 CHAPTER 10

ACCIDENT LIVE UPDATION AND RESPONSIBILITY MODEL

10.1 GENERAL

Accessibility and Interactivity are keywords of information today and that is equally important in science as anywhere else. Information can be encoded into various forms for transmission and interpretation. This thesis has the aim to develop an open source software tool to design a web map application that can be and utilized by any one and institute. The Java platform used in the open source OpenLayers will allow the authorized users to view and manipulate the GIS map data through a web map application. Interactive maps can be a great way of displaying useful information in an engaging and attractive way by inviting the authorized users to take actions. Whenever the e-server is available, it can be shared to users worldwide. The application users can then have a list of available data layers, do operations such as zoom in and zoom out, choose backgrounds pan, change symbols and colours, add own data of theirs and animate it. The data and contents displayed in the respective server can be accessed by the users through their smart phones.
10.2 GEOSERVER

The GIS data can be visualized in two ways. They are paper maps and digital web GIS maps. Paper map contains a fixed resolution and also difficult to carry along with the client and the paper map is outdated and there is no possibilities to update. The linear measurement is not accurate and the area measurement is not applicable. The above limitations led to the development of web GIS map. The digital web GIS is made user friendly and also gives mobile access to data. The user requirements can be satisfied by developing more applications. These applications are developed by functional tools such as zoom in, zoom out, edit, measure, layer switch. In this study, two modules i.e live and non live module were developed. Live module is implemented by open source software (Geoserver, Post GIS, PostgresSQL, Geoexplorer). The live applications are hosted in cloud. The non live module is developed using the open layer Java Script. Postgres SQL is powerful open source database which permits multiple users to access the data simultaneously. Multiple users can manage and manipulate a single database. This software is secured because the privilege is granted only to authorized users by the admin.

10.3 CREATION OF LAYERS IN GEOSERVER

The overview of the developed of live module is shown in Figure 10.1 through the flowchart.
To display spatial data in latest web browser, Java script library is used in the open layer. The open layer documentation includes frontend, layers, controls, formats, overlays, styling, mobile browsing, deploying, requesting remote data and spherical Mercator. Here some of the sample coding are explained.
var map = new ol.Map({
layers: [
new ol.layer.Tile({
source: new ol.source.MapQuest({layer: 'sat'})
}),
new ol.layer.Image({
source: new ol.source.ImageVector({
source: new ol.source.GeoJSON({
projection: 'EPSG:3857',
url: 'data/geojson/countries.geojson'
}),
style: new ol.style.Style({
fill: new ol.style.Fill({
color: 'rgba(255, 255, 255, 0.6)'
}),
stroke: new ol.style.Stroke({
color: '#319FD3',
width: 1
})
})
}),
],
target: 'map',
view: new ol.View({
center: [0, 0],
zoom: 1
})
});
var featureOverlay = new ol.FeatureOverlay({
    map: map,
    style: new ol.style.Style({
        stroke: new ol.style.Stroke({
            color: '#f00',
            width: 1
        })),
        fill: new ol.style.Fill({
            color: 'rgba(255,0,0,0.1)'
        })
    })
});

var highlight;
var displayFeatureInfo = function(pixel) {

    var feature = map.forEachFeatureAtPixel(pixel, function(feature, layer) {
        return feature;
    });

    var info = document.getElementById('info');
    if (feature) {
        info.innerHTML = feature.getId() + ': ' + feature.get('name');
    } else {
        info.innerHTML = ' ';
    }

    if (feature !== highlight) {
        if (highlight) {
            featureOverlay.removeFeature(highlight);
        }
        highlight = feature;
        featureOverlay.addFeature(feature);
    }

    if (feature) {
        info.innerHTML = feature.getId() + ': ' + feature.get('name');
    } else {
        info.innerHTML = '&nbsp;';
    }

    if (feature !== highlight) {
        if (highlight) {
            featureOverlay.removeFeature(highlight);
        }
        highlight = feature;
        featureOverlay.addFeature(feature);
    }
}
if (feature) {
    featureOverlay.addFeature(feature);
}

highlight = feature;

};

map.on('pointermove', function(evt) {
    if (evt.dragging) {
        return;
    }
    var pixel = map.getEventPixel(evt.originalEvent);
    displayFeatureInfo(pixel);
});

map.on('click', function(evt) {
    displayFeatureInfo(evt.pixel);
});

The above mentioned input data are the layers to be created as open layer

var source = new ol.source.MapQuest({layer: 'sat'});

var layer = new ol.layer.Tile();

var map = new ol.Map({}
layers: [layer],
renderer: exampleNS.getRendererFromQueryString(),
target: 'map',
view: new ol.View({
center: [0, 0],
zoom: 2
});

document.getElementById('set-source').onclick = function() {
  layer.setSource(source);
};

document.getElementById('unset-source').onclick = function() {
  layer.setSource(null);
};

Controls are the features that are used in this study. The features used are Zoom in, Zoom out, Drag and measure.

/**
* Helper method for map-creation.
*
* @param {string} divId The id of the div for the map.
*/
var createMap = function(divId) {
    var source, layer, map, zoomslider, resolutions;

    source = new ol.source.MapQuest({layer: 'sat'});
    layer = new ol.layer.Tile({
        source: source
    });
    map = new ol.Map({
        layers: [layer],
        target: divId,
        view: new ol.View({
            center: [0, 0],
            zoom: 2
        })
    });
    zoomslider = new ol.control.ZoomSlider();
    map.addControl(zoomslider);
    return map;
};
var map1 = createMap('map1');
var map2 = createMap('map2');
var map3 = createMap('map3');

10.5 COLOR CODE

var categories_adminboundary = {
    "Airport.": [new ol.style.Style({
        stroke: new ol.style.Stroke({
            color: "rgba(0,0,0,1.0)",
            lineDash: null,
            width: 0
        }),
        fill: new ol.style.Fill({
            color: "rgba(224,38,221,1.0)"
        })
    }],
    "B1 Bazaar P.S.": [new ol.style.Style({
        stroke: new ol.style.Stroke({
            color: "rgba(0,0,0,1.0)",
            lineDash: null,
            width: 0
        }),
        fill: new ol.style.Fill({
            color: "rgba(218,84,12,1.0)"
        })
    })]
"B2 R.S. Puram P.S.": [new ol.style.Style({
    stroke: new ol.style.Stroke({
        color: "rgba(0,0,0,1.0)",
        lineDash: null,
        width: 0
    }),
    fill: new ol.style.Fill({
        color: "rgba(168,130,233,1.0)"
    })
})],

"B3 Variety Hall Rd P.S.": [new ol.style.Style({
    stroke: new ol.style.Stroke({
        color: "rgba(0,0,0,1.0)",
        lineDash: null,
        width: 0
    }),
    fill: new ol.style.Fill({
        color: "rgba(72,165,218,1.0)"
    })
})],

"B4 Ukkadam P.S.": [new ol.style.Style({
    stroke: new ol.style.Stroke({
        color: "rgba(0,0,0,1.0)",
        lineDash: null,
        width: 0
    }),
    fill: new ol.style.Fill({
        color: "rgba(72,165,218,1.0)"
    })
})]
"C3 Saibaba Colony P.S.": [new ol.style.Style({
    stroke: new ol.style.Stroke({
        color: "rgba(0,0,0,1.0)",
        lineDash: null,
        width: 0
    }),
    fill: new ol.style.Fill({
        color: "rgba(217,96,134,1.0)"
    })
})],

"C4 Rathnapuri P.S.": [new ol.style.Style({
    stroke: new ol.style.Stroke({
        color: "rgba(0,0,0,1.0)",
        lineDash: null,
        width: 0
    }),
    fill: new ol.style.Fill({
        color: "rgba(210,180,114,1.0)"
    })
})]

var textStyleCache_factor = {}
var clusterStyleCache_factor = {}
var selectedClusterStyleCache_factor = {
var style_factor = function(feature, resolution) {

    if (feature.hide === true) {
        return null;
    }

    var value = feature.get("Type");
    var style = categories_factor[value];
    var selectionStyle = categoriesSelected_factor[value];
    var labelText = "";
    var key = value + "_" + labelText

    if (!textStyleCache_factor[key]) {
        var text = new ol.style.Text({
            font: '16.5px Calibri,sans-serif',
            text: labelText,
            fill: new ol.style.Fill({
                color: "rgba(0, 0, 0, 255)"
            })
        });
        textStyleCache_factor[key] = new ol.style.Style({
            "text": text
        });
    }

    Buffer 10,m

    var textStyleCache_factor1mbuffer = { }

var clusterStyleCache_factor1mbuffer = {};
var selectedClusterStyleCache_factor1mbuffer = {};
var style_factor1mbuffer = function(feature, resolution) {

if (feature.hide === true) {
    return null;
}

var value = ""

var style = [new ol.style.Style({
stroke: new ol.style.Stroke({
color: "rgba(175,179,138,1.0)",
lineDash: null,
width: 0
}),
fill: new ol.style.Fill({
color: "rgba(241,244,199,1.0)"
})
})];

var selectionStyle = [new ol.style.Style({
stroke: new ol.style.Stroke({
color: "rgba(255, 204, 0, 1)",
lineDash: null,
width: 0
}),
fill: new ol.style.Fill({
})]

var selectionStyle = [new ol.style.Style({
stroke: new ol.style.Stroke({
color: "rgba(255, 204, 0, 1)",
lineDash: null,
width: 0
}),
fill: new ol.style.Fill({
})];
color: "rgba(255, 204, 0, 1)"

})
]));
var labelText = "";
var key = value + "_" + labelText

if (!textStyleCache_factor1mbuffer[key]) {
var text = new ol.style.Text({
    font: '16.5px Calibri,sans-serif',
    text: labelText,
    fill: new ol.style.Fill({
        color: "rgba(0, 0, 0, 255)"
    })
});

 textStyleCache_factor1mbuffer[key] = new ol.style.Style({
    "text": text
});
}

var allStyles = [textStyleCache_factor1mbuffer[key]];
var selected = lyr_factor1mbuffer.selectedFeatures;
if (selected && selected.indexOf(feature) !== -1) {
    allStyles.push.apply(allStyles, selectionStyle);
} else {
    allStyles.push.apply(allStyles, style);
}
return allStyles;
};

Buffer 25 m
var textStyleCache_factor25mbuffer = {};
var clusterStyleCache_factor25mbuffer = {};
var selectedClusterStyleCache_factor25mbuffer = {};
var style_factor25mbuffer = function(feature, resolution) {

if (feature.hide === true) {
    return null;
}
    var value = ""
var style = [new ol.style.Style({
stroke: new ol.style.Stroke({
color: "rgba(114,133,132,1.0)",
lineDash: null,
width: 0
}),
fill: new ol.style.Fill({
color: "rgba(165,191,221,1.0)"
})
})];
var selectionStyle = [new ol.style.Style({
stroke: new ol.style.Stroke({
color: "rgba(114,133,132,1.0)"
})
})];
color: "rgba(255, 204, 0, 1)",
lineDash: null,
width: 0
});
fill: new ol.style.Fill({
color: "rgba(255, 204, 0, 1)"
})
});

var labelText = "";
var key = value + "_" + labelText
if (!textStyleCache_factor25mbuffer[key]) {
var text = new ol.style.Text({
font: '16.5px Calibri,sans-serif',
text: labelText,
fill: new ol.style.Fill({
color: "rgba(0, 0, 0, 255)"
}),
});

textStyleCache_factor25mbuffer[key] = new ol.style.Style({
"text": text
});
}

var allStyles = [textStyleCache_factor25mbuffer[key]];
var selected = lyr_factor25mbuffer.selectedFeatures;
if (selected && selected.indexOf(feature) != -1) {

allStyles.push.apply(allStyles, selectionStyle);
    } else {
allStyles.push.apply(allStyles, style);
    }
return allStyles;
};

HALLO

var textStyleCache_factor1mbuffer = {};
var clusterStyleCache_factor1mbuffer = {};
var selectedClusterStyleCache_factor1mbuffer = {};
var style_factor1mbuffer = function(feature, resolution) {
if (feature.hide === true) {
return null;
    }
    var value = ""
var style = [new ol.style.Style({
stroke: new ol.style.Stroke({
    color: "rgba(0,0,0,0.0)",
    lineDash: null,
    width: 0
    })),
    fill: new ol.style.Fill({
    color: "rgba(0,0,0,0.0)"
    })}]}
var selectionStyle = [new ol.style.Style({
  stroke: new ol.style.Stroke({
    color: "rgba(255, 204, 0, 1)",
    lineDash: null,
    width: 0
  })),
  fill: new ol.style.Fill({
    color: "rgba(255, 204, 0, 1)"
  })
})];

var labelText = "";
var key = value + "_" + labelText

if (!textStyleCache_factor1mbuffer[key]) {
  var text = new ol.style.Text({
    font: '16.5px Calibri,sans-serif',
    text: labelText,
    fill: new ol.style.Fill({
      color: "rgba(0, 0, 0, 255)"
    })),
  })
  textStyleCache_factor1mbuffer[key] = new ol.style.Style({
    "text": text
  });
Using the javascript a new accident spot can be added by authorized user is shown in Figure 10.2

![Figure 10.2 Addition of new accident spot](image-url)
10.6 CHOROPLETH MAPPING

10.6.1 GENERAL

Choropleth Maps display divided geographical areas or regions that are coloured, shaded or patterned in relation to a data variable. This provides a way to visualise values over a geographical area, which can show variation or patterns across the displayed location. The data variable uses colour progression to represent itself in each region of the map. Typically, this can be a blending from one colour to another, a single hue progression, transparent to opaque, light to dark or an entire colour spectrum. A common error when producing Choropleth Maps is to encode raw data values (such as population) rather than using normalized values (calculating population per square kilometre for example) to produce a density map.

10.6.2 Characteristics of choroplethic maps

The areas, to which the ratios are related, are styled with different fill colours or patterns. The denser an object is distributed within an area the darker the colour or the more dense the pattern should be.

10.6.3 Types of choroplethic maps

Choropleth maps are classified by their reference areas:

- Choroplethic maps based on administrative areas.
- Dasymetric maps show densities of areas regarding geographic properties.
- Choroplethic maps in relation to geometric areal units such as grid squares.
**Advantages of Choropleth maps**

- Levels of shading/colour represent a range of values
- Visually effective - can see a large amount of information and general patterns
- Groupings can be flexible to accommodate the spread of values

### 10.6.4 Choropleth Map Analysis

In this research Choropleth maps are created for various purposes. This web based Choropleth map will be very useful for administrators and decision makers. This web applications, can be used for getting the statistical report given below:

- No of accidents happen within the jurisdiction of police stations
- Classification of police station base on accident noted
- Type of accidents

The following features were considered while developing the Choropleth map using java script:

1) Overall Accidents
2) Working Days Accident
3) Holidays Accident
4) Peak Hour Accident
5) Non Peak hour Accident
6) Number of Injuries in Accident
7) Number of Deaths
8) Number of Safe persons
9) Four wheeler accident
10) Two wheeler Accident
11) Male Accident  
12) Female Accident  
13) Accident in National Highways  
14) Accident in State Highways  
15) Accidents in Corporation Road  

10.6.5 Classification Method  

Choropleth maps were classified into three types of classification:

- Equal interval classification  
- Natural breaks classification  
- Quantile classification  

10.6.6 Equal Interval Classification  

Equal Interval Classification are found by determining the range of the data. The range is then divided by the number of classes, which gives the common difference. The class limits are established by starting at the lowest value and adding the common difference to the upper limits of the first class, adding the common difference to this to get the limit of the second class, until the upper limit of the data is reached. Equal interval is useful when distribution of the data has a rectangular shape in the histogram. However, in geography, equal interval is most common when the classification units are nearly equal in size.  

10.6.7 Natural Breaks Classification  

Natural breaks classes are based on natural groupings inherent in the data. The features are divided into classes whose boundaries are set where
there are relatively big differences in the data values. Natural breaks are data maps built from different underlying information. This classification is based on the Jenks Natural Breaks algorithm.

10.6.8 Quantile Classification

Each class contains an equal number of features. A quantile classification is well suited to linearly distributed data. Quantile assigns the same number of data values to each class. There are no empty classes or classes with too few or too many values. Because features are grouped in equal numbers in each class using quantile classification, the resulting map can often be misleading. Similar features can be placed in adjacent classes, or features with widely different values can be put in the same class. During the present research, all the three types of Choropleth maps are developed and the user can choose the type of map he/she wants. The Choropleth map for police station-wise accident density values is shown in Figure 10.3

![Figure 10.3 Web Choropleth Map for police station wise Accident Density](image)

From the Figure, the police stations associated with maximum number of accidents and minimum number of accidents can be seen.