ECE496-Design Project Course: Team Progress Report

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: January 17th, 2017

Table of Contents

Project Overview

Connected Vehicles Smart Transportation (CVST) provides a platform to improve the efficiency and safety of transportation systems. The system stores a large variety of information gathered from road cameras and traffic sensors. 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 report engine via web application to create visualizations of CVST data in the form of charts and graphs.

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 serve the data within CVST and perform the needed data manipulation and aggregation. The original cvst data would be stored in Hadoop Distributed File System (HDFS) clusters. Mapreduce will be conducted using Apache Spark in order to aggregate the data. This data will be indexed into elasticsearch. Upon user requests, the server will retrieve the data from elasticsearch and send it to the client. This data is then visualized using the amChart javascript framework. The implementation of Spark is being set up through their Python interface in the back-end 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. In addition, the components of the design will be tested using various frameworks. For the back-end implementation, a combination of cURL and python unit tests will be used. Front-end implementation will use Selenium.

During the implementation of the design, a few changes were made to the design. The original design called for storing the CVST data in elasticsearch with spark being runned on top of it. Instead, this has been changed after consulting with our technical advisor, Hamzeh Khazaei. A block diagram which outlines the changes made to the design is included in Appendix B.4.

The components of the design has been broken down and assigned to individuals based on personal interests as well as experiences. The four components, Spark, Elasticsearch, Back-end and Front-end, were divided amongst team members with each member in charge of the implementation of the components.

Due to the changes made to the design in regards to the back-end components involving Spark, a number of milestones has been delayed or modified. Changes were also made to the functionality of the design but functional objectives were not changed.

Team Progress Summary

Overview of Progress

The team is currently in development progress for both front-end and back-end implementation milestones. Projected completion date for front-end and back-end milestones is set to February 24th, upon which Stage 5: Final Design Assembly will take place. The table below lists out the key stages of all components within the milestones, and the latest status for each stage:

Key Stages	Progress To-Date
Stage 0: Planning and Design	Completed
Stage 1: Back End - Spark Implementation	In Progress
Stage 2: Back End - Elasticsearch Implementation	In Progress
Stage 3: Back End - Server Implementation	In Progress
Stage 4: Front End - Web Application Implementation	In Progress
Stage 5: Final Design Assembly	Not Started

Table 1: List of key stages in design progress within front-end and back-end design milestones, and their current progress

Summary of Changes

After meetings with our project advisor, Hamzeh, the interaction between Spark and Elasticsearch has been changed. Originally in the design, when a user requests data that is not currently residing in the elasticsearch cluster, a spark job would be created to fetch the required data. Instead spark jobs will be runned using a scheduler and the computed data will be stored inside elasticsearch. This caused delays in the implementation of spark as it required us to create our own version of CVST's HDFS (Hadoop Distributed File System). The diagram below represents the overview of modified system design, after applying the design changes in discussion.

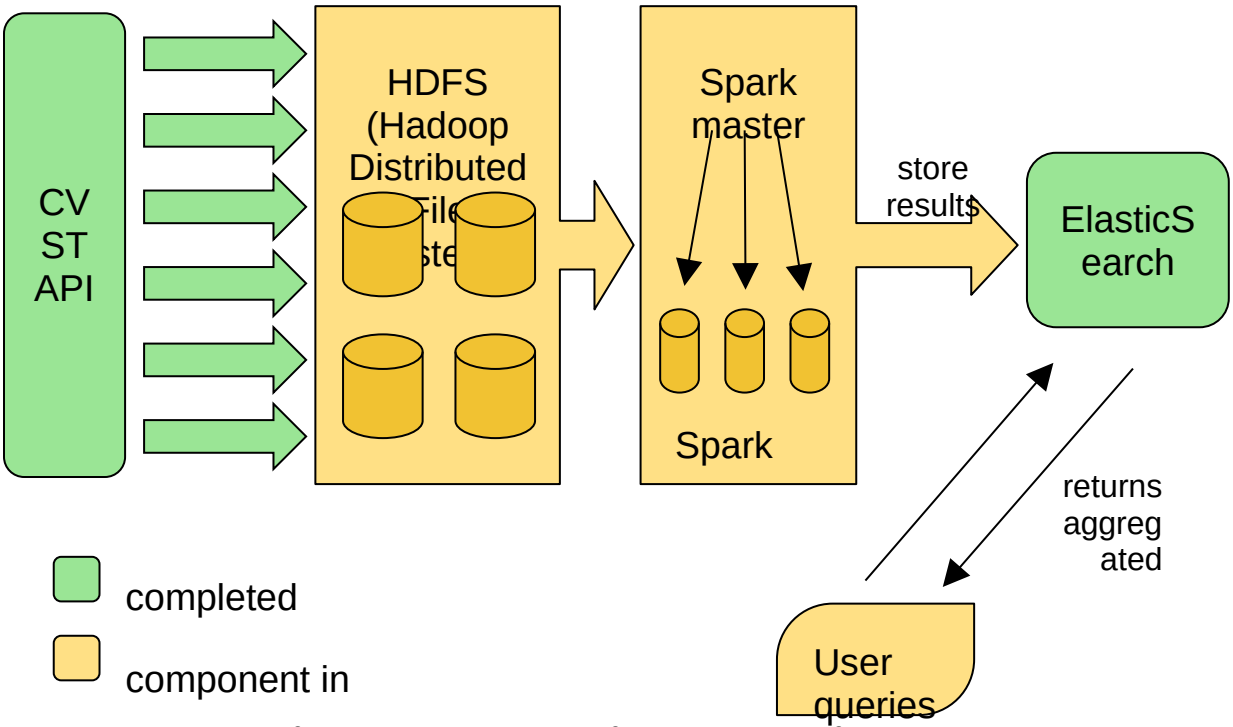

Diagram B.4-2: Modified overview diagram of system design after meeting with project advisor Hamzeh, and the progress of completing each design component

Initial implementation of D3.js proved too complex for the scope of the design. After internal consultation and research, an alternative tool named amCharts.js was chosen as the framework for the visualization component of the design. To note, amMaps library in amCharts.js was proven to be essential to visualize heatmaps and data aggregation over geographical area [2].

To resolve the delays caused by these issues and changes, our schedule (refer to Gantt chart in ***Appendix A***) has been updated to reflect our modified completion dates.

Summary of Progress

The completion of first milestone, which involved planning of the different data aggregation types to be visualized, resulted in a major setback to the initial Gantt chart schedule, as seen in the updated version of the Gantt chart in ***Appendix A***. Proposal for the list of data aggregation was drafted in an iterative process, until the list was approved by our supervisor. We were permitted to proceed with design implementation only after receiving the approval.

While there has been delays for the implementation of spark and the front end client, the server and elasticsearch implementation has been on schedule. We are currently able to query data from within the elasticsearch cluster successfully and serve that data to the client using a web server. The layouts and webpages for the website are also complete and only require minimal refactoring and redesign.

References

[1] "Connected Vehicles And Smart Transportation". [Online] Available: http://portal.cvst.ca/. [Accessed: 8 Sept. 2016]

[2] "JavaScript Maps - amCharts". [Online] Available: https://www.amcharts.com/javascript-maps/. [Accessed: 22 Dec. 2016]

Appendix A: Gantt Chart and Work Breakdown Structure

The updated Gantt chart reflects the schedule adjustments from the setback caused by the first milestone. Below are the two versions of Gantt chart, one proposed in the Project proposal, and the other modified upon completing the Team Progress Report.The two versions of Work Breakdown Structure is attached as well, due to the number of modified tasks that reflect the change in front-end and back-end design decisions.

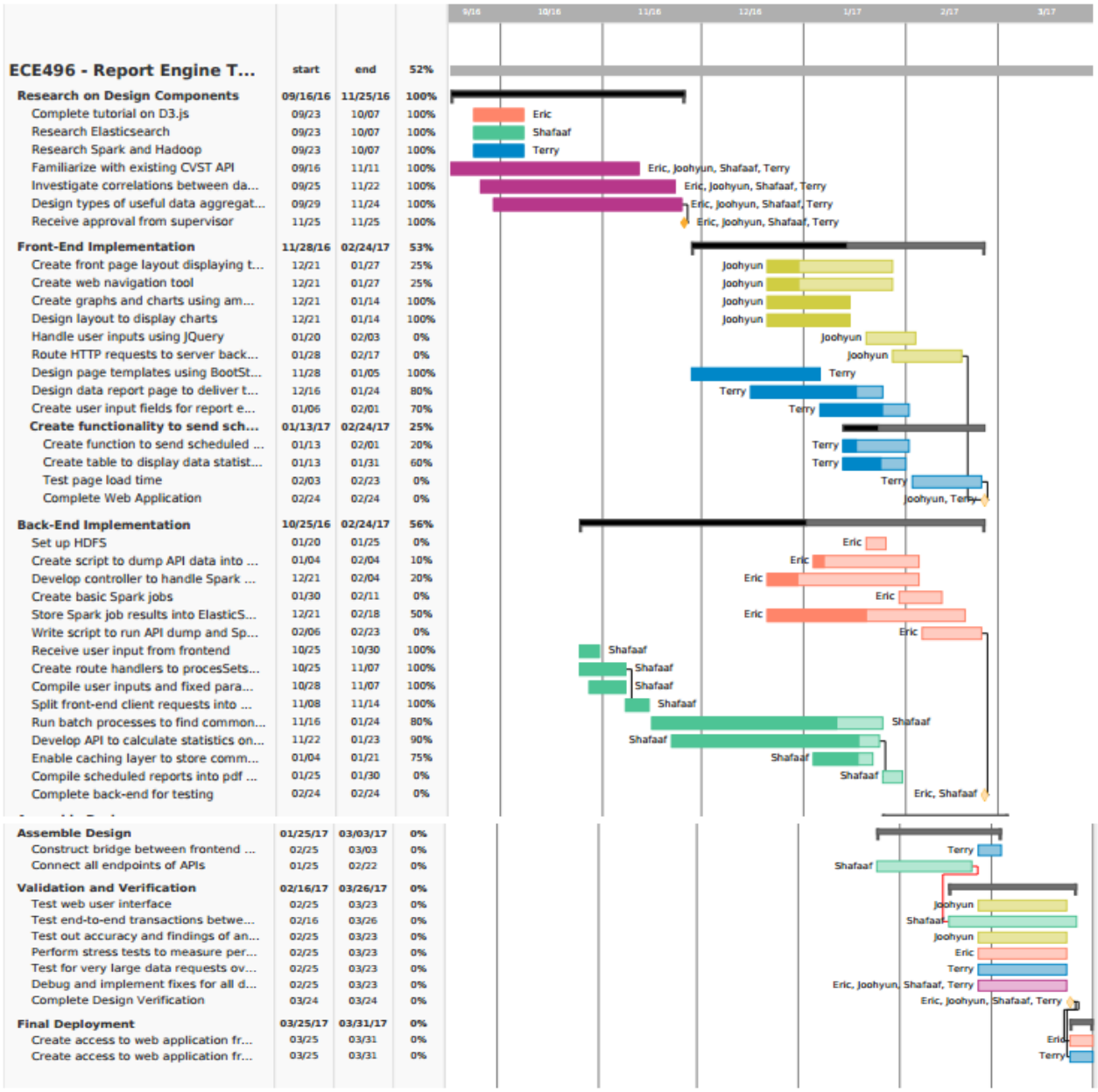

Figure A-1: Updated version of Gantt chart that displays the adjusted work plan schedule.

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	J	J	J	J
5	Investigate correlations between different data types	J	J	J	J
6	Design types of useful data aggregation	J	J	J	J
	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		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 times				R
	Back-End Implementation				
19	**Set up HDFS**	**R**			
20	**Create script to dump API data into HDFS**	**R**			
21	**Develop controller to handle Spark workers**	**R**			
22	**Create basic Spark jobs**	**R**			
23	**Store Spark job results into ElasticSearch**	**R**			
24	**Write scripts to run API dumps and Spark analytics**	**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

Table A-1: Updated Work Breakdown Structure that indicate modified tasks in bold font in highlighted row. An "R" indicates that the team member is primarily responsible for the task, while an "J" indicates that it is a joint task done by multiple members.

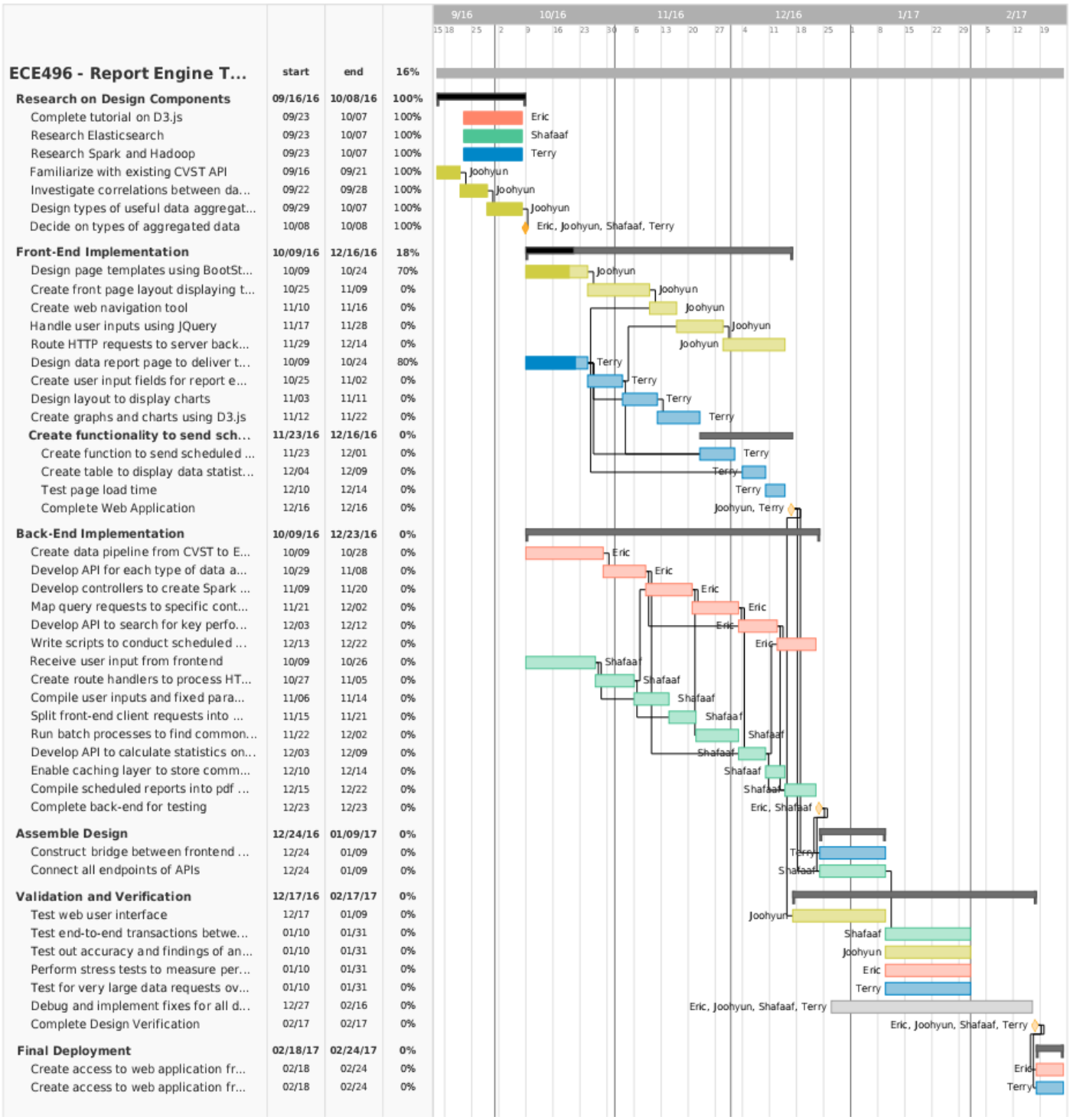

Figure A-2. Previous version of Gantt chart that shows original work plan schedule from the Project Proposal document.

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		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

Table A-2: Original Work Breakdown Structure proposed in the Project Proposal document. An "R" indicates that the team member is primarily responsible for the task, while an "J" indicates that it is a joint task done by multiple members.

Appendix B: Updated Test Document

B.1 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.

B.2 Project Requirements

2.1 *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.

2.2 *Constraints:*

- ~~➢ The system must work with all data types within the scope of CVST.~~
- ➢ The system must work with at least two data types.
- ➢ 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.

2.3 *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.
- ➢ 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 and ISO/IEC 25022.

B.3 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 precomputed 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.

- ➢ 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.
    - ○ Assess the quality of the system using Software Usability Measurement Inventory.

B.4 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.

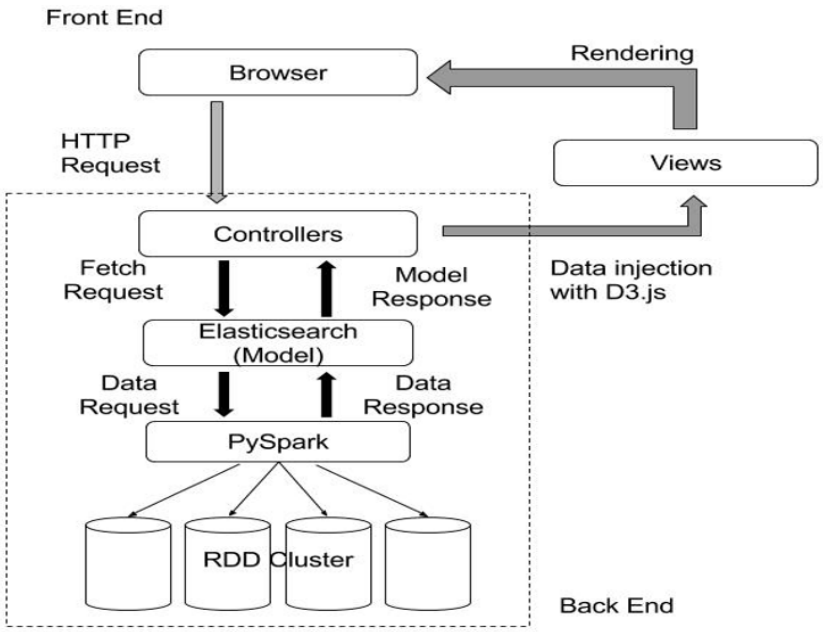

Diagram B.4-1 Original overview diagram of system design

Modified:

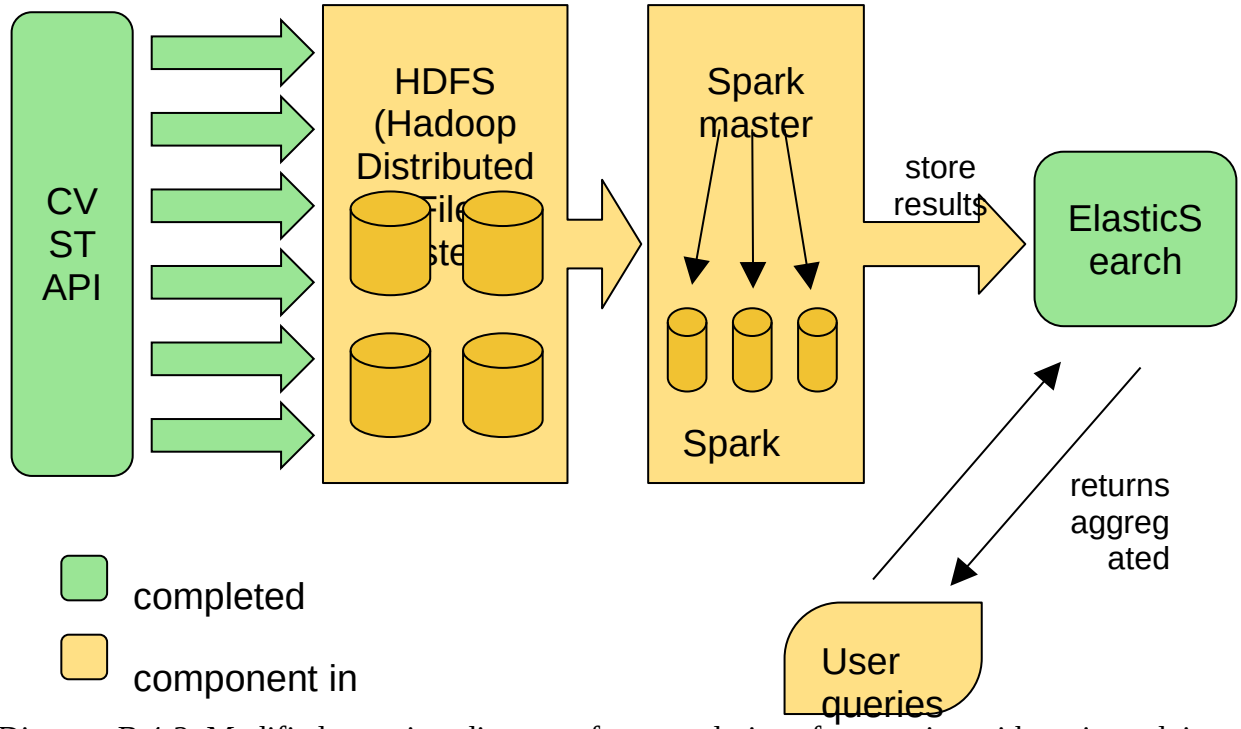

Diagram B.4-2: Modified overview diagram of system design after meeting with project advisor Hamzeh, and the progress of completing each design component