Big Data Visualization

Human Flow Visualization

Mirage Abeysekara
Computer Science & Engineering Department, University of Moratuwa, Sri Lanka
mirage.12@cse.mrt.ac.lk

***Abstract*— Computer based visualizations have been a popular approach that provides accurate and efficient visualization techniques to the users. There is an increasing growth in geographical data with the development of mobile devices. There is a trend to analyze the human behaviors and patterns with respect to the geographical locations in order to take decisions and predictions. This paper presents a Human Movement Visualizer tool to provide populating flow data between regions of a given map. The tool was designed to use with modern web browsers by implementing the complete functionality of the system using client-side JavaScript language. Also the tool can integrate with other visualization tools by introducing component based architecture to the system. Users of this tool can give a Shapefile as a base map and CSV (Comma Separated Values) based data to visualize the geographical data. The main goal of this tool is to provide visualizations for the transport planning in Sri Lanka.**

Keywords—Geographical; Goolge Maps; JavaScript; Client-side; Component based arcutecture; CSV; Shapefile

I. Introduction

Today most of the devices are collecting data related to human behavior with their geographical locations. This includes mobile phones, vehicle GPS systems, geo tagged pictures [1], etc. Based on these data collections, researchers analyze human behaviors and patterns with respect to the geographical locations to take decisions and predictions. As the data grows with time, problems can arise in data analyzing due to the complexity and huge amount of data. Visualizing these data using geographical map is a solution for this problem.

Transport planning is vital in urban areas due to the high crowd and vehicle density. Estimating the flows of human movement between two regions are normally done by using surveys. Visual analysis is used for transport planning to easily understand the human movement [2]. These visualizations either done manually or generate using software. The proposed visualization tool allows generating and modifying visual models easily based on gathered data. Also this tool allows the interactive visualizations to the wider audience.

The main purpose of this data visualization tool is to communicate information clearly and efficiently with the users via a statically or dynamically generated maps [3]. Also the system makes complex data to be more accessible, understandable and usable by visualizing it. Furthermore the users of the system can analyze and predict traffic flow patterns in a country by populating known data on top of the map.

Section II describes the background of data visualizations and importance of a visualization tool. Furthermore it explains currently available visualization tools and their features. Section III describes the system requirements and design of the system using the system architecture. Section IV explains the implementation procedure as well as user interfaces of the system. Section V explains the testing plan of the system including unit testing, user interface testing and performance profiling. Finally the section VI describes the conclusions and further improvements to the system.

II. Literature Review

Big data visualization uses tools to represent data in the form of charts, maps, tag clouds, animations, or any graphical image that makes content easier to understand. The past few years have seen a major improvement of visualization applications, technologies and infrastructure to support increasingly sophisticated visual representations of data [4].

Creating a visual representation of statistics once involved compiling, interpreting, parsing and determining the visual presentation type that would mean for the data. Big data visualization tools provide a simplified process from compiling data to visualizing data [4]. Therefore the most visualization tools work as a black box which the users can paste data or upload spreadsheet, CSV file to the application interface and the tool then turns the data into a visualization, such as a colored map or an interactive chart. A skilled user can use a wide range of technology tools to create any kind of visualization they want, but the importance of newer tools is that they allow nontechnical users to visualize data without any complexity. Therefore finding a tool that can turn data into an appropriate and informative visualization can be difficult [5].

Visualizations help people to see the data in different perspective that cannot be easily seen by just analyzing the data. Even the data volumes are very large, the hidden patterns can be seen quickly and easily [6]. Visualizations make people to share data with others simply and easily. Before the visualization tools available public, it was researchers who designed visuals to analyze data and find out trends. However, current web applications allow anyone with access to data to enter information and easily create a virtualization of those. People can now easily create visualizations that might reveal trends that are not obvious from the numbers alone. Therefore these kind of public visualization tools allow improving the development process from home to large scale businesses.

Geographic information systems allow individuals to gather, transform, and analyze data. Every day the new geographic data visualization tools become available that they allow users to easily create interactive visualizations of big data gathered by geographic information systems. The most common example is the Google Maps [7]. Google Map provides visualization for their API (called Data Layer [8]). This will help for the developer to easily add geographical data and visualize in on top of the google maps. But for the non-technical people it is very difficult to learn and use it. Fig. 1 shows population growth of the world using Google map's Data Layer. The MapsData is another visualization tool which provides customized visualizations [9]. The work done by Koblin [10], visualizes the flight patterns between regions using line based visualization technique to populate data. They have described the Human Flow Visualizer tool that uses the line based data visualization to populate data.

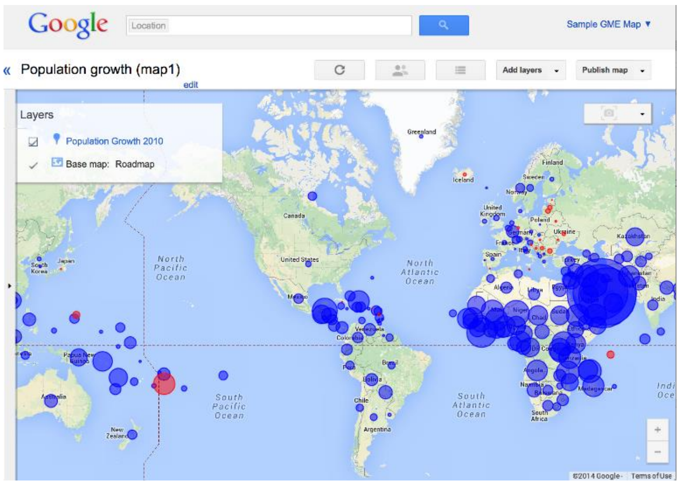


Fig. 1. Google Maps Data Layer - Population growth of the world

III. System Models

A. System Requirements

The web based big data visualization tool for visualizing human flows was design to maximize the analyzer's productivity by providing tools to generating and modifying visualizations in less time. More specifically, this system is designed to allow a user to render a given base map and draw the flow data on top of the map. The generated visualizations can be statically saved for publishing or dynamically saved for interactive presenting.

1) *Functional Requirements*

The visualization tool allows loading maps from Shapefiles and Google Maps as overlays. Also the user must have the ability to load CSV formatted data file which contains visualization data ("source, destination, volume, time" format in the CSV file). After loading the base map and the data file the tool is required to render the visualization with directed arrows or colored lines.

The visualization tool should have visualization filters which allow the user to filter regions. As an example if a user want to see the flows from region X the lines must show flows from region X to other regions. If a time based CSV is loaded user should have ability to select a time based flow from (or to) specific region. Visualization styling is another functional requirement for this tool. Styling options enables users to customize line colors, region colors, changing the line width etc. Furthermore the tool should provide zoom and panning functionalities with content resizing and enables uses to save the visualization as image for publishing or further use.

2) *Non-Functional Requirements*

The tool needs to handle large amount of data (support for visualizing 500,000 flows smoothly) with minimum rendering time without crashing the web browser. The visualized geo locations must be accurate and the tool should have high performances. It is required to provide understandable web based user interface for nontechnical users. Further, the tool should be easily extendable for further development and should be able to integrate with other big data projects.

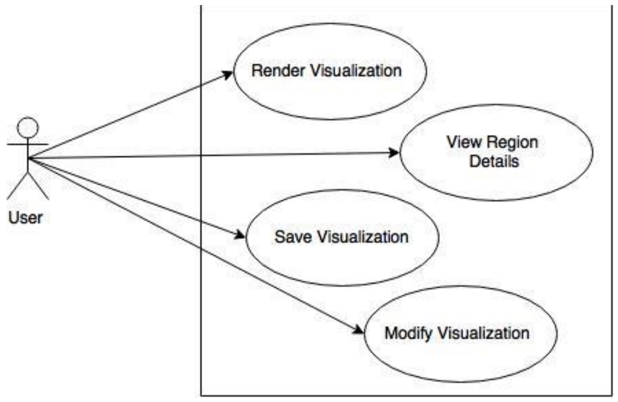


Fig. 2. Use case Diagram

Fig. 2 shows the use case diagram for the tool. The visualization tool has only one actor who is the analyzer of the visualized data. The Render Visualization use case describes loading the Shapefile as the base map and populating the CSV data on top of the loaded map. Also this includes the sub path of visualizing CSV data on top of the Google Maps. View Region Details use case provide the user to view the flow details of specific region by selecting a region on the map. User can zoom or pan if required for resizing the rendered map. The use case Modify Visualization describes the visualization editing features of the tool such as the changing the colors and size of the map, editing the flow line width, etc. The Save Visualization usecase provides user to save the geographical map with given data for further use or publishing. User can choose two save versions. Static save allows user to save the visualized map as image format for publishing. The dynamic save allows user to save the all geographic map and data for view later which enables interactive visualization of the map.

B. System Design

The tool has used component based architecture to improve the reusability and maintainability. (Fig. 3) One of the major architecture goals is to provide integration because of this software tool is a part of a major big data visualization tool collection and required to integrate in to a one big data project in future. The visualizations are done only at the client-side (web browser) and the tool is written in JavaScript as components.

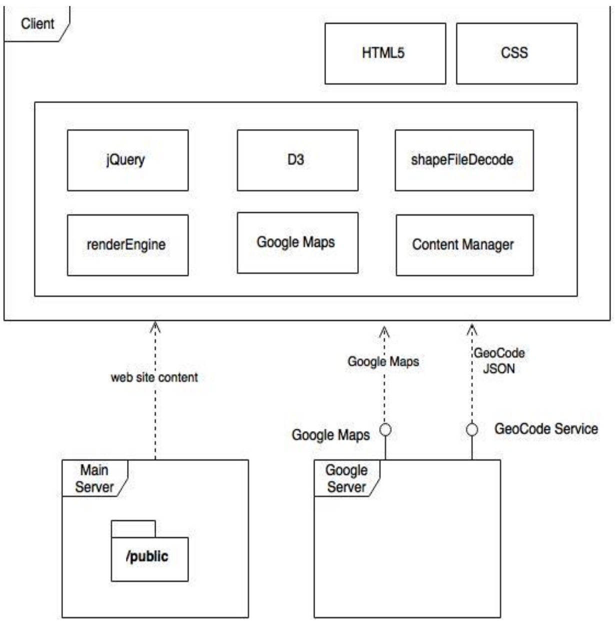


Fig. 3. Abstract architecture of the system

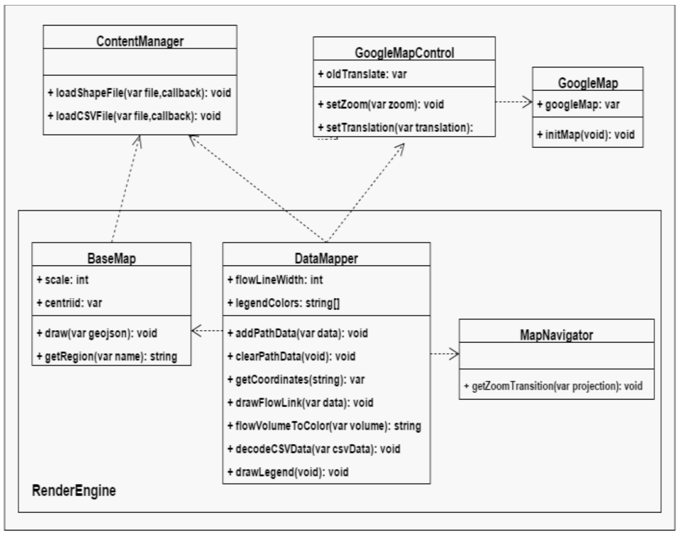

ContentManager
+ loadShapeFile(var file,callback): void
+ loadCSVFile(var file,callback): void
GoogleMapControl
+ oldTranslate: var
+ setZoom(var zoom): void
+ setTranslation(var translation):
GoogleMap
+ googleMap: var
+ initMap(void): void
BaseMap
+ scale: int
+ centriid: var
+ draw(var geojson): void
+ getRegion(var name): string
DataMapper
+ flowLineWidth: int
+ legendColors: string[]
+ addPathData(var data): void
+ clearPathData(void): void
+ getCoordinates(string): var
+ drawFlowLink(var data): void
+ flowVolumeToColor(var volume): string
+ decodeCSVData(var csvData): void
+ drawLegend(void): void
MapNavigator
+ getZoomTransition(var projection): void
RenderEngine

Fig. 4.Class Diagram

In Fig. 4, the RenderEngine component have three major responsibilities which are to render the base map and populate the flow data on top of the map, respond to the user navigations such as zooming and panning, and change the visualization graphics according to the input data.

Inside the RenderEngine the BaseMap component provides base map layer generation on the web browser (takes GeoJSON [11] data) also handles the map region rendering. The MapNavigator takes user navigation inputs such as zoom and pan details to update it on the web browser while saving the current state of zoom level and panning. DataMapper is a one of major component which decode data and overlay the flow data on top of the base map. As well as it provides calculation and rendering the legend based on the data. It also handles map projection and geo coordinates to pixel conversion. The DataMapper is mainly associated with the BaseMap and GoogleMapHandler to get the locations, region names and mappings with region data. ContentManager provides file IO functionalities such as loading the Shapefile, converting Shapefile to GeoJSON and loading CSV data. The GoogleMapControl is used for handle Google Map API functions and map with current visualization data.

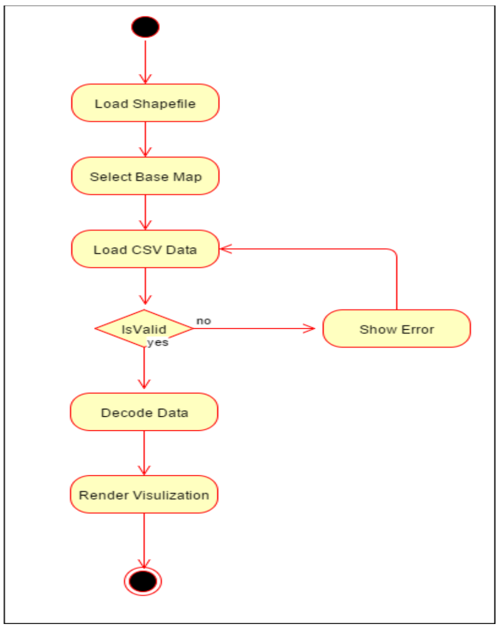


Fig. 5 Activity Diagram

Activity diagram in Fig. 5 shows the activity flow in the Render Visualization main use case and Fig. 6 shows the corresponding sequence diagram. Here, D3 is 3rd party component which provide graphics rendering on the web browser. The ShapeFile decoder provides the conversion between binary data to the GeoJSON format. The RenderEngine component takes the GeoJSON result and render it in the web browser using D3 library. When the user loads the CSV data file it will be sent to the DataMapper (which is inside the RenderEngine component). It will decode the data and populate data on top of the previously rendered map.

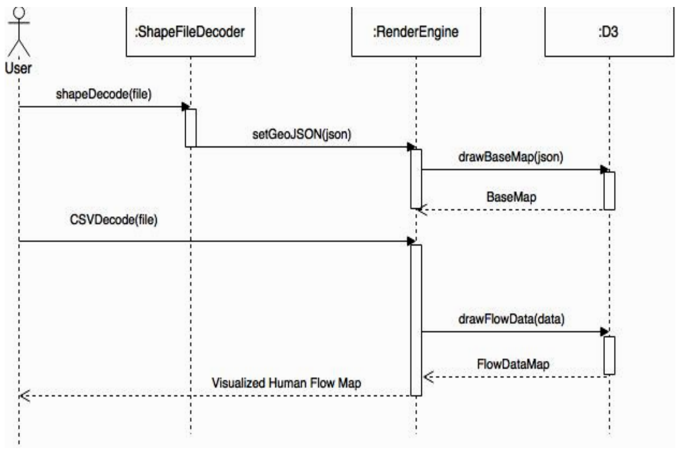


Fig. 6. Sequence Diagram

IV. System Implementation

A. Implementation Procedure

The visualization tool was developed using the waterfall method as the requirements of the application are not likely to change and system requirement specifications are well defined. The system was purely based on the web browsers JavaScript [12] language and map rendering was done using the HTML5 [13] SVG (Scalable vector graphics) [14].. The browser based map rendering and data decoding is choose to reduce the network traffic. If the tool was required to decode Shapefile by sending to a server, it will cause network traffic and time because some of the Shapefile are larger than 20MB.

NetBeans [15] IDE was used for the development of the system. NetBeans chrome extension helps to automatic page rendering in the Google Chrome web browser and allows debugging JavaScript thought the NetBeans IDE. Also the Google Chrome was used for the debugging CSS [16] and HTML in the tool. The D3 and jQuery JavaScript API was heavily used in the application due to the easy rendering and event handling provide by those API's. Shp.js library was used to convert the binary Shapefile in to GeoJSON [11] format the.

```
Function DecodeCSV (csvData){
         minFlowVolume = MAX_VALUE
         maxFlowVolume = 0
         regionListSource = new Array();
         regionListDestination = new Array();
         dateTimeList = new Array();
//calculate min and max volumes and generate source
and destination region and timing arrays
        for each(dataRow in csvData) {
            tmpVolume = dataRow.Volume
            if (maxFlowVolume < tmpVolume)
                  maxFlowVolume = tmpVolume;
            if (minFlowVolume > tmpVolume)
                 minFlowVolume = tmpVolume;
            if
(regionListSource_Not_Contains(dataRow.Source))
          regionListSource.add(dataRow.Source);
            if (regionListDestination_Not_Contains
(dataRow.Destination,) )
regionListDestination.add(dataRow.Destination);
// timeing array if time based csv uploaded
            if
(dateTimeList_Not_Contains(dataRow.Time))
                dateTimeList.push(dataRow.Time);
         }
}
```

Fig. 7. Algorithm for data decoding

The Shapefile provides region names with their actual geo coordinates. Therefore it was used to find geo coordinates based on the given CSV data. DIVA-GIS [17] and StatSilk [18] are examples for those who provide free Shapefiles. Sri Lanka, USA, UK and Russia administrative boundaries Shapefile was used to test the visualization tool. The mock CSV data generation was done using the mockaroo.com which provide up to 1000 entries per CSV [19]. Fig. 7 and Fig. 8, show the algorithms used for data decoding and converting flow volume to color, respectively. The flow volumes are dived into 7 steps and calculate the color based on the step boundaries.

```
Function FlowVolumeToColor (volume) {
    flowVolume = (volume)
    minVolume = decoded csv min volume
    maxVolume = decoded csv max volume
    // calculate the 7 color step size
    colorStepSize = (maxVolume - minVolume) / 7
    color = black
    // check the margins of the volume and assign
colors
    if (minVolume <= flowVolume && flowVolume <
(minVolume + (colorStepSize * 1)))
    {
        color = assign legendColor1
    }
    else if ((minVolume + (colorStepSize * 1)) <=
flowVolume && flowVolume < (minVolume +
(colorStepSize * 2)))
    {
        color = assign legendColor2
    }
Conditions for step size continues…
    else if ((minVolume + (colorStepSize * 6)) <=
flowVolume && flowVolume <= maxVolume)
    {
        color = assign legendColor7
    }
    return color
}
```

Fig. 8. Algorithm for converting flow volume to colour

One of the major challenges were the decoding and rendering the Shapefile on top of the web browser. Because of the Shapefiles are not heavily used in the web designing it was very difficult to find resources. In the early stages of the development it was mainly focused on decoding data and draw lines between regions according to the decoded data.

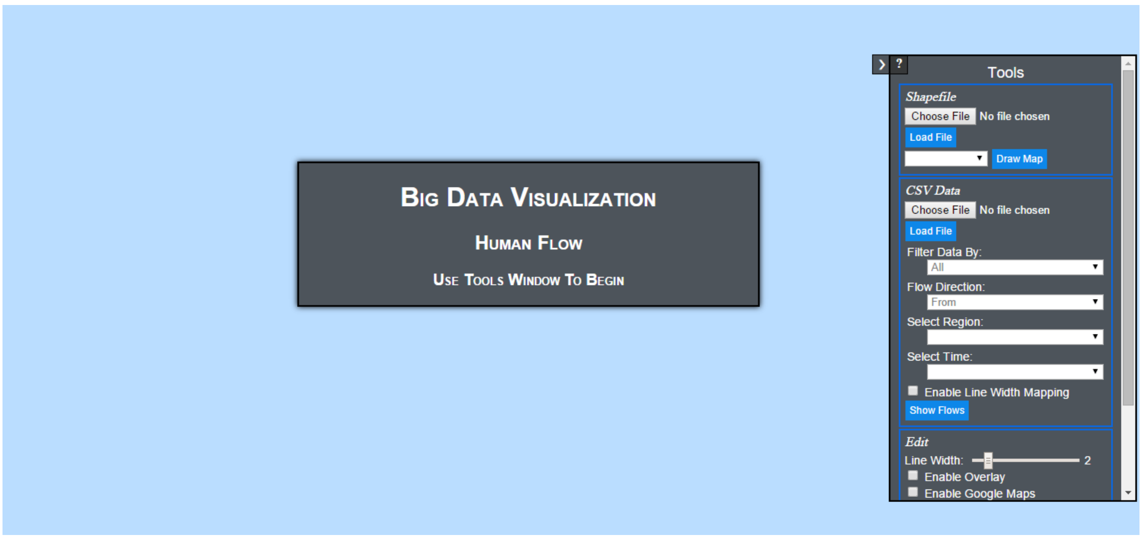
B. Main Interfaces

Fig. 9 shows the main interface after loading the tool. The right side tool panned provide functionalities of the main use cases. Fig. 10 shows flow's going out from Colombo to other regions in Sri Lanka. Fig. 11, shows Google Map overlaid view of the visualization.

V. System Testing and Analysis

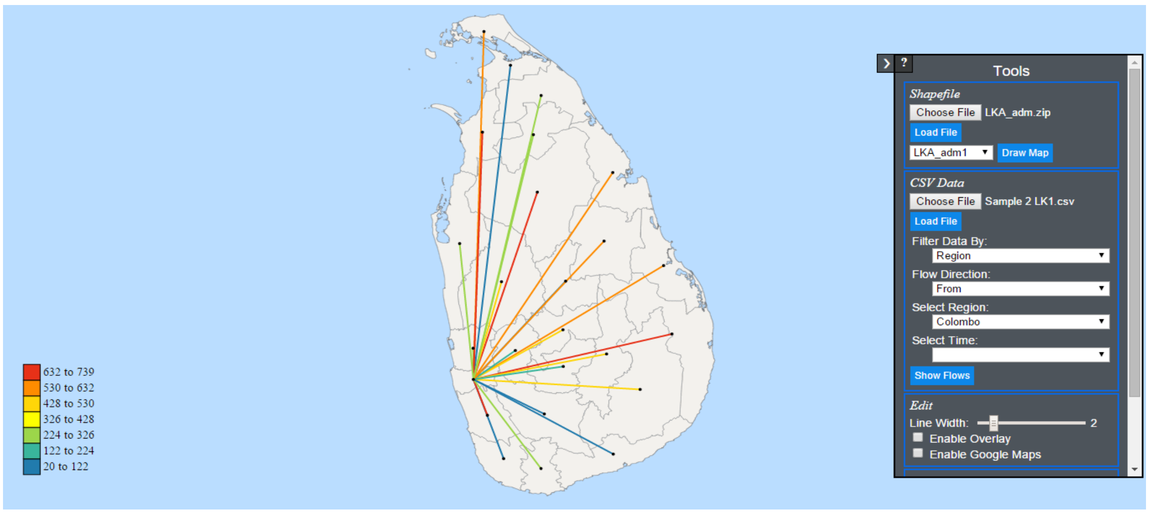
The test plan ensures that the implementation of the visualization tool meets its specifications and design criteria. The major part of the testing is based on the JavaScript testing due to the most of the functionalities of the tool is implemented using the JavaScript. The testing plan will help to deploy the tool for the users with minimal risk of software failure. The main test plan is expect to find all possible bugs and ensure the implemented tool meets the specification of the system requirement document and the design document. The majority of the testing are done as black-box testing [20] technique.

Unit testing [21] consists of testing all the basic functionality of the tool such as Volume to Color mapping, Region to Geocode decoding, etc. The QUnit JavaScript unit testing framework was used to test the functionality of the system. Fig. 12, shows the output of the functionality testing of the RenderEngine and DataMapper components.



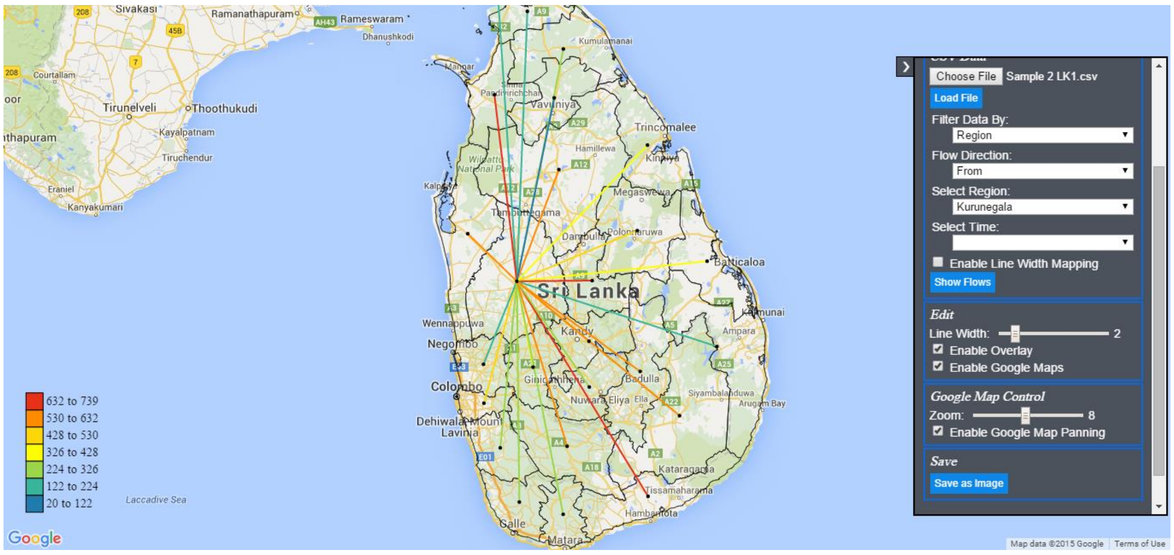
Shapefile: No file chosen
CSV Data: No file chosen
Filter Data By: All
Flow Direction: From
Select Region:
Select Time:
Enable Line Width Mapping
Line Width: 2

Fig. 9. Main Interface



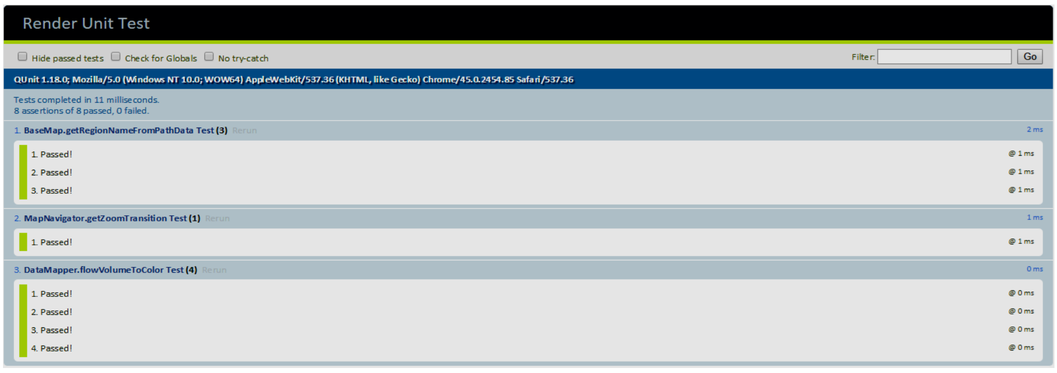
Shapefile: LKA_adm.zip
LKA_adm1
CSV Data: Sample 2 LK1.csv
Filter Data By: Region
Flow Direction: From
Select Region: Colombo
Select Time:
Line Width: 2

Fig. 10 .Visualization flows of Colombo



CSV Data: Sample 2 LK1.csv
Filter Data By: Region
Flow Direction: From
Select Region: Kurunegala
Select Time:
Enable Line Width Mapping
Line Width: 2
☑ Enable Overlay
☑ Enable Google Maps
Zoom: 8
☑ Enable Google Map Panning

Fig. 11.Google Map overlaid visualization



```
Render Unit Test

Hide passed tests   Check for Globals   No try-catch                                   Filter:            Go
QUnit 1.18.0; Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
Tests completed in 11 milliseconds.
8 assertions of 8 passed, 0 failed.
1. BaseMap.getRegionNameFromPathData Test (3)  Rerun                                              2 ms
   1. Passed!                                                                                     @ 1 ms
   2. Passed!                                                                                     @ 1 ms
   3. Passed!                                                                                     @ 1 ms
2. MapNavigator.getZoomTransition Test (1)  Rerun                                                 1 ms
   1. Passed!                                                                                     @ 1 ms
3. DataMapper.flowVolumeToColor Test (4)  Rerun                                                   0 ms
   1. Passed!                                                                                     @ 0 ms
   2. Passed!                                                                                     @ 0 ms
   3. Passed!                                                                                     @ 0 ms
   4. Passed!                                                                                     @ 0 ms
```

Fig. 12. Qunit Test Results

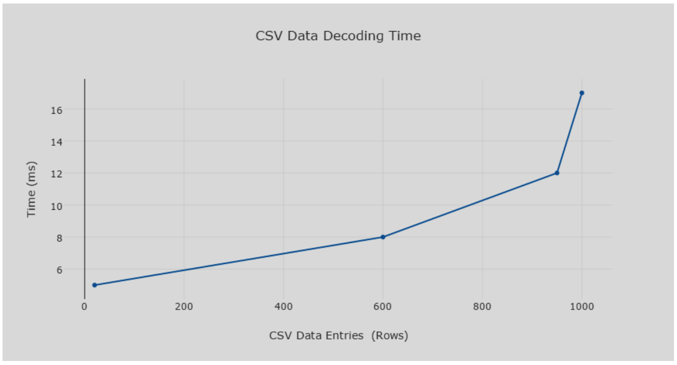


Fig. 13. Data Decoding Performance

User Interface testing are done using Selenium automation tool [22]. Web interface testing such as map navigation, Shapefile uploading, data viewing, cross browser compatibility etc. were performed. User experience and the usability of the tool are expect to evaluate during the test. In Performance profiling testing, the response time of the web tool is evaluated based on the web browser. Due to the nature of the big data tools, data loading and decoding time were tested. Furthermore map loading and rendering time, navigation performance was evaluated. Fig. 13, shows CSV decoding time performance.

VI. Conclusion and Future Work

This paper presents a Human Movement Visualizer tool that visualizes the flow data between regions on top of a map. It provides grater advantages to the transport planning in urban areas by helping analyzer's to visualize high crowd and vehicle density. Therefore the users of can gather large amount of data from the devices and analyze it using this web based tool. This tool is publically available to analyze flow data.

The tool can be extended to integrate with existing big data tools. Further improvement of this architecture is to allow all the visualization tools are integrate in to one website. Therefore users can visualize various kind of data types using a single website. This will improve the country planning, future prediction and further development to the society.

References

[9] Wikipedia, 'Geotagged photograph', 2015. . Available: https://en.wikipedia.org/wiki/Geotagged_photograph.

[1] Fhwa.dot.gov, 'Visualization for Transportation Planning - Visualization In Planning - Scenario And Visualization - Planning - FHWA', 2015. . Available:http://www.fhwa.dot.gov/planning/scenario_and_visualization/visualization_in_planning/visplanning.cfm.

[2] Earth.nullschool.net, 'earth :: a global map of wind, weather, and ocean conditions', 2015. Available: http://earth.nullschool.net/

[3] Wiki.doit.wisc.edu, 'Data Visualization Literature Review - Engage Program - Confluence - DoIT Wiki', 2015. . Available: https://wiki.doit.wisc.edu/confluence/display/engage/Data+Visualization+Literature+Review

[4] Lirneasia.net, 'Big Data for Development » LIRNEasia - a regional ICT policy and regulation think tank active across the Asia Pacific', 2015. . Available: http://lirneasia.net/projects/bd4d/.

[5] Sas.com, 'Data Visualization: What it is and why it's important', 2015. . Available: http://www.sas.com/en_us/insights/big-data/data-visualization.html

[6] Google Maps, 'Google Maps', 2015. . Available: https://www.google.com/maps

[7] Google Developers, 'Data Layer', 2015. . Available: https://developers.google.com/maps/documentation/javascript/datalayer.

[8] Mapsdata.co.uk, 'MapsData | Data visualization and custom maps', 2013. . Available: http://www.mapsdata.co.uk/

[9] Aaronkoblin.com, 'Aaron Koblin - Flight Patterns', 2015. . Available: http://www.aaronkoblin.com/work/flightpatterns/

[10] Wikipedia,'GeoJSON',2015..Available:https://en.wikipedia.org/wiki/GeoJSON

[11] Wikipedia,'JavaScript',2015..Available:https://en.wikipedia.org/wiki/JavaScript

[12] Wikipedia,'HTML5',2015..Available:https://en.wikipedia.org/wiki/HTML5

[13] Mozilla Developer Network, 'SVG In HTML Introduction', 2015. .Available:https://developer.mozilla.org/en/docs/SVG_In_HTML_Introduction

[14] Wikipedia, 'NetBeans', 2015. . Available: https://en.wikipedia.org/wiki/NetBeans.

[15] Wikipedia, 'Cascading Style Sheets', 2015. . Available: https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[16] Diva-gis.org, 'Download data by country | DIVA-GIS', 2015. . Available: http://www.diva-gis.org/gdata.

[17] Statsilk.com, 'Download free shapefile maps | StatSilk', 2015. . Available: http://www.statsilk.com/maps/download-free-shapefile-maps.

[18] Mockaroo.com, 'Mockaroo - Random Data Generator | CSV / JSON / SQL / Excel', 2015. . Available: http://mockaroo.com/

[19] Wikipedia, 'Black-box testing', 2015. . Available: https://en.wikipedia.org/wiki/Black-box_testing.

[20] Wikipedia, 'Unit testing', 2015. . Available: https://en.wikipedia.org/wiki/Unit_testing.

[21] Wikipedia, 'Selenium (software)', 2015. . Available: https://en.wikipedia.org/wiki/Selenium_(software).