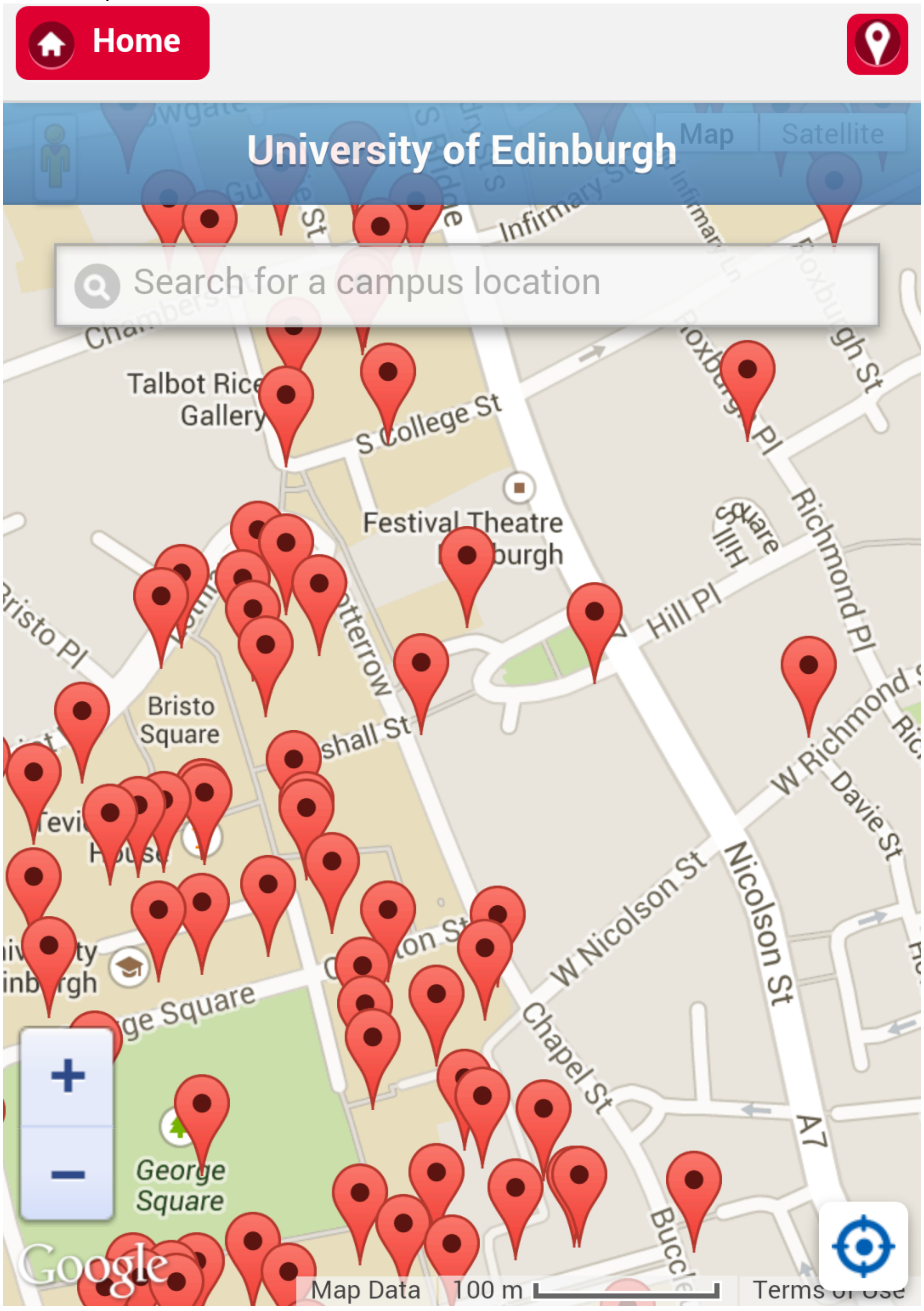
Maps Portlet Functionality:

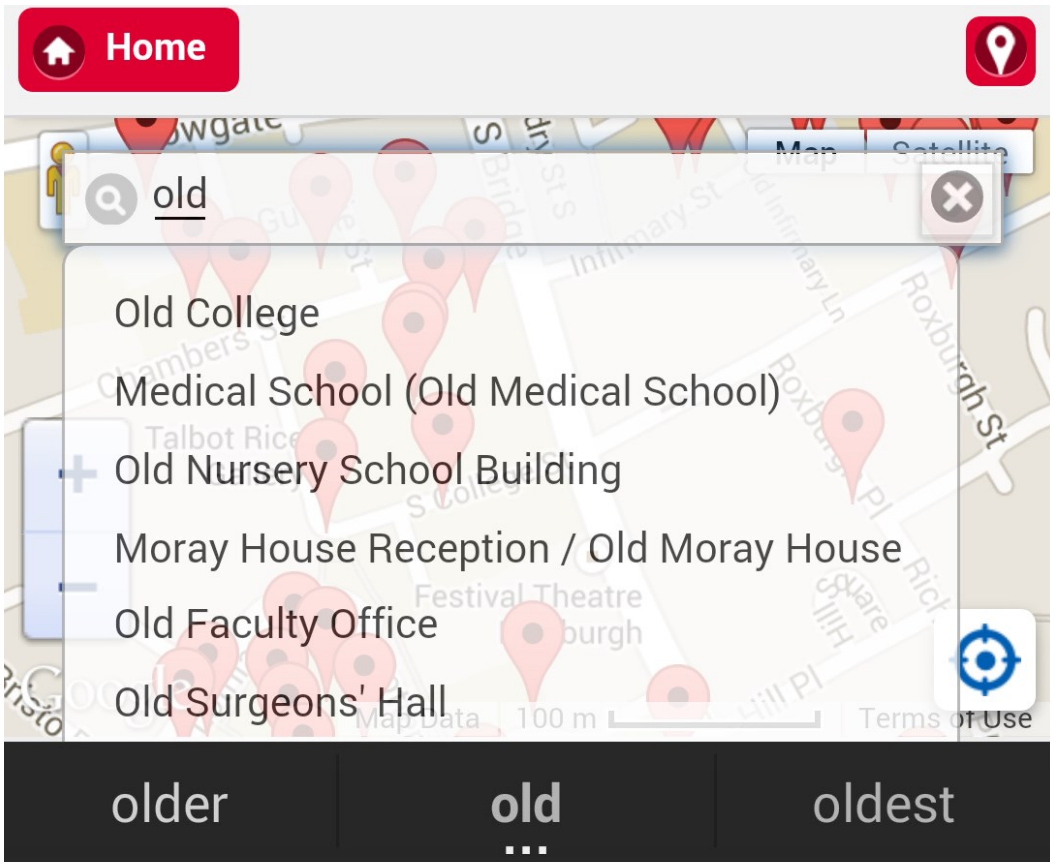
On choosing the maps portlet, the user gets a map centred on the central campus with all pins in place on that map and the search box is visible by default. The search box contains the text 'Search for a Campus Location'.



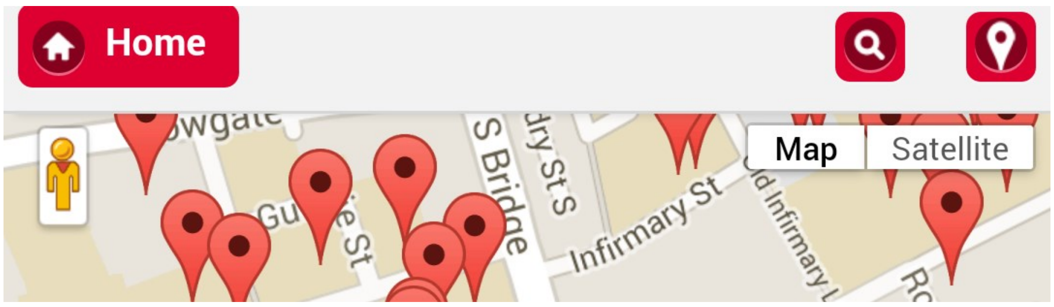
The map should be full screen on any given device – e.g. windows phone, android, iOS.
The blue bar at the top is the 'info-bar', after 4 seconds it fades out and the search input moves upwards to take its place.

Buttons in the header:
Up to **three** buttons appear in the header, 'Home', 'search' and 'menu'. Buttons will be displayed in one line in the header. There is also a 'locate me' button visible in the bottom right of the screen. All buttons except 'home' are just icons but they do have descriptive text in their hyperlinks for screen readers to use.

Search Input:
There is a search input at the top of the page – users can search for a location from the search input which uses autocomplete and a minimum of three characters. The searches are performed **only against the 'name'** of a location (see the JSON feed in section 8). When a user chooses a location from the autocomplete list it is entered into the search input**.** As soon as the user chooses a value it is entered into the search input and the search occurs. If the user attempts to enter a search term not in the list or an empty term and then presses (e.g. 'go' in Android) to perform the search then a modal box appears explaining how to use the search.

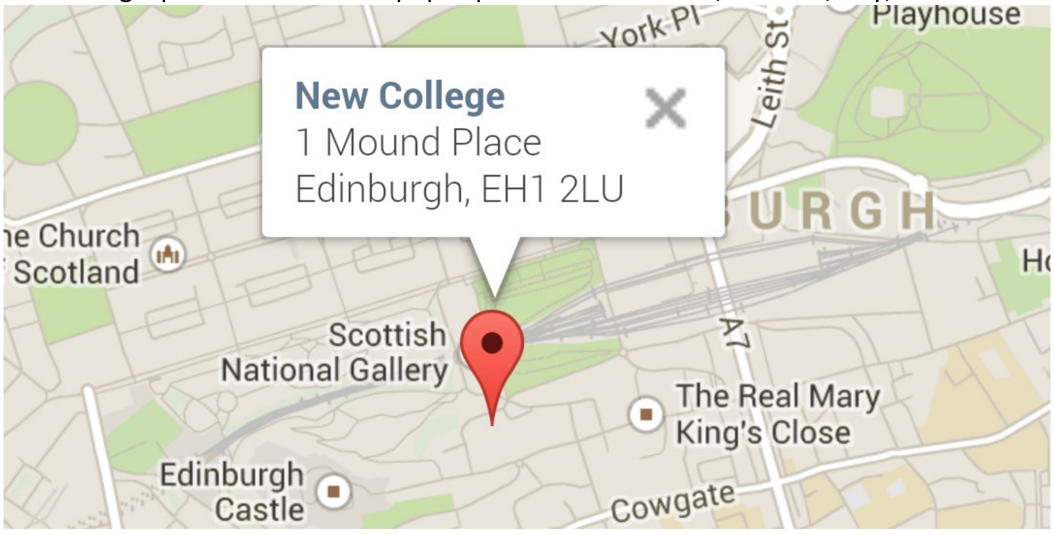

Home
old
Old College
Medical School (Old Medical School)
Old Nursery School Building
Moray House Reception / Old Moray House
Old Faculty Office
Old Surgeons' Hall
older
old
oldest

With the search input visible, if the user taps the screen the search input loses focus and is removed from the screen. If the user just moves the map (touch and hold) then the search input merely loses focus but stays visible.
Once the search input has lost focus the user must tap somewhere inside it to return its focus. When the search input is hidden, the search button appears at the top of the screen.



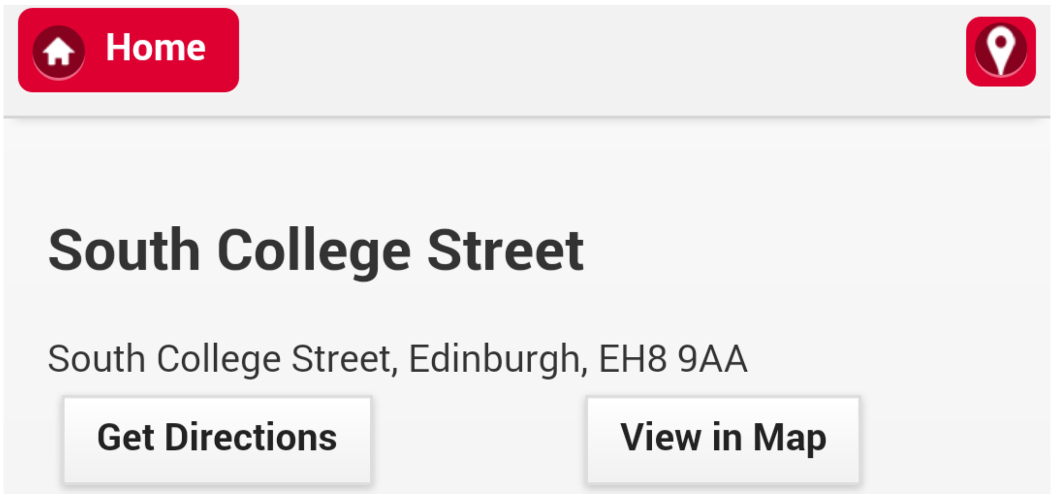
The search input has a raised z-index compared to the underlying map. This is significant as show/hide of the search box from this raised position does not cause any reflow of the map underneath. This is less jarring for the user. It also prevents open info-windows (see below) from closing. Note that when the user taps on a pin in order to make it display its info-window (see below) , that tap will also cause the search input to be hidden.
When a search is actually performed, the map **centres on the result pin and all other pins are hidden**. Once a search has been performed, the search input is also hidden. Clicking on the search button returns the search input to the screen.

Info window:
On clicking a pin an info window pops up and contains: Title, Address, City, and Post Code as:


New College
1 Mound Place
Edinburgh, EH1 2LU

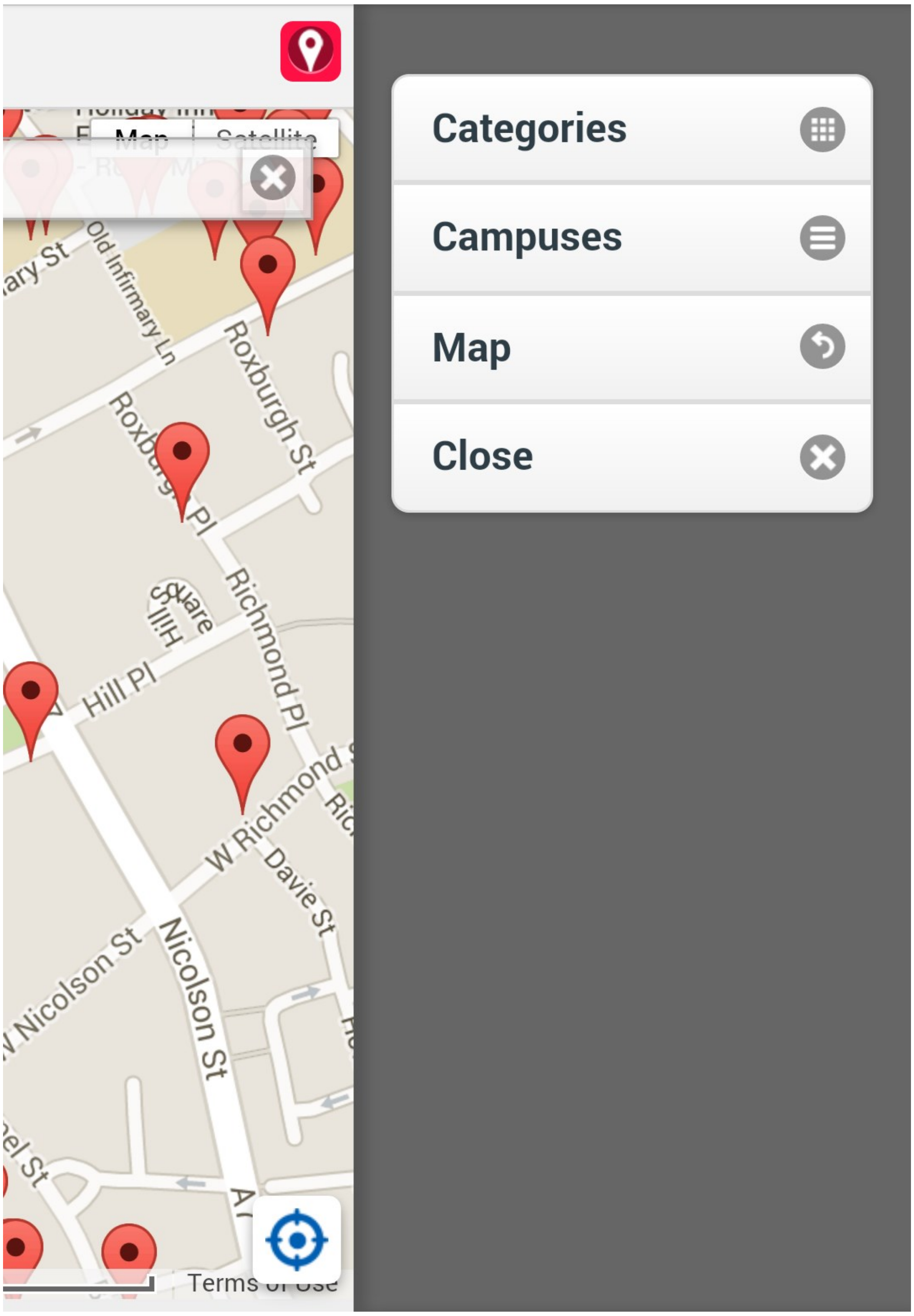
Title should be formatted as a hyperlink which takes the user to the details page. **Clicking the 'X' or tapping the screen elsewhere dismisses the info-window.** Info-windows do not remain after the screen has moved on elsewhere.

Details Page:
This is currently lacking as our database feed lacks any details about the locations. 'Details' is an optional field in the xml schema (and hence in the JSON feed) and if added to the feed will be displayed on this page.

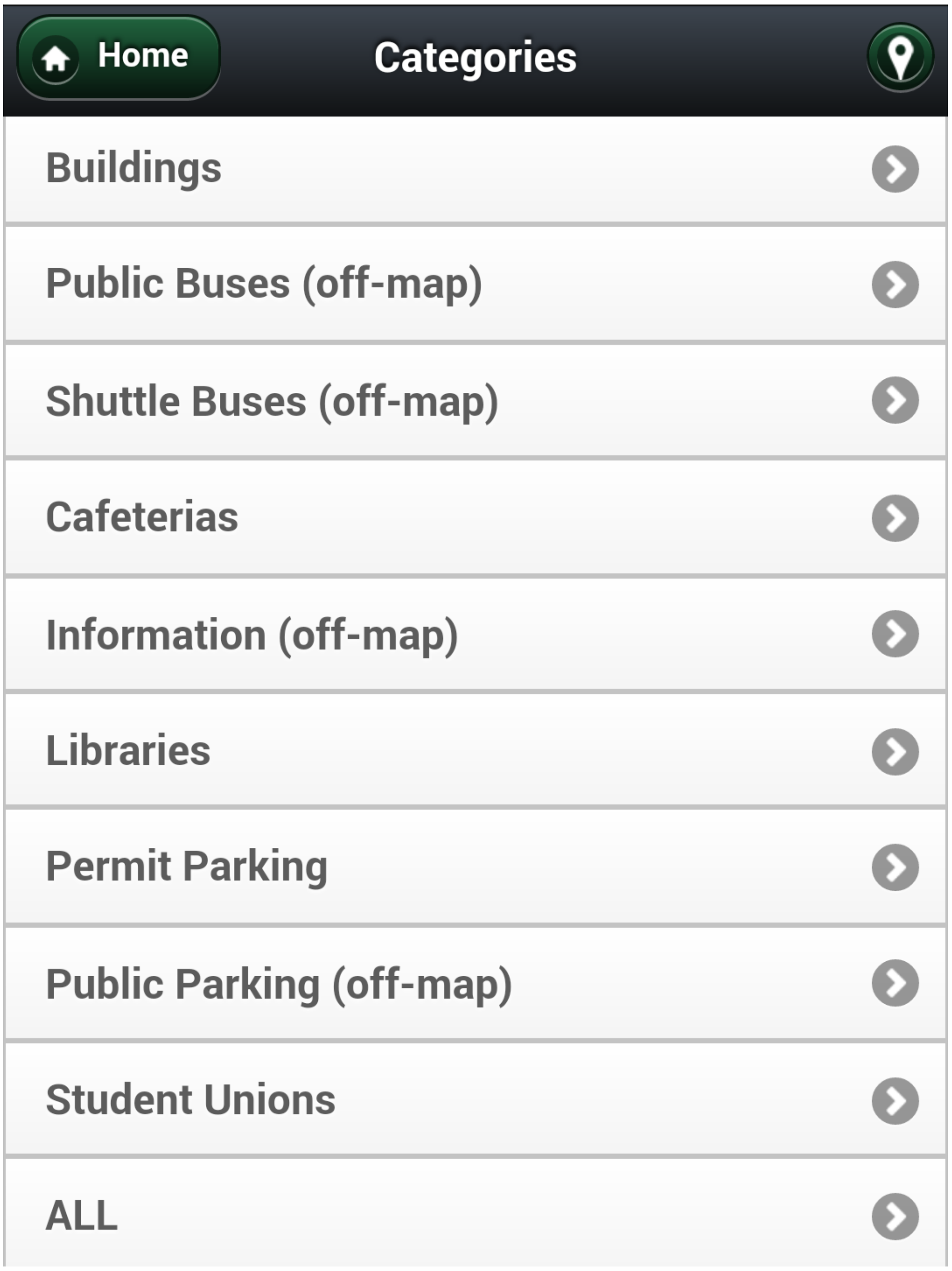


All links on the details page should be buttons. It displays the same information as the info window with the addition of a directions button which will take the user to google maps launched as an external application **in a new browser tab** and a 'view in map' button which takes the user back to the map (The info-window will be closed if the user returns to the map).

'Menu button' and additional panel:
Clicking on ***menu button*** causes a panel to move in from the right, with a raised z-index so that it covers the map rather than causing a reflow. This panel contains buttons that allow the user to choose 'categories', 'campuses', 'map' (goes straight to map from anywhere in the menu hierarchy) or 'close' (closes the menu).



Initially the ‘Campus’ is not set and the category is set to the ‘ALL’ value.
Categories:
Clicking on this displays a list of categories (defined by the campus maps PHP database).
Any category that has no visible pins on the map has ‘(off-map)’ added to its label on the menu.


Home
Categories

Buildings

Public Buses (off-map)

Shuttle Buses (off-map)

Cafeterias

Information (off-map)

Libraries

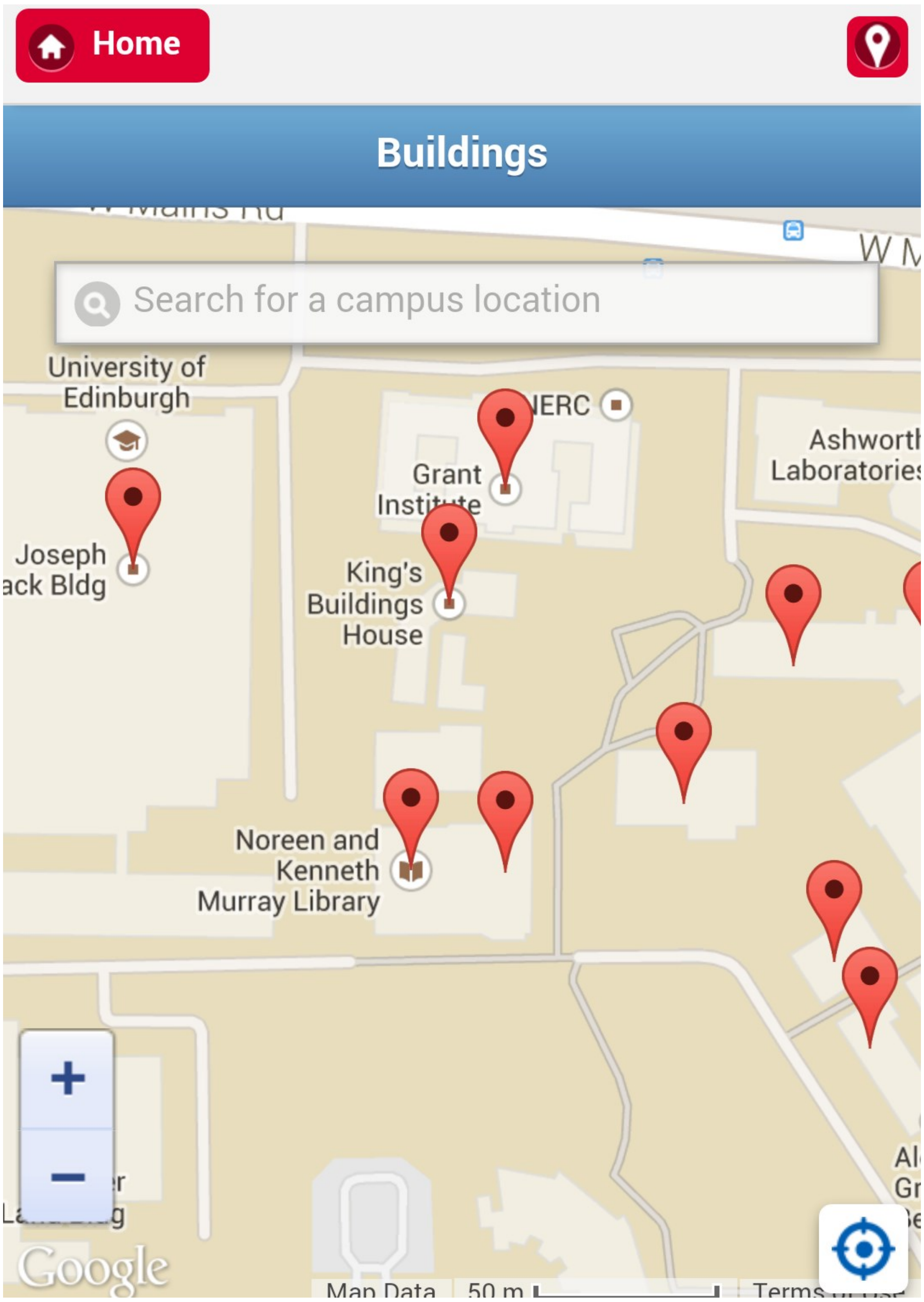
Permit Parking

Public Parking (off-map)

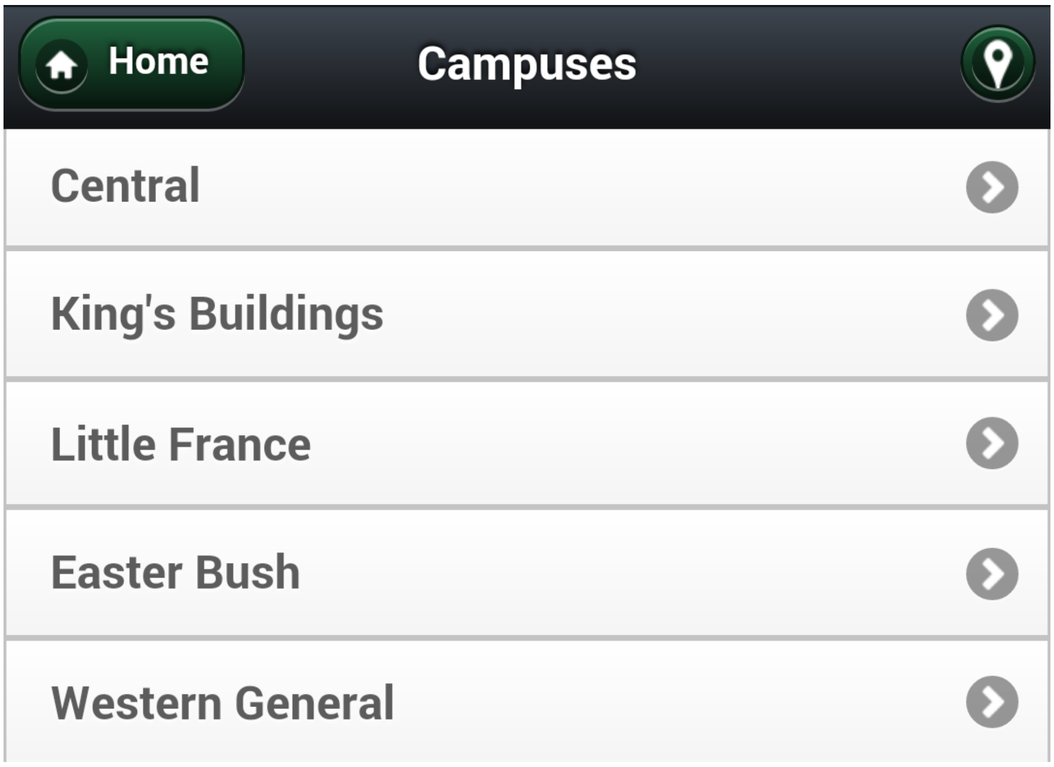
Student Unions

ALL

Clicking on a category returns to the map view with **all pins of that category visible**. **The map does not move from its previous location.** If the user chooses to zoom or move the map other pins of that category will be **visible in all campuses**. A new map header is displayed containing the name of the category, with the search bar below the new header. As previously, after a few seconds the new map header fades and the search bar moves up to take its place. Users can choose the 'ALL' category which will display all locations.

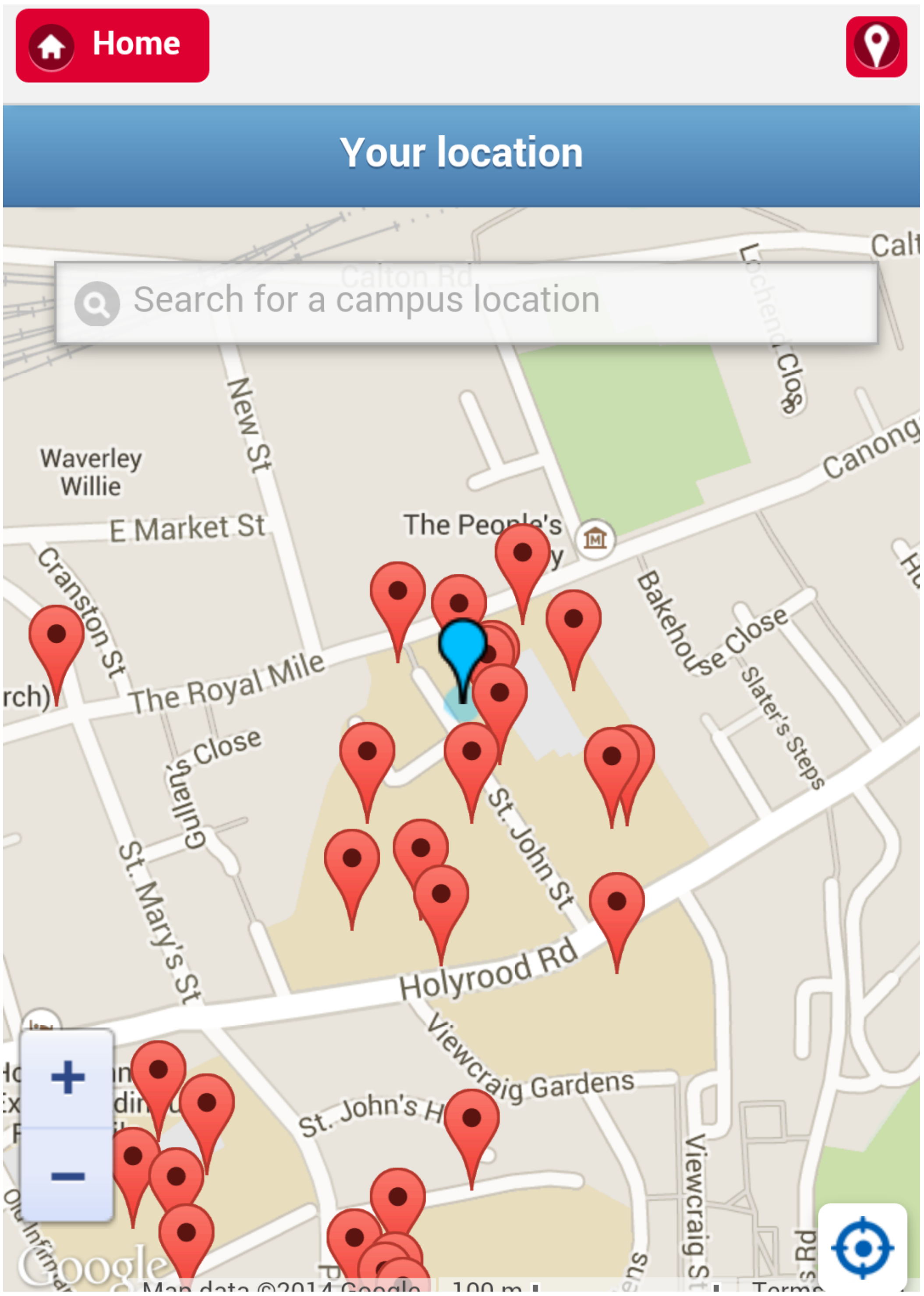


Campuses:
Clicking on campuses from the menu panel displays a list of UoE Campuses:



Choosing a campus recenters the map on the pins of the given campus**. All categories of pin are displayed**. That means the previous category is now reset to the 'ALL' category. A new map header appears displaying the title of the chosen campus. The search box is below the new header and as before the header will fade after a few seconds and the search will move up.

Locate me:
Clicking this results in a geo-location call using the google maps API. The user's location is shown with a light blue circle around it and a marker pin styles to be smaller than the map locations, light blue and with a z-index placing it above all other pins. The map centres on the user's location and the display of pins is unchanged. Note – unlike choosing a campus geo-location does not reset the category to 'ALL'. **As a result you can geolocate to a map position with no location pins displayed.** The map displays an info-header with 'your location' as the text and it fades like all the other info-headers.



Server side Caching:

Once retrieved from the remote feed, map data is cached. In fact it's just held in memory in an instance variable. Every request from clients of the Maps Portlet involves two distinct phases. In the first a request is made to the uPortal server by a client for 'map options'. These are preference values used in the map display. On the server this initial request actually reads the full map data in order to establish what the default location of the map should be. During this phase, a request is sent to the remote feed with an 'if-modified-since' in the http header. The lastmodified value sent is also just an instance variable, read from the responses made by the remote feed. If the lastmodified date has not changed at the remote end, then it should return an HttpStatus code of NOT_MODIFIED and the server side code continues to use the Maps JSON held in memory. If instead the remote feed returns any other response code then another request is made to it for the Map JSON. The returned JSON is stored in the Maps data instance variable along with a new value of lastmodified .

ETagging/Local Storage by clients:
The only time Maps Portlet clients make a request for new Map JSON data over http is when the browser page is refreshed. As a result no Etagging is possible as every refresh reloads all of the JavaScript. I therefore modified the server side response to indicate that caching should be used with a time to live of 24 hours. This ought to result in some browsers caching the JSON response and using that after a browser request and also some servers between the origin server and the client caching the response, which can greatly speed things up even if the browser does not cache the response. To make development work easier, I created maven profiles in the Maps Project for each of our development environments. Local and beta will not cache, dev will cache for 1 hour, test and prod will cache for 24 hours.

I also set up localStorage for devices capable of using it. These devices will cache the JavaScript data structure that is built from the server's JSON response. As a result they will not make any subsequent web requests to the server after their first request (actually they still must make calls to google but around 66% of the web traffic is now removed). The web request ordering in the Maps portlet is as follows:
1) When the map is opened or refreshed a request is sent to the server for the map's preference values
2) The JavaScript on the page makes an AJAX call for the Map's JSON and builds a data structure.
Using localStorage circumvents step 2. Step 1 can be used to expire the locally stored JSON. The JavaScript code checks the values in the Map's preferences sent over in step 1. If the value for mapdatarefreshdate (which is just the lastmodified date that we obtain from the remote feed) has changed from one held in local storage then the JavaScript will make an Ajax call to get fresh map JSON, build a new data structure and replace the one in local storage with this new one. Note that due to the caching above it could take up to 24 hours before a new value for refresh date on the server reaches clients.
As the caching server side is simply data held in instance variables, bouncing uPortal will automatically trigger new calls to the remote feed and but as long as the lastmodified date still matches that held by clients in local storage, no new client requests to the server should occur.