Wits University

School of Electrical and Information Engineering

ELEN7046 - Software Technologies and Techniques

Big Data Visualization using Commodity Hardware and Open Source Software

Solution: Twit-Con-Pro

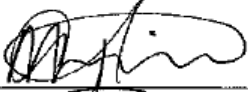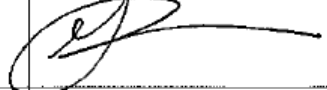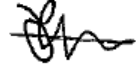| *Authors:*	*Student Number:*
Gareth STEPHENSON	778919
Matsobane KHWINANA	779053
Sidwell MOKHEMISA	1229756
Dave CLOETE	1573016
Kyle TREHAEVEN	0602877N

https://github.com/garethstephenson/ELEN7046_Group2_2016

July 3, 2016

Declaration of Originality

ELEN7046 Group 2 team members hereby declare the following:

- We are aware that plagiarism (the use of someone elses work without their permission and/or without acknowledging the original source) is wrong.

- We confirm that ALL the work submitted for assessment for the above course is our own unaided work except where we have explicitly indicated otherwise.

- We have followed the required conventions in referencing the thoughts and ideas of others.

- We understand that the University of the Witwatersrand may take disciplinary action against each one of us if there is a belief that this is not our own unaided work or that we have failed to acknowledge the source of the ideas or words in our writing.

Member Name	Primary Task	Time Spent (Hours)	Discretionary	Member Signature
Sidwell Mokhemisa	Architecture	111	5 %	
Kyle Trehaeven	Development - Streaming	140	5 %	
Matsobane Khwinana	Development - History	118	5 %	
Gareth Stephenson	Development - MapReduce	205	5 %	
Dave Cloete	Development - Visualization	127	5 %	

ABSTRACT

The project aims to provide a low cost solution to big data processing problems, enabling a commercially viable option to individuals, small businesses and academia using commodity hardware. This report explores the use of Open Source technologies Apache Spark and Scala as well as the low cost and scalable Raspberry Pi's. After building a prototype system to source, process and visualize data from Twitter, it was concluded that commodity hardware is a viable small-scale solution to working with Big Data.

Contents

1   Introduction

This report presents the work done by Group 2 in response to the project brief for ELEN7046: Software Technologies and Techniques.

The report will broadly focus on the following topics:

- The Methodology followed to execute the project;

- The Architecture of the solution developed for the project; and

- The different technologies used to deliver the solution.

1.1   Problem Statement

There is an abundance of big data available to individuals and companies with very limited capacity to refine this data into meaningful information, particularly for small scale endeavours. Big Data processing is often locked behind high cost barriers to entry, and individuals, start ups and academics may find it difficult to be involved in Big Data processing.

1.2   Solution Summary

Commodity hardware is available to provide a means by which the barrier to entry for Big Data projects can be overcome. Simple, low cost components can be leveraged to address each of the parts of a big data processing solution, whether it be data sourcing, transormation, or visualization; and can be scaled according to needs or as required.

1.3   Project Delivery Approach

As the team was not co-located, challenges around communication and general collaboration was raised before the first team meeting. The team made use of the following, online collaboration tools to meet their needs as well as drive the project delivery:

- **Trello:** To manage collaborative tasks and to provide an understanding of the scope of work.

- **WhatsApp:** To initiate any discussion that required an immediate response.

- **Slack:** For knowledge sharing and lengthy, more general discussions.

- **GitHub:** Source Control and Content Management.

- **Google Hangouts:** For collaborative, interactive knowledge sharing .

- **Contact Sessions:** To ensure frequent, face-to-face communication and ensuring the project moved forward, the team decided to meet every Sunday for the duration of the project.

2 Literature Review

2.1 Data Sourcing

This project was intended to deliver a system or solution for the visualization of Big Data. For this reason it is important that we start by defining what Big Data is.

According to Maden[1], Big Data can be defined as "data that is too big, too fast, or too hard for existing tools to process."

The above definition further supports our group's decision to focus on Twitter as the source of data for the project. Statistics have shown the following average figures with regards to the amount of data that one can get from Twitter[2]:

- 6000 per second;

- 350 000 per minute;

- 500 million per day; and

- 200 billion in a year.

2.2 Big Data Processing Algorithm

The team sought to have all the processing of the Big Data received from Twitter done on multiple nodes following the principles of MapReduce.

Apache Spark was used for this project together with MapReduce which in the main delivers the MapReduce functionality based on the algorithm that is broken down into the Mapper class and the Reducer class[3].

The Apache Spark cluster computing framework runs the MapReduce algorithm across all the nodes within the Raspberry Pi network, allowing for fast and efficient querying and transformation of raw data into meaningful information.

2.3 Data Visualization

Data Visualization is key to understanding large sets of data. Graphically representing data allows the human mind to better comprehend abstracted views and makes it easier to identify trends, patterns and anomalies not easily identifiable when looking at data in more traditional forms such as spreadsheets.

3 Success Criteria

- Build a system that uses commodity hardware to solve for big data problems;

- Provide a solution that covers the data sourcing, transformation and presentation of social media data from Twitter relating to the United States and South African elections. The end result must be visualization that provides insight to the sentiment of election candidates on Twitter.

4 Lifecycle Methodology

In order for the team to successfully deliver this project, a development methodology based on IBM Rational Unified Process (RUP) was followed albeit tailored to cater for the specific needs of this project.

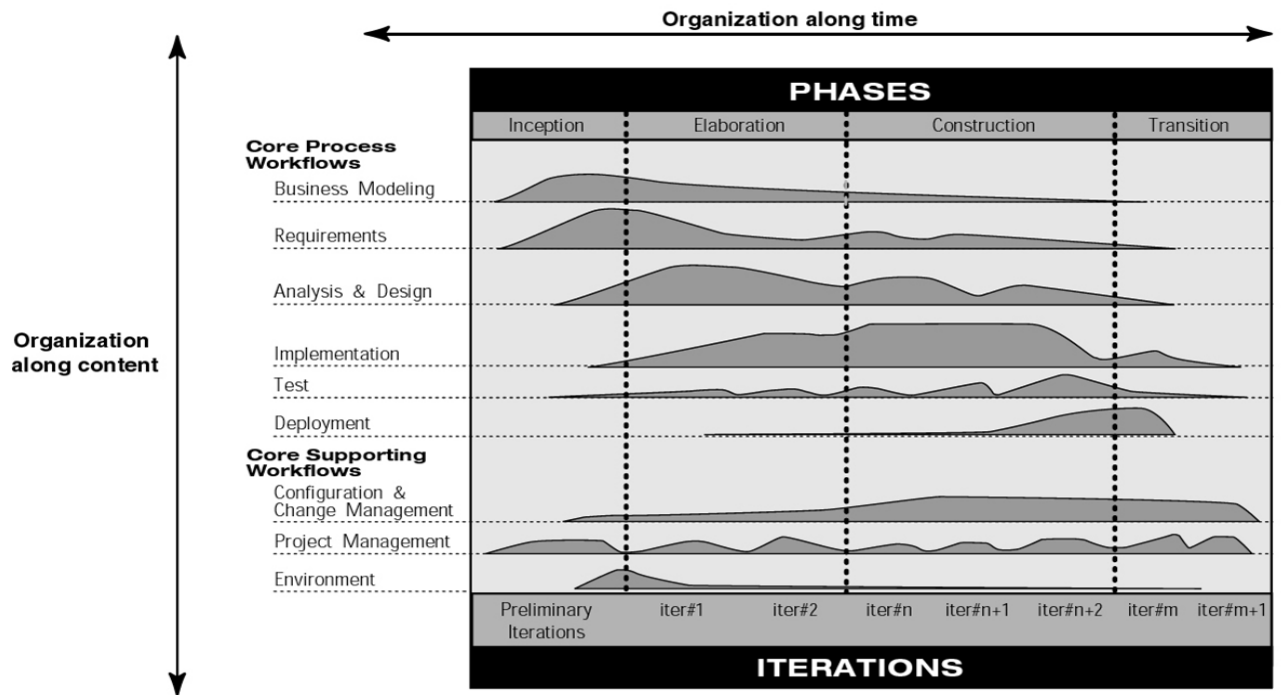The diagram below depicts the IBM RUP model:

Figure 1: IBM Rational Unified Process (Source: RUP, Best Practices for Software Development Teams)

5 Key Risks

Below is the list of the Key Risks that the project team identified at the start of the project:

- Project Technical Risk.

- The Raspberry Pi Infrastructure.

- Team Collaboration.

6 Assumptions and Constraints

6.1 Assumptions and Acknowledgements

- It is assumed that a rudimentary algorithm for sentiment assessment is "Good Enough", forgoing complex natural language inference.

- It is assumed that any incidental tweets sourced (such as EFF referencing the Electronic Freedom Foundation) are minimal and acceptable.

- It is assumed that Twitter is providing us with either the full extent of our search criteria, or if this is not the case, then the sample that is received is a representational slice of all data and is proportionally equivalent.

- It is assumed that information that users provide on twitter (name of their location in particular) is correct and relevant, and that those who do not are in the minority.

- Failing the above, it is assumed that of those who do provide location based information; their distribution is representative of the norm.

- It is assumed that the graphical solution provided can be rendered by any clients that access them.

- It is assumed that the datasets provided for the visualisation components will be small enough to be effectively transferred over the internet to browsers without error.

- It is acknowledged that demographics of the provided data are skewed (younger people more likely to use twitter, poorer South Africans less likely to use twitter).

- It is acknowledged that application software is not optimised extensively.

6.2 Technical Constraints

- **Twitter API:** Twitter only provided100 tweets per request and only allowed 450 tweets within a 15 minutes window[4].

- **GoogleAPI:** Google has a restriction that only allows 2500 requests per day.

- **Hardware:** Slow I/O processing due to cheaper storage (SD Card), limited physical space that could only accommodate just over 1,5 million tweets as well as limited RAM on the Raspberry Pi's.

7 Design Decisions

The table below details all the key design decisions made in the delivery of the solution:

- **History Data:** History data/ batch interface to Twitter was designed to provide past data subscribed to based on date range.

- **Streaming Data:** This interface provides for all subscribed data in real-time starting from now and going into the future.

  One of the motivating factors for splitting the two components mentioned above was that the project followed a divide and conquer strategy to decompose the data collection problem into simpler sub-components. The divide and conquer strategy in software design, helps in avoiding a single point of failure, and it enables parallelism [6].

- **GoogleMaps:** Where privacy settings were not enabled for current location, an interface to GoogleMaps was developed to read the location information of the person tweeting from their Twitter profile and resolve this to actual co-ordinates.

- **Security Directive(s):** It was decided that where security is concerned, only the Twitter and GoogleMaps security requirements be adhered to where integration is concerned. All data acquired from Twitter is data that is already in the public domain, therefore no effort was required to secure the data while in transit, hence the use of only FTP instead of a lot more secure transport mechanism.

8   Solution Design

8.1   High Level Design: Component Architecture

The high level component model below depicts the key features of the solution delivered for the project.

This design can be categorised into the following three features from a functionality point of view:

- Acquiring data from Twitter based on topics subscription - Political Party sentiment in USA and RSA for 2016. This was achieved by using JEE for history and C# for online data acquisition while persisting all data in MongoDB.

- Data processing was achieved using Apache Spark with MapReduce algorithms developed in Scala. Data in MongoDB was extracted and transfered via FTP to the clustered infrastructure where all the necessary transformation happened.

- Extracts from the transformation process were sent to the presentation tier application in JSON format where node.js was used in conjunction with D3 chartz to render the visualized big data content on a web browser.
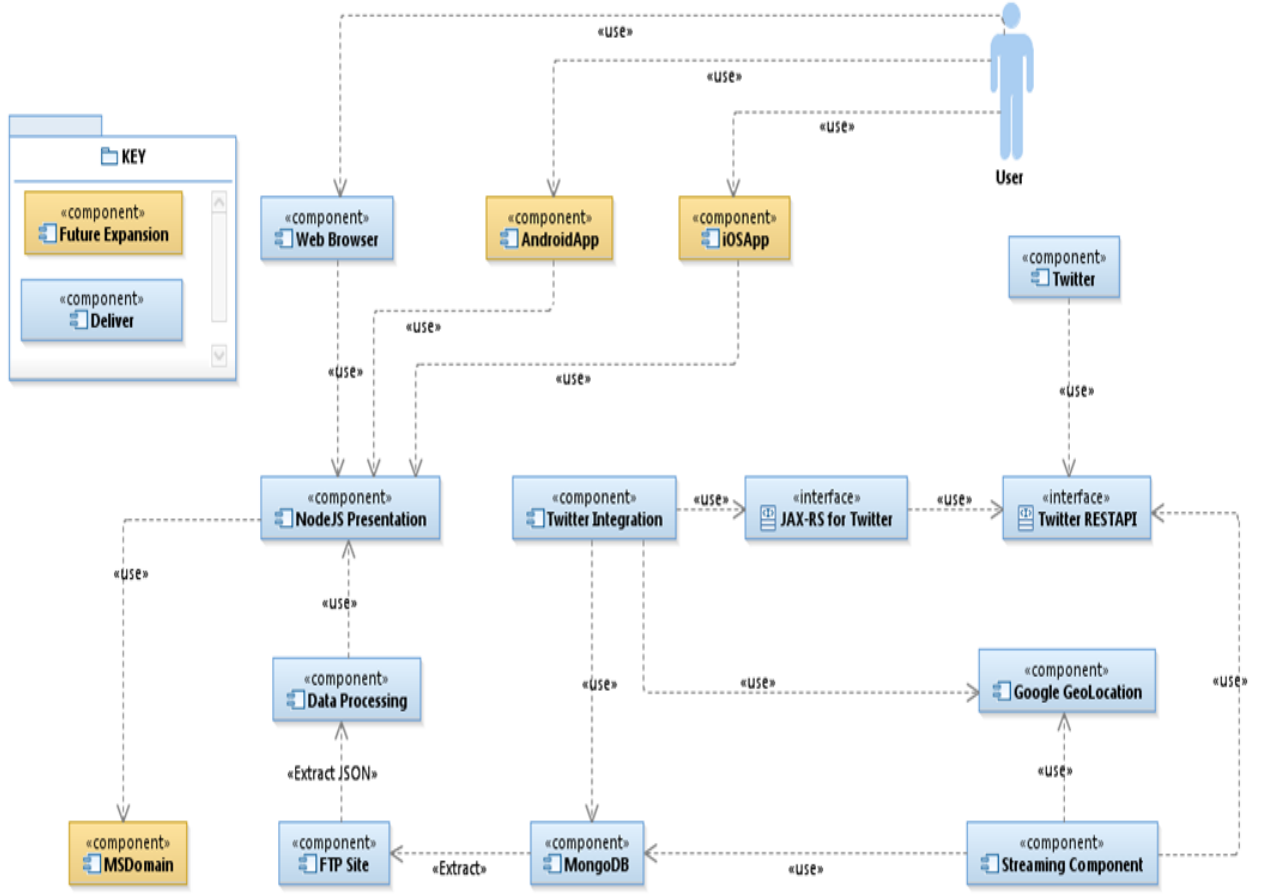
Figure 2: High Level Component Model.

8.2 Detailed Designs

8.2.1 Data Acquisition (History)

The historic data collection component is made up of two major services, namely, Data Extraction Service and Data Distribution Service. These supports the two major use cases for history data collection component, i.e. data acquisition and distribution. The services were modelled using public interfaces, packaged together in the API module. The interfaces are separated from the implementations to avoid tight coupling and thereby making it easy to swap implementations without impacting the API users.

The implementation components perform the actual extraction of the data by interacting with the Twitter REST API; in the process they perform authentication, retrieval of content, mapping the content from Twitter to the applications Tweet entity and persisting the data in MongoDB. Similarly for the distribution of data, the underlying classes handle the retrieval of data from MongoDB and distribute the data using FTP.

The extraction service can be triggered using both the REST interface and the Scheduler, whilst the distribution component can be triggered via the REST interface only.
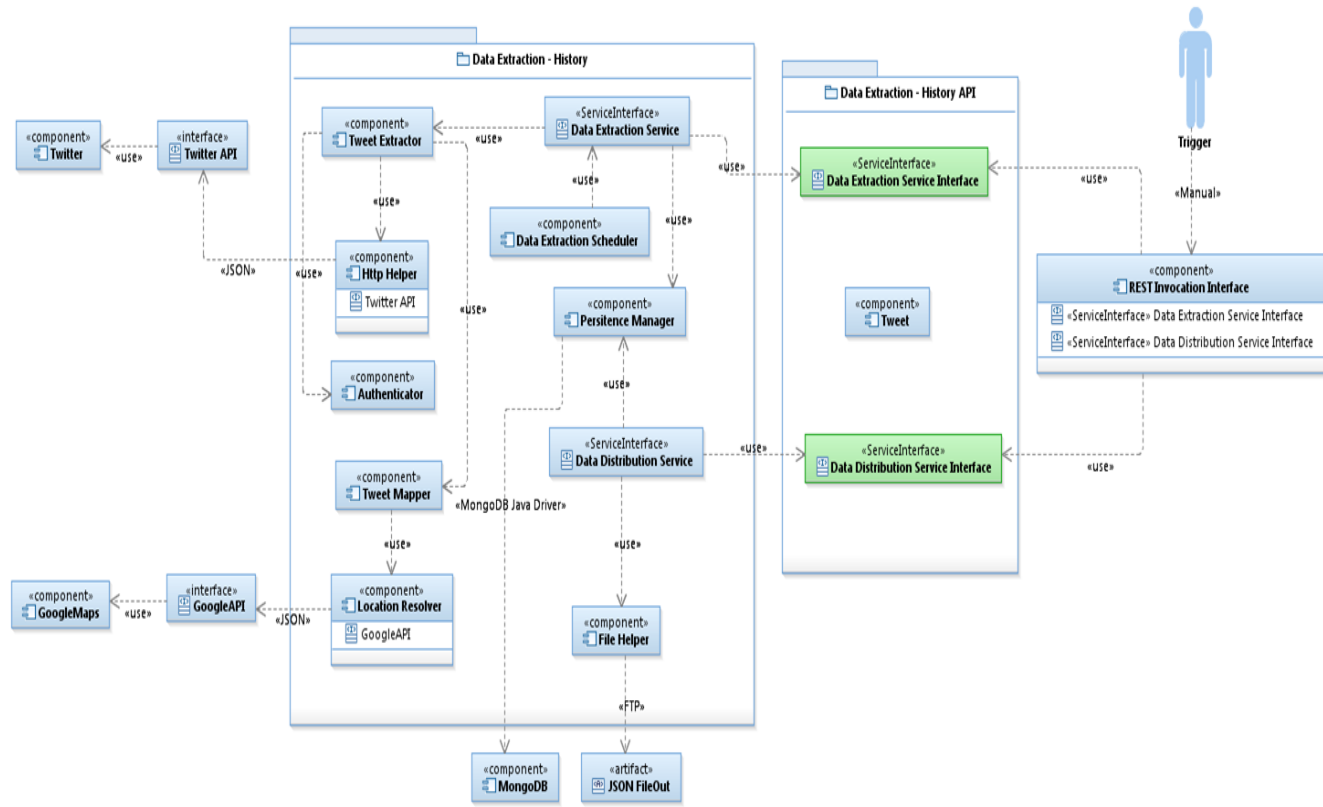
Figure 3: Component Model: History

8.2.2 Data Acquisition (Streaming)

Housed in the Application is a Stream Connection Manager, which manages connections to Twitter leveraging off of the 3rd Party Twitter API wrapper called TweetInvi.

This Connection manager will generate a Twitter Message Handler which will respond to incoming tweets, validate location data with a location verifier component which will consolidate it (if necessary) with the Google Maps API.

The Twitter Message Handler will then submit a Tweet object to the Database manager to persist it. An Extraction manager can subsequently be used to retrieve the tweets as necessary.
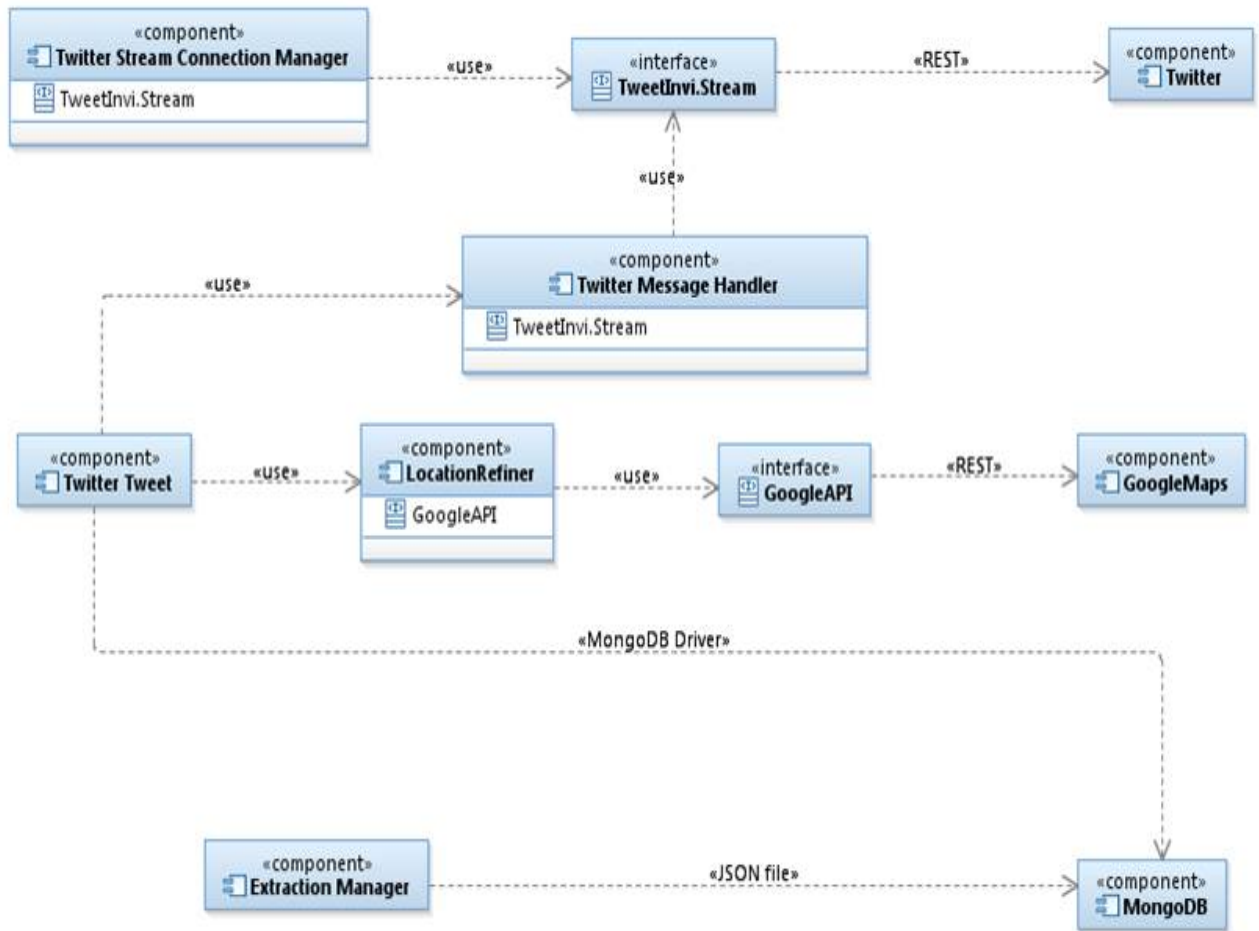
Figure 4: Component Model: Streaming

8.2.3 Data Processing

Raspberry Pi Cluster - Driver Program:

- The program you write creates an instance of a SparkContext.

- The SparkContext then connects to the Cluster Manager which allocates resources across nodes for the application.

- The SparkContext then acquires Executors on Worker Nodes in the cluster, which will run computations and store data for the application hosting the SparkContext.

- The application is then sent to the Executors, and then the Tasks that the Executors need to execute are sent through.

- Executors can communicate with each other using peer-to-peer networking (like bit torrent), in order to transmit shared data (broadcast variables).

- Once the Executors have completed their tasks, they communicate back to the Driver Program.
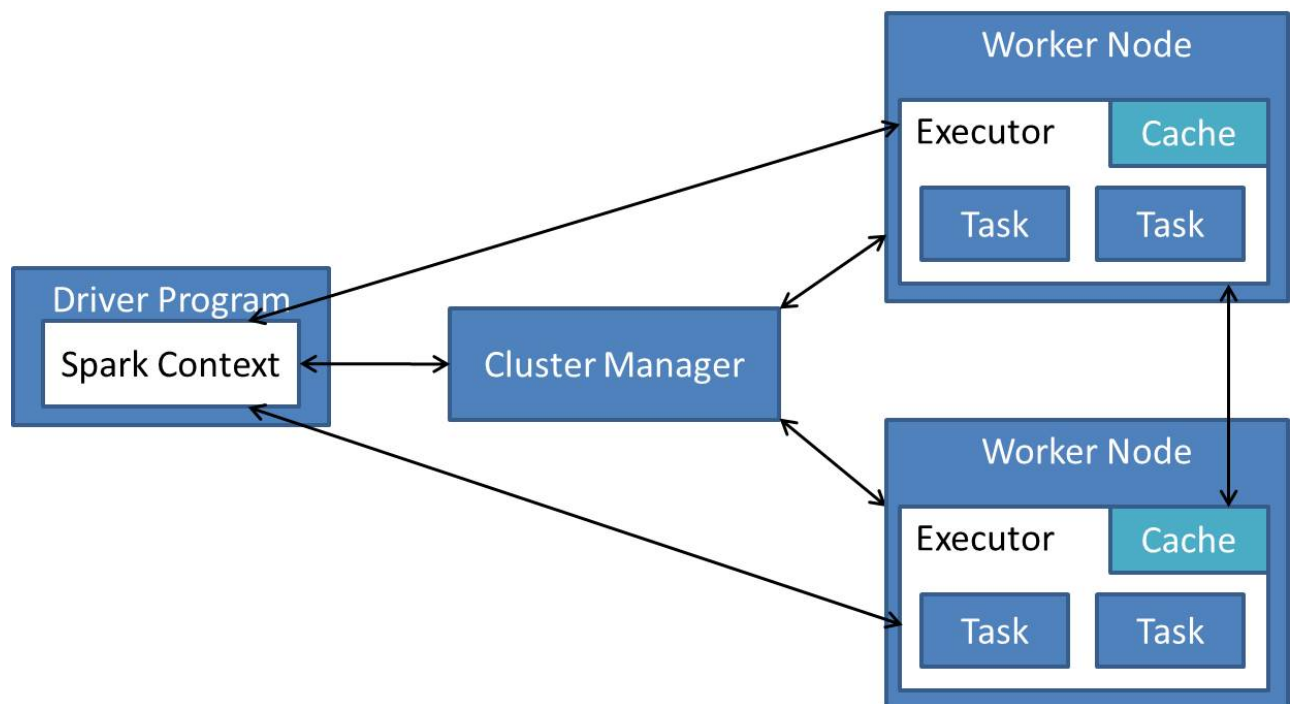
Figure 5: Context Diagram: Data Processing

Application Flow (DAG):

The twitter data is loaded from flat text files, mapped through various functions to strip irrelevant data and transform the text data into Tweet objects. The Tweet objects are then sorted by date. From there, filters are applied to the Tweet objects to only find Tweets for a particular category and hour of day. These results are then summed and reduced to a category count.

Figure 6: Application Flow

8.2.4 Data Visualization

The Visual component implements the MVC pattern to separate the View and controlling logic from the Data. This allows for the data components to be developed separately from the Views. Between the Views and Data Models were Controllers responsible for loading the views based on the set configuration and provide the views with the required data model.

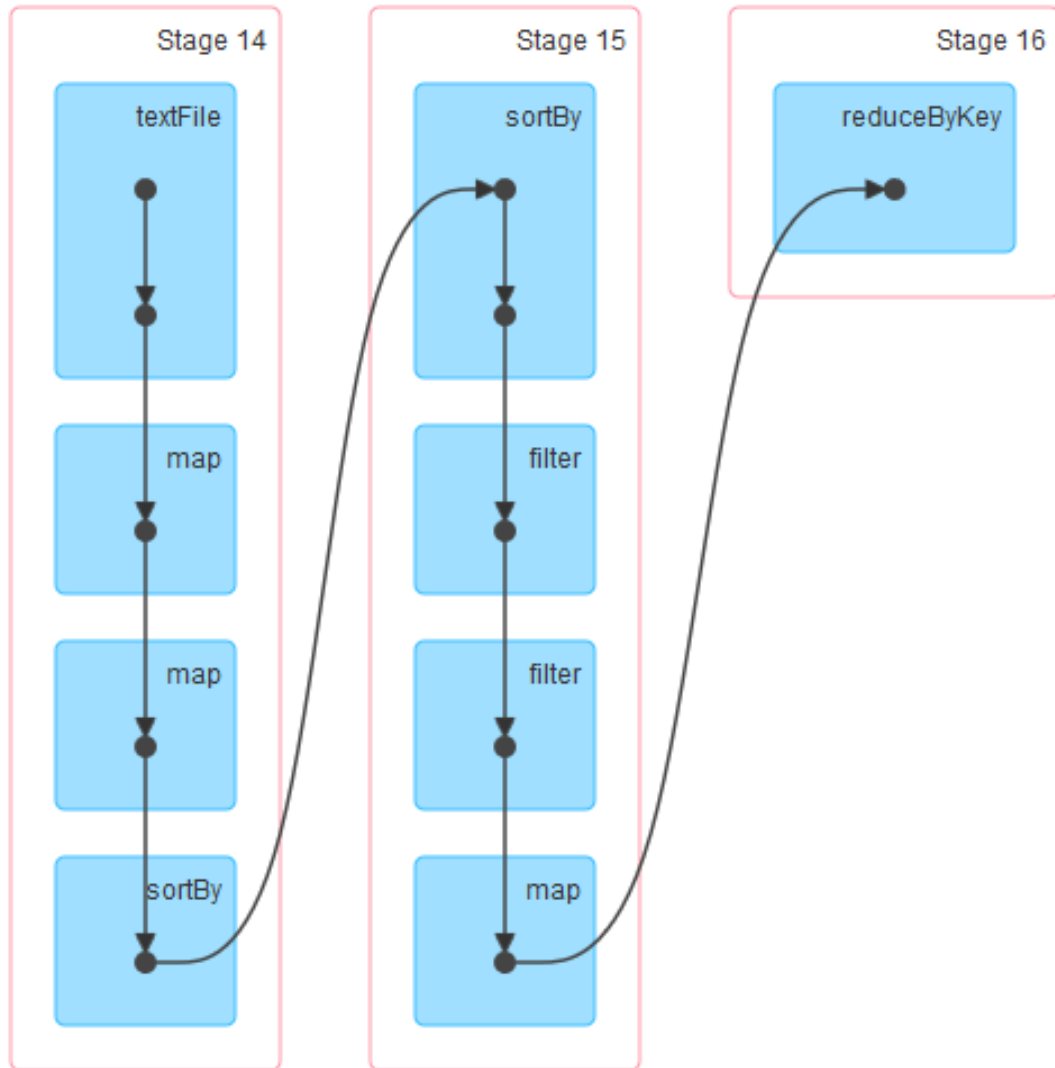The Data Visualisation component was designed to be content agnostic and it allows for the comparison of categories in a topic regardless of the topic or categories. All charts were designed to be scalable. All graphs will render as the view point changes allowing for a pleasant user experience.
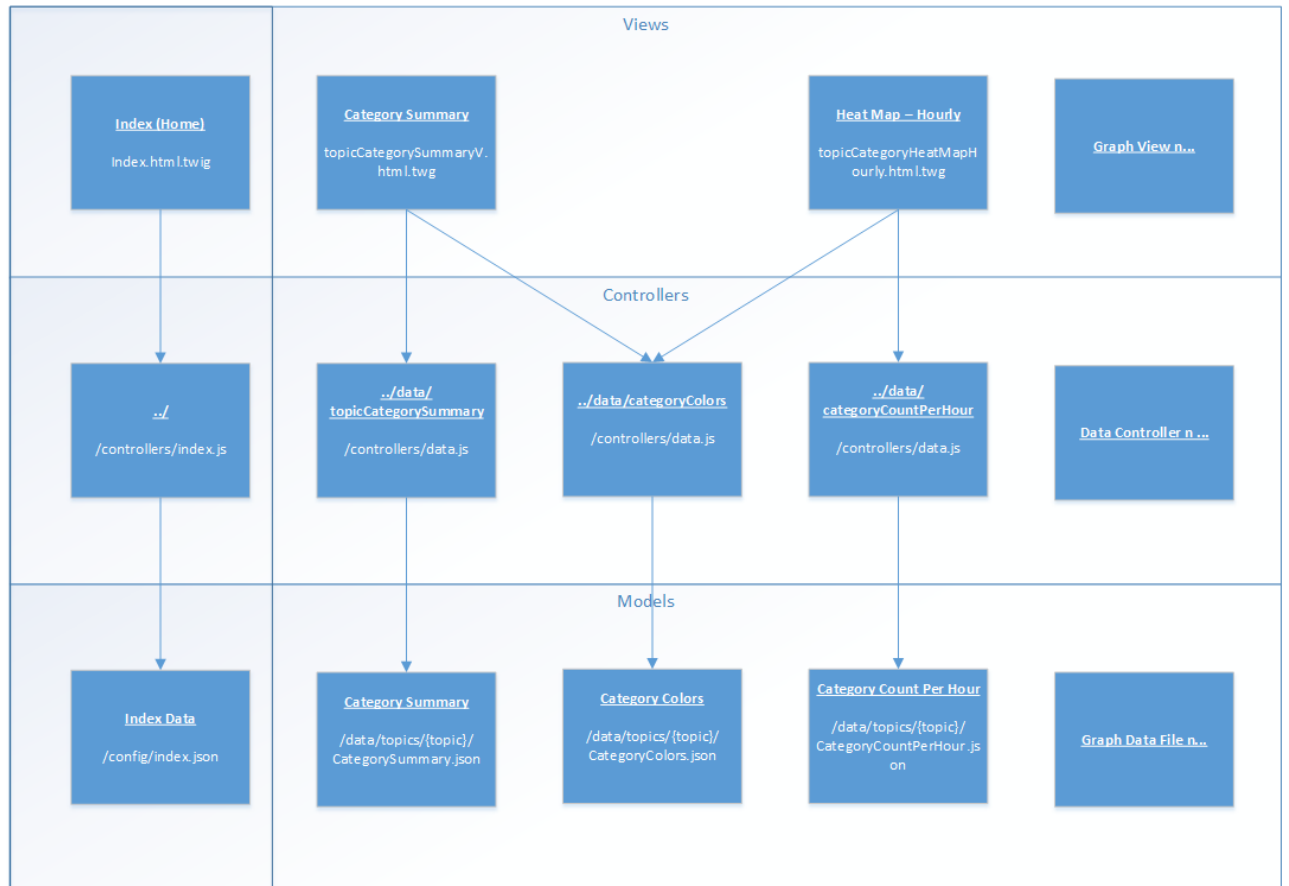

Figure 7: Visualization: MVC Framework

8.3 Operational Model: Infrastructure Design

The Raspberry Pi environment is limited in system resources, specifically RAM. It only has 1GB of RAM which is partitioned for use between the system (accessible by the user) and the GPU.

The Raspberry Pis were configured with the least amount of RAM for the GPU

(16MB), leaving the remaining space available for user applications. They were not going to run any sort of GUI or windowing system (like the X Window System), so any larger allocation of RAM for GPU use was unnecessary. The Raspbian Linux operating system (OS) was installed on the Raspberry Pis, as it has been tailored to best utilise the hardwares resources and features. Further configuration was done on the OS to disable the daemons responsible for controlling the on-board Bluetooth and Wi-Fi modules, thus freeing up even more available RAM. Linuxs Network File System (NFS) service was installed and enabled on the Raspberry Pi master node so the slaves in the Spark configuration could operate on a single source of data, with the least amount of memory overhead. When using HDFS, the RAM used by the slave nodes increased from 99MB to 125MB, and from 181MB to 206MB on the master node. In addition to extra RAM use, the time to upload the source data (1GB of plain text files) to HDFS took around 8 minutes, as the data was partitioned across all the data nodes within the cluster.
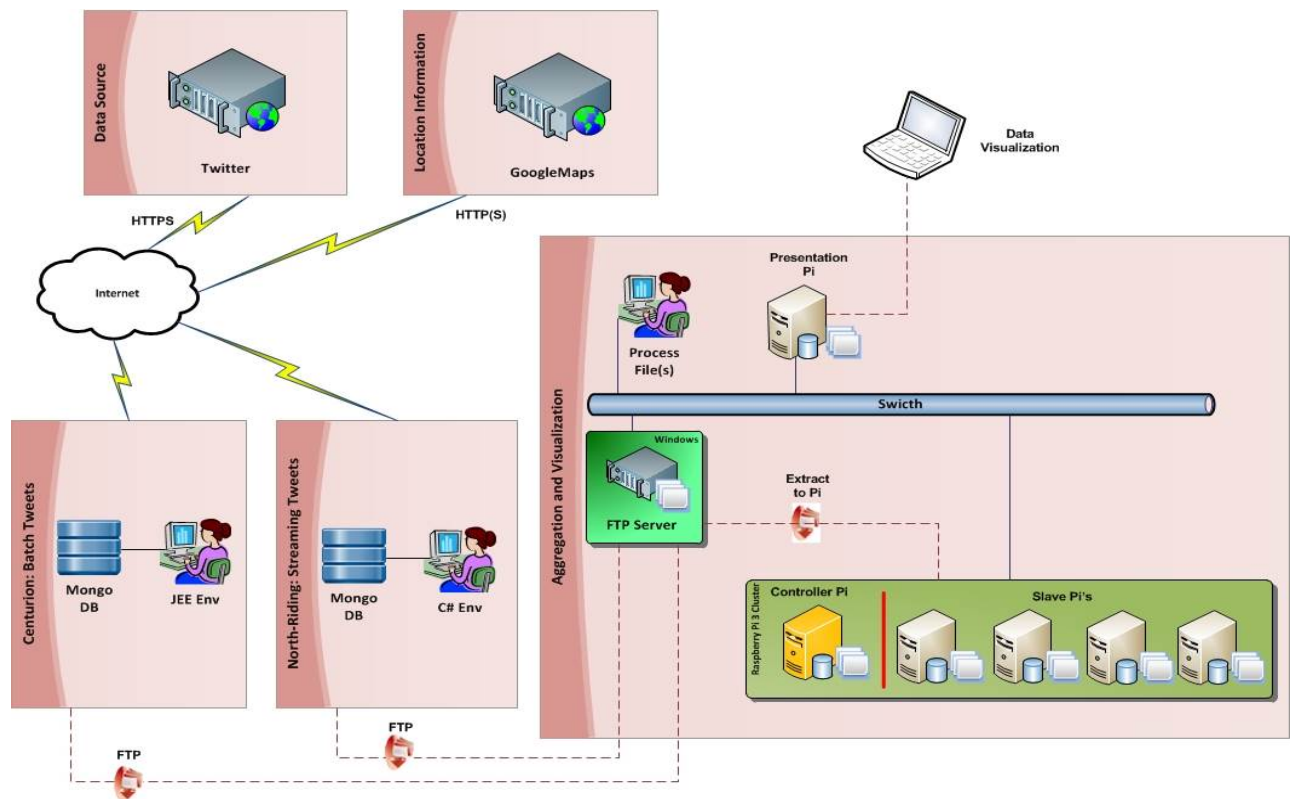

Figure 8: Operational Model: Phyical

19

8.3.1 Raspberry Pi Configuration Sheet

Node Name	IP Address	Role	OS	Software
node01	192.168.1.200	Master	Raspbian	Spark
node02	192.168.1.201	Slave	Raspbian	Spark
node03	192.168.1.202	Slave	Raspbian	Spark
node04	192.168.1.203	Slave	Raspbian	Spark
node05	192.168.1.204	Slave	Raspbian	Spark
dc-pi	192.168.1.101	Web Server	Raspbian	Node.JS

8.4 Testing Approach

The major components of the solution were developed independently with very little external real-time integration except for the parts where the solution integrated to external providers - Twitter and GoogleMaps.

The team decided on a testing approach the involved writing software to facilitate a form of automated testing as far as possible. Details of these testing activities can be found in individual reports.

Non-Functional testing was only focused around the Raspberry Pi cluster in order to establish how quickly it can process large sets of twitter data with limited hardware resources for the individual nodes.

8.5 Possible Extensions

The project used only Twitter as the Big Data source. According to Statista [7], in April 2016 Facebook ranked number one with 1 590 millions of active users, while Twitter recorded 320 millions of active users. Other social media networking platform ranked included Intagram, WhatsApp, WeChat and LinkedIn among others. Most the social networking service provides an API to retrieve social media messages or posts. The project chose Twitter as the data source because Twitters API seemed to be easier to work with. Due to lack of time only one data source was chosen. One of the possible extension in the project lies on the data collection component. The two sub-components can be enhanced to support other social

media platforms as the data providers.

Other potential extensions to the solution include developing a smart-phone/ tablet App to make the same sort of analytics reports available to a category of user that is always on the move.

The system itself can in future be hooked into LDAP in order to better control user access and security from a central point as well as ease of provision for new users.

9   Conclusion

To conclude, Big Data is a resource that no longer needs to be confined to large enterprise processing or expensive computing facilities. There are means by which big data harvesting, transformation and visualisation can be performed using smaller scale endeavours, using open source software and low cost commodity hardware. This conclusion was reached by demonstrating the implementation of low cost hardware and open source software to build a system capable of sourcing tweet data from twitter, transforming the data and providing visualisations that help discern meaningful information about American and South African election trends.

References

[1] S. Madam. From Databases to Big Data. IEEE Computer Society, 2012.

[2] V. Kumar, R. Yuvaraj, C. Anusha. Effective Distribution of Large Scale Datasets Clustering Based on MapReduce. 2016.

[3] http://www.internetlivestats.com/twitter-statistics/

[4] Twitter, "Rate Limits: Chart", https://dev.twitter.com/rest/public/rate-limits, Twitter, San Francisco, Last accessed 29 June 2016.

[5] Google, "Google Maps Geocoding API Usage Limits", https://developers.google.com/maps/documentation/geocoding/usage-limits,Googe,San Francisco,Last accessed 29 June 2016.

[6] H. Makabee, "Divide-and-Conquer: Coping with Complexity", https://effectivesoftwaredesign.com/2011/06/06/divide-and-conquer-coping-with-complexity/, Last accessed 28 June 2016.

[7] Statista, "Leading social networks worldwide as of April 2016, ranked by number of active users (in millions)", http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/, Statista Inc, New York, Last accessed 29 June 2016

10 Appendices

10.1 Appendix A: Time Sheets

ELEN-7046 Group Project – Combined Time sheet			
Task	**Start Date**	**End Date**	**Duration**
Business Modelling			
- Research Technologies	20 April 2016	22 May 2016	32
- Business Case	24 April 2016	01 May 2016	7
Requirements			
- Use-Case Model	01 May 2016	08 May 2016	7
- POC / Prototyping	24 April 2016	22 May 2016	28
- Specifications	01 May 2016	22 May 2016	21
- Review	22 May 2016	22 May 2016	0
Analysis & Design			
- Software Architecture Description	15 May 2016	22 May 2016	7
- Component Design	15 May 2016	22 May 2016	7
Implementation			
- Development	22 May 2016	19 June 2016	28
Testing			
- Component/Unit Testing	22 May 2016	19 June 2016	28
Deployment			
- Demonstration (Description of current release / Roll-out the project to the market)	18 June 2016	22 June 2016	4
Supporting			
- Project Management	24 April 2016	01 July 2016	68
- Documentation Deliverable	26 June 2016	01 July 2016	5

Figure 9: Group Time Sheet
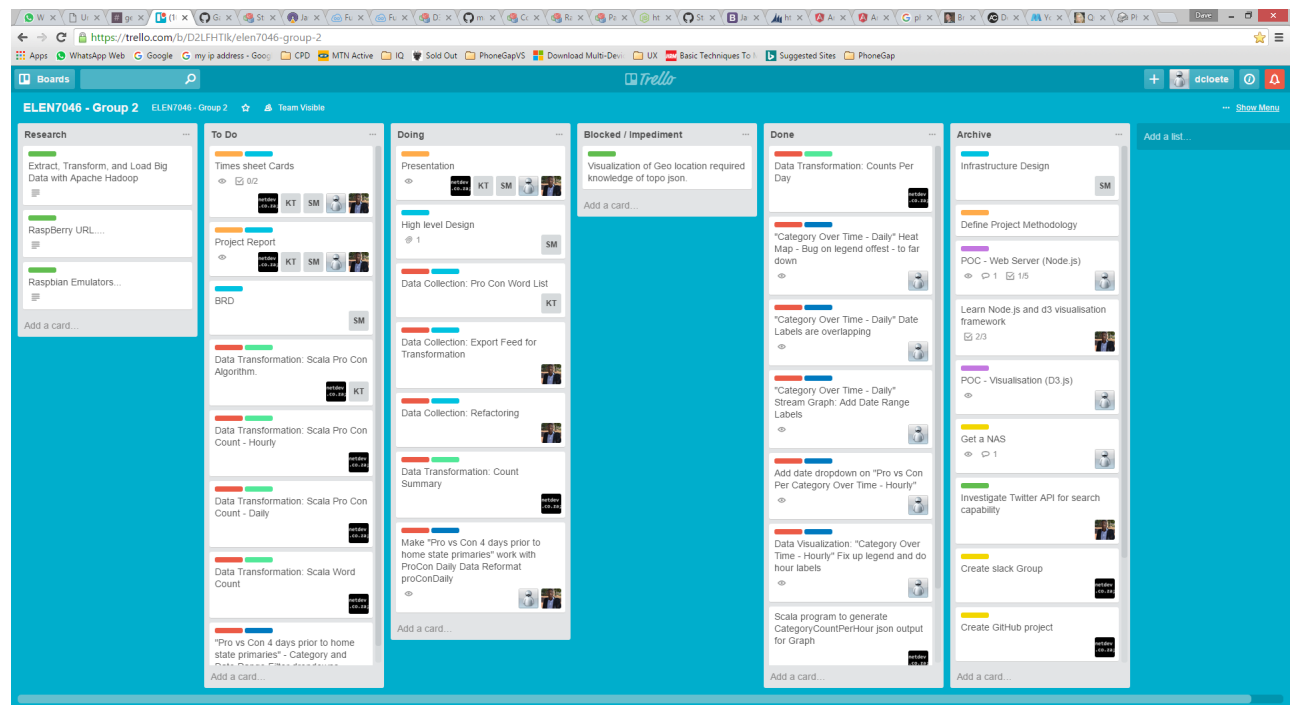
23

10.2   Appendix B: Trello Board

Figure 10: Trello Board: Project Management

10.3   Appendix C: Slack Dashboard

ELEN7046 - Group 2's Weekly Summary

Sunday, June 12th - Saturday, June 18th

Hope you are having a good weekend! Here's a summary of what happened on your team last week:

Your team sent a total of **244 messages** last week (that's 174 more than the week before). Of those, **100% were in public channels**. Your team also uploaded **27 files** (that's 27 more than the week before).

Figure 11: Slack: Team Collaboration

10.4   Appendix D: Git Repository

A comprehensive list of all the artefacts can be found on GitHub by following the link below:

https://github.com/garethstephenson/ELEN7046_Group2_2016

10.5 Appendix E: List of Technologies and Tools

Tool/ Technique	Usage Description
Trello	Mainly project task allocation to individual team members.
Github	Source control and documentation repository. Also used for online document editing and collaboration.
Slack	Used for online collaboration and communication.
WhatsApp	Used for daily communication with project team members.
Hangouts	Used for online meetings and video conferencing.
Rational Software Architect	Used for architecture deliverables such as Use Cases, Component Models and Sequence Diagrams.
MS Visio	Used for modeling infrastructure deliverables and visualization component model.
MS PowerPoint	Project presentation.
MS Word	Documentation of Individual Reports.
TeXstudio	Group Report construction and online collaboration.
.net C# Framework version 4.5.2	Object Oriented Language developed by Microsoft.
Visual Studio Community 2015	IDE for .net built by Microsoft.
TweetInvi version 0.9.14	Third Party Wrapper for Twitter.
Mongo DB	Database solution for Persistence.
Mongo DB Driver version 2.2.4	.net Driver for Mongo DB.
Apache Spark 1.6.1	Cluster Computing Framework.
Scala 2.10	Hybrid functional and object oriented programming language.
SBT package manager	Package manager.
JAVA 8 SDK	Java software development kit
IntellijIDE 2016	Integrated Development Environment.
Raspbian Linux	Raspberry Pi Operating System.

Tool/ Technique	Usage Description
Node.js	Platform that allows for standalone JavaScript application
Node.js Express Framework	Package enabling web features that support MVC implementation.
Node.js - File System	Package enabling File I/O.
Node.js Twig.js	HTML templating engine to allow binding of data to the views.
Bootstrap	UI framework used to aid responsive UI design.
D3 (Data Driven Documents)	A Powerful data visualization component used in the translation of domain data to scalable vector graphics.
Java Enterprise Edition 7 - Jboss Wildfly 10	Development Framework.
Netbeans 8.1	Integrated Development Environment.
Apache Maven	Build Tool
Ubuntu Linux 15.10	Operating System for development and execution of the history component
Test Driven Development	Development technique