ECE496-Design Project Course:
Project Proposal Final Draft

Project Title: Applications for Intelligent Transport - History Timeline Reporting Engine
Team ID: 2016616

Supervisor: Alberto Leon Garcia & Ali Tizghadam
Section: 9, Nick Burgwin

Students: Shafaaf Hossain - 998891515
Joohyun Lee - 1000134632
Terry Shi - 999743201
Eric Deng - 999553772

Date: October 27th, 2016

Executive Summary

Connected Vehicles Smart Transportation (CVST) provides a platform for novel applications and innovations to improve the efficiency and safety of transportation systems. The system mines and stores a large variety of information related to traffic. However, given the sheer size of the data available, the current platform makes it challenging for users to rapidly draw conclusions from the data. This project's primary objective is to improve the CVST platform by providing a reporting web application to create visualizations of CVST in the form of charts and graphs.

Users cannot gather information at a glance from the current CVST system. Without a page to display any general information, a user currently must come to CVST with very specific requests to search for it. It is also difficult to analyze the historical data available, as the user would need to navigate to the location he is interested in and click on the nodes to generate past reports.

The proposed design will consist of two major components, a web app interface which allows users to select and filter data, and display graphs and a back-end server which will access the data within CVST and perform the needed data manipulation and aggregation. The data would be stored in Elasticsearch and retrieved by Apache Spark instances where data aggregation would be performed. The implementation of Spark would be setup through their Python interface in the backend component.

Functionality of the design will be tested by stepping through the report generation process for a variety of different data types and time ranges. The design is deemed fully functional when it produces the desired charts/graphs based on the user's inputs. The report generation process will also be timed to ensure it meets timing objectives.

The project is divided up between members based on interests and experience. To ensure the design is completed in a timely manner, the project is divided into tasks which will be worked on in parallel. The team will begin by setting up the front end in tandem with components of the back end server before completing the back end and integrating the two together.

Table of Contents

Project Description:

Background and Motivation 1
Project Goal 2
Project Requirements 3
Validation and Acceptance Tests 4

Technical Design:

Possible Solutions and Design Alternative 5
System-Level Overview 10
Module-Level Descriptions 11
Assessment of Proposed Solution 12

Work Plan:

Work Breakdown Structure and Gantt Chart 13
Financial Plan 17
Feasibility Assessment 19

Conclusion 20

References 21

Appendices 23

1. Project Description

1.1 Background and Motivation

Connected Vehicles Smart Transportation (CVST) is a platform that serves real-time and historic data related to transportation in the Greater Toronto Area (GTA). These data types consist the following: traffic incidents, road closure, information on TTC transportation, BIXI bikes, weather information, air quality, and traffic sensors. This data is gathered with a network of sensors provided by the university, government and industry and presented through the CVST portal[1].

TTC data is a core component of what the CVST offers. CVST has a comprehensive collection of TTC data due its integration with NextBus which provides services such as transit routing and report generation on data such as bus frequencies, route times and bus density maps.[1] This information can be used to evaluate the effectiveness of the TTC system. However, this fails to utilize one of the key advantages of a big data system like CVST: the ability to compare TTC data with all the other data types in CVST.

In the current CVST portal, data regarding traffic sensors, road closures, and weather are presented on a map as a large number of nodes. At a glance, these nodes provide the current data taken from these sensors. For some sensors, a graph of the data for the past week is provided. However, for most data types, CVST does not provide the ability to generate reports on historical data.[1] This severely hinders the usefulness of CVST for users looking to analyze trends in data such as traffic, road closures and Bixi. In addition, this functionality being missing reduces a user's ability identify correlations between data sets. These correlations include the impact of road closures on traffic, weather on traffic and closures on TTC.

Another shortcoming of CVST is that users cannot gather information at a glance. Upon entering the portal, a user is overwhelmed with a map of Toronto with a large quantity of nodes and several options on the side. Without a page to display any general information, a user currently must come to CVST with a very specific request and search for it on his own.

To improve the current CVST system, the project will begin by extending the report generating functionalities of the TTC data to all data types within CVST. Users will be able to sort and aggregate data by time and location to generate reports such as trending graphs on traffic and road closure frequencies in different areas. In addition, functionalities with regards to data integration between different data types will be provided to utilize the scope of CVST in its entirety. The design team also plans to include a way to display some general data so that users have somewhere to start before they explore the vast amount of data in CVST.

[1]Description of CVST and its capabilities is in Appendix A

These improvements will hopefully improve CVST's purpose as a feedback mechanism for the transit system within the GTA. Users such as city planners or transit officials will be able to use CVST in their decision making process.

1.2 Project Goal

The goal of this project is to implement a comprehensive system that allows for the detailed report generation for all the various data types in CVST. The report engine will provide users with a platform that can navigate the enormous collection of data hosted by CVST by using meaningful data aggregations and comparisons. Finally, the report system will present them in different visual formats including but not limited to bar, pie and line graphs.

1.3 Project Requirements

The following project requirements are listed based on importance.

1.3.1 *Functional Requirements:*

- ➢ The system shall allow users to filter the data within CVST based on time and location.
- ➢ The system shall allow users to select the output format of the data visualization as either line graphs, pie charts, bar graphs, or scatter plots.
- ➢ The system shall access data within the CVST database.
- ➢ The system shall perform data manipulation such as sorting and aggregation based on the the user's inputs and preferences.
- ➢ The system shall produce visualization of CVST data in the form specified by the user.
- ➢ The system shall be integrated into the existing CVST portal.

1.3.2 *Constraints:*

- ➢ The system must work with all data types[1] within the scope of CVST.
- ➢ Any third party software must be open source to allow modification of source code for specific tasks in the future.
- ➢ The data aggregation and analysis presented must be accurate. Any uncertainties and assumptions, particularly with predictive analysis, must be clearly indicated.

3.3 *Objectives:*

- ➢ Analysis of data must be fast to ensure parallelism with real life events. Report generation should complete in 10s, otherwise indicators will be used to notify progress. [2]
- ➢ The reporting engine should be scalable for changes with CVST. As CVST grows, new data types as well as the its hardware system will increase. Integration of new data types should be as simple as possible.
- ➢ The CVST reporting engine should be fault tolerant. The engine should not contain any single point of failure and faults would be isolated by having a modular design and not allowing them to be propagated to other components.
- ➢ The system should provide a scheduling mechanism for users to automatically receive generated reports at set intervals.
- ➢ The system should be easy to use and configure. Evaluation of this will be done in accordance to the usability requirements outlined in ISO/IEC 9126 [3] and ISO/IEC 25022. [4]

[1]A list of all data types currently in CVST is provided in Appendix B.

1.4 Validation and Acceptance Tests

Testing and verification will be conducted to check if the design meets all of the functional requirements and constraints. In addition, the effectiveness of the design will be measured using the following criteria.

- ➢ Functionality:
  - Use a large variety of input options and requests sizes to simulate real user usage.
  - Test sorting and aggregation functionality against pre-computed results to ensure numerical correctness.
  - Test numerical values inputted into the graphing module will output the expected graphs.
  - Test to ensure scheduled reports will be delivered on time and correctly.

- ➢ Performance:
  - Measure the time it takes to retrieve data in the back-end over timespan of years, months and weeks with all data types.
  - Measure the time it takes to generate graphs with all types of data aggregation. Results should be no more than three seconds as not be the bottleneck in the system.
  - Record end-to-end transaction time from when users requesting data by submitting inputs, to when output of data visualizations are generated for users to view. The total transaction time should be under 10 seconds. [2]

- ➢ Usability:
  - Recruit users of various technical backgrounds to make various report requests.
  - Survey user experience from test users.
  - Evaluated with metrics, such as completion rate, number of errors and task times, covered in ISO/IEC 9126 [3].
  - Assess the quality of the system using Software Usability Measurement Inventory. [5]

2. Technical Design

2. 1 Possible Solutions and Design Alternatives

The design of the system can take many forms. From the overall structure to specific software including the database and the visualization, many components of the system have a variety of alternatives to choose from, each with their own advantages and disadvantages. System inputs and outputs can be through a web application for easy user access, or through a Unix shell for better integration with CVST. The following table includes the possible design alternatives considered and the associated advantages/disadvantages.

Table 1: Pros and Cons of Various Design Options

Design options	**Description**	**Advantages**	**Disadvantages**	**Decision**
Modular system	• Separate application from the CVST running its own database of aggregated (monthly periods) and pre-computed data. • Pulls data from the CVST API only when new data is required.	• Independence from CVST can allow the design to be run on different machines and be scaled easily. • Faults or crashes in the application would not affect the CVST system. • Storing aggregated and pre-computed data can provide faster access to commonly used data.	• Large setup time to create aggregated data as well as when new data needs to be synced with CVST. • Too many separate components may increase the processing time for certain layers as more connections and API calls need to be made.	CVST has to be always running and so making the design separate ensures it is not affected from faults. In addition, the network latency from all of the API calls can be improved by adding in more server nodes. Therefore it is chosen.
Embedded in CVST	• New functionalities (i.e. graphing, data aggregation) will be added directly into the existing CVST system.	• Direct access to all datasets, even detailed data (i.e. hourly datasets).	• Slow data aggregation without adding to the CVST database. • Working with existing code can have pitfalls.	Technical failure in the design will also affect the CVST portal; during process of integrating the design, CVST portal will have to be shut down for maintenance for several months

				during the development period. Therefore it is not chosen.
Input Platform				
Reporting Engine Web Application	• An extension of the current CVST portal. • Inputs will be collected through the browser.	• Access for all users with an internet connection.	• Possibly limited flexibility in user inputs.	Having a web application is always available for users on the web and more convenient than downloading a piece of software which constantly needs to be updated. Therefore it is chosen.
Downloadable System Specific UI	• Software application with a specified UI.	• Possibly more flexibility. • Gives the option of user specified queries. • More reliable.	• Not as accessible to all users.	Integrating the design with existing CVST portal is critical and an isolated software application does not meet this requirement. Therefore it is not chosen.
Output Platform				
Embedded JavaScript	• Outputs will be embedded into a formatted HTML page. • The outputs will be viewed through a web browser.	• Large selection of existing software/libraries. • Accessible.	• Not easily stored or printer friendly.	Viewing results on web apps is more convenient than downloading a large file. Users can also use the application on different browser windows with different inputs in parallel and see different results.

				Therefore it is chosen.
Documents	• Formatted documents such as PDF or Excel files.	• Easier manipulation and storage.	• Not easily accessible. • Limited libraries or existing software.	Viewing a downloaded file is much less convenient as the user would have to download the file then navigate to it on his or her PC before opening the file, whereas the results from a web app would be displayed right away. Therefore it is not chosen.
Database				
Elasticsearch	Distributed real time search database with no fixed schema for data.	• Provides customized search features for big data such as stemming and faceted search. [6] • Easily scalable through database sharding and replication.	• Does not support concurrent access methods such as transactional memory. [7]	Scalability is a major concern as more data sets would be added in the future. Elasticsearch can easily accommodate the new data. Users would mostly be reading data and so concurrent accesses to data is not a big concern. Therefore it is chosen.
MySQL	Relational database management system with enforced	• Easy to use and supports most backend programming languages.	• Poor horizontal scalability. Can only be scaled vertically by increasing CPU	If any additional fields are included in any of the data sets in the CVST, the

	schema.	• Used widely in many software applications and has a large community of developers. [8]	speed, RAM, cache size. [8]	SQL schema would need to be modified. Furthermore, SQL is very weak in terms of parallel programming. Therefore it is not chosen.

Data Visualization Tool				
D3.js	• JavaScript library for manipulating documents based on data using HTML and CSS.	• Easy to use and flexible to manipulate any part of the Document Object Model in the user's browser to make any visualization. [12]	• Very Specific. Can result in a large amount of code.	D3 has flexibility, visual appeal and ability to display trends, correlations and patterns in any format the developer wishes to do so. Therefore it is chosen.
Flare	• ActionScript library for creating visualizations that run in the Adobe Flash Player.	• Supports a variety of additional functions like visual encoding, animation, and interaction techniques. [9]	• Requires additional plugins in the user's browser. [9]	Since CVST deals with multiple large quantities of data sets related to weather, TTC, etc. there may be certain specific charts needed to highlight aggregation. These may not be possible to make using Flare as it does not provide that level of flexibility in design.

				Therefore it is not chosen.

General Overview Layout				
Timeline	• Shows data trend for each data type, where the time span can be defined by the user; the default time span can be predefined	• Can provide general information for all the different data types. • Allows the user to adjust the settings to see information that they are interested in.	• Default values may not be useful to all users.	Provides a simplistic visual overview. Users can quickly look at interesting data patterns even before deciding to search. It helps users to make smarter and better search decisions. Therefore it is chosen.
Dashboard	• Page with several “widgets” that show different metrics within CVST that the user can choose to display on the page. The default widgets are pre defined.	• Customizable to show various metrics for different users.	• Requires log in for customization. Therefore user profiles and account management need to be incorporated into our design.	The dashboard may contain too much complexity and not a comprehensive visual overview. It requires excessive effort from users to set up effectively. Therefore it is not chosen.

2.2 Initial Technical Design

2.2.1 System-Level Overview

The proposed design will feature two major components, a front end application to interface with the user and a back end server to access CVST and perform the needed data manipulation. The system is setup to follow Model-View-Controller and the server will follow RESTful design used in industry.

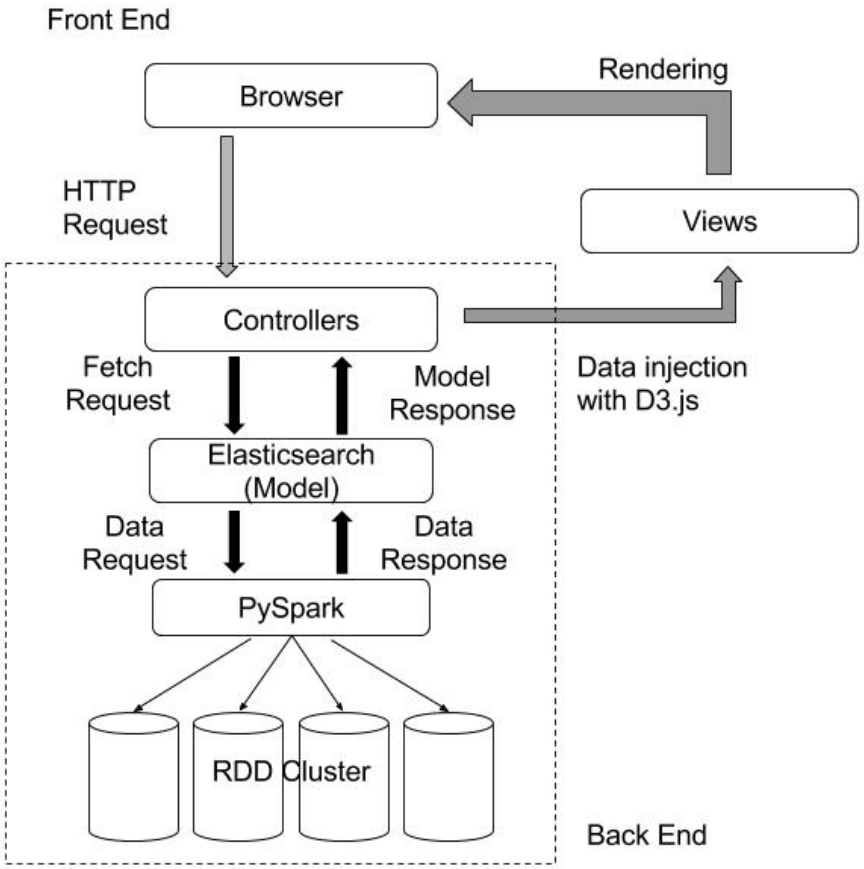


Figure 2. Block diagram of the proposed system design.

2.2.2 Module-Level Description

Web Front-End
Inputs: ● Input Parameters for: ○ Data Type (Traffic Sensors, TTC, Weather, Bixi or Air Quality) ○ Output Chart Type (Line, Pie, Bar) ● HTTP responses
Outputs: ● HTML files used to display web pages and generated reports ● HTTP requests
Function: The web application will be the interface between the report engine and the user. User inputs will allow users to filter the data they wish to generate reports for by data type, time and location. The front-end will use a combination of JQuery and D3 for handling DOM manipulation. D3 will be used to create the graphs/charts needed to visualize data and JQuery will be used to manipulate forms and input fields to handle user inputs.

Server Back-End
Inputs: ● HTTP requests
Outputs: ● HTTP responses
Functions: The Python back end will handle the HTTP requests received from the users. The router will map requests to specific controllers. These controllers handles the requests by requesting for the needed data. The data will be indexed into an elasticsearch database using a combination of PySpark, a python API for Spark, and Elastic-Hadoop. This data is stored inside the Elasticsearch database. The controllers will perform analysis on the data to find trends within the data and key performance indicators. This data is then forwarded to the front end through HTTP responses.

2.3 Assessment of Proposed Solution

The proposed report engine design provides a web application interface for users to customize the report generation process. Data accesses will done with Spark processes and visualization will be done using ElasticSearch combined with Kibana/D3.

Strengths:

- Accessibility: Web applications can be accessed across any platform with an internet connection. [10] Being a web app also means that it can be easily integrated with the CVST web portal.
- Speed: Data accesses, which is the major performance bottleneck, is faster using Spark compared to traditional Hadoop processes. [11] The inclusion of an Elasticsearch database also further improves performance by locally storing frequently accessed data.
- Flexibility: D3 is a very powerful platform and provides a large selection of charts it is able to generate. [12] Also, as a Javascript library, it will easily be integrated with the front end website.

Negatives:

- Complexity: Spark processes and D3 can be very complicated to setup and use. Spark is newer to industry compared to Hadoop and the documentation is not as extensive. We will however have mentors that can help us through the learning curve and give us advice on how we should be approaching challenges.
- Lack of working experience: Several of the team members do not have experience with using these technologies so there will need to be a learning period.

Compared to alternative solutions, the proposed design trades simplicity and ease of implementation for performance and flexibility. The addition complexity of the design poses a greater risk in completing functional modules. However, we feel these risks are worth taking to create the best possible product.

Work Plan

Table 2. Work Breakdown Structure

Task #	Task Description	Eric Deng	Joohyun Lee	Shafaaf Hossain	Terry Shi
	Research on Design Components				
1	Complete tutorial on D3.js	R			
2	Research Elasticsearch			R	
3	Research Spark and Hadoop				R
4	Familiarize with existing CVST API		R		
5	Investigate correlations between different data types		R		
6	Design types of useful data aggregation		R		
	Front-End Implementation				
7	Design page templates using BootStrap		R		
8	Create front page layout displaying timeline overview		R		
9	Create web navigation tool		R		
10	Handle user inputs using JQuery		R		
11	Route HTTP requests to appropriate server back-end handlers Handle user inputs using JQuery		R		
12	Design data report page to deliver to user				R
13	Create user input fields for report engine				R
14	Design layout to display charts				R
15	Create graphs and charts using D3.js				R
16	Create functionality to send scheduled data reports to users in file-attached emails provided by users				R
17	Create table to display data statistics				R
18	Test page load time				R

	Back-End Implementation				
19	Create data pipeline from CVST to Elasticsearch and Spark	R			
20	Develop API for each type of data aggregation	R			
21	Develop the controllers to create Spark worker processes	R			
22	Map query requests to specific controllers to retrieve data from Elasticsearch	R			
23	Develop API to search for key performance indicators in data analysis	R			
24	Write scripts to conduct scheduled data analysis as requested by user	R			
25	Receive user input from frontend			R	
26	Create route handlers to process HTTP requests from users			R	
27	Compile user inputs and fixed parameters into query			R	
28	Split frontend client requests into multiple jobs to be distributed to Spark processes			R	
29	Run batch processes weekly and monthly to find commonly accessed data			R	
30	Develop API to calculate statistics on data results			R	
31	Enable caching layer to store commonly accessed data			R	
32	Compile scheduled data reports into pdf and csv files to send to user's email			R	
	Assemble Design				
33	Construct bridge to deliver data between frontend and backend in JSON format				R
34	Connect all endpoints of APIs			R	
	Validation and Verification				
35	Test web user interface		R		

36	Test end-to-end transactions between APIs			R	
37	Test out accuracy and findings of analytics reports using dummy data		R		
38	Perform stress tests to measure performance of application under heavy load	R			
39	Test for very large data requests over long time spans				R
40	Debug and implement fixes for all defects	J	J	J	J
	Final Deployment				
41	Create access to web application from CVST portal	R			
42	Create access to web application from CVST homepage				R

The table breaks down the work structure of the project. An “R” indicates that the team member is responsible for the task, while an “J” indicates that it is a joint task done by multiple members.

Gantt Chart

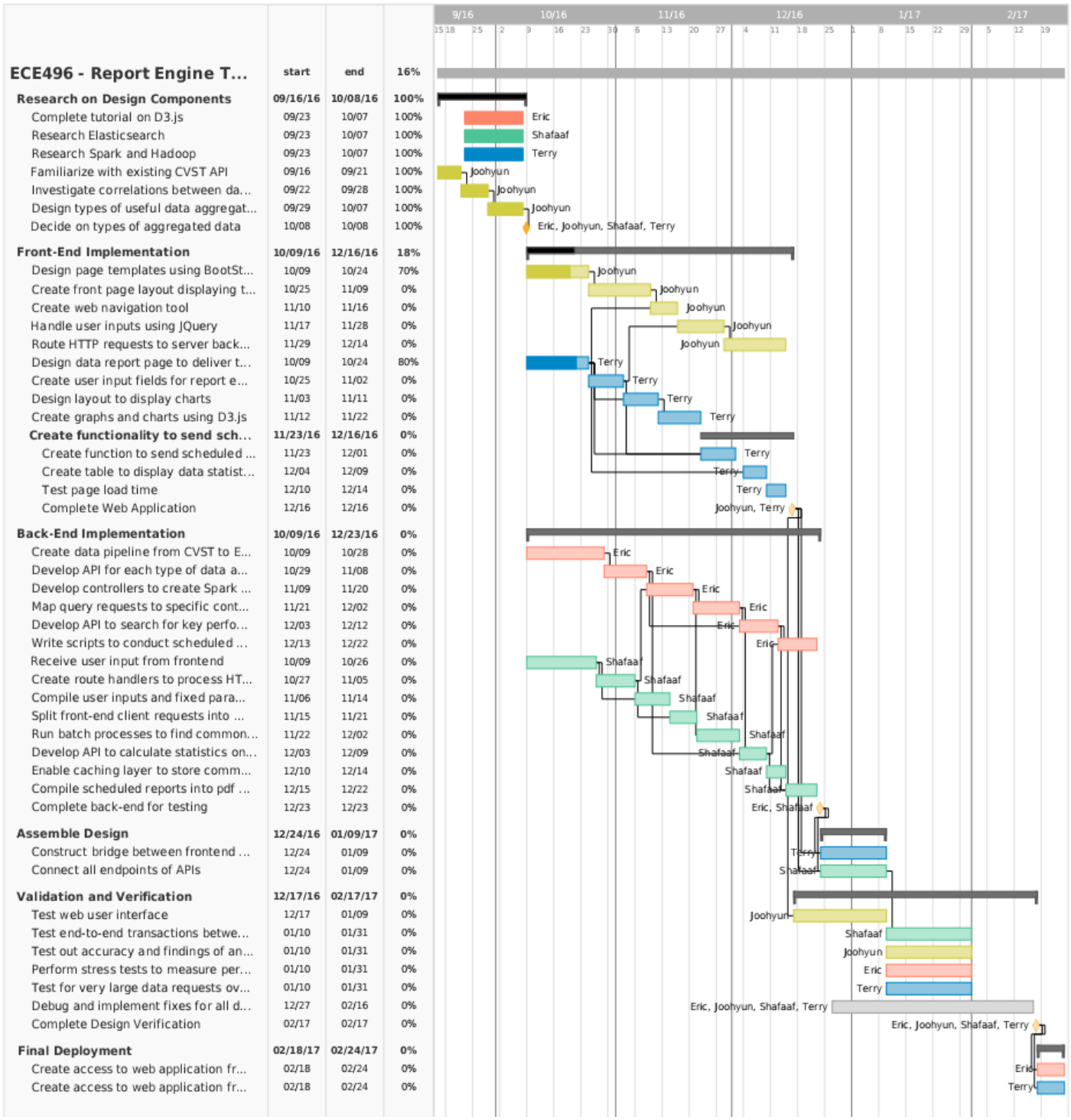


Figure 3. This Gantt chart shows our tentative schedule for project completion.

Financial Plan

The existing infrastructure that is currently being used to the the CVST system will be used to implement our system. However, should that not be possible, the cost for hosting the website and application will require additional funding laid out in the budget table.

Table 3 Project Budget

Student Labor				
Item	**Quantity (hrs)**	**Cost/Unit**	**Total Cost**	
Student 1	350	$26.44	$9,254.00	
Student 2	350	$26.44	$9,254.00	
Student 3	350	$26.44	$9,254.00	
Student 4	350	$26.44	$9,254.00	
Total Student Labor			**$37,016.00**	
Cloud Hosting Cost				
Item	**Priority**	**Quantity**	**Cost/Unit**	**Total Cost**
DB	1	1 Year	$1842.00	$1842.00
Computations	1	1 Year (2 hrs/day)	$0.266/Hour	$194.18
Developer Tools	2	~	$150.00	$150.00
Total Hosting Cost				**$2186.18**
Total Cost of Project				**$39,202.18**
Total Funding				**$500**

The project budget is made with the following assumptions:

- Hourly wages is derived from the average salary of entry level software developers [13]
- The ECE496 Project Proposal Guidelines assumed 50hrs of work will be put in per students for the entire Capstone project.[14] 70% of that is assumed to be time spent working on the project, as well as meetings whereas 30% of it accounts for the administration tasks, such as document writing.
- Hosting Costs are retrieved from AWS (Amazon Web Services). The database selected was db.m3.large which should accommodate a project of this size.
- Total funding currently available is from $100 per student as well as $100 from the supervisor. [15]

Feasibility Assessment

Skills and Resources:

- ➢ Experience in working with data mining and database systems
- ➢ Full stack web application development and web programming using JavaScript and Python
- ➢ Knowledge in using D3.js open source platform for data visualization
- ➢ Knowledge in using Apache Spark
- ➢ Possibly machine learning and artificial intelligence to predict future data trends and perform predictive analysis

Risk Assessment:

The primary risks associated with this project include:

1) The web page front end is unable to either unable to receive user inputs or connect to the backend.

   Solution: Functionality of the reporting engine will be presented using a defined set of input parameters in order to generate the required charts/graphs. Images of the output will be shown instead of having them served through the front end.

2) Spark/Hadoop proves to be too difficult/unsuitable to implement leaving the data accessing functionality defective.

   Solution: The report engine will be limited to a set of data which will be stored inside Elasticsearch. This data will be pulled from CVST and will be used to show limited functionality of the system.

3) The volume of data in CVST is too large for Spark processes to complete in a realistic time frame.

   Solution: Like the previous risk, the report engine would be loaded with pre aggregated data to reduce the processing time in order to show the functionality of the system.

4) Due to the large variety of data types in CVST, not all the data types can be accommodated in the application.

   Solution: Should this occur, the report engine will limit the user inputs to ones that are fully functional. The report engine will then aggregate those data types that are functional to produce requested data reports.

5) The existing CVST infrastructure is unable to host the report engine.

   Solution: Hosting the design on Amazon Web Services will cost an estimated $2,186.18. [15]

Conclusion

By the end of the development cycle, our project will be deployed onto the CVST portal. The application would be setup as a separate modular system and so it would incorporate new data coming in the future through the existing CVST APIs. New applications built on top of the CVST can easily incorporate our reporting application into their design to aid them in searching for specific data.

We hope that the added report generation and analysis functionalities will achieve our goal of improving CVST's capabilities as an analysis tool. Users such as city planners or transit officials will be able to use CVST in their decision making process. They may also see specific results based on their decisions and so improve on their future planning.

References

[1] "Connected Vehicles And Smart Transportation". Cvst.ca. N.p., 2016. Web. 8 Sept. 2016.

[2] Nielsen, Jakob, "Response Time: The 3 Important Limits", [Online] Available: https://www.nngroup.com/articles/response-times-3-important-limits/ [Accessed: 8 Sept. 2016].

[3] ISO/IEC 9126-1:2001, Software Engineering -- Product quality -- Part1: Quality model

[4] ISO/IEC 25010:2011, C.2 Software Quality Measurement

[5] Kirahowski, J, "The Use of Questionnair Methods for Usability Assessment", [Online] Available: http://sumi.uxp.ie/about/sumipapp.html, [Accessed: 15 Oct. 2016]

[6] Chavi Gupta, "Advantages Of Elastic Search", [Online] Available: http://www.3pillarglobal.com/insights/advantages-of-elastic-search [Accessed: 25 October 2016]

[7] "Solving Concurrency Issues", [Online] Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/concurrency-solutions.html [Accessed: 25 October 2016]

[8] John Mack, "Five Advantages & Disadvantages Of MySQL", [Online] Available: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/ [Accessed: 25 October 2016]

[9] "Flare Data Visualization For The Web", [Online] Available: http://flare.prefuse.org/tutorial [Accessed: 25 October 2016]

[10] Miller, Michael, "Are You Ready for Computing in the Cloud," in Cloud Computing: Web-Based Applications that change the way you work and collaborate online. Que Publishing, 2008, pp. 24-25.

[11] Armbrust, Michael, et al, "Scaling Spark in the Real World: Performance and Usability" [Online], Available: https://cs.stanford.edu/~matei/papers/2015/vldb_spark.pdf [Accessed: 15 Oct. 2016].

[12] Bostock, Mike. "D3.Js - Data-Driven Documents". D3js.org, [Online] Available: https://d3js.org/. [Accessed: 21 Sept. 2016].

[13] Bouwkamp, Katie, "What Salary Can you Expect As An Entry Level Software Developer?", [Online] Available: http://www.codingdojo.com/blog/entry-level-software-developer-salary/ [Accessed: 17 Oct. 2016]

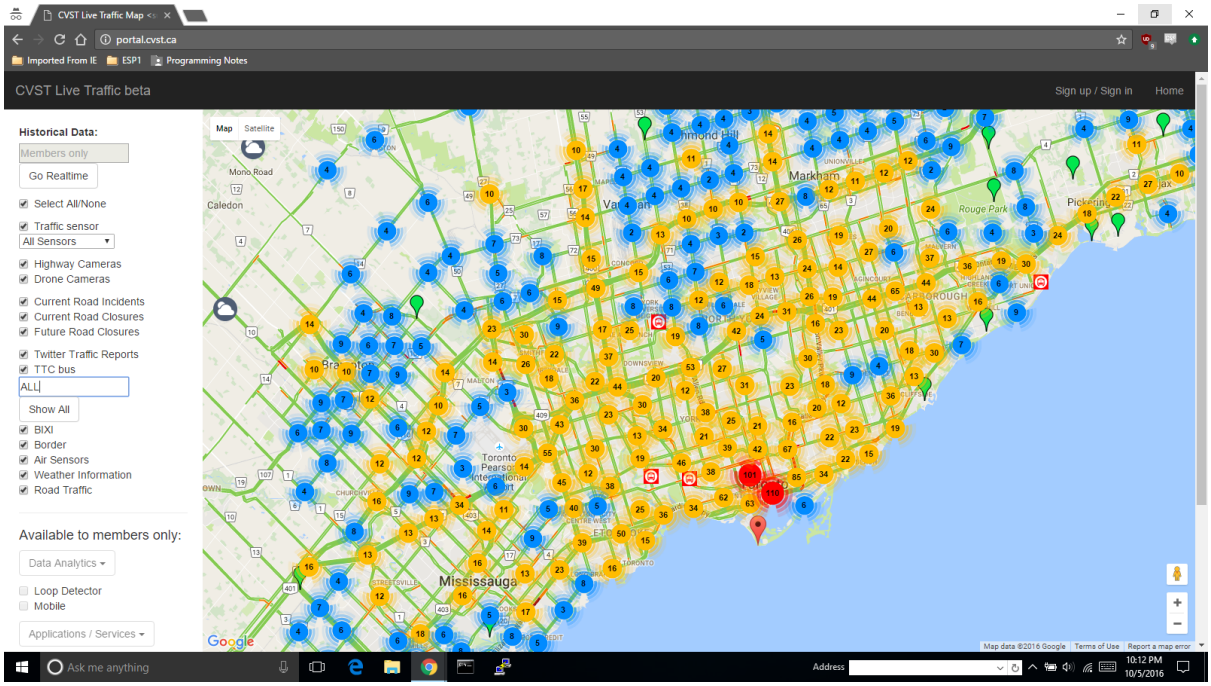
[14] ECE496, “Project Proposal Guidelines”, [Online] Available: https://internal.ece.toronto.edu/ece496.1516/pages/guides/ProjectProposalGuidelines.pdf [Accessed: 22 Sept 2016]

[15] “Amazon RDS Pricing”, [Online] Available: https://aws.amazon.com/rds/pricing/ [Accessed: 26 Oct 2016]

Appendix A: General Overview of Current CVST Portal

The CVST portal is the web application platform for displaying the data contained within CVST.

Diagram A.1

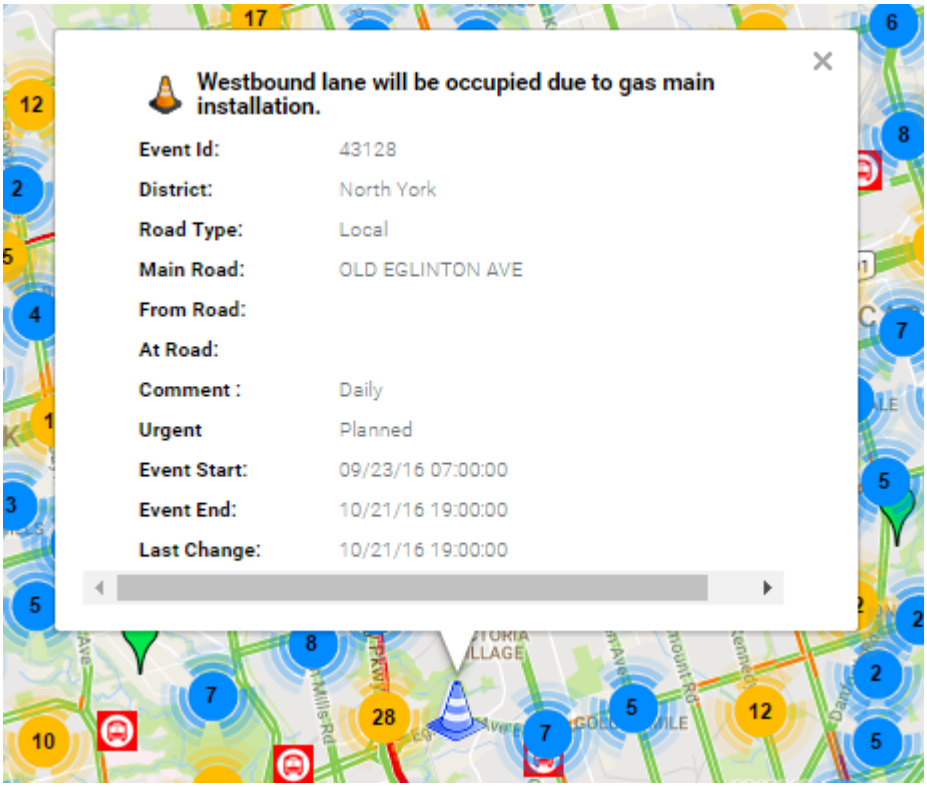


This is a screenshot of the CVST portal. It displays its data on top of a map of the GTA. The blue, red and yellow circles indicate current road traffic. The green balloons show traffic sensor data for major roadways. Clouds indicate weather sensors and red bus signs indicate current bus locations.

Diagram A.2

This is the information that is brought up when one of the traffic sensor nodes are selected. It displays information about the sensor as well as the data collected within a set period.

Diagram A.3



This is the information displayed for road closures.

Appendix B: CVST data types

The following is a list of the available data types on the CVST portal:

- ➢ Traffic Sensor
- ➢ Highway Cameras
- ➢ Drone Cameras
- ➢ Road Incidents
- ➢ Road Closures
- ➢ Future Road Closures
- ➢ Twitter Traffic Reports
- ➢ TTC bus (Nextbus)
- ➢ BIXI
- ➢ Air Sensors
- ➢ Weather
- ➢ Road Traffic