

# **Car trip Simulation System**

## Software Design Document

Aysha Khan  
Rafat Munshi

## TABLE OF CONTENTS

<b>1.</b>	<b>INTRODUCTION</b>	<b>3</b>
1.1	Purpose	3
1.2	Scope	3
1.3	Overview	3
1.4	Reference Material	3
1.5	Definitions and Acronyms	3
<b>2.</b>	<b>SYSTEM OVERVIEW</b>	<b>3</b>
<b>3.</b>	<b>SYSTEM ARCHITECTURE</b>	<b>3</b>
3.1	Architectural Design	4
3.2	Decomposition Description	5
3.3	Process Analyses	12
4	<b>Dependency Description</b>	
4.1	Interprocess Dependency	
4.2	Data dependency	13
<b>5.</b>	<b>HUMAN INTERFACE DESIGN</b>	<b>13</b>
5.1	Overview of User Interface	13
5.2	Database ER Diagrams	19
5.3	Details of DB tables	19
<b>6.</b>	<b>UI and Functional test cases</b>	<b>20</b>

# 1. INTRODUCTION

## 1.1 Purpose

This document outlines the Software Design Specifications as part of the design plan and specifications for developing a web based application for Car trip simulation System for google maps users.

This document expands the functionality described by the features in the Software Requirements Specifications (SRS).

## 1.2 Scope

This document takes the features as outlined in the SRS and expands each of the features to include the design issues of user interface, data flow, process analysis and then module design. The design document will take a Feature based approach to the design document. The interfaces are illustrated in the dataflow and the sequence diagrams. The Graphical User Interfaces are discussed and illustrated with mock ups of the panels to be implemented. The application data is addressed in the discussion of the dataflow diagrams.

## 1.4 Reference Material

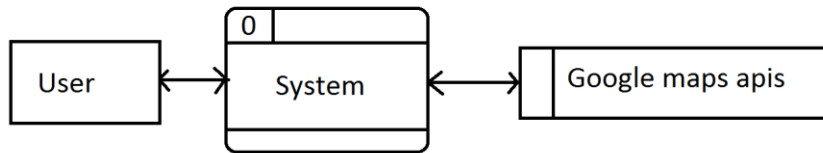
*IEEE Recommended Practice for Software Design Descriptions*, IEEE Std 1016-1998 (Revision of IEEE Std 1016-1987) Available at <http://web.nps.navy.mil/~nschneid/is3020/PDF/1016-1998.pdf>

# 2. SYSTEM OVERVIEW

Car trip simulation System is a web based application built with javascript and google maps API for users to find directions and simulate his/her car's movement on the screen. Zoom in and zoom out options are available to have a view of the proximity areas of the car with various zooming levels. For a particular speed of the car, the zoom in will animate the car movement slowly and the zoom out will cause the car to move faster given a steady speed. An option to increase or decrease the speed of the car is also given for better functionality of the application.

### 3. SYSTEM ARCHITECTURE

#### 3.1 Architectural Design

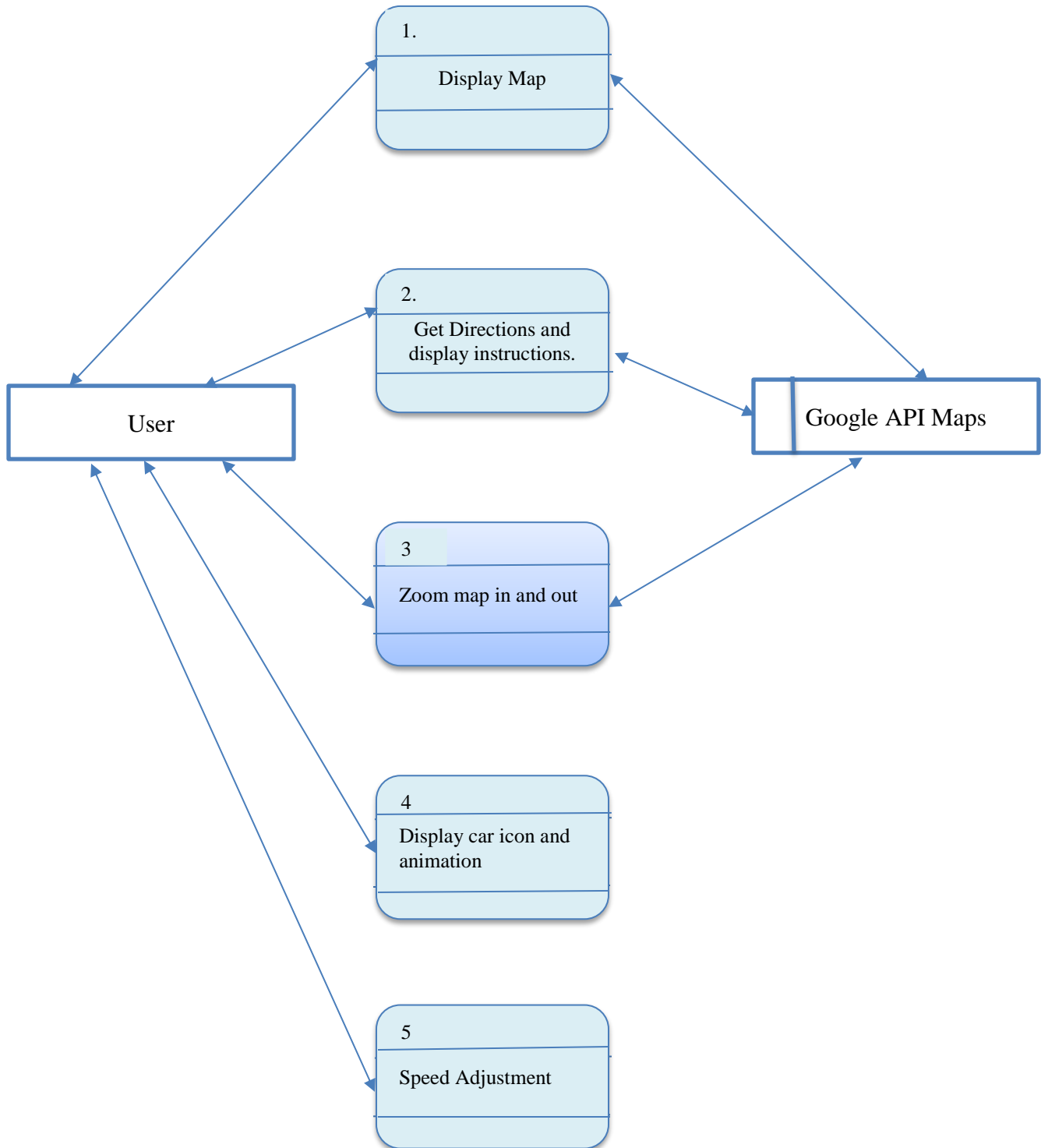


#### 3.2 Decomposition Description

This section of this report decomposes use-case features as provided in the SRS document into its data flow processes by examining its data flow diagram and its process flow through the use of sequence diagrams.

### 3.5.2 Data Flow Description

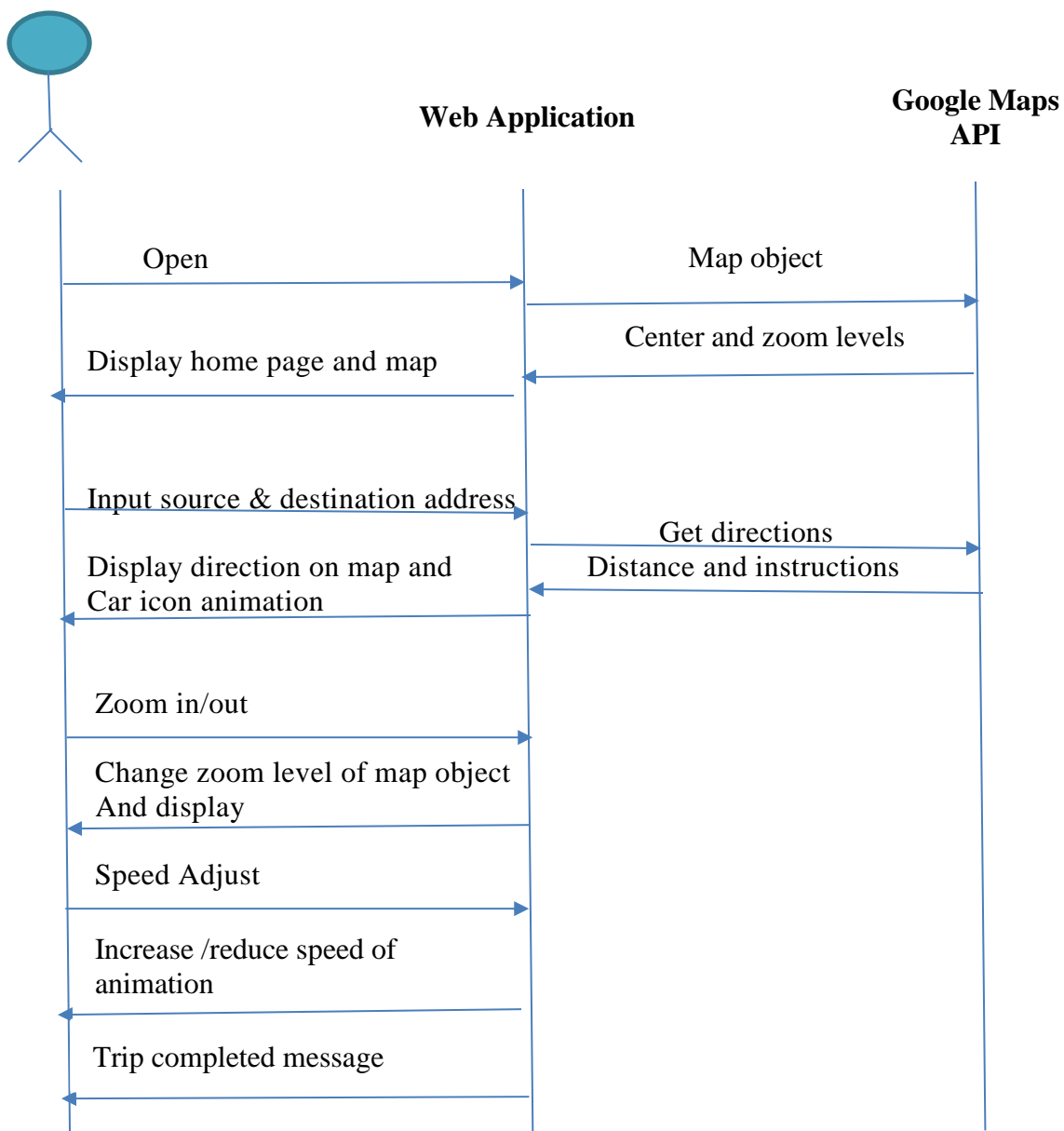
DFD level 1



### 3.3. Process Analyses

The following sequence diagram shows the process involved in finding directions along with the real car movement.

Sequence diagram for directions, instructions and animation display.



## 4. Dependency Description

### 4.1 Interprocess Dependencies

The interprocess dependencies are illustrated in the data flow diagrams of the features. These will be reviewed and incorporated within the same directory of files.

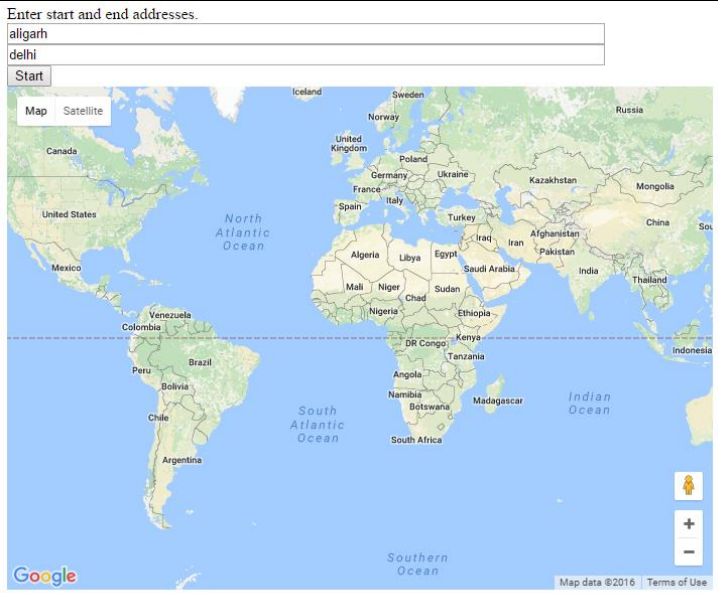
### 4.2 Data Dependencies

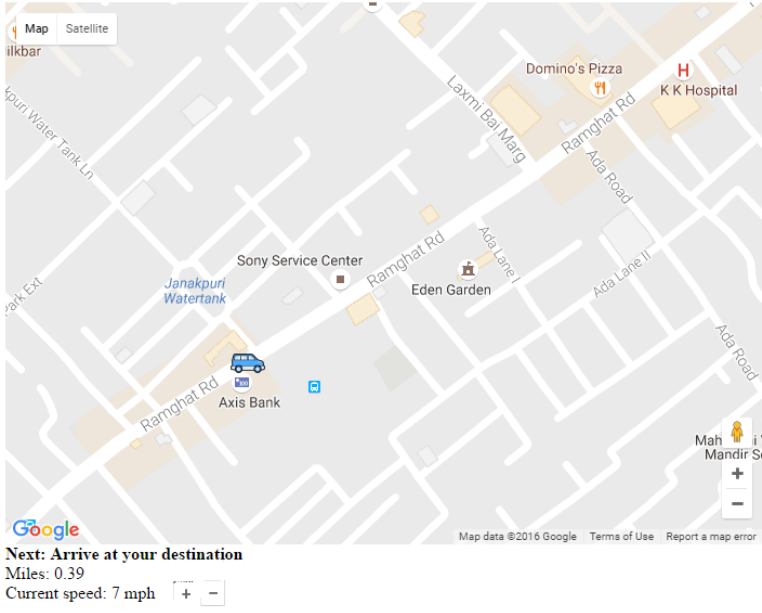
The data dependencies are very limited in this design.

## 5. HUMAN INTERFACE DESIGN

### 5.1 Overview of all the files required

Based on the data flow diagram and the sequence diagram above, a detailed description of the required files and their description along with their UI mockup is shown below.

File name	Description	Mock up
Cartripsimu.htm	The home page display the map along with input boxes for user to input source and destination as shown. The system understands the addresses input by the user on the click of the start button and based on the latitude and longitude, returns the direction and instruction details to the page.	 <p>Enter start and end addresses. aligarh delhi Start</p> <p>Map Satellite</p> <p>North Atlantic Ocean</p> <p>South Atlantic Ocean</p> <p>Indian Ocean</p> <p>Southern Ocean</p> <p>Google</p> <p>Map data ©2016 Terms of Use</p> <p>Miles: 0.00</p>

	<p>The car icon starts moving on the path of the direction received between the source and destination. The animation proceeds based on the selected speed and driving instructions received from the api call is displayed. On click of the zoom icons, the map zooms in and out to 8 levels as defined by the api. The animation fastens or slows down accordingly. The speed adjustment button increases or decreases the speed 5 kmph at a time. The animation and instruction delay is adjusted accordingly.</p>	
--	---	--

## 5.2 Google api data exchange format

1. The google apis can be used to call a service and obtain data in json format which can then be used and traversed to complete the functionality of the application
2. The javascript api can be used to incorporate and display the results without need of traversing json data. Hence this approach is to be used in developing this application. The pseudo code snippet is provided for reference-

### Loading the Google Maps JavaScript API

To load the Google Maps JavaScript API, use a `script` tag like the one in the following example:

```
<script async defer
  src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callba
```



```
ck=initMap">
</script>
```

## Map Options

---

There are two required options for every map: `center` and `zoom`.

```
map = new google.maps.Map(document.getElementById('map'), {
  center: {lat: -34.397, lng: 150.644},
  zoom: 8
});
```

## The Map Object

---

```
map = new google.maps.Map(document.getElementById("map"), {...});
```

Sample JS Function for getting directions-

```
function initMap() {
  var chicago = {lat: 41.85, lng: -87.65};
  var indianapolis = {lat: 39.79, lng: -86.14};

  var map = new google.maps.Map(document.getElementById('map'), {
    center: chicago,
    scrollwheel: false,
    zoom: 7
  });

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

  // Set destination, origin and travel mode.
  var request = {
    destination: indianapolis,
    origin: chicago,
    travelMode: 'DRIVING'
  };
}
```

```

// Pass the directions request to the directions service.
var directionsService = new google.maps.DirectionsService();
directionsService.route(request, function(response, status) {
  if (status == 'OK') {
    // Display the route on the map.
    directionsDisplay.setDirections(response);
  }
});
}

```

#### Sample JS function to show marker-

```

function initMap() {
  var myLatLng = {lat: -25.363, lng: 131.044};

  // Create a map object and specify the DOM element for display.
  var map = new google.maps.Map(document.getElementById('map'), {
    center: myLatLng,
    scrollwheel: false,
    zoom: 4
  });

  // Create a marker and set its position.
  var marker = new google.maps.Marker({
    map: map,
    position: myLatLng,
    title: 'Hello World!'
  });
}

```

#### Sample JS function to get directions between two addresses-

```

function initMap() {
  var directionsService = new google.maps.DirectionsService;
  var directionsDisplay = new google.maps.DirectionsRenderer;
  var map = new google.maps.Map(document.getElementById('map'), {
    zoom: 7,
    center: {lat: 41.85, lng: -87.65}
  });
  directionsDisplay.setMap(map);
}

```

```

var onChangeHandler = function() {
    calculateAndDisplayRoute(directionsService, directionsDisplay);
};
document.getElementById('start').addEventListener('change', onChangeHandler);
document.getElementById('end').addEventListener('change', onChangeHandler);
}

function calculateAndDisplayRoute(directionsService, directionsDisplay) {
    directionsService.route({
        origin: document.getElementById('start').value,
        destination: document.getElementById('end').value,
        travelMode: 'DRIVING'
    }, function(response, status) {
        if (status === 'OK') {
            directionsDisplay.setDirections(response);
        } else {
            window.alert('Directions request failed due to ' + status);
        }
    });
}

```

Sample JS function to get and display text instructions for driving-

```

function initMap() {
    var directionsDisplay = new google.maps.DirectionsRenderer;
    var directionsService = new google.maps.DirectionsService;
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 7,
        center: {lat: 41.85, lng: -87.65}
    });
    directionsDisplay.setMap(map);
    directionsDisplay.setPanel(document.getElementById('right-panel'));

    var control = document.getElementById('floating-panel');
    control.style.display = 'block';
    map.controls[google.maps.ControlPosition.TOP_CENTER].push(control);

    var onChangeHandler = function() {
        calculateAndDisplayRoute(directionsService, directionsDisplay);
    }
}

```

```

};
document.getElementById('start').addEventListener('change', onChangeHandler);
document.getElementById('end').addEventListener('change', onChangeHandler);
}

function calculateAndDisplayRoute(directionsService, directionsDisplay) {
  var start = document.getElementById('start').value;
  var end = document.getElementById('end').value;
  directionsService.route({
    origin: start,
    destination: end,
    travelMode: 'DRIVING'
  }, function(response, status) {
    if (status === 'OK') {
      directionsDisplay.setDirections(response);
    } else {
      window.alert('Directions request failed due to ' + status);
    }
  });
}

```

Sample JS function to get distance  
between two addresses-

```

function initMap() {
  var bounds = new google.maps.LatLngBounds;
  var markersArray = [];

  var origin1 = {lat: 55.93, lng: -3.118};
  var origin2 = 'Greenwich, England';
  var destinationA = 'Stockholm, Sweden';
  var destinationB = {lat: 50.087, lng: 14.421};

  var destinationIcon = 'https://chart.googleapis.com/chart?' +
    'chst=d_map_pin_letter&chld=D|FF0000|000000';
  var originIcon = 'https://chart.googleapis.com/chart?' +
    'chst=d_map_pin_letter&chld=O|FFFF00|000000';
  var map = new google.maps.Map(document.getElementById('map'), {
    center: {lat: 55.53, lng: 9.4},
    zoom: 10
  });
  var geocoder = new google.maps.Geocoder;

```

```

var service = new google.maps.DistanceMatrixService;
service.getDistanceMatrix({
  origins: [origin1, origin2],
  destinations: [destinationA, destinationB],
  travelMode: 'DRIVING',
  unitSystem: google.maps.UnitSystem.METRIC,
  avoidHighways: false,
  avoidTolls: false
}, function(response, status) {
  if (status !== 'OK') {
    alert('Error was: ' + status);
  } else {
    var originList = response.originAddresses;
    var destinationList = response.destinationAddresses;
    var outputDiv = document.getElementById('output');
    outputDiv.innerHTML = "";
    deleteMarkers(markersArray);

    var showGeocodedAddressOnMap = function(asDestination) {
      var icon = asDestination ? destinationIcon : originIcon;
      return function(results, status) {
        if (status === 'OK') {
          map.fitBounds(bounds.extend(results[0].geometry.location));
          markersArray.push(new google.maps.Marker({
            map: map,
            position: results[0].geometry.location,
            icon: icon
          }));
        } else {
          alert('Geocode was not successful due to: ' + status);
        }
      };
    };

    for (var i = 0; i < originList.length; i++) {
      var results = response.rows[i].elements;
      geocoder.geocode({'address': originList[i]},
        showGeocodedAddressOnMap(false));
      for (var j = 0; j < results.length; j++) {
        geocoder.geocode({'address': destinationList[j]},
          showGeocodedAddressOnMap(true));
      }
    }
  }
});

```

```
        outputDiv.innerHTML += originList[i] + ' to ' + destinationList[j] +  
            ':' + results[j].distance.text + ' in ' +  
            results[j].duration.text + '<br>';  
    }  
}  
});  
}
```

```
function deleteMarkers(markersArray) {  
    for (var i = 0; i < markersArray.length; i++) {  
        markersArray[i].setMap(null);  
    }  
    markersArray = [];  
}
```

**6. UI and functional test cases-**

Sr no	Test case	Description	Expected outcome	Pass/Fail	comments
1	Application startup	On clicking the html file you will be directed to an html page where you will get to see a google map having two tabs of source and destination.			
2	Main Window	When the window is opened up, you can write down a source and a destination.			
3	Directions	As soon as you enter the locations, a car icon will be displayed showing the whole distance between the particular source and the destination. When the destination is reached the car icon will disappear marking the distance between the two locations.			
4	Speed variation	As you move over particular locations you can change the different speed values like taking values on scale of some basis of being slow, medium, fast etc.			
5	Zoom in and out	To search for sublocations within in locations you can zoom in the particular areas.eg. To reach aligarh's dodhpur from Delhi you can zoom in to have a perfect view of near by landmark areas. To find the actual locality of a particular sublocation you can zoom out a particular area. Eg. If you at dodhpur area of a Aligarh so to find out in which state and country it lies you can zoom out.			