0

g

Adrian Forrester

s1902834@glos.ac.uk

Introduction to Web Development

CT4009 Assignment 002

Contents

[1. Introduction 3](#_Toc73057789)

[1.1. Purpose 3](#_Toc73057790)

[1.2. Scope 3](#_Toc73057791)

[1.3. Credentials for Accessing Application 3](#_Toc73057792)

[2. Implementation of Functionality 3](#_Toc73057793)

[2.1. Successfully Implemented 3](#_Toc73057794)

[2.2. Outstanding Functionality 4](#_Toc73057795)

[3. Examples of HTML, PHP, CSS, JavaScript, MySQL, and APIs. 4](#_Toc73057796)

[3.1. HTML 4](#_Toc73057797)

[3.2. PHP 4](#_Toc73057798)

[3.3. CSS 5](#_Toc73057799)

[3.4. JavaScript 6](#_Toc73057800)

[3.5. MySQL 7](#_Toc73057801)

[3.6. APIs 7](#_Toc73057802)

[4. Usage Instructions 8](#_Toc73057803)

[5. Testing 11](#_Toc73057804)

[6. Security 11](#_Toc73057805)

[7. Business Values of Different Features 11](#_Toc73057806)

[8. References 11](#_Toc73057807)

[9. Appendices 12](#_Toc73057808)

[9.1. Screenshots of Implemented Pages 12](#_Toc73057809)

[9.2. Source Code 14](#_Toc73057810)

# Introduction

## Purpose

The application has been developed in response to the request to have a web-based application, offering a method for creating a Social Media/Message Board functionality.

## Scope

The web application was expected to include functionality for a User Registrations process, Login system, providing authenticated users the ability to create posts or articles, which could include text, images, and Google Maps locations. In addition to this the users should be able to create an event, allowing users generate an invite and identify those who have responded to the invite.

The is a further request to provide administrator users with the ability to see the number of posts that had been created by users for a given period.

This was to be written using HTML, CSS, JavaScript, PHP and using MySQL as a Data Store.

## Credentials for Accessing Application

|  |  |  |
| --- | --- | --- |
| **Website Hosting Details** | | |
| **Domain name:** | | s1902834-ct4009.uogs.co.uk |
| **Plesk login details:** | |  |
|  | Username: | s1902834 |
|  | Password: | !7fRI3alWD4&fo\_D |
| **FTP login details:** | |  |
|  | SFTP login: | s1902834-ct4009 |
|  | SFTP Password: | 5xskH3?1 |

|  |  |  |
| --- | --- | --- |
| **Application Users** | | |
| **User:** | |  |
|  | Username: | guest |
|  | Password | haveyoutriedguest |
| **Administrator:** | |  |
|  | Username: | admin |
|  | Password: | Adm1nPa55w0rd |

# Implementation of Functionality

## Successfully Implemented

The below is a list of the specific functionalities of pages that have achieved the requirements as outlined:

|  |  |  |
| --- | --- | --- |
| **Function** | **Notes on Functionality** | **Location/URL of Page** |
| User Registration |  | php/registration.php |
| User Login |  | php/login.php |
| Creation of Post | Requires Authentication | index.php |
| Administration Login |  | php/login.php |
| Administration Reports | Requires Admin Authentication | php/admin.php |

## Outstanding Functionality

|  |  |  |
| --- | --- | --- |
| **Functions** | **Notes on Functionality** | **Location/URL of Page** |
| Send Invites | Functionality not achieved. | php/post.php |
| View Invite Status | Functionality not achieved. | php/post.php |

The operation of Sending Invites and handling responses was one that could be implemented several ways. It was intended that the creation of an “Event” from the index.php to send an email to a supplied address, with the email containing a URL which could then be opened to trigger an event, marking the attendee as accepting the invite. The idea was that a user would not necessarily be registered as a user to accept an invite, but registered users could see the accepted invitee list.

# Examples of HTML, PHP, CSS, JavaScript, MySQL, and APIs.

## HTML

All the pages for the application have been written with a focus on PHP, however HTML is interspersed throughout to retain some of the core formatting of pages. This is largely with respect to tables and form elements as can be seen below:

<form action="" method="post" name="login">  
 <input id=logininput type="text" name="username" placeholder="Username" required /></br>  
 <input id=logininput type="password" name="password" placeholder="Password" required /></br>  
 <input name="submit" type="submit" value="Login" />  
</form>

This shows the formation of the Login screen form (login.php), with a similar approach taken with the Registration page form (registration.php).

## PHP

The use of PHP has helped to approach some of the required operations in a more systematic fashion. Especially regarding the Database handling – Including connector events as well as reading and writing of data (Hughes, 2014). Any distinct functionality that needed to allow user intervention had been written in PHP where possible as it handled one-pass functions.

This included the submission of form data, which then triggers the database write events as below:

if (isset($\_POST['subject'])){  
 $articleSubject=$\_POST['subject'];  
 $articleContent=$\_POST['content'];  
 $articleCategory=$\_POST['category'];  
 $articleUser=$\_SESSION['authid'];  
 $articleLat=$\_POST['mapLat'];  
 $articleLng=$\_POST['mapLng'];  
 $articleFile = $\_FILES['image']['name'];  
 $fileTarget = "image/".basename($image);  
  
 if (move\_uploaded\_file($\_FILES['image']['tmp\_name'], $fileTarget)) {  
 $msg = "Image uploaded successfully";

} else {  
 $msg = "Failed to upload image";  
 }

$query = "INSERT INTO `article` (articleSubject, articleContent, articleDate, articleCategory,   
 articleUser, articleLat, articleLng, articleFile)  
 VALUES('$articleSubject','$articleContent',NOW(),'$articleCategory',  
 '$articleUser','$articleLat','$articleLng', '$articleFile')";  
 $formresult = mysqli\_query($con,$query);  
 if (move\_uploaded\_file($tempname, $folder)){  
 $msg = "Image uploaded successfully";  
 } else {  
 $msg = "Failed to upload image";  
}

Whereby the form fields are read as part of the form Post event, each field is allocated a variable, which is then passed to the $query variable, allowing for the definition of the MySQL/MariaDB Insert Statement. The $formresult operation then calls the MySQL/MariaDB Connector and processes the completed query.

Additionally, image handling is managed using PHP, where the file is initially uploaded to a globally accessible location (/tmp on the server) it is then moved to the defined location for $filetarget, which is the images directory, under the local path of the application.

## CSS

As the name implies, Cascading Style Sheet allow for the manipulation of website elements, allowing for more granular customisation of website elements, along with allowing for more achievable uniformity using Classes, Divs or element IDs, which allows common elements to be easily identified, categorised, and grouped; so that styling rules can be applied either globally or per element.

An example of this can be seen below, where the registration form and login form are both styled in the same way. To simplify the styling the Divs were referenced as a single CSS function, with both Divs specified (Shenoy, 2014):

.loginform, .registrationform{  
 margin-left: 25%;  
 margin-right: 25%;  
 width: auto;  
 background-color: white;  
 color: black;  
 border-width: 1px;  
 border-style: solid;  
 border-color: black;  
 border-radius: 6px;  
 font-family: 'Roboto', sans-serif;  
}

It can also be used to import fonts, in this instance Roboto from the Google Fonts service is imported and applied globally, using the guidance provided by Google (*Roboto - Google Fonts*):

@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');  
\* {  
 font-family: 'Roboto', sans-serif;  
}

And finally, using nth-child element calls to create exceptions to rules allows for a simple method for making changes without declaring additional Divs, Classes or Ids:

header table td{

border-right: 1px solid black;

border-collapse: collapse;

padding: 0px 10px 0px 10px;

}

header table td:nth-child(1), td:nth-child(5){  
 border-right: none;  
}

This allows for the discarding of an earlier operation, applying a border-right to a table, for the first and fifth element in the table, this is used as a styling cue in the header, allowing for a ‘pipe’ between each option.

## JavaScript

JavaScript is used for interactive elements of the website that may require more than One-Pass processing. This includes the call to enable Google Maps as well as the passing of data for the Map interactions:

function postMap(){  
 var latitude = +(document.getElementById("latitude").innerHTML);  
 var longitude = +(document.getElementById("longitude").innerHTML);  
 var latLng = new google.maps.LatLng(latitude, longitude);  
  
 map = new google.maps.Map(document.getElementById("googleMap2"), {  
 center: latLng,  
 zoom: 17  
 });  
 var marker = new google.maps.Marker({  
 position: latLng,  
 map: map  
 });  
 map.setOptions({draggable: false});  
}

The above function is called as part of the code that interacts with Google Maps, defining the required data to present a location on the map from a stored Latitude and Longitude value from the MySQL/MariaDB database. The selected values are passed to latitude and longitude variables, combined as a single string, then used to centre and define a marker location.

Another use case is for the manipulation of CSS. The below code will set elements to be made visible or hidden, using a link on the page:

function createPost() {  
 var articlediv = document.getElementById('article');  
 var articlelnk = document.getElementById('articlelnk');  
 if (articlediv.style.display == 'none' || articlelnk.style.display == 'block'){  
 articlediv.style.display = 'block';  
 articlelnk.style.display = 'none';  
 } else {  
 articlediv.style.display = 'none';  
 articlelnk.style.display = 'block';  
 }  
}

Setting the Select Option to 2 (Event) when creating an Article, will set the CSS Elements for the ‘article’, which includes the Post Creation form to visible, whilst setting ‘articlelnk’ which triggered this event, to be hidden.

## MySQL

The use of MySQL is for the storage and retrieval of user data, as it is stored in a MySQL/MariaDB database, this ensures that data is not simply session based, and can be recalled from any device accessing the application. The use of PHP to handle these data interactions allowed for the data to be handled in a way like direct access to the database as the syntax needed little adaptation:

$levelquery = "SELECT userLevel from `users` where userName='$username'";

The above query is used for validating which class of user is logging in, returning the distinct userLevel value depending on whether the user has Administrator Access or simply User Access. Any result returned from a query can be easily defined as a variable utilised as needed.

## APIs

As identified previously, Google Maps is the key API utilised within this application and was easily configured using Google’s Maps Version 3 API (Google, 2018), where an API key along with the additional variables specified in the map.js file for either creation of a new Marker or displaying a previously stored one:

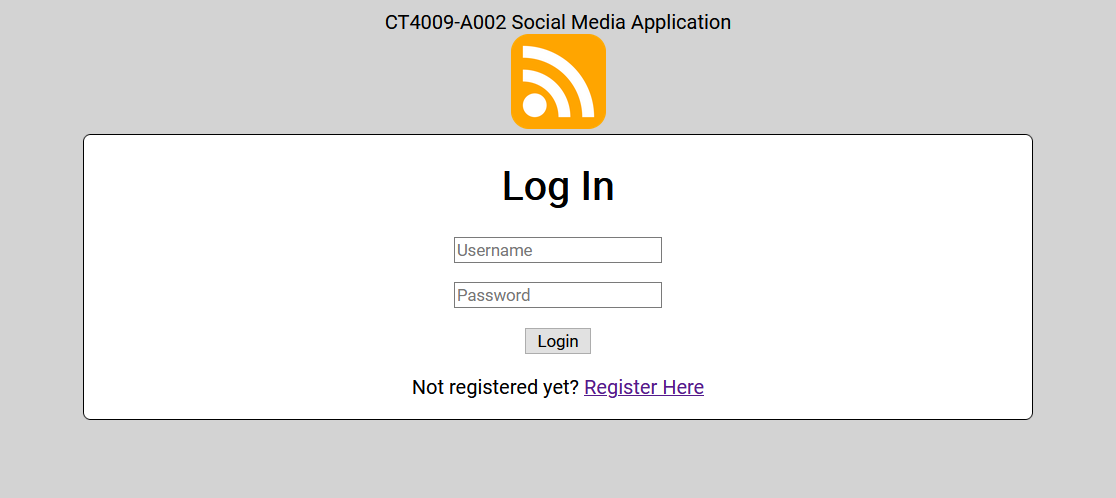
*google*.*maps*.*event*.addListener(*map*,'click',function(event) {  
 placeMarker(*event*.*latLng*);  
 *document*.getElementById("latitude").*value* = *event*.*latLng*.lat();  
 *document*.getElementById("longtitude").*value* = *event*.*latLng*.lng();  
 });  
}

The above illustrates the minimal code required to easily define a marker location when clicking on the Google Map as presented.

# Usage Instructions

You can access the Application be opening the following URL in a browser.

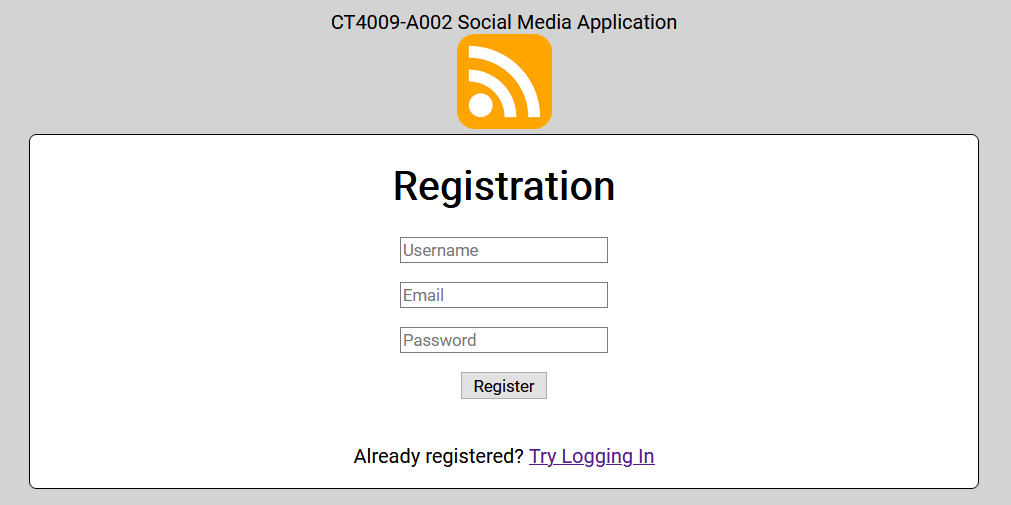
On loading you will be presented with the Login Page:



An already registered user may enter their Username and Password to Login to the application. New users should visit the Register Here link to enrol.

**For New Users:**

The registration page will allow users to enter a Username, Email and Password. Once entered, clicking Register will create the new User Account:



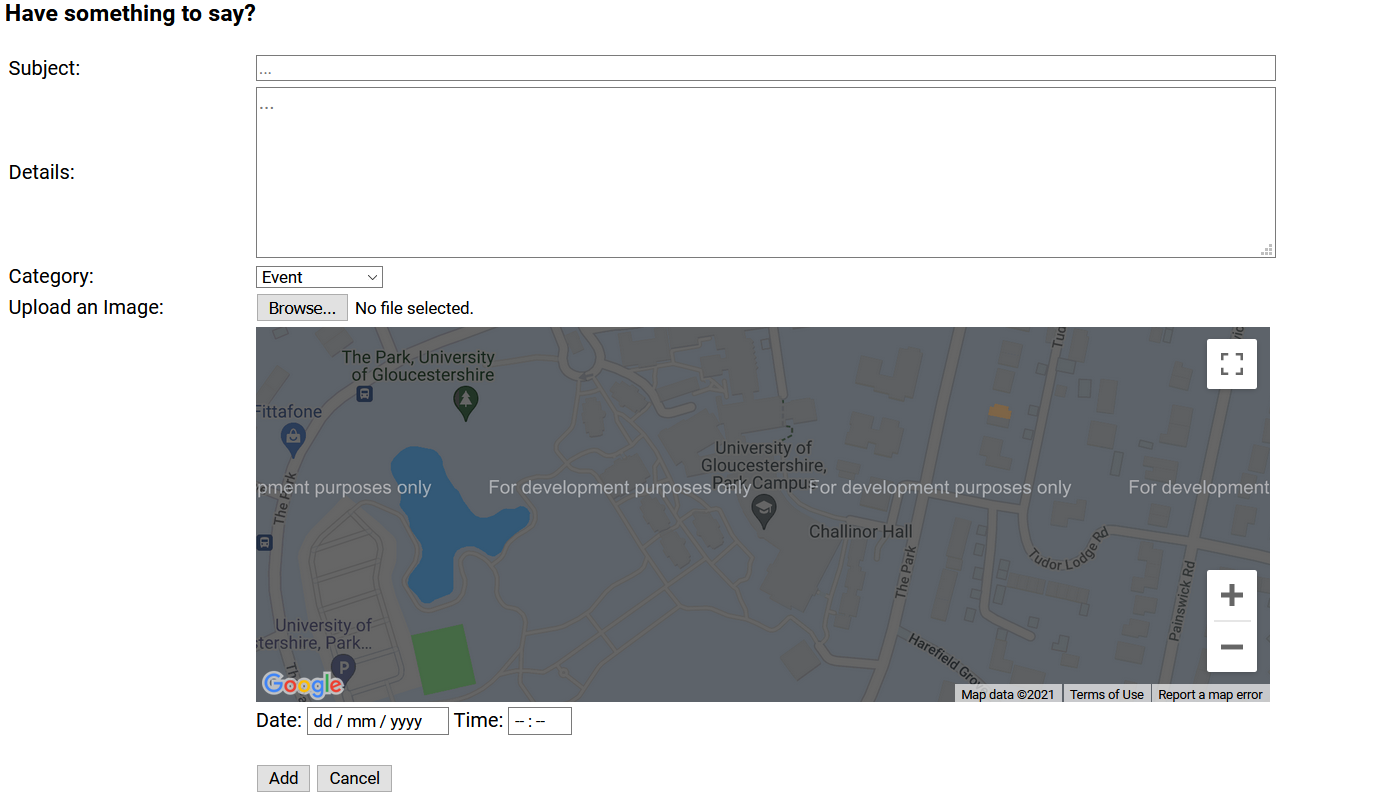
Once Registered, the Login Page will again be displayed. Allowing the new credentials to be used to login.

**For Existing Users:**

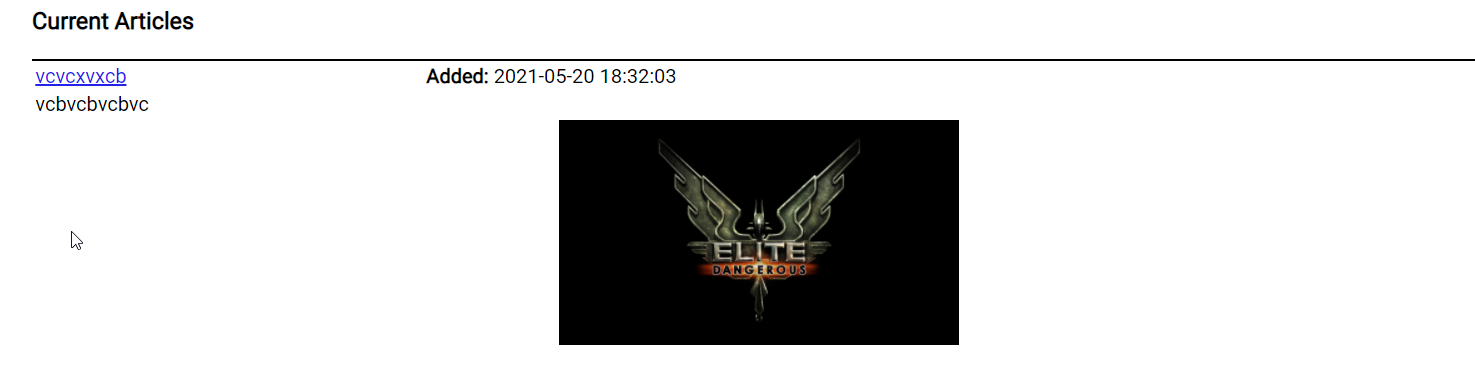
Once authenticated, the Main Page will display. This will allow a User to create new Article posts and view existing posts. Clicking on ‘Create a New Article’ will show the form for creating posts. These are separated into two types by Category. General Posts will allow the creation of a Subject and Details, along with uploading an image.



Event Posts will allow to select a location on the Map along with defining a Date and Time of the Event:



On completing the required details, selectin Add will complete the Article Post and it will then be displayed under the ‘Current Articles’ heading:



These articles will display the Subject, the date and time it was added, a first section of the Details and a thumbnail of an attached image.

You are then able to select the Subject or the thumbnail to then view the whole article. Including the complete Details, along with event date, time and map information if included.

Within these articles, users are also able to create a response post.

A picture containing application

Description automatically generated

Allowing users to provide additional feedback to existing Articles.

**Admin Users:**

The Administrator Users can login using the same login page, but on successful login, will be directed to a separate Admin page. This allows the Administrators to view the number of posts for a specific user across a definable date range.

Graphical user interface, text, application

Description automatically generated

On selecting the ‘User’ and defining the ‘From’ and ‘To’ dates, selecting Search will return the required data – Identifying the user, along with the date range as well as the number of Articles created. These results are grouped by date to easily identify the frequency of usage at a glance.

Chart

Description automatically generated

# Testing

Testing was completed using white box methodologies, where code was checked for functionality, debugging the issues identified and iterating the code until the expected outcome was achieved. This allowed for a more focused approach on the feature presently under development. There was the additional requirement to carry out full functionality tests, these were completed after several functions had been written or modified to ensure that the continued performance and behaviour of code was retained, and unintended issues introduced.

# Security

The use of code used reduces the visibility of the data in transit by utilising POST functionality when interacting with forms ensuring only required data is readily accessible via Browser Development Tools. Additionally, the interactions with databases are only opened when required, with the credentials being stored in a file that is not published as readable by the web service. The password authentication operation uses MD5, which offers protection when passing the details between the PHP code and the MySQL Database engine.

# Business Values of Different Features

The ability to add images allows users to provide information in a personalised and creative way, whether it is posting memes, screenshots, or their own artwork. The files are stored in a way that can be easily utilised in the event of future development.

The maps integration allows the user to provide accurate data for events, which allows the future utilisation of the defined location for further analytics.

Operation of the administration portal provides useful data as to user adoption and usage and overall engagement, which can help identify who the key users are, and how they may be using the software. This may guide further development to include features that would benefit these key users.

# References

Google (2018) *Overview | Maps JavaScript API | Google Developers*. Available at: https://developers.google.com/maps/documentation/javascript/overview (Accessed: 28 May 2021).

Google *Roboto - Google Fonts*. Available at: https://fonts.google.com/specimen/Roboto (Accessed: 28 May 2021).

Hughes, M. (2014) *Learn to Build with PHP*.

Shenoy, A. (2014) *Thinking in CSS*.

# Appendices

## Screenshots of Implemented Pages

Graphical user interface, text, application

Description automatically generated

./php/login.php – The page a non-logged in user will be directed to.

Graphical user interface, application

Description automatically generated

./php/registration.php – The page allowing the registration of new users.

Graphical user interface, website

Description automatically generated

./index.php (requires login credentials to be approved to view).

Chart, box and whisker chart

Description automatically generated

Create Article frame from index.php

Graphical user interface

Description automatically generated with low confidence

./php/post.php – The above screenshot is attributed to post 15 (./post.php?id=15)

Table

Description automatically generated

./php/admin.php – Administration Dashboard (only accessible as the Admin User).

## Source Code

All Source Code, including the populated test database is attached in a separate Zip file titled CT4009 2020-21 002 1902834 28 May 2020.zip attached along with this Report.