# **Places to Stay Implementation Planning**

***Task 1 - There should be a web service to allow clients to look up all accommodation in a given location, and, separately, all accommodation of a given type.***

This task will require a PHP script in which HTTP methods of GET will be needed to get the location and type as learnt in Session 1. These will be assigned to variables to be used later. Using the PHP function of isset() the script can check that both the location or type has actually been provided in the query strings and then perform the appropriate SQL query. If neither have been provided, then an error message will be echoed to the client telling them at least one needs to be provided.

There will be three possible SQL queries which can occur these are:

* Searching the accommodation table for all records where the location matches that of the client value which will be used if only location has been specified in the query string.
* Searching the accommodation table for all records where the type matches that of the client value which will be used if only the type has been specified in the query string.
* Searching the accommodation table for all records where the location and type matches the relative query strings. This will obviously be used when both query strings have been provided.

The results will be assigned to an array using the fetchAll() function. Regardless of the type of search, these results will be an array consisting of associative arrays containing the accommodation details. JSON will then be generated from the results (assigned to the array) using the PHP function json\_encode() as done in Session 2. These can then be echoed out ready for the client to use. Should there not be any results found then the JSON is not encoded and instead an error message is echoed to the client telling them there was no results found.

Alternate Approach

The data output from the script could have been formatted in XML, however, this would have made it harder for the data to be parsed. This is because XML must be parsed using an XML parser whereas JSON can be parsed using standard JavaScript functions (w3schools, 2017). The problem with an XML parser is there are more steps involved just to parse data, as the XML document needs to be retrieved, looped through and the values extracted and stored into variables. The decision to use JSON was also made knowing AJAX will have to be used for later tasks which is also more difficult to do with XML.

Despite this, using XML would have been advantageous for creating a web service which would interact with more than just websites as XML is compatible with Java. This would mean that the web service could have been made available for software and mobile applications to use.

***Task 2 – There should be a web service to allow clients to book a place of accommodation for a given number of people on a given date. This should communicate its success or otherwise to its clients appropriately.***

This task will require a PHP script in which a few HTTP methods of POST will be needed to update the database using the client details (accommodation details, number of people etc.). This will be like the web service task written in Session 4. These details will first need to be checked if they have been provided using isset() and secondly checked if their format is correct for entry in the database. This will be especially important for data such as the date where all records need to have consistent formatting. Should one of the details not be provided or formatted correctly then an appropriate error message will be displayed relating to the problem.

These error messages will take the format as a simple error code rather than an English statement. This is because this script is a web service which will be used by many different developers of websites and by having a simple error code such as INVALID\_DATE rather than “date not valid” it is much easier for the developer to code a reaction to the error.

Once all details have been validated, then an SQL query can be run to input the details as a record in the database. However, before doing this there are some possible errors which may still occur and they will have to be checked using IF statements to check a variable before running an appropriate SQL query. These are:

* Checking that the date has been formatted correctly in the YYMMDD format as it is vital that records in the database use a universal formatting to avoid mistakes being made for booking.
* Checking the user is registered to use the service by checking their username and password matches a record in the database otherwise there could be a security issue where someone can falsely book rooms.
* Checking there is enough availability at the desired accommodation by checking the clients number of people against the number of available rooms.

Should any of these checks return false, then an appropriate error code will be displayed to the client to handle accordingly, otherwise the appropriate SQL query will run.

Alternate Approach

To output error messages to the client I have decided against using the standard of HTTP codes in favour of using custom error messages. My justification for this came after researching HTTP codes from the Mozilla developer web docs (Mozilla Developer Network, 2017). Whilst there are a variety of HTTP codes for numerous scenarios there are some instances in this scenario where none of them apply to a degree which I feel would be informative. An example of a scenario where this is the case is when there are not enough available rooms available to book for a client.

It would have been advantageous to implement HTTP codes for the server side of the implementation. This is because any errors which may occur on the server side could be displayed to the client, indicating it is neither theirs or the web services fault which has an issue.

***Task 3 - PlacesToStay should have an AJAX front end which connects to the web service, and in doing so, allows users to look up all places to stay in a given location. Results must be presented in a user-friendly manner.***

This task will require a HTML web page which will display a search form for which the user can type their desired location to search for. This form will only consist of a text box and button to submit the form. A JavaScript file will also be needed to handle the button functionality and subsequent AJAX function to search the database and display the results.

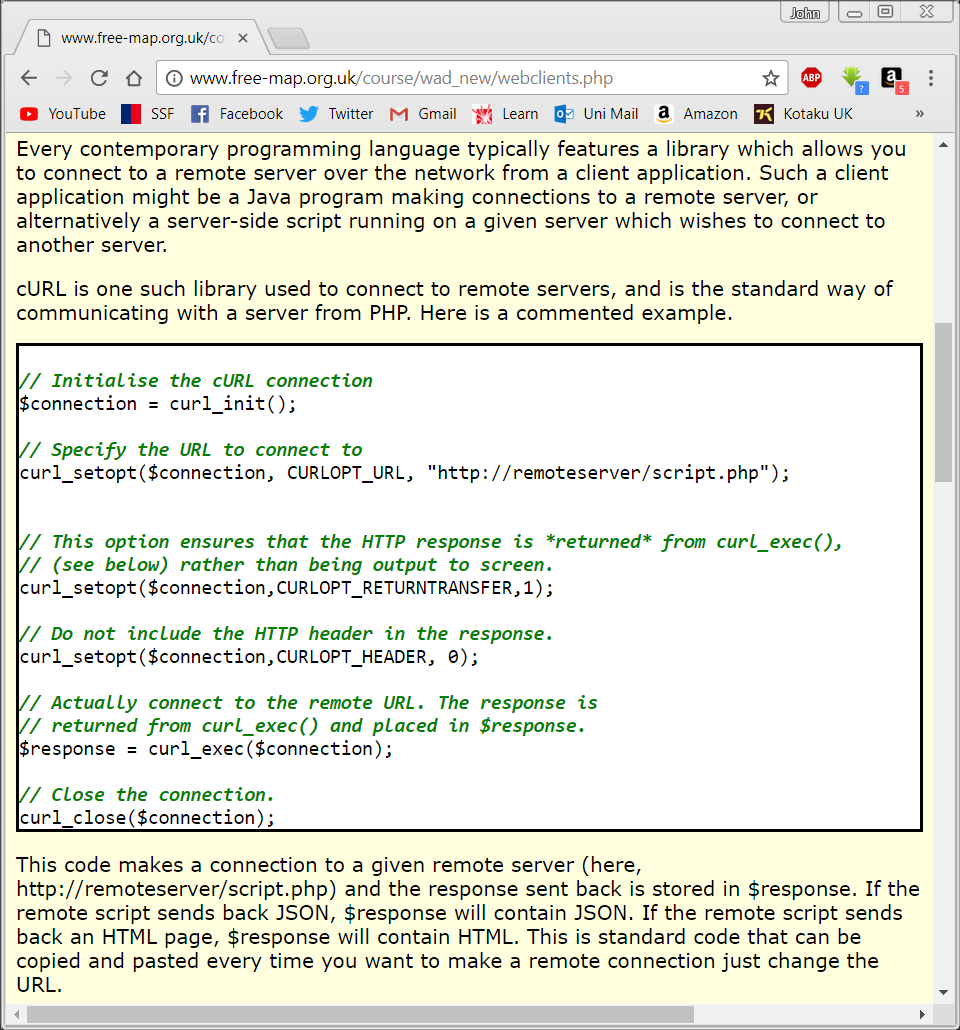
To maintain unobtrusive JavaScript, as learnt about in Session 13, an init() function will be made and will load with the body. Within it, the button will be assigned an onclick event listener which will then run the search when the search button is clicked.

Within the search function an AJAX request will be made to the accommodation search web service with the user’s value as a query string. To maintain tidy code an Arrow function will be used in place of a callback function. Within this arrow function an IF statement will be used to determine if the search succeeded by comparing the response text to the defined error message specified in Task 1. If the search has failed, then a user-friendly message can be displayed to the user. Like in session 7, the JSON returned from the web service will be parsed, loading the JSON into memory as a PHP array which will be looped through using a FOR loop, assigning all useful data to a variable. This variable will then become the search results which will be displayed to the user using the HTML DOM innerHTML property to assign the results to a div. However, before this an IF statement will be used to check if json\_decode() returned null, which will indicate there was no search results. If this is the case an error message can be displayed to the user.

***Task 4 - A third-party website, VisitHampshire, is a tourist information site for the county of Hampshire. At the moment this site uses its own database to store places to stay within Hampshire; however, it wishes to use the PlacesToStay web service instead. It should allow its own users to look up places to stay of a given type within Hampshire, making use of the web service. Results must be presented in a user-friendly manner.***

This task will require a HTML page which will be the homepage for the website. The page will have a form which will take in the type from a drop down box. On submission of the form the type will be assigned to a query string and passed to a new PHP script which will display the results. This will be a PHP page rather than an HTML page as it will make use of the cURL library.

Using cURL the page will connect to the web service in task 1. This task will implement the knowledge learnt in session 3 and will follow similar code shown below.



The only way it will differ is it will connect to the URL of the webservice developed in task 1. An IF statement will be done to check if the json\_decode() function returns null as this will indicate if any results were returned. If no results were returned, then an error code can be echoed to the client, else the $response variable which will contain JSON which will be parsed as done in task 3.

***Task 5 - PlacesToStay's own site (not VisitHampshire!) should include web maps (e.g. Leaflet, Google Maps or similar), and should use the Geolocation API so that on a device with GPS enabled, the site should be centred on the user’s current position.***

As this task will be done using the Leaflet JavaScript mapping library, used in Sessions 11 and 12, the necessary JavaScript and CSS files will need to be imported into the HTML head tag. A div will also be needed to house the map which will be displayed to the user.

In the JavaScript file a map object will be created as a global variable so that it can be used by other functions in later tasks. This map will need to be associated with the map div in the HTML. A map layer will need to be set up so that later markers can overlay the underlying map. This will be done using a TileLayer, a layer of map tiles from the openstreetmap servers.

An IF statement will be needed to test whether the browse the user is using supports GeoLocation. Should it not, they will be simply told their browser does not support it using an alert. If their browser does support it, their current location can be retrieved using the getCurrentPosition method and processed in a separate function. In the position processing function the user latitude and longitude can be retrieved and passed in to the setView method as a two-member array. The setView method will then centre the map on this location.

Alternate Approach

This task could also be completed using a different mapping library such as Google Maps, however, I believe Leaflet is the better choice for this task. The main reason I decided this is the fewer restrictions it has compared to Google Maps. Leaflet is Open Sourced and therefore it doesn’t cost money, and the branding can be more customized to suit the PlacesToStay branding. With Googles restrictions it makes it much more difficult to develop as “All free tier implementations must be publicly available — intranet usage and maps behind a paywall require a premium plan.” (Moore and Walz, 2016).

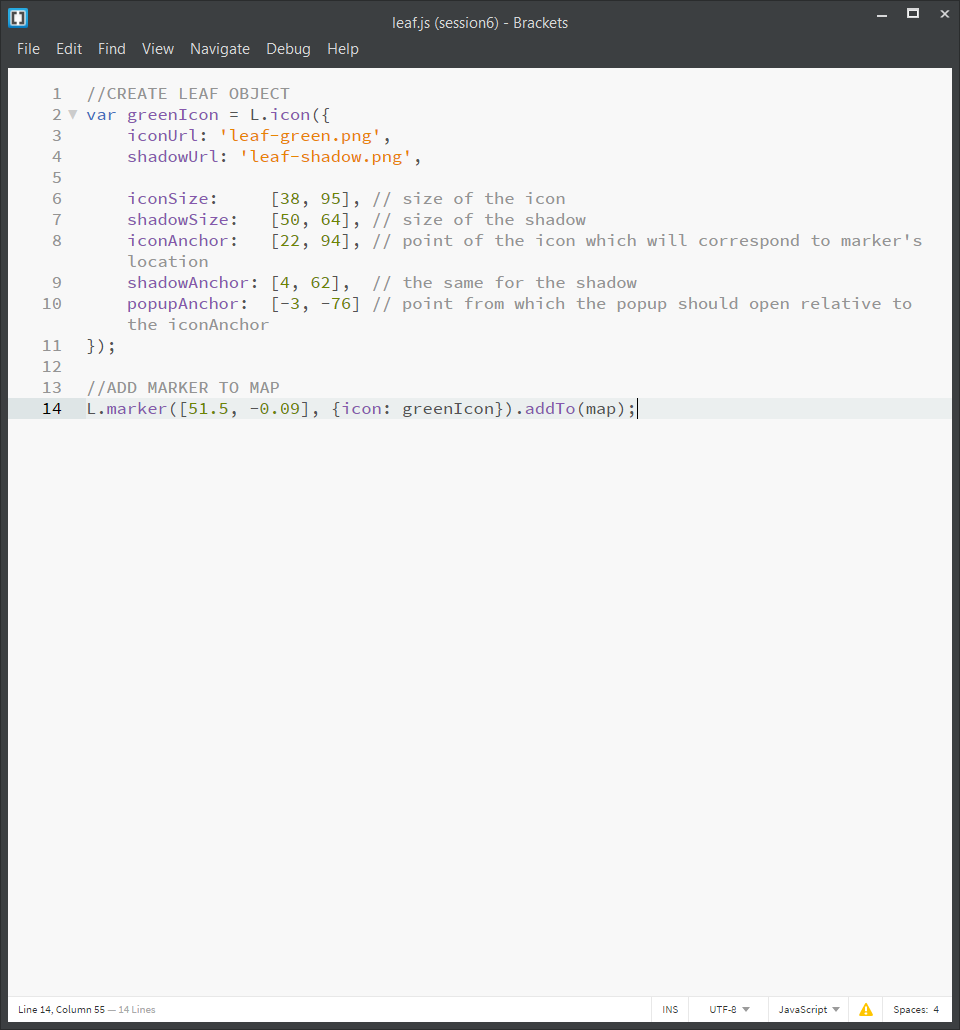
Google, being the multinational company that it is, has a wealth of information that does not compare to Leaflet. Google Maps can provide so much more information on a given location, most notably the Street View perspective which allows a user to see a location as if they were in that location, being able to view a first-person perspective rather than a bird’s eye view. Having this level of detail would have been advantageous for a tourist site such as PlacesToStay.

***Task 6 - On PlacesToStay's own site, the AJAX search (task 3) should be extended so that all items of accommodation in that location will be shown on the map as markers, as well as shown as text search results.***

This task will require extra code to be inserted into the FOR loop which displays the search results as mentioned in Task 3. As the FOR loop retrieves its results from the database it will also assign each locations latitude and longitude to an array in a variable. This variable will then be used to add a marker to the map overlay as learnt in Session 11. On completion of the FOR loop all markers should be added to the map and should match the search results.

Alternate Approach

Using advanced leaflet features custom markers could have been added to the map. This would have been done by creating an icon object and assigning it attributes, which would then be passed to the creation of the marker as done with the standard markers. The code below demonstrates how this process would be done:



(Leafletjs.com, 2017)

This would have given the map a better look and feel for the user, however I decided against doing this as the standard markers Leaflet uses would suffice for this task.

***Task 7 - VisitHampshire should allow a user to book an item of accommodation on a given date, by interacting with the appropriate PlacesToStay web service. This task should be linked with task 4, so that a user can book an item of accommodation after searching. Ideally, the user should be able to specify the number of people; if not, assume one.***

This task will require modifying task 4 so that there will be a form with a method of POST for each search result and a book button to carry out the booking. The form will need to take in the date and the number of people. The accID will also be needed but this will be a hidden field which will already contain the ID when the search results are loaded.

When they click the book button they will be taken to a separate login PHP file which will have a form for them to input their username and password so that the user can log in. Query strings of the date, number of people, accID will also be passed in the URL as this will be needed for the PHP script which will make the booking once the user has submitted correct login details.

In the booking handling PHP script, the query strings passed on from the login form will be assigned to variables. A cURL request will then be used to send all the details to the booking web service made in task 2 with a method of POST. Unlike the cURL request done in task 4 this request will use the CURLOPT\_USERPWD option which will be base64 encoded then placed in an Authorization line in the HTTP header.

The response from the web server will be assigned to a variable as in tasks 3 and 4 using json\_decode. This variable will then be checked using an IF statement to see if any of the potential errors from the web service has come up so that the client can display a corresponding error.

Alternate Approach

OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service (OAuth.net, 2017). For example, rather than signing up to use PlacesToStay a user could use their Google login instead. It works by the third-party application providing an OAuth-token, an encrypted message which specifies that the user is who they say they are. This token is then checked by the requested website and then they provide access. The advantage of this is it provides a secure way for a user to use a service without creating a new login and password for them to remember.

***Task 8 - When the user clicks on each marker, a popup describing that item of accommodation should appear. The popup should have a "Book" link allowing the user to book the item of accommodation. When the user clicks "Book" they should enter a date (plus ideally the number of people, if not, assume one) and the item should be booked for that date. The user should be provided with feedback as to whether the booking was successful or not. The front-end must fully use JavaScript and AJAX without any page refresh required.***

This task will require modifying Task 6. During the FOR loop which displays the search results, a variable will become a two-member array of coordinates for the accommodation currently being looped through. Each time the loop goes through an accommodation the latitude and longitude will be added to this array. This variable will then be used to display markers on the map overlay, with each result returned a new marker will be added. Using the bindPopup method, each marker can be given a popup in which the accommodation details can be displayed as well as a link for the user to book a room at the place. The link for each popup will have an onclick method assigned to it so that it will run a function to handle the booking once clicked. This method will take in a parameter which will be the ID. When the FOR loop is creating the links the accommodation ID will be passed to the function, so every link will correspond with its relative accommodation when rang.

On the PlacesToStay website there will be a hidden form which will popup and become visible when the user clicks the book link within a markers popup. This will be done using JavaScript to change the style of the form from hidden display to block display. The form will require the user to input a date, an amount of people and their username and password. By using the input type of date this makes it harder for the user to make an error as this will show an interactive calendar, allowing them to click a date rather than typing one in.

On submitting the form, the same process as described in task 7 will take place as they both interact with the same web service. If the username and password is not valid the JavaScript will display an error message telling the user, they are incorrect. More importantly if there is not enough room availability on their given day this will not update the database and an error message will be displayed to the user.

Alternate Approach

The user authentication for both this task and task 7 could have been implemented using session-based authentication, where a session variable is used to determine if a user is logged in or not. This method of authentication would not be good for this task as it would create a problem where the download web service would be dependent on another script. It could also result in situations where a mobile app user would not be able to use the site as the session might not be tracked automatically like in browsers. cURL is a much better method of implementing this task as it means the web service script will receive the login details directly rather than being reliant on another script.

**References**

W3schools.com. *JSON vs XML.* [online] Available at:

https://www.w3schools.com/js/js\_json\_xml.asp

Mozilla Developer Network. HTTP Response Status Codes. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Moore, M. and Walz, A. 2016. *Choosing a Mapping Framework.* [online] Available at:

https://luminfire.com/2016/05/10/choosing-mapping-framework/

Leafletjs.com. (2017). Markers with Custom Icons - Leaflet - a JavaScript library for interactive maps. [online] Available at:

http://leafletjs.com/examples/custom-icons/

Oauth.net. (2017). *OAuth Community Site*. [online] Available at:

https://oauth.net/