****

dAnalytics

Design Document

**Version**: 1.0

**Last Modified:** July 06, 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 DevOps team | Initial Draft Version |
| 0.2 | 06/24/2005 | TurningPoint DevOps team | * Adding missing tools and frameworks * Adding the OpenFDA APIs being used * Adding the exhaustive list of dAnalytics APIs available * Adding details of the Java ETL Component * Adding System deployment architecture diagram |
| 0.3 | 07/02/2015 | TurningPoint DevOps team | * Updated the document with QA, Monitoring and Maintenance information |
| 1.0 | 07/06/2015 | TurningPoint DevOps team | * Added license information for 3rd party software used |

Table of Contents

[APPROVALS i](#_Toc423942004)

[REVISION HISTORY ii](#_Toc423942005)

[1. Introduction 5](#_Toc423942006)

[1.1. Purpose 5](#_Toc423942007)

[1.2. Intended Audience 5](#_Toc423942008)

[2. Referenced Links 5](#_Toc423942009)

[3. Referenced Documents 6](#_Toc423942010)

[4. Overview and Architecture 6](#_Toc423942011)

[4.1. The dAnalytics Project 6](#_Toc423942012)

[4.2. Design consideration and goals 6](#_Toc423942013)

[4.3. The dAnalytics application architecture 7](#_Toc423942014)

[9.2.2.1 Presentation tier 7](#_Toc423942015)

[9.2.2.2 Business logic tier 7](#_Toc423942016)

[9.2.2.3 Data Processing tier 7](#_Toc423942017)

[4.4. Software Architecture model 8](#_Toc423942018)

[4.5. Use cases 9](#_Toc423942019)

[4.6. Unit Testing 10](#_Toc423942020)

[4.7. Deployment 10](#_Toc423942021)

[9.2.2.1 QA/Staging 10](#_Toc423942022)

[9.2.2.2 Public Cloud (AWS) 10](#_Toc423942023)

[4.8. Monitoring and Maintenance 11](#_Toc423942024)

[9.2.2.1 Maintenance 11](#_Toc423942025)

[9.2.2.2 Monitoring 11](#_Toc423942026)

[5. UI / Presentation Layer 11](#_Toc423942027)

[5.1. Web Component 11](#_Toc423942028)

[6. Web Services Layer – RESTful Web Services 15](#_Toc423942029)

[6.1. Intended consumers 15](#_Toc423942030)

[6.2. INPUT/OUTPUT FORMAT SUPPORTED 15](#_Toc423942031)

[6.3. AUTHENTICATION MECHANISM 15](#_Toc423942032)

[6.4. HTTP GET Requests 15](#_Toc423942033)

[6.5. Available dAnalytics Web APIs 16](#_Toc423942034)

[9.2.2.1 Get Drug Summary (drugNameList, countryName, startDate, endDate, weight, gender, age) 17](#_Toc423942035)

[9.2.2.2 Get Drug Characterization 17](#_Toc423942036)

[9.2.2.3 Get Drug Reaction Summary 17](#_Toc423942037)

[9.2.2.4 Get Unique DrugList 17](#_Toc423942038)

[9.2.2.5 Get DrugEvent Count 17](#_Toc423942039)

[9.2.2.6 Get Unique Country List 17](#_Toc423942040)

[9.2.2.7 Get Unique Age group List 18](#_Toc423942041)

[9.2.2.8 Get Unique Weight group List 18](#_Toc423942042)

[6.6. OpenFDA Web Services Used 18](#_Toc423942043)

[7. Data Processing Component 19](#_Toc423942044)

[9.2.2.1 Java ETL Data formatter 19](#_Toc423942045)

[9.2.2.2 Hadoop Data Processor 19](#_Toc423942046)

[9.2.2.3 Spike Calculator 19](#_Toc423942047)

[8. STAR Schema Model 21](#_Toc423942048)

[8.1. Drug Summary 21](#_Toc423942049)

[8.2. Drug Event Summary 21](#_Toc423942050)

[8.3. Drug Substance Summary 22](#_Toc423942051)

[8.4. Drug Substance Event Summary 22](#_Toc423942052)

[9. Technology Stack 23](#_Toc423942053)

[9.1. Java (v 1.8) 23](#_Toc423942054)

[9.2. Data Processing 23](#_Toc423942055)

[Data Load/Java ETL 23](#_Toc423942056)

[9.3. Data Analysis 23](#_Toc423942057)

[Apache Hadoop 23](#_Toc423942058)

[HDFS 23](#_Toc423942059)

[Apache Pig 23](#_Toc423942060)

[Apache Sqoop 24](#_Toc423942061)

[9.4. Data Storage 24](#_Toc423942062)

[MariaDB 24](#_Toc423942063)

[9.5. Web API 24](#_Toc423942064)

[Spring Framework 24](#_Toc423942065)

[Spring Boot 24](#_Toc423942066)

[9.6. Presentation Component 24](#_Toc423942067)

[AngularJS 24](#_Toc423942068)

[JavaScript/jQuery 25](#_Toc423942069)

[Twitter Bootstrap 25](#_Toc423942070)

[HTML5 25](#_Toc423942071)

[c3js 25](#_Toc423942072)

[Apache Tomcat Container 25](#_Toc423942073)

[9.7. Build Tools 25](#_Toc423942074)

[Gradle 25](#_Toc423942075)

[9.8. Unit Testing Tools 26](#_Toc423942076)

[JUnit 26](#_Toc423942077)

[9.9. Continuous Integration tool 26](#_Toc423942078)

[Travis CI 26](#_Toc423942079)

[9.10. Collaborative Development tool 26](#_Toc423942080)

[GitHub 26](#_Toc423942081)

[9.11. Licenses for the third party platforms/tools used 26](#_Toc423942082)

[10. Conclusion 27](#_Toc423942083)

Introduction

FAERS, the FDA Adverse Event Reporting System makes the Drug adverse reaction reports publically available through their web site. This data, being raw in nature and millions in numbers, are plain as it is, unless someone analyzes them constructively and publishes the results. TurningPoint has taken up this task of analyzing the Drug Adverse Reaction reports (referenced as Events in this document), in a user centric design and making it publically available.

Here are some FDA terms that need to be understood before reading this document further.

* **Adverse events data.** FDA’s publicly available drug adverse reaction 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.

Purpose

The Design Document (SDD) describes the design goals and considerations for dAnalytics 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, DevOps Engineers and Technical Architects. The document will eventually be made available to the open source community who can make use of the publically available code base to extend 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>
2. Referenced Documents
3. ADS RFQ Final.pdf
4. Nonparametric and Semiparametric Models.pdf
5. Overview and Architecture

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 weight 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.

Design consideration and goals

1. The design must be user centric with intuitive and informative presentation layer.
2. The UI must support all modern browsers, must be responsive to support multiple devices like computer, mobile phone and tablet.
3. The UI must be Section 508 compliant.
4. The system must be high-available.
5. System must be high-performing. The user requests must be responded with a delay not more than 5 seconds.
6. The system must make use of the modern open source software for a quick turnaround in the development of the Project.
7. The system should continue to summarize the public data as and when it becomes available.
8. The system should be modular, with each module to be independently designed, developed and deployed.
9. The system should be extensible.
10. The system deployment must be automated with a continuous integration tool.
11. The system should provide administrators an interface to track and monitor processes and sub-processes within the system.

The dAnalytics application architecture

dAnalytics software architecture follows the modular specified by FISMA guidelines with components developed and deployed in 3-tiers – the presentation, the business logic and the data processing tiers.

### Presentation tier

The dAnalytics UI Component acts as the presentation layer and presents the data for the consumption of the general public in an intuitive graphical UI.

This component will be deployed as a separate entity and would be developed using a client side MVC tool (AngularJS).

For more details, see the chapter 5 of this document.

### Business logic tier

The dAnalytics Web API component publishes the processed and analyzed data to a public platform. This forms the business logic tier which will be used by the presentation component. These web APIs would be made available to any system that may want to utilize the analysis done by the dAnalytics data processors.

More information is provided in Chapter 6 of this document.

### Data Processing tier

The dAnalytics Data Processing tier, explained exclusively in Chapter 7 of this document, is at the core of the dAnalytics application. This deals with data ingestion, data processing and data storage.

Data Ingestion –

The data to be processed is obtained from an external public source, openfda.gov. This data is then mapped to dAnalytics specific format with a Java ETL process.

Data Processing -

The data is then fed into a big data processor, the Hadoop component, for analysis. The Hadoop component consolidates, standardizes, augments, and individualizes the Public FDA data into the API consumable data structure, defined in a star schema model.

Data Storage -

The data storage component is built on the MariaDB relational database. It stores the data to be used by the dAnalytics RESTful API component, and eventually by the dAnalytics presentation component.

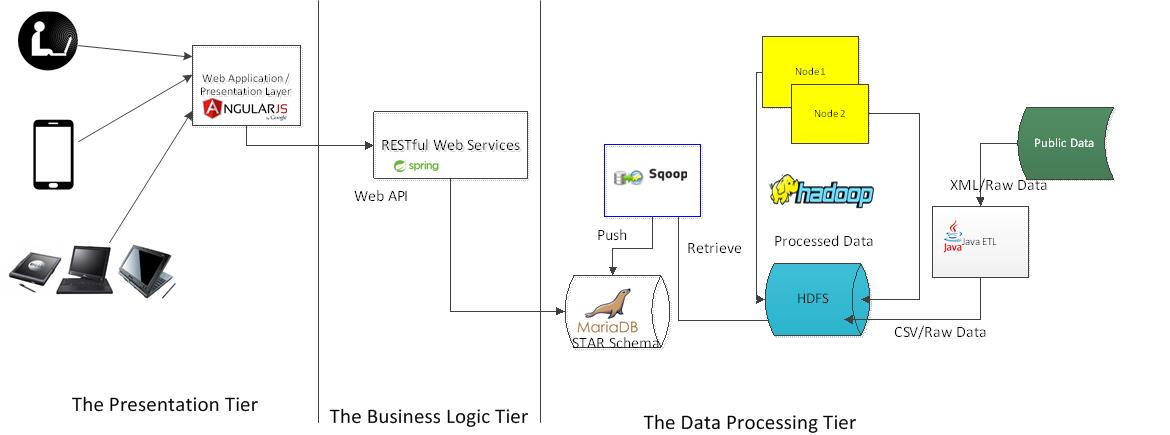
Each of these components are built, deployed and would be made available in the public domain and are accessible via any modern web browser, from any of the devices like computer, mobile and tablet.

The Hadoop component will also be deployed on production, and would be available to be executed. But, it will not have any impact on the run-time status of the application.

Hadoop component is used only for back-end data processing, and not for run-time data crunching.

Software Architecture model

The following Software Architecture diagram shows the different components of the application.



The Presentation Tier is a Web Application that is built using a client MVC tool called AngularJS. This component will be deployed separately and is accessible via a web URL – <http://danalytics.tpgsi.com>

The Business Logic Tier is a Web Application built on the Spring tool, using a framework called Spring Boot which provides detailed set of configuration and hosts Apache Tomcat within.

The Data Processing Layer is mainly a Hadoop component built on the HDFS. Input to Hadoop is a dAnalytics specific Java ETL tool. Part-files containing the processed data are written back in HDFS. Sqoop picks the processed data from HDFS and moves it into the MariaDB.

The Hadoop Data Processor component need not be live until new files are made available by the FDA.

Use cases

**Presentation and Business Logic**

**Use Case 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.

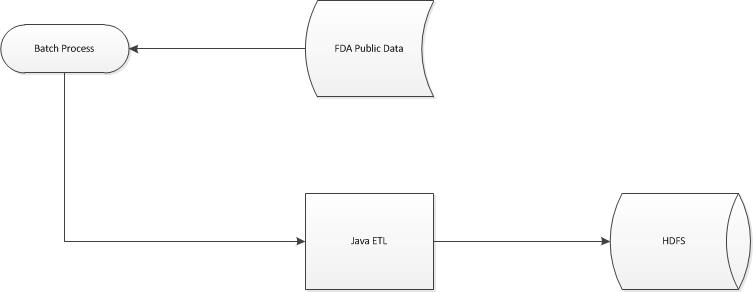


**Use Case 2**: A Web Browser or a client system invokes dAnalytics Web API component to access the processed data stored in the MariaDB.



**Data Processor**

**Use Case 1**: 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



**Use Case 2**: 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.



Unit Testing

The dAnalytics will have modular unit test plans. While the business process component would have automated unit testcases, the UI component will have manual test cases to be executed upon every deployment.

The dAnalytics API unit test cases would be automated using the JUnit tool.

Deployment

### QA/Staging

dAnalytics will have the Travis CI configured to integrate the code changes onto the staging environment on a daily basis.

* + - Travis CI synchronizes code from the GitHub repository
    - Makes the business layer build using gradle
    - Makes the UI layer build using grunt
    - Runs new Database scripts
    - Configures and packages the components
    - Deploys and restarts SpringBoot container
    - Refreshes Apache Web container
    - Runs JUnit test cases on SpringBoot application

### Public Cloud (AWS)

dAnalytics will be deployed on the AWS cloud IAAS platform.

At runtime, the UI and the business components exist along with the MariaDB instance. The Hadoop component will also be deployed and available to be executed, but, it will not have any impact on the run-time status of the application.

Hadoop component is used only for back-end data processing, and not for run-time data crunching.

Monitoring and Maintenance

Maintenance

dAnalytics components would be deployed on the AWS instances with a virtual container – Docker. The Docker container will be updated with custom-dAnalytics-images for each of the dAnalytics components.

The dAnalytics API image would retrieve the database dump at a shared location to start with, creates the data structure, deploys the API image on SpringBoot and restarts the application.

Whereas the UI component is loaded from the UI image.

Monitoring

dAnalytics Application health is designed to be monitored by an open source, highly reliable, easily configurable, monitoring system – Monit.

The Monit application monitors the health and availability of each of the three components (UI, Business and Database) on AWS and sends emails to the configured distro, marked as Alerts or Information.

1. UI / Presentation Layer

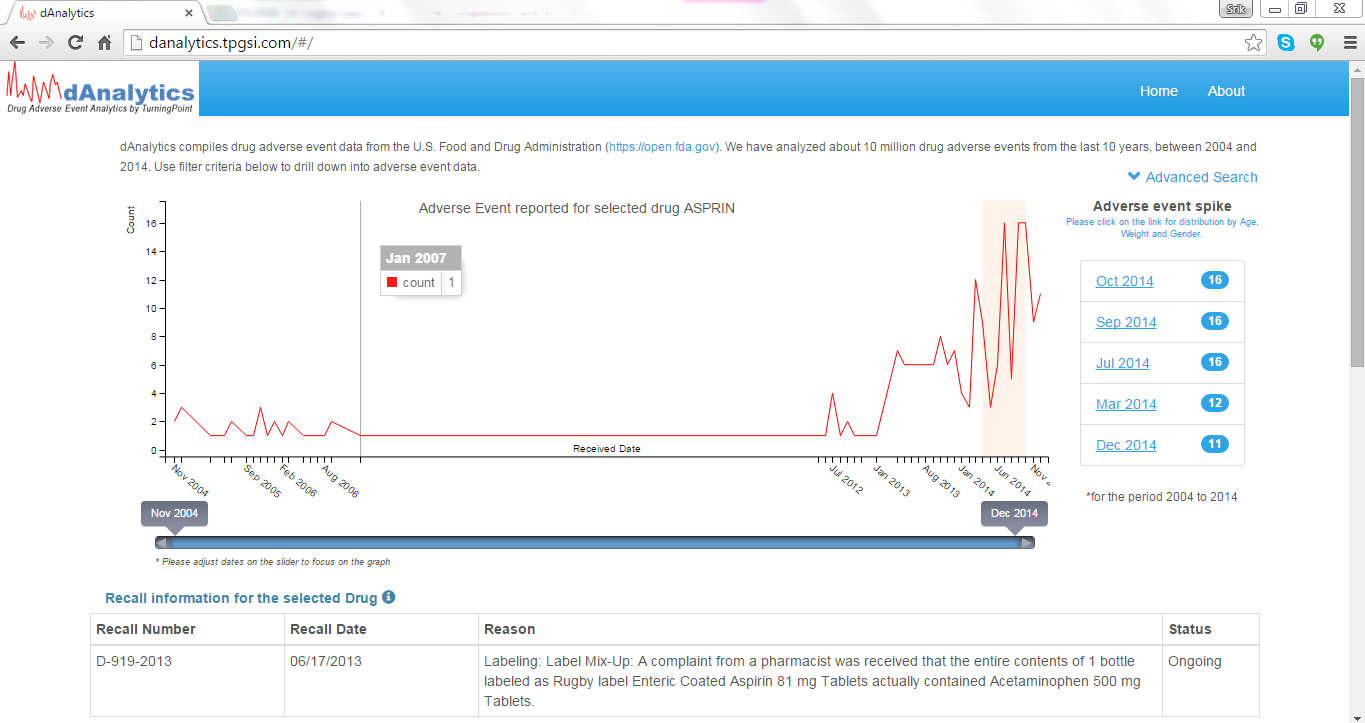
Web Component

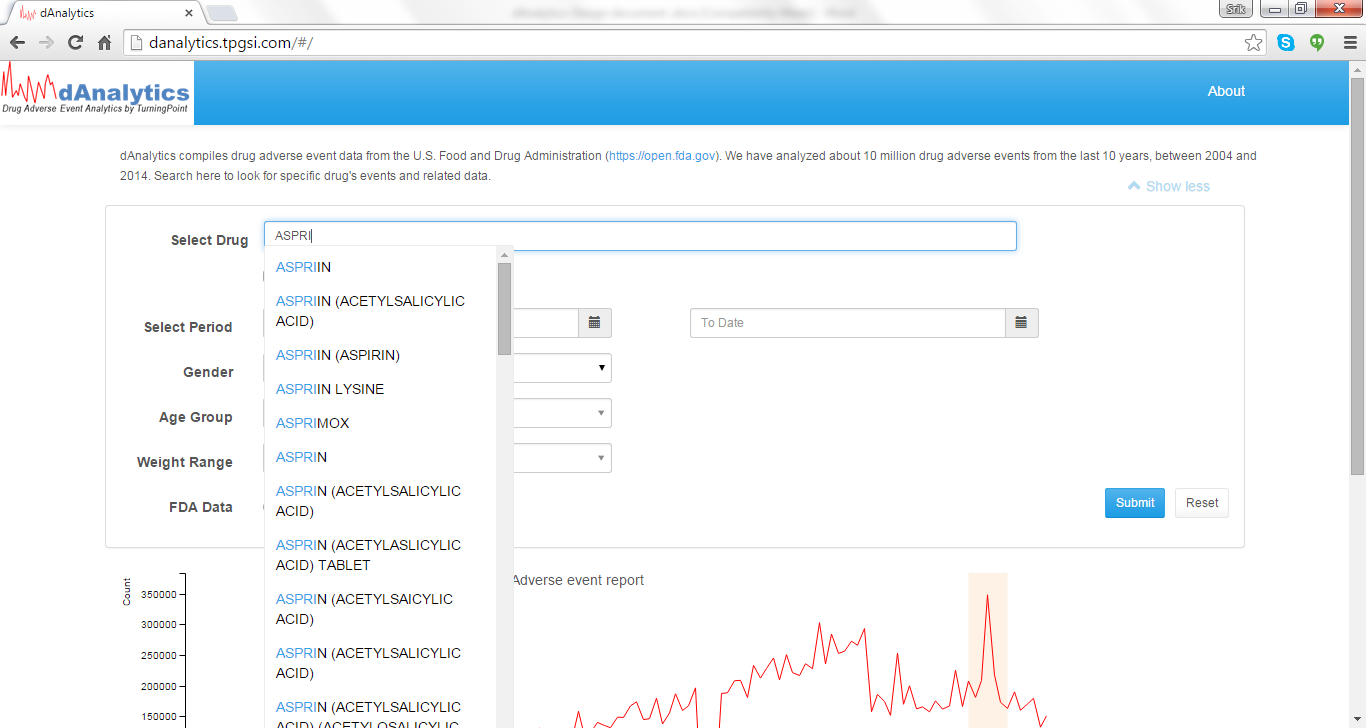
The dAnalytics UI:

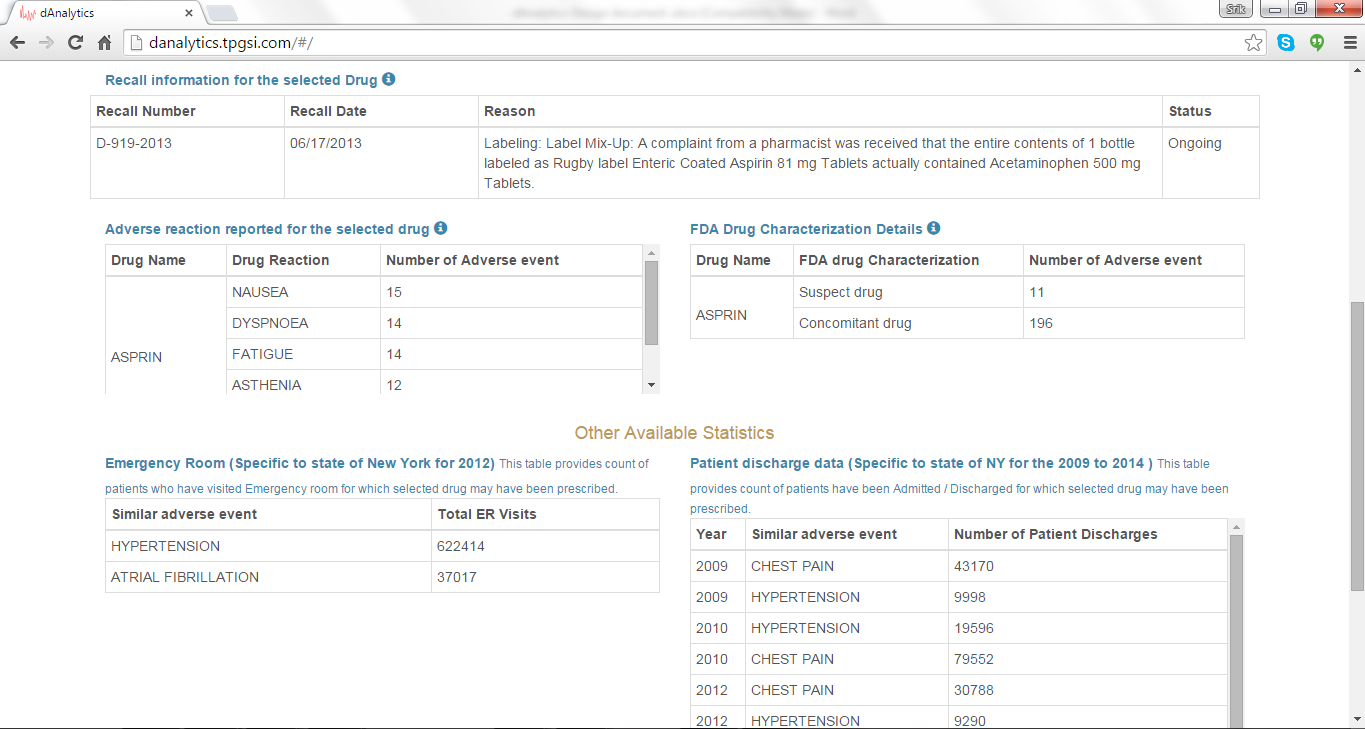
The landing page comes up with a chart showing the Adverse Event counts in the entire database.

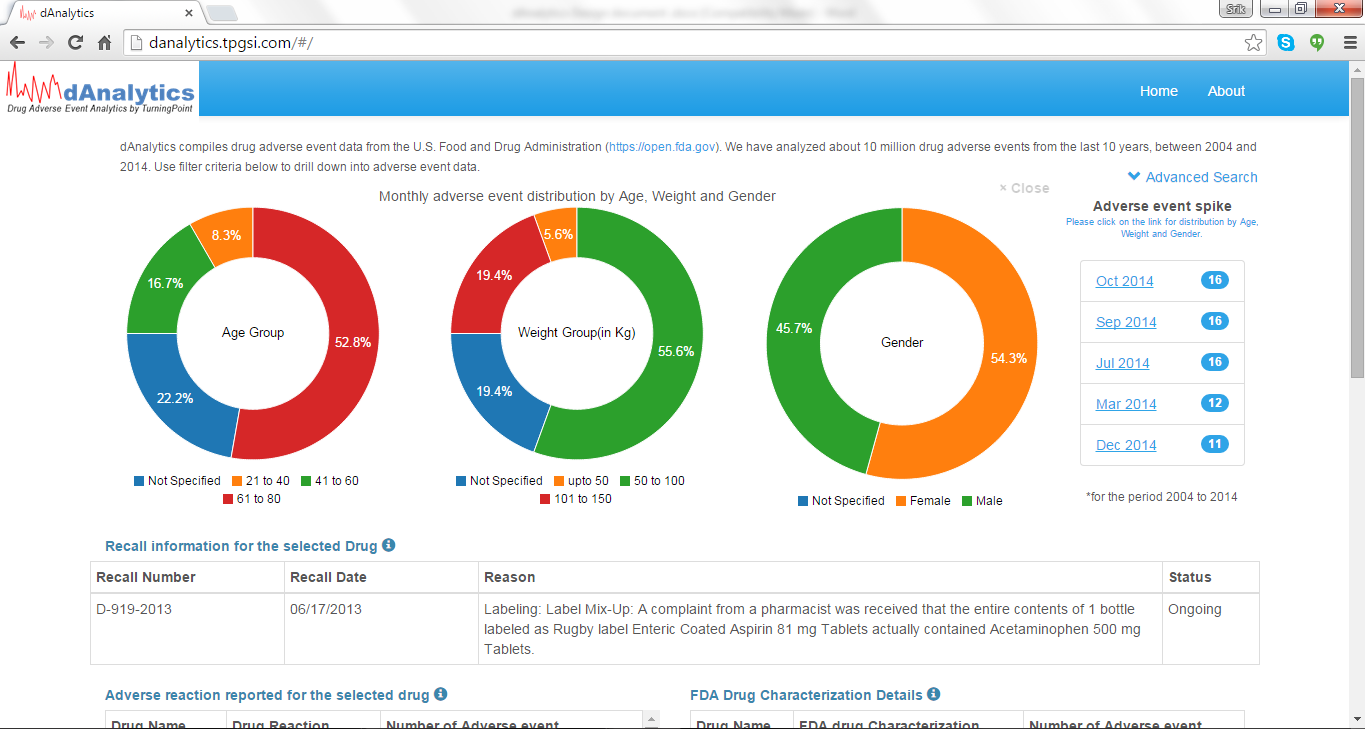
Users can filter to what they want to see by entering a drug name or from weight, age and gender.

The data shown on the graph corresponds to the event count for the drug for the selected period of time, spikes that was noted for the drug, Recall or Enforcement information for the drug, Drug’s reactions reported and the FDA Characterization of the drug.









* + - 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.
    - The system provides five highest spikes that happened for the given drug in other periods of time (other than the selected), and provides links on the UI for the users to view the details of the peak
    - UI also provides any recall information for the specified drug by invoking the OpenFDA provided Drug Enforcement APIs.
    - UI is Section 508 Compliant
    - UI is accessible from all HTML-5 compliant browsers

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. The APIs can be connected on HTTPS.

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

The following table has a brief explanation of some of the important web APIs that dAnalytics publishes. For an exhaustive list, refer the passages after the table.

|  |  |
| --- | --- |
| API | Purpose |
| getDrugSummary | This API returns a list of daily event counts for a given drug in a given time range. Additional filter criteria like age, sex, weight and country can be provided for narrowed down results. |
| getDrugList | This API returns a list of unique popular drug names containing a given string |
| getDrugCharacterization | This API returns the event counts for each of the three FDA drug characterization for the given drug |
| getDrugReactionSummary | Returns the event count for the combination of drug and reaction for the selected time range |
| GetSpikes | Returns the list of spikes available for the selected drug |

### Get Drug Summary (drugNameList, countryName, startDate, endDate, weight, gender, age)

URL - /drugsummary/

Input – any combination of drugNameList, countryName, startDate, endDate, weight, gender, age

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

* List of drugevent summary with event count

### Get Drug Characterization

URL - /drugcharacterization/

Input – drugName

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

* Returns the FDA Drug characterization

### Get Drug Reaction Summary

URL - /drugreaction/

Input – drugName

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

* Returns the Drug Adverse Reactions

### 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>

OpenFDA Web Services Used

1. https://api.fda.gov/drug/event
2. https://api.fda.gov/drug/enforcement

The first API is used to get the Drug Adverse Reports information from the Open FDA

While the second API is used to get the Drug Enforcement Information.

1. Data Processing Component

### Java ETL Data formatter

All the available Open FDA drug adverse event reports from <https://openfda.gov> is formatted into a flat CSV structure to be processed by the dAnalytics Hadoop data processor. This is a Java component which uses SAX parsers to read the input XMLs.

### Hadoop Data Processor

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. Drug Characterization
   3. Drug Reaction
   4. Age groups
   5. Weight ranges
   6. Gender
   7. Country
   8. Substance
   9. Indication
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.

### Spike Calculator

dAnalytics uses an indigenous stored procedure on the MariaDB which scans the processed data and identifies the adverse report spikes for each drug over the period and stores them for reporting purposes.

Algorithm used:

* Regardless of the user input on the graph the peak is calculated on the 10 years data (entire data set).
* Summarize the data monthly. Maximum expected for 10 years that is 120 data points.
* Pick the Maximum 3 numbers of the data points.
* Average of full data points (120 maximum).
* P > average + delta
* Delta = Average \* n.
* N is configurable minimum 2.

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



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

### Data Load/Java ETL

dAnalytics Developers implemented a gateway program to convert multiple input data formats of FAERS and open FDA into dAnalytics specific format which is used for analyzing/processing.

The code was developed in Java and would load all the inputs into CSV files which could be processed by the dAnalytics Hadoop layer.

## Data Analysis

### Apache Hadoop

Hadoop has become a synonym for distributed, Big Data solutions. It is an 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 Pig

Apache Pig is the language of Hadoop. It consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

### 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 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. dAnalytics makes use of JQuery to its benefit in providing a highly intuitive UI.

### 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.

## Continuous Integration tool

### Travis CI

Every time the REST API code gets changed on Git, it is automatically built, deployed and unit tested on the Spring Boot application in AWS, using the configuration provided in Travis CI.

Travis CI is a tailor-made tool for projects running on Git. The pull Requests and any merge requests can be tested before the merge is executed. And every change is deployed on staging/production as soon as the test is successful.

We chose Travis CI the moment we made up our mind on the Git.

## 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

## Licenses for the third party platforms/tools used

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Tool/Platform** | **License Type** | **Freeware?** | **Opensource?** |
| 1 | Java | GNU General Public License | Yes | Yes, mostly |
| 2 | Apache Hadoop | Apache License 2.0 | Yes | Yes |
| 3 | HDFS | Apache License 2.0 | Yes | Yes |
| 4 | Apache Sqoop | Apache License 2.0 | Yes | Yes |
| 5 | Apache Pig | Apache License 2.0 | Yes | Yes |
| 6 | MariaDB | GNU General Public License v2 | Yes | Yes |
| 7 | Spring Framework | Apache License 2.0 | Yes | Yes |
| 8 | SprinBoot | Apache License 2.0 | Yes | Yes |
| 9 | AngularJS | MIT License | Yes |  |
| 10 | Javascript | Oracle TM | Yes | Yes |
| 11 | JQuery | MIT License | Yes | Yes |
| 12 | Bootstrap | Apache License 2.0 | Yes | Yes |
| 13 | C3js | MIT License | Yes | Yes |
| 14 | Apache Tomcat | Apache License 2.0 | Yes | Yes |
| 15 | JUnit | Eclipse Public License (EPL) | Yes | Yes |
| 16 | Gradle | Apache License 2.0 | Yes | Yes |
| 17 | Travis CI | MIT License | Yes | Yes |
| 18 | Monit | AGPL | Yes | Yes |

1. Conclusion

The dAnalytics is an attempt to integrate modern technology with the available public data, put to analyze for the greater public use. The development model followed was Agile Sprint. The design is modular, and complies with FISMA guidelines.

This document brings to the fore, all the important considerations that were made during the DevOps activities of dAnalytics and provides technical guidance to those who would like to build a similar system or may extend dAnalytics.