**Real Time Vehicle Tracking System**

**MTech GIS (2012-13)**



**Submitted by**

**K Venkata Naresh**

**Narahari T**

**Umapathi K**

**KrishnaMohanRaju N**

**Table of Contents**

1. Introduction ………………………………………………………………………..…………… 3
2. Internship objectives ……………………………………………………………..……………. 6
3. Project Description ………………………………………………………………..…………… 8
4. Methodology ……………………………………………………………………..……………. 11
   1. Description …………………………………………………………………..……………. 11
   2. Data Sources ………………………………………………………………..……………… 12
   3. Methodology diagram ………………………………………………………..…………… 13
5. Results and Discussions ………………………………………………………..……………… 35
6. Learning outcomes ……………………………………………………………..……………… 40
7. Other information ……………………………………………………………..……………….. 40

**List of Tables**

Table 1 Compare among web map APIs ……………………………………………………… 8

Table 2 Data sources used in this project……………………………………………………..... 12

**List of Figures**

Figure4.1 Methodology Diagram ……………………………………………………………… 13

Figure5.1 (a, b) Final output of Task1 ……………………………………………………….. 35, 36

Figure5.2 (a, b) Final output of Task2 …………………………………………………………. 37

Figure5.3 (a, b) Final output of Task3 ……………………………………………………...... 38, 39

1. **Introduction:**

**About Company:**

Tech Mahindra has been committed to creating value for its customers, investors, associates and the society at large. This commitment has constantly fuelled the journey, encouraging us to think differently, work with passion and make things happen.

As a leading global information, communications and technology (ICT) company, we are part of the US $16.2 billion Mahindra Group, a global industrial federation of companies and one of the top 10 business houses based in India.

**Our Values:**

* Customer First
* Good Corporate Citizenship
* Individual Dignity
* Professionalism
* Quality Tools

**Services Offering by Tech Mahindra:**

Tech Mahindra helps clients with GIS development and Geospatial Data Management Services to improve the way they do business, with the back-up of expertise in domain competency, workforce management, quality initiatives and proven delivery models.

Our company provides high quality GIS Development and Geodata Processing services for customers around the world.

**GIS Development services** covers the GIS full life cycle and include consulting, pilot projects, database design and development, application design and development, enterprise systems integration, Internet GIS and implementation. GIS development team offers services such as,

* **Location-based services:** Customized application for fleet management, asset tracking & monitoring, route optimization, destination guides, location-based traffic and weather alerts
* **Spatial data management:** Preparation / updating of spatial data, analyzing existing dataset, database fine tuning, data conversion, modeling, integration services, projection and transformation.
* **GIS-enabled asset and facilities management:** Accessing and analyzing information available on the existing facilities and the ability to manage them effectively. AM/FM solutions are best suited for property management, space management, asset procurement, movement and tracking
* **GIS-enabled infrastructure planning services:** Orientation, land use planning, town & urban planning involving roads, service lines, utilities, etc.

**Geodata Processing Services**: Our data conversion, migration, maintenance and mapping services are at par with industry standards and processes. Tech Mahindra is enabled with infrastructure and resources for GIS (Geographic Information System) database Updation and data processing for various industries.

For Telecommunication

* Land use/land cover morphology data (used in wireless network planning)
* Locate, map and monitor cell site data by tower identification, latitude/longitude, location, frequency, height, owner
* Creation/Updation of Cable Network
* Creation/Updation of Facilities Mapping
* Telecom Records Conversion
* Telecom Network Migration

For Transportation

* Road Data Generation
* Improved/Enhanced routing guide with improved tracking
* Point Addressing Database for GPS Navigation System
* Enhanced Street Network Database for In-Car Navigation
* Electronic Navigational Chart Production
* Flight Obstacle Mapping and Airport Map Database Production
* Railway Asset Mapping
* Freight Transportation/Logistics database update

For Utilities

* Creation/Updation of Distribution Network - Power, Water and Gas
* Electric, Gas and Water Records Conversion
* Electric, Gas and Water Network Migration
* Data management of assets - Sensors, Poles, Conduit, Smart meters

Indicative Projects Delivered

* Database Updates
* POI capture
* Geo Coding
* Geo-repositioning
* Address Points and Parcel Mapping
* Data extraction of Topographic features
* Cartographic Map Creation
* Positional Accuracy Improvement

1. **Internship Objectives:**

**Tasks Assigned:**

1. **Geofencing:** Create a Geo fence on Google maps using JavaScript and pass alerts. Below are the rules while designing;
2. Geo Fence should be created through circle and polygon
3. User should be able to draw the polygon or circle in any location at runtime
4. Circle should be drawn at runtime, when user pass the radius and double clicks the map
5. Polygon should be drawn at runtime based on the user selected locations on map.
6. Alerts should be displayed based on the coordinates of the polygon and circle.
7. Retrieve all the Lat/long values of the specified direction path
8. Draw boxes on the specified direction path and retrieve the bounding coordinates. Pass the alerts based on the bounding coordinates.

**Trainings:**

The following trainings have been provided during the internship period:

1. C# concept (Collections – LINQ)
2. Basics of Web applications

**C# Concepts (Collections and LINQ):**

**Collection classes** are specialized classes for data storage and retrieval. These classes provide support for stacks, queues, lists, and hash tables. Most collection classes implement the same interfaces.

Collection classes serve various purposes, such as allocating memory dynamically to elements and accessing a list of items on the basis of an index etc. These classes create collections of objects of the Object class, which is the base class for all data types in C#.

**Language-Integrated Query (LINQ)** is a set of features introduced in Visual Studio 2008 that extends powerful query capabilities to the language syntax of C# and Visual Basic. LINQ introduces standard, easily-learned patterns for querying and updating data, and the technology can be extended to support potentially any kind of data store. It provides a general introduction to the kinds of applications that you can write and the kinds of problems that you can solve with LINQ queries.

**Basics of Web applications:**

Our company given training on basic concepts of web application like

* Definition of web application and its uses
* How web APIs are useful in web applications

Web APIs: - An application programming interface (API) is a set of rules and specifications that software programs can follow to communicate or ‘interface’ with each other. Web APIs enables direct access to the modules from the client side or a third party’s Website.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Javascript API** | **Silverlight API** | **Flex API** |
| **Software** | Visual Studio | Visual Studio | Eclipse |
| **Plug-ins** | Eclipse | Expression blend | Adobe flash builder |
| **Programming languages** | JavaScript | C# , VB.NET | ActionScript |
| **Web Designing** | HTML | Silverlight | Adobe Flex |
| **Pros & cons** | If browser have JavaScript , it does not get any compatible issues | Main drawback of this API is it should have Silverlight runtime plug-in | Main drawback of this API is it should have flash player |

Table1 Compare among web map APIs

1. **Project Description:**

**Introduction to Application: -**

GPS vehicle tracking system (GPS monitoring) is an electronic device installed in a vehicle to enable the owner or a third party to track the vehicle's location (and also receive the additional information about vehicle).

GPS monitoring systems also combine a communications component such as cellular or radio transmitters to communicate the vehicle’s location to a remote user. Vehicle information can be viewed on electronic maps by the specialized monitoring software.

**Active versus passive tracking:**

**Passive devices** store GPS location, speed, heading and others main information like Fuel level in fuel tank, event such as key on/off, door open/closed etc.. Once the vehicle returns to a predetermined point, the device is removed and the data downloaded to a computer for evaluation.

**Active devices** transmit the data in real-time via cellular or radio networks to a computer or data center for evaluation. Some devices also can collect the same information.

**Real Time Vehicle Tracking:** A real-time vehicle tracking system installed in any vehicle (car, bus, truck, boat, etc.) allows, owners, managers, logistic teams and finally loved ones to keep track of where their assets are at any time. Vehicle Tracking System is a turn-key solution for vehicle owners or fleet operators to determine the location of the vehicle. The solution includes Vehicle tracking unit which is installed in vehicles to collect the vehicle information and Vehicle tracking software which is a web based software application to monitor the current location of the vehicle along with other history details of the vehicle via internet

The solution allows users to monitor and control the movement of vehicles viz. trip start and completion time, distance traveled, speed etc. at any point of time. This helps in better fleet management by reducing the operating costs and while delivering mobile assets more effectively. The combination of GPS and cellular technology is used to transmit the vehicle data from unit to software.

The application is designed and developed with easy to use GUI and navigation. The real time reports help managers for better planning, scheduling and forecasting and thereby increasing the customer satisfaction.

**Features of Real Time Vehicle Tracking:**

* Complete Tracking Solution for individual Vehicle or complete Fleet
* Minute-by-minute updates on vehicle location, usage and status
* Manage Routes and Rules for Alerts
* Theft Protection
* Historical reports and real-time alerts
* Easy to use feature rich web based Application for access from any computer
* Useful MIS reports (Management Information System)

**Vehicle Tracking can be used in the following scenarios:**

* **Fleet Management:** When managing a fleet of vehicles, knowing the real-time location of all drivers allows management to meet customer needs more efficiently. Whether it is delivery, service or other multi-vehicle enterprises, drivers now only need a mobile phone with telephony or Internet connection to be inexpensively tracked by and dispatched efficiently. Asset Tracking: Companies needing to track valuable assets for insurance or other monitoring purposes can now plot the real-time asset location on a map and closely monitor movement and operating status.
* **Field Service Management:** Companies with a field service workforce for services such as repair or maintenance, must be able to plan field workers’ time, schedule subsequent customer visits and be able to operate these departments efficiently. Vehicle tracking allows companies to quickly locate a field engineer and dispatch the closest one to meet a new customer request or provide site arrival information.
* **Stolen Vehicle Recovery:** Both consumer and commercial vehicles can be outfitted GPS with cellular transmitter to allow police to do tracking and recovery.
* **Trailer Tracking:** Haulage and Logistics companies often operate Lorries with detachable load carrying units. The part of the vehicle that drives the load is known as the cab and the load carrying unit is known as the trailer. There are different types of trailer used for different applications, e.g. flat bed, refrigerated, curtain sider, box container.

1. **Methodology:**

**4.1 Description:**

To accomplish the specified objectives of this project and develop web applications, we are using Google API with JavaScript. It creates a methodological framework for visualizing in Google map.

**About Google API:**

Google Maps is a freely available web mapping application provided by Google that allows the search and visualization of geographic information. The integration of many Web 2.0 technologies within the google map user interface leads to a smart and intuitive interaction:

* Drag & Drop mechanism for moving/selecting items,
* Mouse scroll wheel to zoom in/out,
* Drag of a map for panning,
* Double click to zoom and center the viewpoint.

From a development point of view, Google licensing policy allows web application developers to integrate google map in their sites, through the google map APIs. These APIs provide a set of utilities for manipulating maps and adding content to them through a variety of services.

**About JavaScript:**

JavaScript is a dynamic language primarily used in client-side web applications. Once regarded as a toy scripting language its popularity has increased in recent years due to its use in complex web applications such as Bing maps and Google Maps.

The Google Maps APIs is a JavaScript API which gives developers several ways of embedding Google Maps into web pages, and allows for either simple use or extensive customization. It is designed mainly for data representation purposes. The Google Maps API works in a browser interface. To use this API, an API key should be obtained from Google. According to the terms, the Websites that use Google Maps API should be accessible to the consumers without charge; it cannot be used in enterprise applications.

JavaScript provides support for object oriented techniques but uses a prototype-based object system as opposed to the more traditional class-based system found in languages such as Java or C++. Objects in JavaScript are collections of mutable key/value properties roughly analogous to maps in Java or dictionaries in Python. The potential objects an application might use are not defined before instantiation, instead objects are instantiated and then mutated into their necessary form as the application executes.

**4.2 Data sources:**

|  |  |
| --- | --- |
| Web Server | IIS7 |
| Web Application | ASP.Net |
| Web Mapping | Google API |
| Dev IDE | Visual Studio 2010 |
| Programming Language | JavaScript |
| DotNet Framework | 4 |
| Web Mapping Services Types | Geo fencing, Direction Services, Route boxer |
| Web Mapping Common Functionalities | Navigation Tools (Pan, Zoom in, Zoom out, Zoom Previous, Zoom Next, and Zoom Extent), resizable polygon and circle with radius |
| Web Mapping Custom Functionalities | Monitoring and Vehicle tracking, providing Alerts. |

Table - 2 Data sources used in this project

**4.3 Methodology Diagram:**

Create Google Maps JavaScript API Key

Initialize Google Maps

Geofence creation on Map ( Circle/Polygon)

Store the Lat/Long of the geofence boundaries in a variable

Display the alerts using the stored Lat/Long values

Call the relevant route path using Direction Service concept

Store the Lat/Long of the selected route path in a variable

Create buffer using route boxer concept

Store the Lat/Long bounds of the buffer area in a variable

Geofence Process

Route Boxer process

Overview path with Direction Service

Figure4.1. Methodology diagram

**Task 1 - Geo Fencing:**

One of the important features of GPS Tracking software using GPS Tracking devices is geo-fencing and its ability to help keep track of assets. Geo-fencing allows users of a GPS Tracking Solution to draw zones (i.e., a Geo Fence) around places of work, customer’s sites and secure areas. These geo fences when crossed by a GPS equipped vehicle or person can trigger a warning to the user or operator via SMS or Email.

**Steps to Implement Geo fencing:**

1. Create and Include Google Maps JavaScript API key.
2. Initialize Google maps

Write the Method of the Initialization in Head and call it in the body by using method called “Onload”.

**Method to Initialize Map:**

It consists of below three elements,

1. Map
2. MapOptions: Consists of properties, used to pass parameters for the Map such as, Zoom, center, MapTypeID,…
3. MapTypeID: Identifiers for common MapTypes.

Consists of 4 Types of Constants as below,

* HYBRID
* SATELLITE
* ROADMAP
* TERRIAN

Pass the above three parameters at the initialize map

1. Create geo-fence as per client specification (Polygon/Circle) using any one of the below way,

Polygon creation can be done in three ways,

1. Create a polyline on single click and close the polyline into polygon when user double clicks on the map.
2. Draw a polygon on map which can be resizable
3. Draw a polygon on map using drawing tool
4. Draw a circle on map by providing radius at runtime.
5. Save the coordinates of the boundary in an array

**LatLng bounds** is the method which is used to store the LatLng values

1. Pass alerts by using the Polygon boundary coordinates

**Contains LatLng** is the method which is used to determine whether the point is within the point or not

**Code for Geofencing:**

* Create a resizable polygon and store the Polygon Lat/Long corners in an array.

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">

<head>

<link rel="stylesheet" type="text/css" href="include.css" />

<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />

<title>Resizable Polygons</title>

<style type="text/css">

v\:\* { behavior:url(#default#VML); }

body { font-family: Verdana; }

h3 { margin-left: 8px; }

#map { height: 400px;

width: 550px;

border: 1px solid gray;

margin-top: 8px;

margin-left: 8px;

}

.button { display: block;

width: 180px;

border: 1px Solid #565;

background-color:#F5F5F5;

padding: 3px;

text-decoration: none;

text-align:center;

font-size:smaller;

}

.button:hover {

background-color: white;

}

#descr { position:absolute;

top:44px;

left: 580px;

width: 250px;

}

</style>

<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAFnFO3ySluSuha6r1vYekiRQNy\_ewe8RNLHRa7cwLE-yWPUZNWBSV43OE1hhpvzIXRf04qzvWdmKBEw&amp;sensor=false" type="text/javascript"></script>

</head>

<body onload="loadMap()" onunload="GUnload()">

<h3>Draw Resizable Polygons</h3>

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

<table id="descr" border="0" cellspacing="10" cellpadding="1">

<tr><td>

Click at least at three different places on the map to draw a polygon.

The corners of the polygon are draggable and removable. You can add further vertices simply by click.

</td></tr>

<tr><td>

Area of polygon:

</td></tr>

<tr><td id="status">&nbsp; </td></tr>

<tr><td height="20">

&nbsp;

</td></tr><tr><td>

<a href="#" class="button" onclick="zoomToPoly();return false;">Zoom To Polygon</a>

</td></tr><tr><td>

<a href="#" class="button" onclick="clearPoly();return false;">Remove Polygon</a>

</td></tr><tr><td style="padding-right:19px; height:30px;">

</td></tr>

</table>

<div id="coords"></div>

<script type="text/javascript">

//<![CDATA[

// Global variables

var map;

var polyShape;

var polyLineColor = "#3355ff";

var polyFillColor = "#335599";

var polyPoints = new Array();

var markers = new Array();

var report = document.getElementById("status");

function loadMap() {

map = new GMap2(document.getElementById("map"), { draggableCursor: 'auto', draggingCursor: 'move' });

map.setCenter(new GLatLng(51.2516, 6.976318), 8);

map.addMapType(G\_PHYSICAL\_MAP);

var hierarchy = new GHierarchicalMapTypeControl();

hierarchy.addRelationship(G\_SATELLITE\_MAP, G\_HYBRID\_MAP, "Labels", true);

map.addControl(hierarchy);

map.addControl(new GSmallMapControl());

map.disableDoubleClickZoom();

GEvent.addListener(map, "click", leftClick);

}

function addIcon(icon) { // Add icon attributes

icon.iconSize = new GSize(15, 10);

icon.dragCrossSize = new GSize(0, 0);

icon.shadowSize = new GSize(11, 11);

icon.iconAnchor = new GPoint(5, 5);

// icon.infoWindowAnchor = new GPoint(5, 1);

}

function leftClick(overlay, point) {

if (point) {

// Square marker icons

var square = new GIcon();

square.image = "http://maps.google.com/mapfiles/kml/shapes/square.png";

addIcon(square);

// Make markers draggable

var marker = new GMarker(point, { icon: square, draggable: true, bouncy: false, dragCrossMove: true });

markers.push(marker);

map.addOverlay(marker);

GEvent.addListener(marker, "drag", function () {

drawPoly();

});

GEvent.addListener(marker, "mouseover", function () {

marker.setImage("mapIcons/m-over-square.png");

});

GEvent.addListener(marker, "mouseout", function () {

marker.setImage("mapIcons/square.png");

});

// Second click listener to remove the square

GEvent.addListener(marker, "click", function () {

// Find out which square to remove

for (var n = 0; n < markers.length; n++) {

if (markers[n] == marker) {

map.removeOverlay(markers[n]);

break;

}

}

markers.splice(n, 1);

drawPoly();

});

drawPoly();

}

}

function drawPoly() {

if (polyShape) map.removeOverlay(polyShape);

polyPoints.length = 0;

document.getElementById("coords").innerHTML = "";

for (i = 0; i < markers.length; i++) {

polyPoints.push(markers[i].getLatLng());

document.getElementById("coords").innerHTML += i + ": " + markers[i].getLatLng().toUrlValue(6) + "<br>";

}

// Close the shape with the last line or not

polyPoints.push(markers[0].getLatLng());

polyShape = new GPolygon(polyPoints, polyLineColor, 3, .8, polyFillColor, .3);

var unit = " km&sup2;";

var area = polyShape.getArea() / (1000 \* 1000);

if (markers.length <= 2) {

report.innerHTML = "&nbsp;";

}

else if (markers.length > 2) {

report.innerHTML = area.toFixed(3) + unit;

}

map.addOverlay(polyShape);

}

function zoomToPoly() {

if (polyShape && polyPoints.length > 0) {

var bounds = polyShape.getBounds();

map.setCenter(bounds.getCenter());

map.setZoom(map.getBoundsZoomLevel(bounds));

}

}

function clearPoly() {

// Remove polygon and reset arrays

map.clearOverlays();

polyPoints.length = 0;

markers.length = 0;

report.innerHTML = "&nbsp;";

}

</script>

</body>

</html>

**Task 2:** Retrieve all the Lat/long values of the specified direction path

**Algorithm for Direction Service:**

You can calculate directions (using a variety of methods of transportation) by using the DirectionsService object. This object communicates with the Google Maps API Directions Service which receives direction requests and returns computed results. You may either handle these directions results yourself or use the DirectionsRenderer object to render these results.

* Direction Service object used to calculate directions
* Communicates with Google Maps API Directions Service which recieves direction requests and returns computed results
* DirectionRenderer object is used to render the results

Process to use Directions in V3 will be done in three steps;

1) Direction Service

2) Direction Status

3) Direction Display

**1) Direction Service:**

Create an object of type DirectionService and call DirectionService. Route() to initiate a request to the Directions service, passing it a DirectionRequest object literal containing the input terms and a callback method to execute upon receipt of the response.

DirectionRequest object contians fields such as origin, destination, waypoints [], travel Mode,…. Which helps to process the request

**2) Direction Status:**

Initiating a directions request to the DirectionService with the route() method requires passing a callback which executes upon completion of the service request. This callback will return a DirectionResult and a DirectionStatus code in the response.

**3) Direction Display:** It displays the Direction Result

Steps to display a DirectionResult using a DirectionRenderer:

i) Create a DirectionRenderer object.

ii) Call setMap() on the renderer to bind it to the passed map.

iii) Call setDirections() on the renderer, passing it the DirectionsResult as noted above. Because the renderer is an MVCObject, it will automatically detect any changes to its properties and update the map when its associated directions have changed.

**Steps to implement Lat/Long retrieval of specified path:**

1. Create and Include Google Maps JavaScript API key.
2. Initialize Google maps

Write the Method of the Initialization in Head and call it in the body by using method called “Onload”.

**Method to Initialize Map:**

It consists of below three elements,

1. Map
2. MapOptions: Consists of properties, used to pass parameters for the Map such as, Zoom, center, MapTypeID.
3. MapTypeID: Identifiers for common MapTypes.

Consists of 4 Types of Constants as below,

* HYBRID
* SATELLITE
* ROADMAP
* TERRIAN

Pass the above three parameters at the initialize map

1. Implement direction service to draw a polyline in a specified path
2. Retrieve the Lat/Long values of the requested specified path

**Overview path** is the method which is used to retrieve the Lat/Long values of the requested specified path

Code:

* Code for Retrieve Lat/Long Values of the specified path

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs" Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8"/>

<title>Google Maps JavaScript API Search Along a Route Example</title>

<script src="http://maps.google.com/maps/api/js?sensor=false" type="text/javascript"></script>

<script type="text/javascript">

var map = null;

var boxpolys = null;

var directions = null;

var path = null;

//var routeBoxer = null;

//var distance = null;

function initialize()

{ // Default the map view to the continental U.S.

var mapOptions = {

center: new google.maps.LatLng(37.09024, -95.712891),

mapTypeId: google.maps.MapTypeId.ROADMAP,

zoom: 4

};

map = new google.maps.Map(document.getElementById("map"), mapOptions);

//routeBoxer = new RouteBoxer();

directionService = new google.maps.DirectionsService();

directionsRenderer = new google.maps.DirectionsRenderer({ map: map });

}

function route() { // Clear any previous route boxes from the map

//clearBoxes();

// Convert the distance to box around the route from miles to km

//distance = parseFloat(document.getElementById("distance").value) \* 1.609344;

var request = { origin: document.getElementById("from").value,

destination: document.getElementById("to").value,

travelMode: google.maps.DirectionsTravelMode.DRIVING

}

// Make the directions request

directionService.route(request, function (result, status) {

if (status == google.maps.DirectionsStatus.OK) {

directionsRenderer.setDirections(result);

// Box around the overview path of the first route

google.maps.event.addListener(map, "bounds\_changed", function () {

path = result.routes[0].overview\_path;

document.getElementById("Listing").innerHTML = "lat/lng of overview path is <BR>";

document.getElementById("Listing").innerHTML += path + "<BR>";

//var boxes = routeBoxer.box(path, distance);

//drawBoxes(boxes);

});

}

else {

alert("Directions query failed: " + status);

}

});

}

</script>

</head> <body onload="initialize();">

<div id="map" style="width: 1150px; height: 400px;"></div>

<div id="controls">

from <input type="text" id="from" value="Hyderabad"/>

to <input type="text" id="to" value="Warangal"/>

<input type="submit" onclick="route()"/>

</div>

<div style="position: absolute; width: 1150px; top:450px; height: 400px;" id="Listing">

</div>

</body>

</html>

**Task 3:** Draw boxes on the specified direction path and retrieve the bounding coordinates. Pass the alerts based on the bounding coordinates.

1. Create and Include Google Maps JavaScript API key.
2. Initialize Google maps

Write the Method of the Initialization in Head and call it in the body by using method called “Onload”.

**Method to Initialize Map:**

It consists of below three elements,

1. Map
2. MapOptions: Consists of properties, used to pass parameters for the Map such as, Zoom, center, MapTypeID,
3. MapTypeID: Identifiers for common MapTypes.

Consists of 4 Types of Constants as below,

* HYBRID
* SATELLITE
* ROADMAP
* TERRIAN

Pass the above three parameters at the initialize map

1. Implement direction service to draw a polyline in a specified path
2. Create buffer using route boxer concept
3. Retrieve the Lat/Long values of the requested specified path

**Overview path** is the method which is used to retrieve the Lat/Long values of the requested specified path

**Code for Route Boxer:**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8"/>

<title>Google Maps JavaScript API Search Along a Route Example</title>

<script src="http://maps.google.com/maps/api/js?sensor=false" type="text/javascript"></script>

<script src="http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/src/RouteBoxer.js" type="text/javascript"></script>

<script type="text/javascript">

var map = null;

var boxpolys = null;

var directions = null;

var routeBoxer = null;

var distance = null;

function initialize()

{ // Default the map view to the continental U.S.

var mapOptions = {

center: new google.maps.LatLng(37.09024, -95.712891),

mapTypeId: google.maps.MapTypeId.ROADMAP,

zoom: 4

};

map = new google.maps.Map(document.getElementById("map"), mapOptions);

routeBoxer = new RouteBoxer();

directionService = new google.maps.DirectionsService();

directionsRenderer = new google.maps.DirectionsRenderer({ map: map });

}

function route()

{ // Clear any previous route boxes from the map

clearBoxes();

// Convert the distance to box around the route from miles to km

distance = parseFloat(document.getElementById("distance").value) \* 1.609344;

var request = { origin: document.getElementById("from").value,

destination: document.getElementById("to").value,

travelMode: google.maps.DirectionsTravelMode.DRIVING

}

// Make the directions request

directionService.route(request, function(result, status)

{ if (status == google.maps.DirectionsStatus.OK)

{ directionsRenderer.setDirections(result);

// Box around the overview path of the first route

var path = result.routes[0].overview\_path;

var boxes = routeBoxer.box(path, distance);

drawBoxes(boxes);

}

else

{

alert("Directions query failed: " + status);

}

});

}

// Draw the array of boxes as polylines on the map

function drawBoxes(boxes) {

boxpolys = new Array(boxes.length);

for (var i = 0; i < boxes.length; i++) {

boxpolys[i] = new google.maps.Rectangle({

bounds: boxes[i],

fillOpacity: 0,

strokeOpacity: 1.0,

strokeColor: '#000000',

strokeWeight: 1,

map: map

});

}

}

// Clear boxes currently on the map

function clearBoxes() {

if (boxpolys != null) {

for (var i = 0; i < boxpolys.length; i++)

{ boxpolys[i].setMap(null); }

}

boxpolys = null;

}

</script>

<style> #map { border: 1px solid black; }

#controls { font-family: sans-serif; font-size: 11pt; margin-top: 10px; margin-left: 20px; </style>

</head> <body onload="initialize();">

<div id="map" style="width: 512px; height: 400px;"></div>

<div id="controls">

Box within at least <input type="text" id="distance" value="4" size="2">miles of the route<br/>

from <input type="text" id="from" value="Rajamundary"/>

to <input type="text" id="to" value="Visakhapatnam"/>

<input type="submit" onclick="route()"/>

</div>

</body>

</html>

1. **Results and Discussion**

**Task 1:**

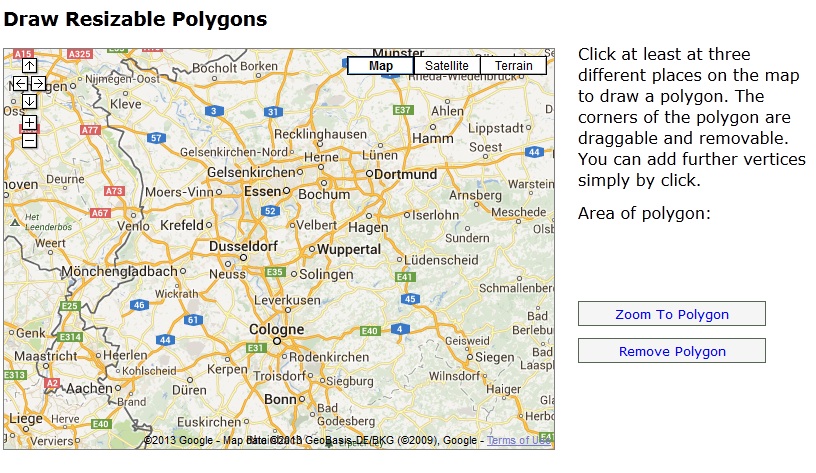


Figure 5.1(a)

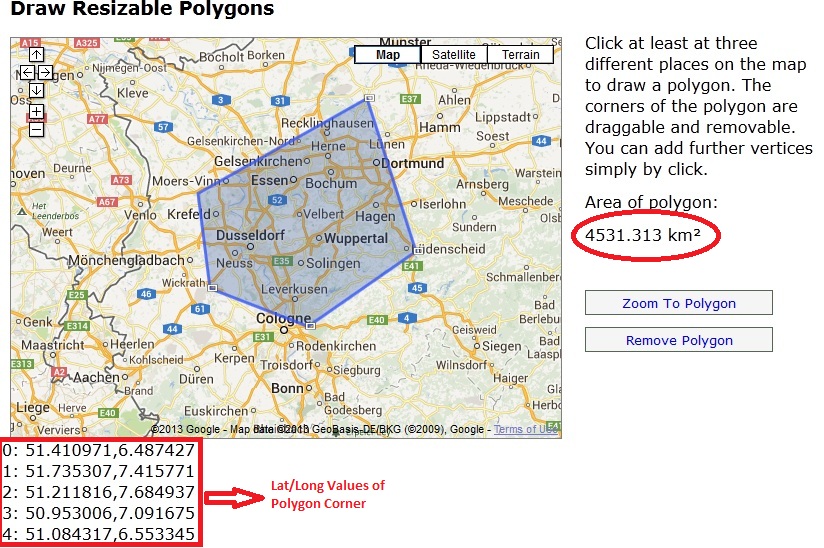


Figure 5.1(b)

In the above snapshot, we can view the Polygon which was drawn at the runtime and the Lat/Long values are displayed below the Google map screen and the Polygon area is also visible in the above figure.” Zoom to Polygon” and “Remove Polygon” are used to adjust the polygon

**Task 2:**

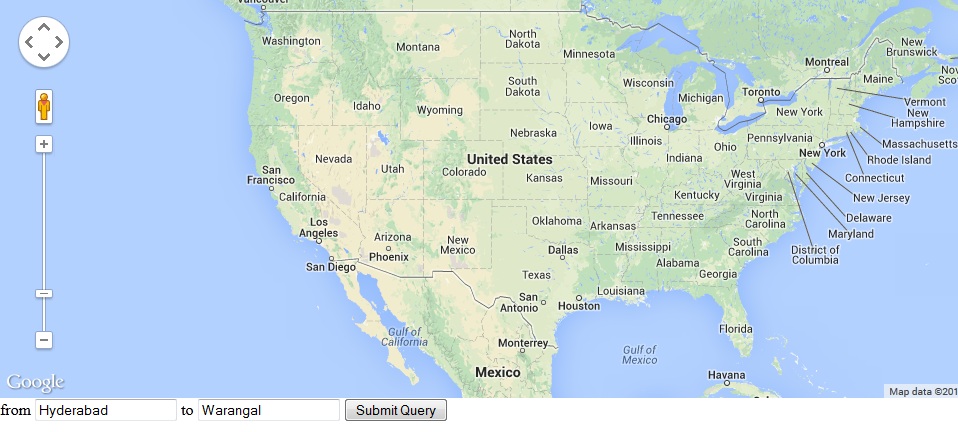


Figure 5.2(a)

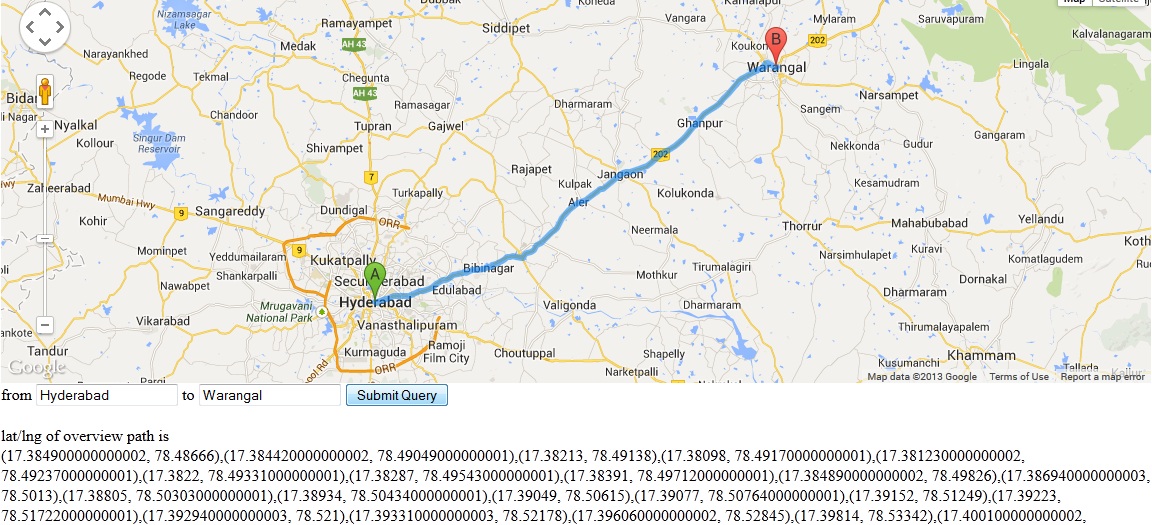


Figure 5.2(b)

In the above figure 5.2 (b), we can view a polyline has been drawn in the specified route path i.e. from Location A to B and the Lat/long values of the specified route path were also displayed below the result map. These Lat/Long values are used to create the alerts.

**Task 3:**

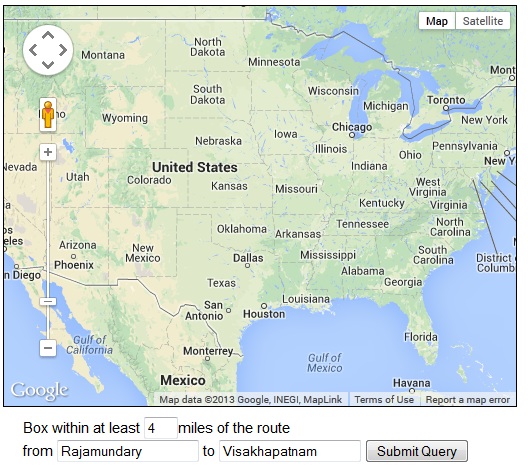


Figure 5.3(a)

In the above figure 5.3 (a), we can see that the Google Map is initialized and we have given the inputs and provided the buffer radius as 4miles to the either sides of the specified route.

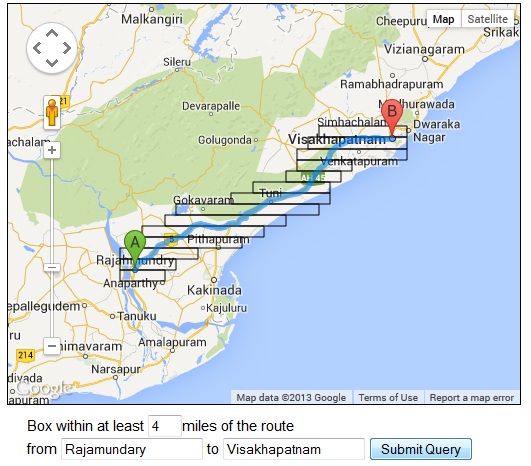


Figure 5.3(b)

In the above figure 5.3(b), we can see that a polyline has drawn from location A to B and boxes are created on both sides of the route path A and B. Retrieve the Lat/Long of the polygon bounds and store the values in an array. These Lat/Long Values helps to display alerts.

1. **Learning outcomes**

* Use variables, arrays, functions and objects in JavaScript.
* Identify popular JavaScript Libraries.
* The basic JavaScript syntax and structures.
* Map HTML using the DOM - Document Object Model.
* How JavaScript programs are used in a web page including the use of event-handlers and the Document Object Model.
* Insert JavaScript into a web page using various approaches including inline code, internal scripts and external JavaScript files.
* How to use simple JavaScript functions and event handlers and how it reflects in common web applications.
* Build maps-enabled web applications.

|  |  |
| --- | --- |
| |  | | --- | | 1. **Other Information:** | |

1. **Daily Routine of an Intern:**

I have been working in Tech Mahindra, located at Hitech city, Hyderabad. I did my in internship in GIS development department in bahdurpally branch. I worked in General shift and working times were 9.00 to 6.30. I worked for 5days week with Saturday and Sunday as my Weekly offs.

My daily activities includes collecting the specifications taken from the respective heads, Prepare the algorithms for the given tasks, designing web page, Developing web application and Executing it, identifying bugs if we get and fixing them.

1. **How Did the Internship Increase Knowledge of the Profession?**

Internship is an excellent opportunity to build professional connections. Therefore these connections will be able to give us a strong recommendation in the future. During this internship I got an opportunity to test out the skills what I have learned in NIIT University and see how it works in the real time projects. I learned various software’s of Geographic Information Systems for developing client side web applications (Using Silverlight), publishing web applications and consuming it using GIS web server and also software customization of ArcGIS Desktop.

I put more efforts on GIS development; I got to know designing and developing web applications as per the project requirement. I learned how to prepare algorithms, how to execute each algorithm, and debugging. Finally, identify challenges which I have faced to develop web applications using Google API.

1. **Relate Course Work at NU to the Internship Experience**

The Course work at NU helped me to explore to GIS and GIS programming. I have gained theoretical knowledge about GIS programming. I have learned basic concepts of Arc objects. That knowledge helped me in my Internship.

1. **How did the Internship Stimulate Personal Growth?**

During Internship I got chance to work with Development team, which helped me to gain programming skills. I came to know technical knowledge of GIS development projects. My communication skills, both written and oral, and technical skills are improved.

1. **Suggestions for Future Interns – Please suggest ways in which we can make the internship a more valuable experience.**

Below are the few suggestions which I would like to share to our feature interns,

* Identify your Area of interest (AOI) and do internship on that stream
* Have a proper knowledge on the basics of the selected stream
* Do research on advanced technology
* Analyze the current Market and be up to date with the updates