Project Design

Version 4.0

2 May, 2016

Revision Signatures

By signing the following document, the team member is acknowledging that he has read the entire document thoroughly and has verified that the information within this document is, to the best of his knowledge, accurate, relevant and free of typographical errors.

Name	**Signature**	**Date**
Sushant Ahuja		
Cassio Lisandro Caposso Cristovao		
Sameep Mohta		

Revision History

The following table shows the revisions made to this document.

Version	Changes	Date
1.0	Initial Draft	15 December, 2015
2.0	Updated Iterations	21 January, 2016
3.0	Updated Development tools, Minor changes in Database Designs	20 April, 2016
4.0	Final Minor Changes	2 May, 2016

Table of Contents

1 Introduction

1.1 Purpose

This document provides a complete description of the Frog-B-Data system design. Included in this document are the design constraints, system architecture, user interface design, and Unified Modelling Language (UML) diagrams, defining their state, class and sequence.

1.2 Project Background

Data now streams from everywhere in our daily lives: phones, credit cards, computers, tablets, sensor-equipped buildings, cars, buses, trains and the list goes on and on. We have heard so many people say “There is a Big Data Revolution”. What does that mean? It is not the quantity of data that is revolutionary. The Big Data revolution is that now we can do something with the data. The revolution lies in the improved statistical and computational methods which can be used to make our lives easier, healthier and more comfortable.

Familiar uses of Big Data to a common man include “recommendation engines” used by Netflix and Amazon, credit card companies, and tech giants like Facebook. In the public realm, there are all kinds of applications: allocating police resources by predicting where and when crimes are most likely to occur; finding associations between air quality and health; or using genomic analysis to speed the breeding of crops like rice for drought resistance. However, this is a very small fraction of what can be done and what is being done. The potential for doing good is nowhere greater than in public health and medicine where people are dying everyday just because data is not being properly shared.

Nowadays, it’s not just about mining data and analyzing results, it is about using data smartly. The purpose of smart data is to filter out the noise from the Big Data and hold the valuable data to solve business problems. There are no formulae to convert Big Data into smart data, but if we understand the clues in the questions around the data and analyze data qualitatively, we can use it smartly.

1.3 Section Overview

This document includes the following seven sections:

Section 2 – Design Constraints: This section contains the overall description of the project, its characteristics, functions, operational requirements, its assumptions and dependencies.

Section 3 – System Architecture This section specifies the architecture of the system used by the project.

Section 4 – Database designs: This section provides with an overall view of the database structure in the data mining environment called Hadoop Distributed File System (HDFS).

Section 5 – UML Diagrams: This section displays the state, class and sequence diagrams.

Section 6 – Glossary of Terms: This section lists definitions of the terms used in this document.

Section 7 – Appendices:

Appendix A – User-Case Model

2 Design Constraints

2.1 Assumptions and Dependencies

- We assume that the user will have Linux operating system, preferably Ubuntu 15.04 or above, with Hadoop and Spark installed and basic knowledge of Java and Python programming language along with Eclipse Mars IDE.
- We assume that all the Java programming will be done using Apache Maven in both, Hadoop and Spark. Apache Maven is a key tool for this section of requirements as it defines the dependencies of any Java Project that the user is working on.

2.2 General Constraints

- **Time Constraints:**
  End of school year limits research and testing time.

- **Data Constraints:**
  Very limited availability of 'real' Big Data files that can be processed.

- **Hardware Constraints:**
  Limited number of secondary computers (workers) in manager/worker structure.

- **Operating System Constraints:**
  Need Linux OS to get the best performance results.

- **Root Access Constraints:**
  Root Access is not available all the time for security reasons, especially in public organizations.

2.3 Development Environment

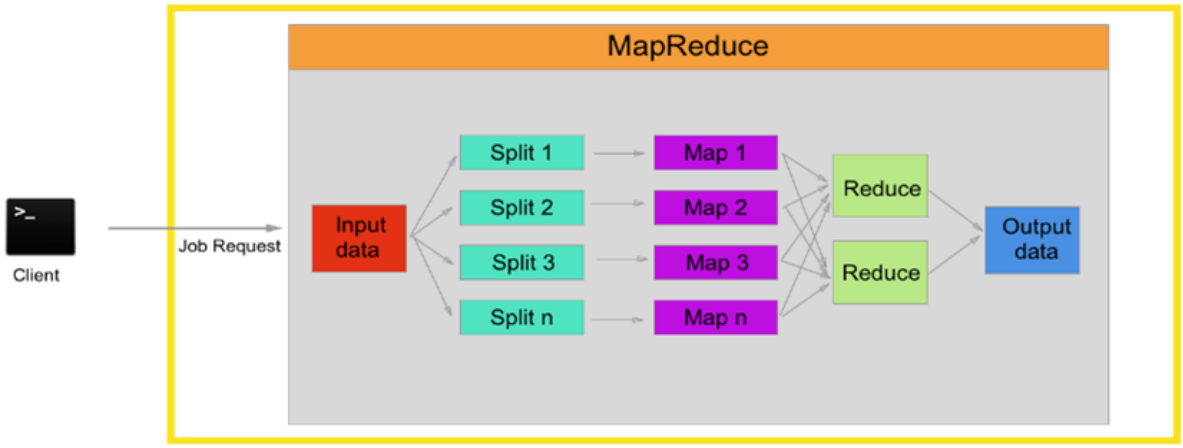
Frog-B-Data can be used on any Linux environment and Eclipse IDE with the help of the user guide. It can be used on either independent machines or in a master/slave system structure.

- **Development tools**
  - Hadoop 2.7.1
  - Spark 1.5.1
  - Java 8.6
  - Eclipse Mars 4.5
  - Scala 2.11.7
  - Python 2.7.9

- **Supporting tools**
  - Github
  - Slack
  - Core FTP 2.2, 1853, 0
  - FileZilla
- **General Utilities**
  - Microsoft Office 2013
  - Google Drive
  - Adobe Photoshop CC 2015 (v2015.0.1)
- **Operating Systems**
  - Ubuntu 15.04

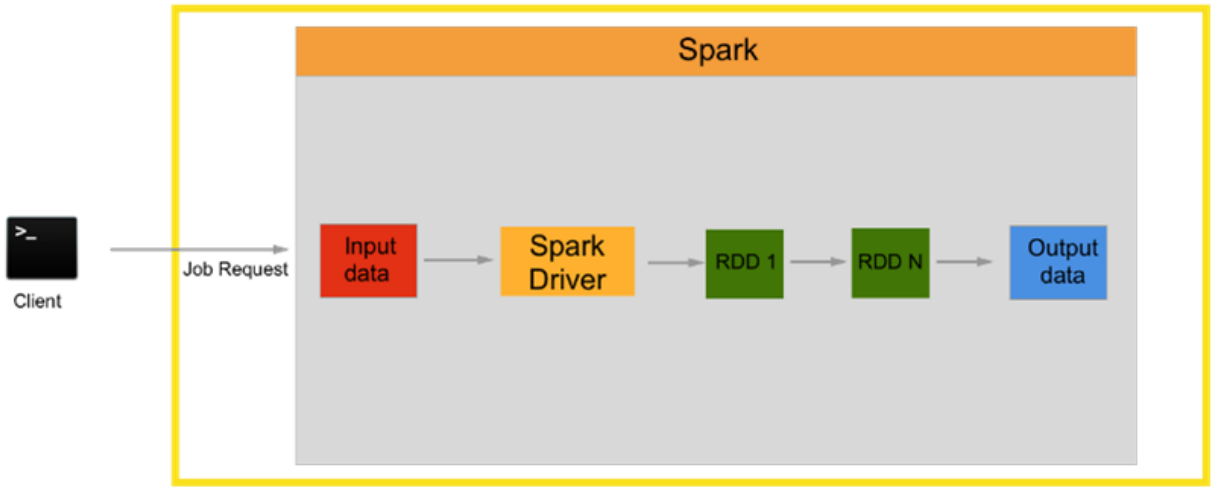
3 System Architecture

3.1 Apache Hadoop Map/Reduce



- The above diagram explains the Hadoop Map/Reduce architecture. Map/Reduce works by breaking the processing into two phases: the map phase and reduce phase.
- Each phase has key-value pairs as input and output, the types of which are chosen by us depending on the type of job.
- We also specify and write code for two functions: map function and reduce function. We write code in Java using Eclipse IDE.
- Map Function: Splits the data into independent and filtered independent chunks which are processed in parallel manner on compute nodes as map tasks.
  i) The Hadoop Framework sorts the output of map tasks
  ii) We use various map sub-functions available in order to have best sorted and filtered data from the map function.
- Reduce Function: Takes the input from map function and we perform the summary operation in this function.
  i) We try to use the shortest and fastest route to perform the necessary operation in the reduce function so that it takes the least amount of time.
  ii) We also specify in the main function to store the output of the reduce function in the HDFS (our database).

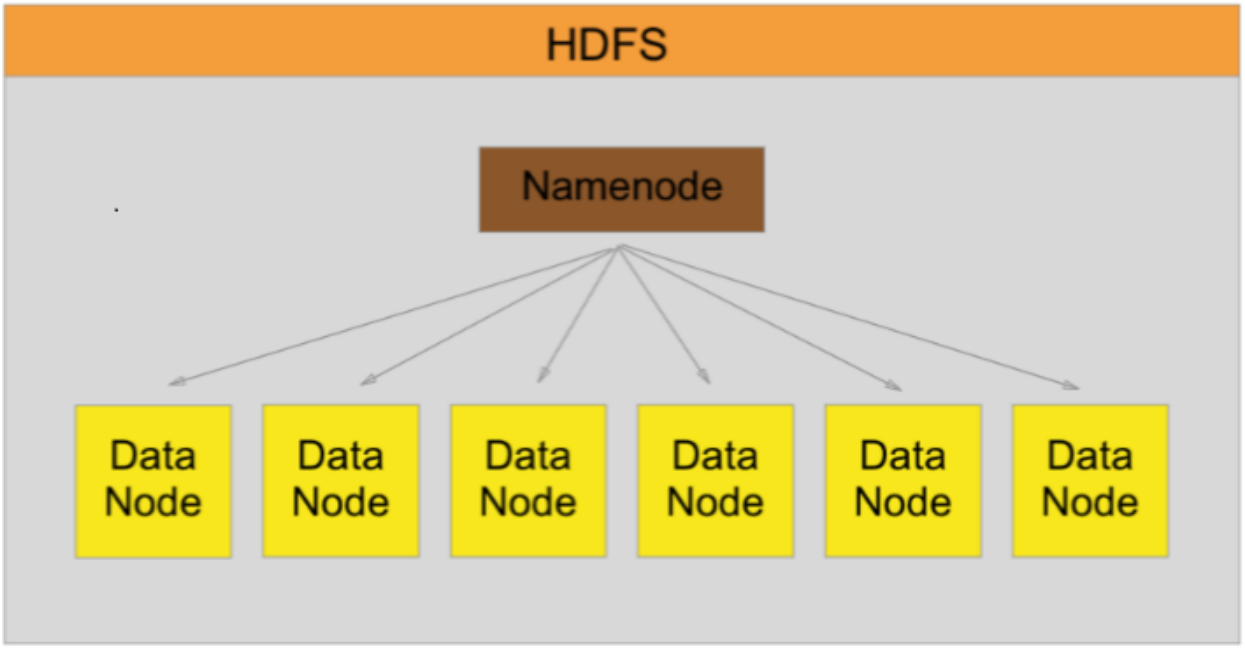
3.2 Apache Spark



- In Apache Spark, we write the Spark command to start a job from the terminal.
- We also write the command to take the input file from HDFS for the specific job.
- All of this information goes to the Spark driver which decides the number of RDDs (Resilient Distributed Datasets) to perform the job.
- Spark uses RDDs, which are way faster than what Hadoop uses and support two types of operations – transformation and action. RDDs do not perform an operation until it is required (Lazy evaluation). RDDs also have persistence, which means that data can be recovered in case of crash of the local memory.
- After the final iteration in the last RDD, the output is stored back into the HDFS which we can access easily from the manager node.

4 Database Designs

4.1 HDFS



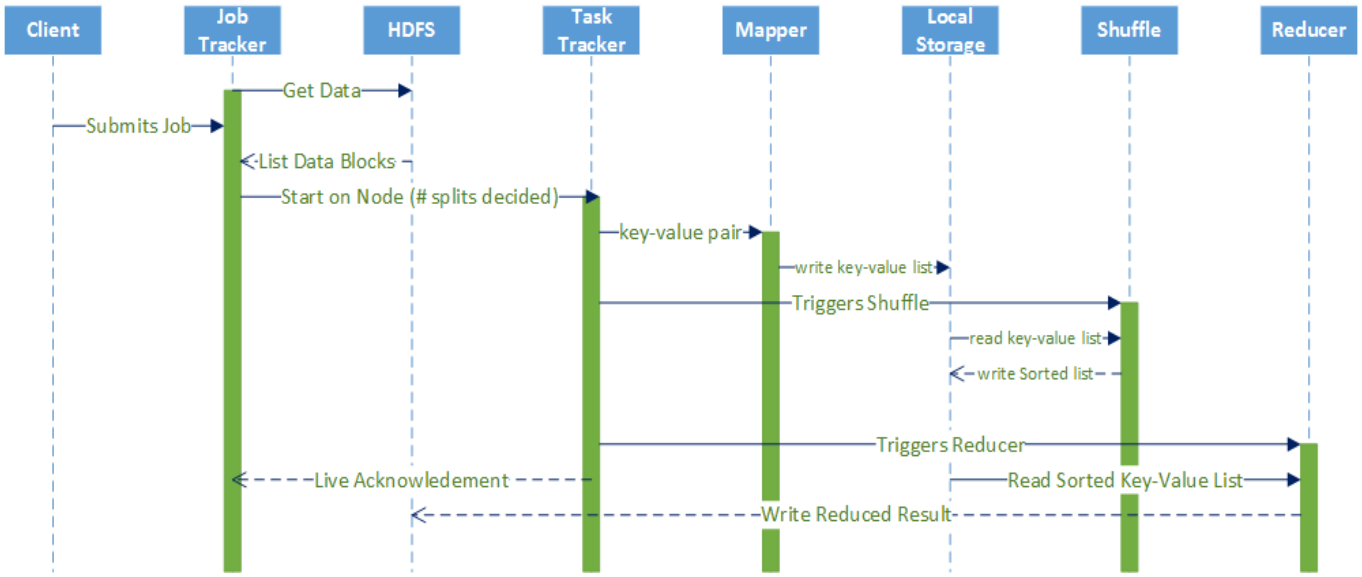
We use Hadoop Distributed Filesystem (HDFS) as our database for both Hadoop and Spark as Spark does not have its own Filesystem.

- HDFS is a filesystem designed for storing very large files with streaming data access patterns. By very large files in this context, we mean the files that are hundreds of megabytes, gigabytes or terabytes in size.
- Each time we upload files on the HDFS we specify the block size we want for those files on HDFS. The Block size depends on the type of user and the type of job that we or the user needs to run. Block size basically means the minimum amount of data that HDFS can read or write.
- The above diagram shows the namenode and datanodes. Our HDFS cluster has two types of nodes operating in a manager-worker pattern: namenode(manager) and a number of datanodes(workers). We can access the jobtracker on the namenode and track our job step by step to analyze which worker is working on what part of the data. Without the namenode, we cannot access our database. The datanodes are the workhorses of the filesystem and are controlled by namenode.
- We use the command line to upload, remove, and create directories and move data on HDFS. We use the Hadoop filesystem commands to access and make changes to our database.

- Spark also uses HDFS as its filesystem, which means that we have the same filesystem for both, Hadoop and Spark, but they are on different machines as we have 2 different machines working as managers of Hadoop and Spark each.

5 UML Diagrams

5.1 Sequence Diagrams



6 Glossary of Terms:

Apache Hadoop: Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets.

Apache Mahout: An Apache software used to produce free implementations of distributed scalable machine learning algorithms that help in clustering and classification of data.

Apache Maven: A build automation tool for projects that uses XML to describe the project the project that is being built and its dependencies on other external modules.

Apache Spark: Apache Spark is an open source cluster computing framework which allows user programs to load data into a cluster's memory and query it repeatedly.

Big Data: Extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions

HDFS: Hadoop Distributed File System is a Java based file system that provides scalable and reliable data storage.

IDE: Integrated Development Environment.

K-means clustering: A way of vector quantization used for cluster analysis in data mining.

Map Reduce: A programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

MLlib: Apache Spark's scalable machine learning library that consists of common learning algorithms and utilities including classification, clustering, filtering etc.

Root Access: Access to install various software and related items on Linux machines.

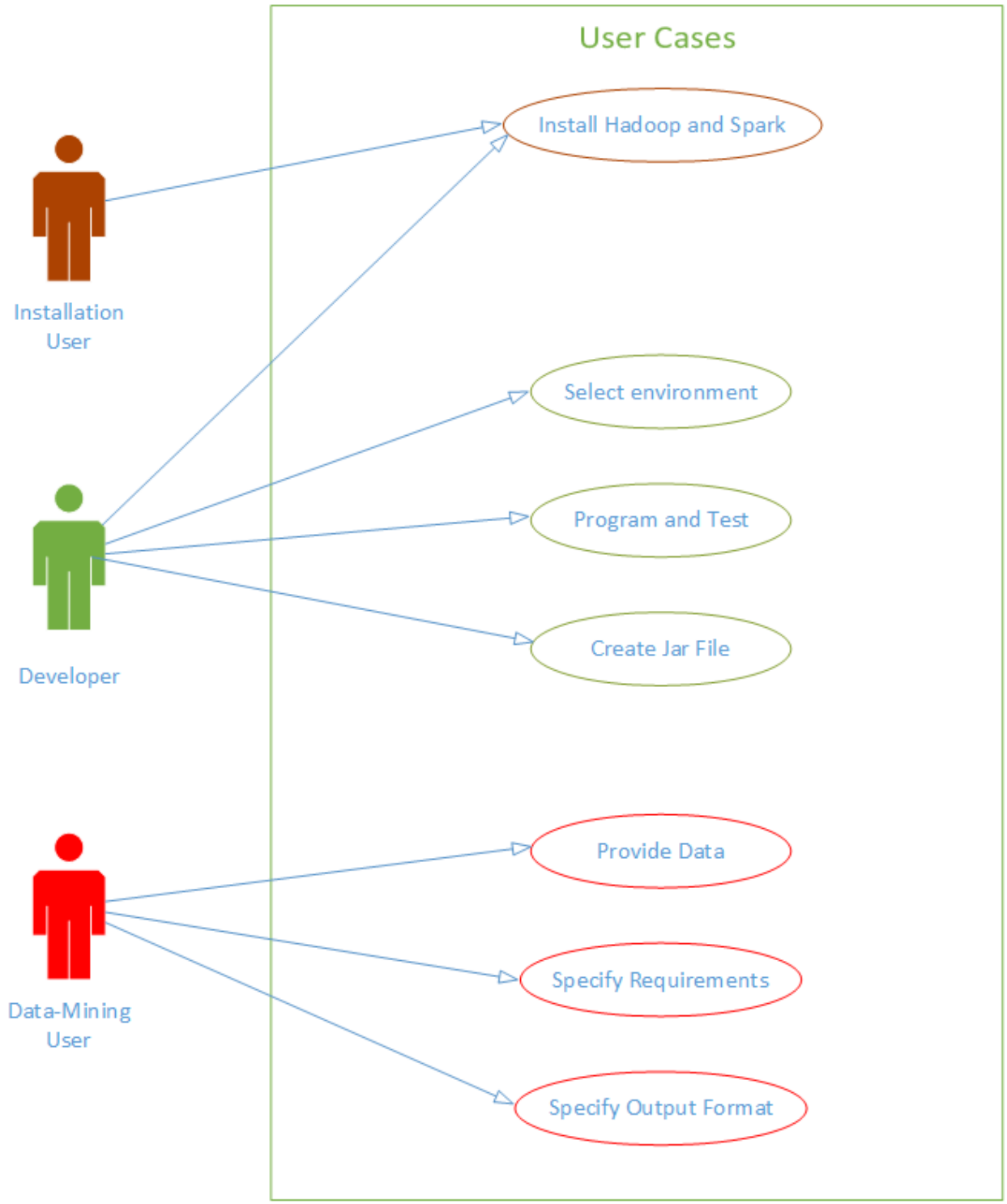
Scala: A programming language for general software applications.

XML: XML stands for Extensible Markup Language that defines the protocol for encoding documents in a format that is both, human and machine-readable.

Apache Hadoop Yarn: YARN (Yet Another Resource Negotiator) is a cluster management technology. It is characterized as a large-scale, distributed operating system for Big Data applications.

Appendices

Appendix A: User Case Model



Install Hadoop and Spark	
Actor	Installation User and Developer
Description	Allows installation user to install Map/Reduce and Spark environments in the machines and the developer to write programs for data-mining purposes.
Goal	To successfully install Hadoop and Spark on the separate systems.
Pre-Conditions	Machines should have Linux environments with appropriate configurations.
Trigger	Failure of processing of Big Data files on Java requires the installation of Hadoop and Spark.
Sequence of Events	Allows the users to run the sample programs on both platforms: IDE and console.

Select Environment	
Actor	Developer
Description	Developer has two choices to process Big Data Files: Hadoop Map/Reduce and Spark. Spark is relatively newer and less common data processing environment.
Goal	The main goal is to process Big Data Files with a great performance in both time and space.
Pre-Conditions	The selection of the environment and the number of nodes to use depends on the size and formatting of the Big Data file. Thus, the data file must be ready in order to reach to a conclusive decision.
Trigger	The selection of the environment and stand-alone/cluster system is triggered by the type of data file to be processed.
Sequence of Events	Once the environment is selected based on the data file, the appropriate program is developed and then executed.

Program and Test	
Actor	Developer
Description	Appropriate program is written to process the data file and generate output results.
Goal	To generate the required output files after processing the data file.
Pre-Conditions	The environment must be selected in order to write the program in either Hadoop or Spark.
Trigger	This event is triggered when the environment is selected based on the data file requirements and an appropriate program is written.
Sequence of Events	A jar file is created once the program is written and is ready for execution.

Create JAR File	
Actor	Developer
Description	The jar file of the JAVA program is created once the program is written.
Goal	The goal is to execute this jar file to process the Big Data file.
Pre-Conditions	The program must be written without any errors and warnings in order to export it as a jar file.
Trigger	This event is triggered once the program is written successfully and the data file is ready to be processed.
Sequence of Events	Once the jar file is executed against the data file, the appropriate output file is generated to analyze the data processed.

Provide Data	
Actor	Data-Mining User
Description	The Data-Mining user provides the data to the installation user and the developer in order to process the data so that he can make relevant decisions based on the output files.
Goal	The goal is to be able to process the data successfully and then analyze it.
Pre-Conditions	The data must be big enough to carry out Map/Reduce and Spark analysis.
Trigger	This event is triggered when a large data set needs to be analyzed and appropriate decisions need to be made based on the results.
Sequence of Events	The data is provided to the installation user and the developer. The data-mining user then specifies his requirements based on which the data shall be processed.

Specify Requirements	
Actor	Data-Mining User
Description	The requirements must be presented to the installation user and the developer in order to make decisions regarding the criteria of data mining, the appropriate environments to use and to decide the proper output format.
Goal	The goal is to be able to process the data and make decisions out of it based on the requirements provided.
Pre-Conditions	The large data set should be selected beforehand.
Trigger	This event is triggered once the big data set is selected and the criteria of data mining are being decided based on the requirements of the data-mining user.
Sequence of Events	Based on the requirements, appropriate output files are generated.

Specify Output Format	
Actor	Data-Mining User
Description	Output files are the most important aspects of the data mining process. Relevant decisions are made based on the generated output files.
Goal	The goal is to be able to analyze the big data based on the output files generated.
Pre-Conditions	Big data set and the specific requirements must be decided prior to generating the output files.
Trigger	This event is triggered by the processing of the jar files against the big data sets on Map/Reduce or Spark.
Sequence of Events	The output files generated are used to analyze the Big Data and make relevant decisions based on them.