CAB230 Web Computing

Project: Implementation of a simple data-driven web site

Worth 50%
Due: Monday 25/5/2015
Teams: The project can be done individually or in pairs. It is preferable that you form team with people in the same practical/workshop class.

Change History

Date	Change Detail	Version
08/03/2015	Stage 1 Implementation: Your web pages must be strictly developed using HTML5 standard	1.1
12/03/2015	Addition: allowing for the use of personal dataset	1.2
17/03/2015	Addition: JavaScript option	1.3

Synopsis

This project will require you to design and implementation of a simple data-driven web site 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. This project consists of several core programming tasks that all submissions must include, some add-on tasks that you may choose from, and the data-driven web functionality using PHP and MySQL; you must submit this project to the CAB230 web server by the due date.

Stage 1

Overview

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.

It is preferred that you will use data from Brisbane City Council's open data project, where they have posted a variety of data about public facilities in Brisbane in machine-readable formats. There are many datasets on their website (http://data.brisbane.qld.gov.au/index.php/datasets/), of which the following are most appropriate for use in this project. Each dataset below 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. You should pick one of the following datasets to base your site around:

- Parks: http://data.brisbane.qld.gov.au/index.php/dataset/parks-2/
- Playgrounds: http://data.brisbane.qld.gov.au/index.php/dataset/playgrounds/
- Public art: http://data.brisbane.qld.gov.au/index.php/dataset/public-art/
- Dog parks: http://data.brisbane.qld.gov.au/index.php/dataset/dog-parks/
- WiFi hot spots: http://data.brisbane.qld.gov.au/index.php/dataset/wifi-hot-spots/
- Tennis courts: http://data.brisbane.qld.gov.au/index.php/dataset/tennis-courts/
- Public toilets: http://data.brisbane.qld.gov.au/index.php/dataset/public-toilets/

Alternate Datasets

If you wish to use an alternative dataset, you must ensure your dataset complies with all the requirements of this specification and the marking criteria (this is your responsibility and any oversight could be costly). You must also ensure the dataset is clean and provide the dataset to the unit coordinator when requesting permission for its use.

In combination with the stage 2, your site will allow registered users to provide a textual review and a rating (numerical, thumbs up/down, stars, whatever you think is best) for an item. All users (registered or unregistered) may search for items based on suburb, name, rating, or proximity to their current latitude and longitude. Users will be able to register for accounts.

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.

Pages to develop

You must create the following pages.

- A search 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 showing the results of a search in a tabular format. 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.
- 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.

    JavaScript Options: Choose either option 1 or 2

    1. Implement JavaScript Validation
        - However, 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, Plus a JavaScript Component
        - 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 validation fails.
- In order to assess your JavaScript skills you must also implement a JavaScript component of your choice; such as the real-time spectrogram demonstrated in the lecture. You may also develop a Slide Show component that contains images that link to your items dataset (*e.g. the BCC dataset you will need to source these images and store them appropriately for access*). The sample results page showing the results of a search in a tabular format, should then also display your Slide Show component.

All of your pages should include a header, navigation menu, and footer. You may also choose have a sidebar if you think it is appropriate. Your navigation menu must contain at least 4 links, although some of them may be broken links because you will not have implemented all of your website in this stage. 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 against the stage 1 marking criteria sheet, before removing your hard-coded results with the stage 2 dynamic server-side components.

Design
You should consider your target audience when designing both the functionality and visual style of the web site. 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 header, footer, and menu).
- You use either a flexible or centred page design that works even for users with screen resolutions as small as 1024 pixels wide. If you complete the mobile add-on task, your design must work for smaller display sizes as well.
- Your web site follows the WCAG guidelines and works reasonably for users with low-bandwidth connections.

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

- Your web pages must be strictly developed using 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. 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 with only minimal use of inline style attributes and no 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.

Add-on Programming Tasks

In addition to the core programming tasks above, you must complete one or more add-on programming tasks. The number of add-on tasks you must complete depends on which unit you are enrolled in and whether you work individually or in pairs:

- Individual or pairs of CAB230 students must complete two add-on tasks.

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 Bin. 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 Leftlet.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

Add-on task 2: Meta-data and microdata

In this add-on task, you will add meta-data and microdata to the sample 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 metadata fields in the header for Facebook's Open Graph protocol, as well as Twitter Cards. Here are some resources to help you:
    a. Description of the Open Graph protocol: http://ogp.me
    b. Facebook's Open Graph debugger: https://developers.facebook.com/tools/debug/
    c. Description of Twitter Cards: https://dev.twitter.com/docs/cards/getting-started
    d. Twitter's Cards validator: https://dev.twitter.com/docs/cards/validation/validator

   e. Google's structured data validator:
      http://www.google.com/webmasters/tools/richsnippets

2. Add geographic microdata using the Place microdata schema to your sample 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
3. Add microdata to your sample 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

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

Stage 2

Overview:

This stage continues the development of your web site 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. Any Add-on tasks that you took on for stage 1 must be continued in this stage 2 by implementing the corresponding server side functionality.

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. 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.

Challenge Task:

6. Implement an admin page to allow a csv file to be uploaded to the server and its data stored into your database. The file itself should not be saved on the server, just its data. (You need to research how this can be done in PHP).

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!

In addition to these functional requirements, you should carefully review the assessment criteria to learn how these requirements should be meet in order to score full marks.

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.

MySQL Database

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

If you wish to have any additional tables or wish to alter any of the above requirements, you must first obtain permission from the Lecturer (Jinglan Zhang).

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.

Re-implementing Common Look and Feel

As previously, you need to maintain a common look and feel across all pages in your web site. 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 used to achieve page layout. All of your existing pages need to be re-implemented to use these include files.

Restrictions

In the stage 1, you cannot use any dynamic server-side components (PHP, ASP.NET, AJAX, etc.), Web Application Framework, or 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.

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, I have posted links to useful sites with HTML, CSS, and JavaScript.

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 me (jinglan.zhang@qut.edu.au) with a question or to request a consultation meeting.

Submission Instructions

You will submit your project as a live web page on the CAB230 web server. You will receive separate instructions on how to access the web server and how to upload files. Stage 1 is not marked separately of Stage 2; it is strongly recommended you follow this step and test stage 1.

The submission instructions file will be available on Blackboard.

If you are working in pairs, only one of you needs to submit your files.

You should include in your folder a file called info.txt, which contains your name and student number (and that of your partner if you are working in pairs), as well as indicates which add-on task(s) you have done.

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 Chrome, 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.

Please see the separate project assessment criteria sheet for the detailed marking criteria.

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 web site. Different web pages can provide different things; one web site might illustrate the use of a font or colour 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 web sites of the same genre might provide some ideas regarding content and functionality.

It is generally acceptable to use small snippets of code you find on the Internet—not more than 5 lines—without attribution. You should not use larger pieces of other people's code in this project.

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 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.

Late Assignments

As per the QUT's new 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.

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. Ware 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?
   You can propose a different dataset to me and I will let you know if you can use it.
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.