

# **TECHNICAL DOCUMENTATION FOR THE GEOQUIZ APPLICATION**

**Developed By:**

**Georgios Floros**

**University College London**

**May 2018**

## CONTENTS

<b>1. BUSINESS REQUIREMENTS.....</b>	<b>3</b>
1.1. PURPOSE OF THE APPLICATION .....	3
1.2. BASIC FUNCTIONALITY .....	3
1.3. FEATURES OF THE APPLICATION .....	4
1.4. CURRENT STATUS & UPDATE CAPABILITIES .....	5
<b>2. PRODUCT &amp; TECHNICAL REQUIREMENTS .....</b>	<b>5</b>
2.1. SYSTEM ARCHITECTURE .....	5
2.2. SUPPORTED PLATFORMS & OPERATING SYSTEM VERSIONS .....	5
2.3. SERVICES, SERVERS & DATABASES .....	5
2.4. OVERVIEW OF THE CODE .....	6
2.4.1. <i>GeoQuiz Mobile</i> .....	6
2.4.2. <i>GeoQuiz Web</i> .....	7
2.4.3. <i>Server</i> .....	7
2.5. APPLICATION PROGRAMMING INTERFACE (API) .....	8
2.6. DEBUGGING.....	8
2.6.1. <i>GeoQuiz Mobile</i> .....	9
2.6.2. <i>GeoQuiz Web</i> .....	11
2.6.3. <i>Server</i> .....	13
2.7. TESTING & IMPLEMENTATION .....	13
<b>3. DEPENDENCIES.....</b>	<b>14</b>
3.1. API DOCUMENTATION .....	14
3.1.1. <i>Geolocation API</i> .....	14
3.1.2. <i>Leaflet API</i> .....	14
3.2. INTERFACE .....	14
<b>4. CONSTRAINTS.....</b>	<b>14</b>
<b>5. REFERENCES.....</b>	<b>14</b>

## FIGURES

Figure 1: Phone-Browser Debugging: Load of PoIs.....	9
Figure 2: Phone-Browser Debugging: Loading the Menu Bar.....	9
Figure 3: Phone-Browser Debugging: Tracking user's location.....	10
Figure 4: Phone-Browser Debugging: Starting the quiz .....	10
Figure 5: Phone-Browser Debugging: Load of User Guide.....	11
Figure 6: Console-Browser Debugging: Load of the map and the menu bar.....	11
Figure 7: Console-Browser Debugging: Browsing of the form .....	12
Figure 8: Console-Browser Debugging: Load of the User Guide .....	12
Figure 9: Terminal-Server Debugging: "Get" request of PoIs .....	13

## 1. Business Requirements

### 1.1. Purpose of the application

GeoQuiz is a location-based quiz application designed to set riddles regarding University College London (UCL) historicity and is developed for the requirements of “Web & Mobile GIS – Apps & Programming” module in MSc. GIS at UCL.

### 1.2. Basic Functionality

GeoQuiz functionality is based on three (3) elemental parts:

- i. GeoQuiz Mobile: The main purpose of this application is to implement the quiz around UCL campus. It is the final product that is implemented by the end-user(client) on their mobile device. The application is composed of a basemap provided by Leaflet (<https://leafletjs.com>) on which the user's location is being tracked every 15 seconds, utilizing the Geolocation capabilities (<https://developers.google.com/maps/documentation/javascript/examples/map-geolocation>). The Points of Interest (PoI), for the purposes of this project are four, but the structure of the system supports the addition of more PoIs according to the administrator's decision. The implemented PoIs are: (i) Cruciform Building, (ii) St. Lewis building, (iii) Anatomy Building and (iv) Roberts Building. The PoIs are automatically loaded when GeoQuiz Mobile starts and appear as blue markers. As soon as the user selects the option to “Start Geolocation”, its position on the map is tracked and appears as a red marker. When the distance between their position and the PoIs is less than 100 m., the user can begin the quiz and the question related to the closest PoI will appear in the bottom of the screen. The user selects one of the four possible answers and the system returns a message showing whether the answer is correct or not. Lastly, the answers are saved and uploaded to the database, alongside with user's device id. The interface of the application is based on Google's Material Design Template (<https://material.io/guidelines/>) and the application is deployed via PhoneGap (<https://phonegap.com>) and currently supports only Android devices. The app can be downloaded from: <https://build.phonegap.com/apps/3151701/builds>
- ii. GeoQuiz Web: The main purpose of this browser-based application is to provide the administrator with tools to maintain and enrich the GeoQuiz mobile application. Specifically, the background is provided by Leaflet API, similarly with GeoQuiz Mobile and is centered around UCL Campus. The administrator is able to navigate over the map and select a UCL Building in order to add a relevant question, since by clicking on the point, its coordinates are automatically returned. Afterwards, the administrator can fill a form which contains the following information:
  - a. Question: The question that refers to the selected building.
  - b. Option A,B,C,D: Four different answers.
  - c. Solution: The correct answer of the question.
  - d. Longitude/Latitude: The coordinates of the selected point as they appear on the map.

The information of the form is stored in the database via the server and applied in GeoQuiz Mobile. The current status of the system does not support the alteration of existing questions and PoIs. The interface is based on Google's Material Design template.

Essentially, GeoQuiz Web is responsible to upload data from the web-application to the server and eventually to the GeoQuiz Mobile, facilitating regular update and enrichment of the mobile application.

The app can be accessed via: <http://developer.cege.ucl.ac.uk:31301>

### 1.3. Features of the application

GeoQuiz Mobile	GeoQuiz Web	Server
<p><u>Starting the App</u></p> <ul style="list-style-type: none"> <li>Loading PoIs from the database</li> <li>Starting Geolocation</li> </ul> <p><u>Quiz</u></p> <ul style="list-style-type: none"> <li>Checking the location compared to the PoIs</li> <li>Showing the question</li> </ul> <p><u>Response</u></p> <ul style="list-style-type: none"> <li>Selecting and submitting an option</li> <li>Message regarding result</li> <li>Upload response, solution and user's mobile id</li> </ul>	<p><u>Browser</u></p> <ul style="list-style-type: none"> <li>Navigating over the map</li> <li>Selecting PoI and acquire its coordinates</li> </ul> <p><u>Form</u></p> <ul style="list-style-type: none"> <li>Filling information about the question, answers, solution and coordinates</li> </ul> <p><u>Data submission</u></p> <ul style="list-style-type: none"> <li>Uploading the form to the database</li> </ul>	<p><u>Upload</u></p> <ul style="list-style-type: none"> <li>Data from the GeoQuiz Web to the database</li> <li>Users' answers from the GeoQuiz Mobile to the database</li> <li>PoIs from the database to GeoQuiz Mobile</li> </ul>

In order for the user to access the features of the GeoQuiz Application, a Menu-based approach is implemented. Concretely, regarding GeoQuiz Mobile, the User Guide provided with the application simplifies the process in five steps. The user interacts only with the menu button in order to start the geolocation, the quiz and the submission of the answer. As for GeoQuiz Web, the administrator interacts with the map and then selects from the menu the option to fill the form as explained in previous section and upload it to the server.

## **1.4. Current Status & Update Capabilities**

The system can currently support the implementation of additional PoIs without affecting the results, since control flows are implemented in the code. Update functions can be applied in terms of being able to modify existing questions, improvements in the design of the application and increased flexibility in GeoQuiz Mobile such as starting the quiz without user's interference.

## **2. Product & Technical Requirements**

### **2.1. System Architecture**

GeoQuiz Application is based on 3-tier architecture (Client-Server-Database) and includes the following components:

- i. GeoQuiz Mobile: The mobile application which is downloaded by the user.
- ii. GeoQuiz Web: The web browser-based application responsible to source the mobile application with additional or updated information and is managed by the administrator/developer
- iii. Server: The server code responsible to handle requests from GeoQuiz Web and Mobile application, as well as upload/download data to/from PostgreSQL database.
- iv. PostgreSQL Database (with PostGIS extension): The database responsible to store information that are necessary to implement the quiz.

### **2.2. Supported Platforms & Operating System Versions**

GeoQuiz is built via PhoneGap and currently supports Android smartphones that their system version is 6.1.2 or more. For detailed information about PhoneGap and supported operating system versions follow this link: <https://build.phonegap.com/current-support>.

### **2.3. Services, Servers & Databases**

GeoQuiz is deployed as a mobile application utilizing PhoneGap services (<https://phonegap.com>). The server that is responsible to upload data from the Web application and deploy them in the mobile application is located in UCL.

The server is responsible to process the requests made by GeoQuiz Mobile and GeoQuiz Web. The current status of the server supports the following actions:

- i. Uploading the data from GeoQuiz Web form to PostgreSQL Database.
- ii. Downloading the data created by GeoQuiz Web and upload them to GeoQuiz Mobile.
- iii. Upload the data regarding users' answers to the database.

The database responsible to store information is open-source database PostgreSQL with the PostGIS extension to enable spatial functionality. For the purposes of this application, two tables are created:

- i. Formdata: Stores information about the question, possible answers, solution and coordinates of the PoIs
- ii. Answers: Stores information about user's answer, solution and device id number.

## **2.4. Overview of the code**

### *2.4.1. GeoQuiz Mobile*

#### Interface (index.html & userform.html)

The code for GeoQuiz Mobile is presented in Appendix I. GeoQuiz Mobile is structured according to Google's Material Design template and deployed as a mobile application via PhoneGap. Geolocation services can be turned on and off from the menu, as well as starting the quiz.

#### Geolocation (location.js & map.js)

Firstly, the background map is created utilizing the Leaflet API and is centered around UCL campus. Then, in order to track user's location, the Geofence API is applied and an interval to update position every 15 seconds is set. This interval can also be removed, enabling the stop of geolocation. The user's position is inserted into the Leaflet Map.

#### Getting PoIs (getdata.js)

A request is made to the server to download the PoIs created by GeoQuiz Web. The downloaded data are parsed as GeoJSON to enhance process flexibility and inserted into the map.

#### Distance calculation & Proximity Alert (location.js)

A single method is responsible to calculate the distances between the current position of the user and the PoIs. In case the user is within the range of at least one PoI (100 meters) a question pops. In case the user is in range of more than one PoIs, the question that is attributed to the closest PoI pops. Otherwise, the user is informed that he/she is too far to play the game. The distance calculation is performed using control flow, so the addition of more PoIs can be supported by the current structure. The question is dynamically updated, using an AJAX request and utilizing the index of the PoI with the minimum distance to associate it with the corresponding question.

#### Answer Upload (location.js)

The answers are stored and uploaded to the server using a single method. The method iterates through the answers of the user to find which one is selected and compare it with the correct solution. According to the result, the user is alerted with a message regarding the validity of the answer and the information is uploaded to the server.

Similarly, the system utilizes control flow techniques, so it can process the question for added PoIs. It does not support at the moment, the implementation of additional potential answers to the question. Also, the system can be “tricked”, since it is possible to check and submit more than one answers simultaneously.

#### Marker customization (leaflet-awesome-markers.js)

In order to render distinct user’s position and the position of PoIs, user’s marker appears red to the map. For this purpose, predefined methods by Leaflet are utilized.

#### *2.4.2. GeoQuiz Web*

##### Interface (index.html & form.html)

The code for GeoQuiz Mobile is presented in Appendix II. GeoQuiz Web is structured according to Google’s Material Design template and deployed as a browser application via PhoneGap. Whenever the administrator locates a PoI, via the menu options a form appears in order to store the required information.

##### Map Navigation (map.js)

The mapping background centered around UCL campus is called utilizing Leaflet’s API and selecting the relevant tile layer. A method facilitates the retrieval of information when the administrator clicks on a point on the map.

##### Data Upload (data upload.js)

A function creates a string of the values that need to be uploaded to the database via the server. Another function makes a request to the server to “post” the data in the database and send an alert to the administrator whether the data have been successfully uploaded or not.

#### *2.4.3. Server*

The code for the server is presented in Appendix III. The server is connected with PostgreSQL database and its parameters are created.

##### PoI download

A “get” method is responsible to respond to the request of the GeoQuiz Mobile and deploy the PoI data.

##### Form upload

A “post” method is responsible to facilitate the request of uploading the data generated in GeoQuiz Web to PostgreSQL database.

##### Answer upload

A “post” method is responsible to facilitate the request of uploading users’ answers, the solution and the device id of GeoQuiz Mobile to PostgreSQL database.

## 2.5. Application Programming Interface (API)

GeoQuiz utilizes two (2) different APIs:

- i. [Geolocation API](#) : This API is responsible to track and update the location of the user in the GeoQuiz Mobile application.
- ii. [Leaflet API \(v. 1.3.0\)](#): This API is responsible to call the mapping background that is utilized in GeoQuiz application.

## 2.6. Debugging

The debugging methods implemented for the development of GeoQuiz application are:

- i. Try/Catch: mostly used when making a request to the server, to process a potential error message.
- ii. Utilizing console.log, known as “Brute Force Debugging”: a particularly useful command to check intermediate results of the code and understand where the process works smoothly and where it fails. The generated code has plenty of console log messages since they are very useful to monitor and understand the code in case of an upgrade.
- iii. Utilizing browser debugging tools: In combination with console log, when running the code, the console option in the browsers provides great assistance to make the code work functionally.
- iv. Server-side debugging: a similar concept with console log placed in multiple positions, with the difference that the message appears on the terminal window. It is helpful to understand which parts work properly and where the code has an error.
- v. Applying the code in the browser console: particularly useful to understand for instance the elements and properties of an object.
- vi. Another tool utilized for debugging, mostly for typing errors is JS Hint (<http://jshint.com>).
- vii. Connecting the phone with the browser as demonstrated below.



### 2.6.1. GeoQuiz Mobile

Figure 1 presents the automatic load of POIs.

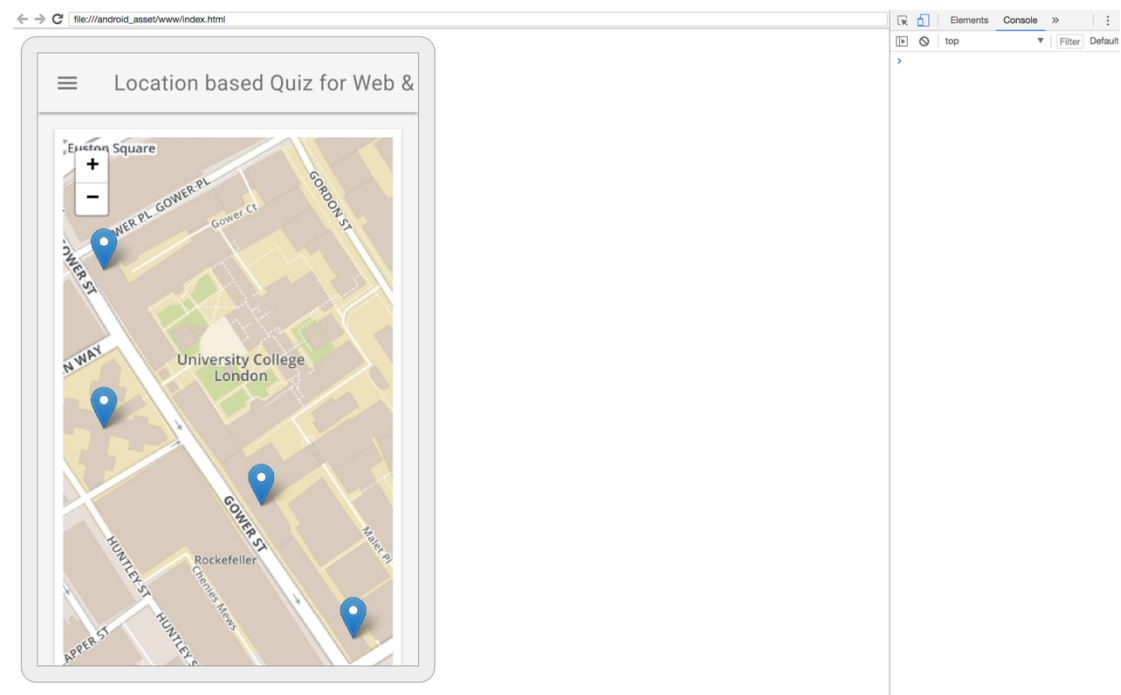


Figure 1: Phone-Browser Debugging: Load of POIs

Figure 2 presents the load of the menu bar.

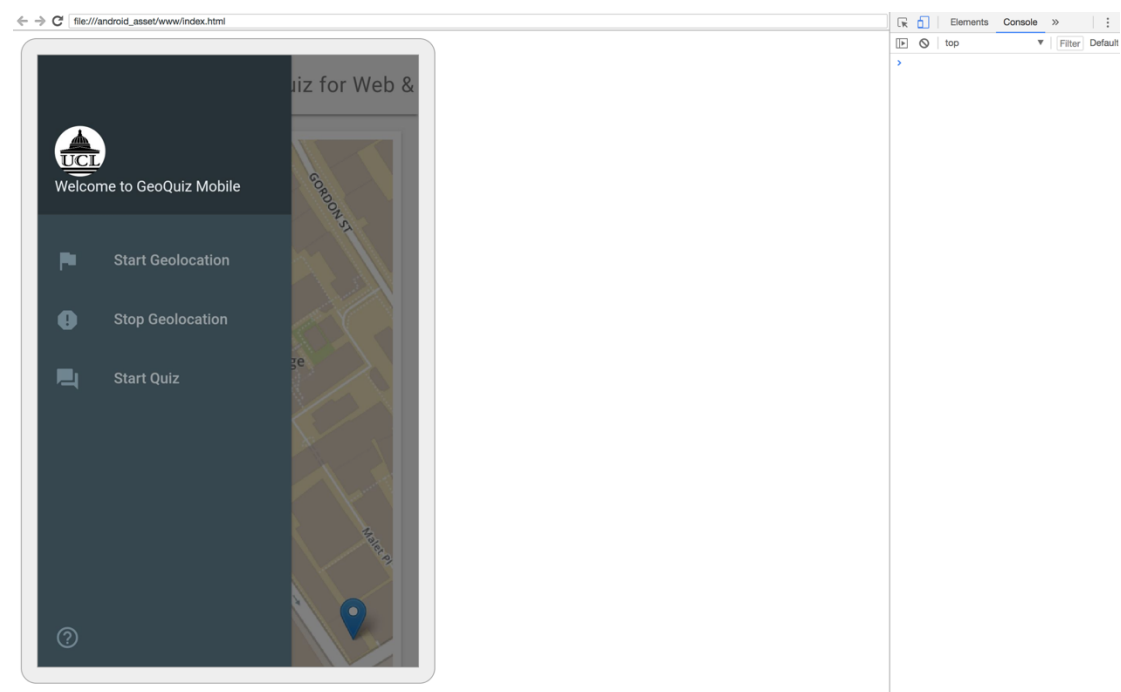


Figure 2: Phone-Browser Debugging: Loading the Menu Bar

Figure 3 presents the geolocation of the user's position. Console log debugging is also demonstrated on the upper right side of the browser.

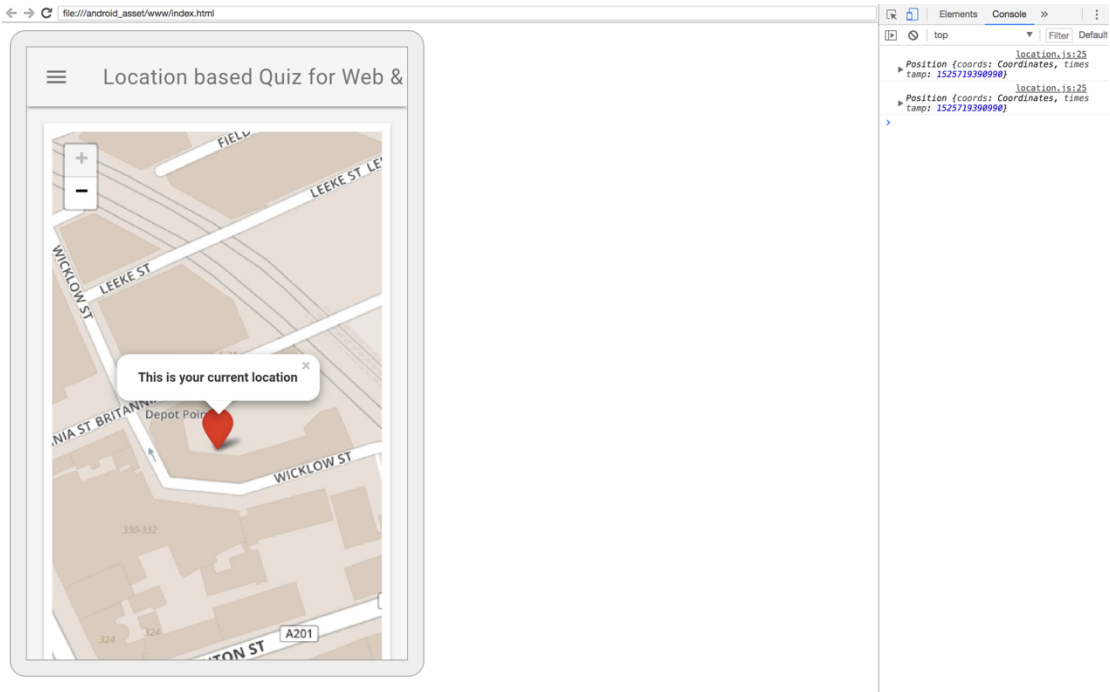


Figure 3: Phone-Browser Debugging: Tracking user's location

Figure 4 presents the beginning of the quiz. The minimum distance is more than 100 meters; therefore, an alert shows on the phone informing the user that is too far from UCL campus.

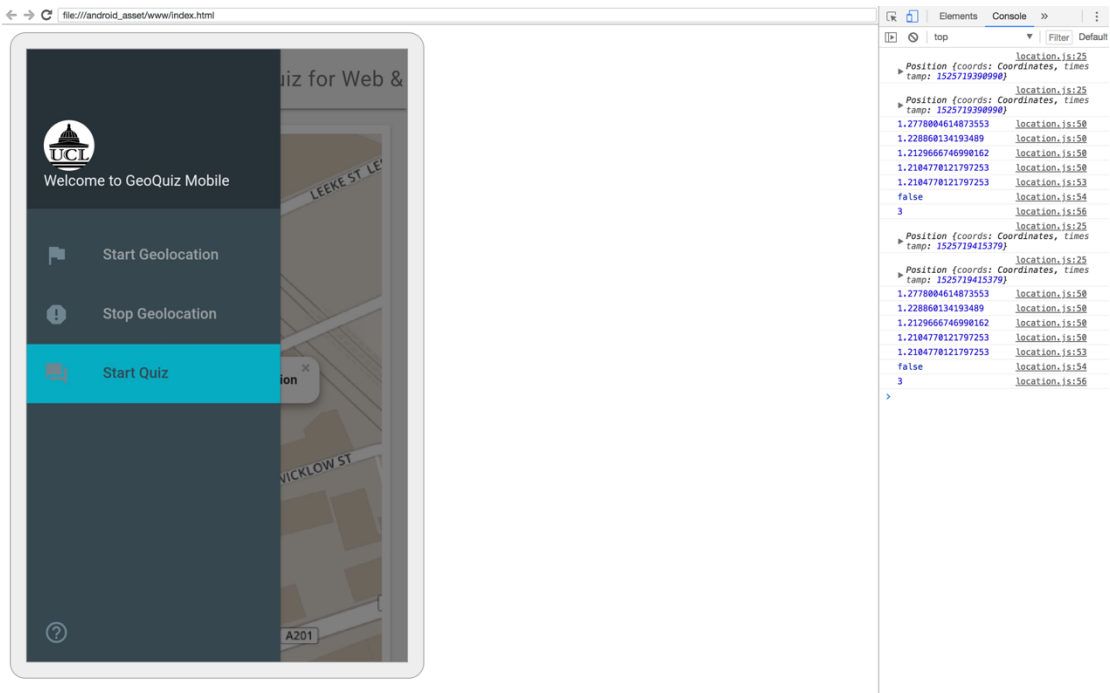


Figure 4: Phone-Browser Debugging: Starting the quiz

Figure 5 presents the incorporated User Guide of GeoQuiz Mobile.

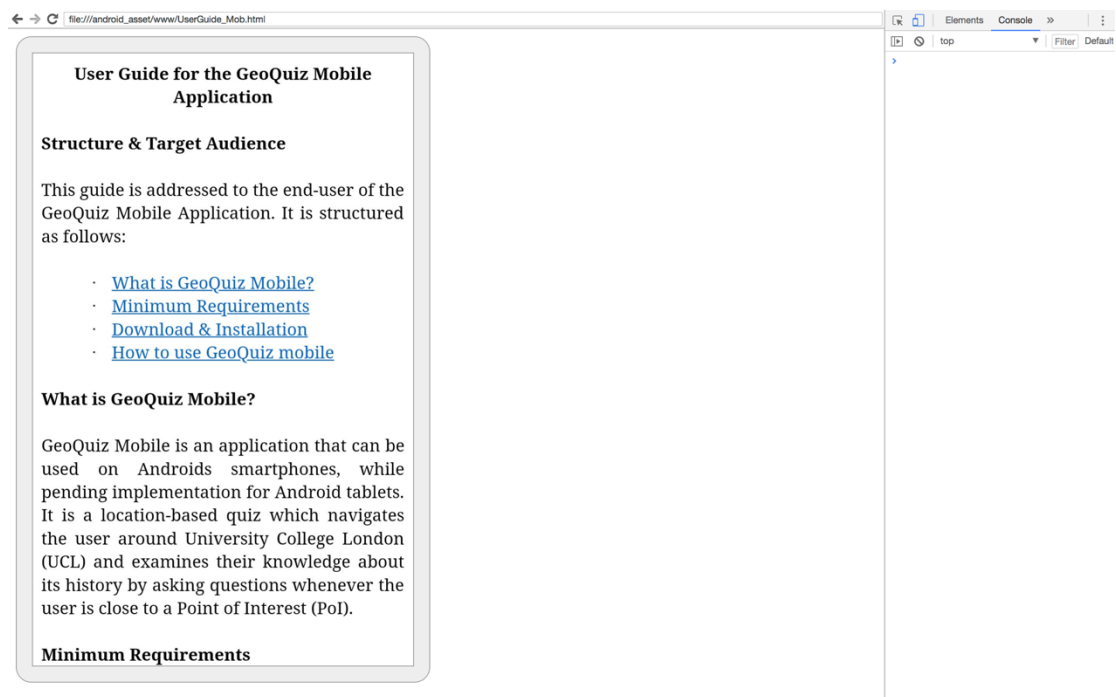


Figure 5: Phone-Browser Debugging: Load of User Guide

## 2.6.2. *GeoQuiz Web*

Figure 6 presents the load of the map and the menu bar.

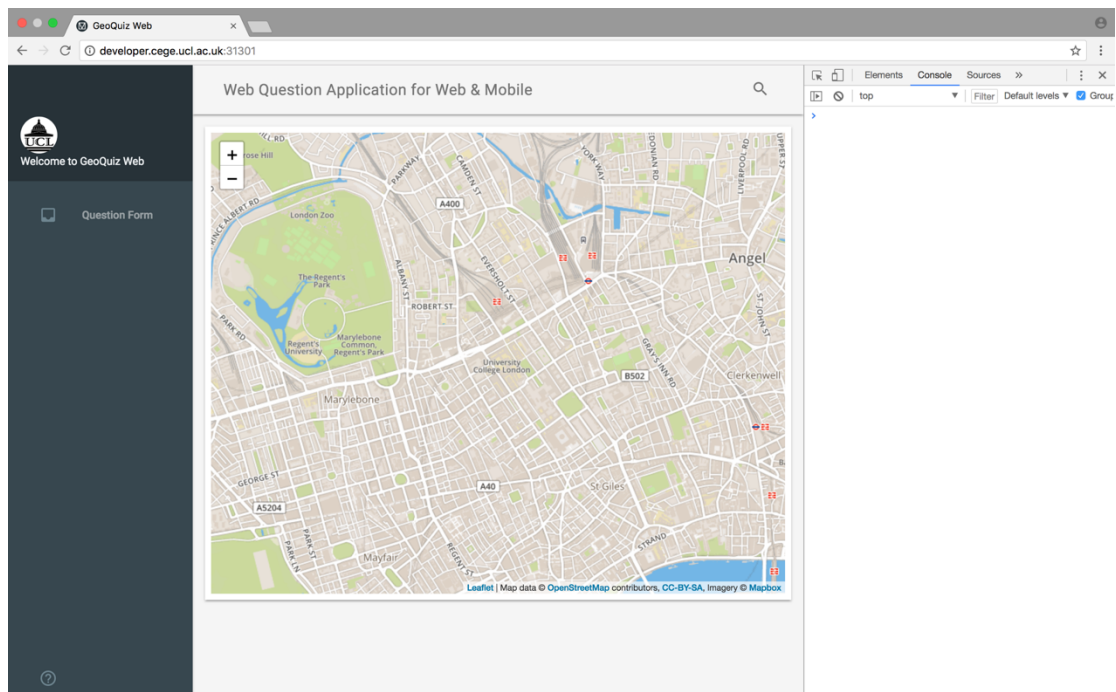


Figure 6: Console-Browser Debugging: Load of the map and the menu bar

Figure 7 presents the browsing of the form.

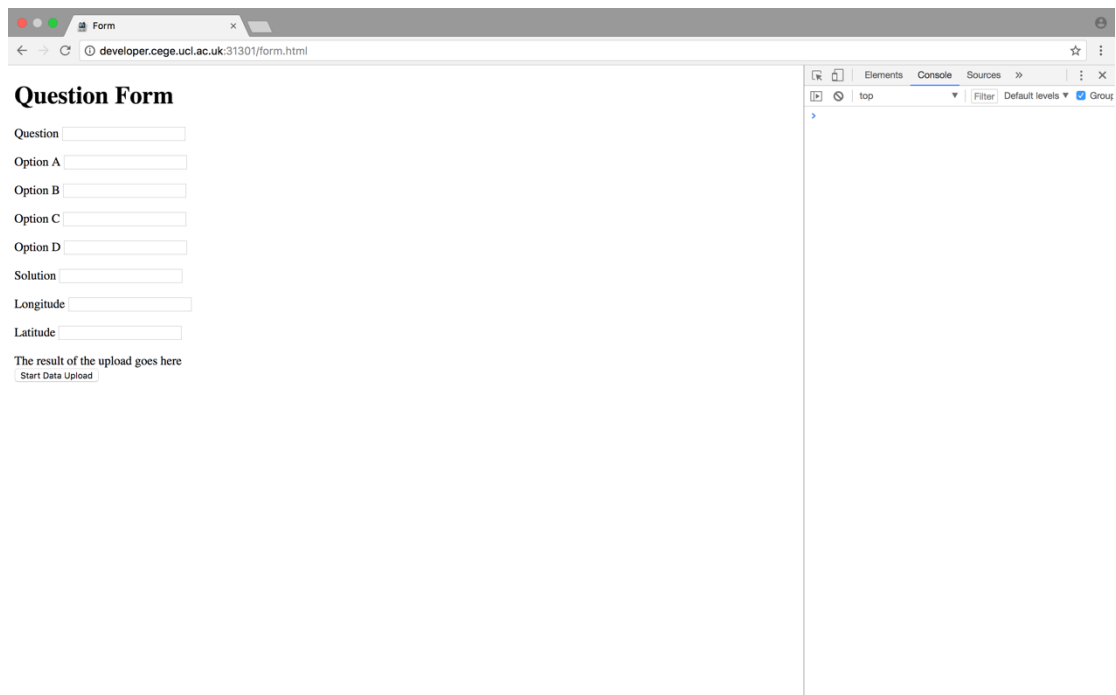


Figure 7: Console-Browser Debugging: Browsing of the form

Figure 8 presents the incorporation of the User Guide for the GeoQuizWeb.

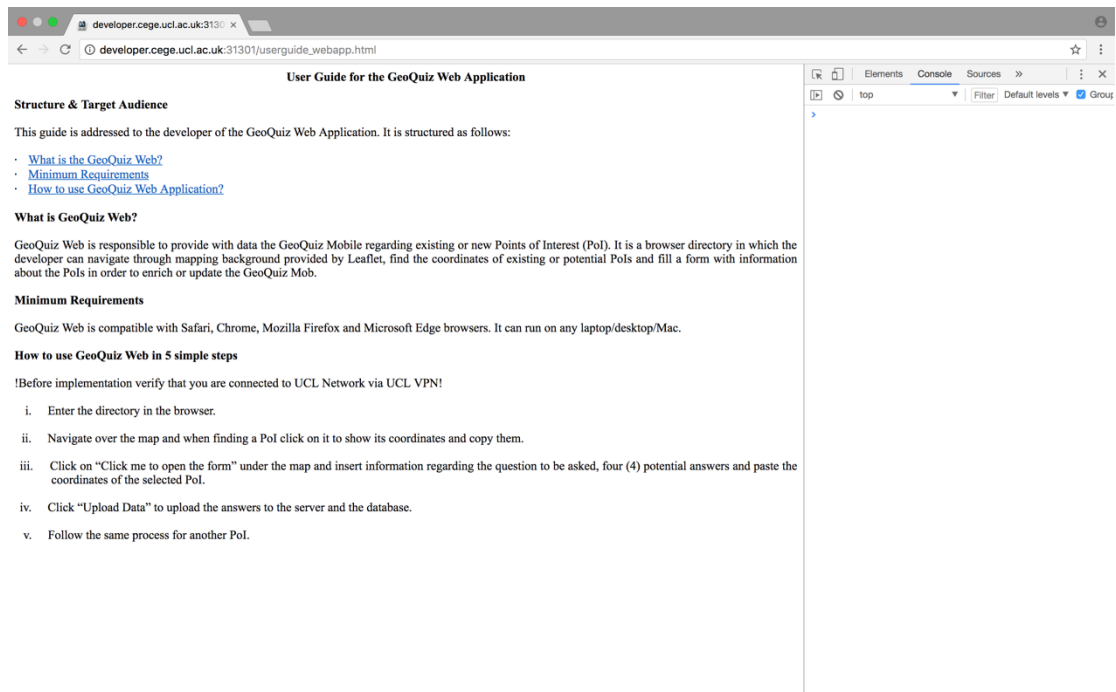
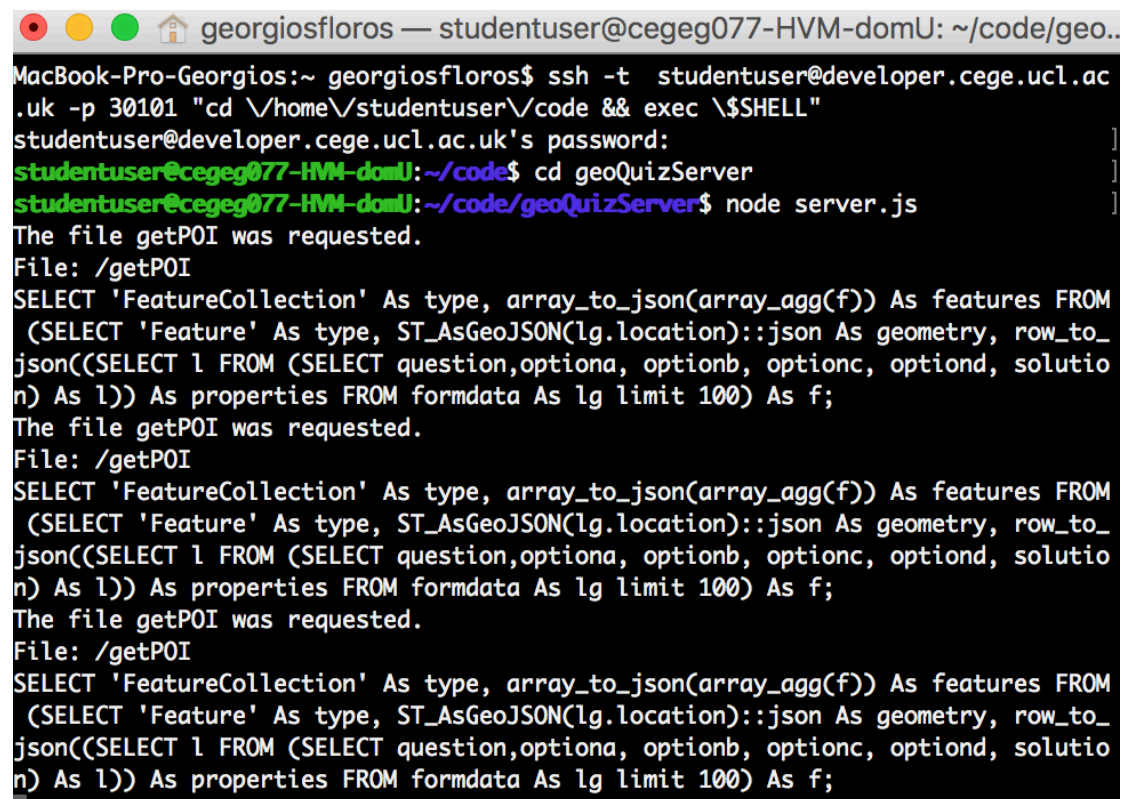


Figure 8: Console-Browser Debugging: Load of the User Guide

### 2.6.3. Server

Figure 9 presents an example of a load of PoIs from the server side.



```
georgiosfloros — studentuser@cegeg077-HVM-domU: ~/code/geo..
MacBook-Pro-Georgios:~ georgiosfloros$ ssh -t studentuser@developer.cege.ucl.ac
.uk -p 30101 "cd \home\studentuser\code && exec \SHELL"
studentuser@developer.cege.ucl.ac.uk's password:
studentuser@cegeg077-HVM-domU:~/code$ cd geoQuizServer
studentuser@cegeg077-HVM-domU:~/code/geoQuizServer$ node server.js
The file getPOI was requested.
File: /getPOI
SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As features FROM
 (SELECT 'Feature' As type, ST_AsGeoJSON(lg.location)::json As geometry, row_to_
json((SELECT l FROM (SELECT question,optiona, optionb, optionc, optiond, solutio
n) As l)) As properties FROM formdata As lg limit 100) As f;
The file getPOI was requested.
File: /getPOI
SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As features FROM
 (SELECT 'Feature' As type, ST_AsGeoJSON(lg.location)::json As geometry, row_to_
json((SELECT l FROM (SELECT question,optiona, optionb, optionc, optiond, solutio
n) As l)) As properties FROM formdata As lg limit 100) As f;
The file getPOI was requested.
File: /getPOI
SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As features FROM
 (SELECT 'Feature' As type, ST_AsGeoJSON(lg.location)::json As geometry, row_to_
json((SELECT l FROM (SELECT question,optiona, optionb, optionc, optiond, solutio
n) As l)) As properties FROM formdata As lg limit 100) As f;
```

Figure 9: Terminal-Server Debugging: "Get" request of PoIs

## 2.7. Testing & Implementation

In terms of testing, GeoQuiz Mobile is tested on all four PoIs by physically walking around and activating the application, returning the expected results. Similarly, GeoQuiz is tested by creating those PoIs using the relevant form. Lastly, server is tested by the capability of GeoQuiz Mobile to receive the selected PoIs and the by checking the population of PostgreSQL database with PoI data and user data.

### 3. Dependencies

#### 3.1. API Documentation

##### 3.1.1. Geolocation API:

Method	Description	Properties
<b>getCurrentPosition()</b>	This method returns user's position	Coords.latitude, coords.longitude
Source: <a href="https://www.w3schools.com/html/html5_geolocation.asp">https://www.w3schools.com/html/html5_geolocation.asp</a>		

##### 3.1.2. Leaflet API:

Method	Description	Properties
<b>L.map</b>	This method creates a map and manipulates it.	setView
<b>L.marker</b>	This method creates the point that represents user's location	toGeoJSON, bindPopup
<b>L.tileLayer</b>	This method loads and displays tile layers in the map	maxZoom, attribution
Source: <a href="https://leafletjs.com/reference-1.3.0.html">https://leafletjs.com/reference-1.3.0.html</a>		

#### 3.2. Interface

GeoQuiz application utilizes the Material Design template (<https://getmdl.io/templates/index.html>).

### 4. Constraints

GeoQuiz Application is developed to meet the requirements of “Web & Mobile GIS – Apps & Programming” module as part of the MSc. GIS at UCL. Therefore, there are limitations in terms of available time and deadlines to develop the app as well as of the basic functionality that is required for the app to work properly.

### 5. References

Google Maps Platform. URL: <https://developers.google.com/maps/documentation/javascript/examples/map-geolocation> . Accessed on 07/05/2018.

JS Hint. URL: <http://jshint.com> . Accessed on 07/05/2018.

Leaflet. URL: <https://leafletjs.com> . Accessed on 07/05/2018.

Material Design. URL: <https://getmdl.io/templates/index.html> . Accessed on 07/05/2018.

PhoneGap. URL: <https://phonegap.com> . Accessed on 07/05/2018.