# PoC - Big Data Health Analytics – Architectural Design of Platform

# Introduction

It is the objective of this document to put forward an enterprise capable architecture for the Big Data Health Analytics platform utilizing the technology stack given in PoC Deliverable #1 – Technology Selection.

This document has been written to support Task #2, Deliverable #2 of the Big Data Health Analytics, proof of concept(PoC) project, which are as follows:

**Task #2:** Design the architecture for the Big-Data processing platform.

**Deliverable #2**: The specific deliverables are as follows:

1. *A comprehensive architectural design of the Big-Data platform.*
2. *Functional schematic of the design.*
3. *Logical schematic of the design.*
4. *Implementation schematic of the design.*

Excerpts from the document PoC Deliverable #1 – Technology Selection, are given for the purposes of establishing a narrative.

# Structure

This document addresses the deliverable described in the introduction in the following manner

|  |  |  |
| --- | --- | --- |
| Section | Synopsis | Deliverable Addressed |
| [Section 4: Comprehensive Architectural Design](#_Comprehensive_Architectural_Design) | Lists the technology stack used in the PoC. Then presents the evolving landscape of the Apache Hadoop Big Data ecosystem, which we have selected a subset for the PoC. To keep up with an evolving ecosystem the principle of loosely-couple-strongly-aligned (LCSA) design principle is put forward and its direct impact on a robust and maintainable system architecture. Presents Spring Architecture Framework which embodies LCSA and its direct utilization to system integrate the components in the healthcare analytics technology stack. | #2.1 |
| [Section 5: Functional and Logical Architecture](#_functional_architecture) | This section addresses (2) and (3) together. (2) is addressed by defining a common scenario within health care analytics and putting forward a strategy to tackle it. (3) provides a set of UML diagrams that show the enacting of that strategy with marked system boundaries taken from the technology stack. | #2.2, #2.3 |
| [Section 6: Implementation](#_implementation_schematic) | This section addresses (4). The section discusses: Data Sets, High Level Process, Deployment, Build Environment, Software Implementation and Test Criteria | #2.4 |
| [Section 7: Conclusion](#_Conclusion) | Summing up the document |  |

# Comprehensive Architectural Design

## Technology Stack

In PoC Deliverable #1 – Technology selection, we proposed a technology stack that would meet the demands of the current and future PoC’s. We also mentioned that this stack was chosen to enable eventual commercialization.

As mentioned the core data processing element of our technology stack centers around Apache Spark. The reasons for Apache Spark, as stated in PoC Deliverable #1 – Technology selection, are as follows.

* Lightning Fast Processing
* Support for Sophisticated Analytics
* Real Time Stream Processing
* Integration with both NoSQL and RDBMS
* Ability to Integrate with Hadoop
* Active and Expanding Community

## High Level Component Diagram

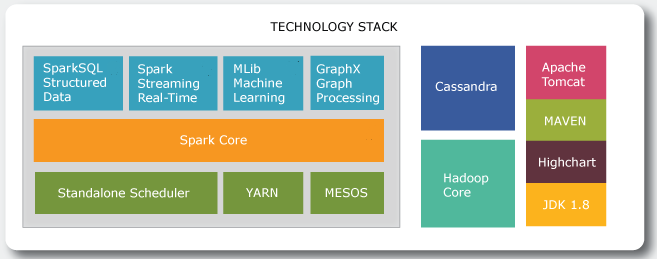


Figure: 1.0 Technology Stack - Ref: Apurba Technology – Original Content

The diagram above describes the technology stack that will form the backbone of the biomedical big data analytics platform.

|  |  |  |
| --- | --- | --- |
| **Spark Framework** | **Component** | **Description** |
| Spark Core | Contains basic Spark functionality. Sparks fundamental programming abstraction, Resilient Distributed Data Set (RDD) represents a collection of items spread across parallel computing nodes. Spark provided an API for creating and managing RDDs. This API also takes care of parallel processing and management of RDDs. |
| SparkSQL Structured Data | Package for handling structured data. Querying via SQL as well as Apache Hive. SparkSQL supports interface with RDBMS, NoSQL Databases, HIVE Tables, Parquet and JSON |
| Spark Streaming Real-Time | Supports streaming of live data. Spark makes use of RDD creation, processing and management subsystem |
| MLib Machine Learning | Common library containing machine learning algorithms including classification, regression, clustering etc. |
| GraphX Graph Processing | Framework for manipulating graphs |
| Standalone Scheduler | Cluster Management Services |
| YARN |
| MESOS |

|  |  |  |
| --- | --- | --- |
| Supporting Technology Stack | **Component** | **Description** |
| Hadoop Core | Hadoops file system management services. Called directly from within Spark Core |
| Cassandra | Industry leading NoSQL database, providing these benefits: High Performance, Elastic Scalability, Open-Source, High Availability and Fault Tolerance, Column Orientated, Tunable Consistency, Schema Free |
| JDK 1.8 | Java SDK – Programming Language |
| MAVEN | Utilized to manage code dependencies |
| Apache Tomcat | Web Application Server |
| Highcharts | Java Script based charting library for data visualization |

## Evolving Ecosystem

Apache Spark is part of a growing ecosystem of technologies built around Hadoop. The diagram below is taken from [Data Science and Analytics Outsourcing – Vendors, Models, Steps by Ravi Kalakota](https://practicalanalytics.co/2015/05/28/data-science-and-analytics-outsourcing-vendors-models-steps/).

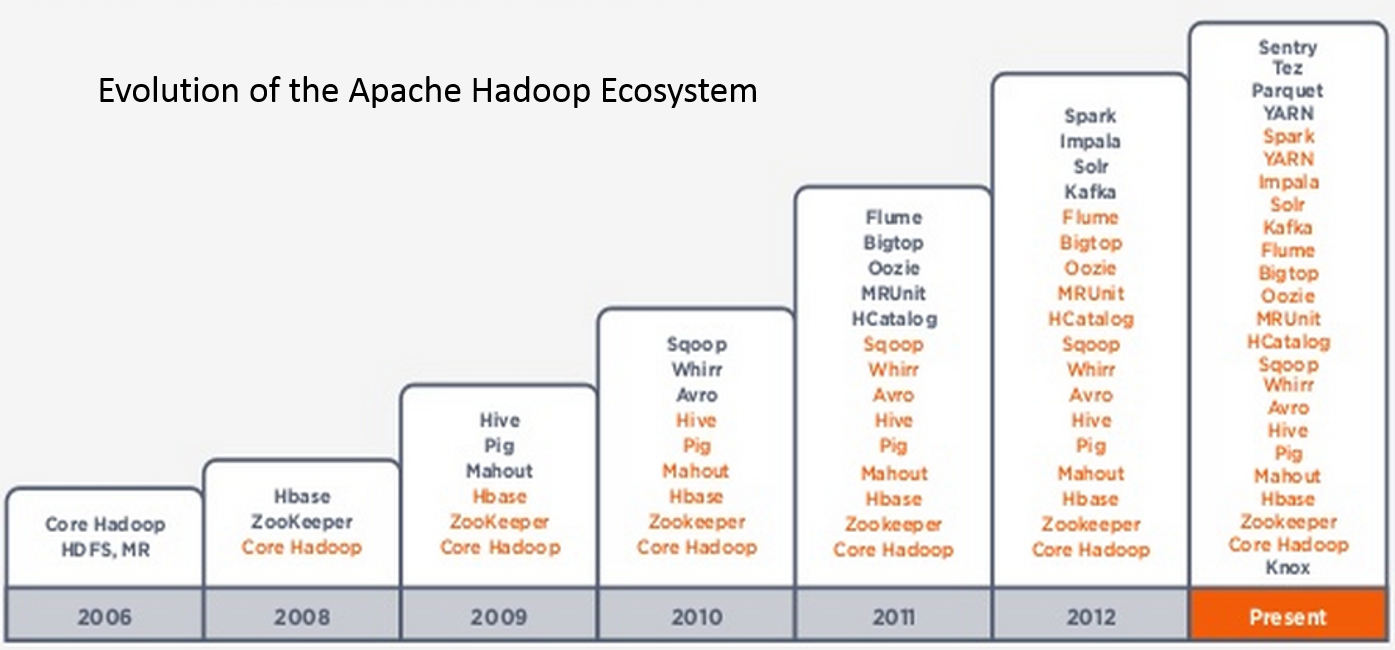


Figure 2.0: Hadoop Ecosystem History – [Ref: Practical Analytics](https://practicalanalytics.co/2015/06/02/the-maturing-nosql-ecoystem-a-c-level-guide/)

It is entirely possible that future PoC will require new technologies not listed in our current stack. To facilitate seamless introduction, we need to ensure the core design principle of loosely coupled – strongly aligned in our architectural framework

## Working with The Spring framework

The design principle of being loosely-coupled – strongly-aligned(LCSA) has been adopted and utilized successfully in other system architectures. There are literally 100’s of scholarly articles in this subject.

The Spring Software framework built from the ground up using Java expresses the LCSA principle. As it’s an extension of Java JDK1.8+ for the purposes of the technology stack, it’s not considered as a separate entity to Java.

The Spring Framework provides this key benefits which will allow this Healthcare Analytics platform to keep pace with new developments in the Hadoop ecosystem:

* Fault Tolerance
* Test Driven Development
* Separation of layers
* Presentation
* Business Logic
* Data Access
* Efficient Deployment
* Code Reuse

### Spring Framework Design Principles in Focus

The Spring framework is one of the most popular, tried and tested J2EE frameworks around today. Using Spring it’s possible to develop an enterprise application that utilizes **model-view-controller (MVC)** design pattern. The MVC pattern allows us to have **separation of layers** between user interface/presentation, business logic and data access.

In terms of the health analytics application this means the front end written in HTML5 can be decoupled from the business logic tier (Apache Spark – enabled components) and data access (Cassandra, HDFS).

The separation of layers is encompassing part of the LCSA principle. However, using Spring allows us to go further by employing these two powerful features offered by this framework:

* **Dependency Injection**
  + Enables Reuse of Classes
  + Facilitates Unit Testing of Classes
  + Utilizes Inversion of Control (IoC) design patter
* **Aspect Orientated Design**
  + Allows handling of functionality that spans multiple points within the Health Care Analytics platform. For example; logging and security are two functional ‘Aspects’ that are required across the application.
  + AOD allows decoupling of these Aspects from the classes that make use of them

### Spring Framework Architecture

The Spring Framework is a set of extensions to the Java programming language that allows development of enterprise applications. Spring is modular in nature and provides the advantage of using only the modules that you want.

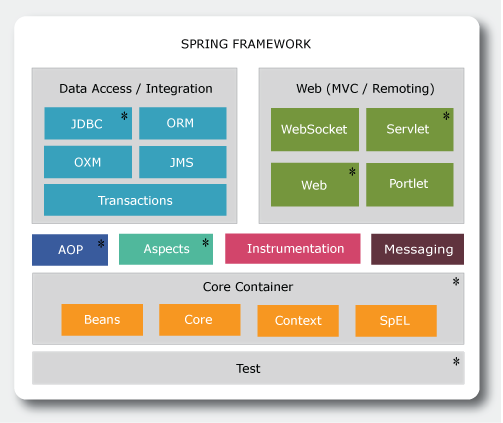


Figure 3.0 Elements of Spring Infrastructure envisioned for use in this PoC based on content from Ref: [Spring Tutorials](http://www.tutorialspoint.com/spring/spring_architecture.htm)

A comprehensive description of the Spring Architecture can be found [here](http://www.tutorialspoint.com/spring/spring_architecture.htm). However, for the purposes of putting forward an architecture for this Healthcare Analytics PoC, the elements of the Spring framework will be discussed in context of the PoC. For this PoC, only the modules marked with \* will be used.

#### Core Container

Springs Core Container consists of the functional elements entitled, Beans, Core, Context and SpEL. These are to be used in the PoC in the following manner.

* Core module encapsulates support for IoC and Dependency Injection design patterns. These design patterns will be used heavily to create an eco-system of interchangeable classes.
* Bean Factory Pattern will be used to generate classes for establishing connection to Apache Spark, Cassandra and other services
* Context builds on top of Core and Bean modules to provide access to any component in the application based on what is called the Application Context. Using this we can have multiple interfaces within our Enterprise Application deployment, each with access to a specific subsystem of classes based on role
* SpEL – allows us to map relationships between objects. This is a very useful feature in situations where you are dealing with an ecosystem of classes

#### Data Access/Integration

Spring Data Access/Integration provides support for JDBC (database connection), ORM (Object Relationship Mapping), OXM (Object – XML representation), JMS (Java Messaging Services) and Transactional support. For this PoC we will be using JDBC only. The advantage of Spring is that this does not exclude the utilization of the other components in this layer if needed at a future date.

#### Web Layer

Spring Web Layer will be utilized to develop the user interfaces for the Health Analytics platform. This layer consists of the following elements Web, Web-MVC, Web-Socket, and Web-Portlet. Within the context of this ecosystem:

* Web – will provide support for file-upload and utilization of the IoC Container
* Web-MVC. We will use this to develop a web application that will leverage the model-view-controller design pattern
* Web-Socket. Provide communication between the client layer and to the services layer (i.e business logic – leveraging Apache Spark and Cassandra)
* Web Portlets – won’t be used for this PoC. It’s for development of portlets that leverage MVC design pattern

#### AOP and Aspects

AOP (Aspect Orientated Programming) framework 7 allow decoupling of functionality that transcend any specific component. For example, utilizing a method interceptor for logging would mean developing just one logging implementation that could then be reused across the entire platform.

Aspects component facilitate the use of AspectJ, an aspect orientated programming extension to Java.

## Enterprise Architecture Diagram

Development of an Enterprise Architecture (EA) is to take into account these objectives:

* Realize Technology Stack to Software Architecture
* Facilitate Technology Evolution
* Robust
* Maintainable
* Performant

The EA for the PoC is to be expressed as a three tier ecosystem:

* Presentation Services Layer
* Business Process Services Layer
* Data Access Services Layer

The elements of the technology stack will fall into one of these three service layers.

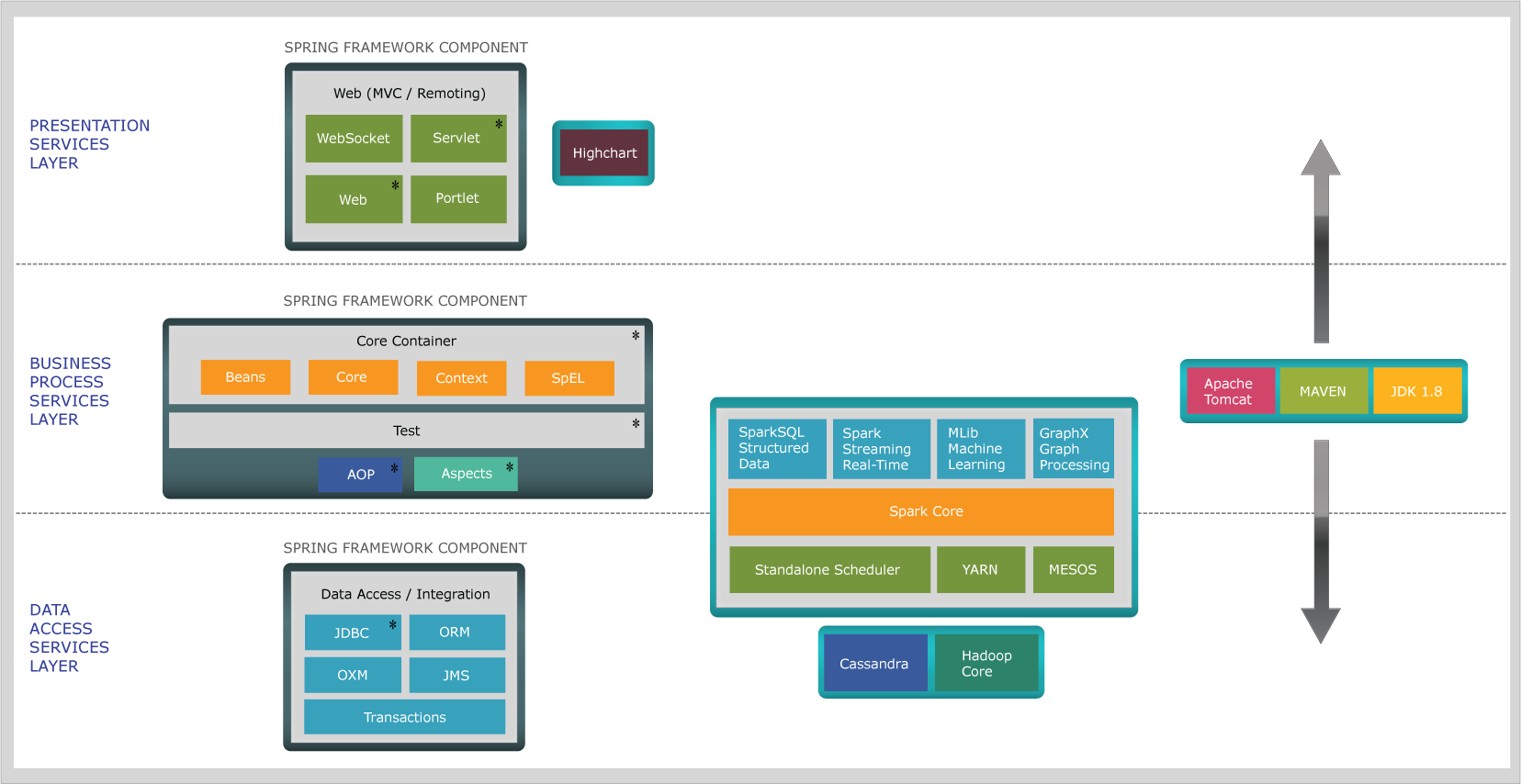


Figure 4.0: Enterprise Architecture Utilizing the PoC Technology Stack – Ref: Apurba Original Content

# functional architecture

## introduction

This section addresses the, “Functional Schematic of the Design” – as stated in the PoC Deliverables and Timeline. The aim of the functional architecture is to:

* Direct development of the core Big Data Healthcare Analytics Infrastructure.
* Provide performance metrics and give insights into scalability and tuning of the platform
* Create reusable components that can form the basis a marketable product.
* Utilize open source medical data sets

## health care analysis scenario

This PoC will focus on analyzing unstructured (free form) and structured (schema based) medical text. The text data will be provided from two open source medical data sources:

* <http://www.healthdata.gov/> - structured data source
* <http://www.guideline.gov/> - unstructured data source

The structured and unstructured features within these medical data sets are in common with other medical data sets that will be utilized in clinical practice

* Patient Notes – complied by the Doctor during consultation
* Medical Notes created via collaborative effort
* Medical Research Notes

Performing analysis on structured and unstructured text is nothing new. This issue has created an industry worth billions of dollars in storing and retrieving this content. This particular sector known as enterprise content management(ECM) takes care of managing this large reservoir of information.

However, it has been observed that ECM based solutions haven’t provided any credible solution to leverage this information for decision making purposes.

The reason is that ECM solutions have been designed to just store the content. At the dawn of ECM, the challenge was the following:

* Storage
* Retrieval
* Records Management
* Security

None of these challenges involved data mining as being able to understand relationships within the content was considered too difficult.

Attempts were made to index the content, however indexing procedures were slow and cumbersome. The reason being was that no thought was given to any kind of scalable computation framework to analyze the content.

Here enters the use of Big Data Analytics applied to silos of amassed text based content!

## Data, Functionality, Benefits

In this PoC we will direct the development of our technology stack to tackle:

* <http://www.healthdata.gov/> - structured data source
* <http://www.guideline.gov/> - unstructured data source

Develop the following functionalities:

* Systematic analysis of large data sets comprised of unstructured text (common problem in document management systems be they of medical focus or not).
* Identification of medical keywords within text
* Distribution and relationship of medical keywords

Realize these benefits:

* Develop the technology infrastructure
  + Assess scalability and performance
* Prepare for future PoC’s
  + Identify patterns.
  + Medical terms that seem to always crop up together
  + Create the basis to identify decision patterns by doctors within particular scenarios in medical practice
  + Identify any methodologies in a particular area that could benefit another

## use case diagrams

This section describes the use cases supported by this PoC. All Use Case diagrams are Apurba Original Content.

The Use Cases revolve around reading from a folder structure containing a large set of documents. This folder structure could scale up from a few GBs on a user’s drive; all the way to a storage area network containing TB’s of data.

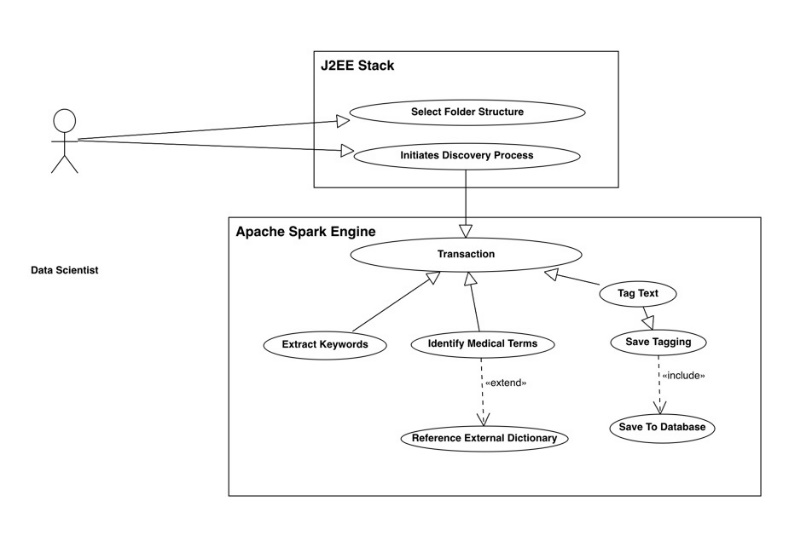


Figure 5.0: Scenario for discovering keyword distributions in a folder structure of documents.

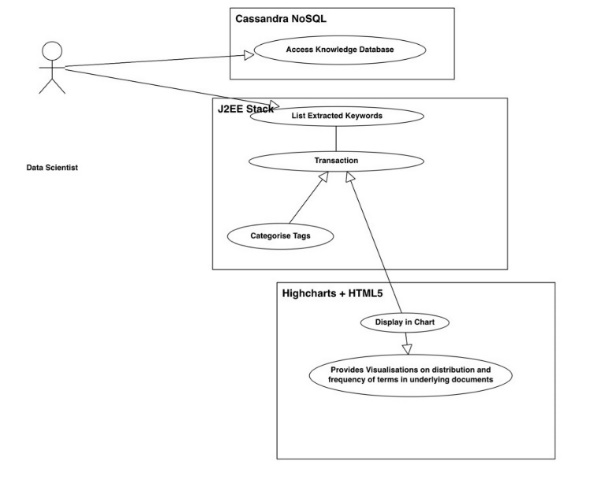


Figure 6.0: Scenario for visualization distribution of keywords within a large repository of documents or data lake.

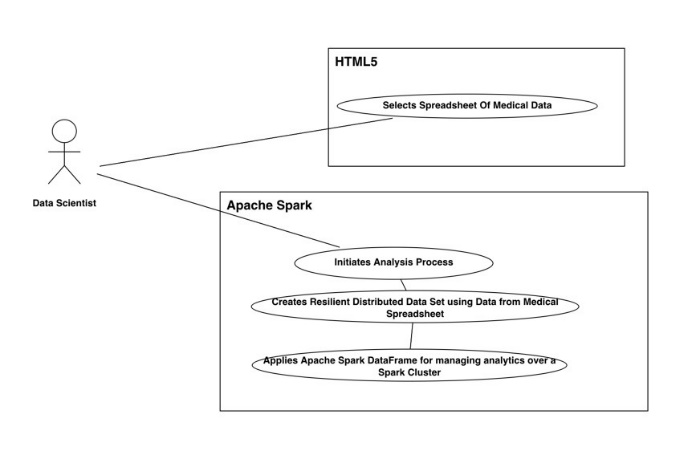


Figure 7.0: Analysis of spreadsheet data via computation framework to demonstrate platform scalability

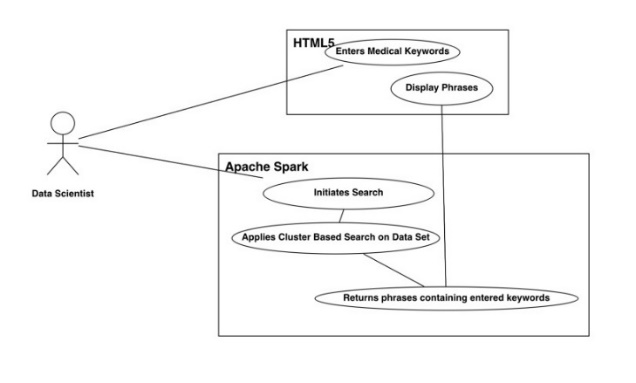


Figure 8.0: Retrieving phrases based on medical keyword search.

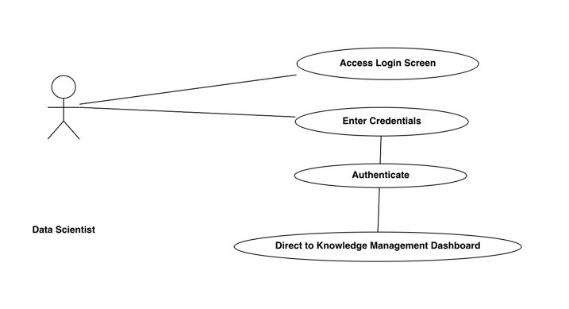


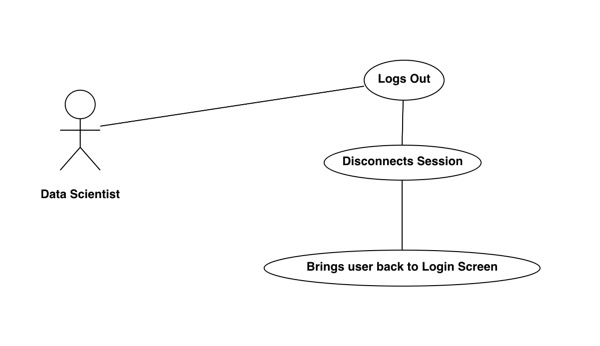
Figure 9.0: Login Use Case

Figure 10.0: Logout Use Case

# implementation schematic

## Introduction

In this section we will be discussing the practical implementation of the technology stack described previously. The section is organized as follows

* Data Sets
  + Where we discuss the medical data sets utilized in this PoC
* High Level Process
  + Synopsis of how the PoC’s computational framework is being utilized for data processing
* Deployment
  + Here we will talk about how the source code is managed and deployed
* Build Environment
  + A schematic diagram of how the software stack is installed.
* Software Implementation
  + Describes at Java Package Level and build script
* Test Criteria

## Data Sets

The data set is key for testing the performance of the enterprise architecture and technology stack. The constraint in using medical data sets from actual clinical practice is patient anonymity and security.

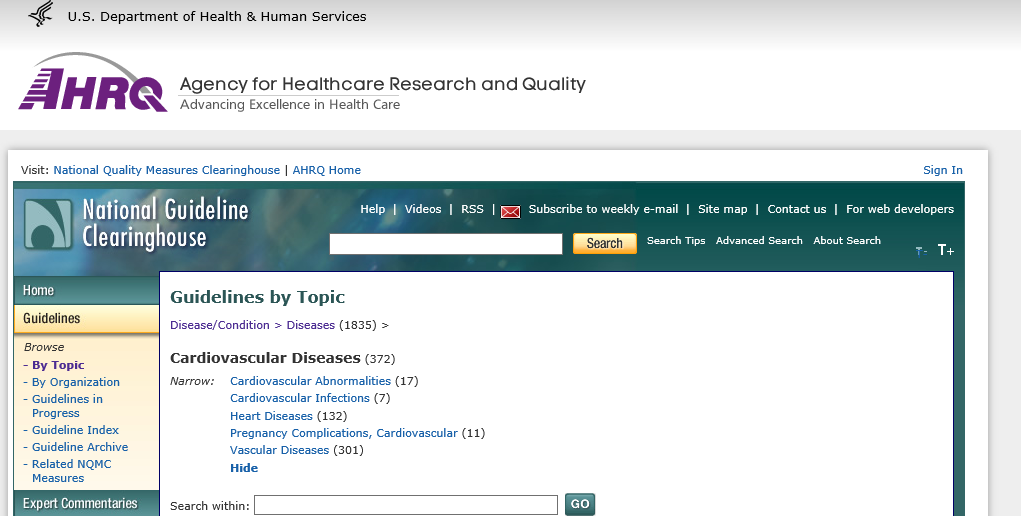
As a result of this constraint, such data sets were unavailable. To ensure that the data sets in this PoC were at least relevant, publically available datasets created for use in clinical practice were used: 

Figure 11.0: Ref: Medical Guidelines used as an example of unstructured (non-schema)data - <http://www.guideline.gov>

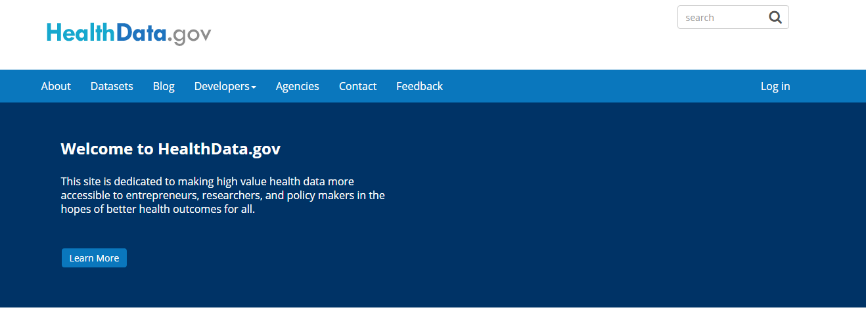


Figure 12.0: Ref: Medical Related Statistics in spreadsheet form (schema related) <http://www.healthdata.gov>

## High Level Process

The Big Data Computational Framework used for this PoC is Apache Spark. The principle workflow is the building of a Resilient Distributed Data Set (RDD). We will be building the RDD from data taken from <http://www.guideline.gov> and <http://www.healthdata.gov>.

Once an RDD is created, Spark automatically distributes the data contained in the RDD across a cluster of computing nodes and parallelizes any operation that is required to run on them.

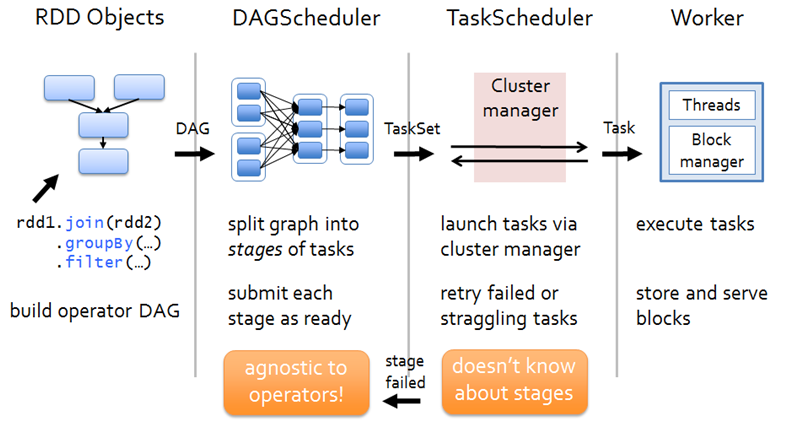


Figure 13: Apache Spark Data Computation Process Flow. Ref: [Spark Blog](https://hxquangnhat.com/tag/spark-submission/)

There are two categories of operations in the Apache Spark framework these are:

* Transformations – which converts one RDD into another
* Actions – which applies functions on the RDD to return a particular result.

We will be using these categories of operation to perform the following tasks, chosen as they are deemed computationally expensive in traditional business intelligence:

* Word Counts
  + Counting all the words in a document and reporting it in the form: word, occurrence
* Word Filtering
  + Filtering phrases, sentences, paragraphs for particular medical keywords.

For this PoC we will also be using the Apache Spark DataFrame to work with our structured data source. The following definition of the Dataframe subsystem is taken from <http://spark.apache.org/>:

*A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. DataFrames can be constructed from a wide array of*[*sources*](http://spark.apache.org/docs/latest/sql-programming-guide.html#data-sources)*such as: structured data files, tables in Hive, external databases, or existing RDDs.*

We will be utilizing the DataFrame to execute SQL like queries and extract result sets.

## Deployment

**As per PoC – Deliverable 2.**

The technology stack used for this PoC has the significant advantage of being scalable. It can be installed on Laptop or Mac so that developers can build locally, peer review before uploading to the cloud. In the Amazon Cloud there will be three environments:

* Development
  + Code will undergo build and unit testing
* System Integration Test
  + Application will undergo regression testing.
* Release
  + Stable releases will be deployed to this environment for evaluation

The diagram below depicts this lifecycle.

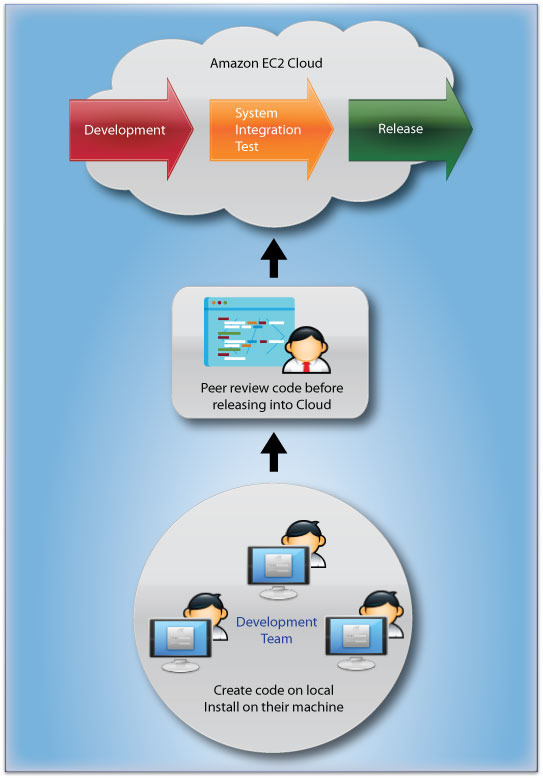


Figure 14: Deployment Diagram. Ref: Apurba Technology – Original Content

### Source code and document management

The system GitHub will be used by the developers as a repository for all source code management. All code as part of the PoC will be documented fully. This documentation will be in the form of both comments and formal implementation documents.

## Build Environment

The diagram below shows the components within our build environment.

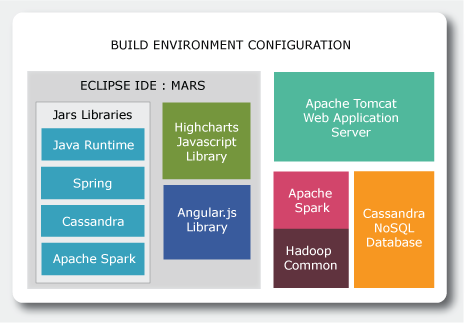


Figure 15: Build Environment Diagram

## Software Implementation

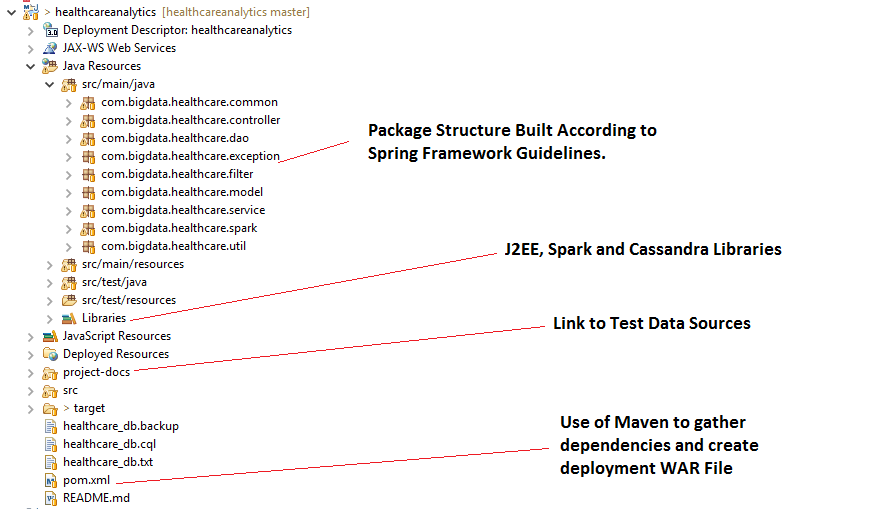
The IDE used for this PoC is Eclipse Mars. The Eclipse project utilizes J2EE (Spring), Apache Spark and Cassandra libraries. In addition, Maven is used to manage dependencies and build the deployment WAR file for the application. 

Figure 16: Screen shot of Eclipse Project. Ref: Apurba Technology – Original Image

## Test Criteria

### Objective

Benchmark the technology stack put forward in this PoC using unstructured (free text) and structured (schema-discernable) medical data.

As mentioned data will be taken from the following sources:

* <http://www.guideline.gov>
  + Unstructured Free Form Medical Text
* <http://www.healthdata.gov>
  + Structured/Schema Discernable Data

The table below shows a mapping between Data Source, Functionality, Spark Processing Feature and Metric:

| **Data Source** | **Functionality** | **Spark Processing Feature** | **Metric** |
| --- | --- | --- | --- |
| <http://www.guideline.gov>  Unstructured Free Form Medical Text | * Word Count * Keyword Search | * RDD Creation * RDD Operations * Utilization of Clustering features | Timings   * Establish Spark Session * Creation of RDD * Perform Single Operation * Reliability |
| <http://www.healthdata.gov> | * Leverage Business Intelligence from schema discernable Data | * RDD Creation * RDD Operations * Utilization of Clustering features * Utilization of Data Frames * Leverage SQL like queries | * Establish Spark Session * Creation of RDD * Establishing a Dataframe * Perform Single Operation * Reliability |

# Conclusion

In this document we have presented the architectural design to express how the technology stack discussed in the deliverable 2 document would actually be used. We presented the use cases and the data being used to realize them. Finally, a high level overview of how it is to be implemented including deployment, build and test criteria.