ECE496-Design Project Course:
Project Proposal Draft B

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 6th, 2016

Executive Summary

Connected Vehicles Smart Transportation (CVST) exists as a system that provides a platform for novel applications and innovations to improve the efficiency and safety of transportation systems. The system, in it's current state, 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 so that users can easily find and use the information they require. To achieve this, we will first analyse the data in order to provide a clear visualization of it. To begin, we intent to start with a smaller, simpler subset of the CVST data to provide a functional base before expanding to include the rest of CVST.

Table of Contents

Project Description

Background and Motivation

Connected Vehicles Smart Transportation (CVST) is a platform that serves as a real-time database system. It is an open API that can be used to develop intelligent applications for transportation. The system mines and stores a variety of information gathered through its network of sensors. This data is presented through the CVST portal.

In the current CVST portal, limited live data is presented and only a few datasets have the capabilities of generating graphs for past data (See Appendix A for description of CVST portal capabilities). Users are not able to access data from past months and years or select the time intervals. In addition, the CVST portal only supports aggregation that are time based for each of its sensors. It is very difficult to aggregate data by location since the sensors appear as nodes on the CVST portal.

Our project will primarily focus on solving this gap in the current CVST system; by implementing a reporting engine component. The report engine will provide functionalities for data manipulation and data visualization to improve a user's ability to draw meaningful conclusions from the CVST system. We hope that these added functionalities to the current CVST system will allow its users to make key decisions in improving the transportation system in Toronto.

Project Goal

Implement a comprehensive history timeline report engine for the large collection of data within the CVST system. The engine will aid users in drawing meaningful conclusions by identifying, performing analytics, analyzing key parameters of historical and real-time datasets and presenting them in a visual format.

Project Requirements

Functional Requirements:

- ➢ The system shall accept a variety of input parameters. These parameters will be used to configure the input data sets and the type of graphs being generated (i.e pie, line, histograms etc).
- ➢ The user shall be able to specify the format of the output and how they want to export the data..
- ➢ The system shall access and/or store data within the CVST database through the existing CVST APIs.
- ➢ The system shall be user friendly and be easy to use and configure.
- ➢ The system shall perform data manipulation such as sorting and aggregating of specific, as well as, general data based on the the user's inputs and preferences. Outliers and anomalies should be clearly highlighted.

Constraints:

- ➢ The data aggregation and analysis presented must be accurate within 90%. Any uncertainties and assumptions, particularly with predictive analysis, must be clearly indicated.
- ➢ The system must work with all datasets within the scope of CVST. See Appendix B for a list of all datasets.
- ➢ Any third party software must be open source to allow modification of source code for specific tasks in the future.

Objectives:

- ➢ Analysis of current data must be fast to ensure parallelism with real life events. The average response time should be around 1-2 seconds. Depending on whether the user requests data sets that are precomputed the latency may increase or decrease.
- ➢ Enhance user experience by designing data representations with clarity in format and visual appeal, and also form an intuitive GUI design for users with all levels of technology proficiency.
- ➢ The system should incorporate new data coming in for future use through the use of new APIs.
- ➢ The reporting engine should be scalable for future data coming in. As CVST would be using data that is constantly increasing, users should be able to dynamically add and configure servers and database instances to obtain more computing power.
- ➢ 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.

- In addition, due to the evolutionary nature of software applications, these are a list of additional features that will be focused in the system's implementation. Listed in order of priority:
    - General timeline of overall data.
    - Scheduled report generation in addition to ad hoc markups.
    - Predictive analysis generation

Validation and Acceptance Tests

In terms of testing, there will be several stages varying with the progress of the application development. The following is in order of increasing application completion.

- Unit Test - This test ensures that all individual components of the source code are functionally correct in isolation. These unit tests will be written by the component's respective developer who created the specific module.

- Integration Test - The test verifies if each component correctly works/communicates with its adjacent components. The correctness of test results confirms that all pipelines of the architecture are functional, such as the data pipeline from CVST API to the backend component, collecting user variable inputs from frontend shell to data search algorithm, transferring data analysis to data visualization module, and delivering the visual model of data analysis from the backend to frontend.

- User Acceptance Test - The intention is to make sure the user experience is satisfactory. In this step, we will employ test users to use the system and give feedbacks with regards to whether or not the ergonomics and practicality in design are up to their standards.

For all stages of testing, we will execute both black-box and white-box methods:

- Black-box Method - The method verifies that each component produces correct outputs. This ranges from the basic outputs of unit-tested components to the accurate data visualization and data analysis content produced to end-users.
- White-box Method - This ensures all internal structures the software architecture are free of defects and unexpected behaviours. This includes using a fault injection method to ensure all possible failures can be handled, particularly in the instance that the user could input conflicting variables to output invalid data analysis. It also involves API testing, which directly tests APIs in isolation to determine if the end-to-end transactions demonstrate expected behaviors. Test automation scripts will be devised to save time, as this is mostly an iterative process.

Technical Design

Possible Solutions and Design Alternatives

The design of the system can take many forms. From the overall structure to the specific software, 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 alternatives considered and the associated advantages/disadvantages.

Table 1: Pros and Cons of Various Design Options.

	Description	**Advantages**	**Negatives**
Program Setup			
Modular System	• Separate application from the CVST running its own database of aggregated (monthly periods) and precomputed data. • Pulls data from the CVST API only when new data is required.	• Storing aggregated and precomputed data can provide faster access to commonly used data. • Independence from CVST and can run from different machines.	• Large setup time to create aggregated data as well as when data needs to be synced with CVST. • Difficulties with future compatibilities if new datasets are added to CVST.
Embedded in CVST	• New functionalities (i.e graphing, data aggregation) will be added to 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.
Input Platform			
Reporting Engine Web App	• 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
System Specific UI	• Software application with a specified UI.	• Possibly more flexibility. • Gives the option of user specified queries. • More reliable.	• Not as accessible.

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
Documents	• Formatted documents such as PDF or Excel files.	• Easier manipulation and storage	• Not easily accessible. • Limited libraries or existing software.

Decisions also need to be made about the following component of the system. However, the advantages/disadvantages of each choice are not as apparent as those previously listed. The guidelines used to to select the best alternative are listed as well as some of the possible choices.

- Data Analysis Algorithms - Algorithms selected for our data analysis engine will be based on ability to recognize critical/associated data, time/space complexity, and application versatility. The datasets used in the project will another factor in this consideration. Examples include:
  - K-means
  - MapReduce
  - SVM (Support Vector Machine)

- Data Visualization Platform - The visualization platform will be chosen based on its level of flexibility, and ergonomic design considerations such as visual appeal. Examples include, but are not limited to:
  - D3.js
  - ChartJS
  - ElasticSearch

Initial Technical Design

The proposed design will feature a web application for handling user inputs and outputs. Outputs will be in the form of embedded Javascript in HTML. The report engine manages a database for precomputed data and generates the Javascript files requested by the user. In the event that the user requires data not available in the precomputed dataset, the report engine communicates with the CVST system to fetch the required data. This is then computed to generate the required JS file which is then served by the web application.

System-Level Overview

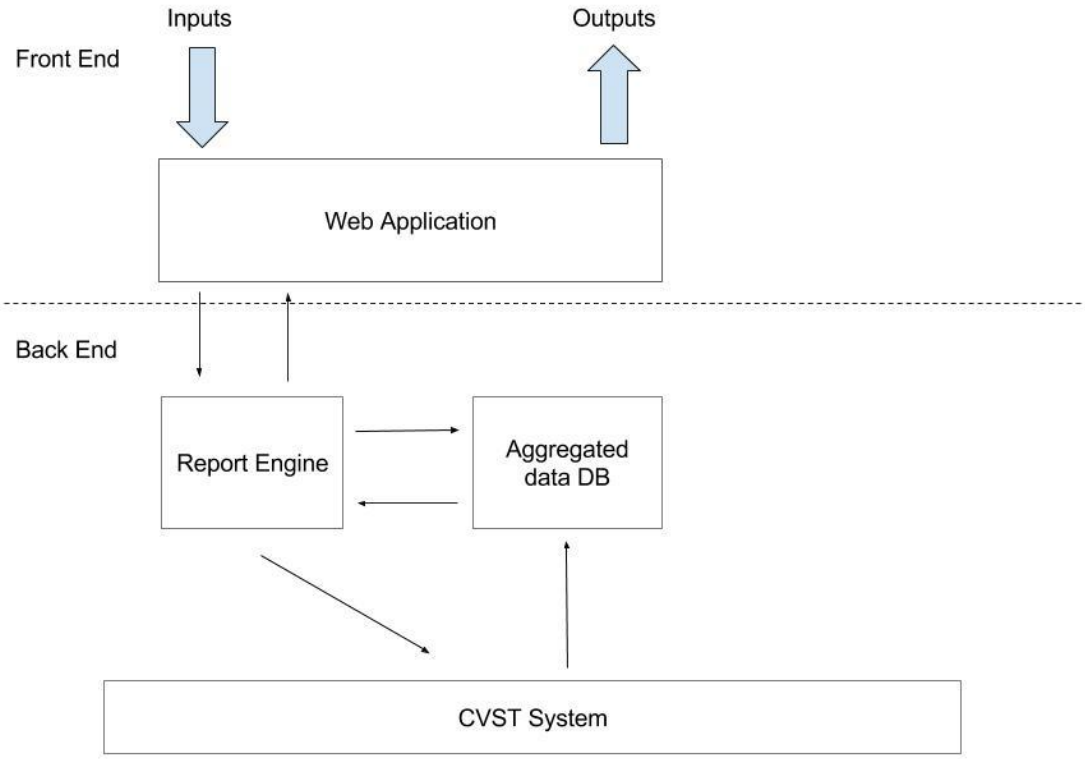


Figure 2. Block diagram of the proposed system design.

Assessment of Proposed Solution

Our decision to have the user input and outputs conducted through a web application was made with a few factors in mind. The ease of accessibility was a key advantage due to the existence of the CVST portal website. We felt that having a web based application closely tied to the CVST portal provides users an easier way to access both the current data presented on the portal and the historical data visualization and analysis provided by our system. Outputs will be through embedded javascript in HTML files due to the abundant selection of existing software platforms.

The report engine will be a modular program running outside of the CVST. Having a modular program provides advantages during the development cycle. Modification and deployment are more difficult if the system is intertwined with the CVST system. It also provides flexibility with the additional database system for precomputed data. Rather than fetching and computing data at every request, we would precompute and aggregate certain data sets which are commonly used to speed it up. We would still need to fetch the other uncommon data sets which are requested by the user, perform data aggregation and display the appropriate graphs. We would also keep track of the data sets being requested and if they are seen to be requested a lot they would be added to the precomputed data component.

Work Plan

Table 2. Work Breakdown Structure

Task #	Task Description	Eric	Joohyun	Shafaaf	Terry
	Design Proposal				
1	Research and familiarization of CVST	A	A	R	A
2	Design decision and planning	A	A	A	R
	Design Review and Approval				
3	Design review meeting	A	A	A	A
	Backend Implementation				
4	Data pipeline from CVST API to back-end			R	
5	Algorithm to search/analyze/predict data				R
6	Data visualization		R		
7	Test each backend functionality	A	A	A	A
	Frontend Implementation				
8	Interface to receive user input				R
9	Platform to present visual data output		R		
10	Test frontend shell	A	A	A	A
	Interim Report				
11	Gather powerpoint slide		R		
12	Configure presentation script				R
13	Conduct oral presentation	A	A	A	A
	Assemble Design				
14	Construct bridge between frontend and backend	R			
	Validation and Verification				

15	Design and run test for any failures	A	A	A	A
16	Debug and implement fixes	A	A	A	A
	Final Deployment				
17	Organize visuals for display	R			
18	Present final product at design fair	A	A	A	A

The table breaks down the work structure of the project. An “R” indicates the team member is responsible for the task, while an “A” indicates a team member is assisting in the task.

Gantt Chart

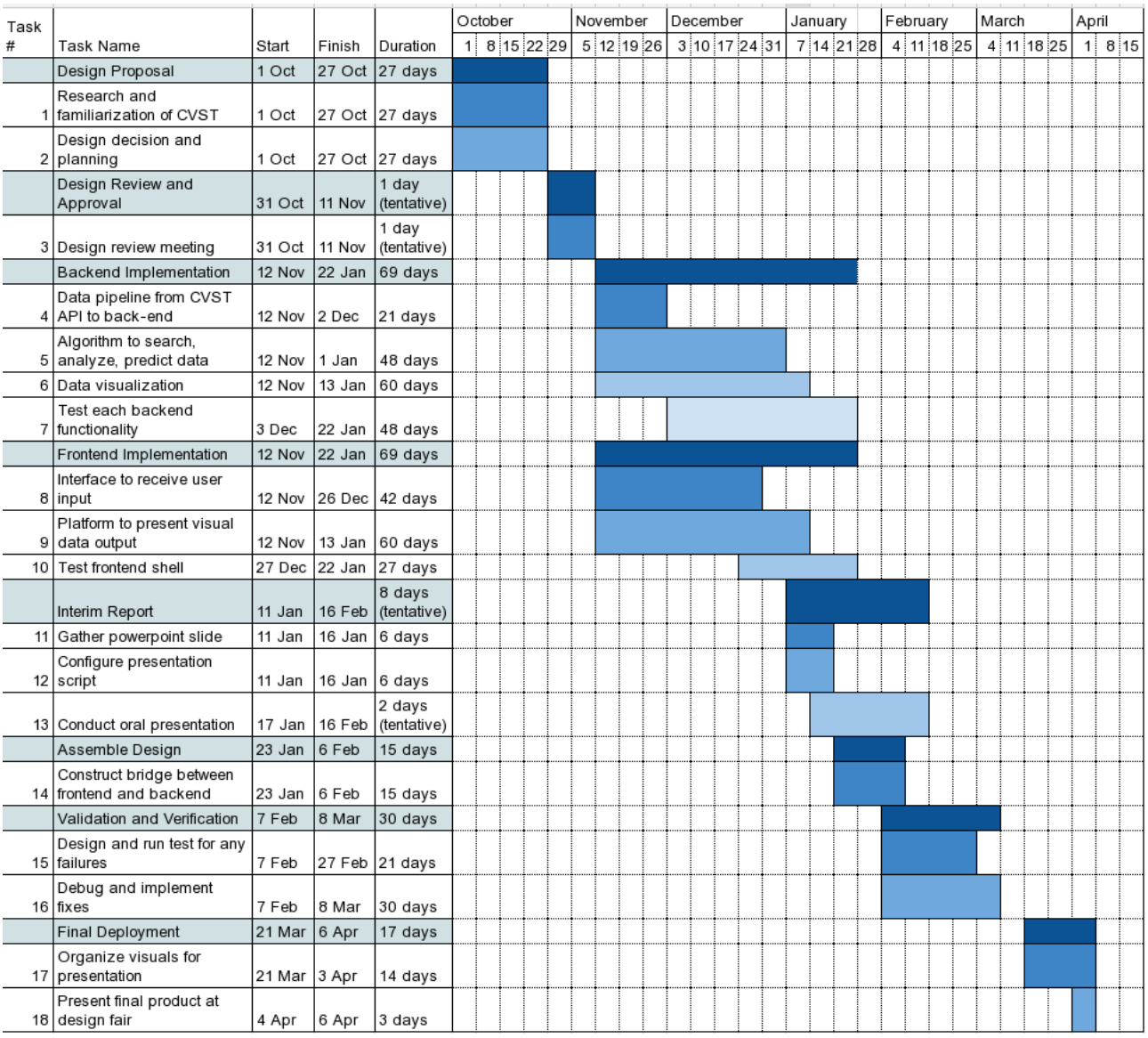


Figure 3. This Gantt chart shows our tentative schedule for project completion.

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. Included in Appendix C are the details of any assumptions made for budgeting.

Table 3. Project Budget

Student Labor				
Item	**Quantity (hrs)**	**Cost/Unit**	**Total Cost**	
Student 1	500	$11.25	$5,625.00	
Student 2	500	$11.25	$5,625.00	
Student 3	500	$11.25	$5,625.00	
Student 4	500	$11.25	$5,625.00	
Total Student Labor			**$22,500.00**	
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				**$24686.18**
Total Funding				**$500**

Feasibility Assessment

Skills and Resources:

- ➢ Knowledge in data mining and database systems
- ➢ Possibly machine learning and artificial intelligence to predict future data trends and perform predictive analysis
- ➢ Open source platform for data visualization, such as Pentaho, D3, BIRT, etc.
- ➢ Full stack web application development

Risk Assessment:

The core functionality of our project is data visualization of CVST data. Our project relies on being able to interact with the CVST API fetch the data which our reports are based on. A critical risk in this design is the of being unable to connect to the CVST API. Although unlikely, in the event of this type of failure, a portion of the CVST data will be loaded into our database manually. This way, our system will have a set of data to work with as a proof of concepts.

In addition, the scope of the CVST system is very large. It contains data from traffic sensors but as well as data from NextBus, Cameras and Drones. Working with those datasets can be difficult and time consuming. The primarily focus will initial be on a small set of data from traffic sensors, dated in the past 3 years, to ensure functionality before moving onto the more complex, larger datasets. Supplementary functionalities, scheduling, more complex charts/diagrams, will be added over the course of the development cycle.

References

M. K. Jiau, S. C. Huang, J. N. Hwang and A. V. Vasilakos, "Multimedia Services in Cloud-Based Vehicular Networks," in IEEE Intelligent Transportation Systems Magazine, vol. 7, no. 3, pp. 62-79, Fall 2015. ISSN 1939-1390

Lian Duan, Li Da Xu, “Business Intelligence for Enterprise System: A Survey” Industrial Informatics IEEE transactions on, pp. 670-687, 2012, ISSN 1551-3203.

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

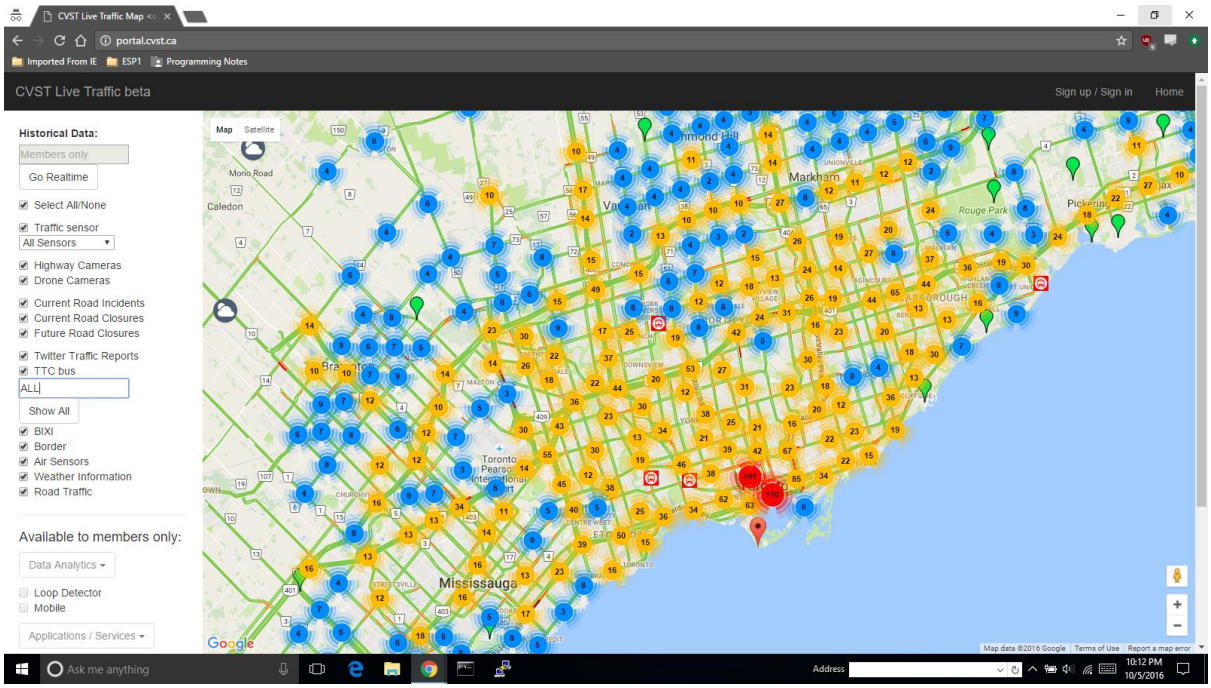
"Eclipse BIRT Project Home". eclipse.org/birt/. N.p., 2016. Web. 21 Sept. 2016.

Bostock, Mike. "D3.Js - Data-Driven Documents". D3js.org. N.p., 2016. Web. 21 Sept. 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

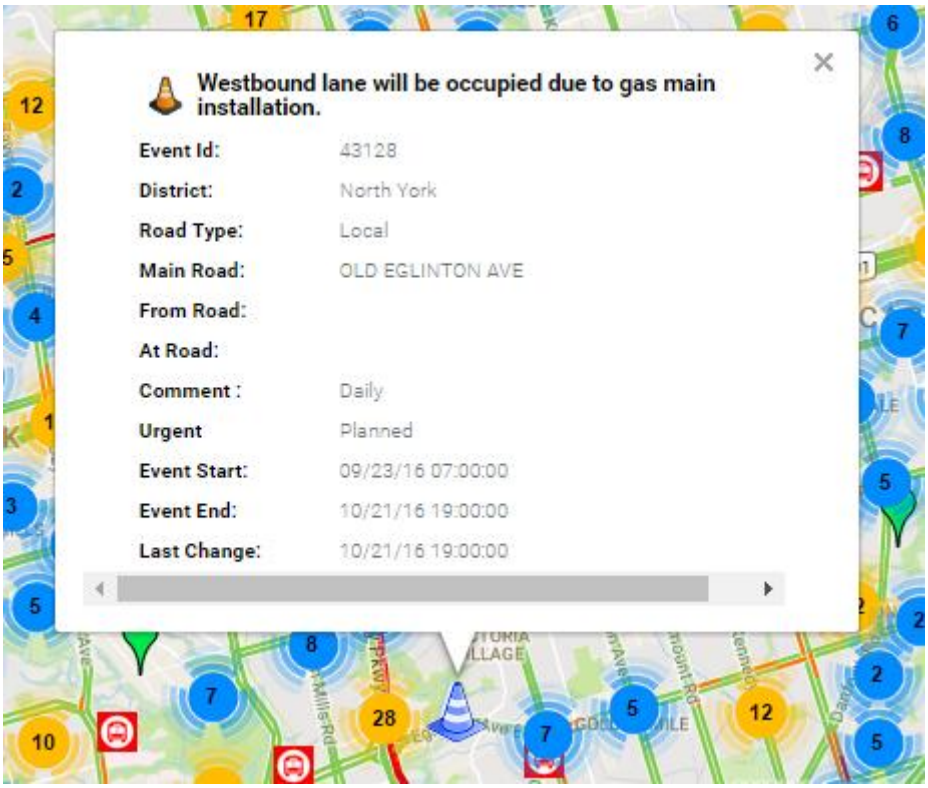


This is a screenshot of the CVST portal. It displays its data on top of a map of the GTA. The blue, red and yellow circles indicate current road traffic. The green balloons show traffic sensor data for major roadways. Clouds indicate weather sensors and red bus signs indicate current bus locations.

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

Appendix C: Budgeting Assumptions

The budget for the project are made with these following assumptions:

- ➢ Student wages uses Ontario's Minimum wage
- ➢ 500 Hrs per student was taken from the ECE496 Project Proposal Guidelines
- ➢ Hosting Costs are retrieved from AWS (Amazon Web Services)
- ➢ Database used was db.m3.large
- ➢ Total funding is from $100 per student as well as the $100 from Supervisor