CS458-Software Verification and Validation

2020-2021 Spring

PROJECT # 3

Group Members:

Zeynep Ayça Çam-21502269
Elif Beril Şaylı-21502795
Cenk Er-21600937
Ebru Kerem-21501859

Table of Contents

1.Test-Driven Development

Test-driven development is a test development principle where the development of test codes happen before the development of the actual code of the program. Test-driven development has very short development cycles. Before these cycles, we need to determine all test cases, and for each test case: write test code (red line),
write program code that will pass the test case (green line), do the test, refactor code that it becomes more efficient, effective and readable.This approach is bottom to top approach and with this principle, it is aimed to have the final program code that has fewer errors compared to another approach since we developed while we are testing.

2.Our Experiences with Test-Driven Development

According to the project documentation, we develop a responsive web page that is supported by both mobile and web devices. With coordinates which are latitude and longitude, the map demonstrates to the city via Google Maps API. Also, it gets GPS coordinates of user devices automatically and it shows user distance to the nearest city center. Moreover, according to coordinates or it gets GPS of the user device, then it shows user distance to the earth center. Since none of the group members have experience with test-driven development, at first we need to do some research about this development principle. It is quite different from what we have done so far, develops software according to meet requirements, and then test the code. However, later we get more familiar with this approach and in the end, we figure out that our overall development velocity is higher than other projects (1 and 2) because we did tests for whole development cycles and we confront problems at earlier stages. Our TDD experience started with planning what we will do so before starting coding we made a simple model. Then, we write simple unit tests since our functionalities were determined in the first place. To test our piece of code whether it has some vulnerabilities and other problematic features, we apply selenium unit tests. Then, we considered them as red line codes and we tried to solve problems. Therefore, we started to write new codes that are green line codes that will be integrated into our previous code and solve the problems. After we tried to integrate them into our previously written codes, we refactored them. We realize that TDD makes a safer and clean code. It also improves code quality since we did refactor in each development cycle. We are happy to use this approach.

3. Test Cases

Test Case 1	Test whether give true results when press find me button
Test Case 2	Test whether give alert or not if invalid characters entered.
Test Case 3	Test the behaviour when enter blank latitude or longitude
Test Case 4	Test the correctness of given nearest location
Test Case 5	Test whether given alert correct or not

Test Case 6	Test the nearest city based on reverse geocode
Test Case 7	Test whether distance of location to earth center is true or not

4. Implementation

We develop our test cases with Selenium in java. We develop responsive page with JavaScript, HTML, Css and Google Map API. The web page of application is also compatible with mobile devices. In each cycle, green line code is the new code segment added to red line code segment. The aim of the refactoring is optimizing code, increasing the speed, readability and reusability. We benefit from refactoring techniques which are simplify method calls, conditional expressions and composing methods. We change methods names, get rid of unnecessary variables and long methods and add comments to increase readability of code. We eliminate unnecessary more complicated logics from conditional expressions and overall code. We consider extendibility of functions to increase quality of code via refactoring.

4.1. Cycle 1

Test Case: Check when press find me button whether it finds correct output

Program Code (Red Line):

```
function findme(geocoder, map, infowindow){
      if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(function(position) {
            var pos = {
              lat: position.coords.latitude,
              lng: position.coords.longitude
            };
            document.getElementById("lat").value=position.coords.latitude;
            document.getElementById("lng").value=position.coords.longitude;
            }, function() {
            handleLocationError(true, infowindow, map.getCenter());
            });
      } else {
          // Browser doesn't support Geolocation
          handleLocationError(false, infowindow, map.getCenter());}}
```

Program Code (Green Line):

```
geocoder.geocode({'location': pos}, function(results) {
      var marker = new google.maps.Marker({
      position: pos,
      map: map });
infowindow.setContent(results[0].formatted_address);
var add= results[0].formatted_address ;
var value=add.split(",");
var count=value.length;
var country=value[count-1];
```

```
infowindow.open(map, marker);
map.setCenter(pos);
map.setZoom(11);
```

Refactored Program Code: We benefit from refactoring techniques of simplifying method calls and composing methods.

```
function findMyLocation(geocoder, map, infowindow){
      // If browser support Geolocation
        if (navigator.geolocation) {
          navigator.geolocation.getCurrentPosition(function(position) {
            var pos = {
              lat: position.coords.latitude,
              lng: position.coords.longitude
            }; // update elements
            document.getElementById("lat").value=position.coords.latitude;
            document.getElementById("lng").value=position.coords.longitude;
            geocoder.geocode({'location': pos}, function(results) {
                    // mark map and zoom on it
              var marker = new google.maps.Marker({
                position: pos,
                map: map,
                         optimized:false
              });
              infowindow.setContent(results[0].formatted_address);

              var add = results[0].formatted_address;
                    var value = add.split(",");
              document.getElementById("city").value = value[value.length-1] +
" , " + city ;
              infowindow.open(map, marker);
              map.setCenter(pos);
              map.setZoom(11);
            });

          }, function() {
            handleLocationError(true, infowindow, map.getCenter());
          }); }
            else { // Browser doesn't support Geolocation
              console.log("Browser doesn't support Geolocation");
              window.alert("Your browser doesn't support this app");
          handleLocationError(false, infowindow, map.getCenter()); } }
```

4.2. Cycle 2

Test Case: Case of invalid character in input (latitude && longitude)

Program Code (Red Line):

```
function geocodeLatLng(geocoder, map, infowindow) {
      var latitute = document.getElementById('lat').value;
      var longitude = document.getElementById('lng').value;
      var coordinatesLL = {lat: parseFloat(latitute), lng:
      parseFloat(longitude)};
      geocoder.geocode({'location': coordinatesLL}, function(results, status)
      {
      if (status === 'OK') {
            if (results[0]) {
                  map.setZoom(11);
                  var googleMarker = new google.maps.Marker({
                  position: coordinatesLL,
                  map: map });
                  infowindow.setContent(results[0].formatted_address);
                  infowindow.open(map, googleMarker);
                  map.setCenter(coordinatesLL);
                  var city = "";
                  var c, lc, component;
                  for (c = 0, lc = results[0].address_components.length; c <
                  lc; c += 1) {
                        component = results[0].address_components[c];
                        if (component.types[0] ===
                        'administrative_area_level_1') {
                        city = city + component.long_name + " "; } }
            var add= results[0].formatted_address ; var
            value=add.split(","); var count=value.length;  var
            country=value[count-1];
            document.getElementById("city").value= country + ", " + city ;}
            else {
            window.alert('No results found'); } }
      else {
            window.alert('Location cannot found because: ' + status);
      } }); }
```

Program Code (Green Line):

```
if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
     alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
     alert("Invalid character in latitude or longitude")
else
{
  // program code
}
```

Refactored Program Code: We benefit from refactoring techniques of simplifying method calls, composing methods and simplifying conditional expressions.

```
function findLocation(geocoder, map, infowindow) {

if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else{
      var latitute = document.getElementById('lat').value;
```

```
	var longitude = document.getElementById('lng').value;
	var coordinatesLL = {lat: parseFloat(latitute), lng:
	parseFloat(longitude)};
	geocoder.geocode({'location': coordinatesLL}, function(results, status)
	{
	if (status === 'OK') {
		if (results[0]) {
			map.setZoom(11);
			var googleMarker = new google.maps.Marker({
			position: coordinatesLL,
			map: map
			});
			infowindow.setContent(results[0].formatted_address);
			infowindow.open(map, googleMarker);
			map.setCenter(coordinatesLL);
			var city = "";
			var c, component;
			for (c = 0, c < results[0].address_components.length; c +=
			1) {
				component = results[0].address_components[c];
				if (component.types[0] ===
				'administrative_area_level_1') {
				city = city + component.long_name + " "; } }
		var add= results[0].formatted_address ;
		var value=add.split(",");
		var count=value.length;
		var country=value[count-1];
		document.getElementById("city").value= country + ", " + city ;}
		else { window.alert('No results found'); } }
	else { window.alert('Location cannot found because: ' + status); }});}}
```

4.3. Cycle 3

Test Case : Check whether program gives alert when latitude or longitude empty

Program Code (Red Line):

```
if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
	alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
	alert("Invalid character in latitude or longitude")

else{
	var latitute = document.getElementById('lat').value;
	var longitude = document.getElementById('lng').value;
	var coordinatesLL = {lat: parseFloat(latitute), lng:
	parseFloat(longitude)};
	geocoder.geocode({'location': coordinatesLL}, function(results, status)
	{
	if (status === 'OK') {
		if (results[0]) {
```

```
                    map.setZoom(11);
                    var googleMarker = new google.maps.Marker({
                    position: coordinatesLL,
                    map: map
                    });
                    infowindow.setContent(results[0].formatted_address);
                    infowindow.open(map, googleMarker);
                    map.setCenter(coordinatesLL);
              }
              var add= results[0].formatted_address ; var
              value=add.split(","); var count=value.length;  var
              country=value[count-1];
              }
              else {
              window.alert('No results found');
              }
        }
        else {
              window.alert('Location cannot found because: ' + status);
        }
        });
}
```

Program Code (Green Line):

```
if(document.getElementById('lat').value == "")
      alert("Latitude cannot be empty")
else if(document.getElementById('lng').value == "")
      alert("Longitude cannot be empty")
else
{
  // program code
}
```

Refactored Program Code: We benefit from refactoring techniques of composing methods.

```
function findLocation(geocoder, map, infowindow) {
if(document.getElementById('lat').value == "")
      alert("Latitude cannot be empty")
else if(document.getElementById('lng').value == "")
      alert("Longitude cannot be empty")
else if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")

else{
      var latitute = document.getElementById('lat').value;
      var longitude = document.getElementById('lng').value;
      var coordinatesLL = {lat: parseFloat(latitute), lng:
      parseFloat(longitude)};
```

```
        geocoder.geocode({'location': coordinatesLL}, function(results, status)
        {
        if (status === 'OK') {
                if (results[0]) {
                        map.setZoom(11);
                        var googleMarker = new google.maps.Marker({
                        position: coordinatesLL,
                        map: map
                        });
                        infowindow.setContent(results[0].formatted_address);
                        infowindow.open(map, googleMarker);
                        map.setCenter(coordinatesLL);
                }
                var value= (results[0].formatted_address).split(",");
                var count=value.length;
                var country=value[count-1];
                }
                else {
                window.alert('No results found');
                }
        }
        else {
                window.alert('Location cannot found because: ' + status);
        }
        });
}}
```

4.4. Cycle 4

Test Case: Check if the nearest location function works fine

Program Code (Red Line):

```
function findnearestcity(geocoder, map, infowindow){
        var address = document.getElementById('city').value;
        geocoder.geocode({'address': address}, function(results, status) {
        if (status === 'OK') {
                if (results[0]) {
                        map.setZoom(11);
                        var googleMarker = new google.maps.Marker({
                        position: results[0].geometry.location,
                        map: map});
                }
                else {
                        window.alert('No results found');
        }}
        else {
                window.alert('Location cannot found because: ' + status);

        }});}
```

Program Code (Green Line):

```
var city = "";
 var c, lc, component;
for (c = 0, lc = results[0].address_components.length; c < lc; c += 1) {
      component = results[0].address_components[c];
       if (component.types[0] === 'administrative_area_level_1') {
             city = city + component.long_name + " ";
       }
 }
var add= results[0].formatted_address ;
var value=add.split(",");
var count=value.length;
var country=value[count-1];
var lat1 = results[0].geometry.location.lat();
var lng1 = results[0].geometry.location.lng();
var lat2 = document.getElementById("lat").value;
var lng2 = document.getElementById("lng").value;
var flightPlanCoordinates = [{lat:
parseFloat(results[0].geometry.location.lat()), lng:
      parseFloat(results[0].geometry.location.lng())},
      {lat: parseFloat(document.getElementById("lat").value), lng:
      parseFloat(document.getElementById("lng").value)}
 ];
var flightPath = new google.maps.Polyline({
            path: flightPlanCoordinates,
            geodesic: true,
            strokeColor: '#FF0000',
            strokeOpacity: 1.0,
            strokeWeight: 2,
            map: map });
infowindow.setContent(results[0].formatted_address);
infowindow.open(map, googleMarker);
map.setCenter(results[0].geometry.location);
var r = 6371;
var dLng = (lng2 - lng1) * Math.PI / 180;
var dLat = (lat2 - lat1) * Math.PI / 180;
lat1 = lat1 * Math.PI / 180;
lat2 = document.getElementById("lat").value * Math.PI / 180;
var a = Math.sin(dLat/2) * Math.sin(dLat/2) + Math.sin(dLng/2) *
Math.sin(dLng/2) *
          Math.cos(lat1) * Math.cos(lat2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
var d = r * c;
document.getElementById("citydistance").value = d + " Km";
```

Refactored Program Code: We benefit from refactoring techniques of simplifying conditional expression and composing methods.

```
function findNearestCityCenter(geocoder, map, infowindow){
      var address = document.getElementById('city').value;
      geocoder.geocode({'address': address}, function(results, status) {
      if (status === 'OK') {
```

```
if (results[0]) {
      map.setZoom(11);
      var googleMarker = new google.maps.Marker({
      position: results[0].geometry.location,
      map: map
      });
      var city = "";
       var c, component;
      for (c = 0; c <results[0].address_components.length; c +=
      1) {
            component = results[0].address_components[c];
             if (component.types[0] ===
      'administrative_area_level_1') {
                   city = city + component.long_name + " ";
             }
       }
      var value= results[0].formatted_address.split(",");
      var count=value.length;
      var country=value[count-1];
      var lat1 = results[0].geometry.location.lat();
      var lng1 = results[0].geometry.location.lng();
      var lat2 = document.getElementById("lat").value;
      var lng2 = document.getElementById("lng").value;
      var flightPlanCoordinates = [{lat:
      parseFloat(results[0].geometry.location.lat()), lng:
            parseFloat(results[0].geometry.location.lng())},
            {lat:
      parseFloat(document.getElementById("lat").value), lng:
            parseFloat(document.getElementById("lng").value)}];
      var flightPath = new google.maps.Polyline({
                  path: flightPlanCoordinates,
                  geodesic: true,
                  strokeColor: '#FF0000',
                  strokeOpacity: 1.0,
                  strokeWeight: 2,
                  map: map });
      infowindow.setContent(results[0].formatted_address);
      infowindow.open(map, googleMarker);
      map.setCenter(results[0].geometry.location);
      var r = 6371;
      var dLng = (lng2 - lng1) * Math.PI / 180;
      var dLat = (lat2 - lat1) * Math.PI / 180;
      lat1 = lat1 * Math.PI / 180;
      lat2 = document.getElementById("lat").value * Math.PI /
      180;
      var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
      Math.sin(dLng/2) * Math.sin(dLng/2) *
                Math.cos(lat1) * Math.cos(lat2);
      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
      var d = r * c;
```

```
                         document.getElementById("citydistance").value = d + "
                        Km";
                         }
                        else { window.alert('No results found');}}
        else {
              window.alert('Location cannot found because: ' + status);
        }});}
```

4.5. Cycle 5

Test Case: Check whether program gives true alert when the lat or long is out of interval

Program Code (Red Line):

```
function findNearestCityCenter(geocoder, map, infowindow){
      var address = document.getElementById('city').value;
      geocoder.geocode({'address': address}, function(results, status) {
      if (status === 'OK') {
            if (results[0]) {
                  map.setZoom(11);
                  var googleMarker = new google.maps.Marker({
                  position: results[0].geometry.location,
                  map: map });
                  var city = "";
                   var c, lc, component;
                  for (c = 0, lc = results[0].address_components.length; c <
                  lc; c += 1) {
                        component = results[0].address_components[c];
                         if (component.types[0] ===
                  'administrative_area_level_1') {
                               city = city + component.long_name + " "; }}
                  var add= results[0].formatted_address ;
                  var value=add.split(",");
                  var count=value.length;
                  var country=value[count-1];
                  var lat1 = results[0].geometry.location.lat();
                  var lng1 = results[0].geometry.location.lng();
                  var lat2 = document.getElementById("lat").value;
                  var lng2 = document.getElementById("lng").value;
                  var flightPlanCoordinates = [{lat:
                  parseFloat(results[0].geometry.location.lat()), lng:
                        parseFloat(results[0].geometry.location.lng())},
                        {lat:
                  parseFloat(document.getElementById("lat").value), lng:
                        parseFloat(document.getElementById("lng").value)}];
                  var flightPath = new google.maps.Polyline({
                              path: flightPlanCoordinates,
                              geodesic: true,
                              strokeColor: '#FF0000',
                              strokeOpacity: 1.0,
                              strokeWeight: 2,
                              map: map});
                  infowindow.setContent(results[0].formatted_address);
                  infowindow.open(map, googleMarker);
                  map.setCenter(results[0].geometry.location);
                  var r = 6371;
                  var dLng = (lng2 - lng1) * Math.PI / 180;
                  var dLat = (lat2 - lat1) * Math.PI / 180;
                  lat1 = lat1 * Math.PI / 180;
                  lat2 = document.getElementById("lat").value * Math.PI /
                  180;
                  var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                  Math.sin(dLng/2) * Math.sin(dLng/2) *
                            Math.cos(lat1) * Math.cos(lat2);
                  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
                  var d = r * c;
                   document.getElementById("citydistance").value = d + "
                  Km"; }
                  else {
                  window.alert('No results found');}}
          else {
            window.alert('Location cannot found because: ' + status);
          }}); }
```

Program Code (Green Line):

```
if(parseFloat(document.getElementById('lat').value) > 90 ||
    parseFloat(document.getElementById('lat').value) < -90)
          alert("Latitude should be in the interval -90,90")
else if(parseFloat(document.getElementById('lng').value ) > 90||
     parseFloat(document.getElementById('lng').value ) < -90)
          alert("Longutude should be in the interval -90,90")
```

Refactored Program Code: We benefit from refactoring techniques of simplifying conditional expression and composing methods.

```
function findNearestCityCenter(geocoder, map, infowindow){
      if(parseFloat(document.getElementById('lat').value) > 90 ||
          parseFloat(document.getElementById('lat').value) < -90)
                alert("Latitude should be in the interval -90,90")
      else if(parseFloat(document.getElementById('lng').value ) > 90||
           parseFloat(document.getElementById('lng').value ) < -90)
                alert("Longutude should be in the interval -90,90")
      else{
      var address = document.getElementById('city').value;
      geocoder.geocode({'address': address}, function(results, status) {
      if (status === 'OK') {
            if (results[0]) {
                  map.setZoom(11);
                  var googleMarker = new google.maps.Marker({
                  position: results[0].geometry.location,
                  map: map });
                  var city = ""; var c, component;
```

```
                    for (c = 0; c < results[0].address_components.length; c +=
                    1) {
                          component = results[0].address_components[c];
                           if (component.types[0] ===
                    'administrative_area_level_1') {
                                 city = city + component.long_name + " "; } }
                    var value= results[0].formatted_address.split(",");
                    var count=value.length;
                    var country=value[count-1];
                    var lat1 = results[0].geometry.location.lat();
                    var lng1 = results[0].geometry.location.lng();
                    var lat2 = document.getElementById("lat").value;
                    var lng2 = document.getElementById("lng").value;
                    var flightPlanCoordinates = [{lat:
                    parseFloat(results[0].geometry.location.lat()), lng:
                          parseFloat(results[0].geometry.location.lng())},
                          {lat:
                    parseFloat(document.getElementById("lat").value), lng:
                          parseFloat(document.getElementById("lng").value)}];
                    var flightPath = new google.maps.Polyline({
                                path: flightPlanCoordinates,
                                geodesic: true,
                                strokeColor: '#FF0000',
                                strokeOpacity: 1.0,
                                strokeWeight: 2,
                                map: map});
                    infowindow.setContent(results[0].formatted_address);
                    infowindow.open(map, googleMarker);
                    map.setCenter(results[0].geometry.location);
                    var r = 6371;
                    var dLng = (lng2 - lng1) * Math.PI / 180;
                    var dLat = (lat2 - lat1) * Math.PI / 180;
                    lat1 = lat1 * Math.PI / 180;
                    lat2 = document.getElementById("lat").value * Math.PI /
                    180;
                    var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                    Math.sin(dLng/2) * Math.sin(dLng/2) *
                              Math.cos(lat1) * Math.cos(lat2);
                    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
                    var d = r * c;
                     document.getElementById("citydistance").value = d + "
                    Km";}
                    else {
                    window.alert('No results found');}}
         else {
            window.alert('Location cannot found because: ' + status);
      }}); }}
```

4.6. Cycle 6

Test Case: When entering Latitude and Longitude check if it gives city

Program Code (Red Line):

```
function findLocation(geocoder, map, infowindow) {
if(document.getElementById('lat').value == "")
      alert("Latitude cannot be empty")
else if(document.getElementById('lng').value == "")
      alert("Longitude cannot be empty")
else if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else{ var latitute = document.getElementById('lat').value;
      var longitude = document.getElementById('lng').value;
      var coordinatesLL = {lat: parseFloat(latitute), lng:
      parseFloat(longitude)};
      geocoder.geocode({'location': coordinatesLL}, function(results, status)
      {
      if (status === 'OK') {
            if (results[0]) {
                  map.setZoom(11);
                  var googleMarker = new google.maps.Marker({
                  position: coordinatesLL,
                  map: map
                  });
                  infowindow.setContent(results[0].formatted_address);
                  infowindow.open(map, googleMarker);
                  map.setCenter(coordinatesLL);
            else {
            window.alert('No results found');
            }
      }
      else {
            window.alert('Location cannot found because: ' + status);
      }});}}
```

Program Code (Green Line):

```
var city = "";
var c, lc, component;
for (c = 0, lc = results[0].address_components.length; c < lc; c += 1) {
    component = results[0].address_components[c];
    if (component.types[0] === 'administrative_area_level_1') {
        city = city + component.long_name + " ";
    }
}
var add= results[0].formatted_address ; var value=add.split(","); var
count=value.length;  var country=value[count-1];
document.getElementById("city").value= country + ", " + city;
```

Refactored Program Code: We benefit from refactoring techniques of simplifying conditional expression and composing methods.

```
function findLocation(geocoder, map, infowindow) {
if(document.getElementById('lat').value == "")
      alert("Latitude cannot be empty")
else if(document.getElementById('lng').value == "")
      alert("Longitude cannot be empty")
else if(!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else if(!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
      alert("Invalid character in latitude or longitude")
else{
      var latitute = document.getElementById('lat').value;
      var longitude = document.getElementById('lng').value;
      var coordinatesLL = {lat: parseFloat(latitute), lng:
      parseFloat(longitude)};
      geocoder.geocode({'location': coordinatesLL}, function(results, status)
      {
      if (status === 'OK') {
            if (results[0]) {
                  map.setZoom(11);
                  var googleMarker = new google.maps.Marker({
                  position: coordinatesLL,
                  map: map });
                  infowindow.setContent(results[0].formatted_address);
                  infowindow.open(map, googleMarker);
                  map.setCenter(coordinatesLL);
                  var city = "";
                  var c, lc, component;
                  for (c = 0; c < results[0].address_components.length; c +=
                  1) {
                        component = results[0].address_components[c];
                        if (component.types[0] ===
                        'administrative_area_level_1') {
                        city = city + component.long_name + " ";}}
            var value= results[0].formatted_address.split(",");
            var count=value.length;
            var country=value[count-1];
            document.getElementById("city").value= country + ", " + city ;}
            else {
            window.alert('No results found'); } }
      else {
            window.alert('Location cannot found because: ' + status);
      }});}}
```

4.7. Cycle 7

Test Case: Whether distance of location to earth center is true or not

Program Code (Red Line):

```
function distanceTo(geocoder, map, infowindow){
if (document.getElementById('lat').value == "")
```

```
        alert("Latitude cannot be empty")
    else if (document.getElementById('lng').value == "")
        alert("Longitude cannot be empty")
    else if (!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
        alert("Invalid character in latitude or longitude")
    else if (!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
        alert("Invalid character in latitude or longitude")
      else if(parseFloat(document.getElementById('lat').value) > 90 ||
parseFloat(document.getElementById('lat').value) < -90)
        alert("Latitude should be in the interval -90,90")
    else if(parseFloat(document.getElementById('lng').value ) > 90||
parseFloat(document.getElementById('lng').value ) < -90)
        alert("Longutude should be in the interval -90,90")
    else {
        var address = document.getElementById('city').value;
        geocoder.geocode({'address': address}, function(results, status) {
        if (status === 'OK') {
        if (results[0]) {
          map.setZoom(11);
          var googleMarker = new google.maps.Marker({
          position: results[0].geometry.location,
          map: map
          });
          var lat1 = 0;
          var lng1 = 0;
          var lat2 = document.getElementById("lat").value;
          var lng2 = document.getElementById("lng").value;
          var flightPlanCoordinates = [
            {lat: parseFloat(lat1), lng: parseFloat(lng1)},
            {lat: parseFloat(document.getElementById("lat").value), lng:
parseFloat(document.getElementById("lng").value)}
          ];
          var flightPath = new google.maps.Polyline({
            path: flightPlanCoordinates,
            geodesic: true,
            strokeColor: '#FF0000',
            strokeOpacity: 1.0,
            strokeWeight: 2,
            map: map
          });
          infowindow.setContent(results[0].formatted_address);
          infowindow.open(map, googleMarker);
          map.setCenter(results[0].geometry.location);
          document.getElementById("citydistance").value = d + " Km";
        }
        else {
          window.alert('No results found');
        }
        }
        else {
          window.alert('Location cannot found because: ' + status);
        }
      });
    }
}
```

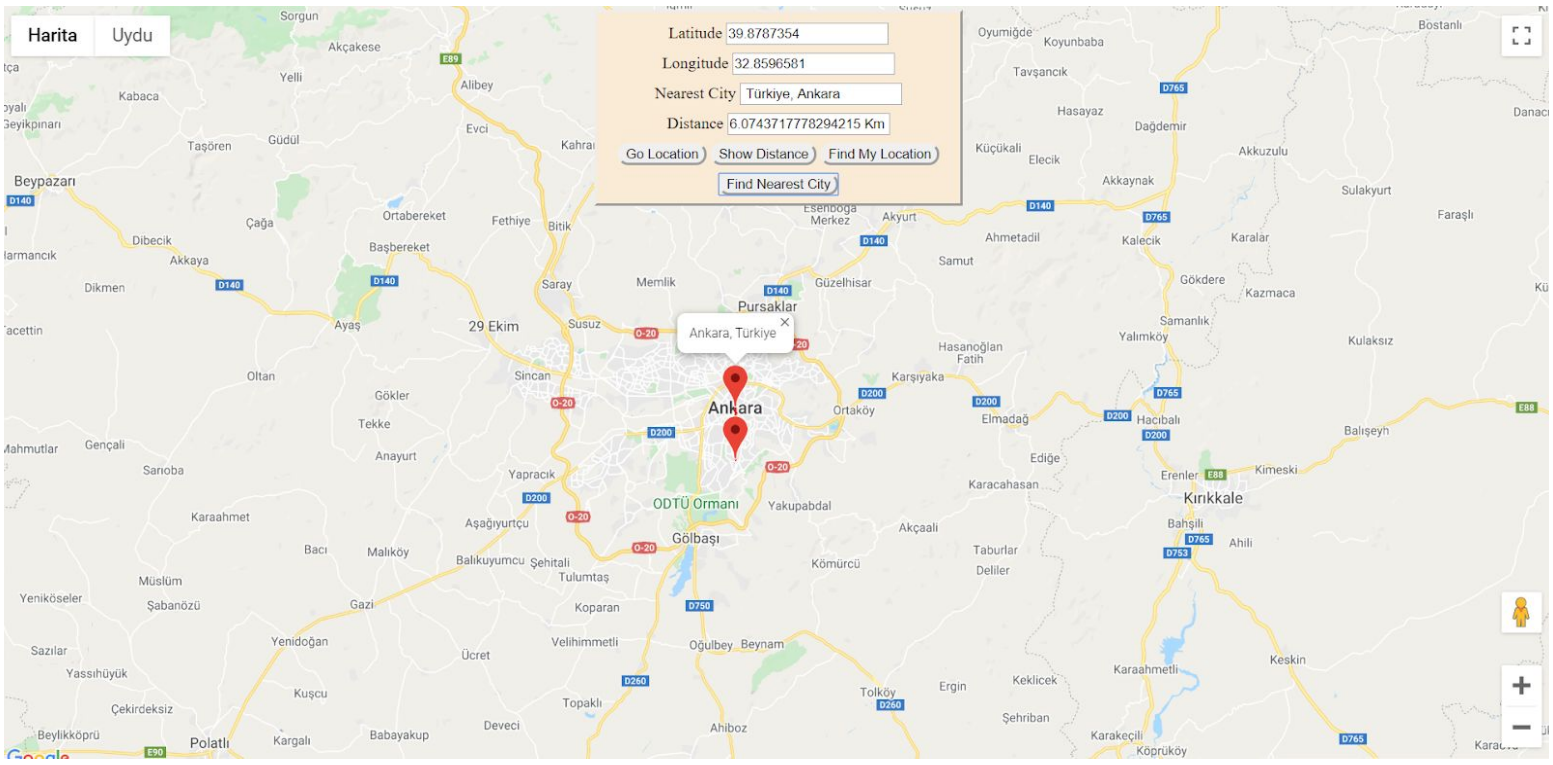
Program Code (Green Line):

```
var r = 6371;
var dLng = (lng2 - lng1) * Math.PI / 180;
var dLat = (lat2 - lat1) * Math.PI / 180;
lat1 = lat1 * Math.PI / 180;
lat2 = document.getElementById("lat").value * Math.PI / 180;
var a = Math.sin(dLat/2) * Math.sin(dLat/2) + Math.sin(dLng/2) *
Math.sin(dLng/2) * Math.cos(lat1) * Math.cos(lat2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
var d = r * c;
document.getElementById("citydistance").value = d + " Km";
```

Refactored Program Code: We benefit from refactoring techniques of simplifying method calls and composing methods.

```
function distanceToEarthCenter(geocoder, map, infowindow){
if (document.getElementById('lat').value == "")
        alert("Latitude cannot be empty")
    else if (document.getElementById('lng').value == "")
        alert("Longitude cannot be empty")
    else if (!document.getElementById('lat').value.match("^[0-9\.\-]+$"))
        alert("Invalid character in latitude or longitude")
    else if (!document.getElementById('lng').value.match("^[0-9\.\-]+$"))
        alert("Invalid character in latitude or longitude")
      else if(parseFloat(document.getElementById('lat').value) > 90 ||
parseFloat(document.getElementById('lat').value) < -90)
        alert("Latitude should be in the interval -90,90")
    else if(parseFloat(document.getElementById('lng').value ) > 90||
parseFloat(document.getElementById('lng').value ) < -90)
        alert("Longutude should be in the interval -90,90")
    else {
        var address = document.getElementById('city').value;
        geocoder.geocode({'address': address}, function(results, status) {
        if (status === 'OK') {
                  if (results[0]) {
                    map.setZoom(11);
                    var googleMarker = new google.maps.Marker({
                    position: results[0].geometry.location,
                    map: map
                    });
                    var lat1 = 0;
                    var lng1 = 0;
                    var lat2 = document.getElementById("lat").value;
                    var lng2 = document.getElementById("lng").value;
                    var flightPlanCoordinates = [
                        {lat: parseFloat(lat1), lng: parseFloat(lng1)},
                        {lat: parseFloat(lat2), lng: parseFloat(lng2)}
                    ];
                    var flightPath = new google.maps.Polyline({
                        path: flightPlanCoordinates,
                        geodesic: true,
                        strokeColor: '#FF0000',
                        strokeOpacity: 1.0,
                        strokeWeight: 2,
                        map: map
                    });
                    infowindow.setContent(results[0].formatted_address);
                    infowindow.open(map, googleMarker);
                    map.setCenter(results[0].geometry.location);
                    var dLng = (lng2 - lng1) * Math.PI / 180;
                    var dLat = (lat2 - lat1) * Math.PI / 180;
                    lat1 = lat1 * Math.PI / 180;
                    lat2 = document.getElementById("lat").value * Math.PI /
180;
                    var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
Math.sin(dLng/2) * Math.sin(dLng/2) * Math.cos(lat1) * Math.cos(lat2);
                    var d = 6371 * 2 * Math.atan2(Math.sqrt(a),
Math.sqrt(1-a));
                    document.getElementById("citydistance").value = d + "
Km";
                  }
                  else { window.alert('No results found'); }
        }
        else { window.alert('Location cannot found because: ' + status); }
      });
    }
}
```

5. Screenshots of Program



Latitude 39.8787354

Longitude 32.8596581

Nearest City Türkiye, Ankara

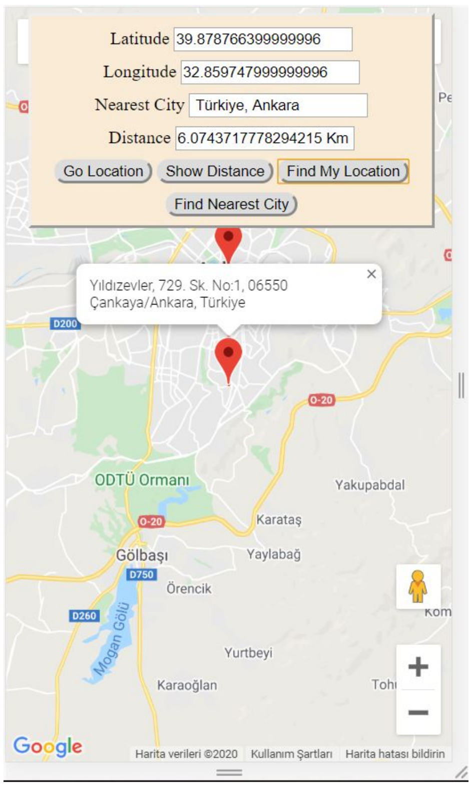
Distance 6.0743717778294215 Km

Go Location | Show Distance | Find My Location

Find Nearest City

Ankara, Türkiye

Figure 1: Screenshot of web page



Latitude 39.878766399999996

Longitude 32.859747999999996

Nearest City Türkiye, Ankara

Distance 6.0743717778294215 Km

Go Location | Show Distance | Find My Location

Find Nearest City

Yıldızevler, 729. Sk. No:1, 06550 Çankaya/Ankara, Türkiye

Figure 2: Screenshot of mobile web page