in the eclipse plugins.
7. The user needs then to set the environment variable as follows (Figure A.8):



User variables for 200657271

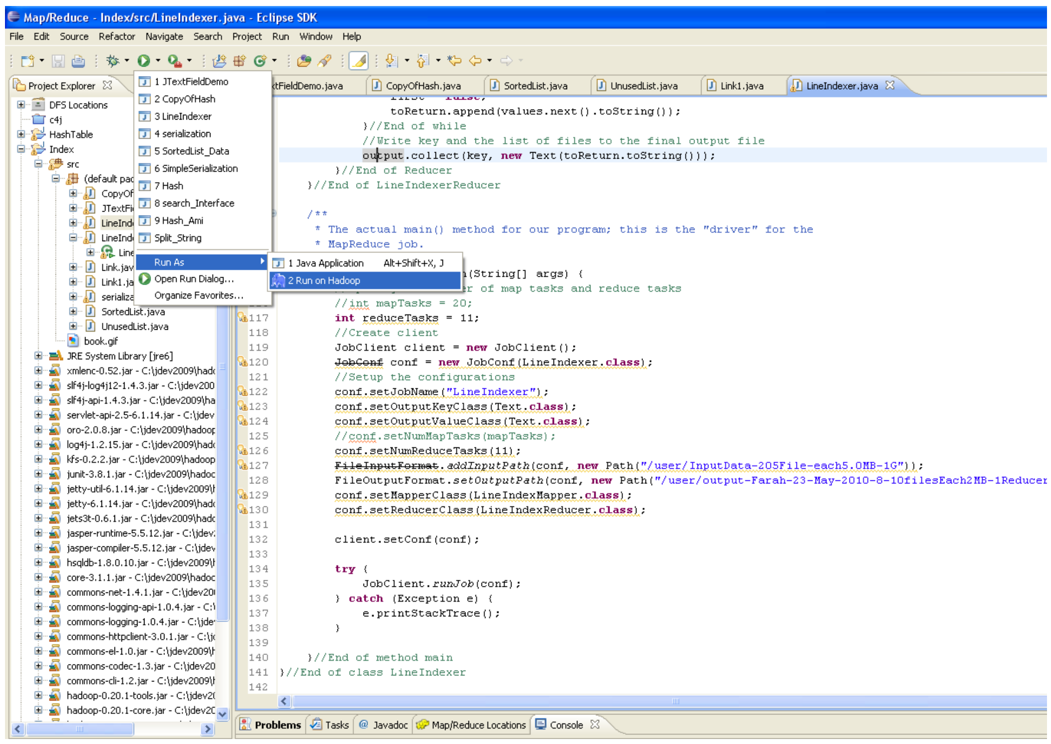
Variable	Value
HADOOP	c:\jdev2009\hadoop-0.20.1
HADOOP_CONF	c:\jdev2009\hadoop-0.20.1\conf
JAVA_HOME	c:\progra~1\java\jre6
PATH	C:\Program Files\OpenVPN\bin;c:\jdev2...
TEMP	D:\Documents and Settings\200657271...

System variables

Variable	Value
CLASSPATH	.;C:\Program Files\Java\jre1.6.0_07\lib...
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
KMP_DUPLICAT...	TRUE
MKL_SERIAL	YES

Figure A.8: Environment Variables

8. Next, the user needs to set up hadoop locations in Eclipse. For this step you can follow the website http://v-lad.org/Tutorials/Hadoop/17%20-%20set%20up%20hadoop%20location%20in%20the%20eclipse.html from 1 till 7. Note: in the steps, instead of putting the localhost, put the IP address of the namenode (last VM in the cloud interface Figure A.3 has namenode 10.160.2.16 for example). Another point to be changed is replace the Map/Reduce Master port number for 9101 to 9001 and DFS Master from 9100 to 9000.
9. The needed following steps are well demonstrated in the following link: http://v-lad.org/Tutorials/Hadoop/23%20-%20create%20the%20project.html : step 13 "create and run a test project" starting from step 1 till step 3.
10. After creating the MapReduce project, the user needs to copy the src directory provided in the CD to his project src directory.
11. The user then needs to change the input and output files in the HDFS as he wants and then run the LindeIndexer.java class provided in the CD. The code should run as a MapReduce project on the configured Hadoop location (As figure A.9 shows).



```
                toReturn.append(values.next().toString());
            }//End of while
            //Write key and the list of files to the final output file
            output.collect(key, new Text(toReturn.toString()));
        }//End of Reducer
    }//End of LineIndexerReducer

    /**
     * The actual main() method for our program; this is the "driver" for the
     * MapReduce job.
                   (String[] args) {
                   r of map tasks and reduce tasks
        //int mapTasks = 20;
        int reduceTasks = 11;
        //Create client
        JobClient client = new JobClient();
        ~~JobConf~~ conf = new JobConf(LineIndexer.class);
        //Setup the configurations
        conf.setJobName("LineIndexer");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);
        //conf.setNumMapTasks(mapTasks);
        conf.setNumReduceTasks(11);
        ~~FileInputFormat~~.addInputPath(conf, new Path("/user/InputData-205File-each5.0MB-1G"));
        FileOutputFormat.setOutputPath(conf, new Path("/user/output-Farah-23-May-2010-8-10filesEach2MB-1Reducer

        client.setConf(conf);

        try {
            JobClient.runJob(conf);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }//End of method main
}//End of class LineIndexer
```

Figure A.9: Run the LineIndexer code on hadoop location.

12. If all the steps are followed as mentioned previously, the LineIndexer code should terminate successfully and the output file will be created in the HDFS location as the user specified the output path.
13. The next step is to upload the output file to the user local machine.
14. Then, the user needs to run the CovertTextToHash.java class and change the input path to be the directory of the output file from LineIndexer code.
15. The output from the previous step will produce a .ser file. To continue the steps till the end you can refer to User Manual in Appendix B.

APPENDIX B

User’s Manual

Guidelines for using the search interface

In this ReadMe file, you will be able to know how to use the attached Ouput.ser and Searching.jar files , so you can use the search interface designed for the project: Data Analysis Using MapReduce Programming Model on the Cloud.

****

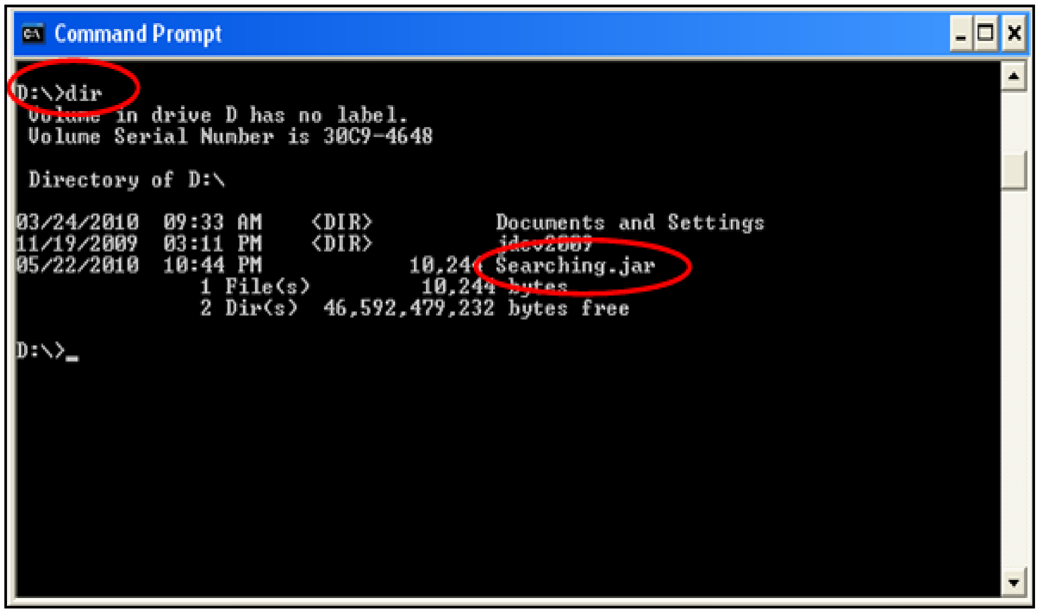
Before using the interface, you have to follow the next instructions:

1.Save the Output.ser file in directory C:/

2.Save Searching.jar in any directory you prefer.

3.Open the command prompt, and change the directory to the one where you saved Searching.jar

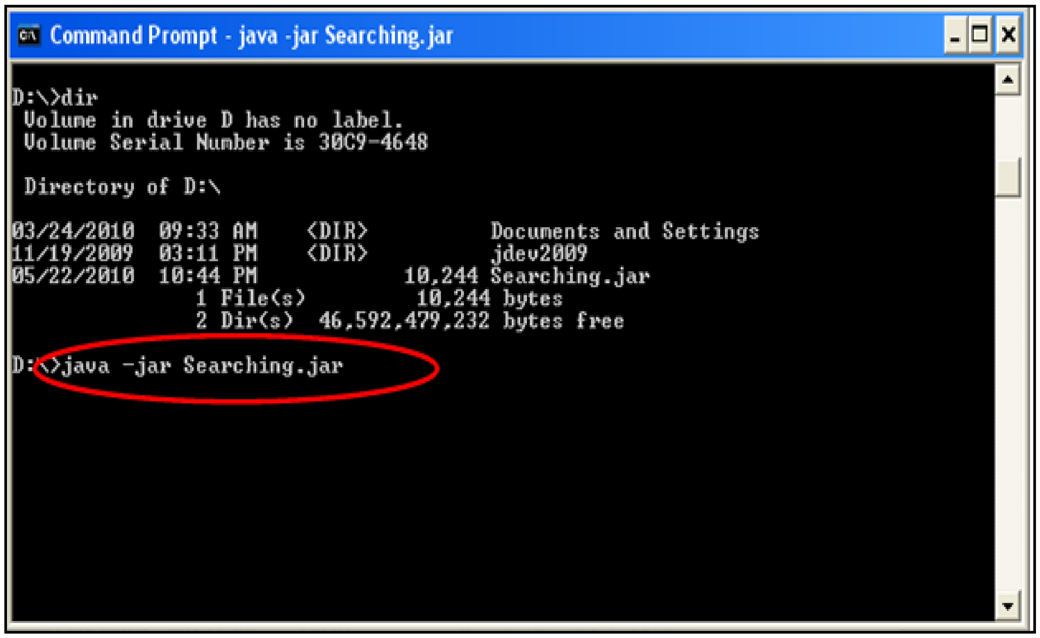
4. To make sure that you are in the correct directory and path, use the command dir, and as shown in figureb.1, you can see that it is the needed path where Searching.jar is found.



```
D:\>dir
 Volume in drive D has no label.
 Volume Serial Number is 30C9-4648

 Directory of D:\

03/24/2010  09:33 AM    <DIR>          Documents and Settings
11/19/2009  03:11 PM    <DIR>          idev2009
05/22/2010  10:44 PM            10,244 Searching.jar
               1 File(s)         10,244 bytes
               2 Dir(s)  46,592,479,232 bytes free

D:\>_
```

Figureb.1:dir command is used to see what is in the current directory

5.To open the runable Search.jar file use the following command: java –jar Searching.jar (shown in figureb.2).



```
D:\>dir
 Volume in drive D has no label.
 Volume Serial Number is 30C9-4648

 Directory of D:\

03/24/2010  09:33 AM    <DIR>          Documents and Settings
11/19/2009  03:11 PM    <DIR>          jdev2009
05/22/2010  10:44 PM            10,244 Searching.jar
               1 File(s)         10,244 bytes
               2 Dir(s)  46,592,479,232 bytes free

D:\>java -jar Searching.jar
```

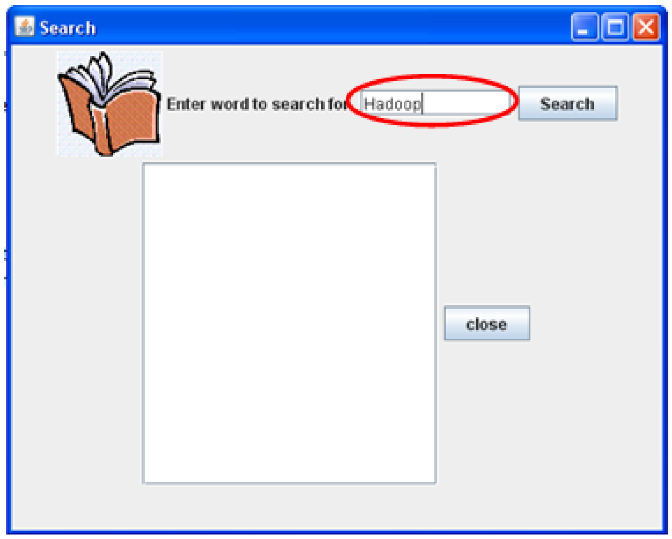
FigureB.2: use java –jar file.jar command

6. As a result for step 5, the interface will be opened to you, and now you can search for any word you want.

****

Way of using the interface itself:

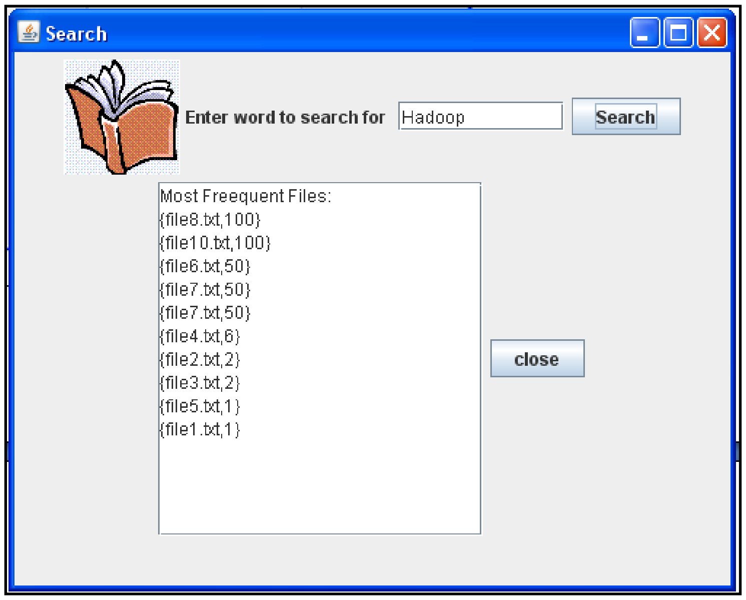
1. You just have to enter the needed word that you are looking for in the field shown in red circle in figure b.3.



FigureB.3:field to enter the needed word to search for

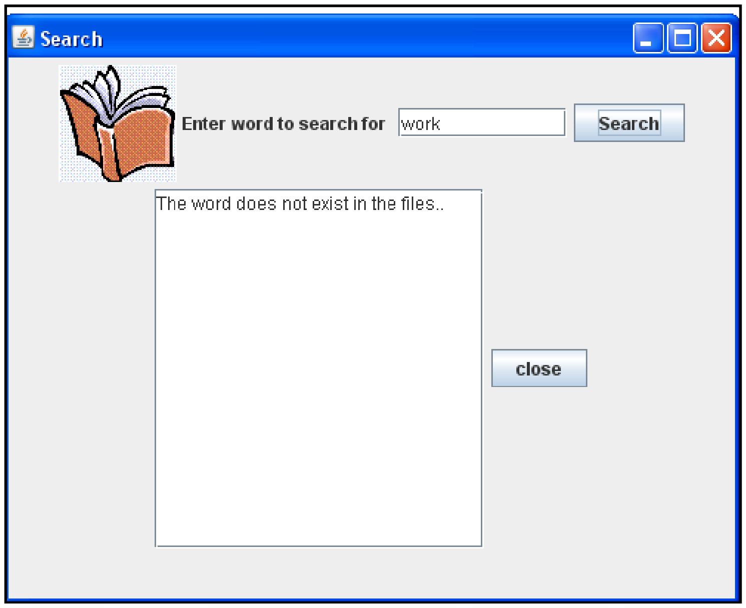
2.Press on the button Search.

If the word exists, you will get the result that shows a list of ten names of files or less, where the word has most frequency, and the frequency is attached to the name of file as shown in figureb.4. And if the word doesn't exist, "the word does not exist in the files" sentence is printed (Figure B.5).



```
Most Freequent Files:
{file8.txt,100}
{file10.txt,100}
{file6.txt,50}
{file7.txt,50}
{file7.txt,50}
{file4.txt,6}
{file2.txt,2}
{file3.txt,2}
{file5.txt,1}
{file1.txt,1}
```

Figure B.4: Example of an output for available word in the files



```
The word does not exist in the files..
```

Figure B.5:Example of an output for unavailable word in the files

****

For any questions, contact us using the following e-mails:

200657271@qu.edu.qa

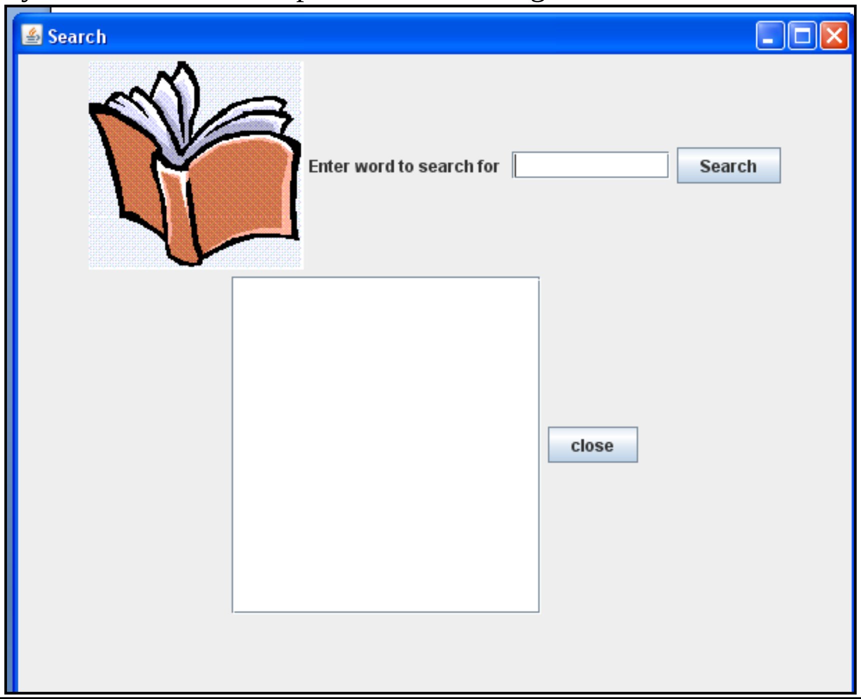
200652758@qu.edu.qa

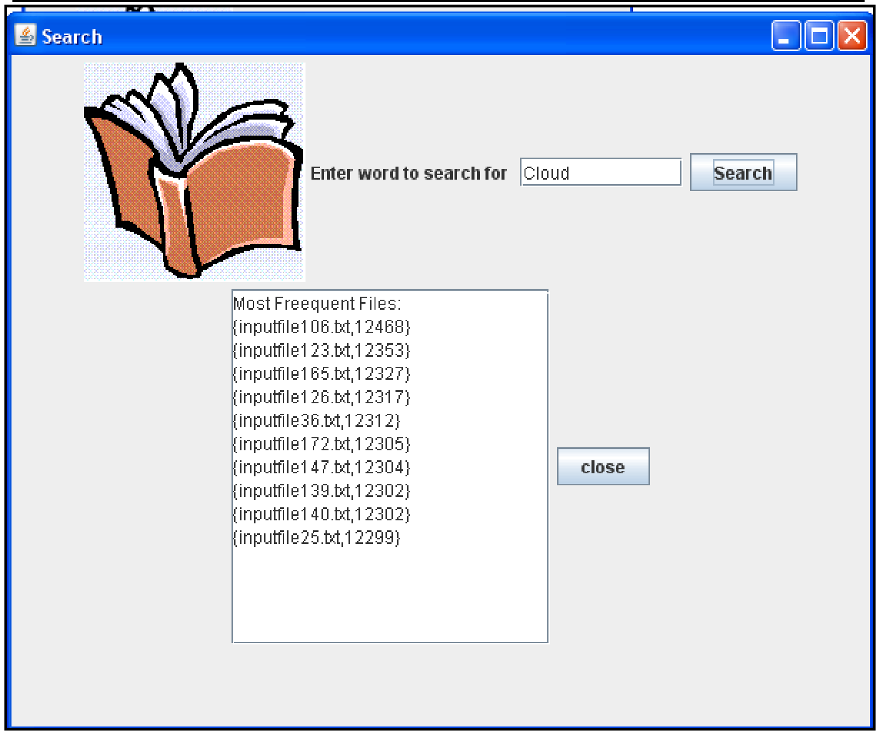
200653782@qu.edu.qa

APPENDIX C

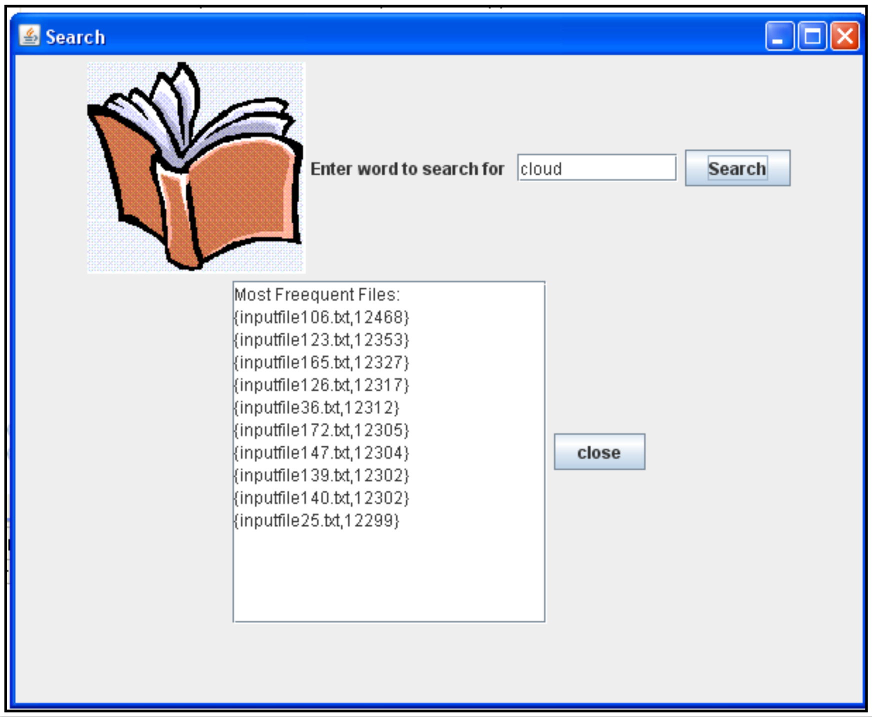
Graphical User Interface

In this appendix you can see some snapshots for the designed interface:

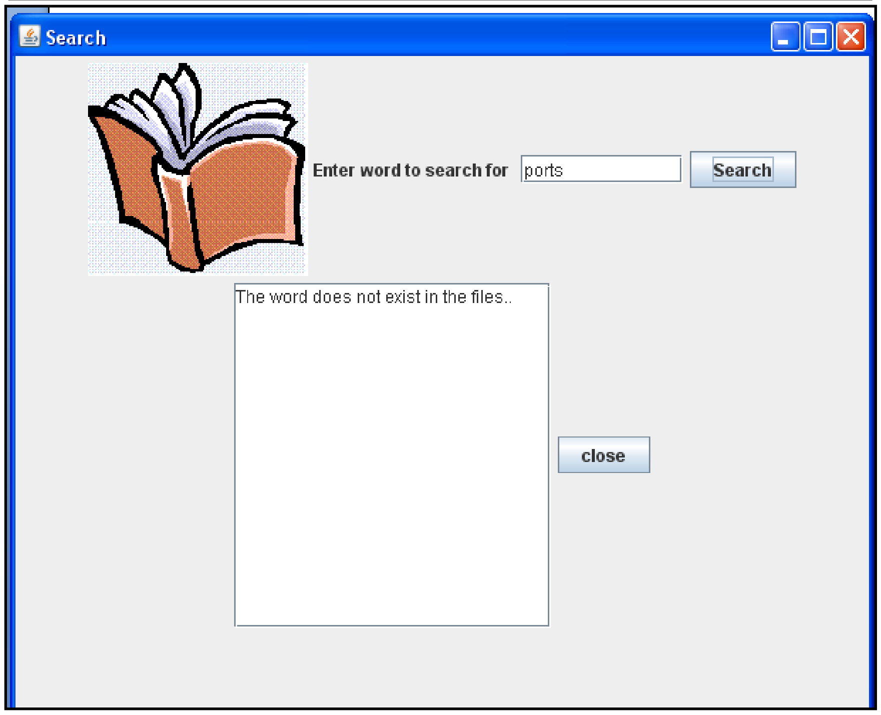




```
Most Freequent Files:
{inputfile106.txt,12468}
{inputfile123.txt,12353}
{inputfile165.txt,12327}
{inputfile126.txt,12317}
{inputfile36.txt,12312}
{inputfile172.txt,12305}
{inputfile147.txt,12304}
{inputfile139.txt,12302}
{inputfile140.txt,12302}
{inputfile25.txt,12299}
```



```
Most Freequent Files:
{inputfile106.txt,12468}
{inputfile123.txt,12353}
{inputfile165.txt,12327}
{inputfile126.txt,12317}
{inputfile36.txt,12312}
{inputfile172.txt,12305}
{inputfile147.txt,12304}
{inputfile139.txt,12302}
{inputfile140.txt,12302}
{inputfile25.txt,12299}
```

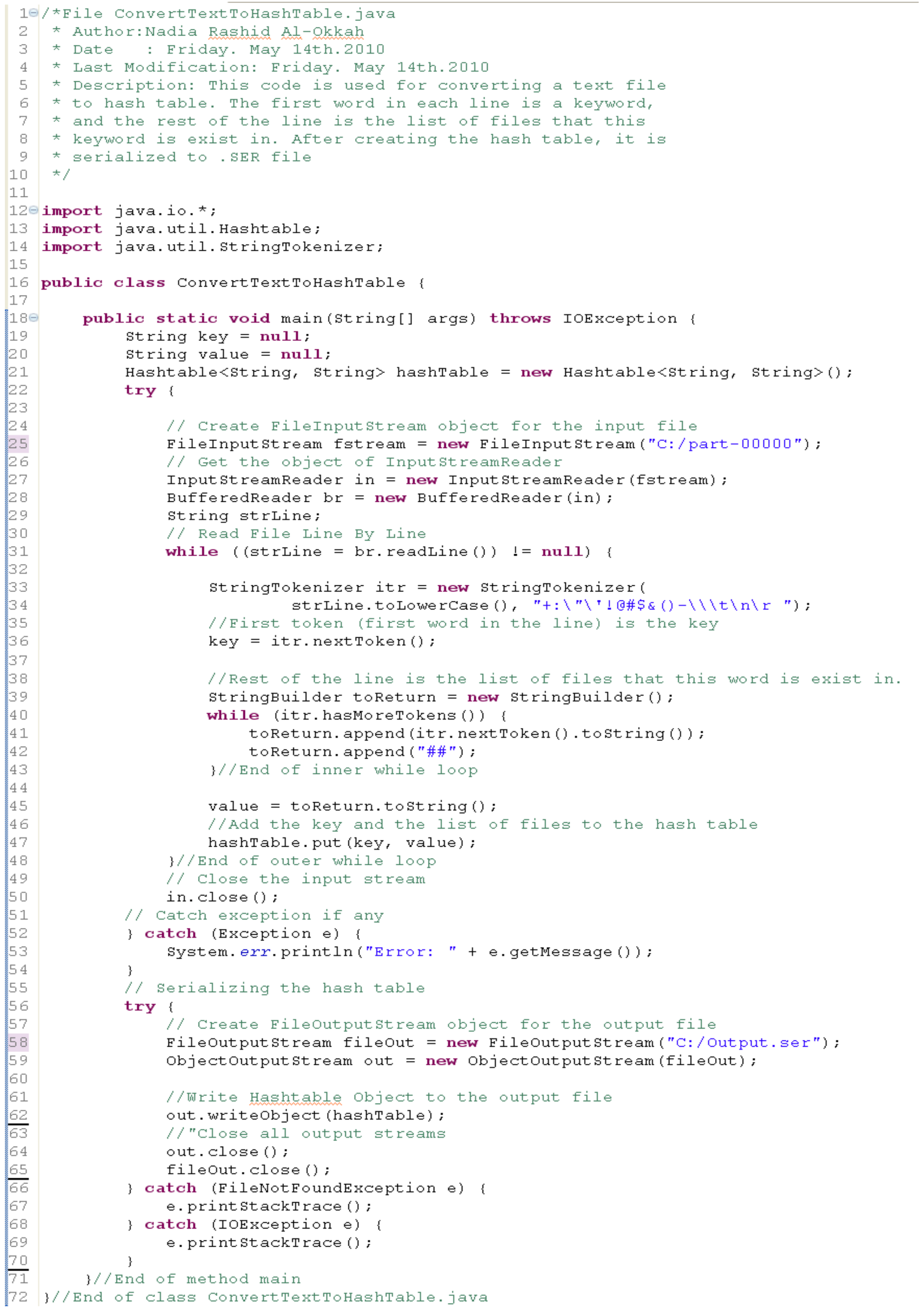


```
The word does not exist in the files..
```

APPENDIX D

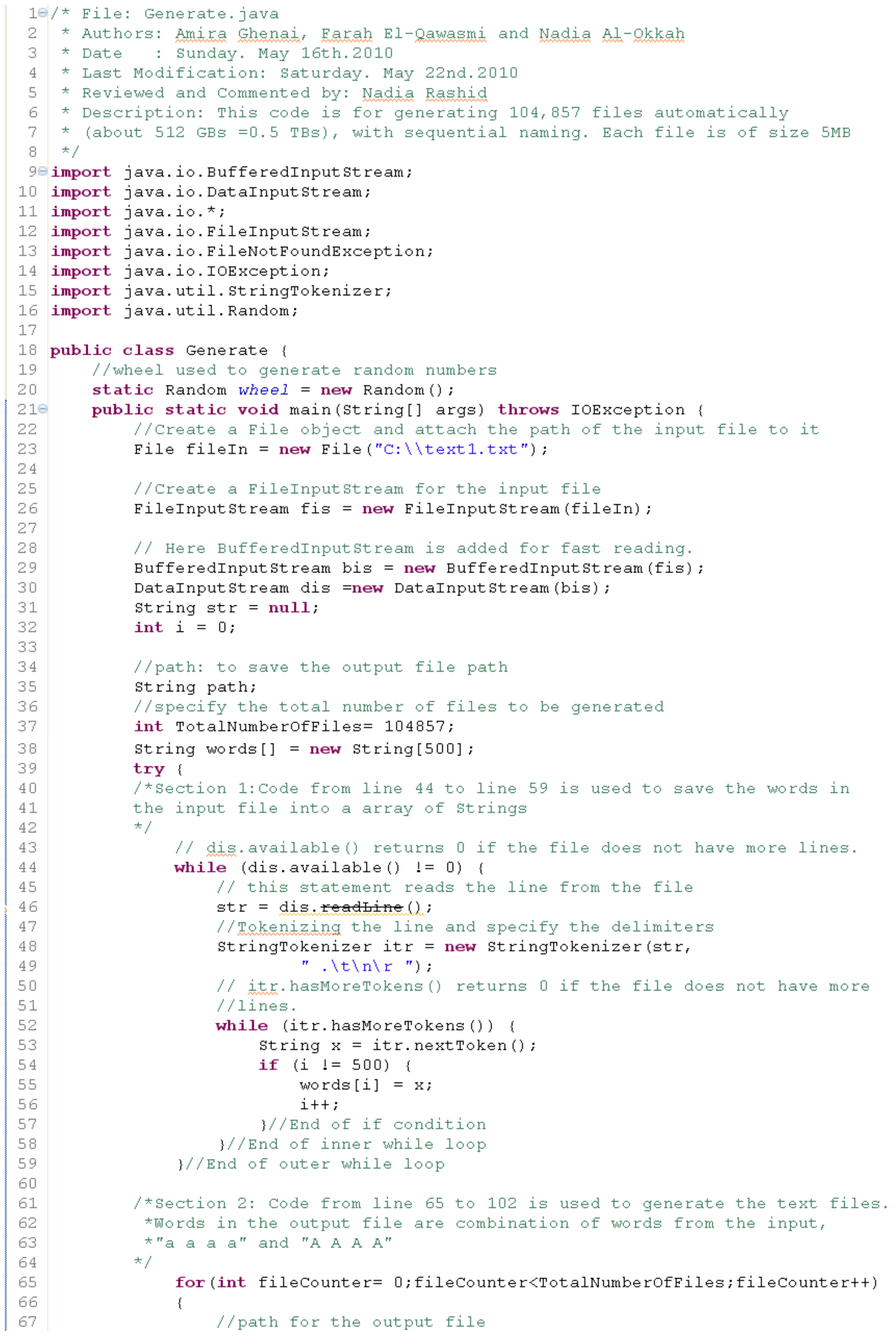
Source Codes

1. *ConvertTextToHashTable.java*



```java
/*File ConvertTextToHashTable.java
 * Author:Nadia Rashid Al-Okkah
 * Date   : Friday. May 14th.2010
 * Last Modification: Friday. May 14th.2010
 * Description: This code is used for converting a text file
 * to hash table. The first word in each line is a keyword,
 * and the rest of the line is the list of files that this
 * keyword is exist in. After creating the hash table, it is
 * serialized to .SER file
 */

import java.io.*;
import java.util.Hashtable;
import java.util.StringTokenizer;

public class ConvertTextToHashTable {

    public static void main(String[] args) throws IOException {
        String key = null;
        String value = null;
        Hashtable<String, String> hashTable = new Hashtable<String, String>();
        try {

            // Create FileInputStream object for the input file
            FileInputStream fstream = new FileInputStream("C:/part-00000");
            // Get the object of InputStreamReader
            InputStreamReader in = new InputStreamReader(fstream);
            BufferedReader br = new BufferedReader(in);
            String strLine;
            // Read File Line By Line
            while ((strLine = br.readLine()) != null) {

                StringTokenizer itr = new StringTokenizer(
                        strLine.toLowerCase(), "+:\"\'!@#$&()-\\\t\n\r ");
                //First token (first word in the line) is the key
                key = itr.nextToken();

                //Rest of the line is the list of files that this word is exist in.
                StringBuilder toReturn = new StringBuilder();
                while (itr.hasMoreTokens()) {
                    toReturn.append(itr.nextToken().toString());
                    toReturn.append("##");
                }//End of inner while loop

                value = toReturn.toString();
                //Add the key and the list of files to the hash table
                hashTable.put(key, value);
            }//End of outer while loop
            // Close the input stream
            in.close();
        // Catch exception if any
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
        // Serializing the hash table
        try {
            // Create FileOutputStream object for the output file
            FileOutputStream fileOut = new FileOutputStream("C:/Output.ser");
            ObjectOutputStream out = new ObjectOutputStream(fileOut);

            //Write Hashtable Object to the output file
            out.writeObject(hashTable);
            //"Close all output streams
            out.close();
            fileOut.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }//End of method main
}//End of class ConvertTextToHashTable.java
```

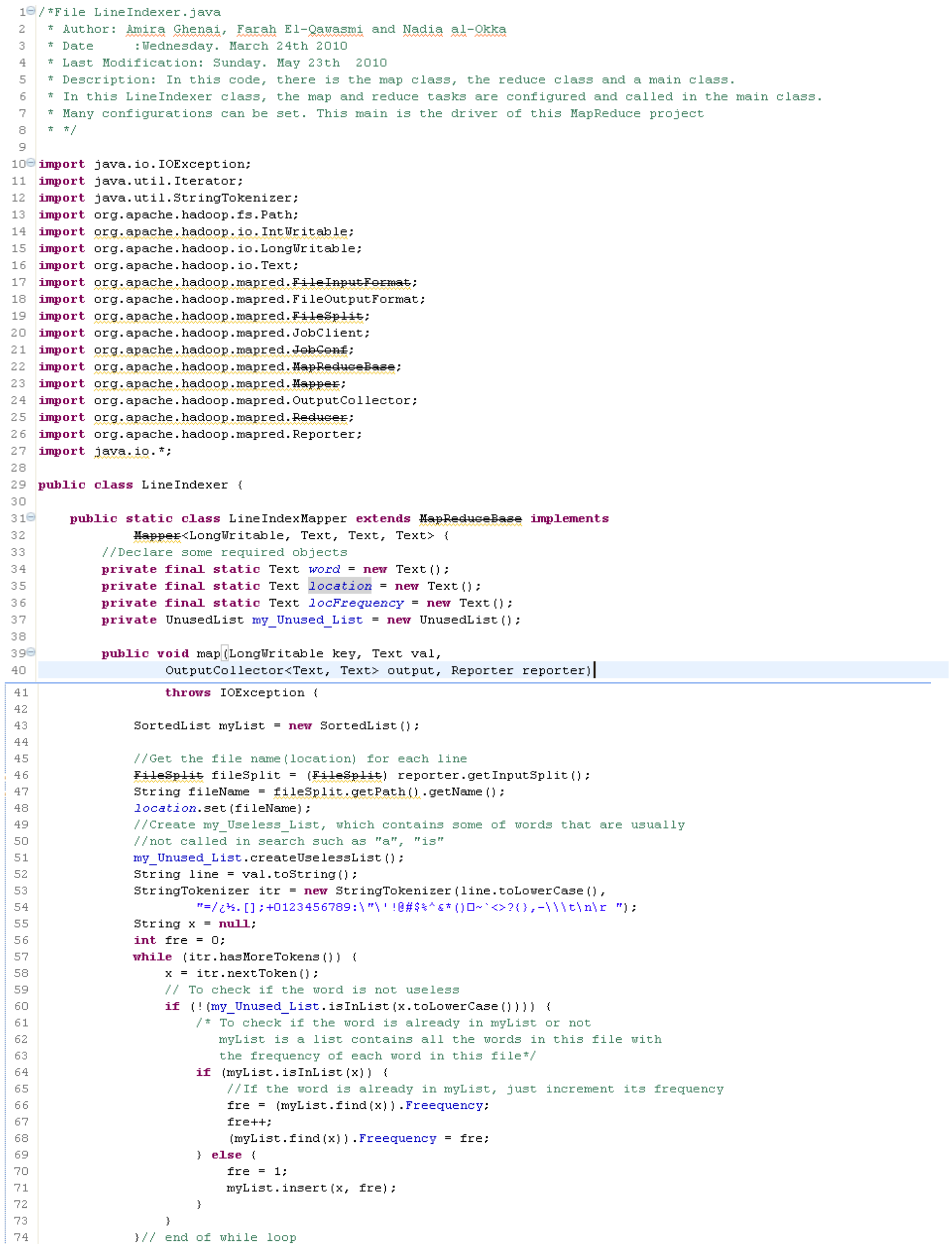
2. *Generate.java*



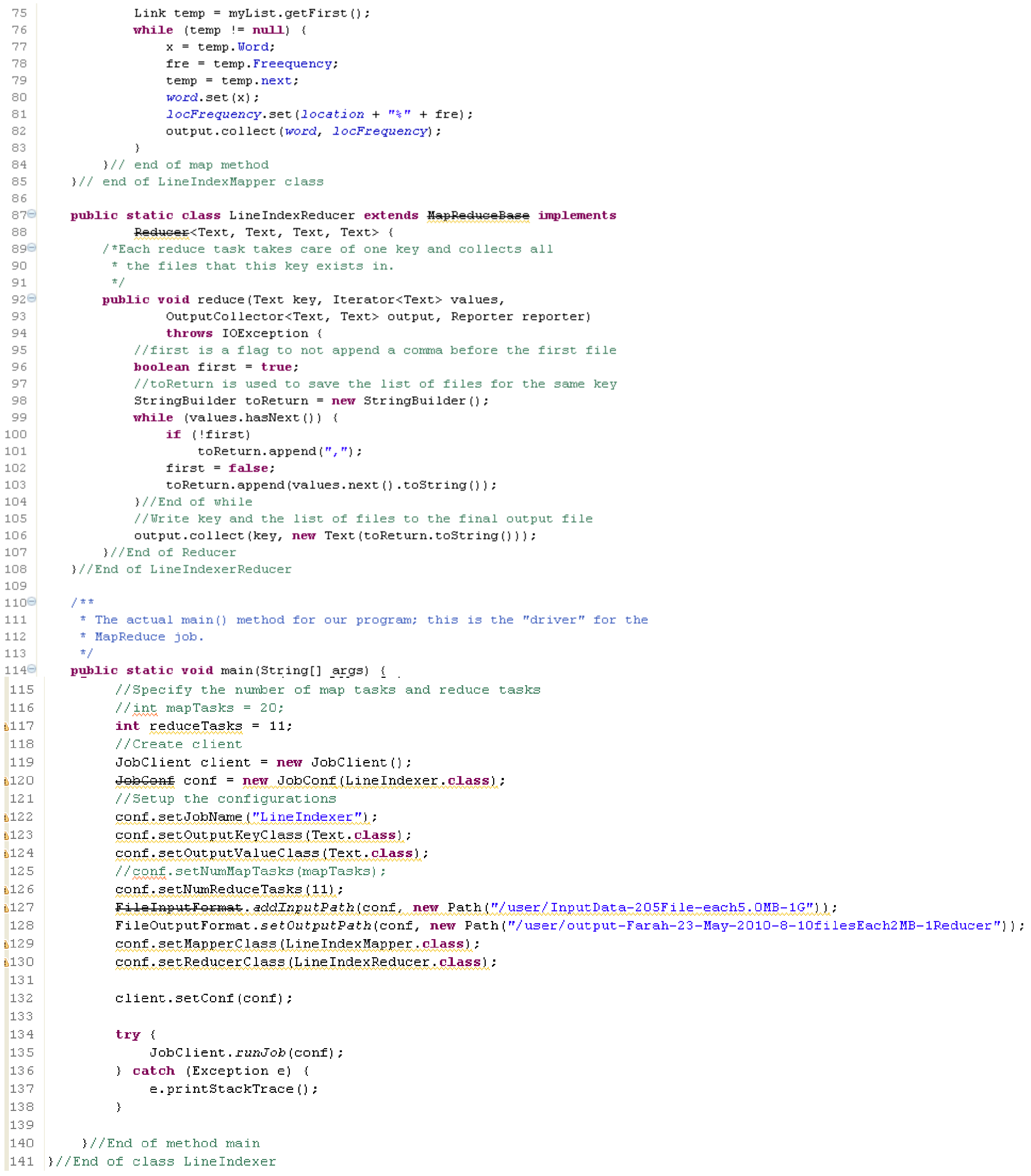
```
 1 /* File: Generate.java
 2  * Authors: Amira Ghenai, Farah El-Qawasmi and Nadia Al-Okkah
 3  * Date   : Sunday. May 16th.2010
 4  * Last Modification: Saturday. May 22nd.2010
 5  * Reviewed and Commented by: Nadia Rashid
 6  * Description: This code is for generating 104,857 files automatically
 7  * (about 512 GBs =0.5 TBs), with sequential naming. Each file is of size 5MB
 8  */
 9 import java.io.BufferedInputStream;
10 import java.io.DataInputStream;
11 import java.io.*;
12 import java.io.FileInputStream;
13 import java.io.FileNotFoundException;
14 import java.io.IOException;
15 import java.util.StringTokenizer;
16 import java.util.Random;
17 
18 public class Generate {
19     //wheel used to generate random numbers
20     static Random wheel = new Random();
21     public static void main(String[] args) throws IOException {
22         //Create a File object and attach the path of the input file to it
23         File fileIn = new File("C:\\text1.txt");
24 
25         //Create a FileInputStream for the input file
26         FileInputStream fis = new FileInputStream(fileIn);
27 
28         // Here BufferedInputStream is added for fast reading.
29         BufferedInputStream bis = new BufferedInputStream(fis);
30         DataInputStream dis =new DataInputStream(bis);
31         String str = null;
32         int i = 0;
33 
34         //path: to save the output file path
35         String path;
36         //specify the total number of files to be generated
37         int TotalNumberOfFiles= 104857;
38         String words[] = new String[500];
39         try {
40         /*Section 1:Code from line 44 to line 59 is used to save the words in
41         the input file into a array of Strings
42         */
43             // dis.available() returns 0 if the file does not have more lines.
44             while (dis.available() != 0) {
45                 // this statement reads the line from the file
46                 str = dis.~~readLine~~();
47                 //Tokenizing the line and specify the delimiters
48                 StringTokenizer itr = new StringTokenizer(str,
49                         " .\t\n\r ");
50                 // itr.hasMoreTokens() returns 0 if the file does not have more
51                 //lines.
52                 while (itr.hasMoreTokens()) {
53                     String x = itr.nextToken();
54                     if (i != 500) {
55                         words[i] = x;
56                         i++;
57                     }//End of if condition
58                 }//End of inner while loop
59             }//End of outer while loop
60 
61         /*Section 2: Code from line 65 to 102 is used to generate the text files.
62          *Words in the output file are combination of words from the input,
63          *"a a a a" and "A A A A"
64         */
65             for(int fileCounter= 0;fileCounter<TotalNumberOfFiles;fileCounter++)
66             {
67                 //path for the output file
```

```
                path="H:\\InputData-104652Files-each5.0MB\\inputFile"
                    +(fileCounter+1)+".txt";
                File fileOut = new File(path);
                Writer output = new BufferedWriter(new FileWriter(fileOut));
                //Generate two random numbers for the word to e picked and the
                //number of words per line
                int pickWords = wheel.nextInt(500) ;
                int m;
                //This loop is used for inserting " a a a a a " in the output file
                for (m = 0; m < 59000; m++) {
                        output.write(" a a a a a  ");
                }
                //This loop for inserting random words from the input
                //file to the output file
                for (m = 0; m < 550000; m++) {

                        output.write(words[pickWords]);
                        output.write(" ");
                        pickWords = wheel.nextInt(500) ;
                }
                //This loop for inserting " A A A A A A " in the output file
                for (m = 0; m < 58000; m++) {
                        output.write(" A A A A A A ");
                }
                // dispose all the resources after using them.
                fis.close();
                bis.close();
                dis.close();
                output.close();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }//End of main
}//End of class Generate
```

3. LineIndexer.java

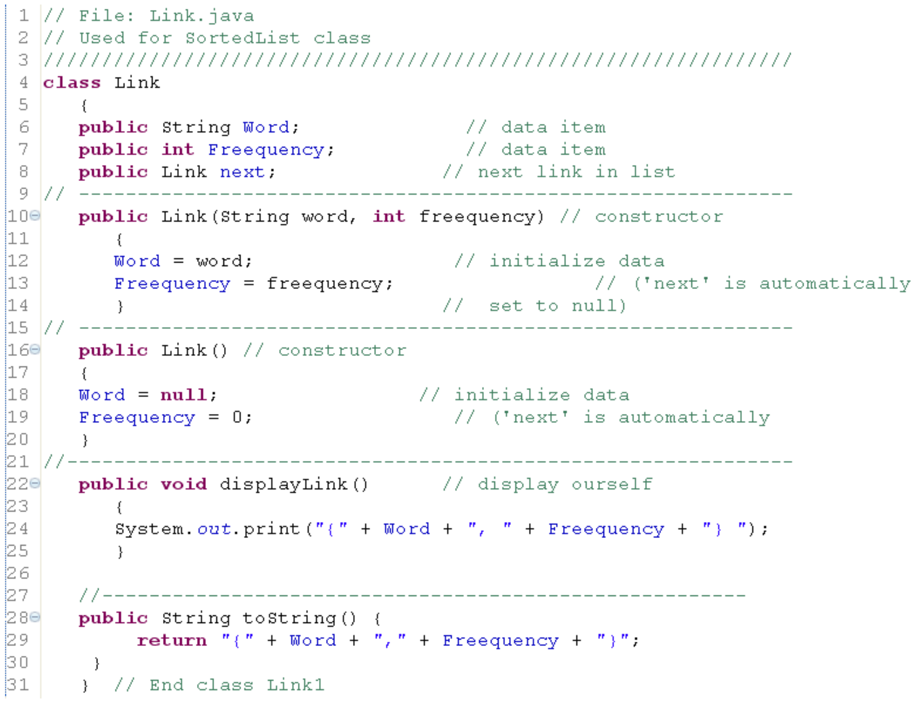


```java
/*File LineIndexer.java
 * Author: Amira Ghenai, Farah El-Qawasmi and Nadia al-Okka
 * Date     :Wednesday. March 24th 2010
 * Last Modification: Sunday. May 23th  2010
 * Description: In this code, there is the map class, the reduce class and a main class.
 * In this LineIndexer class, the map and reduce tasks are configured and called in the main class.
 * Many configurations can be set. This main is the driver of this MapReduce project
 * */

import java.io.IOException;
import java.util.Iterator;
import java.util.StringTokenizer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.~~FileInputFormat~~;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.~~FileSplit~~;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.~~JobConf~~;
import org.apache.hadoop.mapred.~~MapReduceBase~~;
import org.apache.hadoop.mapred.~~Mapper~~;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.~~Reducer~~;
import org.apache.hadoop.mapred.Reporter;
import java.io.*;

public class LineIndexer {

    public static class LineIndexMapper extends ~~MapReduceBase~~ implements
            ~~Mapper~~<LongWritable, Text, Text, Text> {
        //Declare some required objects
        private final static Text word = new Text();
        private final static Text location = new Text();
        private final static Text locFrequency = new Text();
        private UnusedList my_Unused_List = new UnusedList();

        public void map(LongWritable key, Text val,
                OutputCollector<Text, Text> output, Reporter reporter)
                throws IOException {

            SortedList myList = new SortedList();

            //Get the file name(location) for each line
            ~~FileSplit~~ fileSplit = (~~FileSplit~~) reporter.getInputSplit();
            String fileName = fileSplit.getPath().getName();
            location.set(fileName);
            //Create my_Useless_List, which contains some of words that are usually
            //not called in search such as "a", "is"
            my_Unused_List.createUselessList();
            String line = val.toString();
            StringTokenizer itr = new StringTokenizer(line.toLowerCase(),
                    "=/¿½.[];+0123456789:\"\'!@#$%^&*()□~`<>?(),-\\\t\n\r ");
            String x = null;
            int fre = 0;
            while (itr.hasMoreTokens()) {
                x = itr.nextToken();
                // To check if the word is not useless
                if (!(my_Unused_List.isInList(x.toLowerCase()))) {
                    /* To check if the word is already in myList or not
                       myList is a list contains all the words in this file with
                       the frequency of each word in this file*/
                    if (myList.isInList(x)) {
                        //If the word is already in myList, just increment its frequency
                        fre = (myList.find(x)).Freequency;
                        fre++;
                        (myList.find(x)).Freequency = fre;
                    } else {
                        fre = 1;
                        myList.insert(x, fre);
                    }
                }
            }// end of while loop
```



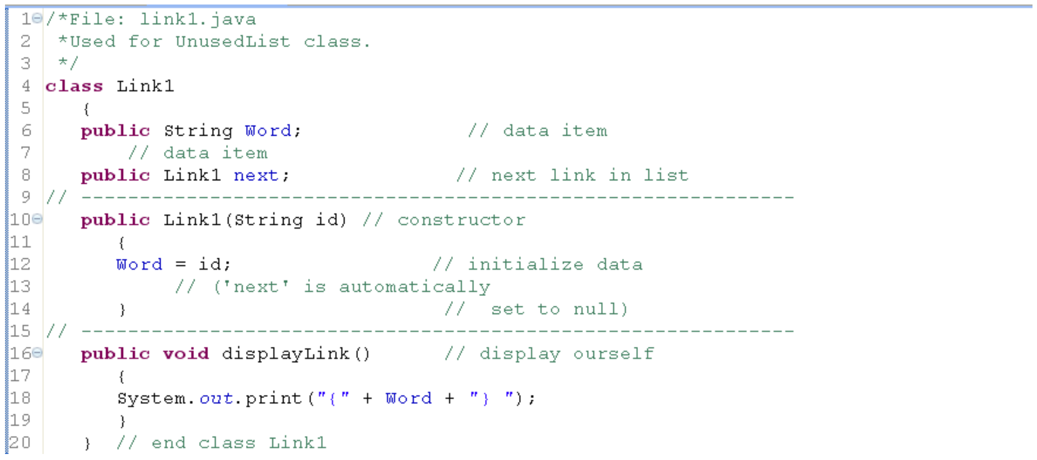
```java
            Link temp = myList.getFirst();
            while (temp != null) {
                x = temp.Word;
                fre = temp.Freequency;
                temp = temp.next;
                word.set(x);
                locFrequency.set(location + "%" + fre);
                output.collect(word, locFrequency);
            }
        }// end of map method
    }// end of LineIndexMapper class

    public static class LineIndexReducer extends ~~MapReduceBase~~ implements
            ~~Reducer~~<Text, Text, Text, Text> {
        /*Each reduce task takes care of one key and collects all
         * the files that this key exists in.
         */
        public void reduce(Text key, Iterator<Text> values,
                OutputCollector<Text, Text> output, Reporter reporter)
                throws IOException {
            //first is a flag to not append a comma before the first file
            boolean first = true;
            //toReturn is used to save the list of files for the same key
            StringBuilder toReturn = new StringBuilder();
            while (values.hasNext()) {
                if (!first)
                    toReturn.append(",");
                first = false;
                toReturn.append(values.next().toString());
            }//End of while
            //Write key and the list of files to the final output file
            output.collect(key, new Text(toReturn.toString()));
        }//End of Reducer
    }//End of LineIndexerReducer

    /**
     * The actual main() method for our program; this is the "driver" for the
     * MapReduce job.
     */
    public static void main(String[] args) {
        //Specify the number of map tasks and reduce tasks
        //int mapTasks = 20;
        int reduceTasks = 11;
        //Create client
        JobClient client = new JobClient();
        ~~JobConf~~ conf = new JobConf(LineIndexer.class);
        //Setup the configurations
        conf.setJobName("LineIndexer");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);
        //conf.setNumMapTasks(mapTasks);
        conf.setNumReduceTasks(11);
        ~~FileInputFormat~~.addInputPath(conf, new Path("/user/InputData-205File-each5.0MB-1G"));
        FileOutputFormat.setOutputPath(conf, new Path("/user/output-Farah-23-May-2010-8-10filesEach2MB-1Reducer"));
        conf.setMapperClass(LineIndexMapper.class);
        conf.setReducerClass(LineIndexReducer.class);

        client.setConf(conf);

        try {
            JobClient.runJob(conf);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }//End of method main
}//End of class LineIndexer
```

4. Link.java



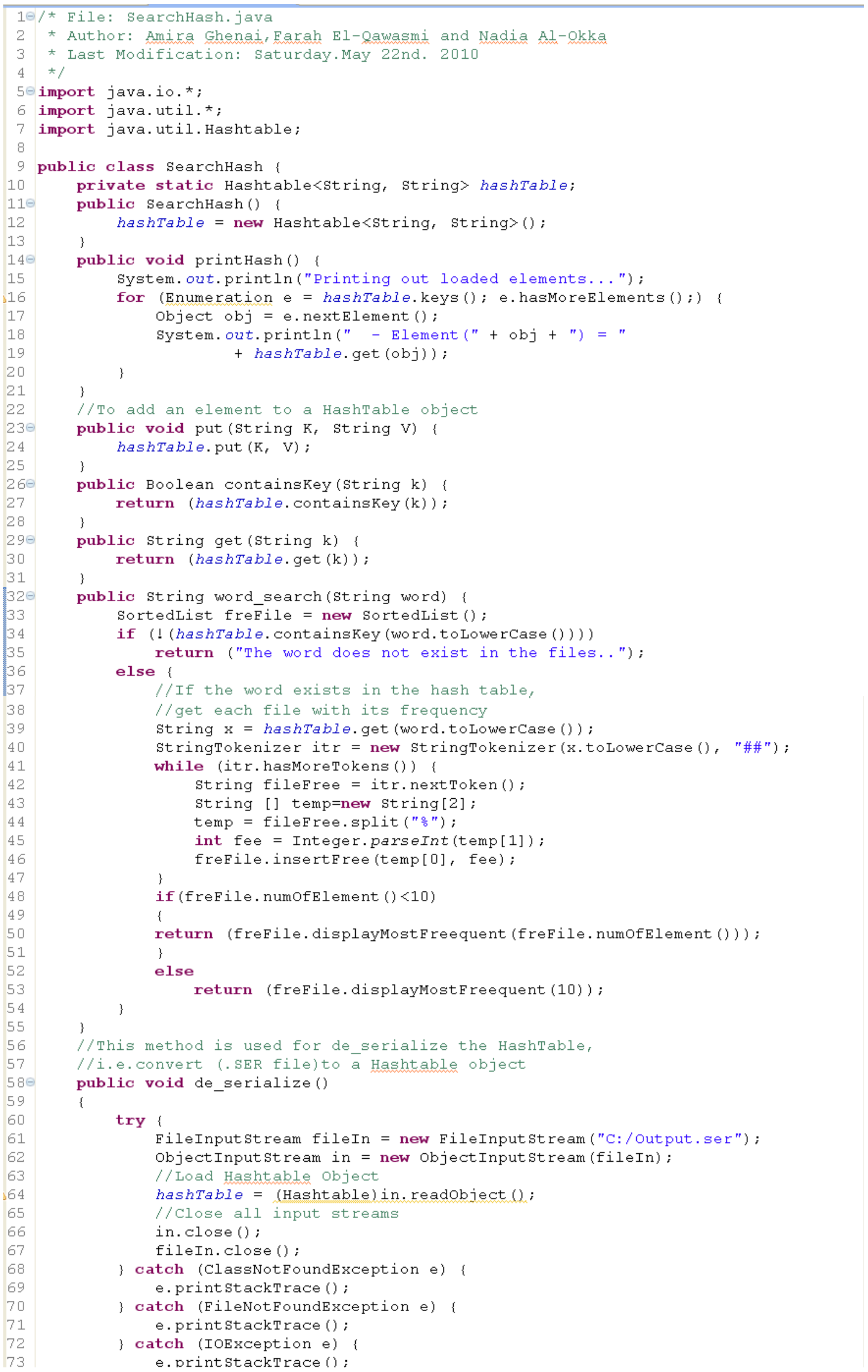
```java
// File: Link.java
// Used for SortedList class
///////////////////////////////////////////////////////////////
class Link
   {
   public String Word;              // data item
   public int Freequency;           // data item
   public Link next;              // next link in list
// ------------------------------------------------------------
   public Link(String word, int freequency) // constructor
      {
      Word = word;                    // initialize data
      Freequency = freequency;                // ('next' is automatically
      }                           //  set to null)
// ------------------------------------------------------------
   public Link() // constructor
   {
   Word = null;                 // initialize data
   Freequency = 0;                 // ('next' is automatically
   }
//-------------------------------------------------------------
   public void displayLink()      // display ourself
      {
      System.out.print("{" + Word + ", " + Freequency + "} ");
      }

   //-------------------------------------------------------
   public String toString() {
        return "{" + Word + "," + Freequency + "}";
    }
   }  // End class Link1
```

5. Link1.java

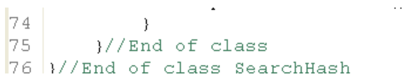


```java
/*File: link1.java
 *Used for UnusedList class.
 */
class Link1
   {
   public String Word;              // data item
       // data item
   public Link1 next;              // next link in list
// ------------------------------------------------------------
   public Link1(String id) // constructor
      {
      Word = id;                 // initialize data
           // ('next' is automatically
      }                           //  set to null)
// ------------------------------------------------------------
   public void displayLink()      // display ourself
      {
      System.out.print("{" + Word + "} ");
      }
   }  // end class Link1
```

6. SearchHash.java

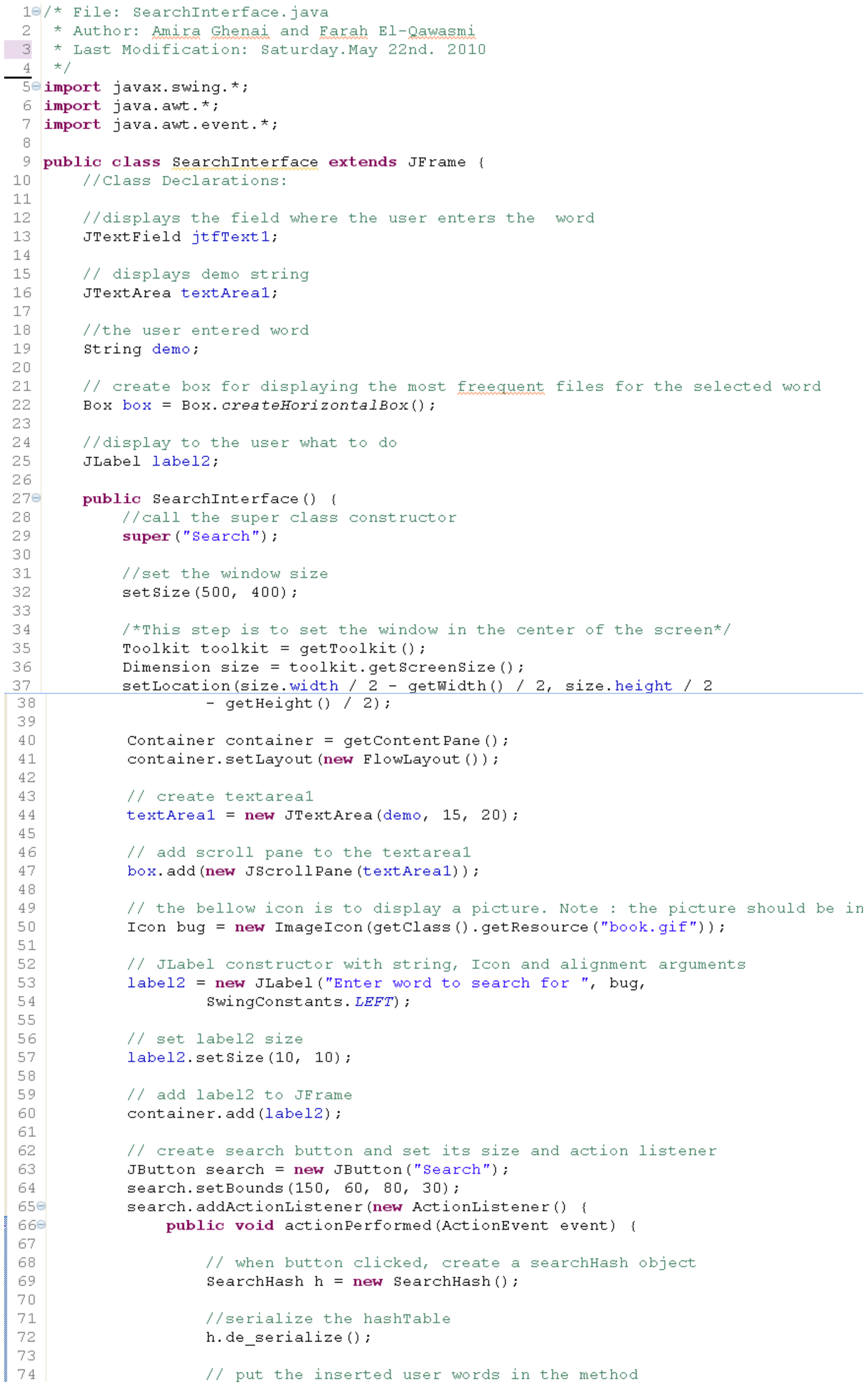


```java
/* File: SearchHash.java
 * Author: Amira Ghenai,Farah El-Qawasmi and Nadia Al-Okka
 * Last Modification: Saturday.May 22nd. 2010
 */
import java.io.*;
import java.util.*;
import java.util.Hashtable;

public class SearchHash {
    private static Hashtable<String, String> hashTable;
    public SearchHash() {
        hashTable = new Hashtable<String, String>();
    }
    public void printHash() {
        System.out.println("Printing out loaded elements...");
        for (Enumeration e = hashTable.keys(); e.hasMoreElements();) {
            Object obj = e.nextElement();
            System.out.println("   - Element(" + obj + ") = "
                    + hashTable.get(obj));
        }
    }
    //To add an element to a HashTable object
    public void put(String K, String V) {
        hashTable.put(K, V);
    }
    public Boolean containsKey(String k) {
        return (hashTable.containsKey(k));
    }
    public String get(String k) {
        return (hashTable.get(k));
    }
    public String word_search(String word) {
        SortedList freFile = new SortedList();
        if (!(hashTable.containsKey(word.toLowerCase())))
            return ("The word does not exist in the files..");
        else {
            //If the word exists in the hash table,
            //get each file with its frequency
            String x = hashTable.get(word.toLowerCase());
            StringTokenizer itr = new StringTokenizer(x.toLowerCase(), "##");
            while (itr.hasMoreTokens()) {
                String fileFree = itr.nextToken();
                String [] temp=new String[2];
                temp = fileFree.split("%");
                int fee = Integer.parseInt(temp[1]);
                freFile.insertFree(temp[0], fee);
            }
            if(freFile.numOfElement()<10)
            {
            return (freFile.displayMostFreequent(freFile.numOfElement()));
            }
            else
                return (freFile.displayMostFreequent(10));
        }
    }
    //This method is used for de_serialize the HashTable,
    //i.e.convert (.SER file)to a Hashtable object
    public void de_serialize()
    {
        try {
            FileInputStream fileIn = new FileInputStream("C:/Output.ser");
            ObjectInputStream in = new ObjectInputStream(fileIn);
            //Load Hashtable Object
            hashTable = (Hashtable)in.readObject();
            //Close all input streams
            in.close();
            fileIn.close();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
```

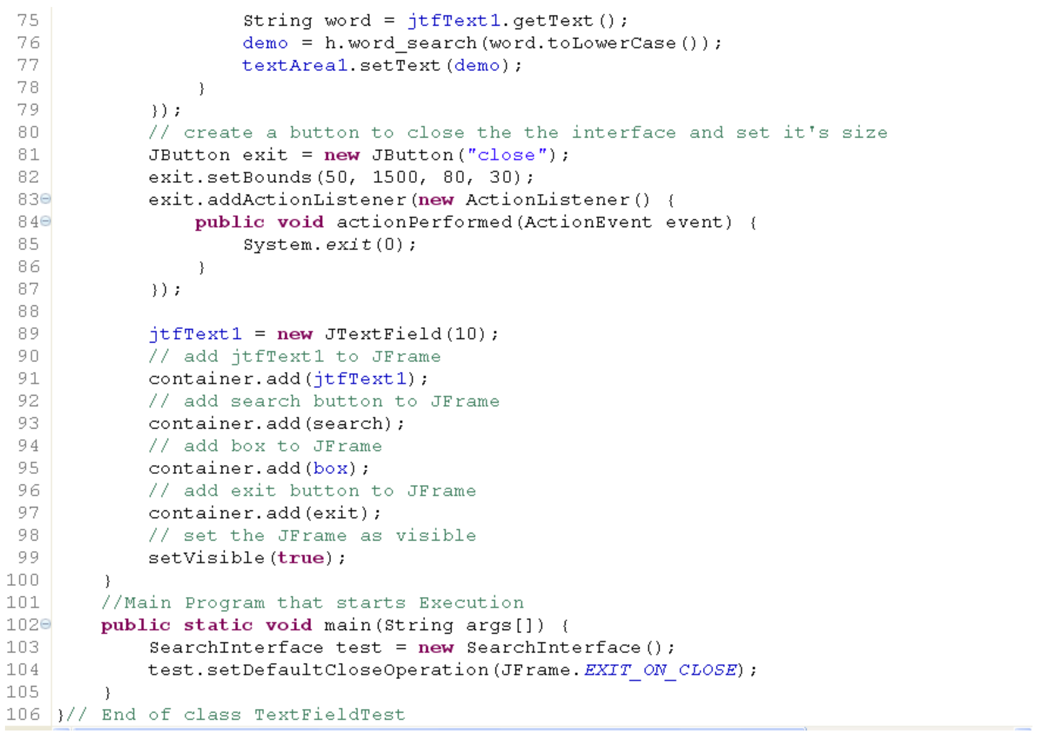


```
        }
    }//End of class
}//End of class SearchHash
```

7. SearchInterface.java

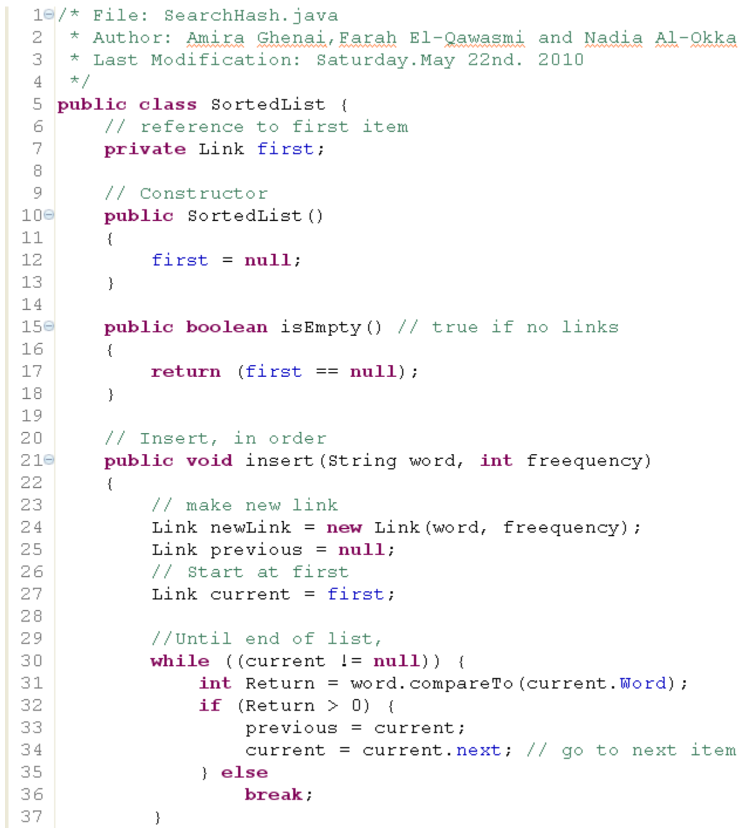


```java
/* File: SearchInterface.java
 * Author: Amira Ghenai and Farah El-Qawasmi
 * Last Modification: Saturday.May 22nd. 2010
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SearchInterface extends JFrame {
    //Class Declarations:

    //displays the field where the user enters the  word
    JTextField jtfText1;

    // displays demo string
    JTextArea textArea1;

    //the user entered word
    String demo;

    // create box for displaying the most freequent files for the selected word
    Box box = Box.createHorizontalBox();

    //display to the user what to do
    JLabel label2;

    public SearchInterface() {
        //call the super class constructor
        super("Search");

        //set the window size
        setSize(500, 400);

        /*This step is to set the window in the center of the screen*/
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2
                - getHeight() / 2);

        Container container = getContentPane();
        container.setLayout(new FlowLayout());

        // create textarea1
        textArea1 = new JTextArea(demo, 15, 20);

        // add scroll pane to the textarea1
        box.add(new JScrollPane(textArea1));

        // the bellow icon is to display a picture. Note : the picture should be in
        Icon bug = new ImageIcon(getClass().getResource("book.gif"));

        // JLabel constructor with string, Icon and alignment arguments
        label2 = new JLabel("Enter word to search for ", bug,
                SwingConstants.LEFT);

        // set label2 size
        label2.setSize(10, 10);

        // add label2 to JFrame
        container.add(label2);

        // create search button and set its size and action listener
        JButton search = new JButton("Search");
        search.setBounds(150, 60, 80, 30);
        search.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {

                // when button clicked, create a searchHash object
                SearchHash h = new SearchHash();

                //serialize the hashTable
                h.de_serialize();

                // put the inserted user words in the method
```

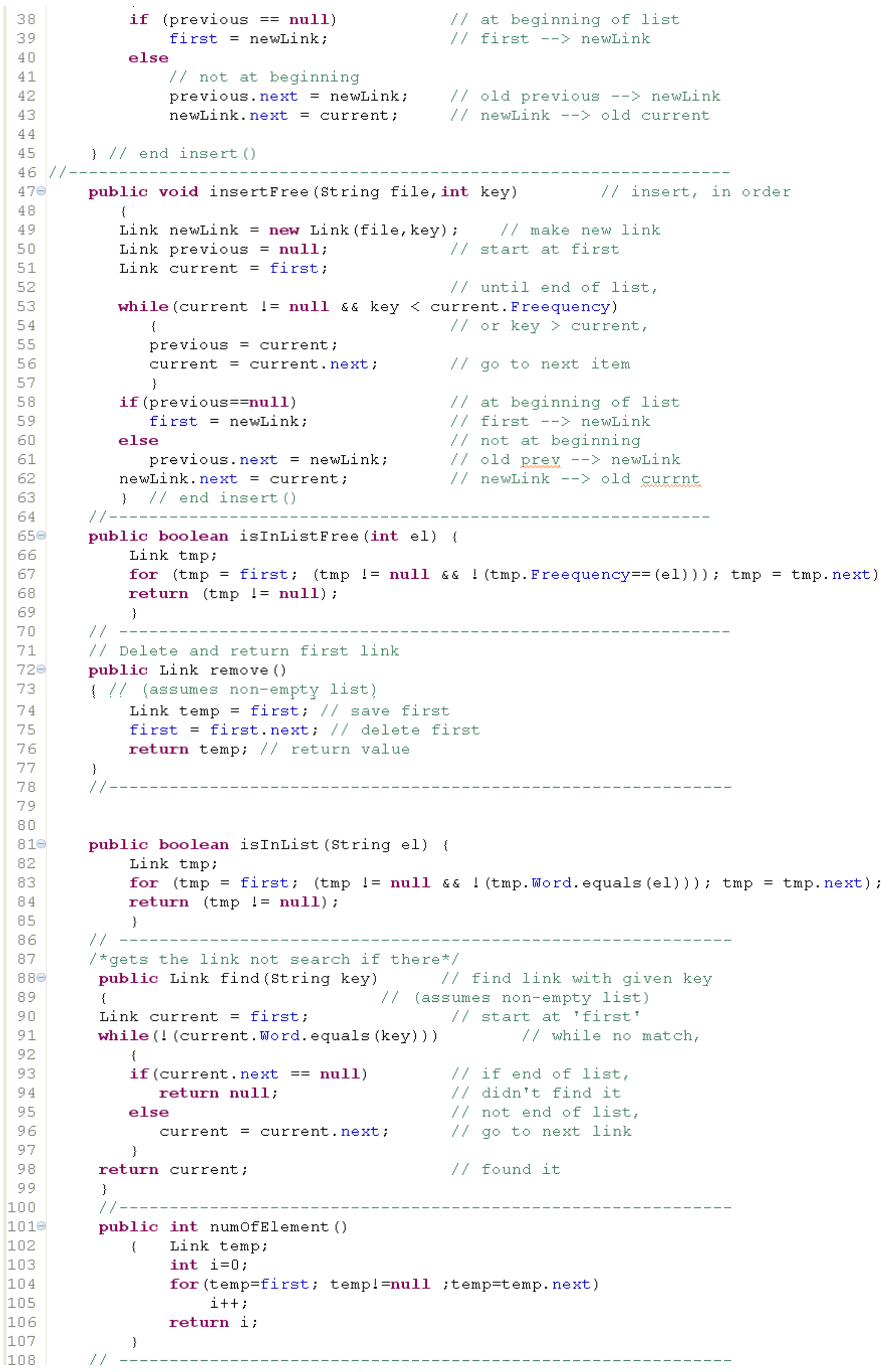


```
                String word = jtfText1.getText();
                demo = h.word_search(word.toLowerCase());
                textArea1.setText(demo);
            }
        });
        // create a button to close the the interface and set it's size
        JButton exit = new JButton("close");
        exit.setBounds(50, 1500, 80, 30);
        exit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        jtfText1 = new JTextField(10);
        // add jtfText1 to JFrame
        container.add(jtfText1);
        // add search button to JFrame
        container.add(search);
        // add box to JFrame
        container.add(box);
        // add exit button to JFrame
        container.add(exit);
        // set the JFrame as visible
        setVisible(true);
    }
    //Main Program that starts Execution
    public static void main(String args[]) {
        SearchInterface test = new SearchInterface();
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}// End of class TextFieldTest
```

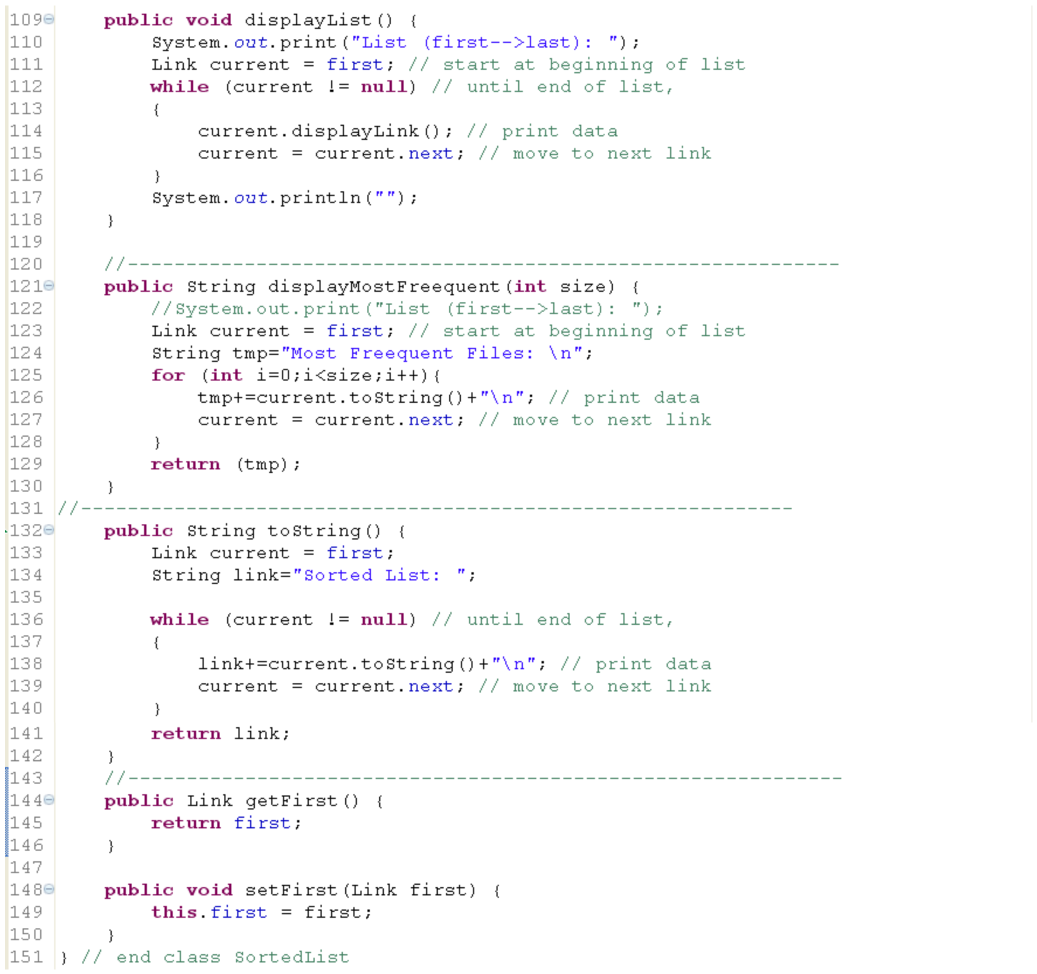
8. SortedList.java



```java
/* File: SearchHash.java
 * Author: Amira Ghenai,Farah El-Qawasmi and Nadia Al-Okka
 * Last Modification: Saturday.May 22nd. 2010
 */
public class SortedList {
    // reference to first item
    private Link first;

    // Constructor
    public SortedList()
    {
        first = null;
    }

    public boolean isEmpty() // true if no links
    {
        return (first == null);
    }

    // Insert, in order
    public void insert(String word, int freequency)
    {
        // make new link
        Link newLink = new Link(word, freequency);
        Link previous = null;
        // Start at first
        Link current = first;

        //Until end of list,
        while ((current != null)) {
            int Return = word.compareTo(current.Word);
            if (Return > 0) {
                previous = current;
                current = current.next; // go to next item
            } else
                break;
        }
```

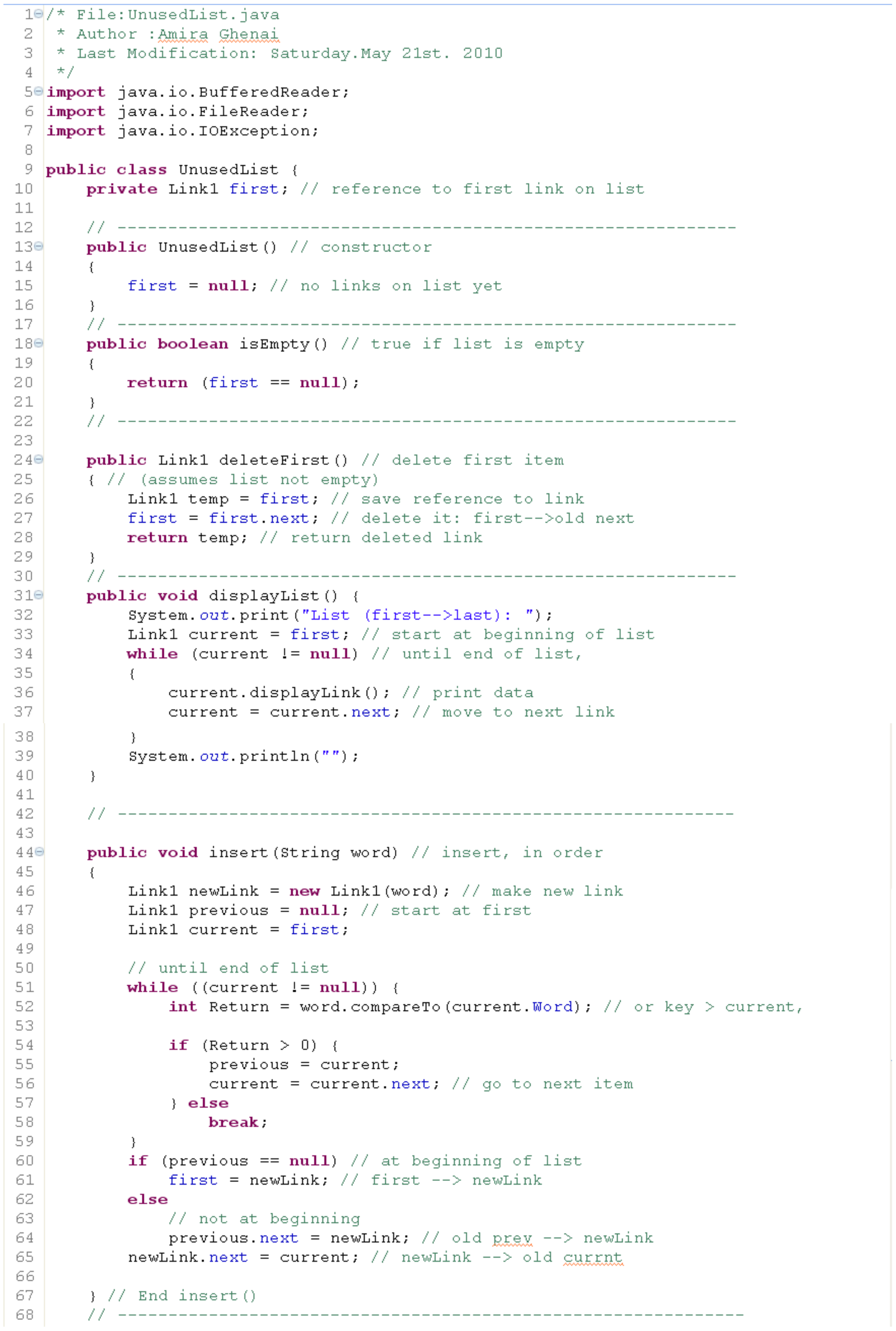


```
        if (previous == null)           // at beginning of list
            first = newLink;            // first --> newLink
        else
            // not at beginning
            previous.next = newLink;    // old previous --> newLink
            newLink.next = current;     // newLink --> old current

    } // end insert()
//-------------------------------------------------------------------
    public void insertFree(String file,int key)        // insert, in order
       {
       Link newLink = new Link(file,key);     // make new link
       Link previous = null;            // start at first
       Link current = first;
                                        // until end of list,
       while(current != null && key < current.Freequency)
          {                             // or key > current,
          previous = current;
          current = current.next;       // go to next item
          }
       if(previous==null)               // at beginning of list
          first = newLink;              // first --> newLink
       else                             // not at beginning
          previous.next = newLink;      // old prev --> newLink
       newLink.next = current;          // newLink --> old currnt
       }  // end insert()
    //------------------------------------------------------------
    public boolean isInListFree(int el) {
        Link tmp;
        for (tmp = first; (tmp != null && !(tmp.Freequency==(el))); tmp = tmp.next)
        return (tmp != null);
        }
    // -------------------------------------------------------------
    // Delete and return first link
    public Link remove()
    { // (assumes non-empty list)
        Link temp = first; // save first
        first = first.next; // delete first
        return temp; // return value
    }
    //--------------------------------------------------------------


    public boolean isInList(String el) {
        Link tmp;
        for (tmp = first; (tmp != null && !(tmp.Word.equals(el))); tmp = tmp.next);
        return (tmp != null);
        }
    // -------------------------------------------------------------
    /*gets the link not search if there*/
     public Link find(String key)      // find link with given key
     {                           // (assumes non-empty list)
     Link current = first;              // start at 'first'
     while(!(current.Word.equals(key)))        // while no match,
        {
        if(current.next == null)        // if end of list,
           return null;                 // didn't find it
        else                            // not end of list,
           current = current.next;      // go to next link
        }
     return current;                    // found it
     }
     //-------------------------------------------------------------
     public int numOfElement()
        {   Link temp;
            int i=0;
            for(temp=first; temp!=null ;temp=temp.next)
                i++;
            return i;
        }
    // -------------------------------------------------------------
```

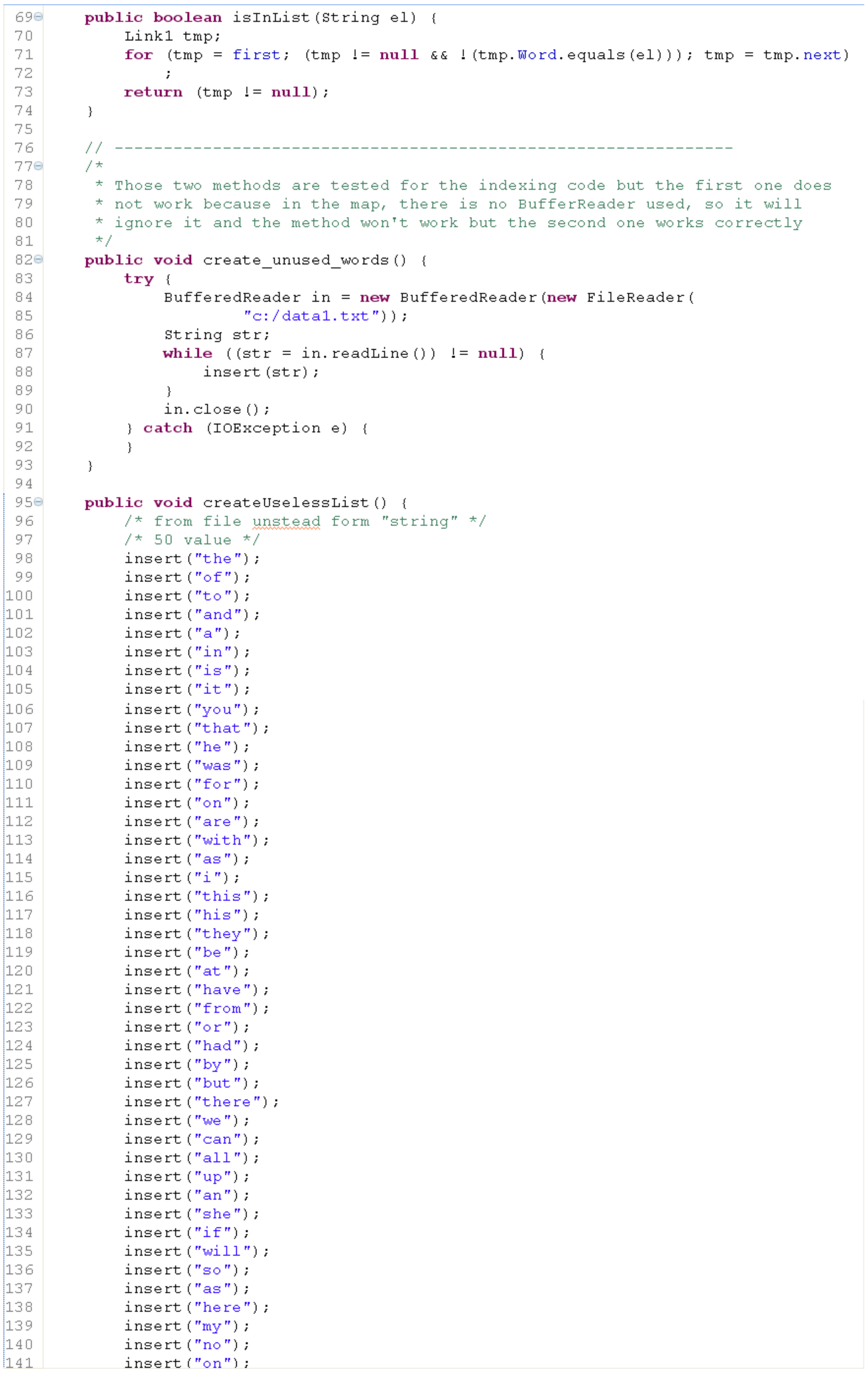


```java
    public void displayList() {
        System.out.print("List (first-->last): ");
        Link current = first; // start at beginning of list
        while (current != null) // until end of list,
        {
            current.displayLink(); // print data
            current = current.next; // move to next link
        }
        System.out.println("");
    }

    //-------------------------------------------------------------
    public String displayMostFreequent(int size) {
        //System.out.print("List (first-->last): ");
        Link current = first; // start at beginning of list
        String tmp="Most Freequent Files: \n";
        for (int i=0;i<size;i++){
            tmp+=current.toString()+"\n"; // print data
            current = current.next; // move to next link
        }
        return (tmp);
    }
//-------------------------------------------------------------
    public String toString() {
        Link current = first;
        String link="Sorted List: ";

        while (current != null) // until end of list,
        {
            link+=current.toString()+"\n"; // print data
            current = current.next; // move to next link
        }
        return link;
    }
    //-------------------------------------------------------------
    public Link getFirst() {
        return first;
    }

    public void setFirst(Link first) {
        this.first = first;
    }
} // end class SortedList
```

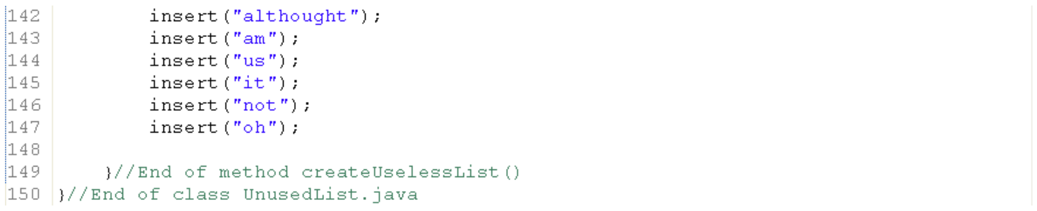
9. *UnusedList.java*



```java
/* File:UnusedList.java
 * Author :Amira Ghenai
 * Last Modification: Saturday.May 21st. 2010
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class UnusedList {
    private Link1 first; // reference to first link on list

    // -------------------------------------------------------------
    public UnusedList() // constructor
    {
        first = null; // no links on list yet
    }
    // -------------------------------------------------------------
    public boolean isEmpty() // true if list is empty
    {
        return (first == null);
    }
    // -------------------------------------------------------------

    public Link1 deleteFirst() // delete first item
    { // (assumes list not empty)
        Link1 temp = first; // save reference to link
        first = first.next; // delete it: first-->old next
        return temp; // return deleted link
    }
    // -------------------------------------------------------------
    public void displayList() {
        System.out.print("List (first-->last): ");
        Link1 current = first; // start at beginning of list
        while (current != null) // until end of list,
        {
            current.displayLink(); // print data
            current = current.next; // move to next link
        }
        System.out.println("");
    }

    // -------------------------------------------------------------

    public void insert(String word) // insert, in order
    {
        Link1 newLink = new Link1(word); // make new link
        Link1 previous = null; // start at first
        Link1 current = first;

        // until end of list
        while ((current != null)) {
            int Return = word.compareTo(current.Word); // or key > current,

            if (Return > 0) {
                previous = current;
                current = current.next; // go to next item
            } else
                break;
        }
        if (previous == null) // at beginning of list
            first = newLink; // first --> newLink
        else
            // not at beginning
            previous.next = newLink; // old prev --> newLink
        newLink.next = current; // newLink --> old currnt

    } // End insert()
    // -------------------------------------------------------------
```



```java
    public boolean isInList(String el) {
        Link1 tmp;
        for (tmp = first; (tmp != null && !(tmp.Word.equals(el))); tmp = tmp.next)
            ;
        return (tmp != null);
    }

    // ---------------------------------------------------------------
    /*
     * Those two methods are tested for the indexing code but the first one does
     * not work because in the map, there is no BufferReader used, so it will
     * ignore it and the method won't work but the second one works correctly
     */
    public void create_unused_words() {
        try {
            BufferedReader in = new BufferedReader(new FileReader(
                    "c:/data1.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                insert(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }

    public void createUselessList() {
        /* from file unstead form "string" */
        /* 50 value */
        insert("the");
        insert("of");
        insert("to");
        insert("and");
        insert("a");
        insert("in");
        insert("is");
        insert("it");
        insert("you");
        insert("that");
        insert("he");
        insert("was");
        insert("for");
        insert("on");
        insert("are");
        insert("with");
        insert("as");
        insert("i");
        insert("this");
        insert("his");
        insert("they");
        insert("be");
        insert("at");
        insert("have");
        insert("from");
        insert("or");
        insert("had");
        insert("by");
        insert("but");
        insert("there");
        insert("we");
        insert("can");
        insert("all");
        insert("up");
        insert("an");
        insert("she");
        insert("if");
        insert("will");
        insert("so");
        insert("as");
        insert("here");
        insert("my");
        insert("no");
        insert("on");
```



```
        insert("althought");
        insert("am");
        insert("us");
        insert("it");
        insert("not");
        insert("oh");

    }//End of method createUselessList()
}//End of class UnusedList.java
```

APPENDIX E

Block 3 in Generate-Part 2 Flowchart

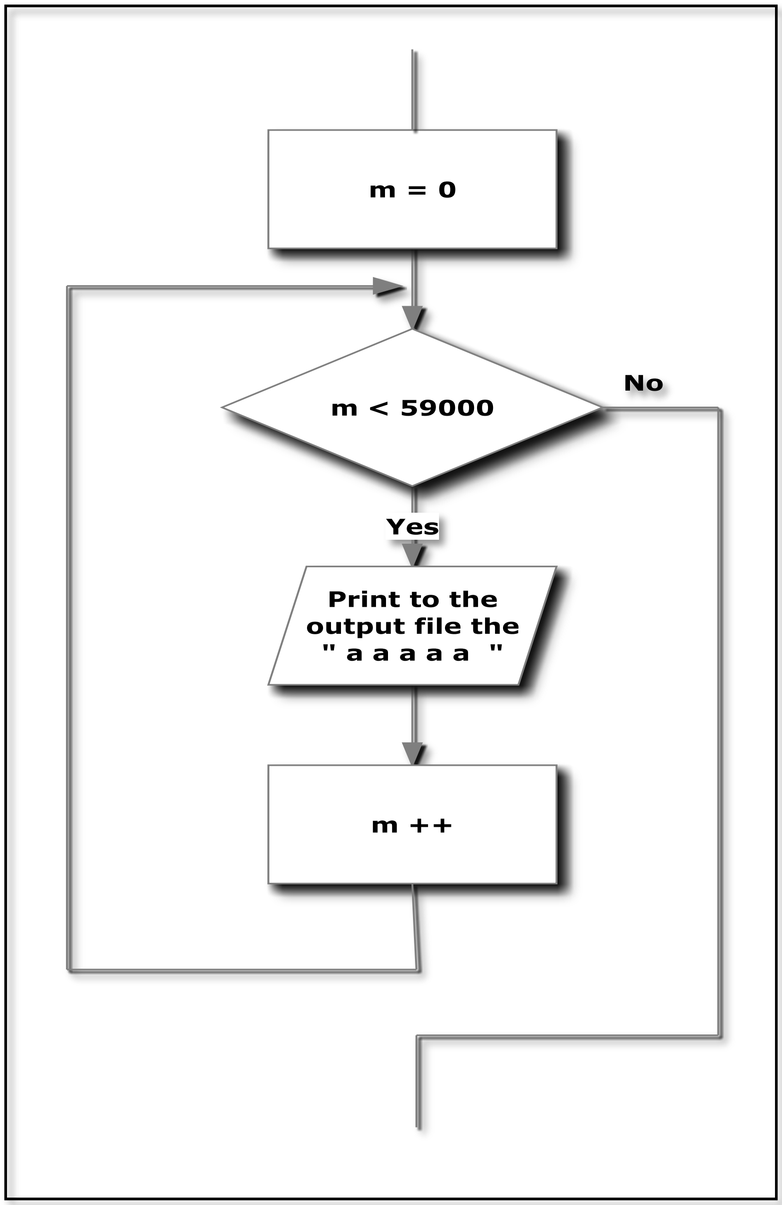


Figure E.1:Block 1 in Generate-Part 2 Flowchart

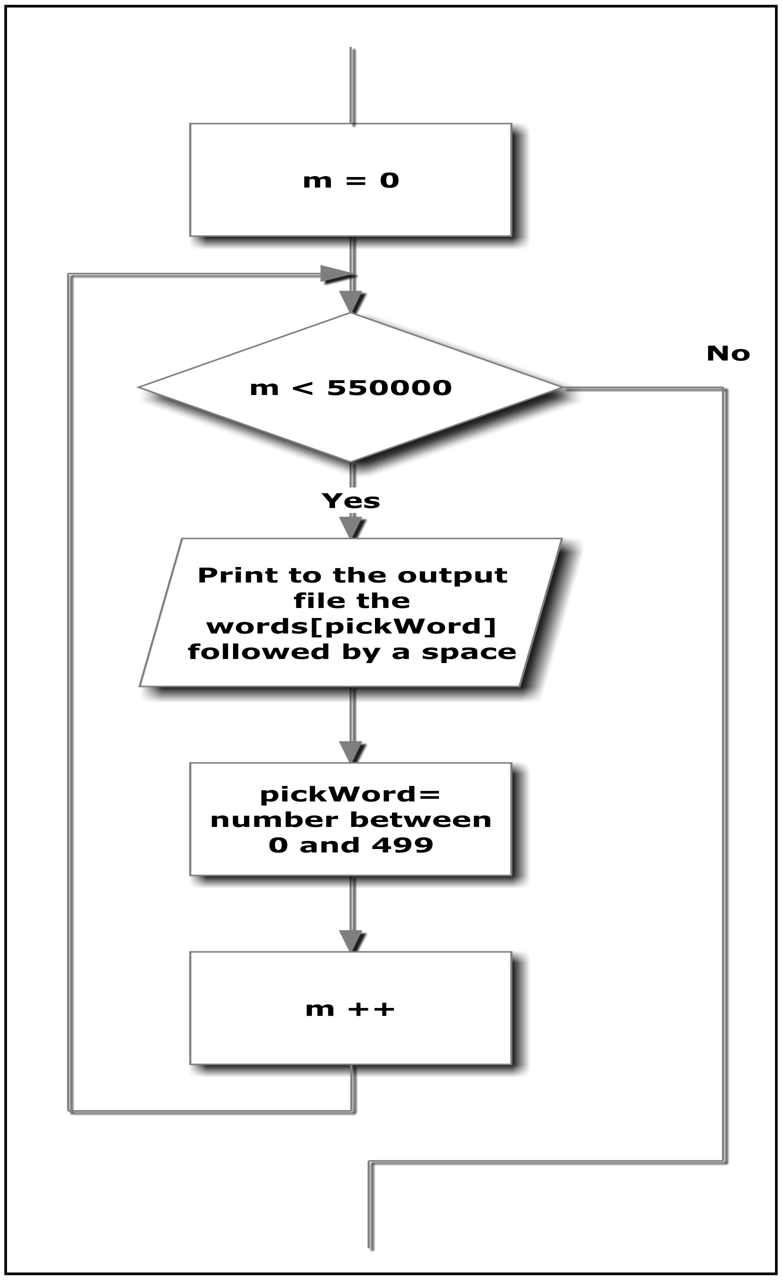

m = 0
m < 550000
No
Yes
Print to the output file the words[pickWord] followed by a space
pickWord= number between 0 and 499
m ++

Figure E.2: Block 2 in Generate-Part 2 Flowchart

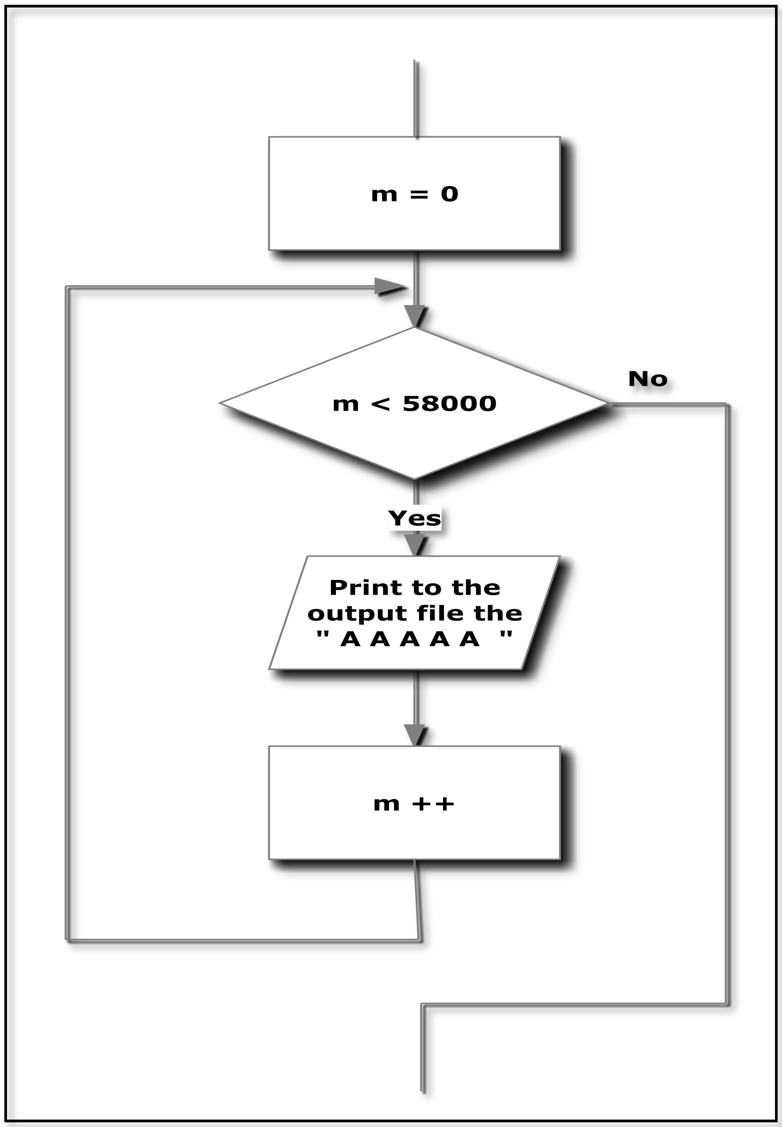


Figure E.3: Block 3 in Generate-Part 2 Flowchart

APPENDIX F

MEETINGS AND BULLETIN BOARD

Meeting no.1:

Tuesday 6th .October.09 – (09:00-10:00 am)

- **Items discussed:**
  - The idea of the project
  - The main tasks for the project.
  - General idea about the data mining and map reduce.
  - Working on the Cloud in Carnegie Mellon
- **Tasks agreed for next meeting:**

Task	For
• Write the UREP project proposal • Searching and reading about the data mining and map reduce ○ Techniques, algorithms and applications	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

- **Others**
  - Dr. Malluhi sent an email to QP to have the permission to get the data and to arrange meeting with them.
  - Hardware of the cloud was ordered to be instal1ed in Qatar university (QU).

جامعة قطر
QATAR UNIVERSITY

Meeting no.2

Tuesday 13th .October.09 – (09:00-10:00 am)

- **Items discussed:**
  - Downloading SSH program to login into the cloud (PUTTY)
- **Tasks completed:**

Task	For
• Reading many articles and power point presentation about data mining and map reduce and made some notes. • Filing the project proposal template (not complete)	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Logging to the cloud using the accounts and passwords given from • To be familiar with the cloud and how to run programs on it • Reading more about the map reduce and data mining	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

- **Others**
  - User names, passwords and login instructions has been sent to the students emails.

Meeting no.3:

Wednesday 28th .October.09 – (2:00-2:30 pm)

- **Items discussed:**
  - project proposal
  - Running the browser in the cloud
  - Map reduce environment.
- **Tasks completed:**

Task	For
• Logging to the cloud using the accounts and passwords. • Filing the project proposal template. • Some reading about map reduce and data mining	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Run a map reduce example, then try to modify it and run it again • To be familiar with the WEKA software	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

- **Others**
  - Drop box is used to upload the needed files for the project easily.

Meeting no.4:

Tue 3rd .November.09 – (09:00-10:00 am)

- **Items discussed:**
  - The meetings IBM.
  - Problems in running a java program importing a Hadoop or weka packages
  - Proposal of the project is completed and submitted.
- **Tasks completed:**

Task	For
• Exploring WEKA software • Finding many MapReduce examples	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Contacting with Mr.Shuja to access the ssh in the QU campus • Contacting with CMUQ to use the URL • Installing the Hadoop package • Run a map reduce example, then try to modify it and run it again • To be familiar with the WEKA software	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

- **Others**
  - We contact Dr. AbdulKarim Errdai to have a help with Hadoop.

Meeting no.5:

Tuesday 17th .November.09 – (09:00-10:00 am)

- **Items discussed:**
  - Qatar Qloud Hardware (installation on 6th -7th, Dec.[09
  - Training and tutorial on 8th-9th-10th ,Dec.2009
  - Meeting with Dr. AbdulKarim Errdai and some useful information
  - Contacting QP and the preparation of some presentations.
  - Plan “B”: if no data is available from QP, an indexing procedure may be discussed as a plan “B”
  - Many Problems in installing the Hadoop package and interacting with eclipse
- **Tasks completed:**

Task	For
• Contacting with Mr.Shuja to access the ssh in the QU campus • Contacting with CMUQ to use the URL • Installing the Hadoop package • Trying to run a MapReduce example	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Complete installing the Hadoop package and run the examples included in the package.	Amira Farah Nadia

- **Attendance (Students):**
  Amira- Farah- Nadia

Meeting no.6:

Tuesday 24th .November.09 – (09:00-10:00 am)

- **Items discussed:**
  - Contacting QP and some meetings.
  - Many Problems in installing the Hadoop package and interacting with eclipse

- **Tasks completed:**

Task	For
• Installing the Hadoop but with many problems such as how to connect our virtual machine with our real machine, and this solved by using the ssh (Tunnelier)	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Complete installing the Hadoop package, solving the remained problems with Mr.Brian Geek(CMUQ) and run the examples included in the package.	Amira Farah Nadia

- **Attendance (Students):**

Amira- Farah- Nadia

- **Others:**

The meeting with Mr.Brian is arranged to be on Tue. 24th ,Nov.2009 at 12:00pm in CMUQ compus .

Meeting no.7:

Tuesday 8th .December.09 – (09:00-10:00 am)

- **Items discussed: (should be modified)**
  - Discuss some points regarding the first senior project report such as the problem statement, the goals and main objectives, the project scope.
  - Take the decision that QGPS data from QP will not be used and another data sourse should be found.
  - Plan B: Getting Data from Other Resource (e.g.: United Nation Documents)
- **Tasks completed:**

Task	For
• Finishing installing the local Hadoop. • Visit Mr.Brian Geek(CMUQ) and access Hadoop of CMU cloud.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Look and search for other data recourses: United Nations. • Solve the Eclipse plug-ins problem of MapReduce on the Eclipse environment. • Start Documentation of the project.	Amira Farah Nadia

- **Attendance (Students):**

Amira- Farah- Nadia

جامعة قطر
QATAR UNIVERSITY

Meeting no.8:

Tuesday 13th .Decembre.09 – (09:00-10:00 am)

- **Items discussed: (should be modified)**
  - Discuss some points regarding the first senior project report.
- **Tasks completed:**

Task	For
• Completed some documentation of the project such as the cloud and Hadoop literature.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Continue Documentation of the project. • Attend the IBM training which will be scheduled on the 15th of December.	Amira Farah Nadia

- **Attendance (Students):**

Amira- Farah

Meeting no.9:

Sunday 21st .February.10 – (11:00-12:00 am)

- **Items discussed:**
  - The accounts status in the QU cloud and the VPN configuration.
  - Setting up the meeting time for this semester.
  - Discussing some issues regarding the indexing concept.
  - How to gather 1 TB of data: the suggested solusion was a crawler program.
  - The WordCount code :discuss some problems in the code and how to run it on the QU cloud.
- **Tasks completed:**

Task	For
• Decided to work on the indexing example for MapReduce. • Attended some lectures in the CMU university for the course “Introduction to Cloud Computing” presented by Dr.Majd F.Sakr.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Contact Mr. Shuja Ashfaq for setting the VPN to connect to the cloud outside the campus. • Contact Mr. Zeyad Ali to see the accounts status in the QU cloud. • Fix the problems in the WordCount code and run it without the use of the QU cloud. • Start reading about the indexing algorithms, find some examples and understand and try to run them. • Understand the crawler java code and run it correctly to start gathering the needed amount of data.	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia
- **Others:**

Contact Dr.ElSayed to get a help with the Crawler4j code

جامعة قطر
QATAR UNIVERSITY

Meeting no.10:

Sunday 28th .February.10 – (11:00-12:00 am)

- **Items discussed:**
  - Preparing the senior project computer to run the crawler program on it
  - Crawler program is working successfully
  - Failed to connect the cloud to run the MapReduce program
  - WordCount.java without errors
  - Cloud accounts are customer accounts, cannot create projects
  - Emailing Mr.Zeyad Ali to create a Hadoop project

- **Tasks completed:**

Task	For
• Crawler program worked successfully. • Contacted Miss Sara(IT) to prepare the senior project computer to run the crawler program on it. • Get customer accounts on cloud • Found some useful papers about indexing with MapReduce algorithms	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Contact Mr. Shuja Ashfaq for setting the VPN to connect to the cloud outside the campus • Try again to connect the local Hadoop with eclipse to run the WordCount example	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

Meeting no.11:

Sunday 7th .March.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Buy hard disks of 1T.
  - Problems with connecting cloud Hadoop with local eclipse.

- **Tasks completed:**

Task	For
• VPN is working properly. • Connect local Hadoop with eclipse but with errors. • Partially connect cloud Hadoop with local eclipse (with errors). • Started reading some papers about indexing.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Try to solve the password problem faced with connecting eclipse with cloud Hadoop by starting each node individually. • Look for more simple and specific published scientific papers about indexing using MapReduce algorithm. (suggested search engine: Lucene)	Amira Farah Nadia

- **Attendance (Students):**
  Amira- Farah- Nadia

Meeting no.12:

Thursday 11nd .March.10 – (08:30-09:00 am)

- **Items discussed:**
  - Problems with connecting cloud Hadoop with local eclipse.
  - Difficulties in finding two compatible versions of Hadoop and eclipse
  - Suggested solutions in different web sites.

- **Tasks completed:**

Task	For
• Try to solve the problems with the local Hadoop (solved partially) • Try to connect two different version of Hadoop(0.16.4) and eclipse(3.4) errors in Java scripts in this version of Hadoop.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Send an e-mail to Mr.Brian to give us the Hadoop folder used in the CMU with the eclipse used there. • Request new accounts in the CMU cloud with a project with specific requirements and also new accounts in the Hadoop server of the CMU	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

Meeting no.13:

Sunday 14th .March.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Problems with connecting cloud Hadoop, local Hadoop with local eclipse.
  - Inverted Indexing
  - Find some problems regarding the local Hadoop installed in our QU cloud.
  - A visit to the QU IT people to tell them about what we need for the Hadoop server, and show them some of the faced problems in the cloud.

- **Tasks completed:**

Task	For
• E-mail was sent to Mr. Brain asking for the Hadoop folder and creating accounts.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Read about inverted indexing, and try to find codes for doing this • Read about some MapReduce examples that implements the indexing. • Read about tokenization. • Order a useful book about MapReduce or Hadoop.	Amira Farah Nadia

- **Attendance (Students):**
  Amira- Farah- Nadia

Meeting no.14:

Sunday 21st .March.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Details in the code (tokenization and delimiters)
  - The visit to CMU and the help from Mr.Suheel
  - The algorithm of the inverted index
- **Tasks completed:**

Task	For
• Read about inverted indexing, and a good code from yahoo was found • Read about some MapReduce examples that implements the indexing. • Read about tokenization. • A useful book about MapReduce or Hadoop is bought.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Get the right directories (Hadoop-0.20.1 and eclipse from Mr. Brian from CMU) • Work on enhancing the indexing code to meet some specifications.	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

جامعة قطر
QATAR UNIVERSITY

Meeting no.15:

Sunday 28th .March.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Trying to run codes on separate clusters
  - Searching about generating random text files.
  - Running the crawler on the cloud
  - QU cloud is not sattled yet and our project is deleted.
  - Discussing some topics about the meeting with Dr.Erradi.

- **Tasks completed:**

Task	For
• Got the right directories (Hadoop-0.20.1 and eclipse from Mr. Brian from CMU) • Gave the directories to the IT team to be configured on the cloud	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Make some design decisions about the application and its interface • Generate random text files • Complete the not completed tasks from the previous meeting	Amira Farah Nadia

- **Attendance (Students):**
  Amira- Farah- Nadia

Meeting no.16:

Sunday 4th .April.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Discuss the QU cloud status (problem solved and wait for Dr. Qutaiba to contact the IT group).
  - Take the data from Al-Jazira Networks.
  - Discuss the indexing application with the use of search (simple) and determining the frequency for each word in each file.
  - Starting to work on the final report.

- **Tasks completed:**

Task	For
• Ran the crawler on the cloud by making the whole project as a runnable jar file(slow not that efficient)	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Choose the final idea (path) of the application if it is going to be an analysis improvement or application based. • Look for algorithms related to security that can be implemented in embarrassingly parallel way and easy to get then, test them (try to talk and discuss the idea with Dr. Ryan Riley). • Check cloud connectivity using the ping command. • Think of finding another data source (by data from the net). • Start working on the final report for the senior project.	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

Meeting no.17:

Sunday 18th –Thursday 22nd .April.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - How to deal with a tokenizer, and skip the delimiters.
  - The structure of the output files from the map and reduce methods, hash tables are suggested.
  - How to find the frequency of a token in one file.
    - Find this frequency in the map method, because in the map function each file is processed alone.
  - Discuss the interface of the project:
    - Using web Interface.
    - Let the user to choose the parameters of the MapReduce function, the input, output paths.
  - Discuss the problems faced with the crawler.
  - A visit to the QU IT team to follow up with them about the condition of the cloud, and to test if we can use it depending on their work.(The CMU cloud accounts will be terminated on 20th April, 2010).

- **Tasks completed:**

Task	**For**
• Finalize our idea (path) of the project to an application based. • Fix some problems in the crawler by understanding some things about the staring path of search and know how to deal with it. • Run the crawler code from the cloud, and save the output in the cloud (the connection using the CMU cloud was slow). • Start working on the final report for the senior project, and find out what is missing from the report.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	**For**
• Apply the discussed method for the delimiters on the indexing code. • Read about hashing tables, and how to save .txt file into hashed table. • Find out how to get the frequency of the token in each file. • Visit to the QU IT people to follow up with them about the condition of the cloud.	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Farah- Nadia

Meeting no.18:

Sunday 25th .April.10 – (11:00am-12:00 pm)

- **Items discussed:**
  - Problems in the crawler
  - Problem in creating the sorted list in the code.
  - Other problems in the code.

- **Tasks completed:**

Task	For
• The discussed method for the delimiters on the indexing code was applied and it worked. • Read about hashing tables, and how to save .txt file into hashed table. • The visit to the IT was useful and the cloud finally is ready.	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Work on the crawler to solve its problem • Continue working in the coding part • Start working on the final report	Amira Farah Nadia

- **Attendance (Students):**

  Amira- Nadia

Meeting no.19:

Sunday 2nd .May.2010 – (11:00am-12:00 pm)

- **Items discussed:**
- Getting data from the united nations website instead of the crawler because it is very slow
- Coding:
  - Making the unused list in a configuration file not in the code itself.
  - Comparing Strings in Java.
  - The output of the map has only 2 variables, we need more..so we would create new class , the object of this class has 2 variables frequency and file name

Tasks completed:

Task	For
• Editing the report and completing • Looking for another source of data (buy or another crawler) • Working on the code .. still working and debugging	Amira Farah Nadia

Tasks agreed for next meeting:

Task	For
• Collect data manually • Continue working on the code	Amira Farah Nadia

Attendance (Students):

Amira- Farah- Nadia

Meeting no.20:

Sunday 9th .May.2010 – (11:00am-12:20 pm)

- **Items discussed:**
  - The mapper is working probably
  - Search on a hash tables is working
  - Hash tables in the reduce is not working
  - The interface options (Web application or GUI component)

- **Tasks completed:**

Task	For
• Collecting data manually(very very slow and inefficient) • A fast B plane : make a list of about 160 URLs to feed the crawler • Deeper look to the configuration for the MapReduce programs • Parts of the code - Mapper - Searching in Hash Tables - Listing files according to the frequency	Amira Farah Nadia

- **Tasks agreed for next meeting:**

Task	For
• Modify the crawler to get URLs from a file and run it • Complete the reduce part of the code • Make the interface as simple as possible	Amira Farah Nadia

- **Attendance (Students):**

Amira- Farah- Nadia

جامعة قطر
QATAR UNIVERSITY

Meeting no.21

Sunday 16th .May.2010 – (11:00am-12:20 pm)

- **Items discussed:**
  - Gathering data using generating files.
  - The codes of Indexing MapReduce Algorithm, converting .txt output to Hash tables then to .ser files, searching code and the designed interface.
  - The number of mappers and reducers tasks in the LineIndexer code.
  - Discussion of the report most important points.

Tasks completed:

Task	For
• The crawler was modified. But we end with a result that it is not effective. • Reduce part of the coded is completed. • User Interface for searching is designed.	Amira Farah Nadia

Tasks agreed for next meeting:

Task	For
• Write the code of generating files. • Generate files of total size of 0.5 TB. • Write the documentation of the Project	Amira Farah Nadia

Attendance (Students):

Amira- Farah- Nadia

جامعة قطر
QATAR UNIVERSITY

Meeting no.22

Sunday 23th .May.2010 – (9:00am-10:00 pm)

- **Items discussed:**
  - The unstable state of Qatar University cloud.
  - Performance testing plan.
  - Creating accounts on the CMU-Qatar again, so we could test our codes.

Tasks completed:

Task	For
• Generation files code is done, and data is gathered. • Number of chapters are written in the final report • Reduce part of the coded is completed. • User Interface for searching is designed.	Amira Farah Nadia

Tasks agreed for next meeting:

Task	For
• Finalize the project documentation. • Try to test the performance after the submitting the report.	Amira Farah Nadia

Attendance (Students):

Amira- Farah- Nadia