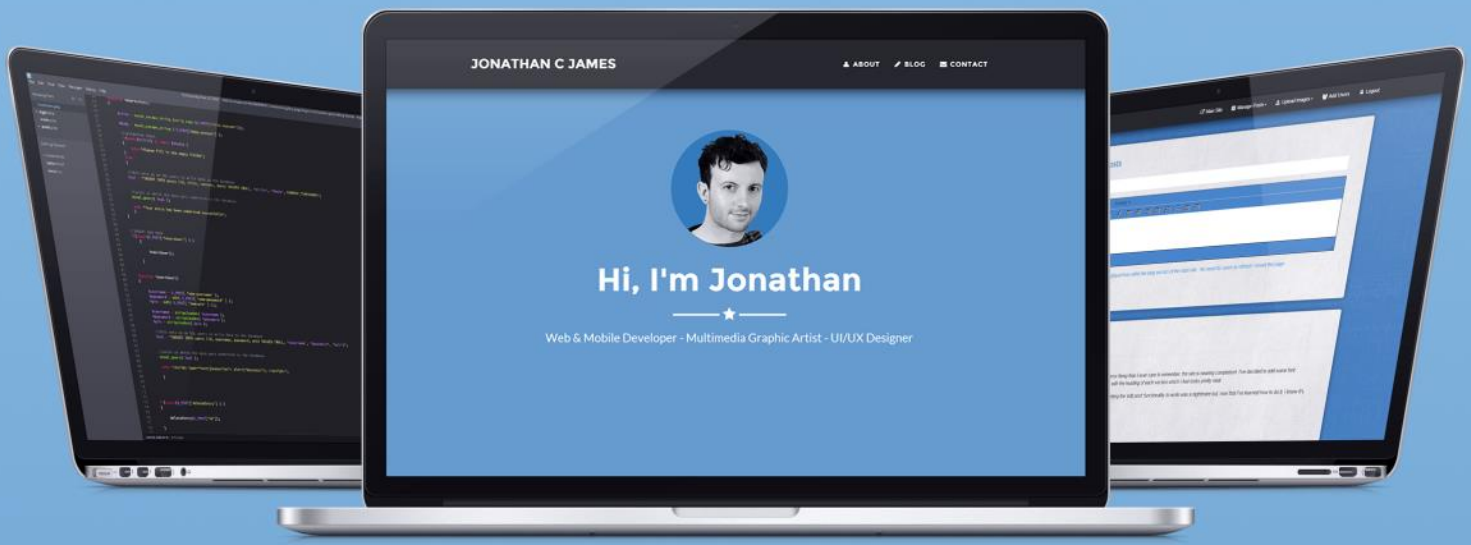


Personal Portfolio

CMS

{ Read Me }

Website URL
http://ct5017-14t.studentsites.glos.ac.uk/

Final Version
18

CMS Login Account
Username: zayd
Password: webdev
PIN: 2014

CMS Phone Login
Number: 07790 382 989

Email Login Information
Username: jcjwebdev@yahoo.com
Password: jcj.web.dev

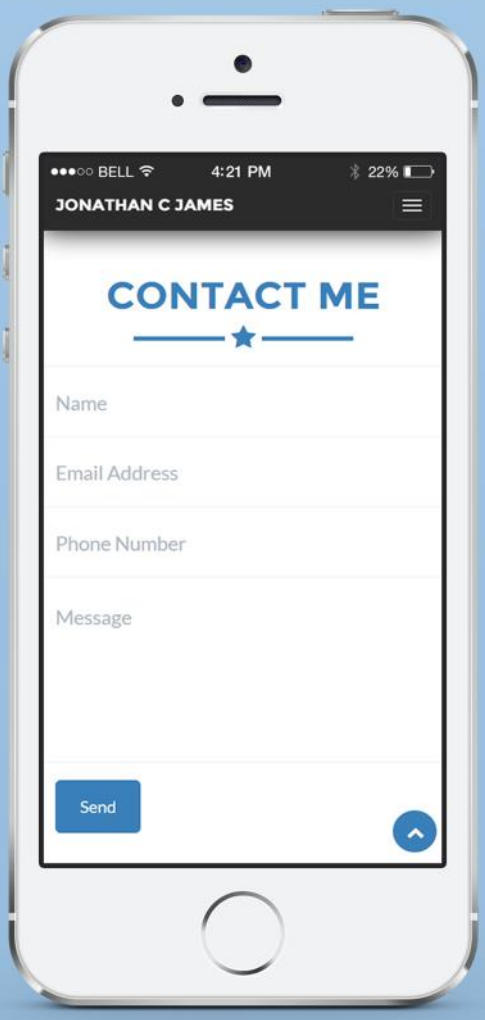
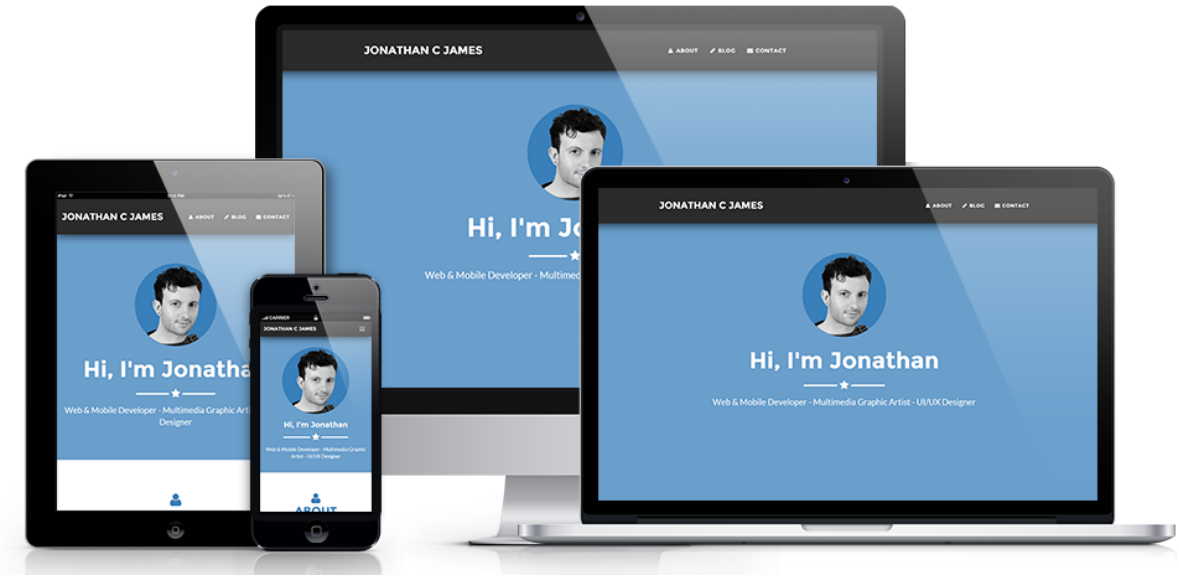


Table of Contents Here

Introduction

The site is responsive on all devices through use of the design framework Bootstrap 3.



Bootstrap works by pre-loading CSS classes and media-queries into the document object model that can be referenced by the developer in the HTML mark-up. This allows for fast, efficient responsive web design.

The site was built in versions so as to protect each progress stage during development.

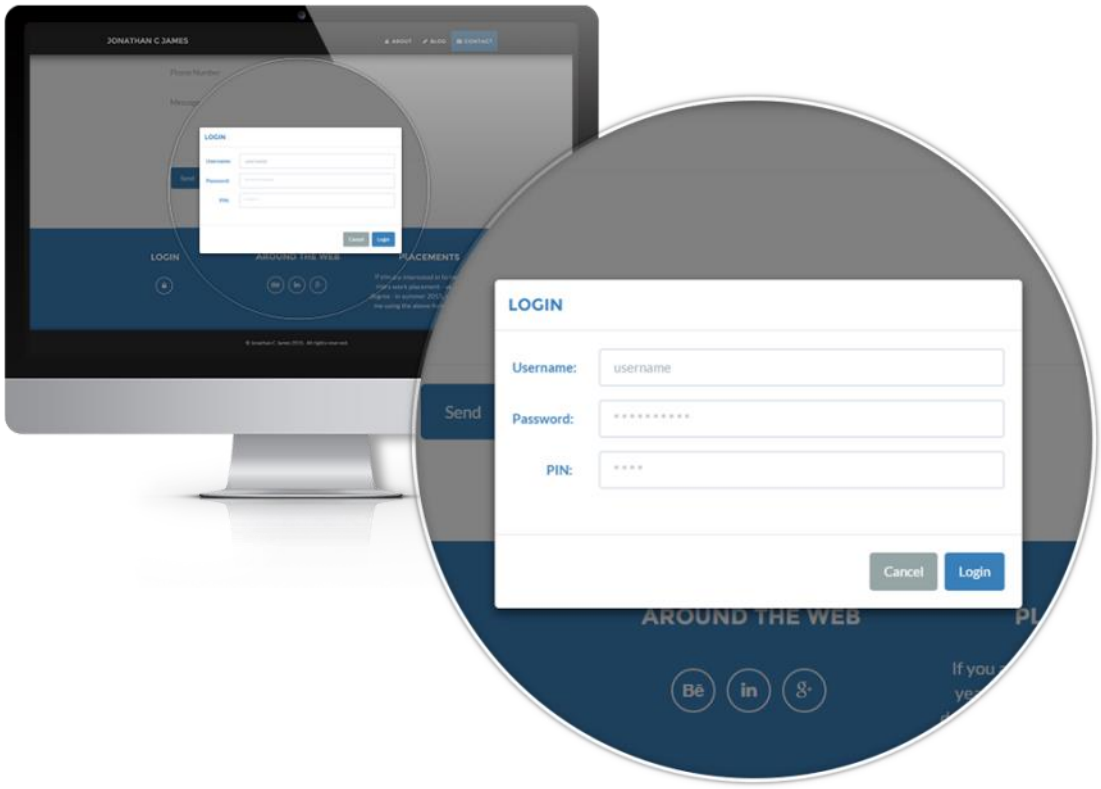
- version 1 (working blog)
- version 2 (complete css and structure)
- version 3 (home page login - working)
- version 4 (home page login - styled)
- version 5 (home page login - validated)
- version 6 (custom denided access pages)
- version 7 (cms - styled)
- version 8 (cms - edit post unstyled)
- version 9 (cms - edit post styled)
- version 10 (cms - edit post modal)
- version 11 (working contact form)
- version 12 (dial pad login)
- version 13 (dial pad login - integrated)
- version 14 (gallery page)
- version 15 (gallery page - integrated)
- version 16 (slimmed down, commented)
- version 17 (custom TinyMCE)
- version 18 (debugging)

As developmental milestones are reached, a copy of the site is taken, with the original being kept as a working model whilst further development is carried out.

Innovative Login

The site uses a multi-phase login system, akin to those used in conjunction with online banking systems.

Making use of the aforementioned Bootstrap 3 framework, the front-end login process begins with a HTML5 form contained within a modal.



The information inserted into this form is retrieved and handled using a PHP script that utilises the $_POST method; this method allows information to be sent anonymously from an HTML form to a PHP script for processing - in this case, login.php.

Within login.php, the submitted form data is extracted from the $_POST method and stored in variables.

```php
// Define $username and $password ($_POS
$username = $_POST[ 'username' ];
$password = md5( $_POST[ 'password' ] );
$pin = md5( $_POST[ 'pin' ] );
$submit = $_POST[ 'submit' ];

if ( isset( $submit ) )
{
```

```php
//SQL query to fetch information of registerd users and finds user match.
$query = mysql_query( "SELECT * FROM usersWHERE password='$password'
AND username='$username' AND pin='$pin'", $connection );

$rows = mysql_num_rows( $query );

//does the query return a true result?  i.e. the username and password matched
if ( $rows == 1 ) {

    //Initializing Session
    $_SESSION[ 'login_user' ] = $username;
```

The data within these variables is then compared with the relative table data. If they match, the user is logged in.

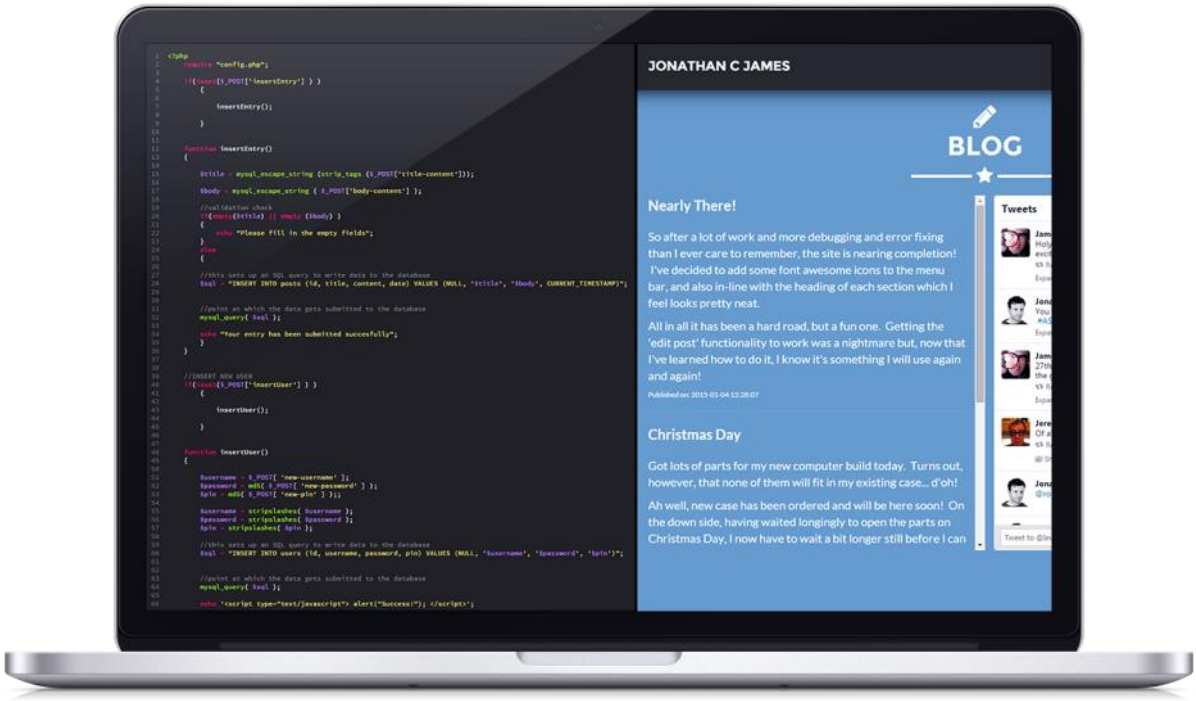
The existing login form has extra security with the addition of a 4-digit PIN number necessity for each user.

```
// Define $username and $password ($_POST
$username = $_POST[ 'username' ];
$password = md5( $_POST[ 'password' ] );
$pin = md5( $_POST[ 'pin' ] );
$submit = $_POST[ 'submit' ];
```

Additional security is provided through a jQuery dial-pad. Users must enter a secret number in order to access the main content management system (CMS).

Insert Posts

The insertEntry() function within functions.php also uses the $_POST method to gather data from title input and content textarea fields on forum submission.



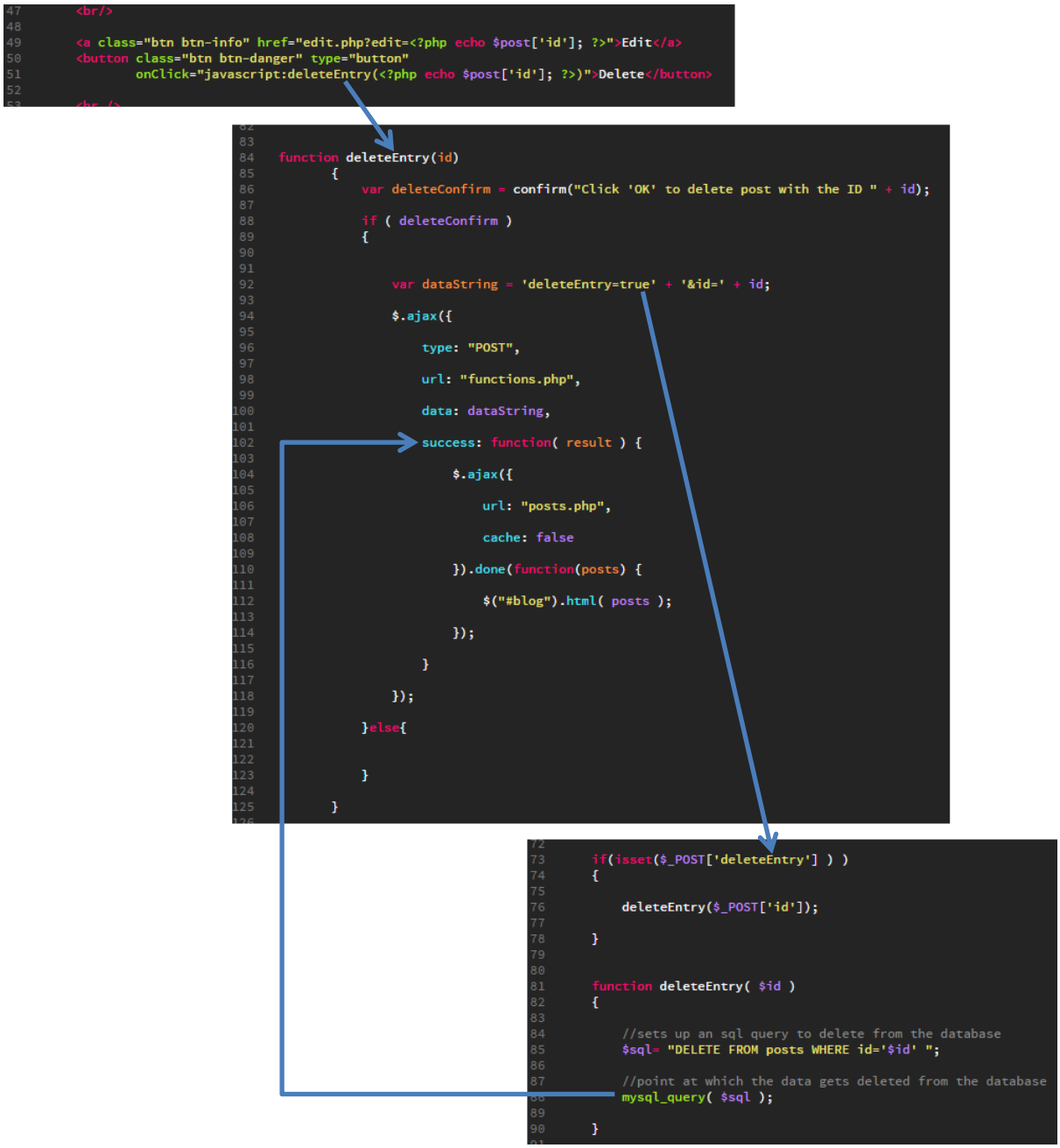
However, whereas login.php compares $_POST values against existing data, insertEntry() uses a mysql_querry to insert the values into a database (line 28).

```php
<?php
    require "config.php";

    if(isset($_POST['insertEntry'] ) )
        {

            insertEntry();

        }


    function insertEntry()
    {

        $title = mysql_escape_string (strip_tags ($_POST['title-content']));

        $body = mysql_escape_string ( $_POST['body-content'] );

        //validation check
        if(empty($title) || empty ($body) )
        {
            echo "Please fill in the empty fields";
        }
        else
        {

        //this sets up an SQL query to write data to the database
        $sql = "INSERT INTO posts (id, title, content, date) VALUES (NULL, '$title', '$body', CURRENT_TIMESTAMP)";


        //point at which the data gets submitted to the database
        mysql_query( $sql );

        echo "Your entry has been submitted succesfully";
        }
    }
```

Delete Posts

The deleteEntry() process begins with the relevant $post['id'] being pass from a button click to an AJAX request. This request then sends and receives data with the deleteEntry() function within functions.php.

A mysql_querry deletes the $post['id'] associated values from the database. This delete data is then sent back through AJAX, resulting the #blog div being updated.



Edit Posts

The ability to edit posts is thanks to the PHP $_GET method. Just like $_POST, this method creates an array of key value pairs. The difference is $_GET method passes its array variables through the URL.

The id of the post to be edited is passed via a button click:

```
<a class="btn btn-info" href="edit.php?edit=<?php echo $post['id']; ?>">Edit</a>
```

The edit.php script then pre-loads the relevant $post['id'] data into a HTML textarea by inserting the results of a mysql_query into a variable as a mysql_fetch_array.

Once the pre-populated data has been amended within a form, the $_POST method then inserts the new data back into the relevant row using the $post['id'].

```
<?php

    //start a session and identify the user
    session_start();

    include "../check-logged-in.php";

    if( isset( $username ) )
    {
        require "config.php";

        if( isset( $_GET['edit']) ) ---------- 1
        {

            $id = $_GET['edit'];
            $res = mysql_query("SELECT * FROM posts WHERE id='$id'");          2
            $post = mysql_fetch_array($res);

        }

        if( isset($_POST['update-post'])) ---------- 3
        {
            $newTitleContent = mysql_escape_string (strip_tags ($_POST['new-title-content']));
            $newBodyContent = mysql_escape_string ( $_POST['new-body-content'] );          4
            $id = $_POST['id'];

    5       $sql = "UPDATE posts SET title='$newTitleContent', content='$newBodyContent' WHERE id='$id'";
            $res = mysql_query($sql) or die("Could not update".mysql_error());
            echo "<meta http-equiv='refresh' content='0;url=index.php'>";
        }

?>
```

New users can be registered through the CMS via the InsertUser() function. This function uses the $_POST method, much like the insertEntry() function.

Areas for Improvement

End Session

As it stands, logging out from the site ends the session but closing the browser does not. This feature would be easily integrated to add additional security to the login area.

Dial-Pad Login Phase

This is currently handled using jQuery. However, replicating this feature using PHP in with a MySQL database would keep sensitive login information hidden from the browser.

References

1. Spurlock, J, 2013. *Responsive Web Development Bootstrap*. 1st ed. CA: O'Reilly Media, Inc.
2. Nixon, R, 2012. *Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites*. 2nd ed. CA: O'Reilly Media, Inc.
3. Rutter, J, 2011. *Smashing jQuery*. 1st ed. West Sussex: Wiley.
4. How to Prevent a Directory Listing Your Website with .htaccess. 2015, Online, Available at: http://www.thesitewizard.com/apache/prevent-directory-listing-htaccess.shtml
5. Jonathan James School of Computing and Technology Student Potfolios. 2015, Online, Available at: http://ct.glos.ac.uk/students/jonathanjames/
6. CSS – Bootstrap. 2015, Online, Available at: http://getbootstrap.com/css/
7. Components – Bootstrap. 2015, Online, Available at: http://getbootstrap.com/components/
8. JavaScript – Bootstrap. 2015, Online, Available at: http://getbootstrap.com/javascript/
9. jQuery API Documentation. 2015, Online, Available at: http://api.jquery.com/
10. PHP: PHP Manual. 2015, Online, Available at: http://php.net/manual/en/
11. Brinzarea, B. (2009) *AJAX and PHP building modern web applications, 2nd ed*. Birmingham, UK : Packt Pub.
12. Dubois, P. (2013) *MySQL 5th ed*. Sebastopol, Indianapolis: Addison-Wesley.
13. Merkel, D. (2010) *Expert PHP 5 tools: proven enterprise development tools and best practices for designing, coding, testing, and deploying PHP applications*. Birmingham, UK: Packt.
14. Ullman, L.E. (2011) *Effortless E-commerce with PHP and MySQL*. Berkeley, CA : New Riders.
15. MySQL :: MySQL Documentation. 2015, Online, Available at: http://dev.mysql.com/doc

Product Criteria Grid

Name: Jonathan C James **Student number:** s1308501

Grade	Content	Claim
To achieve <30	Some requirements met, but very limited and not recoverable. Copyright violation. No comments in any code.	
To achieve <40	Deliverables partially complete, e.g. not all components attempted. No CMS. Scripting patterns inconsistent. Poor quality comments in code. Limited weekly progress report. Limited supporting documentation. No references.	
To achieve 40+	All components attempted. Basic front-end of web site. Not responsive. Some evidence of a basic CMS. Basic understanding of server-side scripting. Some comments in code. Some evidence of weekly progress report. Supporting documentation submitted, with references.	
To achieve 50+	All components functional. Database communications robust. Suitable file and code structure. Front-end of web site responsive. Good combination of server-side and client-side technologies. Good comments in code. Clear progress report. Evaluation contains evidence of criteria.	
To achieve 60+	Original and intuitive ideas. Front-end and back-end of web site responsive. Very good use of web technologies to provide an innovative design. Clear evidence of progress, showing evidence of problem-solving along the way. Clear comments indicating understanding of code. Evidence of critical reflection in evaluation.	
To achieve 70+	Excellent progress reports and completed product. High quality, aesthetically pleasing, interactive interface designs. Explored and integrated APIs. Included suggestions for further work. Excellent level of commenting ready for auto-documentation. Extensive and detailed evaluation.