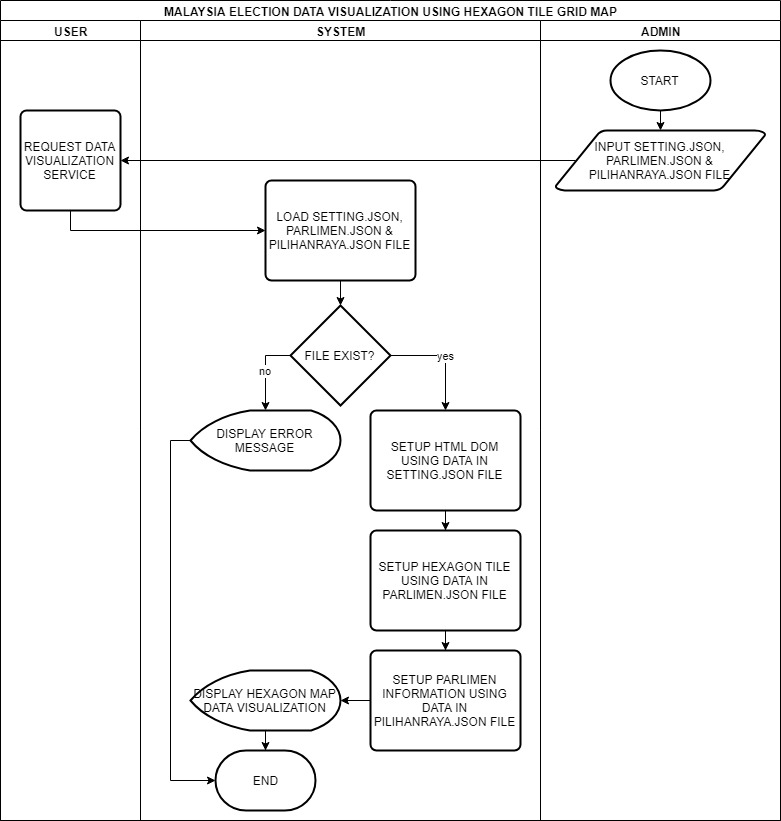
**CHAPTER 4**

**RESULT AND ANALYSIS**

This chapter provides the result and analysis of this project which includes the system flow, system development flow and system user interface (UI) explanation. This chapter will also focuses on the process and the challenges faced during the development of the system.

* 1. **Introduction**

The main functionality for this system is to convert the election data into a hexagon tile grid map visualization. The election data were manually stored into a json file in order to make the retrieval of the data as smooth as possible. The diagram for the system flow are stated as in figure 4.1:

  
Figure 4.1: System flow flowchart

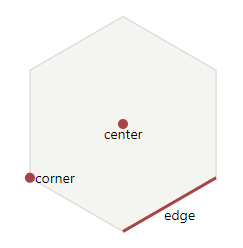
The system must have the json file data (setting.json, parlimen.json and pilihanraya.json) in order to run properly. The admin must include those files first before the system can be deployed. Only the, the user can request and view the data visualization.

After the user had requested for the data visualization service, the system will automatically load all data from ‘setting.json’, ‘parlimen.json’ and ‘pilihanraya.json’ file. The system will use the data in setting.json file to setup the width, height and padding of the Scaleble Vector Graphic (SVG) for the Document Object Model (DOM) element in the Hypertext Markup Language (HTML) and the radius, position of tooltip and colors of a single hexagon. Then, it will load the data in ‘parlimen.json’ file that will be used together with the data in ‘pilihanraya.json’ file in order to populate the hexagons for each parliament into DOM. Finally, the system will use the data in ‘pilihanraya.json’ file to setup the election data for each of the parliament that were populated together.

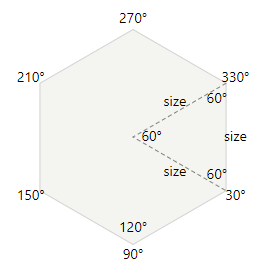
The system were developed by using a combination of HTML, Cascading Stylesheet (CSS), Javascript (JS) and a JS library called Data Driven Document (D3.js) together with hexagon plugin, hexbin. The HTML and CSS were used to setup the foundation of the User Interface (UI) of the system including the placement of the SVG DOM and the background color. The JS and D3.js were used in order to implement the algorithm flow of the system such as creating and populating the hexagon based on the coordinate data in ‘parlimen.js’. The hexbin plugin of D3.js were used to help creating the hexagon tile for a better coordination. Further feature of the hexbin plugin are explained in the modeling topic 4.2.4.

* 1. **Development Flow**
     1. **Business Understanding**
     2. **Data Understanding**
     3. **Data Preparation**
     4. **Modeling**
        1. **Single Hexagon Tile Setup**

There are certain properties that needed to be considered in order to develop the system, mainly the hexagon tile. Hexagon is a 6 sided polygon and a regular hexagon contains same length of 6 edges, 6 corners (P. Amit, 2015). The interior angle are 120° and the ‘wedges’ (a total of 6) is 60° (P. Amit, 2015). The properties of the hexagon are shown in the figure 4.2:

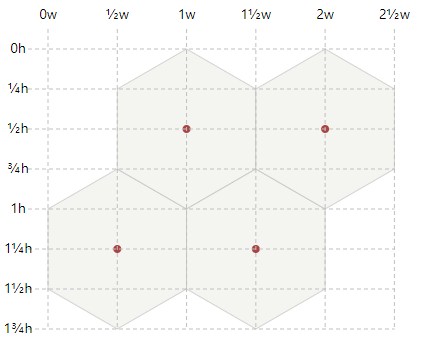
  
Figure 4.2: Basic hexagon properties

(Source: Amit, 2015)

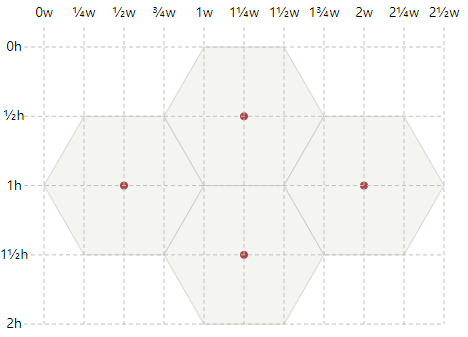
  
Figure 4.3: Angles in a regular hexagon

(Source: Amit, 2015)

There are 2 types of hexagon, which is pointy-top and flat-top hexagon. The pointy-top hexagon are as shown in Figure 4.2 and the flat-top hexagon are the 90° rotated version of the pointy-top hexagon (P. Amit, 2015). The view of a hexagon tile grid map when using the pointy-top hexagon is more upright than flat-top hexagon. Figure 4.4 shows both types of hexagon:

  
Figure 4.4: Pointy-top hexagon

(Source: Amit, 2015)

  
Figure 4.5: Flat-top hexagon

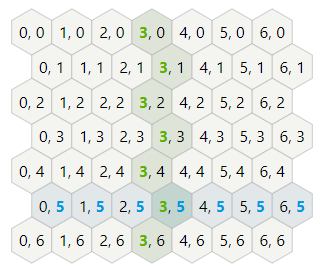
(Source: Amit, 2015)

This project uses the hexbin plugin for D3.js in order to draw the hexagon tile. The ability to easily create a single hexagon tile using the hexbin plugin is one of the reasons why this plugin was chosen at the first place. Hexbin plugin will produce a SVG path based on the developer’s configuration. The SVG path will eventually create a hexagon tile. The plugin generates the hexagon from the perspective of a circle. Therefore, the main parameter that the developer had to include is the hexagon tile’s radius because it is considered the size of the hexagon. After that, the developer can plot the hexagon anywhere inside the SVG plain.

This project implements the pointy-top hexagon because the hexbin plugin are not developed with a flat-top hexagon in mind. However, by using hexbin plugin, flat-top hexagon can still be created nonetheless just by transforming (rotate) a single hexagon, but there are further problems related with the rotation of the hexagon. The problems are discussed further in Hexagon Tile Grid Map Plotting sub-topic.

* + - 1. **Hexagon Tile Grid Map Plotting**

There are a number of approaches to plot a hexagon tile into a map and each approaches have their own coordinate system which is offset coordinate, cube coordinate and axial coordinate (P. Amit, 2015). Offset and axial coordinate involves 2 dimensional (2D) coordinate system which is X and Y coordinate while cube coordinate involves 3 dimensional (3D) coordinate system which is X, Y and Z (P. Amit, 2015). Figure 4.6 shows the examples of the said approaches:

  
Figure 4.6: Example of offset coordinate system

(Source: Amit, 2015)

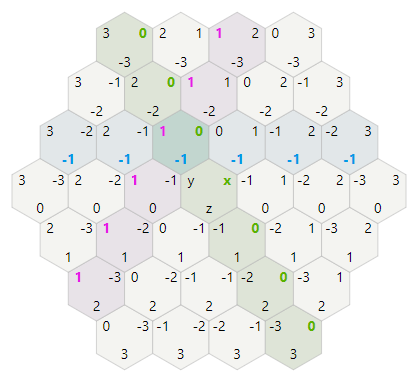
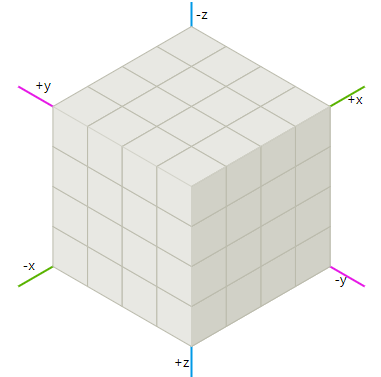
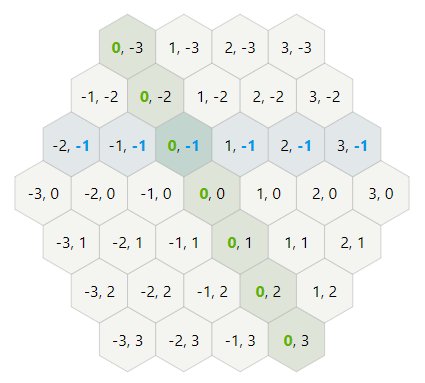


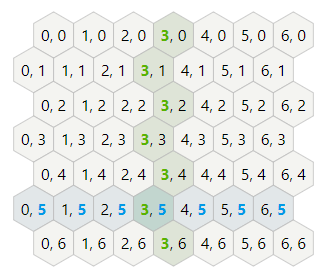
Figure 4.7: Example of cube coordinate system

(Source: Amit, 2015)

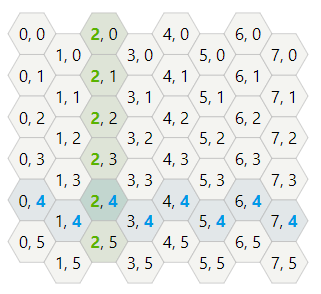
  
Figure 4.8: Example of axial coordinate system

(Source: Amit, 2015)

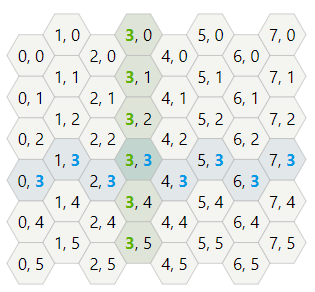
This system uses the offset coordinate system in plotting the hexagon across the map. This is because it is the most common approach for map like shape (P. Amit, 2015). The name of the coordinate system represents the way the coordinate system are implemented. There should be offsets between alternate columns or rows. Based on Figure 4.6, the offset row is the odd row because the odd row are indented. This type of approach is called “odd-r horizontal layout” (P. Amit, 2015) which means the offset will the odd row and the coordinate are arranged sequentially in a horizontal manners. Figure 4.9 until Figure 4.11 shows the examples of other possible layouts for offset coordinate system:

  
Figure 4.9: Example of “even-r horizontal” layout

(Source: Amit, 2015)

  
Figure 4.10: Example of "odd-q vertical" layout

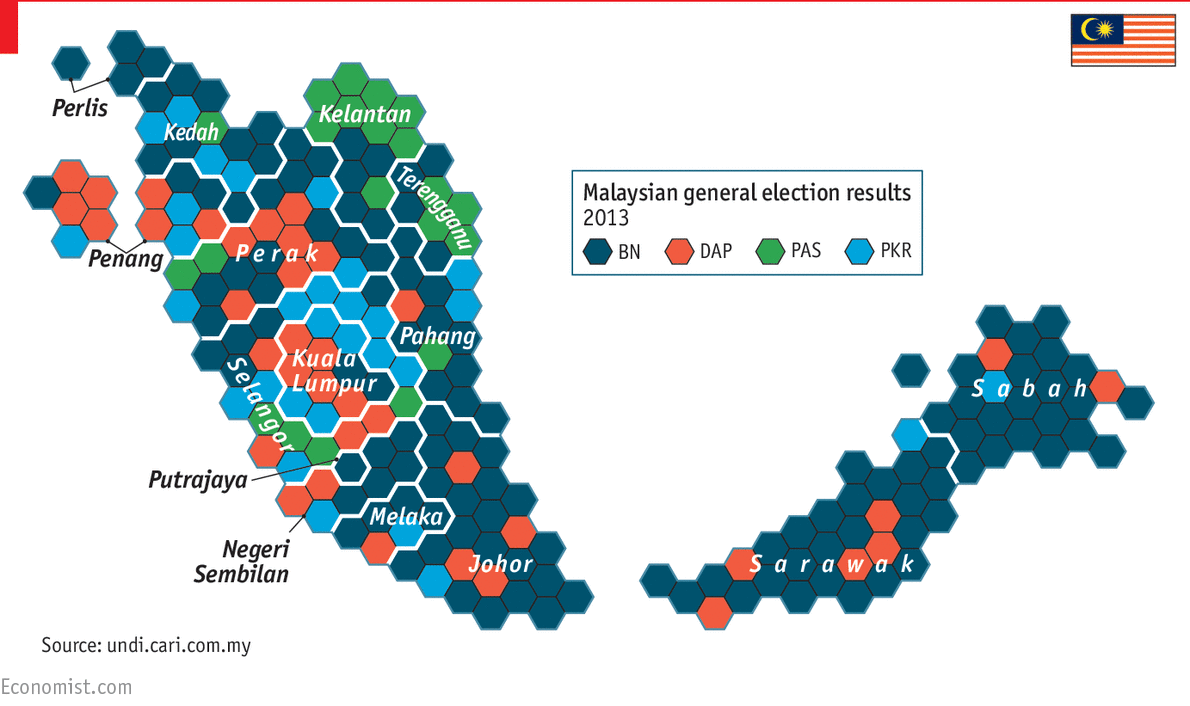
(Source: Amit, 2015)

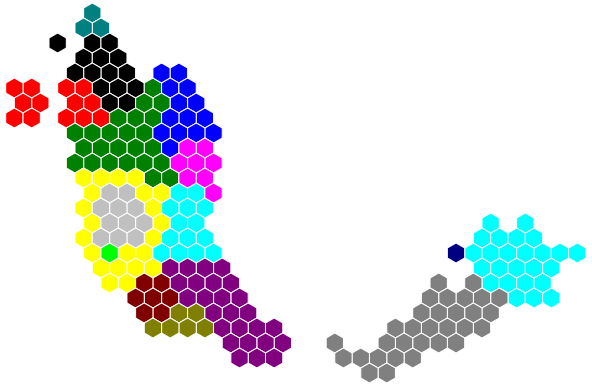
  
Figure 11: Example of "even-q vertical" layout

(Source: Amit, 2015)

The map were populated by states by using the odd-r horizontal layout. For example, the hexagon of parliaments in Perlis will be populated first before parliaments in Kedah. Each parliaments will have the same radius of hexagon.

However, as mentioned previously, the hexbin plugin were not developed with a flat-top hexagon in mind. As a result, the simplest and best approach of using the plugin is by only implementing the pointy-top hexagon. The difference between the result of pointy-top and flat-top hexagon is that whether the resulting map looks more upright or horizontal. The pointy-top hexagon tile grid map are seen more upright map while the flat-top hexagon are seen more horizontal map. Figure 4.9 and Figure 4.10 shows the difference between both types of hexagon:

  
Figure 4.12: Example of flat-top hexagon tile grid map

  
Figure 4.13: Example of pointy-top hexagon tile grid map

Even though flat-top hexagon can still be achieved by using a transformation method (rotation), but the coordinate system will not be the same and this will lead to a randomly generated hexagon even though the developer uses the same coordinate for each of the hexagon. For example, a pointy-top hexagon coordinate is at (0, 0), but when it is rotated 90°, the coordinate that were registered will still be the same which is (0, 0), but the actual coordinate that the were shown by the browser is (-10, 10) which in this case is not the same as the expected output. The expected output is a rotated hexagon tile with a same coordinate.

* + - 1. **JSON Data File Setup**

**[3 data file, settings.json, parlimen.json & pilihanraya.json, problem with reading data in json file using d3.js]**

All of the data that were used in this project were stored in JSON format because JSON is the easier approach to store and process data (L. Vogel, 2017). JSON short for Javascript Object Notation is a notation that were widely used in current technologies. Its primary functionality as a data structure used for easier data storage. The notation is based on the native Javascript language which represents an array or objects. Other reason why JSON data format were used in this project is because this project uses the D3.js library which is the extension of Javascript itself. So, it is easier to retrieve the data stored in files with the JSON notation.

There are a total of 3 files that were used by the system which is “settings.json”, “parlimen.json” and “pilihanraya.json”. The “settings.json” file were used to store the settings for the layout of the application, the width of the SVG plain, hexagon radius, tooltip location and the color that represent each political party. The “parlimen.json” file stores the information of the hexagons and the parliaments such as the coordinate of the hexagon, name of the state, parliament code and parliament name. The “pilihanraya.json” file includes the data that were used to represent the election result such as state name, parliament code, total number of votes for that particular parliament, the information regarding the winning party and their candidate.

* + - 1. **Mouse On Hover Tooltip Setup**

**[tooltip will show parliament code & name]**

* + - 1. **Mouse On Click Popup Setup**

**[popup will show the election data]**

* + - 1. **Hexagon Tile Color Setting**
      2. **State Border setting**
    1. **Evaluation**
    2. **Deployment**
  1. **User Interface**
     1. **First Level Information**
     2. **Second Level Information**

**tooltip dan event handling hover over shows parliament number and name**

* + 1. **Third Level Information**

**pop up and event handling mouse click to show the parliament details information, candidate name, and bar chart**