| WAD | **JAKE LAMBERT**  **Places To Stay - Report** |
| --- | --- |

### Requirement 1.

Lab work

In Lab 2, we learnt how to write a web service and send back data to the client in JSON format. The idea is we take in the destination and store it in $a. we send an SQL SELECT statement to the database server using the $conn->query() function, and obtain a result sent back. We then loop through each result set, row by row.

HTML format can be used to send information back to client but it is not good, as it contains not only data, but also page structure information. As we are a web service and providing just raw information to other applications, we want to send back the data as XML or JSON format. The client might want to arrange the information in a different way so I have chose to use JSON over XML as it represents the data alone and has a much simpler format. The function json\_encode(), automatically generates JSON from PHP arrays or associative arrays. As we are indexing using non-numerical indices representing the columns from the database, we will be using associative arrays. The clients to the search web service will be using associative arrays when decoding the JSON data to display to their users.

Some pros to JSON are its more compact and can be easily loaded into JavaScript. In addition, there is more structural information in the document so it is easier to convert into arrays and associative arrays. Some cons are it has no error handling for JSON calls. In addition, it’s quite dangerous if used with untrusted services or browsers, it can be hacked.

Pros for XML are namespaces allow you to divide the XML. Has better representation for inheritance. XML is structured by schemas and since they are interpreted by software, human errors in interpretation are eliminated.

References

Anon, (2017). [online] Available at: https://www.eetimes.com/document.asp?doc\_id=1277514 [Accessed 15 Nov. 2017].

Anon, (2017). [online] Available at: https://ezeelive.com/blog/json-advantages-disadvantages/ [Accessed 15 Nov. 2017].

Disadvantages?, X. (2017). *XML and JSON -- Advantages and Disadvantages?*. [online] Stackoverflow.com. Available at: https://stackoverflow.com/questions/5615352/xml-and-json-advantages-and-disadvantages [Accessed 15 Nov. 2017].

Slideshare.net. (2017). *It ain’t just for JavaScript*. [online] Available at: https://www.slideshare.net/simon/how-to-make-ajax-work-for-you/53-It\_aint\_just\_for\_JavaScript [Accessed 15 Nov. 2017].

Application to the scenario

The scenario requires us to develop a search web service to search for accommodation by type or location or both. Accommodations are stored in the database, therefore we need to write a PHP script that sends a SELECT statement to the database to find all accommodations matching the type or location or both the user specified.

The script is reading in the accommodation type & location or both. Firstly, we need to use ‘isset’ to check which query string parameter is set.

In an IF statement we check to see if both parameters have been set, if so then constructing an SQL SELECT statement we use type and location in the WHERE clause. The set of results is encoded into a JSON format to be sent back to the client.

If both parameters are not set then we will test if only ‘location’ has been set using elseif. If so, then we do the same as above apart from changing the WHERE clause to match only location.

Then if this is not set, we do the same above but with ‘type’ instead.

If none of the parameters are set then we send a HTTP header response of header("HTTP/1.1 400 Bad Request"); back to the client.

### Requirement 2

Lab work

In lab 4-5, we learnt POST-based web services that allows clients to update data. We created a download web service that reads in the ID as a POST request, then with a SELCT statement adds one to the downloads column.

We learnt about error checking, using an IF statement to check if the user has enough funds for purchase.

In Lab 6, we learnt to communicate back to the client we can use either custom error codes or HTTP codes. Custom error codes are easily readable by humans but pure-data status HTTP codes means any client, including smartphone apps, can connect to the web service, interpreting data easily to determine if its completed successfully. HTTP provides a standard set of error codes that make it efficient when communicating with different clients.

We looked at returning HTTP codes from web services. You can send back an appropriate HTTP code with a simple header function, header(“HTTP/1.1 404 Not Found”).

Session-based introduces a dependency of one script to another, meaning it needs a session-based login script to be run before a script that required authentication. This becomes difficult for the client as they need to keep track of the session variable, this is ok for web browsers but smartphone apps it might not.

HTTP authentication is more suited for web services. We use a HTTP header to send username & password to a script that requires authentication. Now this does not depend on any login scripts as it receives the login details directly.

Other authentication methods are cookies, a server would receive a HTTP request in the response, and it can send a set-cookie header. They occupy less memory and do not require any server resources and get stored on the user’s computer, so no burden for a server.

Application to the scenario

The scenario requires us to allow users to book accommodation. We are going to use HTTP error codes to send back to the client as it uses set error codes to communicate its status, whether its successful or not. These will be used to test if the user is logged in or if the ID parsed is true.

Authentication details need to be sent from clients to the web service, we are going to use HTTP authentication. The client uses CURLOPT\_USERPWD to pass username & password, where we use base64 to encode them and places them in an authorization line in the HTTP header. Then we read them in with $\_SERVER[“PHP\_AUTH\_USER”] & $\_SERVER[“PHP\_AUTH\_PW”] respectively. We run a SELECT statement to test if the username and password exist in the database table acc\_users using $conn->fetch() and fetch the result, then we run an IF statement to test if the result is false, if so then we send back 401 unauthorized back to the client.

Otherwise, we proceed bookings we read in the ID, the date, number of people and the username from the client. Then we run another SELECT statement to test if the ”ID” passed over is in the “accommodation” table, we fetch the result and put in a variable $row2. We run another IF statement so see if the result fetched is false, if so we send back an error message 404 not found back to the client which means that “ID” was not found in the database. If not then we run another SELECT statement to test if the accID and thedate from acc\_dates match the user input $id and $date respectively and fetch the result to put in a variable. Again we run an IF statement to test if the result is false which will result in 404 not found to be sent back. If true, we proceed to inserting accid, thedate, username and npeople into their respected columns. Also we update “acc\_dates” finding the record in the table that match the id and date, then update $availability by *using availability = availability - $npeople.* In addition, we send back a 200 OK response back to the client that means the request succeeded.

### Requirement 3

*Lab work*

In Lab 7, we learnt DOM that is an addressing system for web pages or XML documents. With DOM, we can access and manipulate properties on the page using Javascript. One of the main functions is getElementById() where it takes the value inside a input field and uses it to put in a variable. We can change the contents of a page element with innerHTML.

Classic web pages vs AJAX. Classic web pages give no instant feedback, when the application state changes then it just reloads the entire page. Whereas AJAX is dynamic which allows browsers to communicate with web servers but does not have to reload the page. Requests to the server are sent and get a response received without any interruption to the user’s interaction as it is done in the background.

We learnt to use JSON in conjunction with AJAX. We can use the JSON>parse() function to convert JSON into corresponding JavaScript arrays and objects. This will be done in the callback function responseReceived(), we parse the reponseText and put it into a variable *data.*

Arrow functions can be used as parameters where the callback is expected. Therefore, we write in the whole function where the callback is expected. We don’t need to use the separate function responseReceived() to handle the AJAX response.

In Lab 8, we learnt to test the HTTP status codes returned using AJAX callback function. We do not need cURL as the front end is on the same server as the web service. We can use the status property of e.target in the callback.

*Application to the scenario*

We need to have an AJAX front-end for Places To Stay, where we can connect to the web service to search for places of accommodation. We chose to use Arrow functions, we put the responseReceived() function in as a parameter where a callback is expected. This is very effective as it simplifies the code to become more readable.

So we setup an ajaxrequest() function to retrieve the location that the user entered and stores it in variable a using getElementById. We need to open the connection to the server with a dynamic query string.

We want the responseReceived() function to run when the response has been received and the load event has occurred, this is where we implement the arrow function to use responseReceived() as a parameter in the event listener.

In this parameter function we use JSON.parse() function to convert JSON into corresponding JavaScript arrays and objects. We use a for loop to loop through each set of results and displays them individually by their database column names, then puts each line in turn to the variable output.

We then test if there are any error codes returned using an AJAX callback function. We can simply use the status property of e.target to identify the error code. So we test IF(e.target.status == 404), if this is true then we display an alert box stating the server cannot find requested source. ELSE if the above is false then we can proceed with sending the data to change the contents of the div using document.getElementById('aside').innerHTML = output;.

We display this in a user-friendly format by inputting the data into table, we create the table column names in output variable when we create it, then in the FOR loop we add in the header and standard cells for the HTML table.

### Requirement 4

*Lab work*

In Lab 3, we learnt Networking Programming with cURL, it is a library used to connect to a remote server and it is the basic way of communicating with a server from PHP.

We had to write a fan website, to display a list of all their no. 1s using the JSON web service. We used cURL and had to parse the JSON using json\_decode(). To illustrate the fact that it is a separate website we saved it to the other Edward server, whilst the web service was saved on the edward2 server.

We initialized the cURL connection then specified the URL to connect to with the set query string. We then returned the HTTP response and placed it in a variable named $response then closed the connection.

To parse the JSON data we obtained the JSON returned from the server, and then we use json\_decode to load it into the memory as either a PHP array or PHP associative array. We used ‘true’ as the second parameter as we wanted to load the JSON data in array of PHP objects. A FOR loop is used to loop through each set of results using the variable $i and the column name from the database to display each result.

In Lab 8, we learnt how to create a form and use the Form Data object. This was only to POST data into the database where for the scenario we need to GET the data from the database.

*Application to the scenario*

As we are creating, a third-party website VisitHampshire we only want to search accommodation of the user’s chosen type in Hampshire. For this, we need the user to enter the accommodation type in a form then we send a cURL request. Firstly we initialize the connection then specify the URL to connect too, within the URL we connect to our web service that we only want accommodation of the user’s chosen type in Hampshire, [*http://edward2.solent.ac.uk/~lambertj/WAD/code/searchwebservice.php?location=Hampshire&type=$type*](http://edward2.solent.ac.uk/~lambertj/WAD/code/searchwebservice.php?location=Hampshire&type=$type)*.* Our web service will take in both location and type then send back the data to the front-end site.

We then ensure the HTTP response is returned and we do not want to include the HTTP header in the response.

We then connect to the remote URL, the response is received in the curl\_exec and placed in $response *$response = curl\_exec($connection);*. Then the JSON returned has to be decoded and we set the second parameter to true to load the JSON into a PHP associative array, *$data = json\_decode($response, true);*.

To display the data we use a FOR loop that runs through each set of results and displays them individually by their column names from the database.

To handle errors returned from the web service we use the curl\_getinfo method to return the error code from the web service then store it in variable $httpcode. Now we can run an IF statement to test for 200 the request has succeeded and 404 not found.

### Requirement 5

*Lab work*

In Lab 11, we learnt websites and smartphones use different ways to show maps. One way, Google Maps is used vastly and has a wealth of information. Google Maps provides Street View perspective, allowing you to see houses and roads from a drivers POI. However, Google Maps has its restrictions on their users and do not cater for specialized users like cyclists & walkers.

*https://www.techwalla.com/articles/disadvantages-advantages-of-using-the-google-maps-website*

OpenStreetMap is a Project that produces all types of maps and gives access for anyone to edit them. Its advantage is anyone can contribute themselves and do this by gong to the route/path and can use their current location GPS then use editing software afterwards to draw on the mapping routes themselves.

Leaflet is an open-source Javascript mapping library; it allows the user to embed a “slippy” map into a webpage. In addition, it offers similar functionalities to commercial web mapping services like Google Maps.

In the first example Hello World in Leaflet was created, it linked in the Leaflet library as an external Javascript file. In addition, the Leaflet CSS file was linked in. A div called map1 was used to hold the map then in the init() function a map object is created and associated with map1. Then a TileLayer was placed over the map to overlay the locations of points of interest.

In Lab 12, we learnt Geolocation API. In an example an init() functions runs to test whether geolocation is possible in the browser used by checking if the navigator.geolocation variable, if it doesn’t exist then an error alert box is displayed. If it does, then it takes the first parameter that is a function that runs as soon as the location is sent back from the GPS chip. Then the longitude and latitude are obtained and put inside the <div> with the ID of info.

This method was only obtaining the current position once, whereas we want it to update in real time. We can use a method called watchposition() instead of getCurrentPosition().

*Application to the scenario*

From the scenario we need to include a web map using the Geolocation API so the map in centered at the users location. To create a map we have a <div> with an id of maps and set the width and height. As we are using the Leaflet library we link an external Javascript file, also we want to use some advanced CSS features so we link in the Leaflet CSS file. We create a function init() where we create a map object and link it to the <div> map1 with the L.map method. Next, we setup a TileLayer to overaly the POI’s, the map tiles come from tile.openstreetmap.org where Leaflet automatically works out the correct values for Z, Y & X. This layer is added to the map and then we can setView to set the default positon of the map.

If we want the center of the map to follow the user’s current location then we can use Geolocation API. When the page runs the init() function runs. We are testing whether Geolocation is possible in the browser used, check existence of navigator.geolocation variable, if not we display an error message. If it exists, we use watchPosition(), the result will communicate the current location back to processPosition() as it’s the first parameter. The second parameter is an errorHandle() if there’s an error obtaining location. The third we set to true to enable use of the GPS chip. The forth we specify how frequent we want a refreshed reading of location.

processPosition(), this is supplied with one parameter gpspos, represents the position from the GPS device. We obtain the longitude & latitude and put them in a variable. We then need to put the current location into an array, we obtain the long and lat *with gpspos.coords.latitude* and *gpspos.coords.longitude*, then we setView to the position array that centers the map to the longitude and latitude coordinates.

### Requirement 6

*Lab work*

In Lab 11, we learnt to add features to our map like adding markers. In addition to our map from requirement 5, in the function init() we set the long and lat into a variable named pos and then set the map view like before but with the variable pos as the first part of the 2-member array. Then use L.marker to take in 2 array members long and lat then set them to the map with a marker.

The exercise in this lab wanted us to combine the marker and mouse click examples. To do this we use an on() method to attach the event handler to the map. It takes two parameters, event type and event handler. Then setup an onMapClick function where we use the event object e to obtain details about the event. In here, we can use latlng.lat & latlng.lng to set the map view.

*Application to the scenario*

As we are only changing the AJAX front-end search, which is also the PlacesToStay website we only need to add code into the AJAX function.

The JSON code gets loaded into a variable “data”. This variable “data” will contain whatever data structure is described in the JSON. So “data” will contain the search results and we access each object just like ordinary Javascript objects, “[i]” this represents the index number to obtain an object and this goes up by one for each search result.

In the function ajaxrequest() we want to add in a FOR loop that loops through each result and takes the longitude and latitude to store into two variables, then add them as a marker.

for(var i=0; i<data.length; i++) this is what the FOR loop tests. To add the longitude to a variable we use var Thelon = data[i].lon; and we do the same for latitude. To add the marker we use the L.marker method, L.marker([thelat,thelon]).addTo(map) this takes in the two parameters long & lat, and it adds them to the map as a marker.

One additional feature will can implement is to have custom markers dependent on type of place. We can create several icons then define our own icon class containing the shared options, inheriting from L.marker. Then we can create our markers and put them in a class with them individually linking to a ‘type’. Then to put them on the map we call L.marker and set the long & lat and the corresponding icons.

### Requirement 7

*Lab work*

In Lab 4-5, we learnt to create a web service that allows clients to update information. We had to write a download web service that will read in the ID from a form as a POST request to the database, that will increase downloads by 1.

We had to modify our web service to check for errors, so we can check using an IF statement whether the user has enough funds for example.

Also in the lab, we learnt to send a POST request using cURL. We put the data into an associative array and set this into the CURLOPT\_POSTFIELDS. We send this data over to our update script where we read in the data using $\_POST[“….”], and add them into the database.

Lab 6, we learnt to test the HTTP status code from a client. Once the cURL has been created this should be straight forward, you query the you query the *curlinfo\_http\_code* property to obtain the HTTP code returned from the web service. Then we can test for the error code with an IF statement.

*Application to the scenario*

The scenario wants us to book an item of accommodation by a chosen date and connect to the PlacesToStay search web service.

Firstly, we need to add a “Booking” link to the VisitHampshire website to link to a PHP form. In the FOR loop that displays the data from the search results we need to add a booking link for each result *<a href='form.php?id=" . $data[$i]["ID"] . "'>Book</a>* and pass over the “ID” of the accommodation item. The query string in the link has the id equal to $data which is the data returned in the json\_decode and [i] which represent the variable number in the FOR loop statement.

Onto the next page we have a PHP form for the user to enter the date and number of people, we will make the date field required but for the number of people field it doesn’t matter as if they don’t enter a number we will just assume one persons. We can do this with a simple drop down menu and have it always set to 1, or they can chose more if they need to. We will use a hidden input field to pass over the ID and the form action will be sent to a clientbooking HTML page.

In the clientbooking page we use the POST cURL request, we need to read in the data from the form and the ID from the hidden field and store them in their separate variables i.e. *$id = $\_POST[“ID”]*. We first setup the cURL connection variable then we specify the URL to connect to being the booking web service. We then setup an array with the ID, date & npeople variables i.e. *array("ID" => $id )* and store them into a variable $dataToPost. Rather than adding the data to the end of a URL using a query string, we can set the CURLOPT\_POSTFIELDS option with the data we want to POST as an associative array, thus putting $dataToPost as a parameter. This data is sent over to the booking web service.

Staying the client booking page we need to test the HTTP response, once we’ve done the cURL request we query the *CURLINFO\_HTTP\_CODE* property to obtain the HTTP code returned from the booking web service *curl\_getinfo($connection,CURLINFO\_HTTP\_CODE)* then set it in a $httpCode variable.

After closing the cURL connection we run an IF statement to test the HTTP code response, so if its 200 then we know the request has been succeeded, if its 405 then the accommodation is not available or if its 400 then the accommodation “ID” doesn’t exist.

### Requirement 8

*Lab work*

In Lab 11, we learnt to use a commonly encountered feature in web mapping, popups that the user clicks on the marker and then is presented with information about it. We call the bindPopup() method of the feature to attach the popup to that feature. Its takes one parameter, which is the text.

In Lab 8, we completed the more advanced exercises using the DOM to add a download functionality. In our responseReceived callback function we changed the existing code in the FOR loop then we replace it with a DOM paragraph (p) element for each hit returned using *var p = document.createElement(“p”)*. within the p we created a text node *var textNode = document.createTextNode(“”)* and had it contain the details of the song using *data[i].artist* etc. We add an input download button then we add both to the paragraph node using appendChild().

Preferably we are going to use AJAX functions with the FormData() method over using DOM. AJAX function are cleaner and make it easier to pass through variables.

*Application to the scenario*

We want the marker to display a description once the user clicks on it, we need to call the method bindPopup() to attach a popup to the feature. Referring back to the FOR loop we used for requirement 6 to display items of accommodation as markers, we can add a bindPopup() to the FOR loop. So then we can retrieve the description using data[i].description, so 'data' will contain a row from the database and [i] will access an object individually. In addition, we will add a button in the bindPopup using the onclick event that will link to the AJAX function ‘form’ and pass through the ID of the accommodation item *onclick=’form(data[i].ID)’*.

In the AJAX function, it will create an XMLHttpRequest variable and then read in the ID as an integer. We specify the callback function, when we get a response, the callback function will run. Then open a connection with the server using a ‘POST’ request, the URL will be the ‘form’ page and we’ll use a query string to pass through the ID. In the callback function we use *document.getElementById(‘’).innHTML =e.target.responseText* to add the form into a div on the VisitHamshire page.

In the form, we have to read in the ID and store it in a hidden field. We will have an input field for the user to type in the date; this will have an id of ‘accID’. A drop-down list will be used for the number of people, the default value will always be 1 just in case the user doesn’t specify how many people, this will have an id of ‘npeople’. Lastly we will have submit button that will link to an AJAX function ‘clientbook’ where we will pass over the ID again the same way as before.

In this AJAX function we read in the npeople and date from the form using *document.getElementById(“”).value* and store them in variable a & b respectively. Then read in the ID as an integer, also in the function name we put int into the brackets like so *function clientbook(int).* To send this POST request from AJAX, we need to create a FormData object and then append each item of POST data to it. The FormData object will be supplied as a parameter to the xhr2.send() function. To do this we create a new FormData() variable ‘data’, then we append the data using *data.append(“npeople”, ‘a’);* the npeople refers to the database column and ‘a’ refers to the npeople user input variable. We send a request to the server “clientbook.php” and there is no need for query strings as the FormData will pass the data over for us.

We let the ‘clientbook.php’ do its thing reading in the 3 variables then the appropriate HTTP error codes will be sent back to the VisitHamshire page where we can use an IF statement to test for what error code was sent, then the user can be displayed with feedback of an appropriate custom error message.

The error codes sent back could be 200 ok the request has been succeeded, 400 bad request the server couldn’t understand the request due to invalid syntax, 404 not found the server cannot find requested resource maybe invalid ID.

### Additional database tables

For users of the booking web service, I believe there should be a database table to store all usernames and passwords. This will be so we can authenticate who the user is, thus they will have to login before making any sort of booking.