

1.UCL Question System App	1
1.1 Description	1
1.2 Target Audience	1
1.3 Product Goals	1
1.4 Tech Details	1
1.5 System Architecture	1
1.7 Featureset	2
2.UCL Quiz App	5
2.1 Description	5
2.2 Target Audience	5
2.3 Product Goals	5
2.4 Tech Details	5
2.5 System Architecture	6
2.7 Featureset	6
2.6.1 Menu-based Options	7
2.6.2 Map-based Options	10
3. Server Side Code - Node JS and REST	12

Technical Guide

1.UCL Question System App

1.1 Description

The [UCL Question System app](#) is a browsed-based web application that works in conjunction with the UCL Quiz app. The main operation of the application is data collection and the population of the back-end database that is used by the [UCL Quiz app](#). In particular, it enables to a user to record POIs in which information has been attached that is employed by the quiz app.

1.2 Target Audience

The web application can be used by anyone of any age. The user does not need to have any experience in order to be able to use the application.

1.3 Product Goals

The goal of the current app is to enhance the capabilities of ULC Quiz app, so that a user can easily populate the POI dataset that is present in the database. As a result, the dataset is constantly expanded and no more procedures are needed to adopt the changes that are happening in the dataset.

1.4 Tech Details

The web application is compatible with the following browsers:

- 1) Google Chrome Version 66.0.3359.139 (64-bit)
- 2) Google Chrome Version 48.0.2564.109 (64-bit)
- 3) Mozilla Firefox Quantum Version 59.0.2 (64-bit)
- 4) Mozilla Firefox Quantum Version 45.2.0 (64-bit)
- 5) Internet Explorer 11 Version 11.0.9600.18920 (64-bit)

A user can access the application in the following link:

<http://developer.cege.ucl.ac.uk:31277/>

1.5 System Architecture

While a variety of models are available about how the communication between a client and a server is constructed, the **three-tier architecture** was chosen and implemented for the purpose of our assignment. In these terms, the created app, which comprises the **presentation tier** of our architecture, displays information related to the requirements and specifications of the project, and any user-interface interaction is processed by the **application tier** and send it on the **data tier**. At this layer, the server and the database are maintained.

1.6 Constraints

The map display has been set such only that the UCL campus is shown. Therefore, the app will not work normally if it is launched to an area outside of UCL campus.

1.7 Featureset

The main functionality of the web application is based on the [Leaflet](#) library, using a variety of functionalities that the library is compatible with. [Plugins](#) are also employed to improve the capabilities of the application.

The UCL Question System app is equipped with a variety of commands that are described below:

Search Command

A commands that enables a user to search a building based on its corresponded **question**. The result is indicated with a marker on the map.

This command has been built using the [leaflet search](#) plugin.

Clean Command

Markers that may remain from a search process are cleaned from the map. This command uses the [removeFrom](#) function to remove the markers from the map.

Full Zoom Command

The map zoom changes so that a full display of the data is given. This command uses the [fitBounds](#), and [getBounds](#) methods to set the bounding boxes of the map display and the buildings dataset equal.

Geolocation Command

Enables the geolocation of the user and marks his/her location on a map. The accuracy of the geolocation is shown with a buffer around the marked position of the user. Based on the user desire, the geolocation command can be activated or deactivated.

The functionality of the geolocation command is based on the [locate](#), [stopLocate](#) methods, and [locationfound](#) and [locationerror](#) events. The geolocation has been adjusted using the desired location options.

Labels - Pop up windows

While both images below show the same content, their foundations are based on different principles and, therefore, they are categorised in two distinct label types. The left image is a **class-based** label, whereas the right image is a **popup-based** label.

1) Class-based:

The construction of this label type is based on leaflet [classes](#) extension. In particular, a leaflet class is extended and a **div** element is created using the [L.DomUtil](#) utility function of Leaflet to communicate with the [DOM](#) tree. At this point, an update method is set on the div element, as well as mouse events that permit its update when the mouse moves.

2) Popup-based:

This type of label is created using the [L.popup](#) class that is by default provided by leaflet. At this case, a pop up is created and then is attached to the [geojson](#) layer. The [pointToLayer](#) method is employed to append the pop up on the geojson layer.



Fig. 1: The two types of labels that are employed for the UCL Question System App.

Clusters

The UCL POIs are shown as clusters using the [marker cluster](#) plugin. For different zoom levels, the map display is adapted so that a certain portion of buildings is shown.

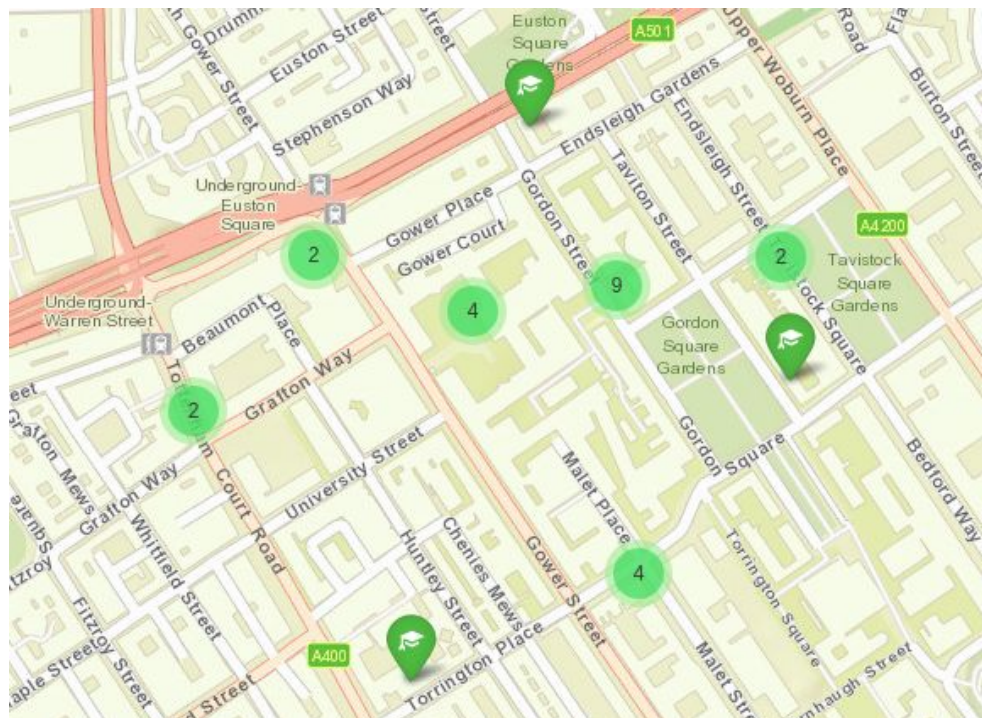
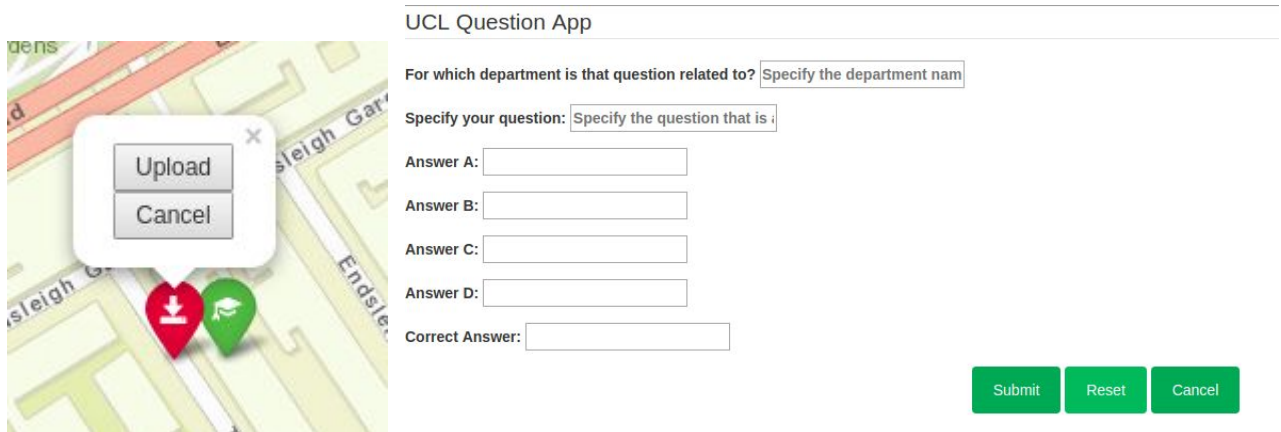


Fig. 2: The figure shows how the map display is adapted in order to show a certain portion of the POIs. The non-shown POIs are indicated as clusters.

Form

Initially, with a double click on the map, a pop up window with two possible options is displayed (Fig. 3). This has been created by binding a pop up on a [marker](#) that is constructed when the user [click](#) on the map. Selecting the Upload option, a form is loaded through an AJAX GET request, which uses the [XMLHttpRequest](#) function. Using Javascript and interacting with the [DOM](#) tree of the document, the HTML elements of the form are gained, and therefore, the user's information is POSTed back on the server, after the form is submitted.



UCL Question App

For which department is that question related to?

Specify your question:

Answer A:

Answer B:

Answer C:

Answer D:

Correct Answer:

Fig. 3: The process of form loading is shown in this figure.

2.UCL Quiz App

2.1 Description

The **UCL Quiz app** is a location-based smartphone/tablet app, which employs the user location and prompts him/her on a quiz test. The questions, which have a historical related content, are based on the user's proximity with respect to the buildings of UCL.

2.2 Target Audience

The web application can be used by anyone of any age. The user does not need to have any experience in order to be able to use the application.

2.3 Product Goals

The fact that the app could be employed for educational purposes and be used by the students of UCL is the main reason that led us on the development of the application.

2.4 Tech Details

The UCL quiz app can be used on any Android device that supports an android model above **2.3 (API Level 10) (Gingerbread)**, and is implemented based on the **Cordova Phonegap build**¹. As a rule of thumb, Android versions become unsupported by Cordova as they dip below 5% of the Android users. A better intuition about the model's compatibility can be taken by the following Google's android device dashboard: <https://developer.android.com/about/dashboards/>

A user can download the application from the following link:

<https://build.phonegap.com/apps/3150421/builds>

How to install UCL Quiz mobile app?

The app can be downloaded from this link (<https://build.phonegap.com/apps/3145369/builds>) using *two* different ways:

1) Download the **APK (Android application Package)** file and transfer it to you mobile device using a *usb cable*. Before you start the installation, you need to check the *unknown sources*² options on your mobile, which can be found on the security settings tab (Fig. 4).

This needs to be done as the mobile app is an unknown source app for your mobile device, and the user's privileges are needed to accept the installation. Then, the UCL Quiz app can be normally installed on your device and an app icon is created on your home screen.

1. <http://docs.phonegap.com/phonegap-build/overview/>

2. In most mobile devices the unknown sources options can be found on the Settings > Security tab. The path may differ for different mobile devices.

3. APK : Android Package (APK) is the package file format used by the Android operating system for distribution and installation of mobile apps and middleware.

2) The application can be downloaded using a **QR app reader** (https://play.google.com/store/search?q=qr%20reader&hl=en_GB). Install one of the recommended QR apps, open it, and place the camera of your device over the QR code that is given for the UCL Quiz app. Then, download the app and install it. Similarly to step 1, the unknown sources option needs to be checked in order to permit the installation. After the installation, an app icon is created on your home screen.

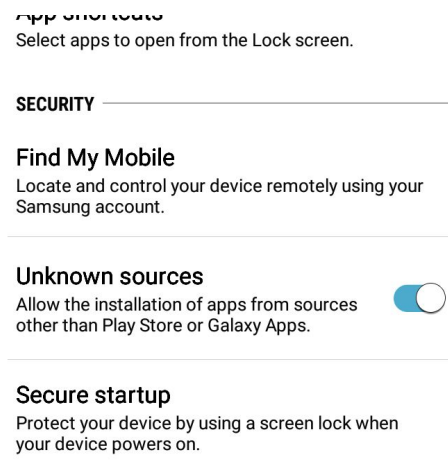


Fig 4. The unknown sources option needs to be enabled so that the mobile app could be normally installed.

2.5 System Architecture

While a variety of models are available about how the communication between a client and a server is constructed, the **three-tier architecture** was chosen and implemented for the purpose of our assignment. In these terms, the created app, which comprises the **presentation tier** of our architecture, displays information related to the requirements and specifications of the project, and any user-interface interaction is processed by the **application tier** and send it on the **data tier**. At this layer, the server and the database are maintained.

2.6 Constraints

The map display has been set such as the UCL campus is only shown. Therefore, the app will not work correctly if it is launched to an area outer of UCL campus.

2.7 Featureset

Besides the main purpose of the mobile app to inform a user and prompts him/her on a quiz challenge, the app's deployment was attained so that a user can interact with the facilities of the campus of UCL. A variety of commands and operations are supported, which are described in that section.

The commands that are supported are categorised in two different categories, the **menu-based** and the **map-based** commands.

The main functionality of the web application is based on the [Leaflet](#) library, using a variety of functionalities that the library is compatible with. [Plugins](#) are also employed to improve the

capabilities of the application, and a [bootstrap](#) template is used to make the app more visually attractive.

The UCL Question System app is equipped with a variety of commands that are described below:

2.6.1 Menu-based Options

A menu bar is available that the user might use and changes some of the functionalities of the app (Fig. 5). The menu bar options are described below:

Basemaps:

The app supports the **Street**, **Imagery**, and **Topo** basemaps that are provided by *Esri ArcGIS*. Both, the user location and the buildings of UCL are formulated in a such way that changes on the map layers do not interrupt the normal operation of the datasets. The **Street** map layer is by default loaded when the mobile app launches. The layer that is activated is shown with a different font weight (Fig. 6). **CSS** and **Javascript** are used to provide these functionalities, while the [leaflet providers](#) plugin is employed to get access in the ArcGIS layers of Esri. The map's methods [addLayer](#) and [clearLayers](#) of the [L.featureGroup](#) class are mainly used.

Layers:

The **buildings of UCL** are activated or deactivated with that option (Fig. 7). This option can be extended to accommodate more datasets and functionalities. **Javascript**, **CSS** and the **Leaflet** library are used to import this functionality. The map's methods [addLayer](#) and [clearLayers](#) of the [L.featureGroup](#) class are mainly used.

Quiz Time:

The app supports *two* different types of quiz tests, the **Time and Proximity Quiz** (Fig. 8), which are based on different implementations. These options are responsible to start a quiz challenge or even to stop. The **content** of the label of each option determines whether the quiz is activated.

Time Quiz: A question over a constant time interval is shown to the user that needs to answer. Therefore, for **constant time intervals**, the location of the user is compared with the location of all the buildings of UCL, and a question is constructed related to the closest building. When the difference between the latest asked question and the upcoming question exceeds the specified value of the interval, the question pops up on the user. The user is able to choose *different time intervals* picking a different option from the [Quiz Level](#) option.

Proximity Quiz: The location of the user and the buildings of UCL are compared. Based on a specified [Quiz Level](#) option, the building that has the closest distance, its distance is compared with a threshold value, and if **that distance is less than the threshold value**, a question pops up related to the closest building. The process is repeatedly executed every thirty seconds. *The proximity quiz is the default option that is loaded when a user launches the app.*

Time vs Proximity Quiz

Both are location based options, meaning that the questions that are asked are constructed based on the closest building with respect to the user location. However, their implementation deviates with respect to **when** the question is asked. The **time** option performs questions based on a **temporal criterion** (specifying a temporal value that defines the difference between the latest asked and the upcoming question), while the **proximity** option pops up question according to a **distance criterion** (if the distance of the user is less than a specified threshold value).

In both cases, the functionality of the geolocation is based on the [locate](#), [stopLocate](#) methods, and [locationfound](#) and [locationerror](#) events. The geolocation has been adjusted using the desired location options.

[L.popup](#), [L.Marker](#) and [L.circle](#) classes are employed in order to append this functionality. A variety of methods that are contained on the classes' functionalities are also used, as well as the built-in [setInterval](#) function.

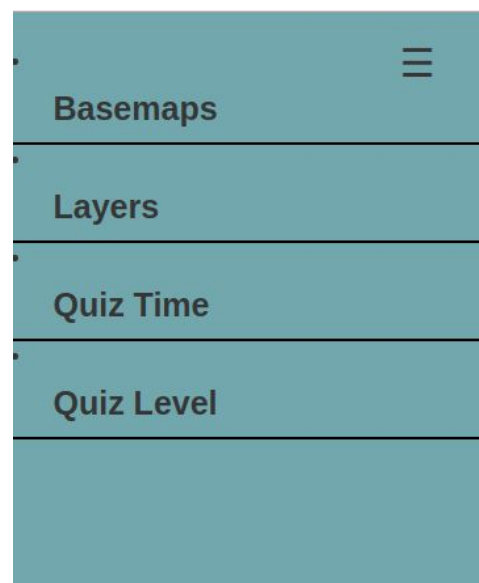


Fig. 5: The app is equipped with a **menu bar** that provides a variety of options, which change the operation of the app.

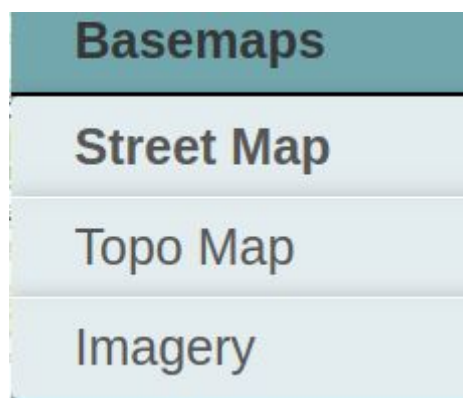


Fig. 6: The **Basemaps** option is capable to use different basemaps using the ArcGIS map layers. The options of Street Map, Topo Map and Imagery are given.

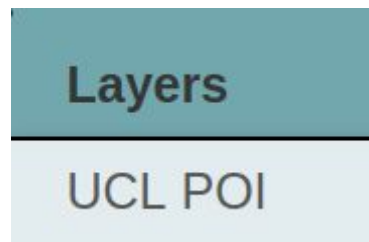


Fig. 7: By simply clicking on the **Layers** label and the UCL POI option, the buildings of UCL are added or removed from the display.

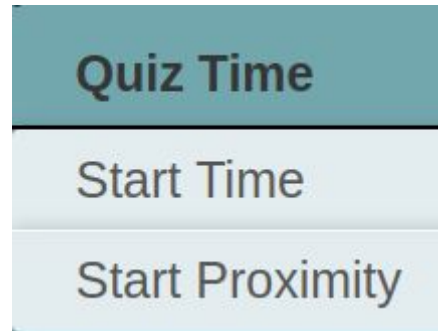


Fig. 8: The **Quiz Time** option provides different options about how the quiz challenge occurs. While both options of the quiz are location-based, in a way that a question is asked according to the user location, each option has a different constraint that will permit a question to be displayed.

Quiz Level:

Three different options are available (Fig. 9) for the time and proximity options that each one depends on its implementation details. A label of **High**, **Medium** and **Easy** is given for each option that represents the level of difficulty of the quiz. The high, medium and easy option for the time quiz corresponds on a **60**, **120** and **300** interval counted in *seconds*, respectively. For example, if a user picks the option of high difficulty, a question will pop up to him/her every one minute.

On the other hand, the proximity quiz is equipped with distance-related options. In that case, a high, medium and easy difficulty option corresponds to a **50**, **100** and **500 metres** distance, respectively. In conjunction with the geolocation service that operates every thirty seconds, a question is shown to the user if his/her distance with respect to a UCL building is less than the specified threshold distance. For example, if the user chooses the option of high difficulty, then a question is only displayed on the user if his/her distance is less than 50 metres from the closest UCL building. The process is continuously executed every thirty seconds. The selected option is shown with a different font weight.

Javascript and **CSS** are employed to accommodate these functionalities.

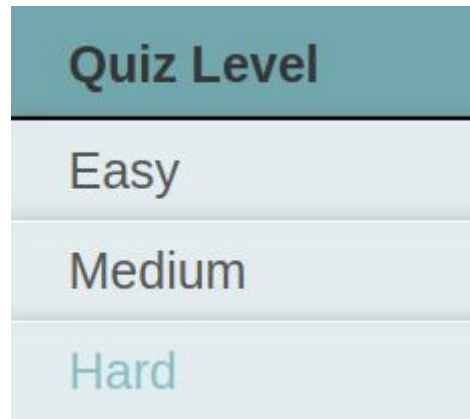


Fig. 9: The **Quiz level** determines different levels of difficulties for the time and proximity quiz option.

2.6.2 Map-based Options

Search Command

A commands that enables on a user to search a building based on its corresponded **question**. The result is indicated with a marker on the map.

This command has been built using the [leaflet search](#) plugin.

Clean Command

Markers that may remain from a search process are cleaned from the map. This command uses the [removeFrom](#) function to remove the markers from the map.

Full Zoom Command

The map zoom changes so that a full display of the data is given. This command uses the [fitBounds](#), and [getBounds](#) methods to set the bounding boxes of the map display and the buildings dataset equally.

Labels - Pop up windows

While both images below show the same content, their foundations are based on different principles and, therefore, they are categorised in two distinct label types. The left image is a **class-based** label, whereas the right image is a **popup-based** label (Fig. 10).

1) Class-based:

The construction of this label type is based on leaflet [classes](#) extension. In particular, a leaflet class is extended and a **div** element is created using the [L.DomUtil](#) utility function of Leaflet to communicate with the [DOM](#) tree. At this point, an update method is set on the div element as well as mouse events that permit its update when the mouse moves.

2) Popup-based:

This type of label is created using the [L.popup](#) class that is by default provided by leaflet. At this case, a pop up is created and then is attached to the [geojson](#) layer. The [pointToLayer](#) method is employed to append the pop up on the geojson layer.



Fig. 10: The two types of labels that are employed for the UCL Question System App.

Clusters

The UCL POIs are shown as clusters using the [marker cluster](#) plugin. For different zoom levels, the map display is adapted so that a certain portion of buildings is shown (Fig. 11).

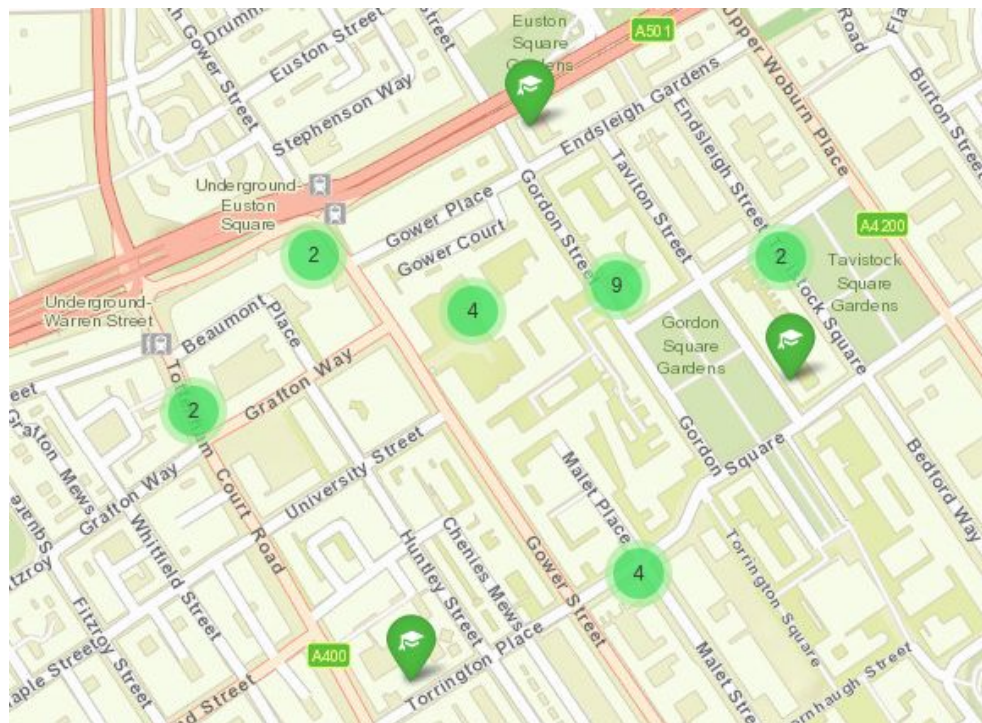


Fig. 11: The figure shows how the map display is adapted in order to show a certain portion of the POIs. The non-shown POIs are indicated as clusters.

Form/Multiple Choice Question

Based on the selected quiz options of the user, the quiz is executed with different ways. However, for all options a multiple-choice question is asked [Fig 12 (A)]. This has been achieved using the [XMLHttpRequest](#) method, which loads the question through an AJAX GET request. While the form is loaded, the initial content of the form is adjusted so that a question related to the closest building is shown. Using [Javascript](#) and [DOM](#), the [HTML](#) elements of the document are gained, and their content is replaced by the content of the question. When the form is filled by the user, the user's input is processed and checked using boolean operators.

The equality between the user input and the correct answer that is contained in the buildings' dataset is therefore checked. A pop up, which is created using the [L.popup](#) class, returns on the user, indicating whether his/her answer is correct, and a piece of information is sent back to the database through an AJAX POST request [Fig 12 (B)]. More specifically, the data contains information about the user device model, a boolean operator that shows whether the user uses a mobile device, the question, the user input, the question's correct answer, and the location of the user.

When the department of political science is founded?

Answer A :1900

☐

Answer B :1923

☐

Answer C :2004

☐

Answer D :2005

☐

Submit

Reset



Fig. 12: (A) The multiple choice question that the user is prompted to answer. (B) A pop up is binded on the user location, indicating whether his/her answer is correct or not.

Initially, with a double click on the map, a pop up window with two possible options is displayed (Fig. 12). This has been created binding a pop up on a [marker](#) that is constructed when the user [clicks](#) on the map. Selecting the Upload option, a form is loaded through an AJAX GET request, which uses the [XMLHttpRequest](#) function. Using Javascript and interacting with the [DOM](#) tree of the document, the HTML elements of the form are gained, and therefore, the user's information is POSTed back on the server, after the form is submitted.

3. Server Side Code - Node JS and REST

The server code is deployed using the [Node JS](#) library. The [REST](#) architecture is deployed to establish an HTTP protocol and perform requests on the server. The open-source [postgreSQL](#) database with [PostGIS](#) extension is also used to create the database. The server as well as the database are located in the UCL campus.