****

dAnalytics

Design Document

Release 0.1

DRAFT

**Version**: 1.0

**Last Modified:** June 21, 2015

Prepared by:

TurningPoint Global Solutions

1355 Piccard Drive, Suite 250

Rockville, MD. 20850

**APPROVALS**

The undersigned acknowledge that they have reviewed the Design Document and agree with the information presented within this document. Changes to this Design Document will be coordinated with, and approved by the undersigned, or their designated representatives.

| Signature: |  | Date: |
| --- | --- | --- |
| Print Name: | David Hong |  |
| Title: | Project Lead |  |
|  |  |  |

**REVISION HISTORY**

| **Version** | **Date** | **Organization/Point of Contact** | **Description of Changes** |
| --- | --- | --- | --- |
| 0.1 | 06/21/2015 | TurningPoint DevelopmentTeam | Initial Draft Version |
|  |  |  |  |

Table of Contents

[APPROVALS i](#_Toc422693522)

[REVISION HISTORY ii](#_Toc422693523)

[1. INTRODUCTION 5](#_Toc422693524)

[1.1. Purpose 5](#_Toc422693525)

[1.2. Intended Audience 5](#_Toc422693526)

[2. REFERENCED LINKS 6](#_Toc422693527)

[3. REFERENCED DOCUMENTS 6](#_Toc422693528)

[4. Overview 6](#_Toc422693529)

[4.1. The dAnalytics Project 6](#_Toc422693530)

[5. DESIGN CONSIDERATIONS 7](#_Toc422693531)

[5.1. System Goals 7](#_Toc422693532)

[6. Architecture 8](#_Toc422693533)

[6.1. Architecture model 8](#_Toc422693534)

[6.2. Flow Diagram 8](#_Toc422693535)

[7. Technology Stack 9](#_Toc422693536)

[7.1. Java (v 1.8) 9](#_Toc422693537)

[7.2. Data Processing 10](#_Toc422693538)

[i. Apache Hadoop 10](#_Toc422693539)

[ii. HDFS 10](#_Toc422693540)

[iii. Apache Sqoop 10](#_Toc422693541)

[7.3. Data Storage 10](#_Toc422693542)

[i. MariaDB 10](#_Toc422693543)

[7.4. Web API 10](#_Toc422693544)

[i. Spring Framework 10](#_Toc422693545)

[ii. Spring Boot 11](#_Toc422693546)

[7.5. Presentation Component 11](#_Toc422693547)

[i. AngularJS 11](#_Toc422693548)

[ii. JavaScript/jQuery 11](#_Toc422693549)

[iii. Twitter Bootstrap 11](#_Toc422693550)

[iv. HTML5 11](#_Toc422693551)

[v. c3js 12](#_Toc422693552)

[vi. Apache Tomcat Container 12](#_Toc422693553)

[7.6. Build Tools 12](#_Toc422693554)

[i. Gradle 12](#_Toc422693555)

[7.7. Collaborative Development tool 12](#_Toc422693556)

[i. GitHub 12](#_Toc422693557)

[8. Data Processing Component 12](#_Toc422693558)

[9. UI / Presentation Layer 14](#_Toc422693559)

[9.1. Web Component 14](#_Toc422693560)

[9.2. Supported Browsers: 14](#_Toc422693561)

[10. Web Services Layer – RESTful Web Services 15](#_Toc422693562)

[10.1. Intended consumers 15](#_Toc422693563)

[10.2. INPUT/OUTPUT FORMAT SUPPORTED 15](#_Toc422693564)

[10.3. AUTHENTICATION MECHANISM 15](#_Toc422693565)

[10.4. HTTP GET Requests 15](#_Toc422693566)

[10.5. Available dAnalytics Web APIs 16](#_Toc422693567)

[i. Get Unique DrugList 16](#_Toc422693568)

[ii. Get DrugEvent Count 16](#_Toc422693569)

[iii. Get Unique Country List 16](#_Toc422693570)

[iv. Get Unique Age group List 16](#_Toc422693571)

[v. Get Unique Weight group List 16](#_Toc422693572)

[11. STAR Schema Model 17](#_Toc422693573)

[11.1. Drug Summary 17](#_Toc422693574)

[11.2. Drug Event Summary 17](#_Toc422693575)

[11.3. Drug Substance Summary 18](#_Toc422693576)

[11.4. Drug Substance Event Summary 18](#_Toc422693577)

INTRODUCTION

The OpenFDA is an innovation project that aims at creating easy access to the public data, to create a new level of openness and accountability, to ensure the privacy and security of public FDA data, and ultimately to educate the public and save lives.

The FDA launched its very first openFDA challenge to the developer community to take advantage of the following datasets and explore the range and extent of its impact for 1) research and 2) consumers.

* **Adverse events data.** FDA’s publicly available drug adverse event and medication error reports, and medical device adverse event reports.
* **Recalls data.** Enforcement report data, containing information gathered from public notices about certain recalls of FDA-regulated products.
* **Labeling data.** Structured Product Labeling (SPL) data for FDA-regulated human prescription drug, OTC drug and biological product labeling.

TurnintPoint has taken up the challenge of data analytics and thus the **dAnalytics** was born.

Purpose

The Design Document (SDD) describes the design goals and considerations for dAnalytis and provides a high-level overview of the system architecture and the data design updates. The high-level system design is further decomposed into a lower-level of detailed design specifications for each of the system’s components.

Intended Audience

The intended audience for this document includes Developers, Technical Architects and openFDA Technical Review Board. The document will eventually be made available to the open source community who can make use of the publically available code base to extened it.

1. REFERENCED LINKS
   1. OpenFDA Developer Challenge Announcement - <https://open.fda.gov/update/an-open-challenge-to-tap-public-data/>
   2. OpenFDA web API - <https://open.fda.gov/api/reference/>
   3. Drug Event API - <https://open.fda.gov/drug/event/>
   4. Drug Label API - <https://open.fda.gov/drug/label/>
   5. Drug Enforcement API - <https://open.fda.gov/drug/enforcement/>
   6. FAERS Public data: <http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/ucm082193.htm>
   7. FDA Enforcement Reports Public Data: <http://www.fda.gov/%20Safety/Recalls/EnforcementReports/default.htm>
   8. U.S. Digital Services Playbook <https://playbook.cio.gov/>
2. REFERENCED DOCUMENTS
3. ADS RFQ Final.pdf
4. Attachment A Labor Category Descriptions.pdf
5. Attachment B Labor Category Mapping and Discount.xlsx
6. Attachment C Price Quote Sheet.xlsx
7. Attachment D Sample Copyright Waiver.docx
8. Attachment E Approach Criteria Evidence Template.xlsx
9. Overview

The dAnalytics Project

The dAnalytics aims to make sense of the open data available from the FDA distribution. The raw data documents the Events, enforcement and labeling information for Drug, Food and Devices.

The project processes millions of rows of this public raw data and tries to analyze it from individual drug's perspective. The dashboard aims to give the user a flexible ability to see if a drug has had any reported reaction in the past. If it did, the common behavior or pattern of the reaction.

User can also see and determine the behavior based on many other factors like age groups, or if it was gender specific, or had the wight range of patients any say in the matter. The behavior can also be analyzed over a specific period of time, to determine the environmental causes.

dAnalytics provides a user the most comprehensive, one-stop answers for all the queries on the drug behavior.

The dAnalytics is a modular design based on five major components, listed as follows:

1. The dAnalytics UI Component, to present the data for the consumption of the general public.
2. A Java RESTful API component, which publishes the processed and analyzed data on a public platform.
3. A Custom Data Processing platform, built on the Hadoop Big Data solution,which consolidates, standardizes, augments, and individualizes the Public FDA data.
4. A Database Repository, which stores the derived information to be used by the RESTful APIS.
5. Java ETL program that transforms the public XML into the format consumable by the Hadoop data processor job

Each component is readymade, will be available in the public domain and confirms to the OpenFDA criteria. These components diligently follow the U.S. Digital Services Playbook.

1. DESIGN CONSIDERATIONS

System Goals

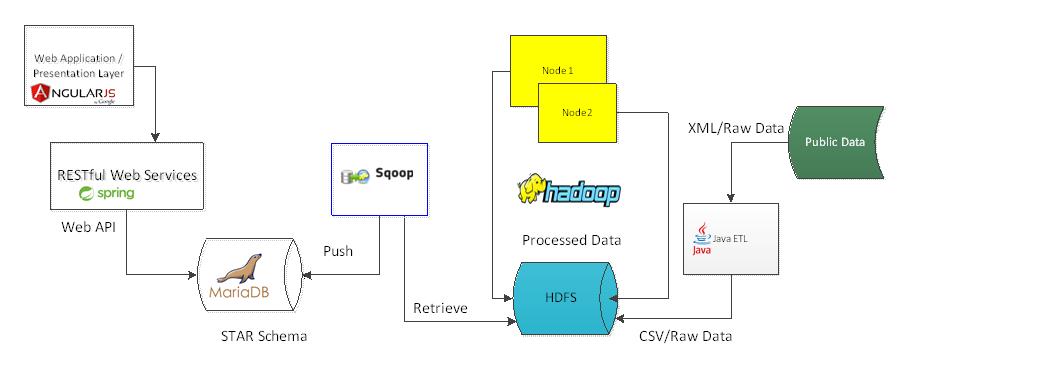
The dAnalytics is designed with the following goals:

1. The system only makes use of the publically available data
2. The system confirms to the true spirit of Open Source Development model
3. The system will address issues and provide a platform for the general public to analyze the drug behavior
4. The system makes use of the modern open source software for a quick turnaround in the development of the Project.
5. The system continues to summarize the public data as and when it becomes available.
6. The system should provide users an interface to track and monitor processes and sub-processes within the system.
7. The system should be extensible.

1. Architecture

Architecture model

The Architecture diagram below shows the System Architecture.

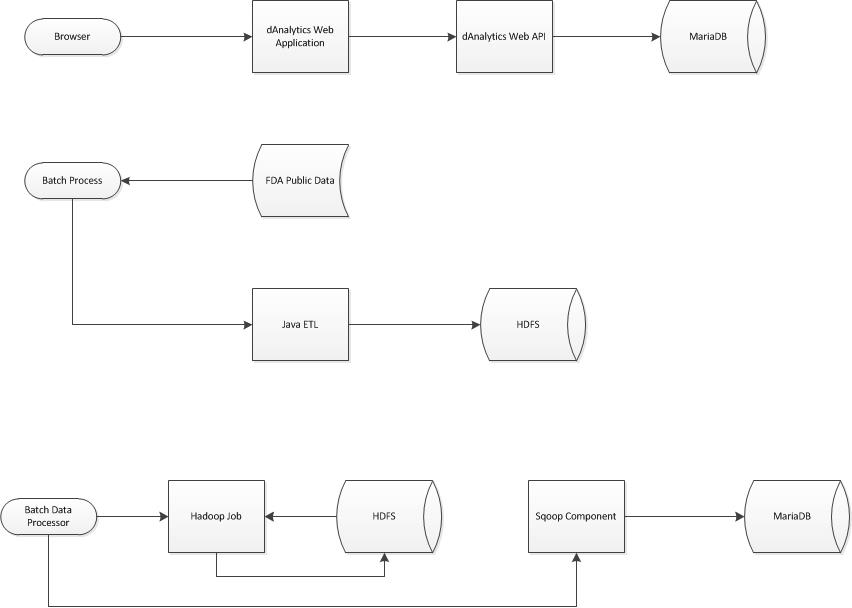


The architecture diagram highlights multiple components of the dAnalytics project.

1. Java ETL (Right most component) – converts the public data into the format that Hadoop component can consume
2. Hadoop component analyzes the transformed data and processes/summarizes it for reporting purposes. Sqoop component moves the data from HDFS into MariaDB (a relational database)
3. MariaDB is the data store from which the reporting is done
4. Web API layer exposes dAnalytics RESTful Web Services using which the external systems can make use of the dAnalytics processed data
5. Presentation Layer (UI) interacts with the dAnalytics Web APIs and provides user intuitive graphs and information

Flow Diagram

The **Flow Diagram** shows all the above components in different flows of the application



**Flow 1**: A Web Browser invokes dAnalytics Web Application component which in turn calls dAnalytics Web API layer to access the MariaDB, the processed data store.

**Flow 2**: An internal Batch (“ETL”) process checks for FDA Public Data availability. If new data is available, it invokes the dAnalytics Java ETL component that maps the data into HDFS

**Flow 3**: Another internal “Data Processor” Batch Job invokes the dAnalytics Hadoop job which checks for HDFS changes; if new changes available, processes the data and stores back in HDFS storage. Sqoop job is then invoked by the batch processor which moves the processed/summarized data into MariaDB data store.

1. Technology Stack

## Java (v 1.8)

Java is an open source, flexible, scalable, platform independent coding language that is widely used by open source developers and users.

## Data Processing

### Apache Hadoop

Hadoop has become a synonym for distributed, Big Data solutions. It is a open-source software for reliable, scalable, distributed computing.  
  
The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.   
  
Hadoop is used in this project for processing large amount of data feed, and analyze it for flexible reporting capabilities of the application. Pig Scripts and Java UDF are used for programming the Hadoop MapReduce jobs.

### HDFS

A distributed file system that provides high-throughput access to application data.

### Apache Sqoop

Apache Sqoop (TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases.

## Data Storage

### MariaDB

MariaDB is a logical choice for database professionals looking for a robust, scalable, and reliable SQL server. The MariaDB development team works closely and cooperatively with the larger community of users and developers in the true spirit of Free and open source software.

## Web API

### Spring Framework

The Spring Framework is a highly extensible and flexible open source application framework and inversion of control container for the Java platform.   
  
The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform, which makes spring framework an invincible option.

### Spring Boot

Spring Boot is a standalone, production grade extension for Spring platform which is used to quickly build applications in Spring. It reduces the overhead of Spring configuration. It embeds Tomcat or Jetty web container, thereby reducing the time in (re)deploying war files. It is highly suitable for modern applications development that are constantly upgraded by automation.

Spring Boot is used in this project to create the **REST API layer** that makes available the processed data for reporting.

## Presentation Component

### AngularJS

AngularJS is an open-source web application framework that lets the programmer write client-side web applications. It uses good old HTML as the template language and extends HTML’s syntax to express the application’s components clearly and succinctly.   
  
It automatically synchronizes data from UI (view) with JavaScript objects (model) through 2-way data binding. To help you structure your application better and make it easy to test, AngularJS provides a client side MVC that provides dependency injection and inversion of control.

### JavaScript/jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.  
  
With a combination of versatility and extensibility, jQuery has changed the way the JavaScript is written.

### Twitter Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

### HTML5

HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web and it operates on CSS3.. This is supported by most of the modern browsers.

### c3js

C3, named to indicate "Comfortable", "Customizable" and "Controllable", is a javascript based graph API which is used in the project to plot intuitive charts dynamically.

### Apache Tomcat Container

Apache Tomcat is an open source web and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

The dAnalytics Web component runs as a web application in the Tomcat Container..

## Build Tools

### Gradle

Gradle is a modern, open source build automation tool. Gradle was chosen as the build automation tool for the project because of its flexible adaptation to different Java versions, command line and IDE code bases. Also, this can be used with any continuous integration tool. This tool brings together the best of ANT and Maven both.

## Unit Testing Tools

### JUnit

JUnit is the de-facto unit testing tool for the Java code. It is open source, highly customizable and ships free with all Java IDEs.

## Collaborative Development tool

### [GitHub](https://github.com/about)

GitHub is a distributed repository and the best tool that has ever been used for collaborative development. The largest code host in the world as of today, Github provides features such as fork, pull requests and branch mergers that makes the code base purely collaborative, yet managed. The features such as wiki pages, issue tracking etc has made "Agile Development" a viable model

1. Data Processing Component

The public data is scanned through and processed in the following manner:

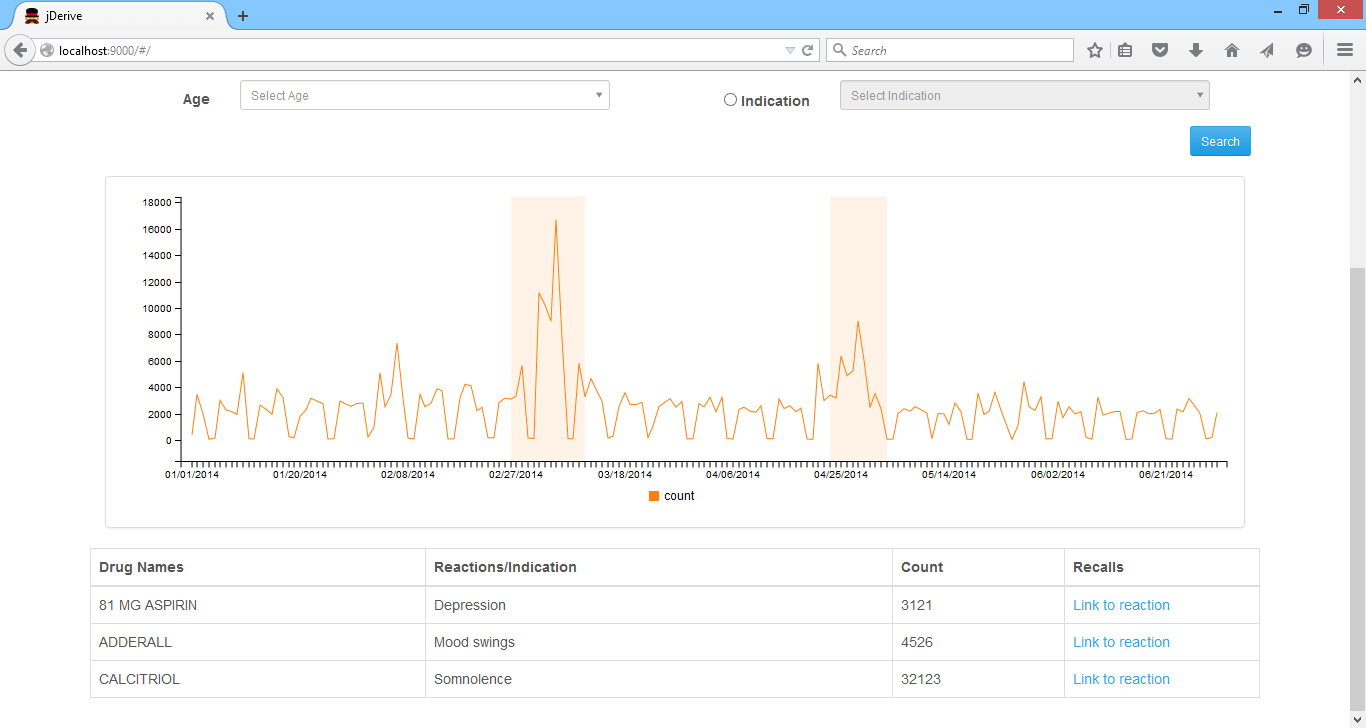
1. Each row of data is processed and unique values of drugs, substances, indications and reactions are extracted.
2. For each of these combinations, the count of events are gathered and generated in the output.
3. The summary data can be read from any of the following dimensions and accurate counts of events can be gathered for the combination.
   1. Drug
   2. Age group
      1. 0-15
      2. 15-40
      3. 40-60
      4. 60-80
      5. Above 80
      6. Unknown
   3. Weight ranges

      2. unknown
   4. Gender
      1. Male
      2. Female
      3. Unknown
   5. Country
   6. Substance
   7. Indication
   8. Reaction
4. The summary is available at daily level as well as the monthly generalization
5. Along with the summary, the processor provides an intelligence on the spike for each of the above mentioned dimensions. Highest spike, and its contents, reasons are segregated and saved to be used by the dAnalytics reports.

1. UI / Presentation Layer

Web Component

Here’s a screenshot of the dAnalytics UI:



* + - The chart shown in the UI intelligently identifies the spike and explanation for the same is provided in the table under it. The highlighted portion on the graph indicates a spike in the adverse event reports in the specified time range.
    - The system also intelligently identifies the time and details of the highest spike for the given input and presents it for the consumption of the user.
    - User can then drill down the graph by zooming on it or filtering on the other filter criteria given in the top panel of the screen.

Supported Browsers:

APS UI supports the following browsers:

* Internet Explorer 9
* Internet Explorer 10
* Internet Explorer 11

1. Web Services Layer – RESTful Web Services

RESTful Web service dAnalytics APIs are the back bone of the dAnalytics Reporting systems. These APIs can be publically invoked and the data could be used by the consumer.

## Intended consumers

* + - dAnalytics Web component
    - External systems submitting request to dAnalytics

INPUT/OUTPUT FORMAT SUPPORTED

JSON

AUTHENTICATION MECHANISM

The Web Services are publically available to be accessed on web.

HTTP GET Requests

Method– HTTP GET

Output Codes -

* + HTTP 200 for successful completion with results
  + HTTP 204 for successful completion with no results
  + HTTP 400 for Bad Request (validation errors)
  + HTTP 405 for any other HTTP Method (Anything other than GET)

All responses to HTTP GET would have the following in the header part of the response

* + Response Codes
  + Number of Rows being returned (in case of no error)
  + Error messages if any. Following error messages can be expected
    - Bad Request
    - Insufficient input parameters
    - Error parsing the input

Available dAnalytics Web APIs

### Get Unique DrugList

URL - /drugs/list

Input - None

Output JSON object (For a search result with more than zero records)

* <TBD>

### Get DrugEvent Count

URL - /drugeventcount/list/

Input - drugid, ageid, weightid, startdate, enddate, countrycode (Any of the input parameters may be null

Output JSON object (For a search result with more than zero records)

* <TBD>

### Get Unique Country List

URL - /country/list

Input - None

Output JSON object (For a search result with more than zero records)

* <TBD>

### Get Unique Age group List

URL - /agegroup/list

Input - None

Output JSON object (For a search result with more than zero records)

* <TBD>

### Get Unique Weight group List

URL - /weightgroup/list

Input - None

Output JSON object (For a search result with more than zero records)

* <TBD>

1. STAR Schema Model

The STAR Schema is the model that has been used in determining the data structure for the relational database – MariaDB. Following is the structure of the STAR model being used:

Drug Summary



Drug Event Summary



Drug Substance Summary



Drug Substance Event Summary

