
# Project Requirements

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 | 2 November, 2015 |
| 1.1 | Changes in Non-functional requirements, added Apache Maven in a sections 2.6 and 4.3 | 8 November, 2015 |
| 1.2 | User Case Diagrams and User Case Scenarios added. Re-structuring of the document | 17 November, 2015 |
| 1.3 | Spark System Architecture | 15 December, 2015 |
| 2.0 | Updated Benchmarks | 21 January, 2016 |
| 3.0 | Updated Functional and General Requirements, Added Python for Spark | 20 April, 2016 |
| 4.0 | Final Minor Changes | 2 May, 2016 |

# Table of Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to provide all the requirements for project Frog-B-Data. This document contains all the functional and non-functional requirements of the project, along with use-case diagrams. All the requirements shall be delivered to the development and the research team by the project client Dr. Antonio Sanchez or the faculty adviser Dr. Donnell Payne.

## 1.2 Intended Audience

This document is created to provide with the necessary requirements for the development and the research team, TCU faculty advisor for the project, project client. This document acts as a reference for what is necessary to complete the project.

## 1.3 Scope and Objectives

The main objective of Frog-B-Data is to test the performances of data mining algorithms in three environments: Stand-alone Java and two true Big Data environments - Hadoop Map Reduce and the recently introduced Apache Spark - both of which provide a processing model for analyzing Big Data. We carry out different tests to validate the feasibility of preparing Big Data files and processing them in an unstructured manner. We perform four comparison tests: Word Count, Matrix Multiplication, Recommendation using Co-occurrence Matrix or Collaborative filtering algorithms, and K-means clustering. Non-structured data files of sizes ranging from a few Megabytes to 10 Gigabytes are being used for comparative studies. Based on the results, we will build our own recommender system in the preferred framework.

## 1.4 References

**Apache Hadoop Resource:** https://hadoop.apache.org/

**Apache Maven Resource**: https://maven.apache.org/

**Apache Mahout Resource**: http://mahout.apache.org/

**Apache Spark Resource:** http://spark.apache.org/

## 1.5 Overview

This document includes the following seven sections:

**Section 2 – Project Overview:** 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 - External Interface Requirements:** This section provides with an overall view of the external interfaces that the system is required to interact with.

**Section 5 - Functional Requirements:** This section contains the functional requirements of the system architecture.

**Section 6 - Non-functional Requirements:** This section contains the non-functional requirements of the system architecture.

**Section 7 - Glossary of Terms:** This section lists definitions of the terms used in this document.

**Section 8 - Appendices:**

**Appendix A -** User Case Models

**Appendix B -** User Case Scenarios

# 2 Project Overview

## 2.1 Product Perspective

Frog-B-Data is a research project that tests the performances of data mining algorithms in three environments: Stand-alone Java, Hadoop and Spark. This project intends to use various data mining tools and algorithm tests to evaluate the speed and performance of Big Data files ranging from a few kilobytes to hundreds of gigabytes. The main focus of this project is going to be on Hadoop Map/Reduce and recently introduced Apache Spark.

## 2.2 Product Functions

Our project will document all the results we obtain during the research where all the performance tests, processing time of the Big Data files in each of the three environments will be properly recorded and will be used for further research. The user can input a large data file and can run it on both Hadoop and Spark and can see the difference in the time taken to analyze and process data. The six tests will be run in a clustered system as well as individual machine structure, to differentiate between the performances in both types of system environments.

## 2.3 User Characteristics

Frog-B-Data is intended for the future Big Data tools' users to get the proper installation guides, Developer's manual, and research results. This project is a gold mine for the users who want to have a good grasp on the concepts of Hadoop File Systems and Apache Spark, both using Linux command line, and Eclipse running with Apache Maven and Apache Mahout.

## 2.4 Constraints

### 2.4.1 Time Constraints:
End of school year limits research and testing time.

### 2.4.2 Data Constraints:
Limited availability of 'real' Big Data files that can be processed.

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

### 2.4.4 Operating System Constraints:
Need Linux OS to get the best performance results.

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

## 2.5 Operating Environment

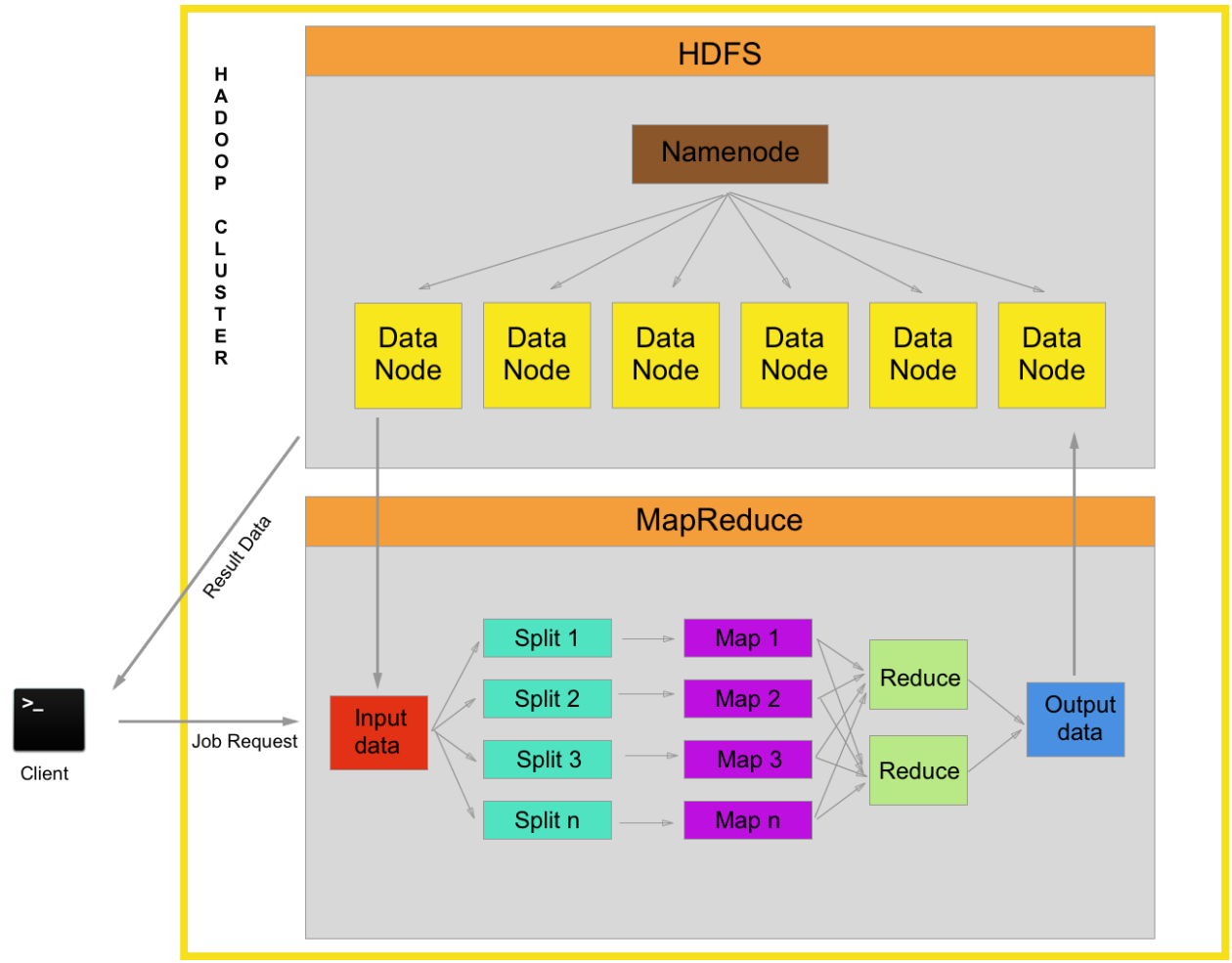
Frog-B-Data can be used on any Linux environment and Eclipse Mars IDE with the help of the User Guide. It can be used on either independent machines or in a manager/worker system structure, but the performance will be much better in a clustered framework.

## 2.6 Assumptions and Dependencies

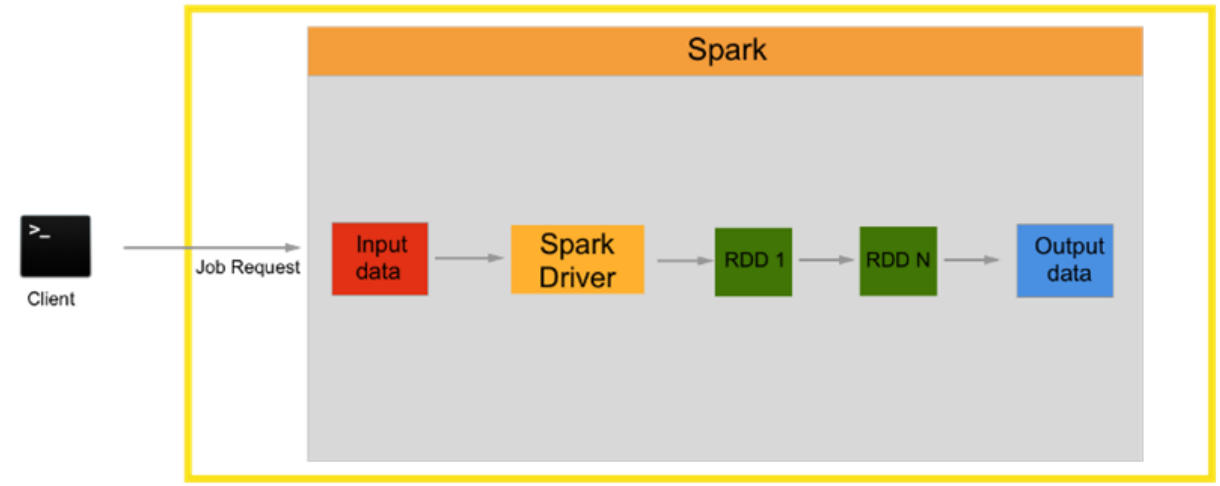
- We assume that the user will have Linux operating system, preferably Ubuntu 15.04 or above, with Hadoop and Spark installed and basic knowledge of Java programming language along with Eclipse IDE.
- We assume that all the Java programming will be done using Apache Maven in both, Hadoop and Spark. Apache 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.
- We assume that the user will have basic knowledge of Python programming if he/she wants to simulate Frog-B-Data's Spark recommender as the recommendation system in Spark is built using Python.

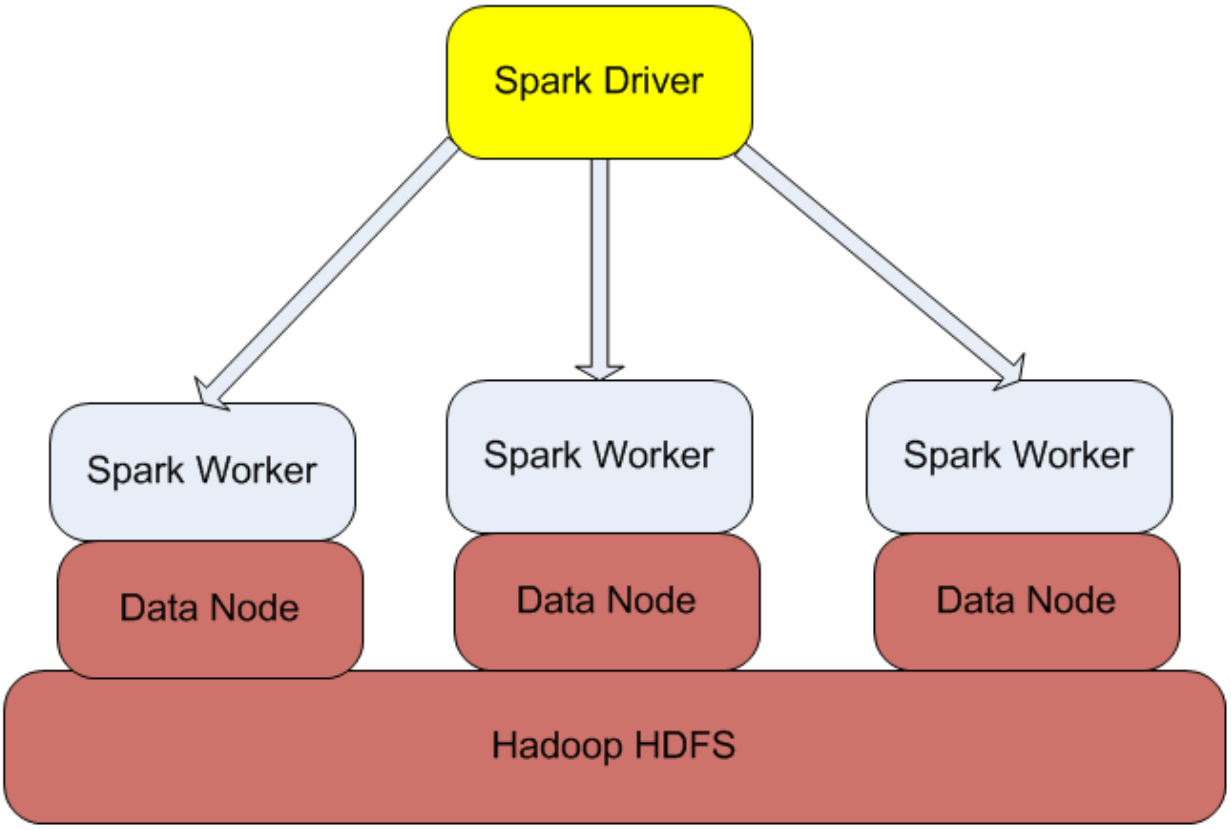
# 3 System Architecture

## 3.1 Hadoop Map/Reduce



### 3.2 Spark


Spark
Client
Job Request
Input data
Spark Driver
RDD 1
RDD N
Output data


Spark Driver
Spark Worker
Spark Worker
Spark Worker
Data Node
Data Node
Data Node
Hadoop HDFS

# 4 External Interface Requirements

## 4.1 User Interfaces

The user shall interface with the project by the Java/Python source code on the Eclipse IDE and the output text file generated by Hadoop and Spark after processing Big Data file, which will have the results of the data analysis. These results can be accessed via Hadoop Distributed File system (HDFS) Web Interface.

## 4.2 Hardware Interfaces

The project can be accessed on any computer with Linux operating system.

## 4.3 Software Interfaces

- Eclipse Mars: Java IDE with Maven and Mahout Plugins, which will be used with Hadoop and Spark
- PyDev: Python plugin to write Python code on Eclipse Mars
- Hadoop: includes HDFS and MapReduce
- Spark: includes MLlib which is Spark's scalable machine learning library.

## 4.4 Communication Interfaces

Eclipse Mars IDE serves as a communication tool between Hadoop/Spark and the user. This can also be achieved using Linux terminal.

# 5 Functional Requirements

## 5.1 General Requirements

- This project shall set up Hadoop Map/Reduce and Spark frameworks on a cluster of nodes.
- The project shall provide all the information and guidelines required to set up both of these frameworks.
- The project shall implement four test benchmarks: Word Count, Matrix Multiplication, Recommendation Systems and K-Means Clustering on each environment.
- The system shall generate, for each execution of the appropriate code, a unique output file displaying all the data analysis results.
- Both Hadoop and Spark recommenders shall provide the user with 'reasonably good' recommendations.

## 5.2 Technical Requirements

### 5.2.1 Hadoop

- The Hadoop System shall be composed of two basic layers: a data storage layer called Hadoop Distributed File System (HDFS), the database, and a data processing layer called Hadoop Map/Reduce Framework.
    - The Map/Reduce Framework shall help the Hadoop programs to perform two separate and distinct tasks. The first is the Map task which shall take a set of data and convert into another set of data, where individual elements shall be broken down into tuples (key/value pairs). The reduce task takes the output from a map as its input and combines those data tuples into a smaller set of tuples. The Map task performs the filtering and sorting of data and the Reduce task performs the summary operation. It works like a divide and conquer technique.
- The user shall be able to work with Apache Mahout whose core algorithms of clustering, classification and filtering are implemented on top of Hadoop Map/Reduce and provide a basic framework on which the user can build upon.
- The user shall be able to work with Apache Maven which is a Project Management tool that helps in building a project and defines all of its dependencies on other external modules.

### 5.2.2 Spark

The Spark System shall be composed of the data layer, HDFS as the database. In contrast to Hadoop's two-stage disk-based Map/Reduce paradigm, Spark shall have a multi-stage in-memory primitives.

Spark shall require a cluster manager and a distributed storage system.

- For cluster management, Spark shall support standalone Hadoop Yarn as a native Spark cluster.
- For distributed storage, Spark shall interface with Hadoop Distributed File System (HDFS)

## 5.3 Data Storage Requirements

- Frog-B-Data uses Hadoop Distributed File System (HDFS) as its database. HDFS is not a "normal" relational database, where data is found and analyzed using queries based on the industry- standard Structured Query Language (SQL). HDFS is like a data warehousing system from where the user shall be able to put and get data without any queries. Rather the user shall use simple Hadoop Filesystem commands to access data.
- The user shall be able to input, delete, and update any Big Data file on the HDFS.
- The Eclipse IDE and the Linux terminal shall have the access to all the files in HDFS to process data.

## 5.4 Benchmarks

Stand-alone Java, Hadoop and Spark shall be tested thoroughly using the following tests:

### 5.4.1 Word Frequency Count:

We shall use a large text file (.txt) to perform a word count on the given file. The output shall be a text file with the words sorted alphabetically, and indicating the count of each word. This test shall be repeated for different file sizes, and the performance shall be noted for each of them. We shall write and test the word count program in an IDE (Eclipse) for all the following environments (for both, single-node and cluster):

- **Java:** The program shall be written in Java programming language.

- **Hadoop:** The program shall be written in Java programming language with Map/Reduce methods.

- **Spark:** The program shall be written in Java programming language with Spark methods. In case of Spark, the output shall be a series of separate text files instead of just a single file, different from the above environments.

### 5.4.2 Large Matrix Multiplication

We shall write and execute matrix multiplication programs in all stated environments that would take two matrix text files (Matrix A and Matrix B) as input and perform a matrix multiplication of the given files. The output shall be a text file with the output of the matrix multiplication (A*B). There shall be 2 types of matrix multiplication:

- One-Step Matrix Multiplication: One Map/Reduce step
- Two-Step Matrix Multiplication: Two Map/Reduce steps

This test shall be repeated for different matrix sizes:

| | | |
|---|---|---|
| • Matrix A: 2x5 | Matrix B: 5x3 | Output Matrix: 2x3 |
| • Matrix A: 10x10 | Matrix B: 10x10 | Output Matrix: 10x10 |
| • Matrix A: 50x50 | Matrix B: 50x50 | Output Matrix: 50x50 |
| • Matrix A: 100x200 | Matrix B: 200x100 | Output Matrix: 100x100 |
| • Matrix A: 1000x800 | Matrix B: 800x1000 | Output Matrix: 1000x1000 |
| • Matrix A: 2000x3000 | Matrix B: 3000x2000 | Output Matrix: 2000x2000 |
| • Matrix A: 5000x6000 | Matrix B: 6000x5000 | Output Matrix: 5000x5000 |
| • Matrix A: 10000x12000 | Matrix B: 12000x12000 | Output Matrix: 10000x12000 |

The performance shall be noted for each matrix file size. These programs shall be tested in the following environments (for both, single node and cluster):

- **Java:** The program shall be written in Java programming language.
- **Hadoop:** The program shall be written in Java programming language with Map/Reduce methods.
- **Spark:** The program shall be written in Java programming language with Spark methods.

### 5.4.3 Recommendation System Test using Mahout and MLlib on Hadoop and Spark (Only on cluster)

- **Hadoop:** Movie Recommendation system using the Co-occurrence algorithm.
- **Spark:** Movie Recommendation system using Collaborative Filtering.

### 5.4.4 K-means clustering on Hadoop and Spark

The algorithm for K-Means clustering is same for both Hadoop and Spark.

# 6 Project Non-Functional Requirements

## 6.1 Product Requirements

- **Reliability:** The reliability of processing the data files depends on the file size. If the file size is too large (in Terabytes), the system may behave unexpectedly.
- **Response Time:** The response time shall vary with the file size to be processed. It shall also depend on the system environment the user shall use.
- **Efficiency:** The user shall give both Hadoop and Spark machines maximum processing power possible to get the best results.

The performance may be different if the user uses a different IDE or a different system configuration. It may also differ if the user uses a different version of Hadoop or Spark.

## 6.2 Organizational requirements

- **Environmental requirements:** For Hadoop, we shall write the Java code in Eclipse IDE with Apache Mahout and Apache Maven plugins and run the code using the terminal. For Spark, we shall write the initial test codes (word count and matrix multiply) in Java and the recommender system code in Python using the PyDev plugin for Eclipse.
- **Development Requirements:** We shall be using Java and Python programming languages to write code for both of our systems.

## 6.3 External Requirements

- The user shall only use legally obtained data.
- The user shall not share the data with unauthorized people for privacy and security reasons.

# 7 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

**Collaborative Filtering:** Method to predict the interests of a user based on interests of other users.

**Co-occurrence algorithm:** Counting the number of times each pair of items occur together and then predicting the interests of a user based on the user's previous interests and most co-occurred items.

**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.

**PyDev**: A Python IDE for Eclipse which is used in Python Development.

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

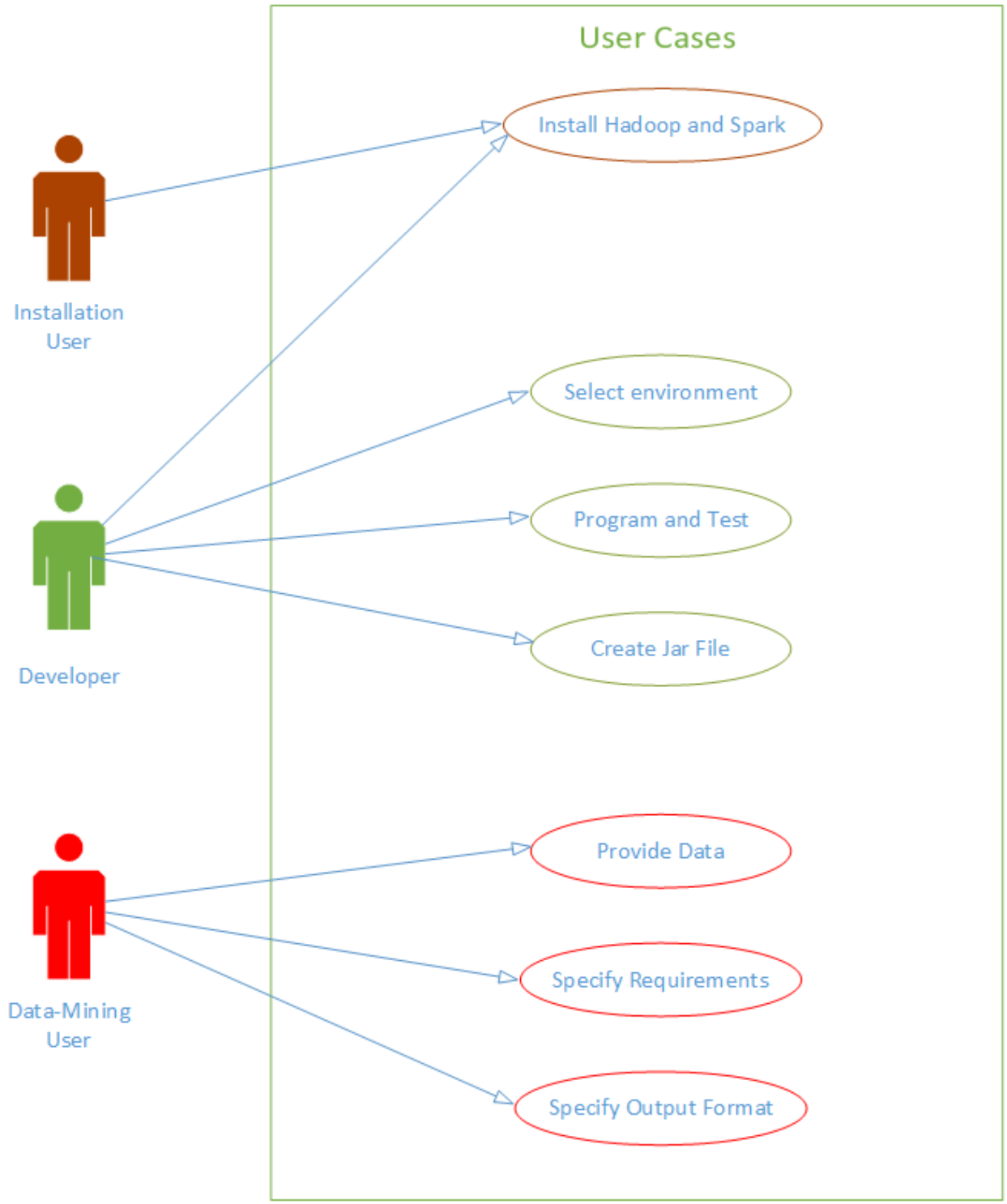
**Scala:** A programming language for general software applications.

**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



## Appendix B: User Case Scenarios

| Install Hadoop and Spark | |
|---|---|
| Actor | Installation User and Developer |
| Description | Allows installation user and the developer to install Map/Reduce and Spark environments in the machines. |
| 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 large 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 large data files: Hadoop Map/Reduce and Spark. Spark is relatively newer, and its use has been growing rapidly. |
| Goal | The main goal is to process large 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 large 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 large data file to be processed, and the requirements given by the data-mining user. |
| Sequence of Events | Once the environment is selected based on the large data file, the appropriate program is developed and tested in the IDE. |

| Program and Test | |
|---|---|
| Actor | Developer |
| Description | Appropriate program is written to process the large data file and generate output results. |
| Goal | To generate the required output files after processing the large data file. |
| Pre-Conditions | The environment must be selected in order to write the program using either Map/Reduce or Spark methods. |
| Trigger | This event is triggered when the environment is selected based on the large data file requirements. |
| Sequence of Events | A jar file is created once the program is written and tested. |

| 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 large 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 large 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 large data file and make relevant decisions based on them. |