CS 458
Software Validation and Verification

Introduction to Test Driven Development (TDD)

Project #3

Okan Şen	21202377
Hassam Abdullah	21701610
Mustafa Tuna Acar	21703639

Introduction to Test Driven Development (TDD)

Project #3

1 Code Details/Information 2

1.1 Screenshot of Web Application 2

Home Page (Find Coordinate tab) 2

Home Page (Find Coordinate tab) Screen Size Reduced 3

Home Page (Find Distance to the Earth’s core tab) 4

Home Page (Find Distance to Big Ben tab) 5

1.2 Excerpts of Web Page Code 5

1.3 Screenshot of Test Code and Test Cases 8

1.3.1 Invalid/Valid Entries for lat/long in the Find Coordinates Tab 8

1.3.2 Invalid/Valid Entries for lat/long in the Distance to Earth Core Tab 9

1.3.3 Invalid/Valid Entries for lat/long in the Distance to Big Ben Tab 9

1.3.4 Testing the automated device location button on all functional Tabs 9

1.3.5 Responsiveness and Multiple Browser Testing 10

1.4 Excerpts of Test Code 11

2 UML Diagrams 13

2.1 Class Diagram 13

2.2 Use Case Diagram 14

2.3 Sequence Diagram 15

3 TDD experience 16

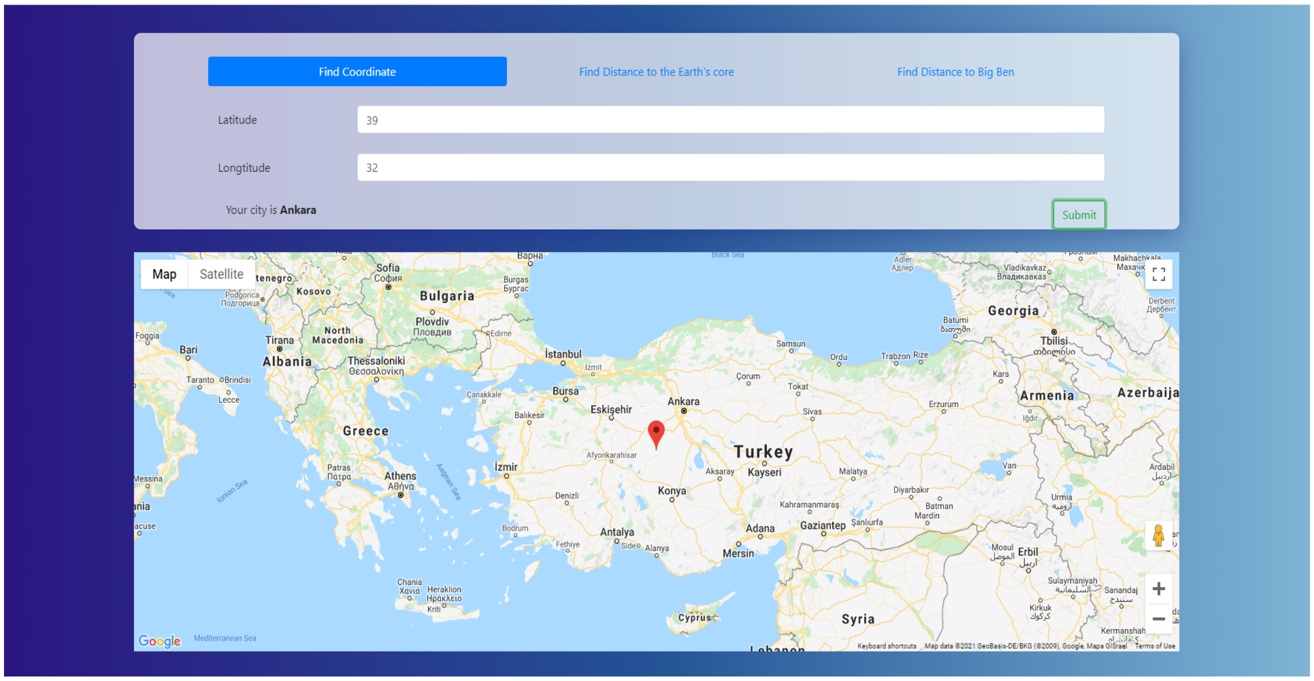
4.1 Development Velocity 16

4.2 Code Quality 16

1 Code Details/Information

1.1 Screenshot of Web Application

Home Page (Find Coordinate tab)



Latitude 39

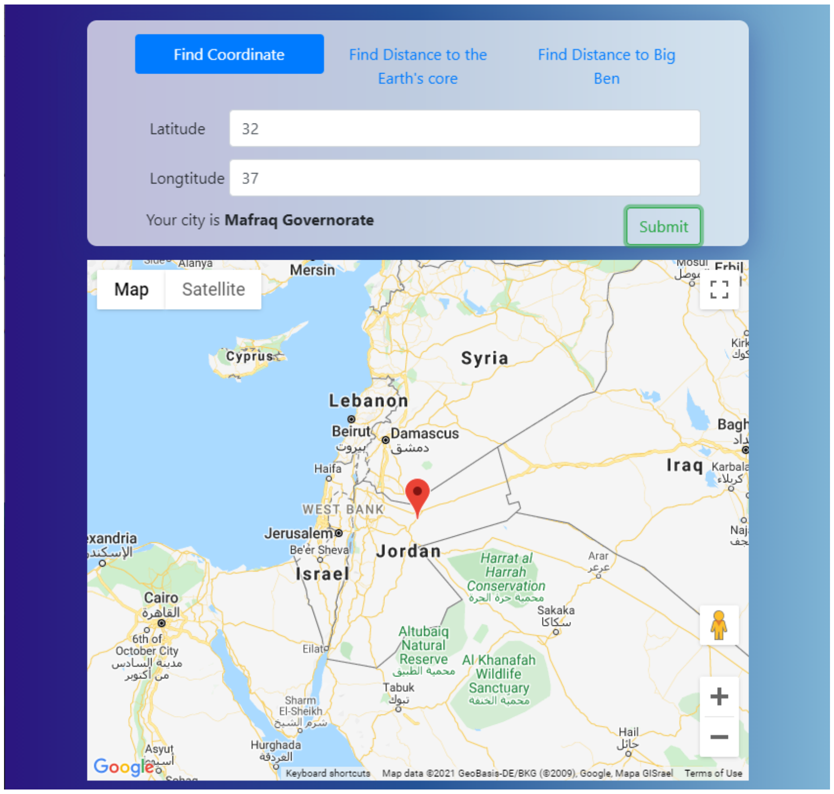
Longtitude 32

Your city is **Ankara**

Figure 1: Home Page

Figure 1, shows the main homepage of the website. By default, the Find Coordinate tab is selected and the user can insert valid latitude and longitude values into the fields shown and find the coordinate on the google maps interface on the website. In the figure, the coordinates for Ankara are inputted and the result is shown on the map.

Home Page (Find Coordinate tab) Screen Size Reduced



Latitude 32

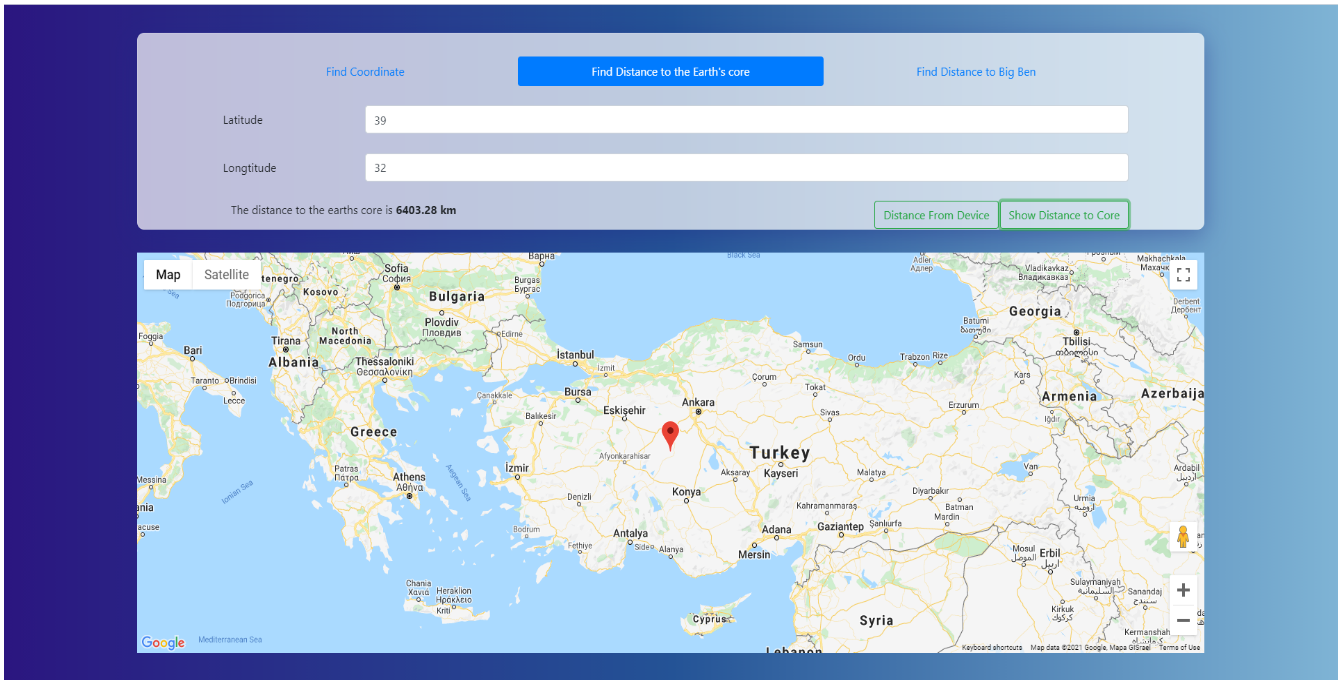
Longtitude 37

Your city is **Mafraq Governorate**

Figure 2: Home Page Screen Size Reduced

Figure 2, shows the responsiveness of the home page when the browser screen is minimized to a smaller resolution and adjusts accordingly to any resolution.

Home Page (Find Distance to the Earth's core tab)



Latitude 39

Longtitude 32

The distance to the earths core is **6403.28 km**

Figure 3: Home Page Find Distance to Earth's Core

Formula for the calculation

latitude B, radius R, radius at equator r_1, radius at pole r_2

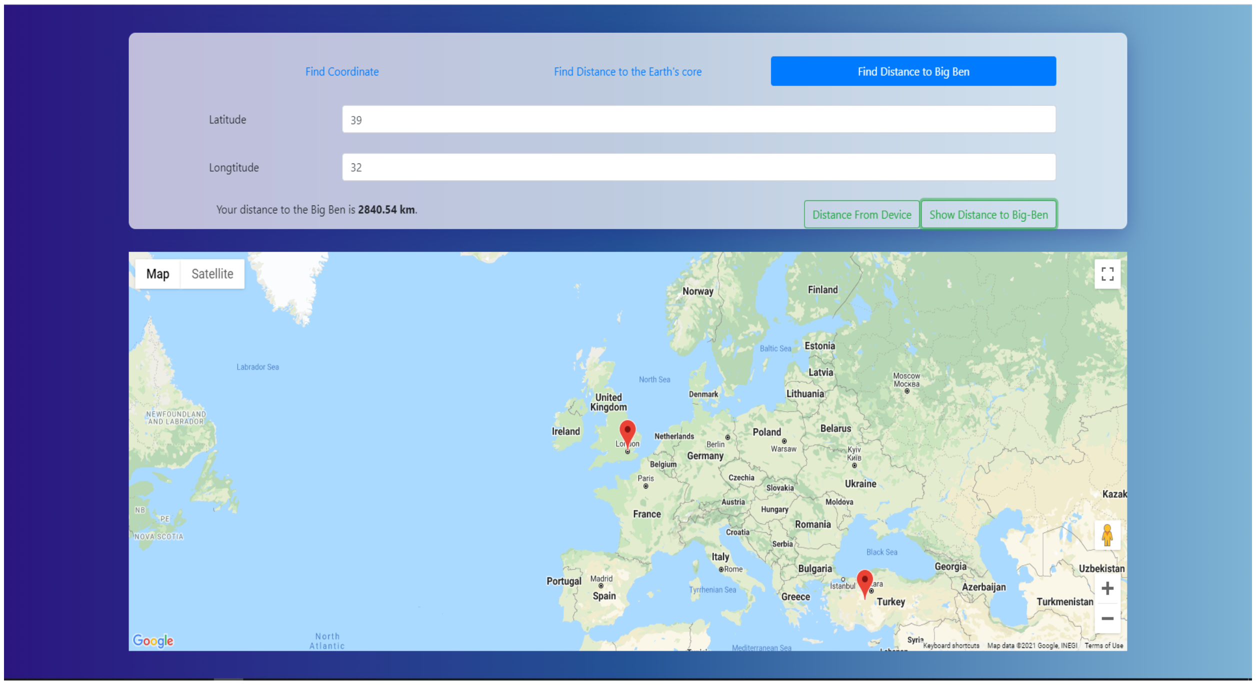
$$R = \sqrt{[(r_1^2 * \cos(B))^2 + (r_2^2 * \sin(B))^2] / [(r_1 * \cos(B))^2 + (r_2 * \sin(B))^2]}$$

Height above sea level can be negative, then you are below sea level.

Figure 4: Formula for Calculating Distance to Earth's Core

Figure 3, shows the second tab of the main homepage selected. The user can insert valid latitude and longitude values into the fields shown and find the distance to the Earth's core. In the figure, the coordinates for Ankara are inputted and the distance to the Earth's core is shown by clicking the Show Distance to Core button. This is calculated using the formula shown in Figure 4 and added with the elevation of the coordinate position found using Maps Elevation API. Furthermore, the user can also click the Distance From Device button to automatically find the distance using the device's location.

Home Page (Find Distance to Big Ben tab)



Latitude 39

Longtitude 32

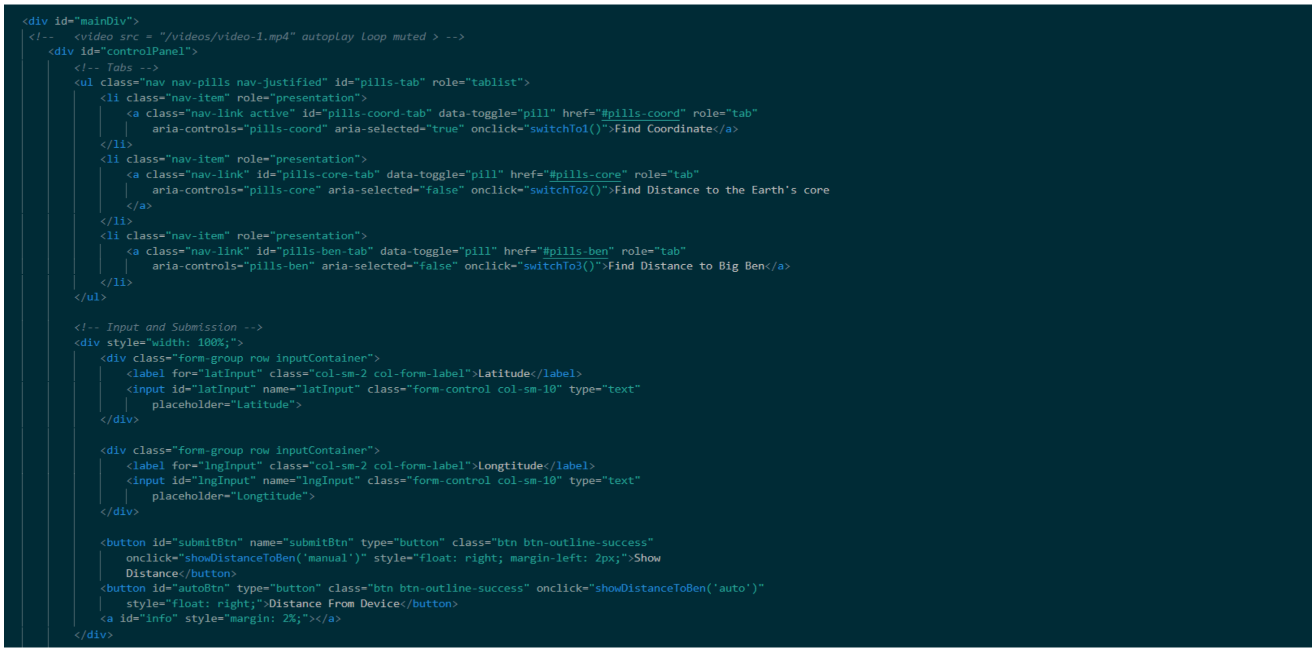
Your distance to the Big Ben is **2840.54 km**.

Figure 5: Home Page Find Distance to Big Ben

Figure 5, shows the third tab of the main homepage selected. The user can insert valid latitude and longitude values into the fields shown and find the distance to Big Ben from the inputted coordinates. In the figure, the coordinates for Ankara are inputted and the distance to Big Ben is shown by clicking the Show Distance to Big Ben button. Furthermore, the user can also click the Distance From Device button to automatically find the distance using the device's location.

1.2 Excerpts of Web Page Code

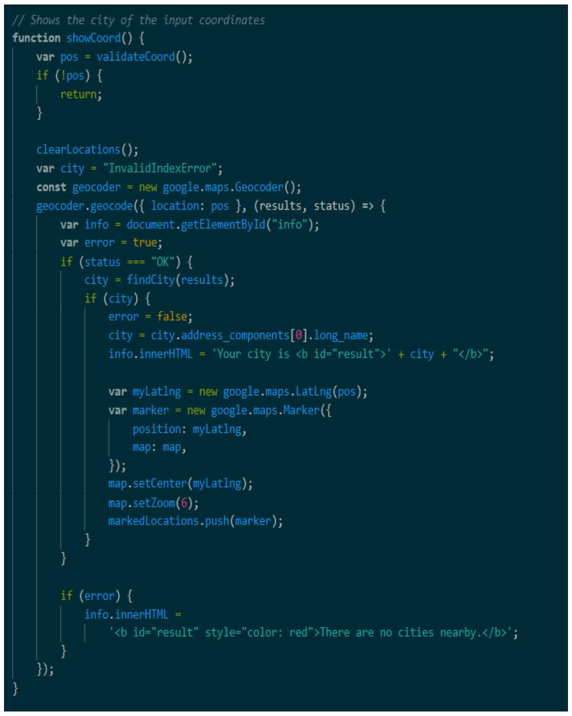
Important excerpts of code containing PHP/HTML/Javascript etc can be viewed below that are integral to the functioning and understanding of the website.



```html
<div id="mainDiv">
<!--   <video src = "/videos/video-1.mp4" autoplay loop muted > -->
    <div id="controlPanel">
        <!-- Tabs -->
        <ul class="nav nav-pills nav-justified" id="pills-tab" role="tablist">
            <li class="nav-item" role="presentation">
                <a class="nav-link active" id="pills-coord-tab" data-toggle="pill" href="#pills-coord" role="tab"
                    aria-controls="pills-coord" aria-selected="true" onclick="switchTo1()">Find Coordinate</a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="pills-core-tab" data-toggle="pill" href="#pills-core" role="tab"
                    aria-controls="pills-core" aria-selected="false" onclick="switchTo2()">Find Distance to the Earth's core
                </a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="pills-ben-tab" data-toggle="pill" href="#pills-ben" role="tab"
                    aria-controls="pills-ben" aria-selected="false" onclick="switchTo3()">Find Distance to Big Ben</a>
            </li>
        </ul>

        <!-- Input and Submission -->
        <div style="width: 100%;">
            <div class="form-group row inputContainer">
                <label for="latInput" class="col-sm-2 col-form-label">Latitude</label>
                <input id="latInput" name="latInput" class="form-control col-sm-10" type="text"
                    placeholder="Latitude">
            </div>

            <div class="form-group row inputContainer">
                <label for="lngInput" class="col-sm-2 col-form-label">Longtitude</label>
                <input id="lngInput" name="lngInput" class="form-control col-sm-10" type="text"
                    placeholder="Longtitude">
            </div>

            <button id="submitBtn" name="submitBtn" type="button" class="btn btn-outline-success"
                onclick="showDistanceToBen('manual')" style="float: right; margin-left: 2px;">Show
                Distance</button>
            <button id="autoBtn" type="button" class="btn btn-outline-success" onclick="showDistanceToBen('auto')"
                style="float: right;">Distance From Device</button>
            <a id="info" style="margin: 2%;"></a>
        </div>
```

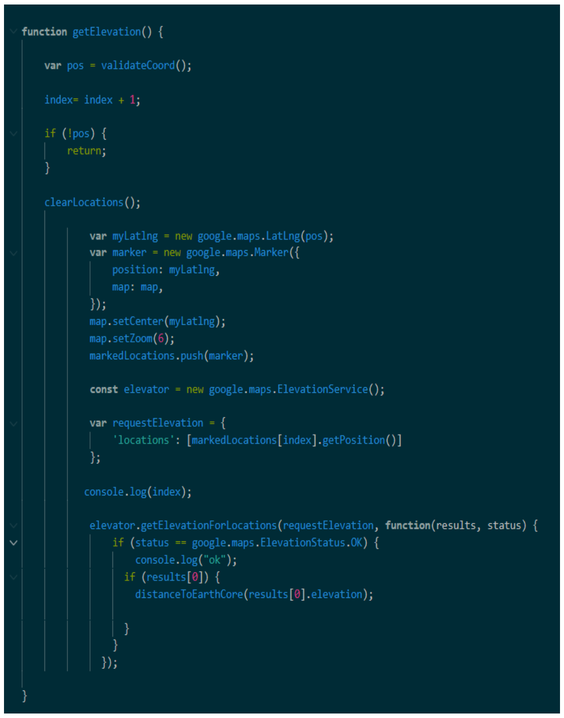
Figure 6: Web Page Code Excerpt 1

Figure 6, shows the HTML of the main homepage of the website that uses Bootstrap and Jquery to ensure the responsiveness of the website.



```javascript
// Shows the city of the input coordinates
function showCoord() {
    var pos = validateCoord();
    if (!pos) {
        return;
    }

    clearLocations();
    var city = "InvalidIndexError";
    const geocoder = new google.maps.Geocoder();
    geocoder.geocode({ location: pos }, (results, status) => {
        var info = document.getElementById("info");
        var error = true;
        if (status === "OK") {
            city = findCity(results);
            if (city) {
                error = false;
                city = city.address_components[0].long_name;
                info.innerHTML = 'Your city is <b id="result">' + city + "</b>";

                var myLatlng = new google.maps.LatLng(pos);
                var marker = new google.maps.Marker({
                    position: myLatlng,
                    map: map,
                });
                map.setCenter(myLatlng);
                map.setZoom(6);
                markedLocations.push(marker);
            }
        }

        if (error) {
            info.innerHTML =
                '<b id="result" style="color: red">There are no cities nearby.</b>';
        }
    });
}
```

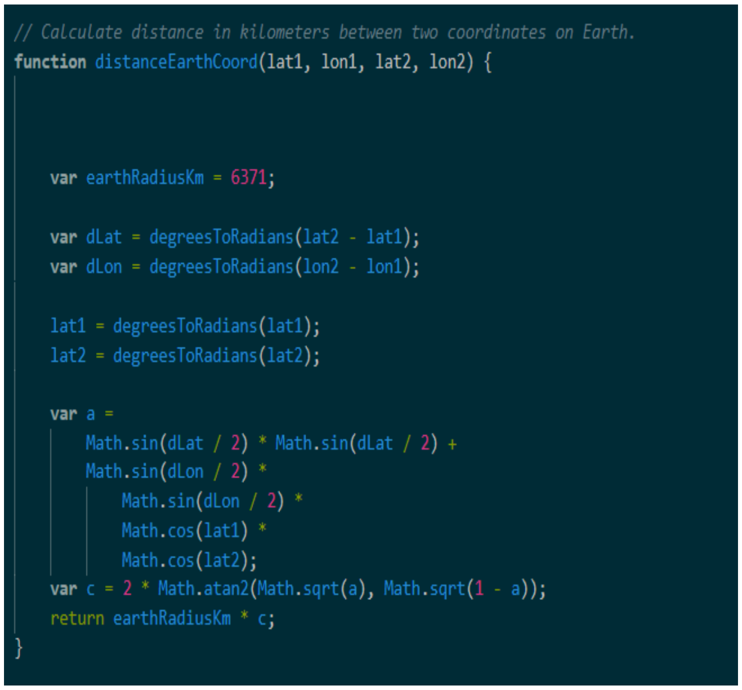
Figure 7: Web Page Code Excerpt 2



```javascript
function getElevation() {

    var pos = validateCoord();

    index= index + 1;

    if (!pos) {
        return;
    }

    clearLocations();

            var myLatlng = new google.maps.LatLng(pos);
            var marker = new google.maps.Marker({
                position: myLatlng,
                map: map,
            });
            map.setCenter(myLatlng);
            map.setZoom(6);
            markedLocations.push(marker);

            const elevator = new google.maps.ElevationService();

            var requestElevation = {
                'locations': [markedLocations[index].getPosition()]
            };

           console.log(index);

            elevator.getElevationForLocations(requestElevation, function(results, status) {
                if (status == google.maps.ElevationStatus.OK) {
                    console.log("ok");
                  if (results[0]) {
                    distanceToEarthCore(results[0].elevation);

                  }
                }
              });

}
```

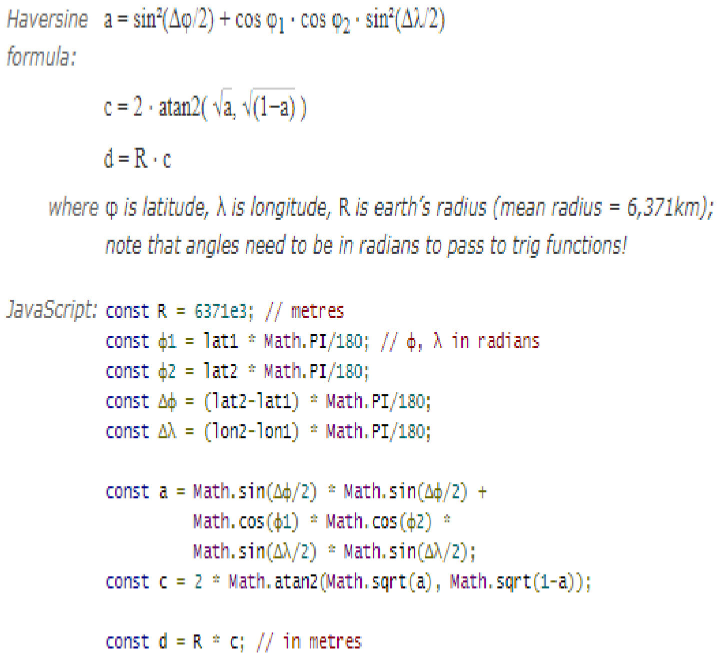
Figure 8: Web Page Code Excerpt 3

Figure 7, shows the method to show the coordinates on the map embedded into the website using the Google Maps API, while Figure 8, shows the getElevation method used in computing the distance to the Earth's core that uses the Maps Elevation API.



```javascript
// Calculate distance in kilometers between two coordinates on Earth.
function distanceEarthCoord(lat1, lon1, lat2, lon2) {



    var earthRadiusKm = 6371;

    var dLat = degreesToRadians(lat2 - lat1);
    var dLon = degreesToRadians(lon2 - lon1);

    lat1 = degreesToRadians(lat1);
    lat2 = degreesToRadians(lat2);

    var a =
        Math.sin(dLat / 2) * Math.sin(dLat / 2) +
        Math.sin(dLon / 2) *
            Math.sin(dLon / 2) *
            Math.cos(lat1) *
            Math.cos(lat2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    return earthRadiusKm * c;
}
```

Figure 9: Web Page Code Excerpt 4



Haversine formula: $a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!

JavaScript:

```javascript
const R = 6371e3; // metres
const φ1 = lat1 * Math.PI/180; // φ, λ in radians
const φ2 = lat2 * Math.PI/180;
const Δφ = (lat2-lat1) * Math.PI/180;
const Δλ = (lon2-lon1) * Math.PI/180;

const a = Math.sin(Δφ/2) * Math.sin(Δφ/2) +
          Math.cos(φ1) * Math.cos(φ2) *
          Math.sin(Δλ/2) * Math.sin(Δλ/2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

const d = R * c; // in metres
```

Figure 10: Calculation of Distance to Earth's Core

Figure 9, shows the method used to calculate the distance between two coordinates. This method is integral since it is used in the third tab to calculate the distance between the user's selection and the Big Ben using the formula shown in Figure 10.

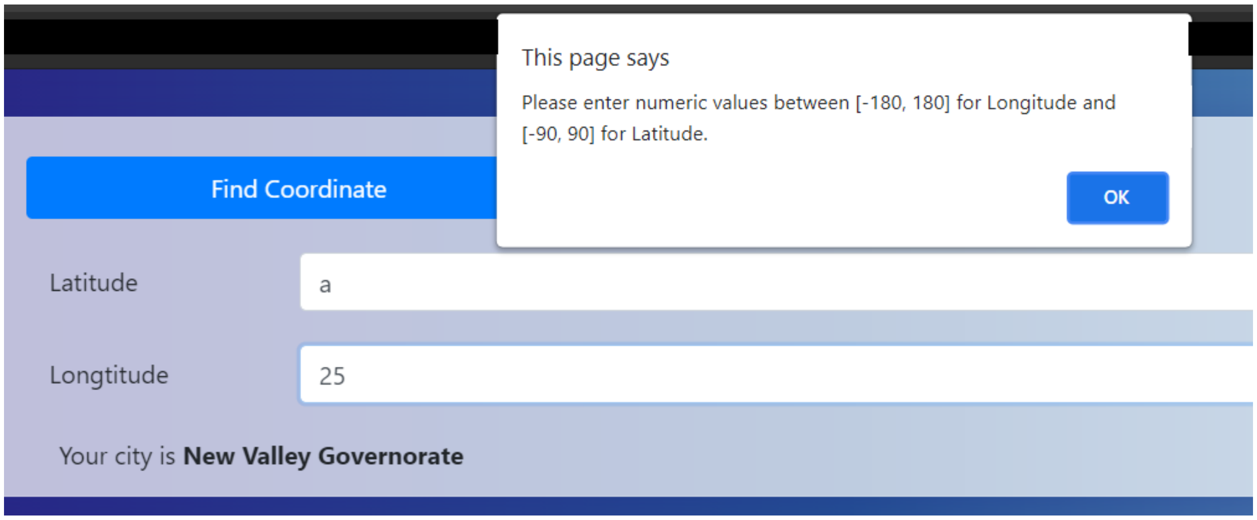
1.3 Screenshot of Test Code and Test Cases

The following test cases have been implemented to ensure the integrity of the web page's functionalities:-

1) Invalid/Valid Entries for lat/long in the Find Coordinates Tab
2) Invalid/Valid Entries for lat/long in the Distance to Earth Core Tab
3) Invalid/Valid Entries for lat/long in the Distance to Big Ben Tab
4) Testing the automated device location button on all functional Tabs
5) Responsiveness and Multiple Browser Testing

1.3.1 Invalid/Valid Entries for lat/long in the Find Coordinates Tab

The fields will not accept special characters/strings/white spaces at all times.


This page says

Please enter numeric values between [-180, 180] for Longitude and [-90, 90] for Latitude.

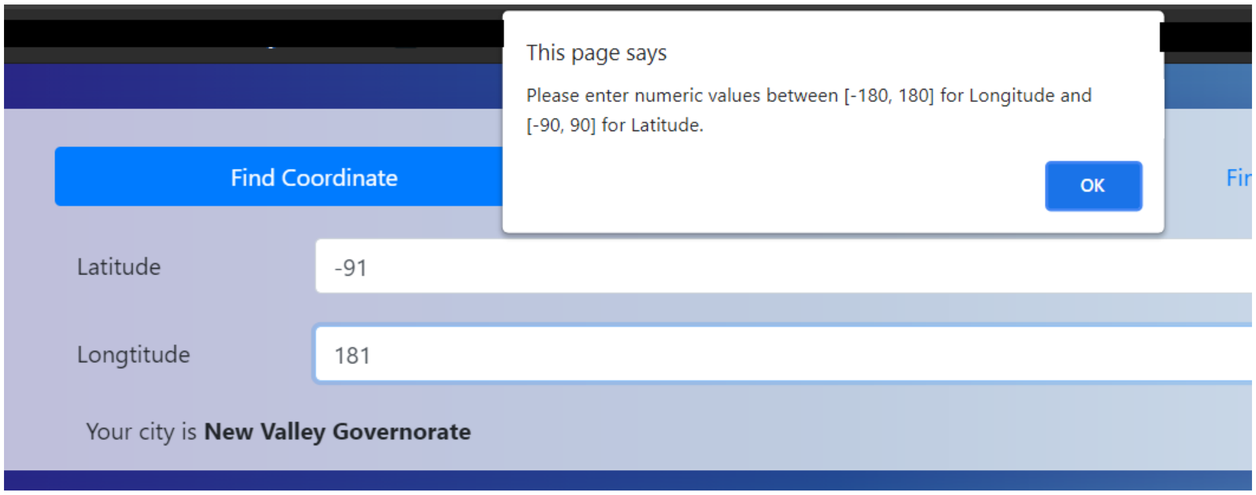
Latitude: a

Longtitude: 25

Your city is **New Valley Governorate**

Figure 11: Invalid Error Message

Additionally, all integer values must be between [-180,180] for Longitude and [-90,90] for Latitude.


This page says

Please enter numeric values between [-180, 180] for Longitude and [-90, 90] for Latitude.

Your city is **New Valley Governorate**

Figure 12: Invalid Error Message 2

1.3.2 Invalid/Valid Entries for lat/long in the Distance to Earth Core Tab

The same pattern can be observed in the distance to the Earth core tab of the web page which can be seen in the Find Coordinates Tab, also can be seen in the demo video.

1.3.3 Invalid/Valid Entries for lat/long in the Distance to Big Ben Tab

Additionally, this feature can be viewed again in the previous two tabs, as well as in the demo video.

1.3.4 Testing the automated device location button on all functional Tabs

The web page, when given permission (not needed when doing automated testing), can automatically detect the current location of the user, as can be seen in Figure 13.

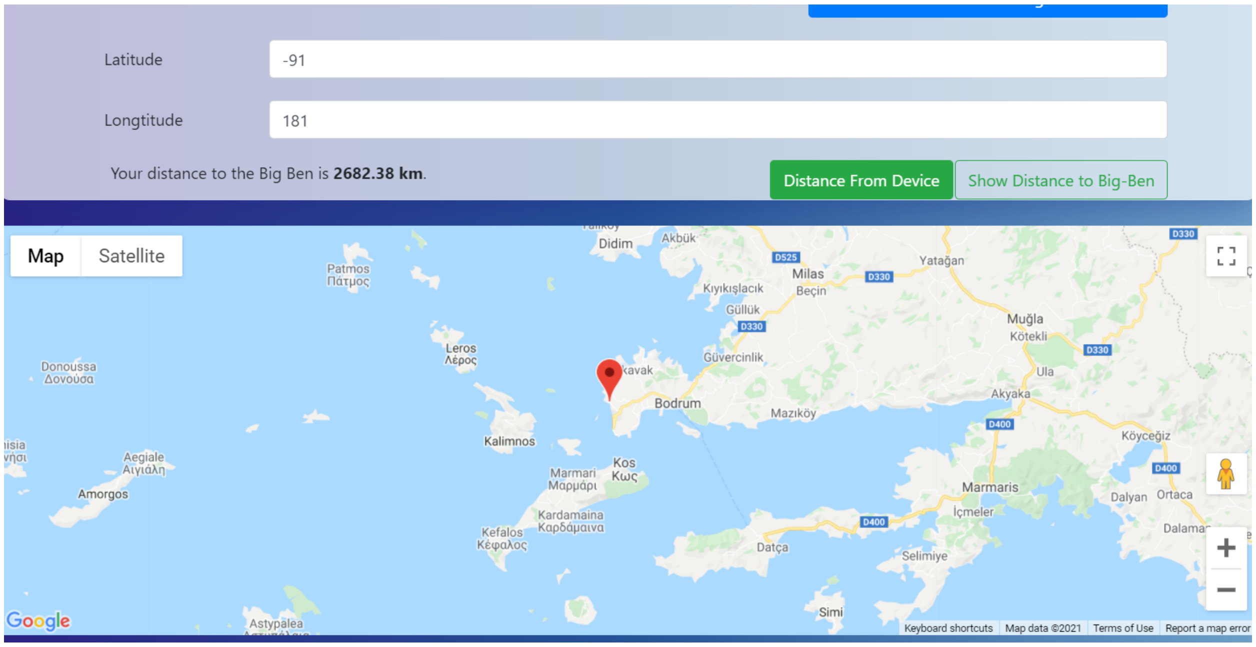
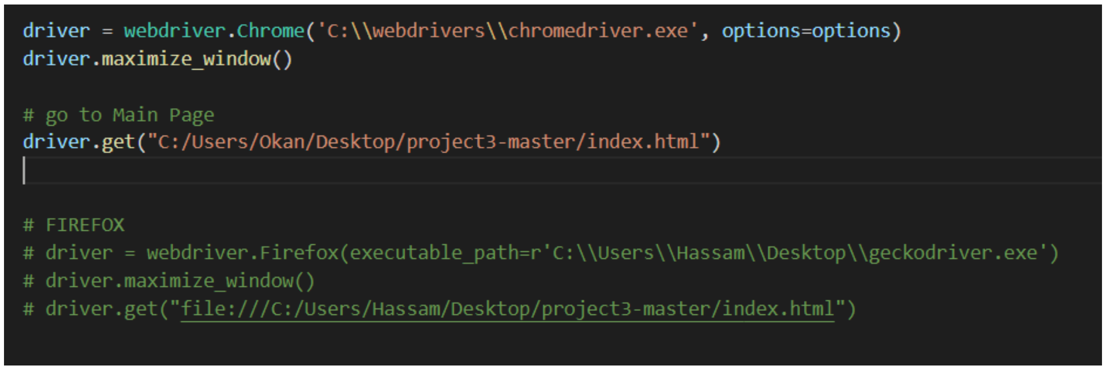


Figure 13: Location Detection Test

1.3.5 Responsiveness and Multiple Browser Testing

Firstly, it should be mentioned that the tests were done both in Chrome and Firefox. However, the code that's been uploaded has the Firefox driver part commented out. If you wish to test using Firefox, make sure to set up gecko drivers for Firefox automation, and comment out Chrome driver parts in the code.



```
driver = webdriver.Chrome('C:\\webdrivers\\chromedriver.exe', options=options)
driver.maximize_window()

# go to Main Page
driver.get("C:/Users/Okan/Desktop/project3-master/index.html")


# FIREFOX
# driver = webdriver.Firefox(executable_path=r'C:\\Users\\Hassam\\Desktop\\geckodriver.exe')
# driver.maximize_window()
# driver.get("file:///C:/Users/Hassam/Desktop/project3-master/index.html")
```

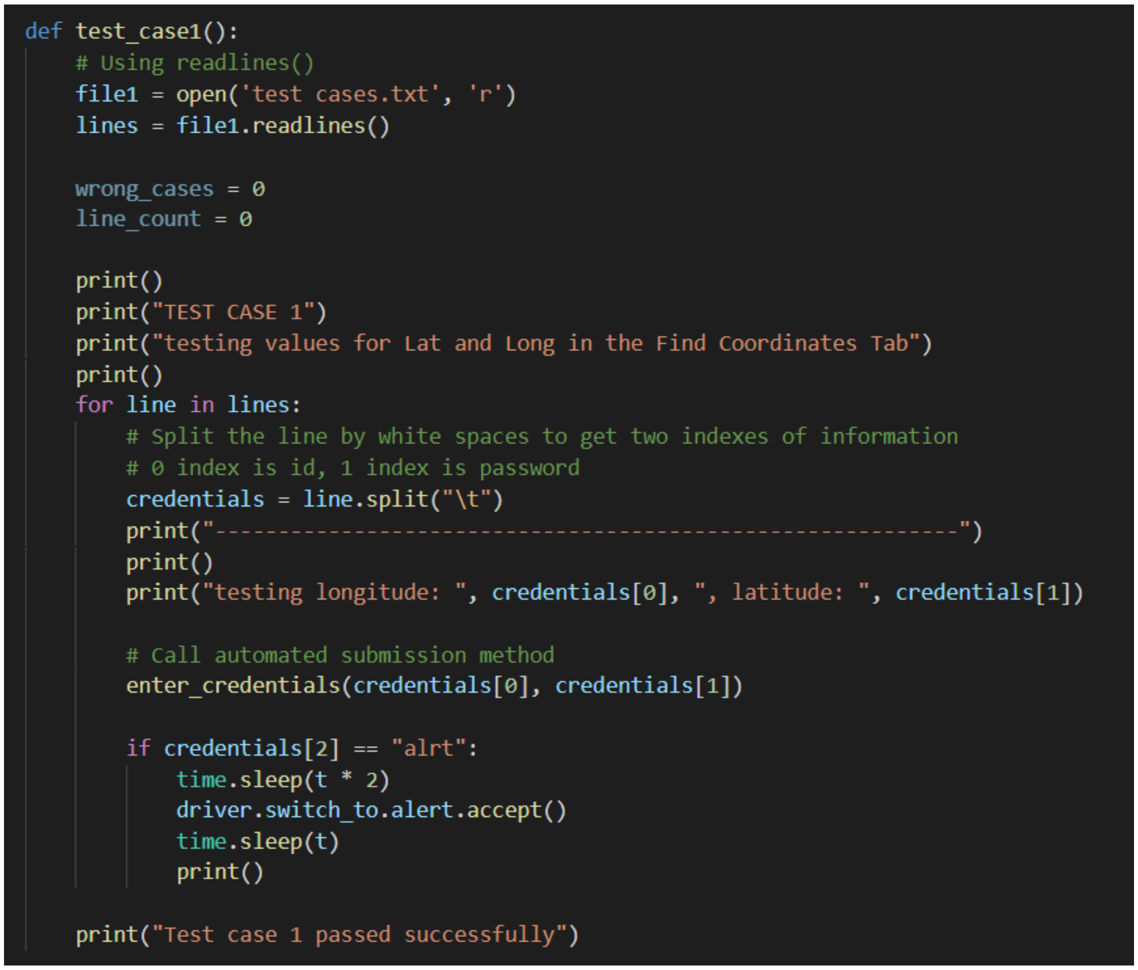
Figure 14: Driver Selection Code

The web page also responds to user rescaling the browser which can be viewed in the demo video.

1.4 Excerpts of Test Code

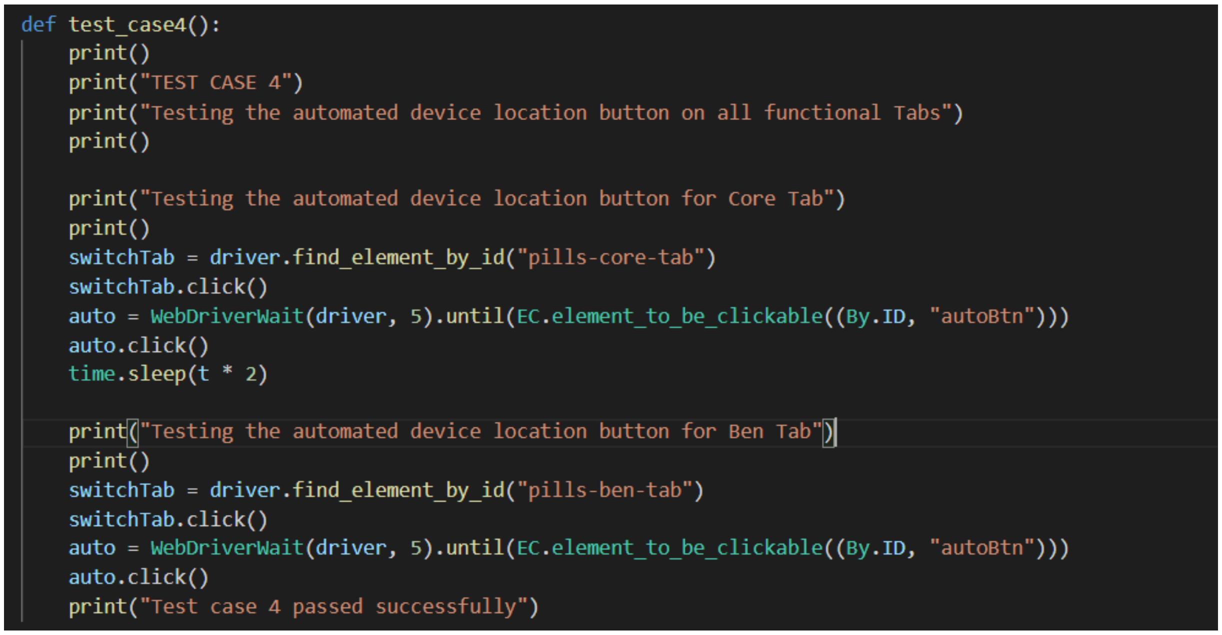
The test code runs on either Chrome or Firefox. They can be tested after the other, however, the test cases would have been redundant if such was a case. As mentioned in the previous section, either driver can be toggled on or off by commenting out or commenting in related codes.

Additionally, we have divided our test cases into separate methods and these methods are called in the main function one after another. These methods read test cases from separate text files.



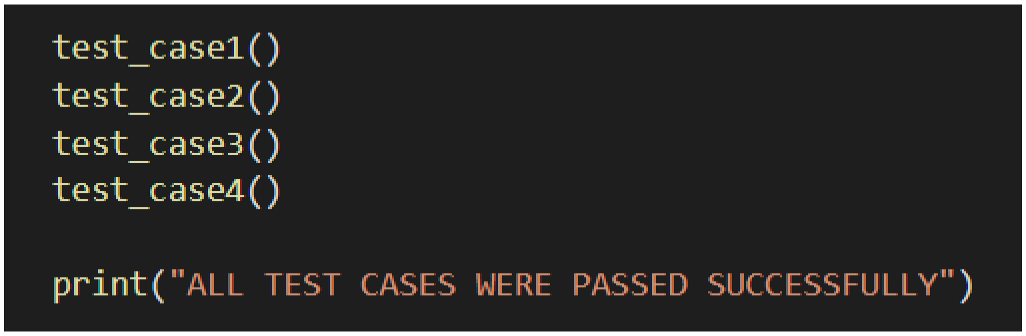
```python
def test_case1():
    # Using readlines()
    file1 = open('test cases.txt', 'r')
    lines = file1.readlines()

    wrong_cases = 0
    line_count = 0

    print()
    print("TEST CASE 1")
    print("testing values for Lat and Long in the Find Coordinates Tab")
    print()
    for line in lines:
        # Split the line by white spaces to get two indexes of information
        # 0 index is id, 1 index is password
        credentials = line.split("\t")
        print("----------------------------------------------------------")
        print()
        print("testing longitude: ", credentials[0], ", latitude: ", credentials[1])

        # Call automated submission method
        enter_credentials(credentials[0], credentials[1])

        if credentials[2] == "alrt":
            time.sleep(t * 2)
            driver.switch_to.alert.accept()
            time.sleep(t)
            print()

    print("Test case 1 passed successfully")
```

Figure 15: Test Case 1 Code



```
def test_case4():
    print()
    print("TEST CASE 4")
    print("Testing the automated device location button on all functional Tabs")
    print()

    print("Testing the automated device location button for Core Tab")
    print()
    switchTab = driver.find_element_by_id("pills-core-tab")
    switchTab.click()
    auto = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID, "autoBtn")))
    auto.click()
    time.sleep(t * 2)

    print("Testing the automated device location button for Ben Tab")
    print()
    switchTab = driver.find_element_by_id("pills-ben-tab")
    switchTab.click()
    auto = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID, "autoBtn")))
    auto.click()
    print("Test case 4 passed successfully")
```

Figure 16: Test Case 4 Code



```
test_case1()
test_case2()
test_case3()
test_case4()

print("ALL TEST CASES WERE PASSED SUCCESSFULLY")
```

Figure 17: Main Function Code

2 UML Diagrams

2.1 Class Diagram

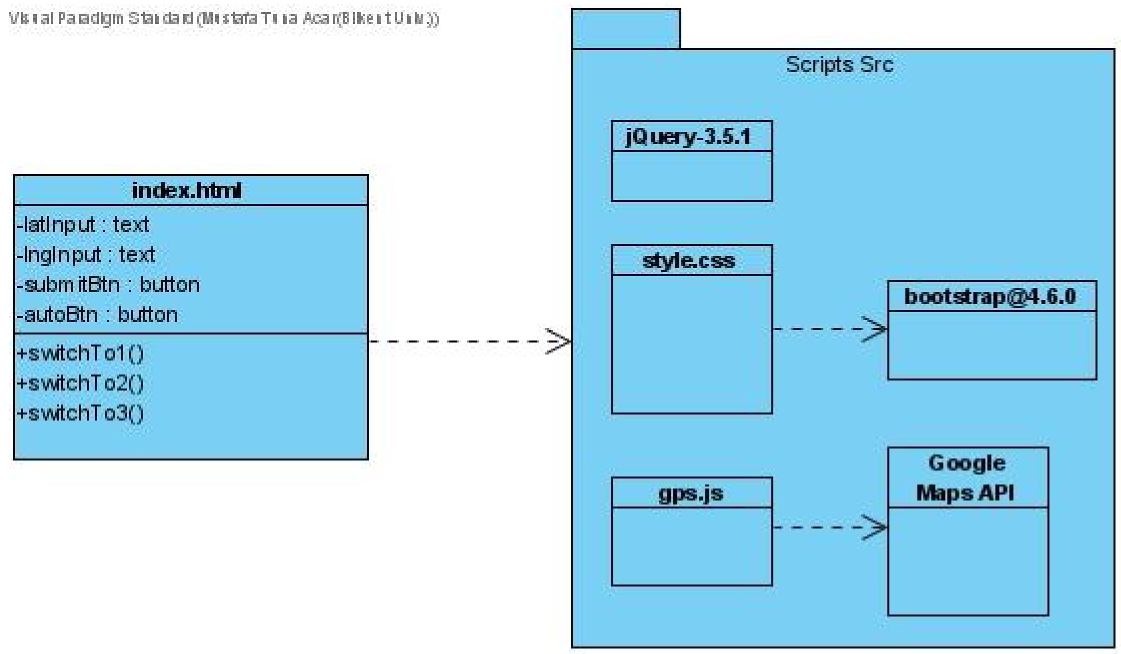


Figure 18: Class Diagram

2.2 Use Case Diagram

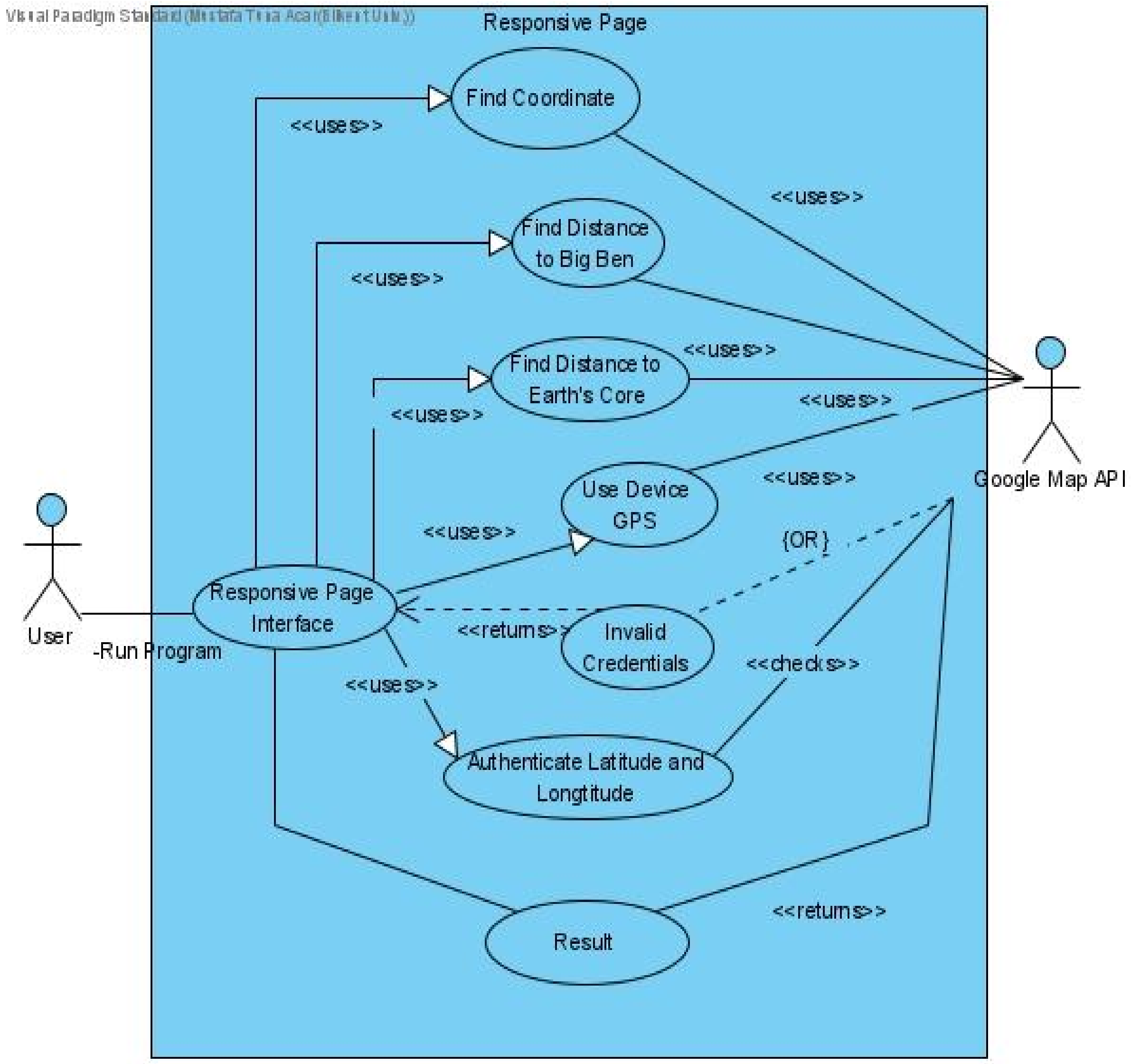


Figure 19: Use Case Diagram

2.3 Sequence Diagram

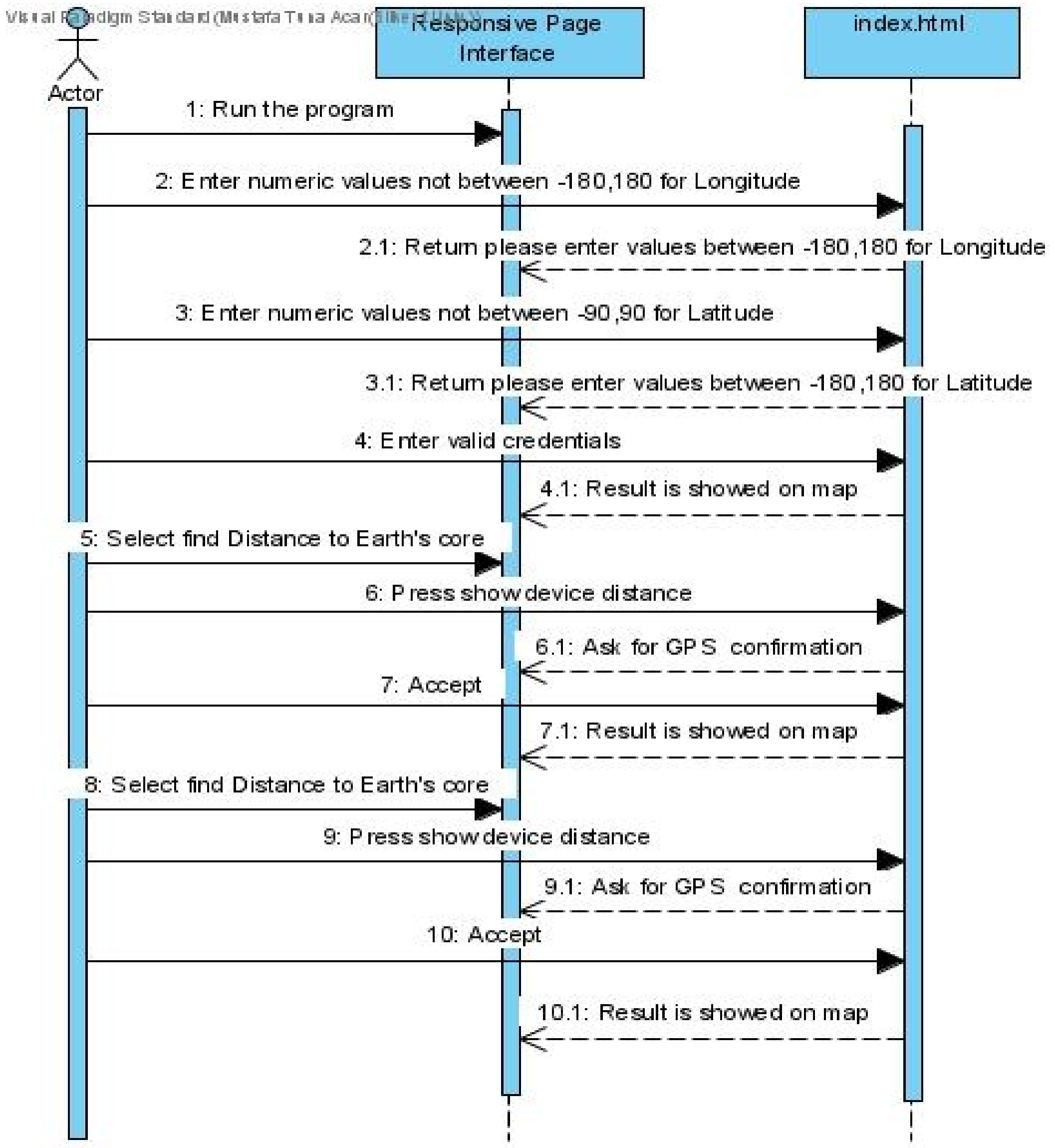

Visual Paradigm Standard (Mustafa Tuna Acar(Bilkent Univ.))
Actor
Responsive Page Interface
index.html
1: Run the program
2: Enter numeric values not between -180,180 for Longitude
2.1: Return please enter values between -180,180 for Longitude
3: Enter numeric values not between -90,90 for Latitude
3.1: Return please enter values between -180,180 for Latitude
4: Enter valid credentials
4.1: Result is showed on map
5: Select find Distance to Earth's core
6: Press show device distance
6.1: Ask for GPS confirmation
7: Accept
7.1: Result is showed on map
8: Select find Distance to Earth's core
9: Press show device distance
9.1: Ask for GPS confirmation
10: Accept
10.1: Result is showed on map

Figure 20: Sequence Diagram

3 TDD experience

4.1 Development Velocity

Normally, TTD helps improve development velocity from its ability to detect bugs quickly by easing the refactoring process.TTD is useful when there are software unit tests throughout development. However, in our case, TTD proved to be a rather time consuming process since the web application was relatively simple with one class and pre-defined requirements. It proved cumbersome to bring a testing framework layer around our main code, which already had unknown components in it, which we were not familiar with before, such as Google's Map API and Bootstrap. Although, as stated above, it would help if the project was of a longer duration and a more dynamic one.

4.2 Code Quality

TTD enabled us to design and develop tests for every functionality of the gps web application. It ensured that new code was only written when an automation test failed and hence, helped us avoid duplication of code. In addition, it allowed us to make the code simpler and clear via making the refactoring process easy, which allowed team members to produce extensible code and work on it simultaneously without any issue. It saved us time in the debugging process which was spent in better and more efficient code.