E-CAD: WEB-BASED INFORMATION SERVICE FOR LAND MANAGEMENT

M. Neupane, R. Jaiswal, R. Khati*, S. Dhakal, S. Sharma

Department of Civil and Geomatics Engineering, Pashchimanchal Campus, IOE, TU, Pokhara, Nepal,- (ruinner17, rajapkgk, rhtkhati, ddsandyya, shraddha.sharma2100)@gmail.com

Commission V, WG V/7 & Commision IV, WG IV/6

KEY WORDS: Dynamic Web Platform, Spatial, GeoDjango, Mapping Server, Administrators, Spatial Operation

ABSTRACT:

The status quo of the land management and information system in Nepal is a far cry from where the developed world stands. Paper-based system is still the spine of this system which is tedious, less accurate and difficult to store and update. So, there is a need for a digitized system providing country's land authorities with a powerful tool that creates a unified platform for the creation of an accurate spatial database, timely maintenance and updating of the database in a dynamic web platform. The database including spatial data of land features can be utilized to create a web-based thin-client mapping application which meets the needs of all the stakeholders (government authorities from Survey Offices, Land Revenue Offices, Land Reform Offices, etc.). This paper shows the use of open-source software for the creation of a web mapping system including QGIS, PostGIS extension of PostgreSQL for spatial database, HTML for markup, CSS for styling, JavaScript, Leaflet, Open-Layers for client-side scripting and Geo-Django for backend designing. The developed methodology can be utilised for the preparation of an interactive thin-client web mapping server that enables general users to dynamically view data, zoom, pan and search from the database and obtain land information. The login system enables administrators with various access to upload, verify and edit data along with performing various spatial operations while the super admin is entitled to access to the PostgreSQL database. The major finding is that the use of a thin client application for a land information system is beneficial for all stakeholders. It is also a measure of the performance of land authorities allowing better planning, preparedness, and allocation of resources.

1. INTRODUCTION

Land is important because humans not only live but also perform all economic activities on land. Besides, Land also supports wildlife, natural vegetation, transport, and communication activities. Most of our basic needs and requirements are obtained from land. Land is a nonincreasing factor of production that houses an increasing population. A great percentage of man's activities depend on land. The quests for land ownership, registration, transfer of ownership, etc.are concepts that affect land management. Estimations show that about 70% of people-land relationships worldwide are not documented, whereas the population grows and the pressure on land and natural resources increases. These results in many land conflicts and competing claims on land. .

Land Information System can be defined as a geographical information system for cadastral and land-use mapping, typically used by local governments. It consists of land records and associated attributes as well as spatial information. Land records include land resources, land use, environmental impact, and fiscal data. It basically deals with legal boundaries of land tenure. LIS provides a base layer capable of integration into other geographic systems. Mainly LIS uses cadastre as a primary document. Hence, it consists of two components i.e. spatial data and non-spatial data. Spatial data refers to the data related to location or position of an object in geographic data e.g. map whereas non-spatial data describes the spatial data e.g. field books, registers, etc. LIS considers land parcels as key features. The parcel is only considered as a property after the rights and restrictions acclaimed to the owner. LIS guarantees land ownership and security of tenure. Concept of LIS and the cadastral system was first developed in Europe to support taxation. The Western LIS concept nourished the multipurpose role of the cadastral system. The Dutch Cadastral System enabled the web to the parcel-based system. It consists of a pre-modeled database with ISO standards. The system is capable enough to handle spatial and attribute data in web browsers. It has a Web-based query facility as well as a login facility for data security. It also combined an online verification facility (Tuladhar, 2005). The concept of LIS was first introduced in Nepal in the 8th periodic plan by the government. The Implementation of Land Information System in Nepal Project was carried out, in close co-operation with the Swedish counterpart Swede survey. The map-based land recording system was started after the establishment of Cadastral Survey in Bhaktapur district in 1980 B.S. . The Survey Goswara was established in Kathmandu in 1996 B.S. The Survey Department and the Department of Land Revenue were established respectively in 2014 B.S. and 2016 B.S. After the Land Survey and Measurement Act, 2019 B.S., came into effect, the maintenance of a map-based land records system was taken into practice. The general objectives of this act were preparation of up to date land-ownership records that were essential for the collection of land revenues, collection of the tenants and land mensuration works (esp. needed for land reformation program). In the meantime, the software was designed named DLIS to acquire and manage non-spatial data(Ministry of Land Management, 2019).

The advent of the Internet has seen a number of Geographic Information Systems utilizing its potential to disseminate Geographic Information. Web-based GIS comprises of relatively small pieces of software or components, which perform particular GIS operations, namely Cartographic Visualization. However, it is still a tedious task for the collection and maintenance of the database due to a multi-tier system for data acquisition. In the current day and age where the internet has vastly simplified this process, the old multi-tier system is a hindrance to an effective management system. Thus, a thin client-based application is best suited for this purpose. This also serves as a justification for the selection of web-based thin-client

* Corresponding author

application creation as described in this paper. The overall paper describes the indispensability of a web-based cadastral information management system in ensuring a much more informative and users' participatory cadastral information system for land management.

2. METHODOLOGY

The succeeding sub-sections describe all the procedures required for developing digital information system. Figure 1 summarizes all the work procedure followed to develop the system. Additional geo-data like orthophotos, digital elevation model (if available) along with development plans are needed for the betterment of the system.

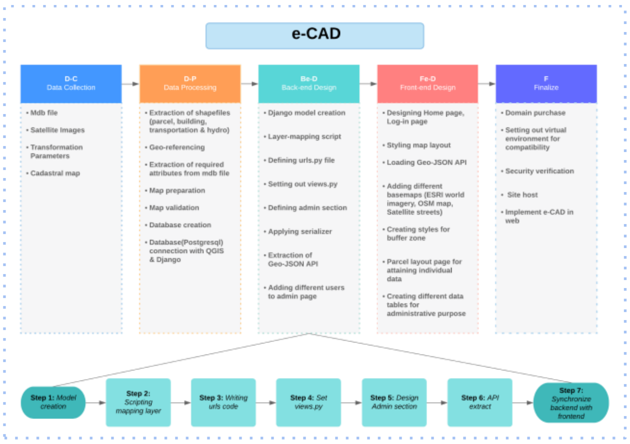

e-CAD
D-C Data Collection
• Mdb file
• Satellite Images
• Transformation Parameters
• Cadastral map
D-P Data Processing
• Extraction of shapefiles (parcel, building, transportation & hydro)
• Geo-referencing
• Extraction of required attributes from mdb file
• Map preparation
• Map validation
• Database creation
• Database(Postgresql) connection with QGIS & Django
Be-D Back-end Design
• Django model creation
• Layer-mapping script
• Defining urls.py file
• Setting out views.py
• Defining admin section
• Applying serializer
• Extraction of Geo-JSON API
• Adding different users to admin page
Fe-D Front-end Design
• Designing Home page, Log-in page
• Styling map layout
• Loading Geo-JSON API
• Adding different basemaps (ESRI world imagery, OSM map, Satellite streets)
• Creating styles for buffer zone
• Parcel layout page for attaining individual data
• Creating different data tables for administrative purpose
F Finalize
• Domain purchase
• Setting out virtual environment for compatibility
• Security verification
• Site host
• Implement e-CAD in web
Step 1: Model creation
Step 2: Scripting mapping layer
Step 3: Writing urls code
Step 4: Set views.py
Step 5: Design Admin section
Step 6: API extract
Step 7: Synchronize backend with frontend

Figure 1. Workflow diagram

2.1 Data Collection

Data is a fundamental part of any management tool. Spatial information with its attribute value was collected in .mdb format. Verification of data is a must. For that, satellite images along with OSM were collected. Cadastral plan for area of interest provided the validation of information obtained from .mdb file.

2.2 Data Processing

- Shapefiles were extracted from Microsoft Access Database File (.mdb) format using Microsoft Access Database Engine in QGIS.
- Extracted shapefile had Nepal_87 Transverse Mercator coordinate system. The coordinate system was changed to WGS_1984_UTM_Zone_45N and spatial adjustment for each layer of shapefile was performed.
- Unwanted information was removed from the attribute table.

2.3 Map Preparation

The map to be published on the website was prepared from shapefiles (parcel.shp, hydro.shp, building.shp, transportation.shp) obtained from Microsoft access database (.mdb) format. Using QGIS, which is a free and open-source cross-platform desktop geographic information system application that supports viewing, editing, and analysis of geospatial data, map was prepared using various tools. OpenStreetMap was added as a base map for the verification of the data.Before that, we also connected QGIS with our database so that any changes in data made on QGIS could be automatically modified in the database system.

2.4 Back-end design

Back end deals with the inner mechanism of data along with the front end. It is used to store, create, manipulate the data. Django, a high-level Python Web framework, was used for the creation of the backbone of the website. A project with the name 'e-CAD' was created along with applications 'pages' & 'shapefiles'. First one was for creating the front end of the website & later one was for processing the shapefiles.

- Database was connected with Django by defining the engine required. Spatial database was connected with the Django project as well as with QGIS.
- Apps required for project, for example: third party apps and own created apps were defined in 'settings.py'.
- The 'admin.py' file was used to display models in the Django admin panel. Also, admin panel can be customized using 'admin.py' file.
- Admin homepage can be accessed by entering URL 'http://127.0.0.1:8000/admin/' or 'http://localhost:8000/admin/'.
- For each shapefile, models were prepared. Models are the definitive source of information for our data. They contain the essential fields and behaviors of the data used. Each model maps to a single database table.
- Unicode_literals was used to speed up the porting process. Models were imported from a database of Django. MultiPolygonField, not being supported by 'django.db', 'gis_models' was imported from 'django.contrib.gis.db'.
- A Parcel class including various fields with field type id was defined. Field type filters the exact type of data required by field. Likewise, Building, Transportation & Hydro classes were also defined along with various fields and their types.
- To import data, Layer-mapping was used. It helped us to load all the information stored in a shapefile to the model developed. All the function interacting with the operating system was imported by import 'import os'. The 'django.contrib.gis.utils' module contains various utilities that are useful in creating geospatial Web applications. One of them is the LayerMapping class that provides a way to map the contents of vector spatial data files into GeoDjango models.
- Each key in the 'parcel_mapping' dictionary corresponds to a field in the 'Parcel' model. The value is the name of the shapefile field that data will be loaded from. The location of 'parcel.shp' was provided by defining 'parcel_shp'.
- All fields which correspond to each other were imported with layermapping.
- Similarly, 'building_mapping', 'transportation_mapping' & 'hydro_mapping' were defined for importing fields from respective shapefiles.
- All the basic requirements for setting up back-end like URL, ADMIN, VIEWS were set up.
- The system is supposed to be dynamic. Dynamic in the sense that our model should update the information if we change something in frontend.
- Also, for data to function and be processed according to the requirement of user , we changed our shapefile to GeoJSON.

- Serializers allow complex data such as query sets and model instances to be converted to native Python data types that can then be easily rendered into JSON, XML or other content types.
- Doing so, we were able to achieve API. API is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

2.5 Front-end design

Front-end web development is the practice of converting data to graphical interface for the user to view and interact with data through digital interaction using HTML, CSS, J-QUERY, TURF & JavaScript.

- Front-end design started by designing the homepage.
- In login page, three different login sections were made for different kinds of users: administrative users, authentic users and normal users.
- Three different pages were developed for redirecting each users after login which display the parcel including building feature, hydro feature, transportation feature along with Satellite_streets map as default with:
  default zoom:16
  default bound: [[27.63404, 85.51547],[27.62401, 85.52434]]
- Icons were prepared for various functions like zoom in, zoom out, geolocation, measure, search and buffer.
- Geolocation function shows current location of user and measure tool gives coordinates of a point, distance between two drawn points and area of drawn polygon.
- Floating widget was made for selection of base map and map layers.
- Buffer function was made for buffering transportation & hydro feature with default buffer radius of 5m.

After designing both back-end & front-end, they should be connected. It should be ensured that every element gets connected so that the front-end shows exactly the same feature it asks for. By using leaflet plugins, HTML & CSS, web pages were created that display map feature, login pages and home pages.

3. DEVELOPED WEB GIS PLATFORM

Following items provide a brief description of the interactive web map application thus created:

- While accessing 'localhost:8000/home' or http://127.0.0.1:8000/home, the user is directed to homepage containing button labelled with 'e-CAD'.
- After clicking the button, one is directed towards the login section. There are three login sections, each for three different kinds of users as seen in figure 2.
- Considering data security, each type of users have been provided with different permissions. A base map has been added for the user's convenience. Users can select among different base maps. Similarly, there is a choice of selection of data layers too.

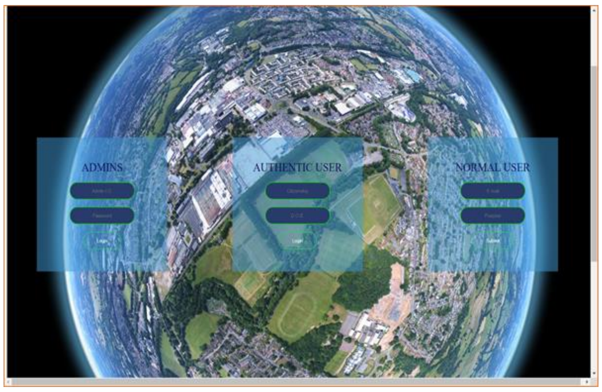


Figure 2. Login Sections for different kind of users

- Admin users can see all the information about the features included. They can use spatial functions and also edit the data.They can update, split or add the information of features. If required they are able to create new features too.

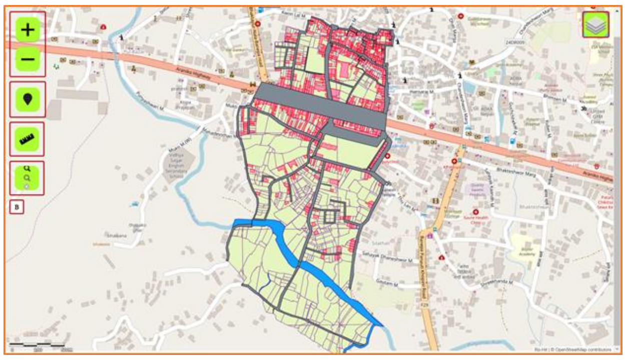


Figure 3. Admin View of the webpage.

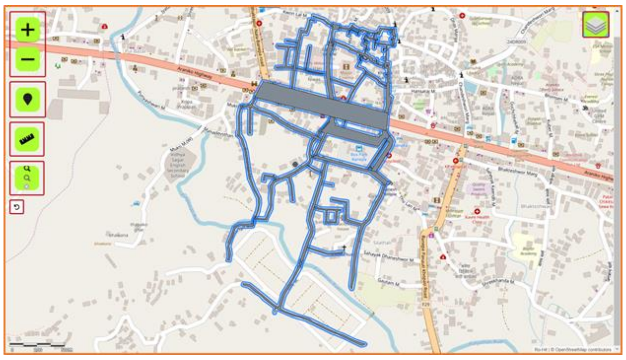


Figure 4. Buffering function applied to transportation feature (5m)

- Some spatial functions like zoom in, zoom out, buffering(default radius of 5m), geocoding, area calculations, distance calculations, coordinate view, search window, scale bar along with mini-map have been introduced.
- When a feature in the map is clicked, a popup window appears with detailed information about that feature as shown in figure 5. It shows the information of parcel along with its area, east parcel, west parcel, north parcel, south parcel, parcel key.

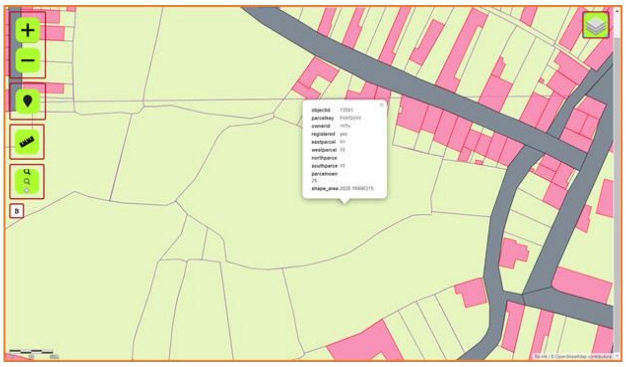


Figure 5. Operation to locate the position of the clicked point

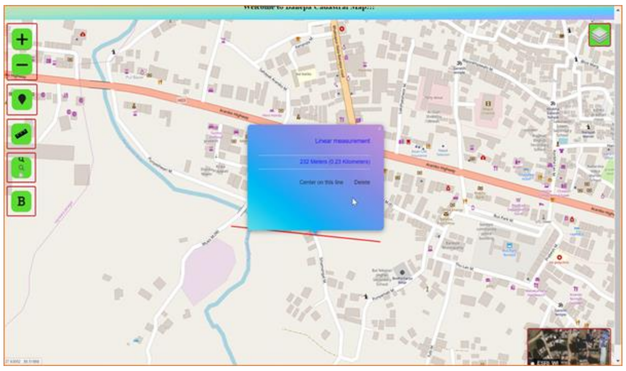


Figure 6. Distance between two drawn points on the map

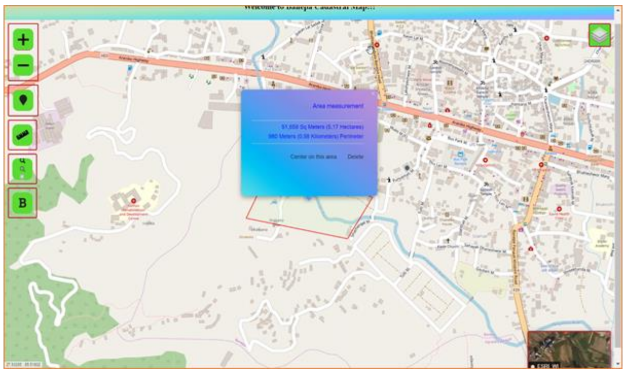
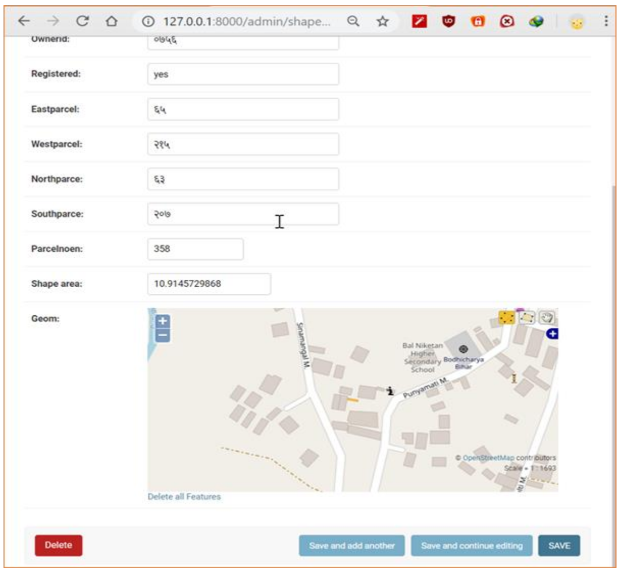


Figure 7. Area of the drawn polygon on the map

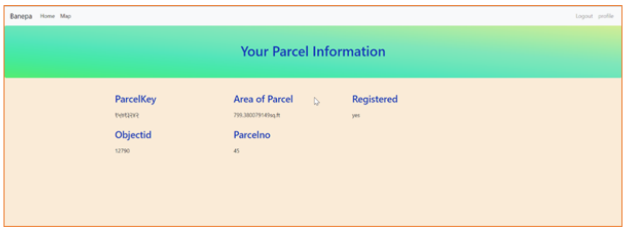
- Data can be added from the admin site and all the required information can be saved.
- Building, transportation and hydro also can be added/modified in a similar way as in figure 8.
- Only administrative members can access the admin page.
- An interactive secondary homepage has been added so that admin can visit the page they require.
- Authentic users are able to see their information i.e. one can obtain detailed information of the parcel only if he owns it.
- Authentic users have to use their citizenship number and date of birth to log in.
- Normal users can just view the map. They can neither use spatial functions nor obtain detailed information.
- Normal users don't need any ID to login. They only need to submit the purpose form for their visit to the website.



127.0.0.1:8000/admin/shape...

Ownerid:	०७५६
Registered:	yes
Eastparcel:	६५
Westparcel:	२९५
Northparce:	६३
Southparce:	२०७
Parcelnoen:	358
Shape area:	10.9145729868
Geom:	

Figure 8. Adding/ Modifying parcel data to the database



Your Parcel Information

ParcelKey	Area of Parcel	Registered
[illegible]	799.380079149sq.ft	yes
Objectid	**Parcelno**	
12790	45	

Figure 9. Viewing information by authentic user

4. CONCLUSION

Management of land information is imperative at present in Nepal. Hence, Web-based and Information System provides the country's land authorities with a powerful tool which creates a unified platform for all the stakeholders and concerned individuals and organizations which can be utilized for creation of accurate spatial and non-spatial database, timely maintenance and updating of the database in a dynamic web platform. It also provides government officials with easier analysis of geospatial data. This system implements the OGC Web Services: WMS and WFS in order to achieve the visualization of static and dynamic spatial data respectively and the various spatial operations on those data to perform simple spatial and statistical analysis. This paper is an example of how better data dissemination can be achieved of spatial and non-spatial datasets for various stakeholders with different access. It is also a measure of the performance of land authorities allowing better planning, preparedness, and allocation of resources. Alongside with this, land valuation, utility management, land taxation can be appended in this system. Data dissemination to the general people also would be easier for both the stakeholders and public.

REFERENCES

Anaconda Software Distribution, 2019. Computer Software, Version 3 anaconda.com

Andersson, S., 1981. *LIS, What is that? - An Introduction*. FIG 301.1, XVI International Congress of Surveyors, Montreux

Bernhardsen, T., 1999. *Geographical Information Systems: An Introduction* , New York, John Wiley & Sons, Inc.

Burrough, P.A., 1986. *Principles of Geographical Information Systems for Land Resources Assessment*, Clarendon, Oxford. pp 13- 15.

Django Development Team, Django Software Foundation, Version 2.2., djangoproject.com

Ministry of Land Management, C. a., 2019. Molrm.gov.np

Mohamed, M. and S. Ventura, 2000. *Use of Geomatics for Mapping and Documenting Indigenous Tenure Systems*, *Society & Natural Resources*, vol. 13, pp. 223 - 235.

Python Software Foundation, Python Language Reference, Version 3.7, python.org

QGIS Development Team (2018), Quantum Geographical Information System (QGIS) Version 3.8, qgis.osgeo.org

Swedesurvey. 2002. Final Report of Implementation of LIS in Nepal: Swedesurvey, LISP, HMG/Nepal. Kathmandu.

Tuladhar, A. 2005: Towards Strategic Planning For Building Land Information System (LIS) in Nepal.

APPENDIX

Map Interface

The map can be added using a leaflet. Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Leaflet content delivery network (CDN) can be started in our HTML file as:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js"></script>
```

Division of map can be stated as:

```
<div id="map"></div>
```

Styling of map division is followed using cascading style sheet (CSS) as:

```
<style>
     #top{
    width:100%;
    height: 5%;
    background-image: linear-gradient(to right bottom, ■#d3dbb5, ■#c
    }

html, body, #map {
    width: 100%;
    height: 100%;
    padding: 0;
    margin: 0;
    float: right;
}
</style>
```

Mapping parameters like zoom control, maximum & minimum zoom level as well as bounding is set as:

```
var map = L.map('map', {
    zoomControl:true,
    maxZoom:28,
    minZoom:1,
}).fitBounds([[27.63404, 85.51547],[27.62401, 85.52434]]);
```

Now, the shapefiles that are imported in databases from Django models by running ‘load_shp.parcel_run()’ are added to the map. The creation of ‘.json’ file made it possible to pull API wherever required. It is then bounded and added to layer as:

```
var layer_shapefiles_parcel_0 = new L.GeoJSON.AJAX("{% url 'parcel' %}", {
    attribution: '',
    pane: 'pane_shapefiles_parcel_0',
    onEachFeature: pop_shapefiles_parcel_0,
    style: style_shapefiles_parcel_0_0,
});
bounds_group.addLayer(layer_shapefiles_parcel_0);
map.addLayer(layer_shapefiles_parcel_0);
```

Similarly, building, transportation & hydro is added to layer as done for parcel.

For operating buffering on a transportation feature of 5m, a button is created which buffers the feature on click. It is removed by double-clicking the same button.

```
var toggle = L.easyButton({
    states: [{
      stateName: 'add-buffer',
      icon: 'fa-bold',
      title: 'Buffer',
      onClick: function(control) {
        const incidences2 = layer_shapefiles_transportation_3.toGeoJSON();
        var buff=turf.buffer(incidences2,0.005,{'unit':'kilometers'});
        bufferLayer=L.geoJSON(buff).addTo(map);
        control.state('remove-buffer');
      }
    }, {
      icon:'fa-undo',
      stateName:'remove-buffer',
      title:'Remove Buffer',
      onClick: function(control) {
       map .removeLayer(bufferLayer);
       control.state('add-buffer');
      },

    }]
  });
```

A small division is added to the bottom right side of the map. It shows the ESRI World Imagery of the place which is represented by cursor in main map division.

```
L.control.mousePosition().addTo(map);
var baseMap = {
  "ESRI_WI": Esri_WorldImagery
    };
L.control.mousePosition().addTo(map);
L.control.layers.minimap(baseMap, {}, {collapsed: false}).addTo(map);
```

To add a base map layer in our map, variables must be initialized for the required number of base maps.

```
var Esri_WorldImagery = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/servic
    OpenStreetMap_Mapnik = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.
    Satellite_streets = L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.p
                        maxZoom: 28,
                        id: 'mapbox.satellite',
                        accessToken: 'pk.eyJ1Ijoicm9oaXQ4MSIsImEiOiJjano0aXFlNjAwZGsyM2
                    });
Satellite_streets.addTo(map);
var baseMaps = {
    "ESRI_WI": Esri_WorldImagery,
    "OSM": OpenStreetMap_Mapnik,
    "Satellite_streets": Satellite_streets
};
L.control.layers(baseMaps,{"<img src= {% static 'images/transportation.png'%}/> Transpo
                              "<img src={% static 'images/hydro.png' %} /> Hydro": layer_
                              "<img src={% static 'images/building_2.png' %}/> Building":
                              "<img src={% static 'images/images_2.png' %}/> Parcel": lay
}).addTo(map);
setBounds();
L.control.betterscale().addTo(map);
```