31st May 2017

*Application to preferably be tested in Google Chrome*

Project Report (50%)

By Calum Oke (n9716751) & Douglas Kumar (n9714375)

**Website link:**<http://fastapps04.qut.edu.au:8080/n9716751/cab230project-master/>

**OR** select files via the **cab230project-master** folder

Contents

[Test Plan: 2](#_Toc484030550)

[Accessing the home screen: 2](#_Toc484030551)

[Registering as a new user: 3](#_Toc484030552)

[Valid Input: 3](#_Toc484030553)

[Invalid Input: 4](#_Toc484030554)

[Logging in as a new user: 9](#_Toc484030555)

[Logging in as an existing user: 10](#_Toc484030556)

[Logging out: 11](#_Toc484030557)

[Adding review: 12](#_Toc484030558)

[Searching for an item that exists in the database: 14](#_Toc484030559)

[Searching for an item that does not exist in the database: 15](#_Toc484030560)

[Accessing an individual item page: 16](#_Toc484030561)

[Attempting to use a cross site scripting attack but not being successful: 17](#_Toc484030562)

[Attempting to use an SQL injection attack but no being successful: 18](#_Toc484030563)

[Unregistered user not being able to log in: 19](#_Toc484030564)

[Operating gracefully in multiple resolutions: 20](#_Toc484030565)

[An Example of a SQL Query that has been implemented a description of where this Query is located (for example the file and method names): 21](#_Toc484030566)

[On the search results page: a map showing markers for all search results. (Add on #1): 22](#_Toc484030567)

[On the individual item page: a map showing the item. (Add on #1): 23](#_Toc484030568)

[Evidence that the geographic microdata is valid as reported by Google’s structured data validator (Add on #2) 24](#_Toc484030569)

[Evidence that the microdata is valid as reported by Google’s structured data validator (Add on #2) 25](#_Toc484030570)

[Project Meeting Web Design Principles: 26](#_Toc484030571)

[User Experience: 26](#_Toc484030572)

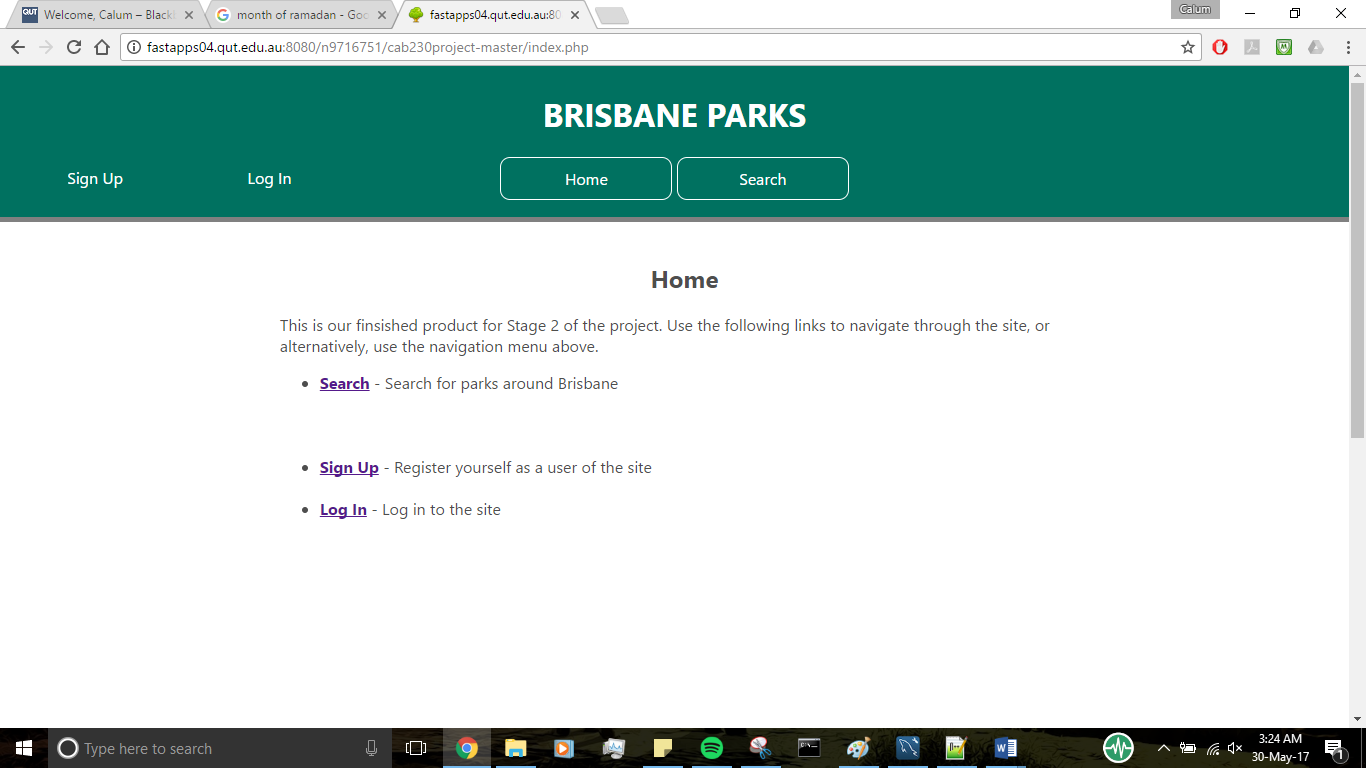
[Visual Design: 30](#_Toc484030573)

[Page Layout: 31](#_Toc484030574)

[Standards: 32](#_Toc484030575)

# Test Plan:

## Accessing the home screen:



An image of the index screen or home screen (index.php)

## Registering as a new user:

### Valid Input:

|  |  |  |
| --- | --- | --- |
| **Input Type** | **Screenshot** | **Explanation** |
| First Name |  | User entered an appropriate length of characters and valid characters |
| Last Name |  | User entered an appropriate length of characters and valid characters |
| Date of Birth |  | User entered date in correct format (dd-mm-yyyy) and within a future date and earliest possible date (1st January 1900) |
| Email Address |  | User entered an appropriate email address in correct format ([string of characters], [‘@’], [string], [‘.’], [string] |
| Address |  | User entered a numeral, with a sequential string to represent their address |
| State |  | User selects an option for their state |
| Postcode |  | User enters a 4-digit numeral value and no other character types |
| Password |  | User enters an 8-digit password as it meets the 8-digit minimum requirement (“password” is password) |
| Confirm Password |  | User enters the same 8-digit password that matches the ‘Password’ input |

|  |  |
| --- | --- |
|  | The result of a successful registration, the user is automatically logged in displaying their first name |

### Invalid Input:

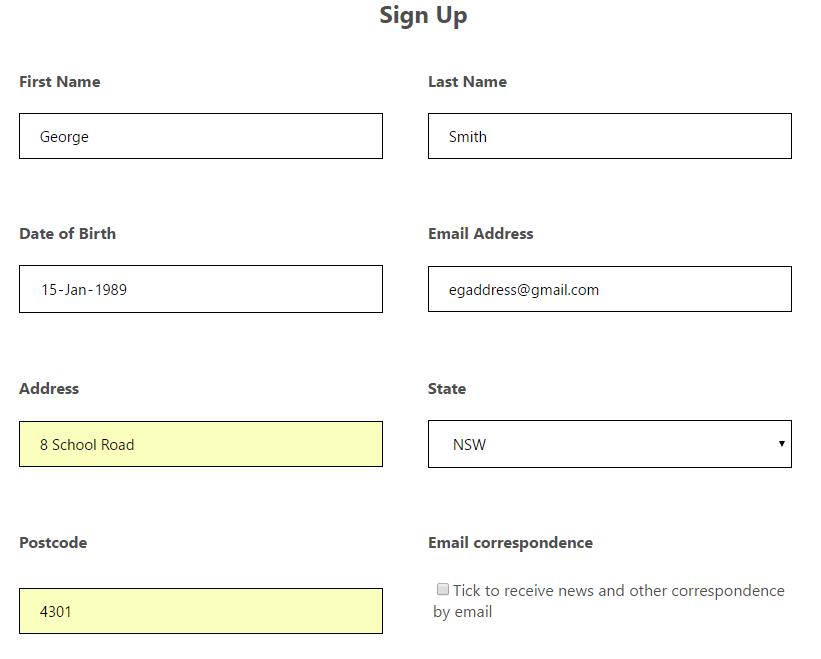
|  |  |  |
| --- | --- | --- |
| **Input Type** | **Screenshot** | **Explanation** |
| First Name |  | Users entering numerals |
| First Name |  | Users entering symbols or special characters |
| First Name |  | Users not entering input triggers HTML5 validation by default using “required” attribute |
| First Name |  | Users entering whitespace characters |

|  |  |  |
| --- | --- | --- |
| First Name |  | Users entering a single valid character (letter) |
| First Name |  | Users entering a string longer than 30 chars (tested 41 chars in example) |
| Last Name |  | Same testing methods as above  (single character) |
| Last Name |  | (numeral inputs) |
| Last Name |  | (symbols/special characters inputs) |
| Last Name |  | (length of string > 30 characters [31 characters inputted]) |
| Last Name |  | (no input) |
| Last Name |  | (whitespace characters) |
| Date of Birth |  | Users entering a future date of birth (tested on 27th of May 2017) |
| Date of Birth |  | Users entering a date too far into the past (minimum requirement 1st of January 1900) |
| Date of Birth |  | Users not entering any form of date |
| Date of Birth |  | Users only partially entering a date of birth |
| Email |  | Users not entering any input |
| Email |  | Users not entering a ‘@’ character to detect as an email |
| Email |  | Users not appending an address after ‘@’ character |
| Email |  | Users not appending a legitimate address (requires a ‘.’ character between 2 string values) |
| Email |  | Users entering an invalid character type into an email address |
| Email |  | Users failing to enter a sequential string value after the ‘.’ character for a domain |
| Address |  | User enters a numeral, but no sequential string. Or users enter a string, but no number |

|  |  |  |
| --- | --- | --- |
| Address |  | Users enter the string representing the address first, and the number after (needs to be the other way around) |
| State |  | Users fails to select a state |
| Postcode |  | User fails to input a postcode |
| Postcode |  | User enters not enough, too many numerals or letters/other characters |
| Password |  | Users enter a character length < 8 (4 characters entered) |
| Password |  | Users fails to input a password |
| Confirm Password |  | Users fail to input a password |

|  |  |  |
| --- | --- | --- |
| Confirm Password |  | Users enter a strong of different values in each password type (first input is “password” and second is “passwerd” |

## Logging in as a new user:



A new user is registering an account with valid information details. After submitting the new user is automatically logged in which is shown in an image below:



A log out option appears and the user is greeted with a message to notify their logged in.

## Logging in as an existing user:

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Screenshot** | **Description** |
| Registered email, wrong password |  | The email address is a registered account, however the password they entered does not match the email address’s account, which provides the error (password entered was ‘passwerd’ |
| Wrong email, registered password |  | The email address is not a registered account, but the password was right, and other registered users have the same password. Due to using hash values and concatenating salt values, this prevents a user logging into another one with a bad combination |
| Right combination |  | The email address and password match the registered pair in the database, thus allowing them to log in. The following image shows they have successfully logged in |

## Logging out:

|  |  |  |
| --- | --- | --- |
| **Event** | **Screenshot** | **Event Description** |
| Already logged in, logging out |  | The ‘Log Out’ option is available if the user is already logged in |
| Logging out |  | If the user presses the “Log Out” label, they will be redirected back to the log in screen, and terminates their session |

## Adding review:

|  |  |
| --- | --- |
| **Status** | **Screenshot** |
| The user has searched for a park they wish to visit. However, since they are not logged in, they physically do not have the option to write a review, so instead the page shows an option to sign up or log in to review the park |  |
| The user has searched for a park they wish to visit. This time they are logged into a registered account and have privileges to write a review. To proceed in writing a review, they must press the button highlighted in the screenshot |  |
| After the user has clicked the button, a text-area and slider appears. The text area is for users to input their description of their experience with the park. The slider provides the rating value the user wishes to rate the park from a scale of 1 (lowest) to 5 (highest). |  |
| After the user clicks the ‘Post’ button, the user is redirected to the same page (basically a page refresh) and displays the review the logged in user has just posted including the first name, rating value in image format, the text/description of the park, and the date posted. |  |

## Searching for an item that exists in the database:

|  |  |
| --- | --- |
| **Item** | **Screenshot** |
| The data about each park was imported into the database from a csv file provided via Blackboard. All existing items can be searched via the website. In this example, Downey park is an existing item that was searched via the website. |  |
| In the database, a query is run to find the park searched in the website. |  |

## Searching for an item that does not exist in the database:

|  |  |
| --- | --- |
| **Event** | **Screenshot** |
| The searched item is park that has not been imported in the database, as it was not in the csv file. |  |
| A query is conducted to search for the park item, however no results were found as displayed to the right |  |

## Accessing an individual item page:

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| The user searches for a park in the search bar (7th Brigade park) and clicks the search button |  |
| The user is then directed to the results page (results.php) depending on what they searched for. The user clicks on the link in the ‘Name’ column which them directs them to the park specified (or they could click on the map marker and direct them from there) |  |
| The user is then directed to the individual item page (park.php) depending on the park they chose to view. It displays the overall rating of the park, depending on the average results of all rating submitted, and the reviews, plus the location of the park |  |

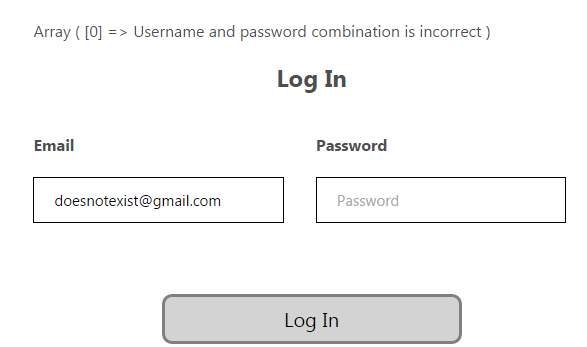
## Attempting to use a cross site scripting attack but not being successful:

|  |  |
| --- | --- |
| **Attack** | **Screenshot** |
| The user writes a html script tag with JavaScript syntax to redirect the user to Facebook |  |
| Since the htmlspecialchar() function is used, the tags are read as just normal text, so the review is posted as plaintext to the individual item page. |  |

## Attempting to use an SQL injection attack but no being successful:

|  |  |
| --- | --- |
| **Event** | **Screenshot** |
| In the park.php page, a user writes a SQL query to delete all registered users with the first name containing ‘John’ in the database |  |
| The user posts the review containing the SQL query, however this does not do anything, due to a “prepare(query)” statement used in all SQL query PHP variables. Htmlspecialchars() also prevents the user from script attacks |  |

## Unregistered user not being able to log in:



The email account inputted above is not registered to the database, and therefore resulting in the PHP error displayed above. The user would have to sign up first in order to register the account to the database and log in successfully.

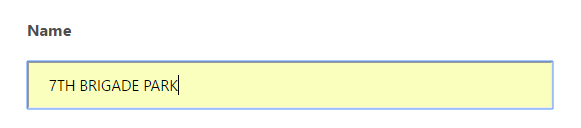
## Operating gracefully in multiple resolutions:

All resolution testing was completed using Screenfly (<http://quirktools.com/screenfly/>) and webpage used to test it (http://fastapps04.qut.edu.au:8080/n9716751/cab230project-master/park.php?itemId=1):

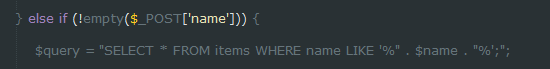
|  |  |
| --- | --- |
| **Resolution** | **Screenshot** |
| 1024 x 600 Desktop |  |
| 375 x 667 Apple IPhone 6/7 |  |

|  |  |
| --- | --- |
| **768 x 1024** Apple IPad (rotated) |  |
| **1920 x 1080** 1080p Television |  |

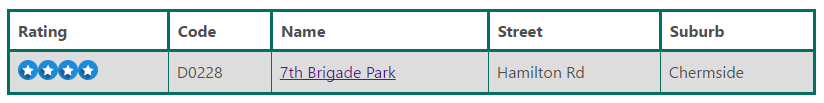
## An Example of a SQL Query that has been implemented a description of where this Query is located (for example the file and method names):

The screenshot provided above shows a list of different SQL queries that change value depending on which methods the user searches for a park (in search.php). These queries are located in ‘results.inc’ which produce the results depending on how the user searches for parks (producing the individual items page).   


**Submits to Execute**

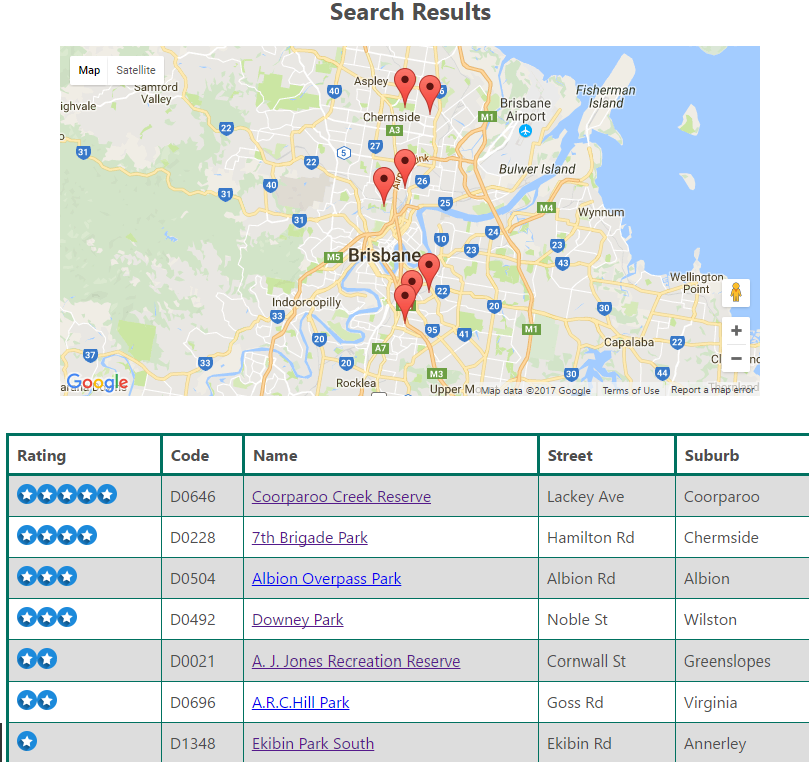


**Executes to Produce**

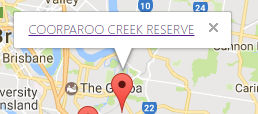


## On the search results page: a map showing markers for all search results. (Add on #1):

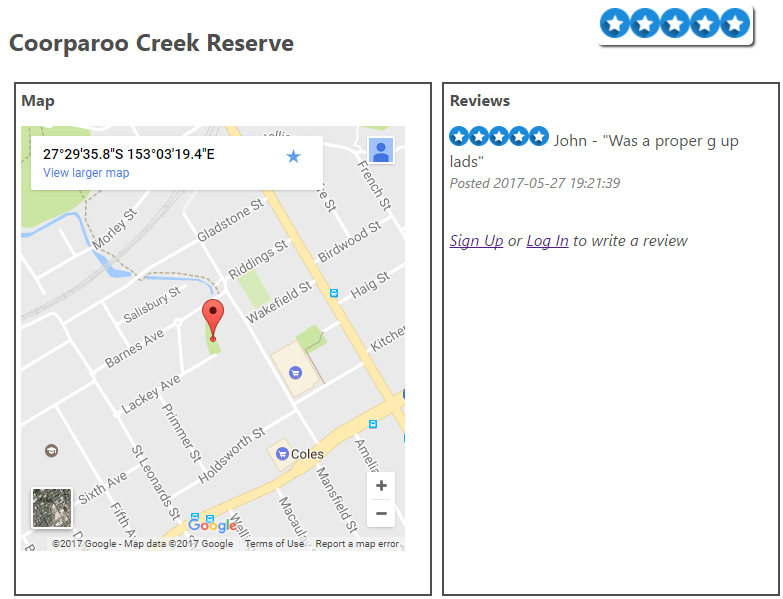
The following results showed up when the user does not input any search methods and clicks ‘Search’



As seen above, the amount of map markers matches the quantity of parks that are displayed in the table. A marker can be clicked on to show the name of the park and a link that can be clicked to lead straight to the individual item page/park.



## On the individual item page: a map showing the item. (Add on #1):

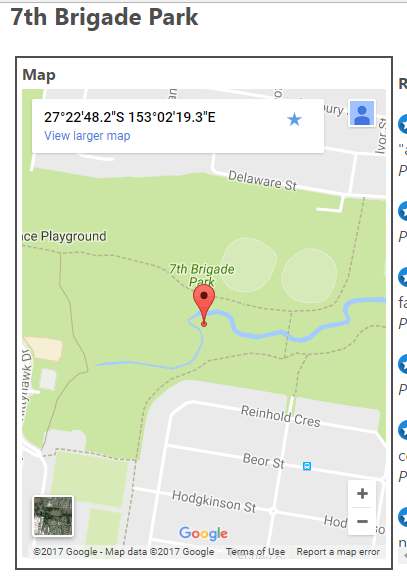
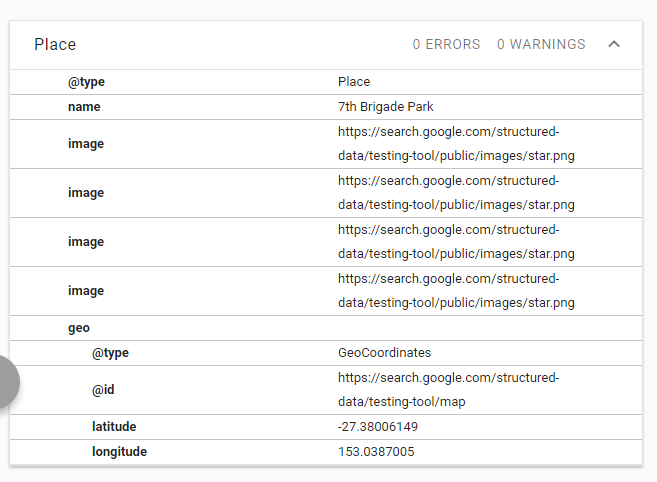


Clicking on the link in the previous section, sequentially leads to the individual items page, which is displayed above. The marker is pointed to the position where the park is located with the exact coordinates on the map.

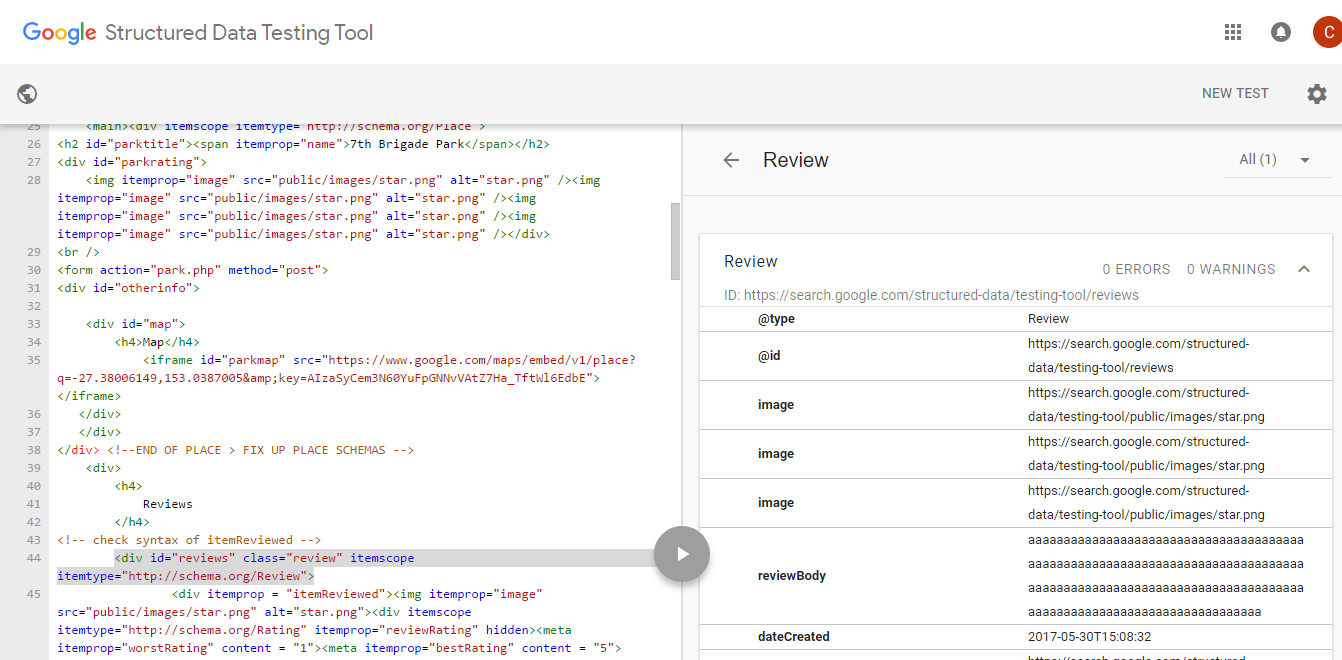
## Evidence that the geographic microdata is valid as reported by Google’s structured data validator (Add on #2)

The following screenshots below display the microdata implemented with the Place class/schema and implement the properties from it. The arrows from the webpage point to the relation that is found in the microdata. The Google data structured tester resulted in no warnings or errors.

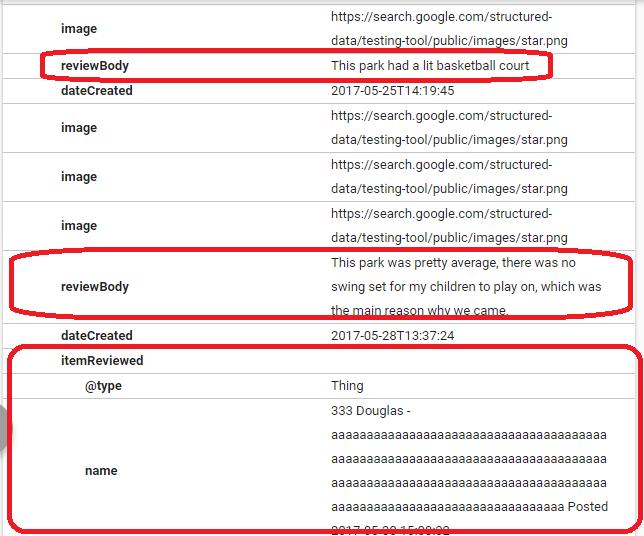


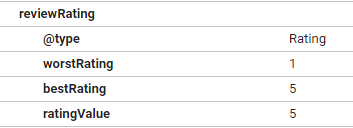


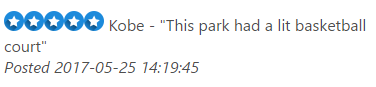
## Evidence that the microdata is valid as reported by Google’s structured data validator (Add on #2)



As seen above, all microdata implemented for Reviews (<http://schema.org/Review>) contain no errors or warnings. The 3 properties that are implemented in Review is reviewBody, itemReviewed and reviewRating which have all been implemented below, matching with the reviews entered in the webpage







# Project Meeting Web Design Principles:

## User Experience:

To create a simple and easy user experience (UX), a sitemap is created to demonstrate how the sight flows:

**IF LOGGED IN**

Index/Home

Search

Login

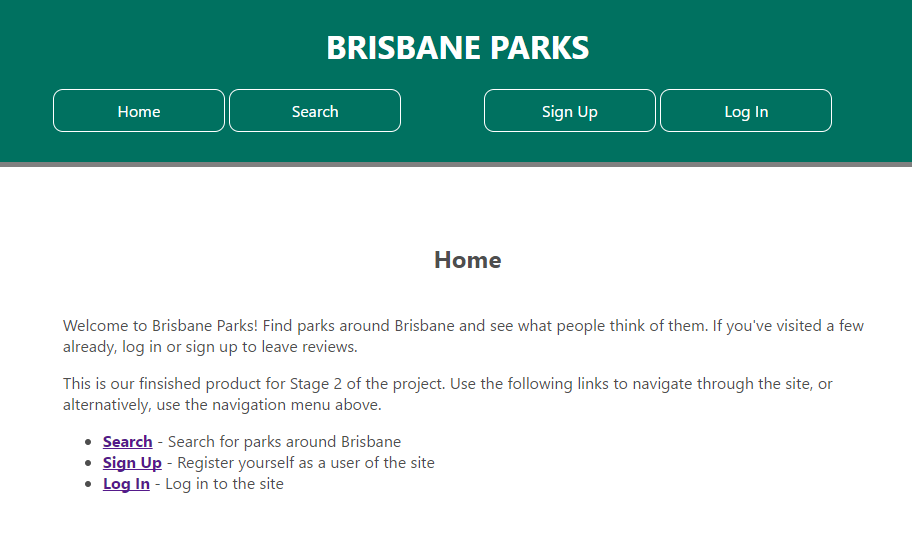
Sign Up

Results

Individual Item

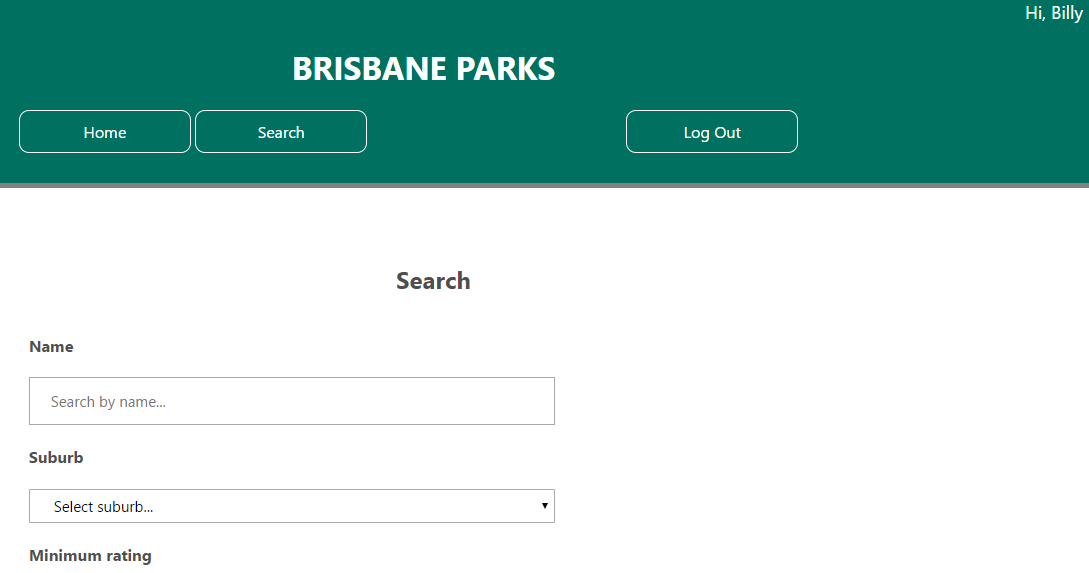
Review

The flow diagram above displays the sequence of webpages the user can possibly interact with or actions to complete. From index, the user can sign up for an account or log into their account. Then the user proceeds to the search page and inputs a method of searching with leads to the results page. From the results page, the user can navigate to the individual park/item and if they are logged in, can leave a review. Process is repeated until the user logs out. The screenshots below show the implementation of the flow diagram incorporating UX into the website.



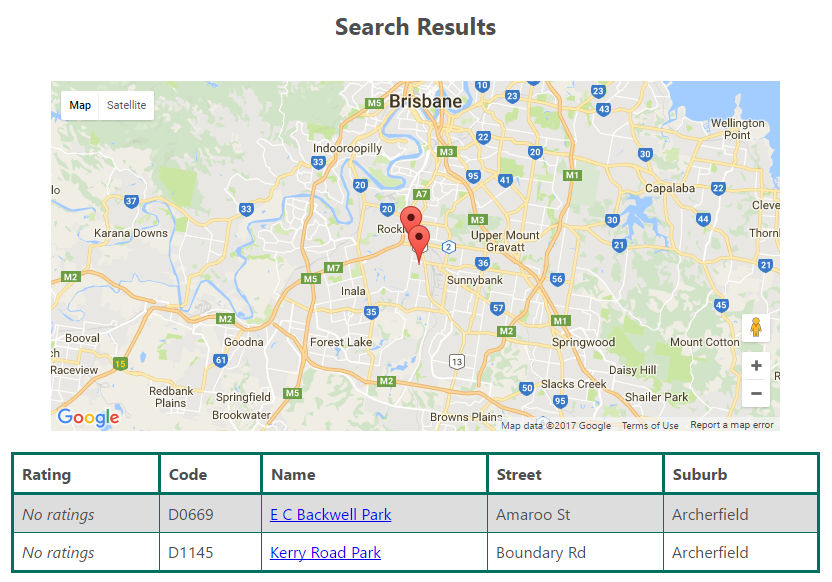
Menu tabs to indicate to the user how to interact with it

Header to indicate user which page they’re on

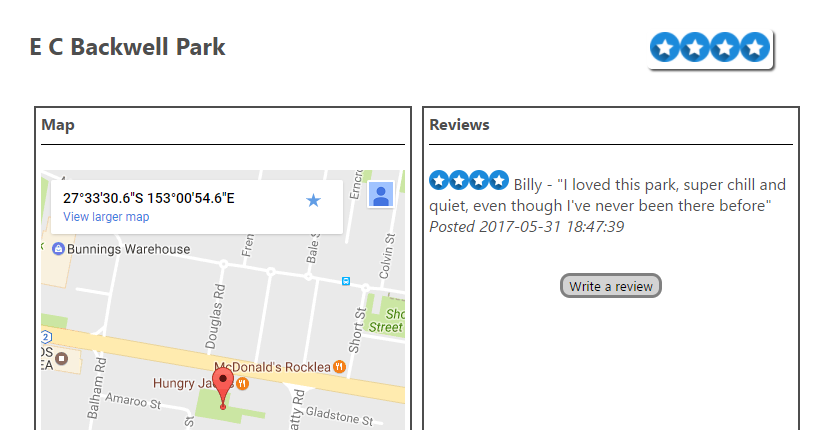
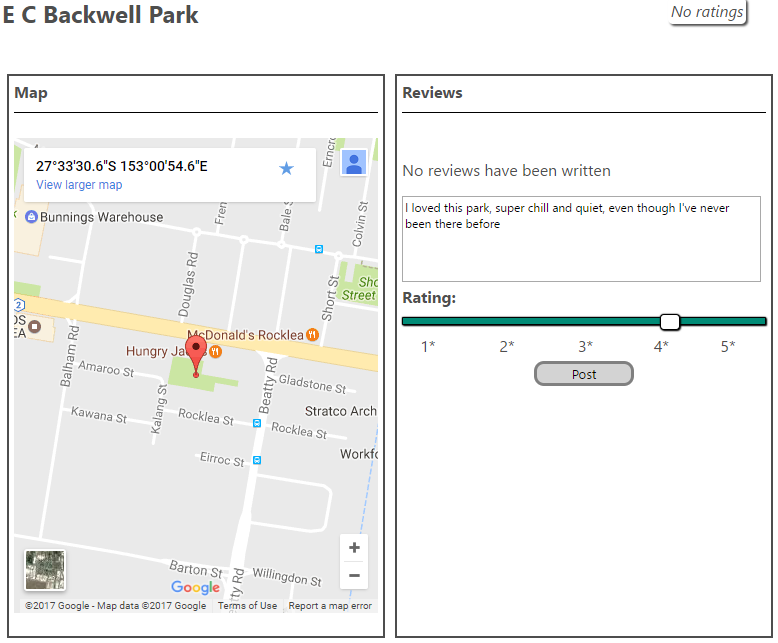


Methods/Options for the user to complete to do a search

Indicates to the user they’re logged in and option to log out



Map of results based of the user’s search and list of results

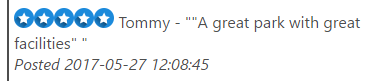


The individual results page with a written reviews before submission, user can submit because they’re logged in

User can easily see they’re review has been posted and repeat the whole process from home page again

## Visual Design:

The website’s purpose is to act as a service to users that allows them to search for parks, discover what other users think about these parks, and contribute to the reviews, continually enhancing the collective knowledge of the user base.

The user is a person who wishes to search for parks using varied criteria and analyse the reviews to select the best park for them to visit. They may also be a person who wishes to express their opinion about a park they have visited before.  


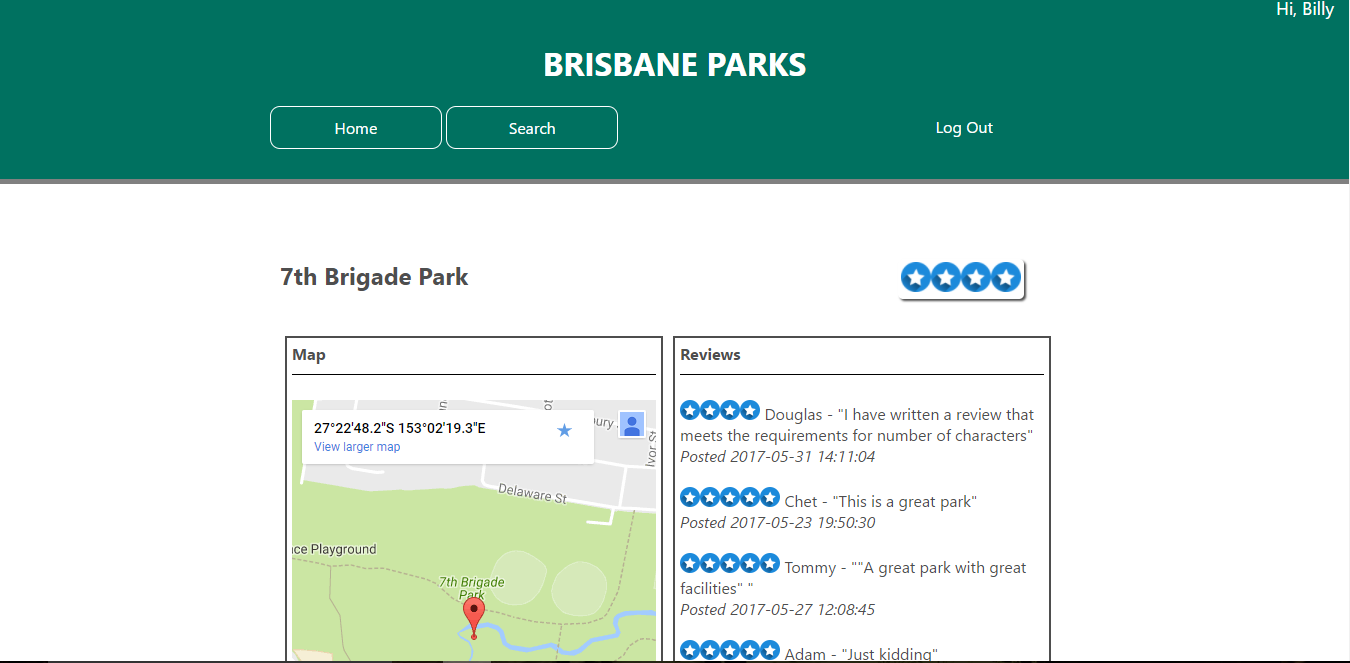
As the site is a platform for constructive social discussion and expression, a welcoming and conservative feel was desired so that users’ tendency to continue using the site would be increased. This was achieved through a consistent theme and layout using a cooler green shaded colour scheme.



This green (screenshot from banner in webpages) creates imagery similar to that which would be associated with a typical park. Helpful instructions and placeholders are provided for users to gently guide their use of the site. Deliberate use of whitespace along with structured content on each page conveys a contemporary feel and is suggestive that users should act calmly and maturely when using the site. The use of plain fonts and easily readable text allows the users to quickly gather the information they need use the site in an efficient manner, never straining or scrambling for information. When the site is viewed on different browsers, devices or at changing resolutions its visual features react and restructure themselves accordingly, making the site highly portable and usable under various conditions.

## Page Layout:

It can be observed that the website includes just the essential elements on each page that allow the user to use it effectively. There are no excessive blocks of text or clusters of html elements. In this way, the user receives only the information they want and aren’t frustrated by having to waste time scanning for it. The elements that are present on each page are laid out in a sensible fashion down the page, with each aligned and placed in organised grid style.



Main content  
is centred with even  
measures on both sides

Where possible, according to the screen resolution and size of the browser window, a frame of whitespace is kept around the main content of the page to create a visual separation from the common elements of the site and draw the user’s eye to it. In addition, borders encase all input elements where appropriate to make it easy for the user to distinguish between them. These input elements are also labelled clearly, allowing them to be easily identified even while skimming the page.

A fluid page layout is maintained across all pages and elements dynamically arrange and resize themselves when the browser dimensions are changed such that the site. This supports the user if they are multitasking alongside their browser window or using a device with more limited pixel dimensions.

## Standards:

HTML5 semantic elements – <main>, <header> etc. – as well as div elements were used to organise content on each page. The code was passed through the W3 Schools validator without errors (<https://validator.w3.org/>). Only 1 warning was alerted, however this is due to some web browsers not being compatible with the date() function in PHP.

|  |  |
| --- | --- |
| **Webpage** | **Validation** |
| Home page (index.php) |  |
| Search page (search.php) |  |
| Login page (login.php) |  |
| Sign up page (signup.php) |  |
| Results page (results.php) |  |
| Individual items page (park.php) |  |