Assignment 3

For this assignment, you are required to add eight new menus to the index.html file, under a group menu called *centennial*

These menus should add and remove the following layers from the *centennial* database, making use of the geoJSON API code you have written for the practical:

For a small screen size
- buildings
- university

For this assignment, the small and large screen sizes refer to Bootstrap small and large screens. No menus should appear on any other screen sizes.

For a large screen size
- rooms
- temperature_sensors

To ensure that you submit a clean piece of work, all other menus should be REMOVED – commented out – in the index.html file.

You will need to make sure your code connects to the *centennial* database for this assignment – this will involve changing your postGISconnection.js file.

Make sure that the location of the PostGIS connection details is not hard coded. The code should work out the Unix username so that it will run on our server.

To submit this assignment, you should create a branch in your GitHub *app* and *api* repositories called *assignment3*, and make sure that all the code from the practical is complete and contained within each branch.

NB – We are testing both API and APP for this practical so you should have a *assignment3* branch in BOTH of these repository

We will test your assignment as follows (example given for the App):

2. For the API, we will run dataAPI.js to start the node server. Make sure you include this file in the API.

3. We will test the functionality of your API with the following URL end points to make sure that GeoJSON is returned and that your server URL is not hard coded anywhere

   Note: you should use the ROUTE name specified in Moodle. This may be different from that in the practical.

   …/ucfscde/buildings/building_id/location
   …/ucfscde/university/university_id/location
   …/ucfscde/rooms/room_id/location
   …/ucfscde/temperature_sensors/sensor_id/location

   Route for this assignment (to be defined in dataAPI.js) is https://<<your CS server name>>/api/geojson24

   Make sure that you don't hard code the location of the server! Add the specific route, end point, schema, tablename, ID column and location column to the following string:

   let URL = document.location.origin+"/api/" ....

4. We will then run the app.js file to start the node server – make sure you have this file included in the branch!

5. We will then test the functionality of the index.html file –e.g.:
   - Can we click on the menu functions and load the and unload the required layers
   - Do the menus change depending on the size of the screen

   For this test we will use:

   https://<<our server IP address>>/app/index.html

Marking scheme:

Task	Mark
Can we see the four datasets on your Data API. NB – this is a foundational requirement – you will lose up to 8 marks if this doesn't work (i.e. marks from the next component are lost too). 0 marks if you have hard coded the path to the PostGIS connection details – the code should dynamically query your user name.	4
Can we load and remove the four datasets on your index.html page. 0 marks if the API doesn't work. 0 marks if you have hard coded the URL to the datasets instead of using the document origin. 0 marks if the file name is not correct 0 marks if you can add the same layer to the map more than once	4
JSDoc documentation should be present for all functions	2

NB: There should not be any other menus on the interface. 2 marks lost if there are extra menus.