# CAB230 Web Computing

## Project: Implementation of a data-driven website

**Worth**: 50%
**Due: Monday 28/5/2018**
**NOTE**: Stage 1 must be completed by the end of the semester break and demonstrated it in Workshop 7 practicals. It is worth 5% of the overall assignment.

If you do not demonstrate it in Workshop 7 and you do not have an approval for extension, you will get zero (0%) for this part.

Stage 2 will be demonstrated in Workshop 12.

**Teams**: The project needs to be done in pairs and occasionally you may have a team of three persons. You need to get approval for working individually. It is preferable that you form team with people in the same practical class.

# Section 1: Change History

| Date | Change Detail | Version |
|---|---|---|
| 18/04/2018 | Updated section 3.2.2 | 1.1 |
| 03/03/2018 | Initial version | 1.0 |

# Section 2: Synopsis

This project will require you to design and implementation of a data-driven website using HTML, CSS, JavaScript and HTML Forms on the client side and PHP and MySQL on the server side. In stage 1, you will only implement the client side of the website. In stage 2, you will implement the dynamic server-side functionality. You must submit your code and database to CAB230 web server and a written report to blackboard by the due date.

# Section 3: Stage 1

## Section 3.1: Overview

NOTE: Stage 1 must be completed for and demonstrated in Workshop 7. It is worth 5% of the overall assignment.

You will design and partially implement a website that allows users to provide reviews of items, similar to a restaurant review site like Yelp or Urbanspoon. You must use the **WiFi Hotspot** dataset from Brisbane City Council's open data initiative. The dataset is available on Blackboard. It contains data about WiFi hotspots in Brisbane in a machine-readable format. The dataset contains a list of items. Each item has: a name, a location in terms of street or suburb, and a location in terms of latitude and longitude coordinates as well as some other data. You must base your site around this dataset

## Section 3.2 Core Programming Tasks

Before you begin coding, you should develop a sitemap to help you figure out the different pages your site will require and how they are related.

## Section 3.2.1 Pages to develop

You must create the following pages.

- A **search page** with a form that allows users to search for items based a suburb (selected from a drop-down list), a name (entered into a text box), a rating, or by automatically determining the user's location using HTML5 Geolocation API.
    - Geolocation tutorials: http://www.w3schools.com/html/html5_geolocation.asp, http://diveintohtml5.info/geolocation.html
- A **sample results page,** that shows the results of a search. From the results table, users should be able to link to a more detailed screen for individual items. Since you will not be implementing the database and dynamic server-side components in stage 1, your sample results page should include a few sample results hard-coded into your HTML source file. The results must be laid-out in a meaningfully structured way, such as in a tabular format using a HTML table or as grid of tiles.
- A **sample individual item page**, with details about the item itself, as well as a list of all reviews and ratings that have been entered by users. Since you will not be implementing the database and dynamic server-side components in stage 1, you only need to create 1 sample individual item page, and it should include a few sample reviews and ratings hard-coded into your HTML source file.
- A **user registration page**, containing a form in which users are asked to enter the information required to sign up for an account. Since you will not be implementing the database and dynamic server-side components in stage 1, your registration page does not need to submit to a server-side script.

    Choose either option 1 or 2

    1. Implement JavaScript Validation
        - You must include client-side validation of the data entered using JavaScript. Your registration page should include several different HTML form elements, including text boxes and check boxes or radio items. Your code should include validation for at least numeric input, alphabetic input, email formats, and dates. You can include validation for any other types of input that you deem appropriate. Your validation should provide helpful error reporting for the user if validation fails.
    2. Implement HTML5 Validation
        - Registration Form will use HTML5 validation only. Your registration page should include several different HTML form elements, including text boxes and check boxes or radio items. Your code should include HTML5 validation for at least numeric input, alphabetic input, email formats, and dates. You can include validation for any other types of input that you deem appropriate. Your validation should provide helpful error reporting for the user if they enter invalid data.

All of your pages should include a main content area, a header, a navigation menu, and a footer. You may also include a sidebar. Your navigation menu must contain the links to the other pages on the website. Your navigation menu might be part of the header or part of the sidebar.

It is STRONGLY RECOMMENDED that you implement the requirements described above before moving on to stage 2. This will allow you to test your client side web application, replacing your hard-coded results with the Stage 2 dynamic server-side components.

### Section 3.2.2 Design

You should consider your target audience when designing both the functionality and visual style of the website. When designing your website you should also ensure that:

- Your site has a unified design with a consistent look and feel achieved via appropriate use of colours, fonts, images, etc. and a common page layout (including at least a main content area, a header, a footer and a menu).
- Your site must meet the Web Design Principles of 1) User Experience; 2) Visual Design; 3) Page Layout and 4) Standards.
- You use either a flexible or centred page design that works even for users with screen resolutions as small as 1024 pixels wide. For the mobile-ready add-on task, your design should work for screen resolutions as small as 320 pixels wide.
- Your website must follow the Web Content Accessibility Guidelines and work reasonably for users with low-bandwidth connections.

### Section 3.2.3 Implementation

Your will need to write HTML, CSS and JavaScript code to implement your web pages. Some requirements for your implementation are as follows.

- Your web pages must be strictly developed using HTML4 or HTML5 standard. Your source must be consistently indented and formatted so that it is easy for humans to read and understand. This implies that you should either write the HTML source manually or at least use a tool that produces clear simple HTML source. In other words, you should not use tools such as Microsoft Word that generate HTML that is hideous to read.
- We will validate your HTML and CSS source code using validators, and you may lose marks if your code does not validate successfully. Note that you should ensure your tags are nested properly so that your pages pass validation.
    - HTML4 validator: http://validator.w3.org
    - HTML5 validator: http://html5.validator.nu
    - CSS validator: http://jigsaw.w3.org/css-validator/
- CSS must be used to perform all styling. Your pages should be formatted using div elements and positioned using CSS or HTML5 semantic elements. Do not use tables to layout items on your pages – only use a table if you are displaying tabulated data.
- CSS code should be imported from external style sheets without inline style attributes or the use of internal style elements. All CSS code should be well formatted and commented.
- All JavaScript must be included from separate source file(s) with a .js file extension. Your JavaScript should be well structured, formatted and commented.
- A larger than normal amount of comments are required to be included in your HTML, CSS and JavaScript code to demonstrate to your tutor (project marker) that you thoroughly understand the meaning of everything that you are using.

# Section 4. Stage 2

## Section 4.1 Overview:

This stage continues the development of your website introduced in stage 1. You will implement the server side component for each of your existing pages. Pages where previously you gave only a sample will need to be dynamically generated by PHP code.

You must implement the following as PHP pages:

1. A user registration page that creates a database entry for new users. The registration page must include a password field where the user can specify their own password and some kind of unique user id (e.g. email address).
2. Your registered users need to be able to login using their user id and password.
3. Logged in users can provide a textual review and a rating for an item.
4. A search form which when submitted returns a dynamically generated results page. Users must be able to search by: item name, suburb, rating or location. The drop down list of suburbs must be populated with every unique suburb listed in your items database table. Each item has a longitude and latitude, so your search should be able to locate items within a specified distance from the user's location. Users do not need to log in to perform a search.
5. Each result on the results page should link to a dynamically generated individual item page.

Above all, your site should be kept simple! While you may add functionality in addition to the minimum requirements listed above, you will not gain any extra marks for that functionality, and you will lose marks if your implementation and source code is not clear, simple and elegant. It is better to have a well implemented simple site than a poorly implemented site with lots of functionality.

It is however very important that users can easily work out how to use your site without having to read any detailed instructions!

You must not:

1. Use any technology that is not a web standard endorsed by the W3C or that would require a browser plug-in (e.g. Java Applets, Flash or SilverLight).
2. Use a database other than MySQL or a server-side technology other than PHP.
3. Use any third party code (e.g. you cannot use Zend, Cake, Joomla, etc) other than as explicitly allowed in the stage 1 specification.

## Section 4.2 MySQL Database

You must use a MySQL database to store the data for this project. You must have one *Members* table. The *Members* table will contain one row for each user and will also contain their (hashed) password and all of their registration data. The primary key for the *Members* table will be the user id. You must have one *Items* table for storing the information from the Brisbane City Council data sets, which can be populated through the MySQL Workbench. You must have one *Reviews* table for storing user reviews of individual items. It must contain at least a reference to the item, the date posted, the user id that posted it and the text of the review and a rating.

### Section 4.2.1 Database Connection

You must use PHP Data Objects (PDO) for connecting to the MySQL database. Your connection string for connecting to the database should be present in only one place in your entire source tree (don't repeat yourself).

You must use the MySQL database provided on the CAB230 server.

## Section 4.3 Security

In order to ensure that your website is safe from malicious attack you will need to ensure that:

1. Only users that have registered with a password can log in;
2. Passwords must be stored in database after being hashed via a cryptographic hash function;
3. The website must not be vulnerable to cross site scripting attack;
4. The website must not be vulnerable to SQL injection attack;

## Section 4.4 Re-implementing Common Look and Feel

As in stage 1, you need to maintain a common look and feel across all pages in your website. This is achieved via a common page layout and development of a style sheet used across the site. For this project, the common page layout must be achieved via the use of PHP include files to factor out the common elements (Don't repeat yourself). A CSS file must be included and div elements or HTML5 semantic elements used to achieve page layout. All of your existing html pages from stage 1 need to be re-implemented to use these include files.

## Section 4.5 Add-on Programming Tasks

In addition to the core programming tasks above, you must complete **three** add-on programming tasks. Both client and server side functionality must be implemented.

### Section 4.5.1 Add-on task 1: Maps

In this add-on task, you will integrate an external mapping service with your sample search results and sample individual item page.

1. On the search results page, include a map showing markers for all search results. You should provide some way for users to click on results on the map and link to the individual item page.
2. On the individual item page, include a map showing the item.

You may choose which mapping provider you use. Popular mapping providers include OpenStreetMaps, Google, and Microsoft Bing. Note that Google and Microsoft will require you to sign up for an account; the OpenStreetMap provider is completely open-source and does not require you to sign up for any account.

- OpenStreetMaps via the Leaftlet.js API:
  - http://leafletjs.com
- Google Maps
  - General documentation: https://developers.google.com/maps/documentation/javascript/
  - Example of simple markers: https://developers.google.com/maps/documentation/javascript/examples/marker-simple
  - Example of markers with information windows: https://developers.google.com/maps/documentation/javascript/examples/infowindow-simple
- Microsoft Bing Maps AJAX Control:
  - http://msdn.microsoft.com/en-us/library/gg427610.aspx

### Section 4.5.2 Add-on task 2: Meta-data and microdata

In this add-on task, you will add microdata to the individual item page so that it displays the way you want it to when shared on social media sites and so that search engines can extract semantic geographic and review data from your site.

1. Add geographic microdata using the Place microdata schema to your individual item page so that advanced web parsers know where the item is geographically.
   a. Details of the Place microdata schema: http://schema.org/Place
   b. Some examples from Google on using Place (and other) microdata: https://support.google.com/webmasters/answer/164506?hl=en
2. Add microdata to your individual item page for each review so that advanced web parsers can recognize reviews and aggregate them into their search results.
   a. Details of the Review microdata schema: http://schema.org/Review

b. Some examples of reviews in a nice tutorial on microdata: http://diveintohtml5.info/extensibility.html

Your microdata can be tested using Google's structured data validator: http://www.google.com/webmasters/tools/richsnippets

### Section 4.5.3 Add-on task 3: Mobile-ready design

In this add-on task, you will ensure that all of the pages of your website display appropriately in a mobile web browser with restricted display space. You should use the principles of responsive design, rather than developing a separate "mobile version" of each page.

1. Use CSS @media queries to adjust the design and layout of the page based on the screen size. Your mobile version should look good at a variety of screen sizes: for example, an iPhone's screen is just 320 pixels wide (even if it's 640 physical pixels due to a high-resolution (retina) display, the operating system presents it to the CSS code as 320 pixels wide).
    a. http://mobile.smashingmagazine.com/2010/07/19/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/
2. Include metadata in your website so that it will be displayed intelligently on the user's home screen.
    a. Tutorial on configuring web site for iOS home screen: http://code.tutsplus.com/tutorials/configuring-an-iphone-web-app-with-meta-tags--mobile-2133
    b. Tutorial on configuring web site for Android home screen: https://developers.google.com/chrome/mobile/docs/installtohomescreen

## Section 4.6 Programming Principles

You must develop code that adhere to standard good programming principles. These principles include:

1. Developing valid code;
2. Separation of content and styling via CSS;
3. Including suitable metadata and titles in the HTML pages;
4. Modularizing code into logical methods, files and directories;
5. Ensuring validation of user input at client and server side and before and database interactions;
6. Ensuring that data is gracefully transferred between client and server by use of query strings and PHP pages posting back to themselves;
7. Developing readable code with respect to formatting; and
8. Including sufficient and useful comments in your code.

## Section 4.7 Report

In addition to the project you are also required to produce a report. The report should contain the following sections:

1. A Title Page that includes the student name and ID of ALL team members. The Title Page should also state whether you application works well on both Chrome and Firefox and if not you would prefer the application to be tested in Chrome or Firefox.
2. A brief Statement of Contribution from the team members (you are expected to make roughly equal contributions).
3. A Test Plan that indicates how your project is able to address the functionality required in Stage 2 both in terms of the core programming tasks and the add on tasks. Your Test Plan

should include examples of both valid and invalid user input as well as examples of output for the user performing the following tasks:

a. Accessing the home screen;
b. Registering as a new user;
c. Logging in as an existing user;
d. Logging out;
e. Adding a review;
f. Searching for an item that exists in the database;
g. Searching for an item that does not exist in the database;
h. Accessing an individual item page;
i. Attempting to use a cross site scripting attack but not being successful;
j. Attempting to use an SQL injection attack but not being successful;
k. Unregistered user not being able to log in; and
l. Operating gracefully in multiple resolutions.

You also need to provide:

m. 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).

To test the add on tasks you need to provide:

n. On the search results page: a map showing markers for all search results. (Add on #1)
o. On the individual item page: a map showing the item. (Add on #1)
p. Evidence that the geographic microdata is valid as reported by Google's structured data validator (Add on #2)
q. Evidence that the microdata is valid as reported by Google's structured data validator (Add on #2)
r. Evidence that the site icon displays on a mobile phone (iOS or Android) home screen (Add on #3)
s. Evidence that the site pages adjust well to a mobile phone screen (Add on #3)

4. A description of how your project is able to meet the Web Design Principles of 1) User Experience; 2) Visual Design; 3) Page Layout and 4) Standards. For each of the four Web Design Principles you need to provide a paragraph explaining how your project meets principles along with an example (via a screenshot or code snippit) of your project meeting that principle.
5. Link to a demo video for the project: Produce a short video demonstrating the main functionalities of your site, upload it to YouTube, and copy and paste the link to this report. You can remove the video after the assignment is marked off.

## Section 5 Restrictions

In the stage 1, you cannot use any dynamic server-side components (PHP, ASP.NET, AJAX, etc.), Web Application Framework, or JavaScript framework such as JQuery (Refer to FAQ): you should only use HTML, CSS, and JavaScript. You may not use any client-side technologies that require plugins, such as Java applets, Flash, or Silverlight.

See the FAQ at the end of this document for information about which external CSS/JavaScript frameworks you are allowed to use.

# Section 6 Getting Help

To achieve top marks in this project, you will need to supplement what you have learned in lectures and tutorials with details from textbooks or online resources. Fortunately, there are many resources available on the Internet for web development. On Blackboard, there are links to useful sites with HTML, CSS, and JavaScript. You may also check out videos posted on Lynda.Com (http://www.lynda.com).

During the workshops, the tutors are able to provide guidance on the core programming tasks. You should be prepared to explain what you are trying and why you are stuck, rather than just asking "how do you do this?".

The add-on programming tasks should be seen as more independent tasks: these are functionalities you need to do a little bit of research on how to use, and the tutors will not provide as much guidance in this area. You are welcome to ask questions of your peers either offline or on the Facebook group. As a last resort, you can email the Unit Coordinator with a question or to request a consultation meeting.

# Section 7 Submission Instructions

## Section 7.1 Stage 1

You must demonstrate your project to your Tutor during your regular tutorial. If you are working in pairs then only one of you need to demonstrate your project, however, you need to supply both members' names and student numbers to the Tutor. It is strongly recommended that both of you attend the demo and answer questions unless you two are not in the same practical class.

## Section 7.2 Stage 2

You will submit your project as a live web application on the CAB230 web server. You will receive separate instructions on how to access the web server and how to upload files including the database. These instructions will be available on Blackboard. Your website should be accessible via: http://cab230.sef.qut.edu.au:8080/nxxxxxxx,

http:// cab230.sef.qut.edu.au:8080/nxxxxxxx/index.html or

http:// cab230.sef.qut.edu.au:8080/nxxxxxxx/index.php; where nxxxxxxx is your student number.

You will also need to submit your report through Blackboard via the appropriate links. Your report should be submitted as a PDF or Word document with your student number or student numbers (if you are doing the project in pairs) as the filename.

If you are working in pairs or a team of three, one and only one of you needs to submit your files.

# Section 8 Marking

Your website should work in any modern mainstream desktop browser: Chrome, Firefox, Internet Explorer, Opera, or Safari. We will test your website at least on the most recent version of either Chrome or Firefox, but we may also use another modern browser to check compatibility. Your website must follow the WCAG guidelines for accessibility. If you do the mobile add-on task, your site should also work on both Android and iOS devices of various screen sizes.

Marks will be distributed as follows:

| Summary | Marks |
|---|---|

| | |
|---|---|
| Section A: Sample Client Side Demonstration (Due Week 7)<br>*You need to demonstrate at least four sample web pages (search page, results page, individual item page, user registration page) using HTML, CSS and JavaScript that met the functionality outlined in Section 3.2 (including input validation).* | /5 |
| Section B: Core Functionality<br>*You need to create a fully functional website that meets the requirements of sections 4.1 to 4.4.* | /20 |
| Section C: User Input Validation<br>*You need to ensure that users can only enter valid input. Response to invalid input should be graceful, provide a useful description of the error and require minimal reentry.* | /5 |
| Section D: Security<br>*Your website needs to confirm to the security guidelines as per Section 4.3.* | /5 |
| Section E: Testing<br>*Your report needs to contain a test plan as per Section 4.7* | /15 |
| Section F: Design<br>*Your report needs to contain a description of how your website design meets the Web Design Principles as per Section 4.7* | /10 |
| Section G: Programming Principles<br>*You need to write code that conforms to the programming principles outlined in Section 4.6* | /20 |
| Section H: Database<br>*Your database needs to conform the requirements outlined in Section 4.2.* | /5 |
| Section I: Add On #1 Maps<br>*You need to add the functionality of Add On Task #1 as outlined in Section 4.5.1 as well as reporting on tests outlined in Section 4.7.* | /5 |
| Section J: Add On #2 Metadata and Microdata<br>*You need to add the functionality of Add On Task #2 as outlined in Section 4.5.2 as well as reporting on tests outlined in Section 4.7.* | /5 |
| Section K: Add On #3 Mobile-ready design<br>*You need to add the functionality of Add On Task #3 as outlined in Section 4.5.3 as well as reporting on tests outlined in Section 4.7.* | /5 |
| **Total Marks** | **/100** |

# Section 9 Academic Integrity and Copyright

Except where allowed below, the work you submit for this project must be your own (or yours and your partner's if you are working in pairs).

A good way for beginners to learn about creating web pages is to examine the source of other high-quality pages available on the web and to learn how they do various things. It is unethical to plagiarise such code verbatim, but it is quite OK to examine them, understand how they work and apply the same techniques and approaches in your website. Different web pages can provide different things; one website might illustrate the use of a font or color scheme that you find effective, while another might use CSS to achieve a page layout that you like. Others still might use JavaScript to achieve dynamic effects, such as drop down menus. Other websites of the same genre might provide some ideas regarding content and functionality.

You may not use any external frameworks, libraries, or templates in developing your website, except as indicated in the FAQ below. You are not permitted to use the core jQuery library, jQuery UI, jQuery mobile, or any jQuery plugins (Refer to FAQ).

If you want to use other people's images or fonts in your website, you may do so provided you abide by all copyright restrictions. For images, this means you should only use images that you make yourself, or images that are either in the public domain or licensed under terms that allow reuse (e.g., certain Creative Commons licenses). For fonts, this means you should only use fonts that are licensed for web embedding, or web-font services such as Google Fonts. Note that when using properly licensed images in building your own website, it is generally considered polite to locally host the image to avoid placing bandwidth load on the original image provider.

We may use similarity-detecting tools to help identify submissions that share code.

Further information on QUT's policy to plagiarism is available at http://www.mopp.qut.edu.au/C/C_05_03.jsp. Information on penalties is available at: http://www.mopp.qut.edu.au/E/E_08_01.jsp#E_08_01.08.mdoc.

## Section 10 Late Assignments

As per the QUT's policies, late assignments without approved extensions will receive a mark of 0. You may apply for extensions before the due date using the procedure described at http://www.student.qut.edu.au/studying/assessment/late-assignments-and-extensions .

## Section 11 FAQ

1. Can I use these beautiful HTML / CSS templates I found on the web?
   No, you must code your own HTML and CSS. While many organizations do use template systems, since this is a first web development unit it is important for you to learn how to write this code on your own.
2. Can I use AngularJS / YUI / Bootstrap / <insert favourite HTML/CSS/JavaScript framework here>?
   No. Web development frameworks are a very dynamic area, and it is very likely that the frameworks that are popular today will be abandoned or obsolete in just a couple of years. As noted above, it is important for you to learn how to write your own code, and once you have those skills, you will be able to work with appropriate frameworks when the need arises. (If you have a very good reason to use a framework to do something super awesome in a way that doesn't defeat the purpose of the assignment, you can propose it to me and I will consider it.) For the maps add-on task, you may include the JavaScript and CSS frameworks from the mapping provider as required.
3. Can I use Sass or LESS when I write my CSS code?
   Yes, but it is recommended that you only do so if you have previous experience with CSS and can explain the problems with CSS that Sass or LESS aim to solve. Be advised that we will only mark your CSS code, so you should ensure that whatever CSS code is generated by Sass or LESS is human-readable and high-quality. (If you did not understand this question or answer, feel free to ignore it.)
4. Can I use jQuery?
   No, you may not use the core jQuery library jQuery UI, jQuery mobile, or any jQuery plugins. we are generally trying to avoid the framework-of-the-month approach in CAB230, jQuery is so widespread that it is a de facto standard. However, if you have been using jQuery for a while, you may be interested to know that newer browsers have much better JavaScript support, making jQuery less of a necessity. For example, the new document.querySelector(x) JavaScript function does much the same as jQuery's $(x) function and is fairly widely supported.

5. Can I use reset.css / normalize.css?
   You may use a CSS reset stylesheet if you want to have more control over the base formatting of CSS objects. See http://www.cssreset.com for some possibilities.
6. Can I use web font's services like Google Fonts?
   Yes. Please see the note above regarding copyright issues.
7. Can I use sIFR for displaying fonts?
   No, because sIFR requires Flash and you may not use Flash.
8. Can I use an icon font like Font Awesome?
   Yes.
9. Can I use a different dataset?
   No.
10. Which editor should I use?
    See Blackboard -> Learning Resources -> Textbook(s) and Online Resources for a discussion about text editors. Note in particular that you may use Adobe Dreamweaver as a text editor, but you should avoid using its WYSISYG / GUI features as they may not output readable code.