|  |  |
| --- | --- |
|  | Development Documentation |
|  |  |
|  | Ewan Donaldson ec1520347  HND Computer Science |

Contents

[Differences from Planning Designs 5](#_Toc484009482)

[Colour Scheme 5](#_Toc484009483)

[Logo changes 5](#_Toc484009484)

[Navigational Map 6](#_Toc484009485)

[Page styling 6](#_Toc484009486)

[Feature Changes 7](#_Toc484009487)

[Splash-Screen / index.html 8](#_Toc484009488)

[Source Code Listing 8](#_Toc484009489)

[User Interface Code Listings 9](#_Toc484009490)

[Error Handling 10](#_Toc484009491)

[Internal Documentation 10](#_Toc484009492)

[Login / login.php 11](#_Toc484009493)

[Source Code Listing 11](#_Toc484009494)

[User Interface Code Listings 12](#_Toc484009495)

[Error Handling 13](#_Toc484009496)

[Internal Documentation 14](#_Toc484009497)

[Register / register.php 15](#_Toc484009498)

[Source Code Listing 15](#_Toc484009499)

[User Interface Code Listings 16](#_Toc484009500)

[Error Handling 17](#_Toc484009501)

[Internal Documentation 18](#_Toc484009502)

[Home / Shop / display.php 19](#_Toc484009503)

[Source Code Listing 19](#_Toc484009504)

[First Area 19](#_Toc484009505)

[Second Area 23](#_Toc484009506)

[Third Area 25](#_Toc484009507)

[Write Reviews 26](#_Toc484009508)

[Admin Access on Navigation Bar 26](#_Toc484009509)

[Add to Cart Function 27](#_Toc484009510)

[Header Section 27](#_Toc484009511)

[Map to Location 28](#_Toc484009512)

[User Interface Code Listings 28](#_Toc484009513)

[Error Handling 31](#_Toc484009514)

[Internal Documentation 32](#_Toc484009515)

[Added / added.php 33](#_Toc484009516)

[Source Code Listing 33](#_Toc484009517)

[User Interface Code Listings 34](#_Toc484009518)

[Error Handling 34](#_Toc484009519)

[Internal Documentation 34](#_Toc484009520)

[Cart / Shopping Cart / cart.php 35](#_Toc484009521)

[Source Code Listing 35](#_Toc484009522)

[User Interface Code Listings 35](#_Toc484009523)

[Error Handling 36](#_Toc484009524)

[Internal Documentation 37](#_Toc484009525)

[See Reviews / forum.php 38](#_Toc484009526)

[Source Code Listing 38](#_Toc484009527)

[User Interface Code Listings 39](#_Toc484009528)

[Error Handling 39](#_Toc484009529)

[Internal Documentation 40](#_Toc484009530)

[Sign Out / goodbye.php 41](#_Toc484009531)

[Source Code Listing 41](#_Toc484009532)

[User Interface Code Listings 41](#_Toc484009533)

[Error Handling 42](#_Toc484009534)

[Internal Documentation 42](#_Toc484009535)

[Admin Page / admin.php 43](#_Toc484009536)

[Source Code Listing 43](#_Toc484009537)

[User Interface Code Listings 44](#_Toc484009538)

[Error Handling 44](#_Toc484009539)

[Internal Documentation 45](#_Toc484009540)

[PayPal Payment / https://www.sandbox.paypal.com/cgi-bin/webscr 46](#_Toc484009541)

[Source Code Listing 46](#_Toc484009542)

[User Interface Code Listings 46](#_Toc484009543)

[Error Handling 46](#_Toc484009544)

[Internal Documentation 47](#_Toc484009545)

[Database Structure / Content 48](#_Toc484009546)

[Users Table 48](#_Toc484009547)

[Shop table 48](#_Toc484009548)

[Orders Table 49](#_Toc484009549)

[Forum Table 49](#_Toc484009550)

[Use of Unfamiliar Constructs and Libraries 50](#_Toc484009551)

[Project Management 52](#_Toc484009552)

[Testing 53](#_Toc484009553)

[Registers Password and Confirm Password Input Boxes 53](#_Toc484009554)

[Search Function 54](#_Toc484009555)

[Filter Dropdown 54](#_Toc484009556)

[User Guide 55](#_Toc484009557)

[Login.php 55](#_Toc484009558)

[Register.php 55](#_Toc484009559)

[Display.php 55](#_Toc484009560)

[Admin.php 55](#_Toc484009561)

[Added.php 56](#_Toc484009562)

[Cart.php 56](#_Toc484009563)

[Paypal form 56](#_Toc484009564)

[Goodbye.php 56](#_Toc484009565)

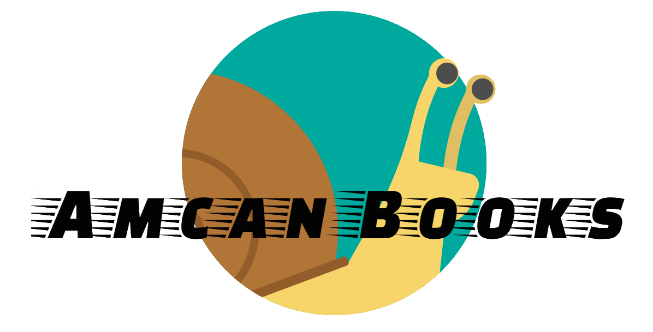
# Differences from Planning Designs

There are many changes from my planning stage of the website, all of which have justifications and reasons.

## Colour Scheme

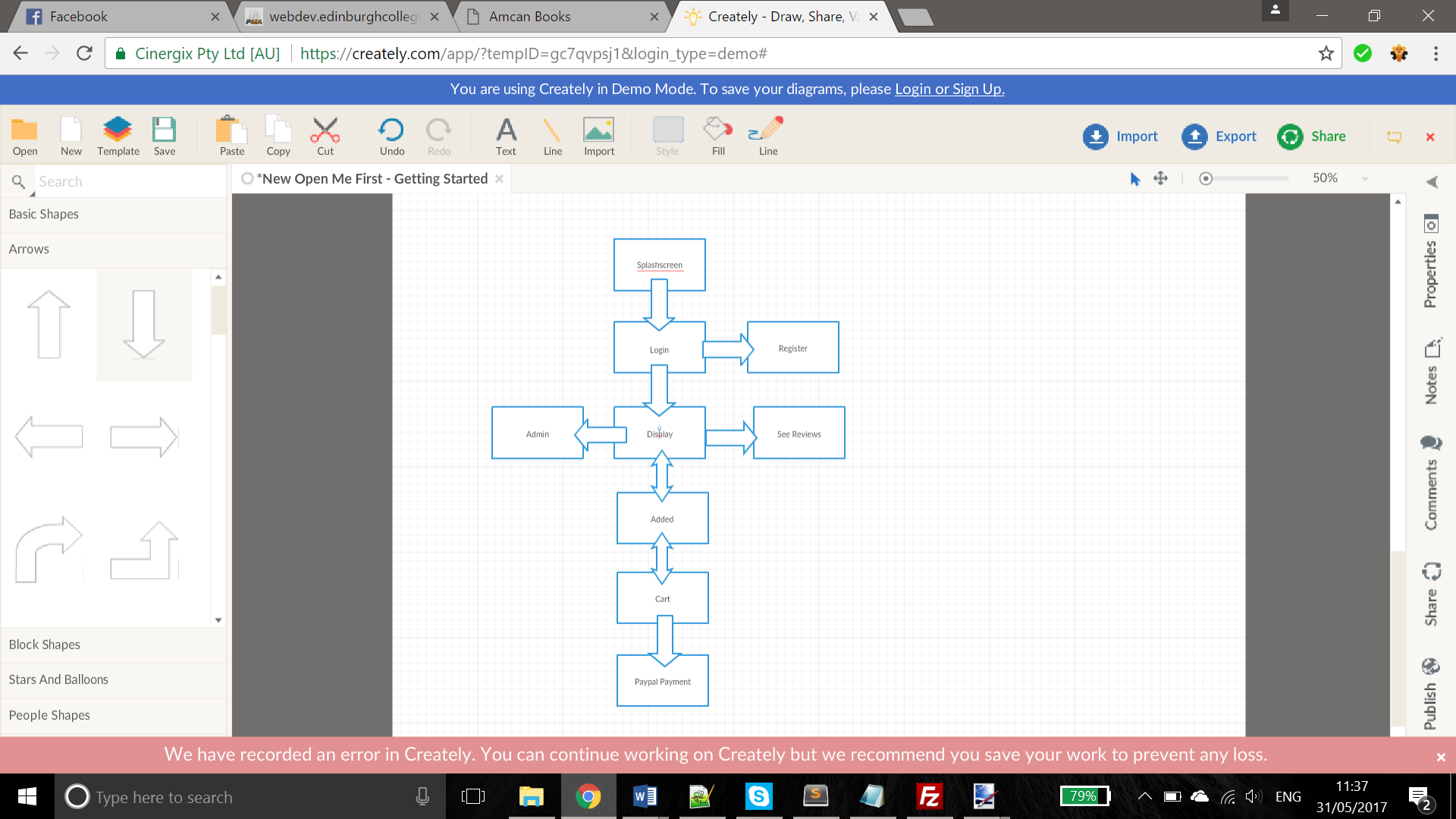
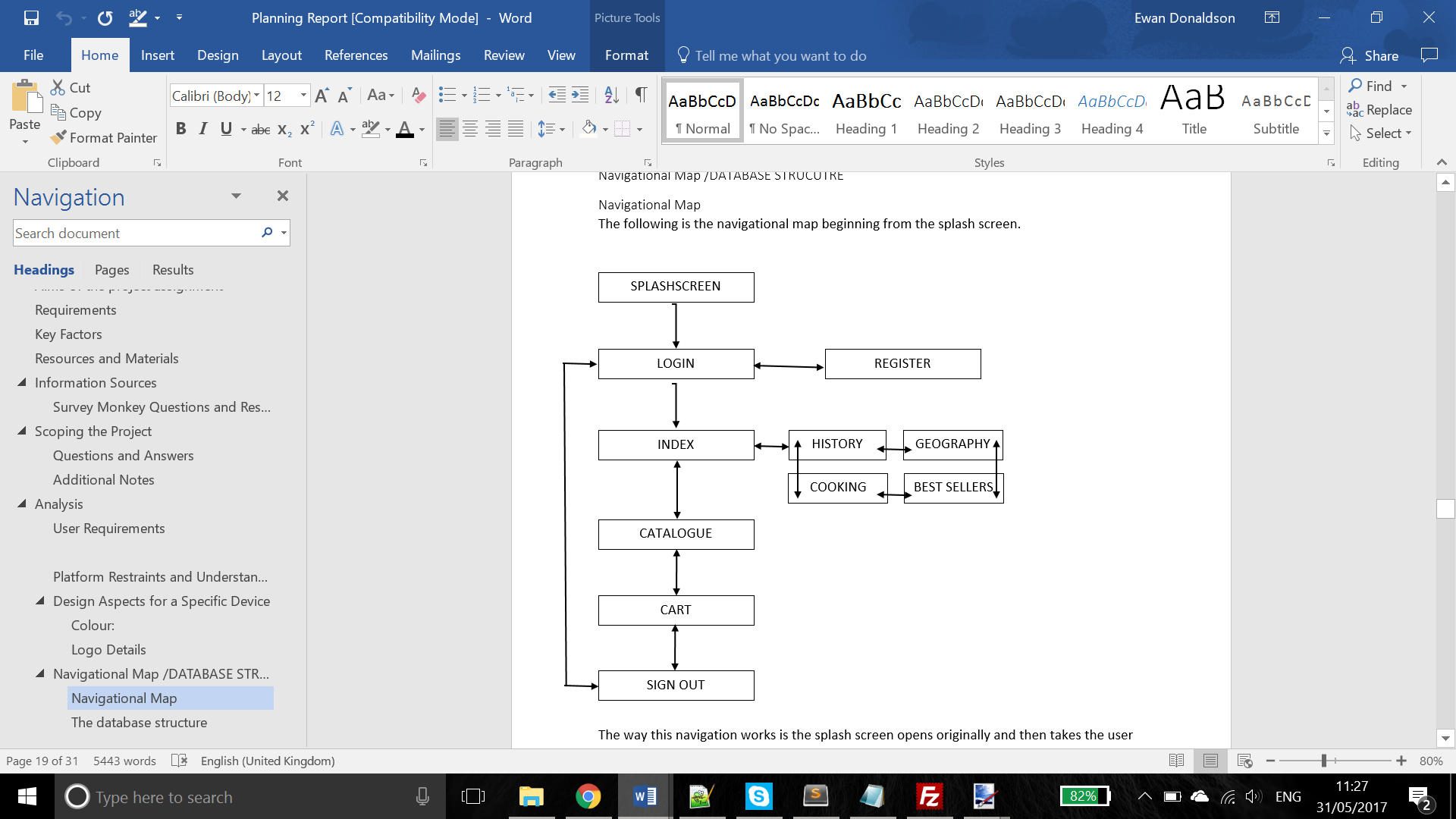
Here are the two-colour schemes for both the planning design and the actual page implementation. The first design was so basic and boring with regards to the colour scheme due to it having no eye-popping colours. With the second colour scheme displayed you can see that it is much more eye catching and appealing. Because of this I have decided to go with the second colour design with the creation of the website.

## Logo changes



Here we see both logos, the first from the planning stage and the second from the development stage. The snail aspect of this logo has stayed consistent through both stages, the text from the first is removed and replaced with a more formal text beneath in white font, and the second image has been given a nice styled <hr> tag to underline the header of the page which is also used throughout the website. The logo with a white background has also been changed to a coloured background whilst keeping the circular image to ensure that the image looks fresh and keeps within the overall style and colour scheme of the page.

## Navigational Map



The navigation of the website has been altered a fair amount from the original design. The splashscreen, login and register page are all the same but then the navigation completely changes. The index page is now a display page which now contains all the books by using a search function and filter rather than separate pages which was changed due to it being too inefficient the original way. The display page now has access to an admin page which has been discussed earlier, this has been added in increase the functionality for admin users. The user can then go to the page added.php when the user clicked the +cart button whilst adding a book to the cart. Afterwards the user is then taken to cart.php which then grants access to payment through paypal. This is different to the original design as before it would have just reserved the books but now everything can be paid for online. The catalogue from the original design has been removed due to this being too inefficient a design for the website.

## Page styling

The styling of the pages has changed as it originally was designed with multiple pages with expand buttons to show additional details, the issue with this style was that the website was very inefficient and unappealing, therefore I have changed the style to make the main page a parallax scrolling webpage, by doing this I can display multiple pieces of the information at the user’s disposal.

I have also changes the style of the navigation bar as it was not possible to fit all the required links on the bottom of the page on a single bar therefore I have put everything at the top where, if the screen is at any time too small to fit all the links, it can shrink into a dropdown menu.

## Feature Changes

The audio file feature has been removed, this feature was going to be a playable recording of the blurb/summary of the book to listen to but I thought it would be pointless to include as I had included it in case somebody could not read so they could listen to the summary, however I realised after the planning stage that this is a bad idea, as if somebody can not read the summary they therefore can not read a book and would most likely not visit a book store.

# Splash-Screen / index.html



## Source Code Listing

The key features of this fairly simplistic page are the circles of which are animated which represent a loading symbol. The code of which follows.

<header style=**"margin-top:-15%;height:100%;"**>

<div class=**"container"** id=**"maincontent"** tabindex=**"-1"**>

<div class=**"row"**>

<div class=**"col-lg-12"**>

<img class=**"img-responsive"** src=**"images/AmcanLogo.png"** alt=**""** style=**"max-width:20%;min-width:20%"**><!--Company logo-->

<div class=**"intro-text"**>

<h1 class=**"name"**>**Amcan Books**</h1><!--Company title-->

<hr class=**"star-light"**>

<span class=**"skills"**>**History - Geography - Cooking - Best Sellers**</span><!--Book categories-->

<div class=**"spinner"** style=**"margin-top:0%;"**>

<div class=**"bounce1"**></div><!--Animated loading bar-->

<div class=**"bounce2"**></div>

<div class=**"bounce3"**></div>

</div>

</div>

</div>

</div>

</div>

</header>

This code is simple html which allowed me to create div tags in which different elements of the page can exist and be scaled properly, for example the image and the loading circles. To animate the circles to give the page the appearance of a loading screen for the user I used css code which follows.

**.**spinner **{**

**margin: 100px auto 0;**

**width: 70px;**

**text-align: center;**

**}**

**.**spinner **>** div **{**

**width: 18px;**

**height: 18px;**

**background-color: #fff;**

**border-radius: 100%;**

**display: inline-block;**

**-webkit-animation: sk-bouncedelay 1.4s infinite ease-in-out both;**

**animation: sk-bouncedelay 1.4s infinite ease-in-out both;**

**}**

**.**spinner **.**bounce1 **{**

**-webkit-animation-delay: -0.32s;**

**animation-delay: -0.32s;**

**}**

**.**spinner **.**bounce2 **{**

**-webkit-animation-delay: -0.16s;**

**animation-delay: -0.16s;**

**}**

**@**-webkit-keyframes sk-bouncedelay **{**

0%**, 80%, 100% { -webkit-transform: scale(0) }**

40% **{ -webkit-transform: scale(1.0) }**

**}**

**@**keyframes sk-bouncedelay **{**

0%, 80%, 100% {

-webkit-transform: scale(0);

transform: scale(0);

} 40% {

-webkit-transform: scale(1.0);

transform: scale(1.0);

}

}

The benefit of using this css in a separate file is that it is easy to change the appearances of all the pages in which these “spinners” circles are used.

## User Interface Code Listings

In this page there are no ways the user truly interacts with the page as it is simply on a timer to ensure the next page is loaded, although to access this page the user only has to enter a simple link to the directory of the website in a search engine and they will be directly taken to this page due to it being called index.html. The link used to access this first page without diving into the directory of the website is webdev.edinburghcollege.ac.uk/~HNDEDONALDSON/Website.

## Error Handling

This page has no error handling in it due to the page’s simplicity. This page has one purpose which is to load the next page which does not require any error handling built in to it, it also does not have access to the session which Is created when a user signs in to their account due to this page being the first they come across therefore before the login page.

## Internal Documentation

I have used internal comments to ensure that the code is readable by anyone who comes across it in the future who may wish to adapt it. The reason internal comments are so important for this page is because of how the page is formatted using div tags, by putting comments in it isn’t as confusing as it could be without them. Examples of the comments are as follows.

<h1 class=**"name"**>**Amcan Books**</h1><!--Company title-->

<hr class=**"star-light"**>

<span class=**"skills"**>**History - Geography - Cooking - Best Sellers**</span><!--Book categories-->

<div class=**"spinner"** style=**"margin-top:0%;"**>

<div class=**"bounce1"**></div><!--Animated loading bar-->

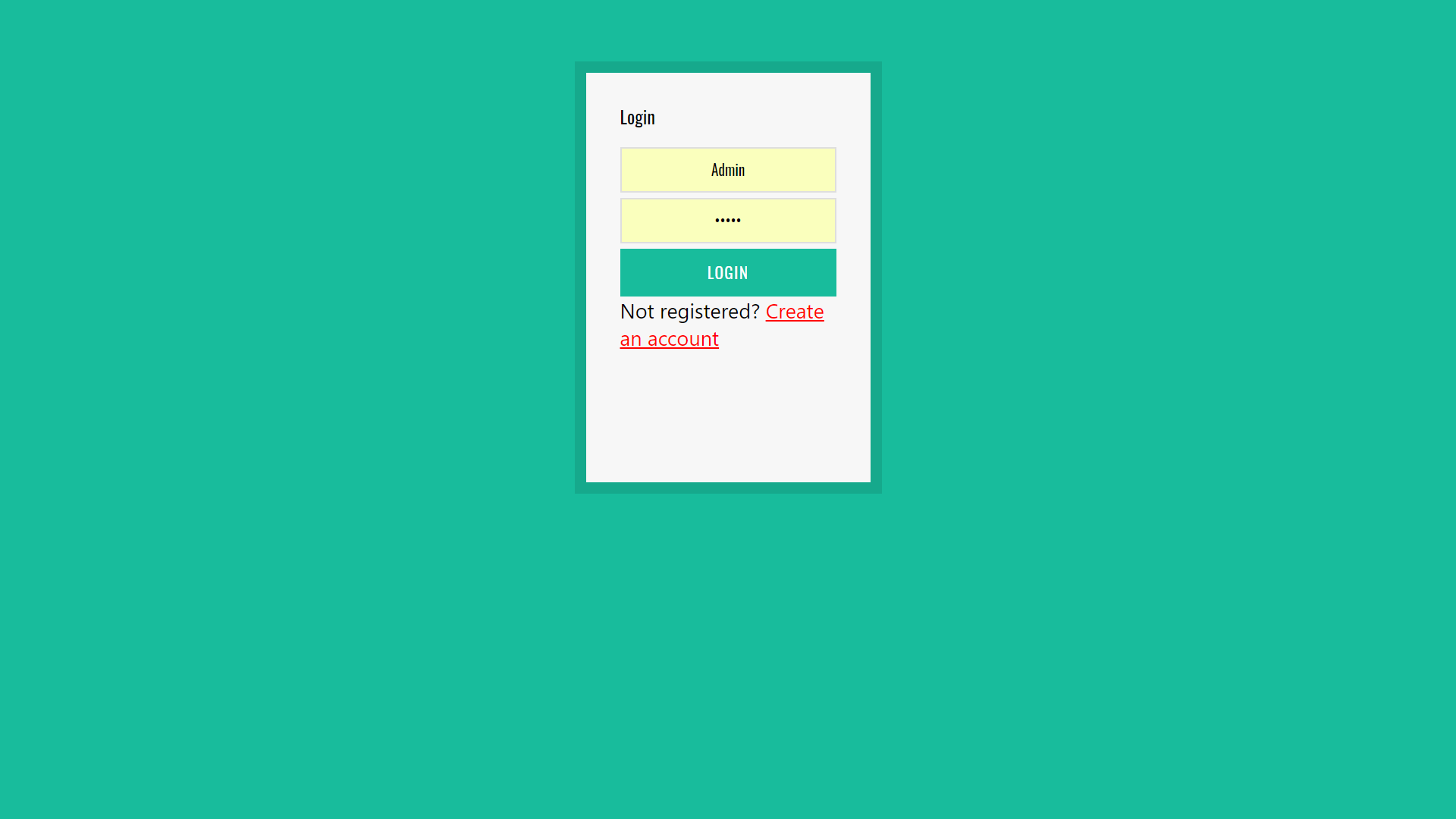
<div class=**"bounce2"**></div>

<div class=**"bounce3"**></div>

</div>

The comments here like “<!—Animated loading circle-->” allows me to remember why these are here and what their purpose are.

# Login / login.php



## Source Code Listing

The main features of this login page consist of the fact that once the user has entered in a username with a corresponding password a session is created which is used throughout all the pages thereafter until the logout page has been reached/accessed. The benefit of sessions is that php data can be carried across from one page to another allowing certain functions and actions to take place which require certain data. login\_tools.php and login\_action.php are also pages that are used in conjunction with login.php to create a session.

# Check form submitted.

**if** ( $\_SERVER[ 'REQUEST\_METHOD' ] == 'POST' )

{

# Open database connection.

**require** ( 'connect\_db.php' ) ;

# Get connection, load, and validate functions.

**require** ( 'login\_tools.php' ) ;

# Check login.

**list** ( $check, $data ) = validate ( $link, $\_POST[ 'email' ], $\_POST[ 'pass' ] ) ;

# On success set session data and display logged in page.

**if** ( $check )

{

# Access session.

**session\_start**();

$\_SESSION[ 'user\_id' ] = $data[ 'user\_id' ] ;

$\_SESSION[ 'first\_name' ] = $data[ 'first\_name' ] ;

$\_SESSION[ 'last\_name' ] = $data[ 'last\_name' ] ;

$\_SESSION[ 'email' ] = $data[ 'email' ] ;

load ( '../loading.php' ) ;

}

This code is on login\_action.php and it simply checks for the users inputs of a username and login and runs them against the database, if ok it will generate the session.

## User Interface Code Listings

The ways the user can interact with the page are through the use of input boxes, these allow the user to input into the page and to therefore login with their required login username and password with regards to existing accounts on the database that have already been created.

<table style=**"width:100%"** align=**"center"**>

<tr align=**"center"**>

<td>

<div class=**"form-group"** style=**"text-align:center;"**>

<input type=**"text"** class=**"form-control input"** placeholder=**"Email"** name=**"email"** style=**"text-align:center;margin-top:10%;width:100%"**>

</div>

</td>

</tr>

<tr align=**"center"**>

<td>

<div class=**"form-group"** style=**"text-align:center;"**>

<input type=**"password"** class=**"input form-control"** placeholder=**"Password"** name=**"pass"** style=**"text-align:center;margin-top:10%"**></h3>

</div>

</td>

</tr>

<tr align=**"center"**>

<td>

<div class=**"form-group"** style=**"text-align:center;"**><!--Div tag to contain everything-->

<p class=**"login-submit"**><!--p tag to assist with scaling-->

<input type=**"submit"** value=**"Login"** class=**"submitbutton button-primary"** style=**"z-index:-1;margin-top:10%"**><!--where the user inserts data-->

</p>

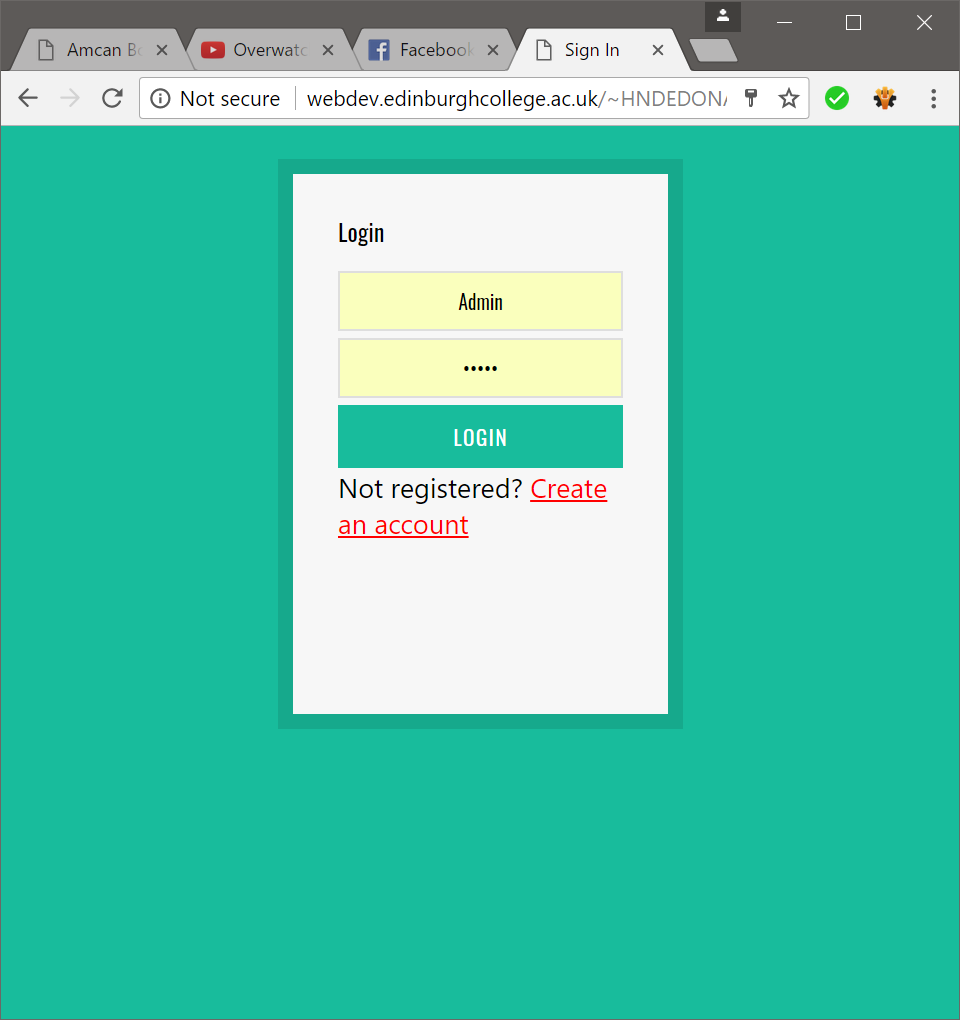
</div>

</td>

</tr>

</table>

The user also has the ability to click on the red text at the bottom which takes them to the register.php page which I will talk about in the next section of this report.



## Error Handling

The login.php page must handle a very specific set of errors as it must handle errors with regards to usernames and passwords not matching the ones on the database. If these errors are encountered then the following code is performed;

The code below is the displaying of the error message.

# Display any error messages if present.

**if** ( **isset**( $errors ) && !**empty**( $errors ) )

{

**echo** '<h3 id="err\_msg">Oops! There was a problem:<br>' ;

**foreach** ( $errors **as** $msg ) { **echo** " - **$msg**<br>" ; }

**echo** 'Please try again or Register</h3>' ;

}

This code states that if any error occurs then the user will be notified with the use of an error message telling them to try again. This following code is from login\_tools.php and is used to find the specific error type.

# Function to check email address and password.

**function** validate( $dbc, $email = '', $pwd = '')

{

# Initialize errors array.

$errors = **array**() ;

# Check email field.

**if** ( **empty**( $email ) )

{ $errors[] = 'Enter your email address.' ; }

**else** { $e = **mysqli\_real\_escape\_string**( $dbc, **trim**( $email ) ) ; }

# Check password field.

**if** ( **empty**( $pwd ) )

{ $errors[] = 'Enter your password.' ; }

**else** { $p = **mysqli\_real\_escape\_string**( $dbc, **trim**( $pwd ) ) ; }

# On success retrieve user\_id, first\_name, and last name from 'users' database.

**if** ( **empty**( $errors ) )

{

$q = "SELECT user\_id, first\_name, last\_name FROM users WHERE email='**$e**' AND pass=SHA1('**$p**')" ;

$r = **mysqli\_query** ( $dbc, $q ) ;

**if** ( @**mysqli\_num\_rows**( $r ) == 1 )

{

$row = **mysqli\_fetch\_array** ( $r, **MYSQLI\_ASSOC** ) ;

**return** **array**( **true**, $row ) ;

}

# Or on failure set error message.

**else** { $errors[] = 'Email address and password not found.' ; }

}

# On failure retrieve error message/s.

**return** **array**( **false**, $errors ) ;

}

## Internal Documentation

<div class=**"form-group"** style=**"text-align:center;"**><!--Div tag to contain everything-->

<p class=**"login-submit"**><!--p tag to assist with scaling-->

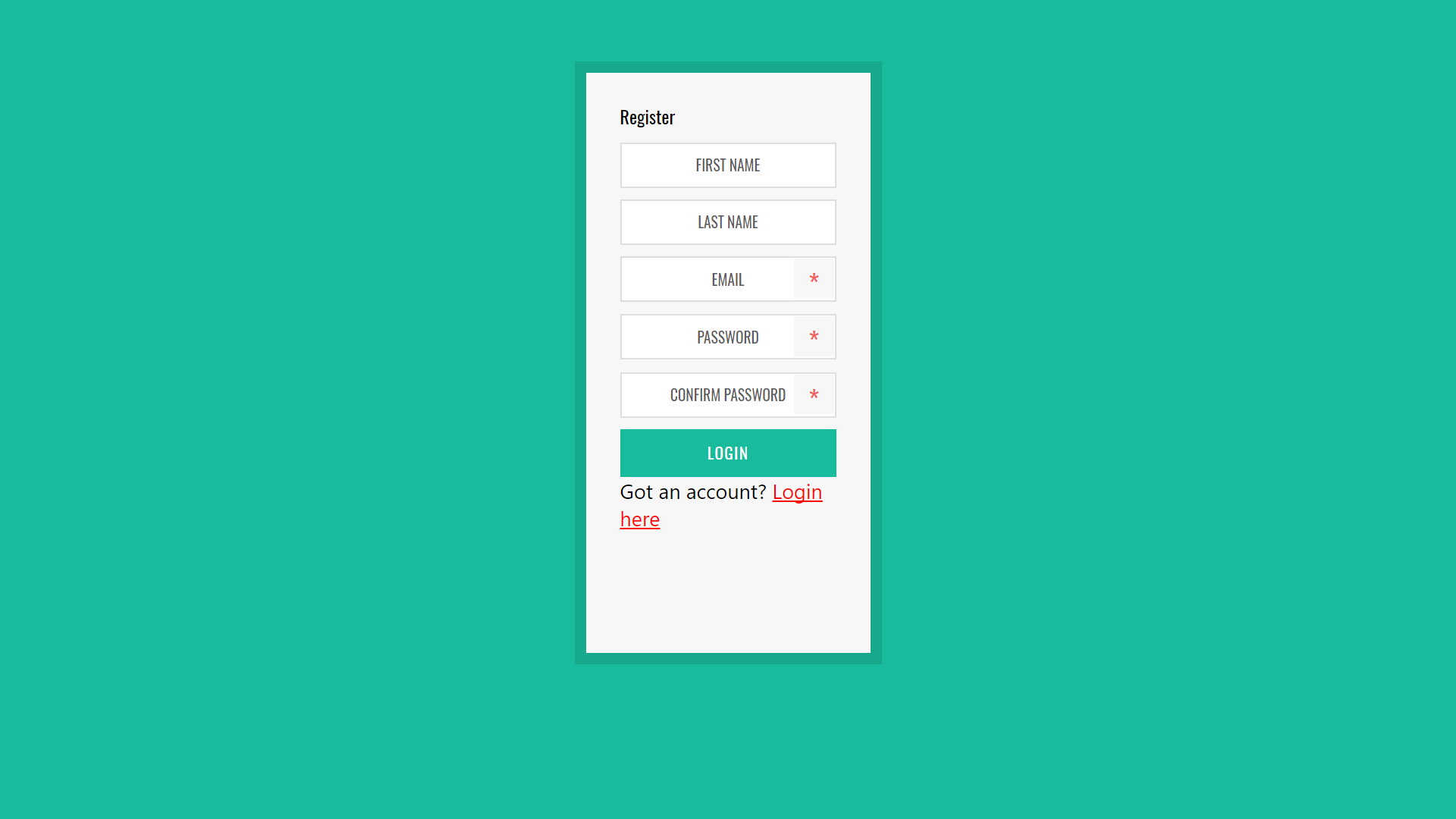
<input type=**"submit"** value=**"Login"** class=**"submitbutton button-primary"** style=**"z-index:-1;margin-top:10%"**><!--where the user inserts data-->

</p>

</div>

The comments in this code state that the div tag is used to contain all the contents and that inside is a p tag to assist with the scaling of the contents and an input box for the user to insert their data.

# Register / register.php



## Source Code Listing

The main feature of this page is to add another account to the database. The page takes the input from the user and sends it to the database to be ran as a query, upon its completion it creates the account and redirects the user to the login.php page.

# On success register user inserting into 'users' database table.

**if** ( **empty**( $errors ) )

{

$q = "INSERT INTO users (first\_name, last\_name, email, pass, reg\_date) VALUES ('**$fn**', '**$ln**','**$e**', SHA1('**$p**'), NOW() )";

$r = @**mysqli\_query** ( $link, $q ) ;

**if** ($r)

{

**header**('Location: login.php');

}

# Close database connection.

**mysqli\_close**($link);

**exit**();

}

## User Interface Code Listings

The user interacts with the webpage through the use of input boxes, similar to the login page. The user enters in multiple details which they would like linked to their account and then submit all information.

<p class=**"login-firstname"**>

<input type=**"text"** name=**"first\_name"** class=**"input form-control"** style=**"text-align:center;"** placeholder=**"First Name"** value=**"**<?php **if** (**isset**($\_POST['first\_name'])) **echo** $\_POST['first\_name']; ?>**"**>

</p>

<p class=**"login-lastname"** style=**"margin-top:5%"**>

<input type=**"text"** name=**"last\_name"** class=**"input form-control"** style=**"text-align:center;"** placeholder=**"Last Name"** value=**"**<?php **if** (**isset**($\_POST['last\_name'])) **echo** $\_POST['last\_name']; ?>**"**>

</p>

<p class=**"login-username"** style=**"margin-top:5%"**>

<input type=**"text"** name=**"email"** class=**"input form-control"** style=**"text-align:center;"** placeholder=**"Email"** value=**"**<?php **if** (**isset**($\_POST['email'])) **echo** $\_POST['email']; ?>**"**>

</p>

<p class=**"login-password"**>

<input type=**"password"** name=**"pass"** class=**"input form-control"** style=**"text-align:center;"** placeholder=**"Password"** value=**"**<?php **if** (**isset**($\_POST['pass'])) **echo** $\_POST['pass']; ?>**"** >

</p>

<p class=**"login-password"**>

<input type=**"password"** name=**"pass2"** class=**"input form-control"** style=**"text-align:center;"** placeholder=**"Confirm Password"**value=**"**<?php **if** (**isset**($\_POST['pass2'])) **echo** $\_POST['pass2']; ?>**"**>

</p>

<p class=**"login-submit"**><input type=**"submit"** name=**"wp-submit"** id=**"wp-submit"** class=**"submitbutton button-primary"** value=**"Login"** />

</p>

<h3 class=**"message"**>**Got an account?** <a href=**"login.php"**>**Login here**</a></h3>

## Error Handling

This page handles errors when it checks the information entered with the database to ensure that there are no duplicate entries for information that must be unique, such as username.

# Check for a first name.

**if** (**empty**( $\_POST[ 'first\_name' ] ) )

{ $errors[] = 'Enter your first name.' ; }

**else**

{ $fn = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'first\_name' ] ) ) ; }

# Check for a last name.

**if** (**empty**( $\_POST[ 'last\_name' ] ) )

{ $errors[] = 'Enter your last name.' ; }

**else**

{ $ln = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'last\_name' ] ) ) ; }

# Check for an email address:

**if** ( **empty**( $\_POST[ 'email' ] ) )

{ $errors[] = 'Enter your email address.'; }

**else**

{ $e = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'email' ] ) ) ; }

# Check for a password and matching input passwords.

**if** ( !**empty**($\_POST[ 'pass' ] ) )

{

**if** ( $\_POST[ 'pass' ] != $\_POST[ 'pass2' ] )

{ $errors[] = 'Passwords do not match.' ; }

**else**

{ $p = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'pass' ] ) ) ; }

}

**else** { $errors[] = 'Enter your password.' ; }

# Check if email address already registered.

**if** ( **empty**( $errors ) )

{

$q = "SELECT user\_id FROM users WHERE email='**$e**'" ;

$r = @**mysqli\_query** ( $link, $q ) ;

**if** ( **mysqli\_num\_rows**( $r ) != 0 ) $errors[] = 'User name already being used. Please re-enter user name' ;

}

The code above checks that the entered data hasn’t been entered before by referencing the database column itself would be entered in.

The other error check is the following;

**if** ( $\_POST[ 'pass' ] != $\_POST[ 'pass2' ] )

{ $errors[] = 'Passwords do not match.' ; }

This code ensures that when the user is entering the password they desire they must enter it a second time and if it is not the same then submit an error message for the user to see.

## Internal Documentation

# Connect to the database.

**require** ('connect\_db.php');

# Initialize an error array.

$errors = **array**();

# Check for a first name.

**if** (**empty**( $\_POST[ 'first\_name' ] ) )

{ $errors[] = 'Enter your first name.' ; }

**else**

{ $fn = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'first\_name' ] ) ) ; }

# Check for a last name.

**if** (**empty**( $\_POST[ 'last\_name' ] ) )

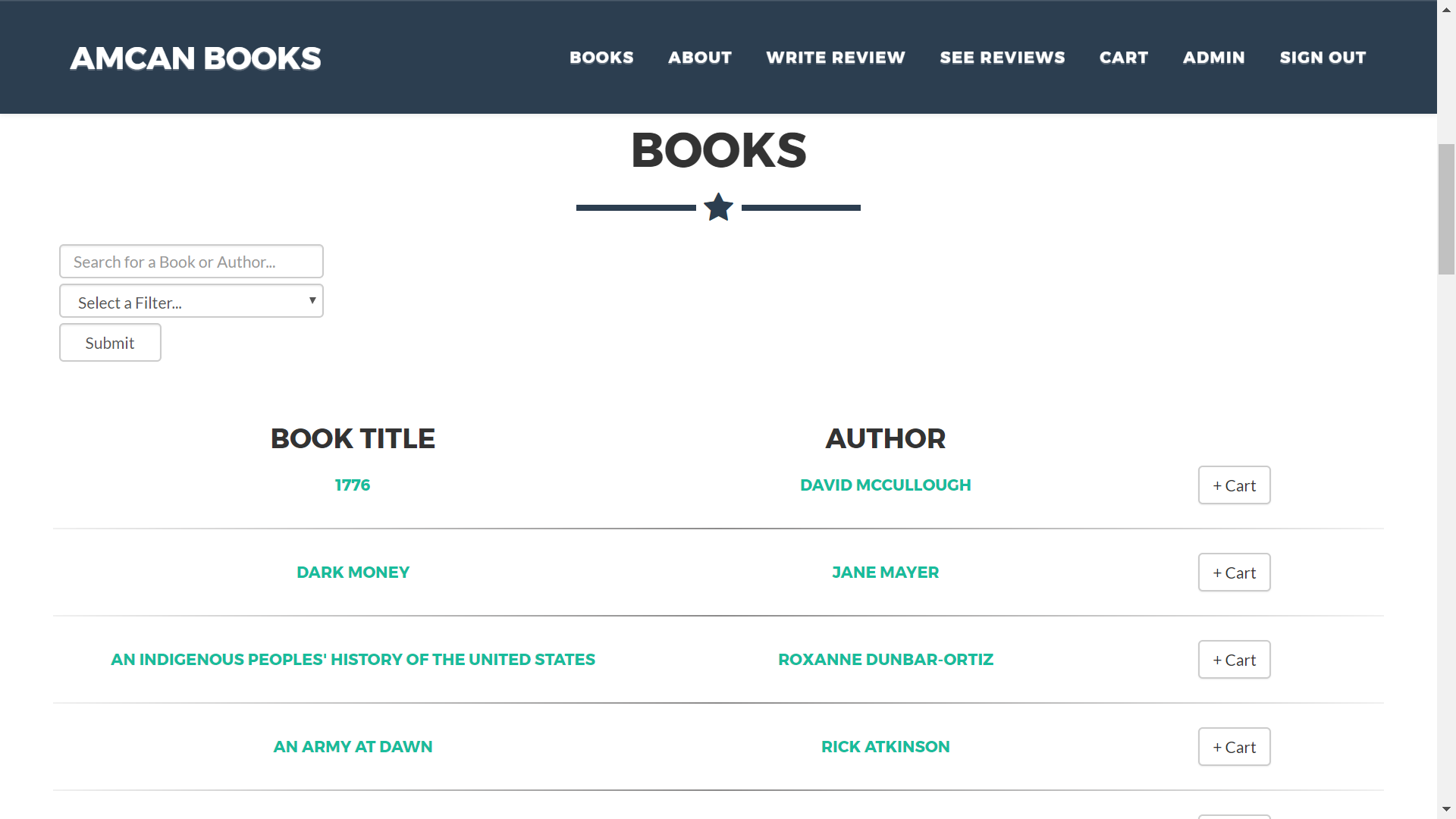
{ $errors[] = 'Enter your last name.' ; }

**else**

{ $ln = **mysqli\_real\_escape\_string**( $link, **trim**( $\_POST[ 'last\_name' ] ) ) ; }

In this code, we see comments which state the purpose of each section of code, this tells anybody else who looks at my code in the future what each section does, for example “#Connect to the database” ensures the user understands what “require (‘connect\_db.php’)” does.

# Home / Shop / display.php



## Source Code Listing

The major features of this page are the displaying of the books available in the shop along with the ability to run a search to find books or authors with certain names and to filter categories of books. The following code is all the code used to create the ability to filter and search for books and authors as well as displaying them and generating pop-ups for each book with relevant information in each.

This code can be broken down into steps/stages in which will be exampled.

### First Area

This first area of code is in place to create the pop-ups with the required information inside them and with the corresponding names to the ones that will be required later. These pop-ups are created with the use of a php WHILE loop which allows me to make it loop as many times as possible when the query is true. I have duplicated the circumstances for the other loop later on to ensure that the correct pop-up is matched to the correct book name in the list/shop. Another thing to implement into this while loop was the ability to create the popups to match the correct item on the list once the list of books had been filtered and searched through by the user which also is seen later.

<?php

# Open database connection.

**require** ( 'includes/connect\_db.php' ) ;

/\* check connection \*/

**if** (**mysqli\_connect\_errno**()) {

**printf**("Connect failed: %s\n", **mysqli\_connect\_error**());

**exit**();

}

**if** ($result = **mysqli\_query**($link, "SELECT \* FROM shop")) {

/\* determine number of rows result set \*/

$row\_cnt = **mysqli\_num\_rows**($result);

/\* close result set \*/

**mysqli\_free\_result**($result);

}

$q = "SELECT \* FROM shop" ;

$r = **mysqli\_query**( $link, $q ) ;

$all\_categories\_query = "SELECT category FROM `shop` GROUP BY category ORDER BY category" ;

$all\_categories = **mysqli\_query**( $link, $all\_categories\_query ) ;

$queryfilter = "SELECT \* FROM `shop`";

**if**( $user\_search == "")

{

$user\_search = "%";

$queryfilter = "SELECT \* FROM `shop` WHERE category LIKE '%' AND item\_name LIKE '%**$user\_search**%' OR category LIKE '%' AND author LIKE '%**$user\_search**%'";

}

$user\_search = $\_POST["book\_name"];

**if** (**isset**($\_POST['var'])) {

**if** ($user\_search == ""){

$user\_search = "%";

}

$placeholder = $\_POST['var'];

$placeholderchecked = **mysql\_real\_escape\_string**($placeholder);

$queryfilter = "SELECT \* FROM `shop` WHERE category LIKE '**$placeholderchecked**' AND item\_name LIKE '%**$user\_search**%' OR category LIKE '**$placeholderchecked**' AND author LIKE '%**$user\_search**%'";

$filterresult = **mysqli\_query**( $link, $queryfilter ) ;

}

$filterresult = **mysqli\_query**( $link, $queryfilter ) ;

$counter = "1";

**while** ($row = **mysqli\_fetch\_array**( $filterresult, **MYSQLI\_ASSOC** )) {

?>

<div class=**"portfolio-modal modal fade"** id=**"**<?php **echo** "portfolioModal**$counter**" ?>**"** tabindex=**"-1"** role=**"dialog"** aria-hidden=**"true"**>

<div class=**"modal-content"**>

<div class=**"close-modal"** data-dismiss=**"modal"**>

<div class=**"lr"**>

<div class=**"rl"**>

</div>

</div>

</div>

<div class=**"container"**>

<div class=**"row"**>

<div class=**"col-lg-8 col-lg-offset-2"**>

<div class=**"modal-body"**>

<h2><?php **echo** $row['item\_name'] ?></h2>

<!--<?php **echo** $counter ?>-->

<hr class=**"star-primary"**>

<img src=**"**<?php **echo** $row['image'] ?>**"** class=**"img-responsive img-centered"** alt=**""** style=**"width:20%;"**>

<p><?php **echo** $row['blurb'] ?></p>

<ul class=**"list-inline item-details"**>

<li>**Client:**

<strong><a href=**"http://startbootstrap.com"**>**Start Bootstrap**</a>

</strong>

</li>

<li>**Date:**

<strong><a href=**"http://startbootstrap.com"**>**April 2014**</a>

</strong>

</li>

<li>**Service:**

<strong><a href=**"http://startbootstrap.com"**>**Web Development**</a>

</strong>

</li>

</ul>

<table>

<tr>

<td>

</td>

<td>

<button type=**"button"** class=**"btn btn-default"** data-dismiss=**"modal"**><i class=**"fa fa-times"**></i> **Close**</button>

</td>

</tr>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

<?php

$counter++;

} ;

?>

<section id=**"portfolio"**>

<div class=**"container"**>

<div class=**"row"**>

<div class=**"col-lg-12 text-center"**>

<h2>**Books**</h2>

<hr class=**"star-primary"**>

</div>

</div>

<div class=**"row"**>

<?php

# Open database connection.

**require** ( 'includes/connect\_db.php' ) ;

/\* check connection \*/

**if** (**mysqli\_connect\_errno**()) {

**printf**("Connect failed: %s\n", **mysqli\_connect\_error**());

**exit**();

}

**if** ($result = **mysqli\_query**($link, "SELECT \* FROM shop")) {

/\* determine number of rows result set \*/

$row\_cnt = **mysqli\_num\_rows**($result);

/\* close result set \*/

**mysqli\_free\_result**($result);

}

$q = "SELECT \* FROM shop" ;

$r = **mysqli\_query**( $link, $q ) ;

$all\_categories\_query = "SELECT category FROM `shop` GROUP BY category ORDER BY category" ;

$all\_categories = **mysqli\_query**( $link, $all\_categories\_query ) ;

$queryfilter = "SELECT \* FROM `shop`";

**if**( $user\_search == "")

{

$user\_search = "%";

$queryfilter = "SELECT \* FROM `shop` WHERE category LIKE '%' AND item\_name LIKE '%**$user\_search**%' OR category LIKE '%' AND author LIKE '%**$user\_search**%'";

}

$user\_search = $\_POST["book\_name"];

**if** (**isset**($\_POST['var'])) {

**if** ($user\_search == ""){

$user\_search = "%";

}

$placeholder = $\_POST['var'];

$placeholderchecked = **mysql\_real\_escape\_string**($placeholder);

$queryfilter = "SELECT \* FROM `shop` WHERE category LIKE '**$placeholderchecked**' AND item\_name LIKE '%**$user\_search**%' OR category LIKE '**$placeholderchecked**' AND author LIKE '%**$user\_search**%'";

$filterresult = **mysqli\_query**( $link, $queryfilter ) ;

}

$filterresult = **mysqli\_query**( $link, $queryfilter ) ;

$counter = "0";

?>

### Second Area

This next part of code below is in place to create a form in which the user can interact with the page to set a filter option or to search for a book or author, this will be spoken more about in the next section of this report.

<div id=**"functions"**>

<form action=**""** method=**"POST"**><!--form for all user inputs-->

<table style=**"width:100%"**>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<input type=**"text"** name=**"book\_name"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;"** placeholder=**"Search for a Book or Author..."** /><!--input for the users search-->

</td>

</tr>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<select name=**"var"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;"** onchange=**"this.form.submit();"**>

<option selected=**"selected"** value=**"%"**>**Select a Filter...**</option><!--default dropdown option-->

<option value=**"%"**>**all**</option><!--all option on dropdown-->

<?php

**while** ($row = **mysqli\_fetch\_array**( $all\_categories, **MYSQLI\_ASSOC** )) {//loop to generate remaining options from available categories

$category[] = ($row['category']);//set variable to the category on certain row being accessed

$counter++;

?>

<option value=**"**<?php **echo** $row['category'] ?>**"**><?php **echo** $row['category'] ?></option><!--creating the option-->

<?php

} ;

?>

</select>

</td>

</tr>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<input type=**"submit"** name=**"submit"** class=**"form-control input"** value=**"Submit"** style=**"width:90px"** />

</td>

</tr>

</table>

</form>

</div>

<?php

$counter = '1';

?>

<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav navbar-right"**>

<li class=**"hidden"**>

<a href=**"#page-top"**></a>

</li>

</ul>

</div>

### Third Area

This next part here is in place to display all the book titles along with the author and an add to cart button on to allow for the purchase of said book. This code also has the ability to update once the user has submitted a filter and/or a search parameter therefore ensuring that only the books required are selected and that the corresponding pop-ups match.

<table class=**"display"** style=**"width:100%;text-align:center;"**>

<col style=**"width:2.5%;min-width:2.5%;"**>

<col style=**"width:40%;min-width:40%;"**>

<col style=**"width:5%;min-width:5%;"**>

<col style=**"width:30%;min-width:30%;"**>

<col style=**"width:22.5%;min-width:22.5%;"**>

<tr>

<td colspan=**"5"**>

<table class=**"filter"** style=**"width:100%"**>

<col style=**"width:5%;min-width:70%;"**>

<col style=**"width:5%;min-width:20%;"**>

<tr>

<td></td>

<td>

</td>

</tr>

</table>

</td>

</tr>

<tr>

<td></td>

<td>

<h3>**Book Title**</h3>

</td>

<td></td>

<td>

<h3>**Author**</h3>

</td>

</tr>

<?php

**while** ($row = **mysqli\_fetch\_array**( $filterresult, **MYSQLI\_ASSOC** )) {

?>

<div class=**"col-sm-4h portfolio-item"**>

<tr>

<td></td>

<td>

<a href=**"**<?php **echo** "#portfolioModal**$counter**" ?>**"** class=**"portfolio-link"** data-toggle=**"modal"**>

<h5><?php **echo** $row['item\_name'] ?></h5>

<!--<?php **echo** $counter ?>-->

</a>

</td>

<td></td>

<td>

<a href=**"**<?php **echo** "#portfolioModal**$counter**" ?>**"** class=**"portfolio-link"** data-toggle=**"modal"**>

<h5><?php **echo** $row['author'] ?></h5>

</a>

</td>

<td>

<?php

**echo** '<a href="includes/added.php?id='.$row['item\_id'].'"><button type="button" class="btn btn-default" data-dismiss="modal">+ Cart</button></a>';

?>

</td>

</tr>

<tr>

<td colspan=**"5"**>

<hr style=**"border: 0;height: 1px;background-image: -webkit-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);background-image: -moz-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);background-image: -ms-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);background-image: -o-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);"**>

</td>

</tr>

</div>

<?php

$counter++;

} ;

?>

</table>

</div>

</div>

</section>

### Write Reviews

Another main feature is the ability to leave a review from the display page. The user enters in required details which are then put into variables and then sent to the database in a query, it then runs and upon successful completion the user is taken to a page in which they can see other reviews, this page however can be accessed form the navigation bar.

### Admin Access on Navigation Bar

The navigation bar has an admin page link if the user is on the admin account, to do this I took the display.php navbar and inserted an if statement to check what the user’s id was and if it matched that of the admins, they could then get access to the page. To ensure that the admin page is highly secure I have set the id for the admin account to 01 which, due to id’s being unique, means that nobody else can have access to the admin panel without the admin account login details. The benefit of having the admin page linked in the navigation bar is that it makes it much easier to access that admin page.

<?php

# Open database connection.

**require** ( 'includes/connect\_db.php' ) ;

/\* check connection \*/

**if** (**mysqli\_connect\_errno**()) {

**printf**("Connect failed: %s\n", **mysqli\_connect\_error**());

**exit**();

}

**if** ($result = **mysqli\_query**($link, "SELECT \* FROM shop")) {

/\* determine number of rows result set \*/

$row\_cnt = **mysqli\_num\_rows**($result);

/\* close result set \*/

**mysqli\_free\_result**($result);

}

$q = "SELECT \* FROM users" ;

$r = **mysqli\_query**( $link, $q ) ;

$counter = "1";

$user\_id = $\_SESSION[ 'user\_id' ];

**if** ($user\_id == "01"){

?>

<li class=**"page-scroll"**>

<a href=**"admin.php"**>**Admin**</a>

</li>

<?php

};

?>

### Add to Cart Function

Another feature this page has is the ability to add a book to the shopping cart, this is done by the user clicking on a button which using the loop previously spoken about to ensure that the correct book is sent to the cart even when the user searches and filters through the books and authors. Upon the click of this button all required information about the book is sent over to another page using the sessions, this then is saved to the cart page which can be accessed through the navigation bar.

### Header Section

Underneath the navigation bar there is a section in which shows the name of the company, “Amcan Books”. This is here to add some style to the page as it conveys a very professional tone to the website whilst also displaying the company logo which is very recognisable. When an image and name is recognisable it can assist to promote the company as they will receive customers through word of mouth between old and soon to be members.

### Map to Location

A map of the location of where Amcan Books is operated out of is required on my webpage so I implemented it into the bottom of my display.php page with a marker on it so it is easy to find.

<!-- Location -->

<section id=**"location"** class=**"success"**>

<div class=**"container"**>

<div class=**"row"**>

<div class=**"col-lg-12 text-center"**>

<h2>**Location**</h2>

<hr class=**"star-light"**>

</div>

</div>

</div>

<div style=**"margin-left:5%"**>

<iframe src=**"https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d1119.5003057725348!2d-4.281098707145652!3d55.8626557959889!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x0%3A0x0!2zNTXCsDUxJzQ1LjYiTiA0wrAxNic0OS45Ilc!5e0!3m2!1sen!2suk!4v1496072204729"** width=**"95%"** height=**"450"** frameborder=**"0"** style=**"border:0"** allowfullscreen></iframe>

</div>

</section>

I used an iframe to implement the google maps and set the location using coordinates which then places a marker there. I then set the size of the map using percentages rather than exact pixel sizes, with regards to width, to ensure it also looks good on phone.

## User Interface Code Listings

The user can interface with this page using a dropdown, input-box and a submit button all of which exist in the same form.

<form action=**""** method=**"POST"**><!--form for all user inputs-->

<table style=**"width:100%"**>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<input type=**"text"** name=**"book\_name"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;"** placeholder=**"Search for a Book or Author..."** /><!--input for the users search-->

</td>

</tr>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<select name=**"var"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;"** onchange=**"this.form.submit();"**>

<option selected=**"selected"** value=**"%"**>**Select a Filter...**</option><!--default dropdown option-->

<option value=**"%"**>**all**</option><!--all option on dropdown-->

<?php

**while** ($row = **mysqli\_fetch\_array**( $all\_categories, **MYSQLI\_ASSOC** )) {//loop to generate remaining options from available categories

$category[] = ($row['category']);//set variable to the category on certain row being accessed

$counter++;

?>

<option value=**"**<?php **echo** $row['category'] ?>**"**><?php **echo** $row['category'] ?></option><!--creating the option-->

<?php

} ;

?>

</select>

</td>

</tr>

<tr>

<td style=**"padding-bottom:5px;padding-left:5px"**>

<input type=**"submit"** name=**"submit"** class=**"form-control input"** value=**"Submit"** style=**"width:90px"** />

</td>

</tr>

</table>

</form>

The code here has the input-box for searching for an author or book which allows the user to enter anything in to it, and if there are any matches with the book/author name or part of the name then it will be displayed. The form also has the dropdown which the user can interface with by clicking on the dropdown then selecting an option, some of these options are predefined but some of the options are made by scanning for all various categories on the database and then creating the filter with said choices. The third user interfacing method in this form is the submit button which requires the user to click on it, the user would click on this button when they have chosen their desired filter and search parameter to the submit their choice.

The user can also interact using the review section of the page, this contain an input-box where the user can type the name of the book they are leaving a review for, and an input-box for the actual review to be written in. Upon completion of their review they can then hit the post button which will then submit the form and take them to see every review available.

<div class=**"row"**>

<div class=**"col-lg-8 col-lg-offset-2"**>

<form action=**"includes/post\_action.php"** method=**"post"** accept-charset=**"utf-8"**>

<div class=**"row control-group"**>

<div class=**"form-group col-xs-12 floating-label-form-group controls"**>

<label for=**"name"**>**Name**</label>

<input type=**"text"** class=**"form-control"** placeholder=**"Book Name"** id=**"name"** name=**"movie"**>

<p class=**"help-block text-danger"**></p>

</div>

</div>

<div class=**"row control-group"**>

<div class=**"form-group col-xs-12 floating-label-form-group controls"**>

<label for=**"message"**>**Message**</label>

<textarea rows=**"5"** class=**"form-control"** name=**"comment"** placeholder=**"Message"** id=**"message"** required data-validation-required-message=**"Please enter a message."**></textarea>

<p class=**"help-block text-danger"**></p>

</div>

</div>

<br>

<div id=**"success"**></div>

<div class=**"row"**>

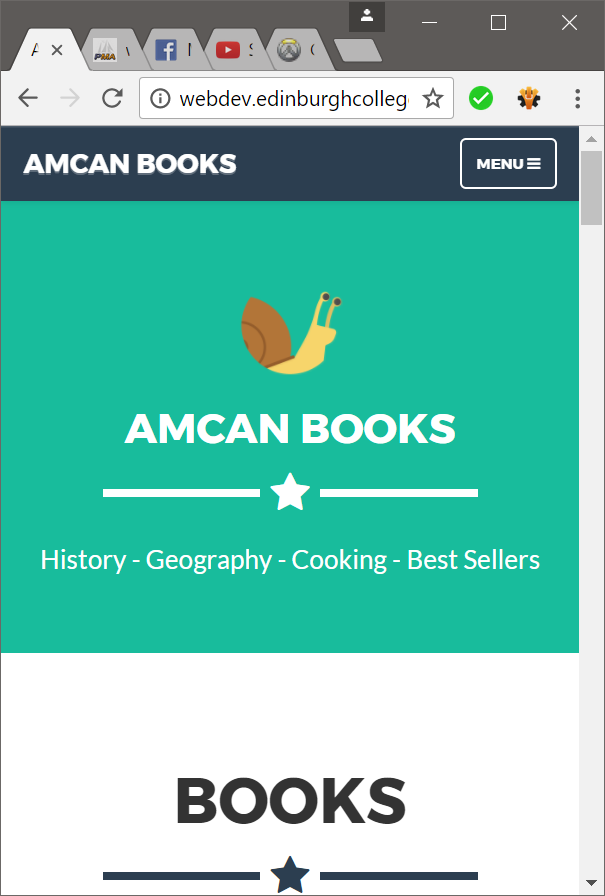
<div class=**"form-group col-xs-12"**>

<input type=**"submit"** value=**"Post Review"** class=**"btn btn-success btn-lg"**>

</div>

</div>

With regards to the navigation bar it is simplistic whilst being operated on a screen with a desktop resolution but if in phone mode it changes the navigation bar into a drop down.



<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav navbar-right"**>

<li class=**"hidden"**>

<a href=**"#page-top"**></a>

</li>

<li class=**"page-scroll"**>

<a href=**"#portfolio"**>**Books**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"#about"**>**About**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"#contact"**>**Write Review**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"includes/forum.php"**>**See Reviews**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"includes/cart.php"**>**Cart**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"includes/goodbye.php"**>**Sign Out**</a>

</li>

</ul>

</div>

## Error Handling

The error handling in this page consists of the usual “if not logged in” error checking of which is on every page with a session in it.

# Redirect if not logged in.

**if** ( !**isset**( $\_SESSION[ 'user\_id' ] ) ) {

**require** ( 'login\_tools.php' ) ;

load() ;

}

The other error checking below exists for a very important purpose with regards to the user inputs. When the user chooses to search for a book but not filter them by category it is important to therefore make sure that it isn’t being filtered or worse allowing it to crash. To prevent anything from happening when the user doesn’t choose a filter I have created an IF statement which when the user sets the search to a null value it then searches the user search variable which is used in queries to a percentage symbol, this allows the database query to then run the search function as required whilst the filter does nothing without corrupting the query as required.

**if**( $user\_search == "")

{

$user\_search = "%";

$queryfilter = "SELECT \* FROM `shop` WHERE category LIKE '%' AND item\_name LIKE '%**$user\_search**%' OR category LIKE '%' AND author LIKE '%**$user\_search**%'";

}

Error checking also takes place when the reviews have been sent by the user, the page post\_action.php is used to convert the user input into a query which is then run on the phpmyadmin database. Error checking occurs here when it checks that a new row on the database has been added, meaning it is checking that the new review has been added.

**if** ( **mysqli\_affected\_rows**( $link ) != 1 )

{

**echo** '<p>Your message has now been posted.</p>'.**mysqli\_error**( $link);

}

**else**

{

load( 'forum.php' );

}

## Internal Documentation

Comments have been used to ensure that the code is easily understood for future work.

<td style=**"padding-bottom:5px;padding-left:5px"**>

<select name=**"var"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;"** onchange=**"this.form.submit();"**>

<option selected=**"selected"** value=**"%"**>**Select a Filter...**</option><!--default dropdown option-->

<option value=**"%"**>**all**</option><!--all option on dropdown-->

<?php

**while** ($row = **mysqli\_fetch\_array**( $all\_categories, **MYSQLI\_ASSOC** )) {//loop to generate remaining options from available categories

$category[] = ($row['category']);//set variable to the category on certain row being accessed

$counter++;

?>

<option value=**"**<?php **echo** $row['category'] ?>**"**><?php **echo** $row['category'] ?></option><!--creating the option-->

<?php

} ;

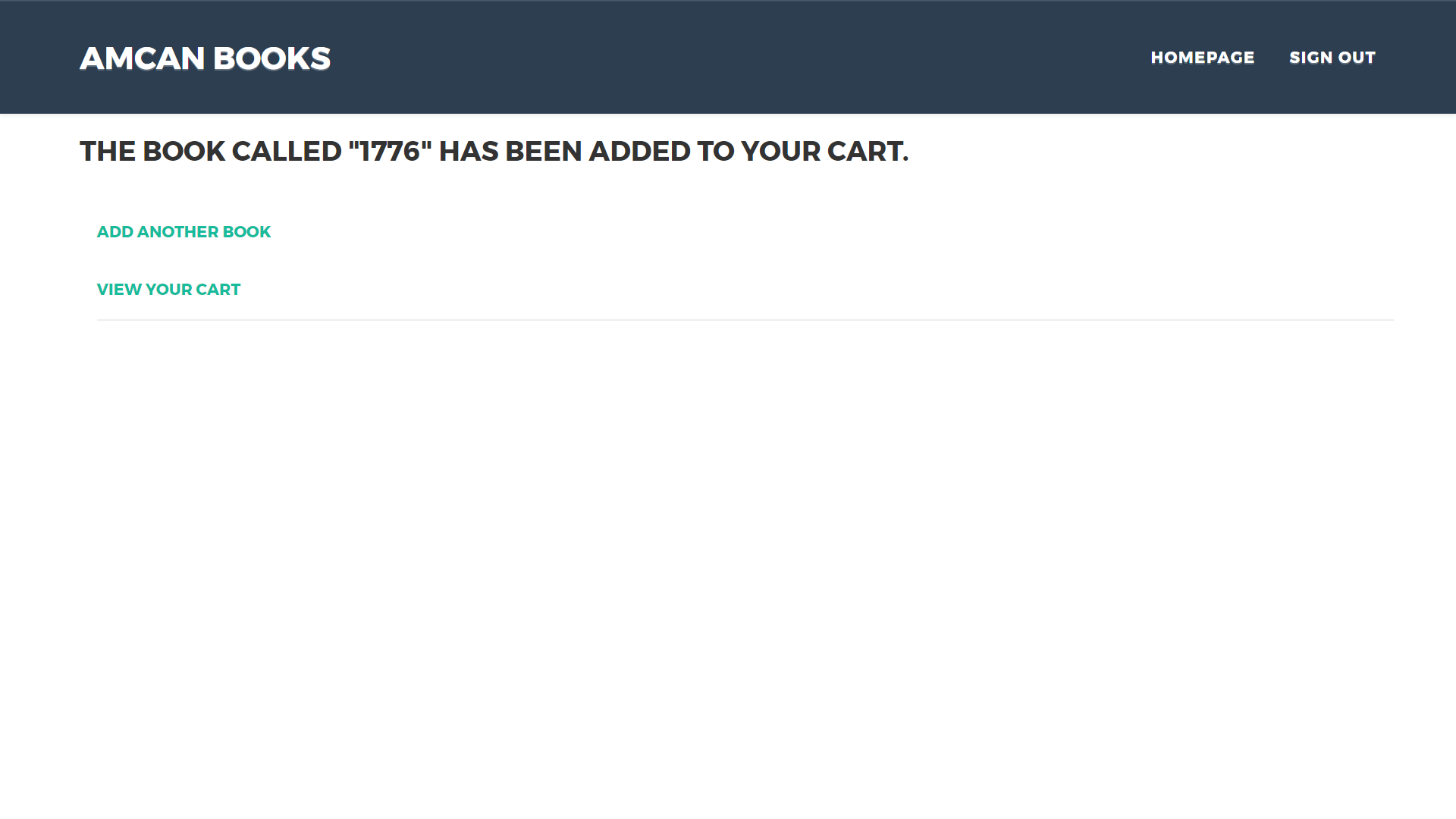
?>

</select>

</td>

Comments here have been used to ensure people know what the purpose of each part within the form does here is full dropdown and as seen above you can see what each stage does due to the comments.

# Added / added.php



## Source Code Listing

This is the page called upon the clicking of “+cart” on the display page, its purpose is to add all the relevant information to the cart page and to inform the user that they have added the book to their cart as well as give them some options on what to do next.

# Retrieve selective item data from 'shop' database table.

$q = "SELECT \* FROM shop WHERE item\_id = **$id**" ;

$r = **mysqli\_query**( $link, $q ) ;

**if** ( **mysqli\_num\_rows**( $r ) == 1 )

{

$row = **mysqli\_fetch\_array**( $r, **MYSQLI\_ASSOC** );

# Check if cart already contains one of this product id.

**if** ( **isset**( $\_SESSION['cart'][$id] ) )

{

# Add one more of this product.

$\_SESSION['cart'][$id]['quantity']++;

**echo** '<div class="container"><h3> The book called "'.$row["item\_name"].'" has been added to your cart.</h3>';

}

**else**

{

# Or add one of this product to the cart.

$\_SESSION['cart'][$id]= **array** ( 'quantity' => 1, 'price' => $row['item\_price'] ) ;

**echo** '<div class="container"><h3> The book called "'.$row["item\_name"].'" has been added to your cart.</h3>' ;

}

}

## User Interface Code Listings

On this page, the user has the ability to click on two simple links, one of which takes them to another page, this page being cart.php. The other link takes the user back to display.php which allows them to add more books to their cart.

<h5> <a href="../display.php" role="button">Add Another Book</a> <h5>

<br>

<h5> <a href="cart.php" role="button">View Your Cart</a> </h5>

## Error Handling

The only error checking this page deals with is the one consistent with all pages with sessions on them, the code following.

# Redirect if not logged in.

**if** ( !**isset**( $\_SESSION[ 'user\_id' ] ) ) {

**require** ( 'login\_tools.php' ) ;

load() ;

}

## Internal Documentation

Comments can be seen on here on the php part of this page in which the title of the page is being accessed and the database is being accessed while obtaining the books unique id from the display.php page.

# Set page title and display header section.

$page\_title = 'Reservation Addition' ;

//include ( 'headerdefault.php' ) ;

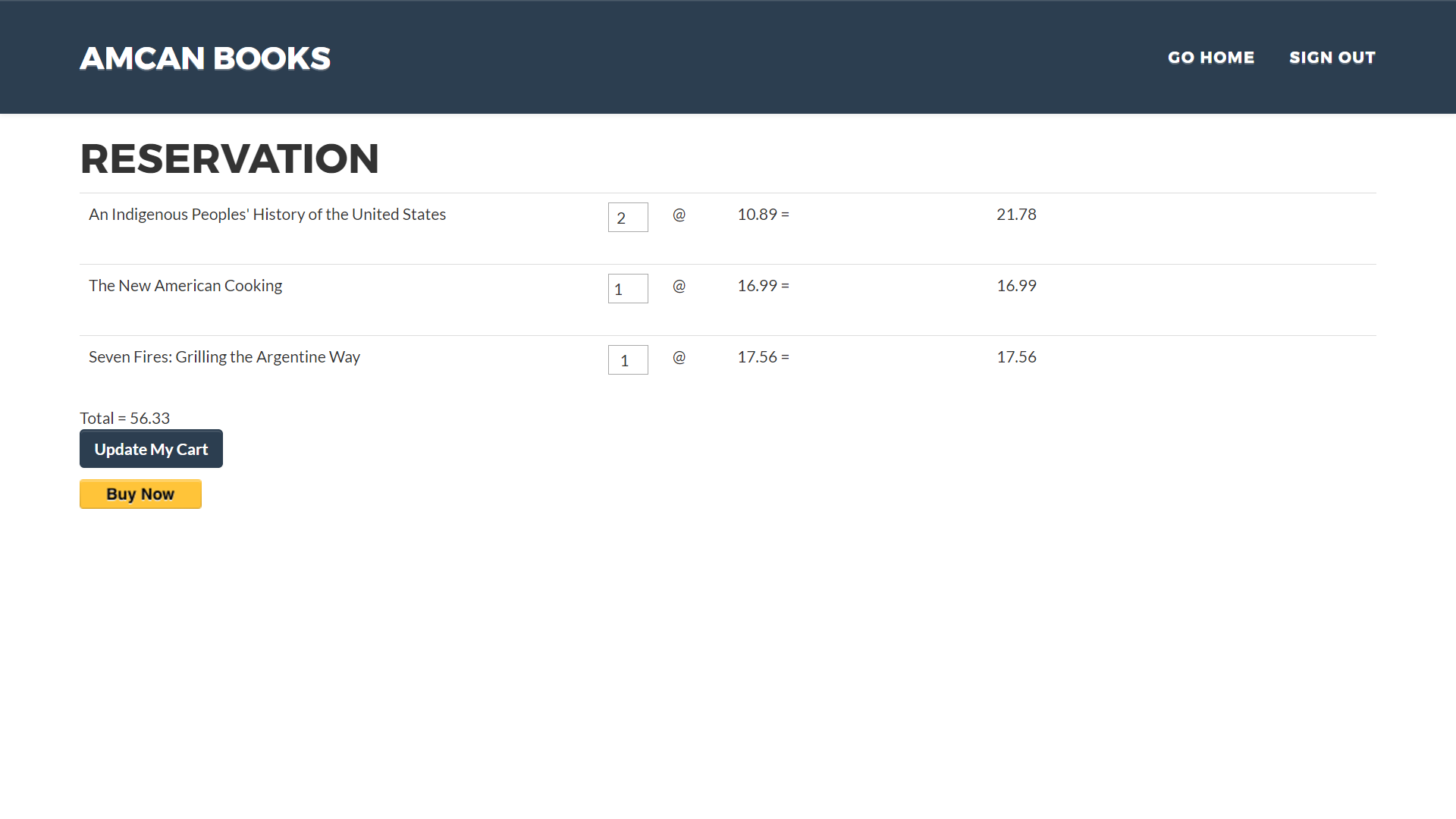
# Get passed product id and assign it to a variable.

**if** ( **isset**( $\_GET['id'] ) ) $id = $\_GET['id'] ;

# Open database connection.

**require** ( 'connect\_db.php' ) ;

# Cart / Shopping Cart / cart.php



## Source Code Listing

The main feature of this page is the ability to enter in the amount of each book you want and to then update you cart with the new costs appearing.

**while** ($row = **mysqli\_fetch\_array** ($r, **MYSQLI\_ASSOC**))

{

# Calculate sub-totals and grand total.

$subtotal = $\_SESSION['cart'][$row['item\_id']]['quantity'] \* $\_SESSION['cart'][$row['item\_id']]['price'];

$total += $subtotal;

Once the new amounts of each product have been selected the while loop is ran again with most things staying the same such as the books in the cart, but the new subtotals and overall total changing allowing the user to see the new costs.

## User Interface Code Listings

The user interacts with the cart page using both input-boxes for the quantity of each book and buttons for updating the cart and continuing with the purchase.

<div class"container" style="width:100%">

<p><input type="submit" name="submit" class="btn btn-primary" value="Update My Cart"></p>

</div>

</form>';

?>

<form action=**"https://www.sandbox.paypal.com/cgi-bin/webscr"** method=**"post"**>

<!-- Identify your business so that you can collect the payments. -->

<input type=**"hidden"** name=**"business"** value=**"ewandonaldson5facilitator@gmail.com"**>

<!-- Specify a Buy Now button. -->

<input type=**"hidden"** name=**"cmd"** value=**"\_xclick"**>

<!-- Specify details about the item that buyers will purchase. -->

<input type=**"hidden"** name=**"item\_name"** value=**"Books"**>

<input type=**"hidden"** name=**"amount"** value=**"**<?php **echo** $total; ?>**"**>

<input type=**"hidden"** name=**"currency\_code"** value=**"GBP"**>

<!-- Display the payment button. -->

<input type=**"image"** name=**"submit"** border=**"0"**

src=**"https://www.paypalobjects.com/webstatic/en\_US/i/btn/png/btn\_buynow\_107x26.png"**

alt=**"Buy Now"**>

<img alt=**""** border=**"0"** width=**"1"** height=**"1"**

src=**"https://www.paypalobjects.com/en\_US/i/scr/pixel.gif"** >

</form>

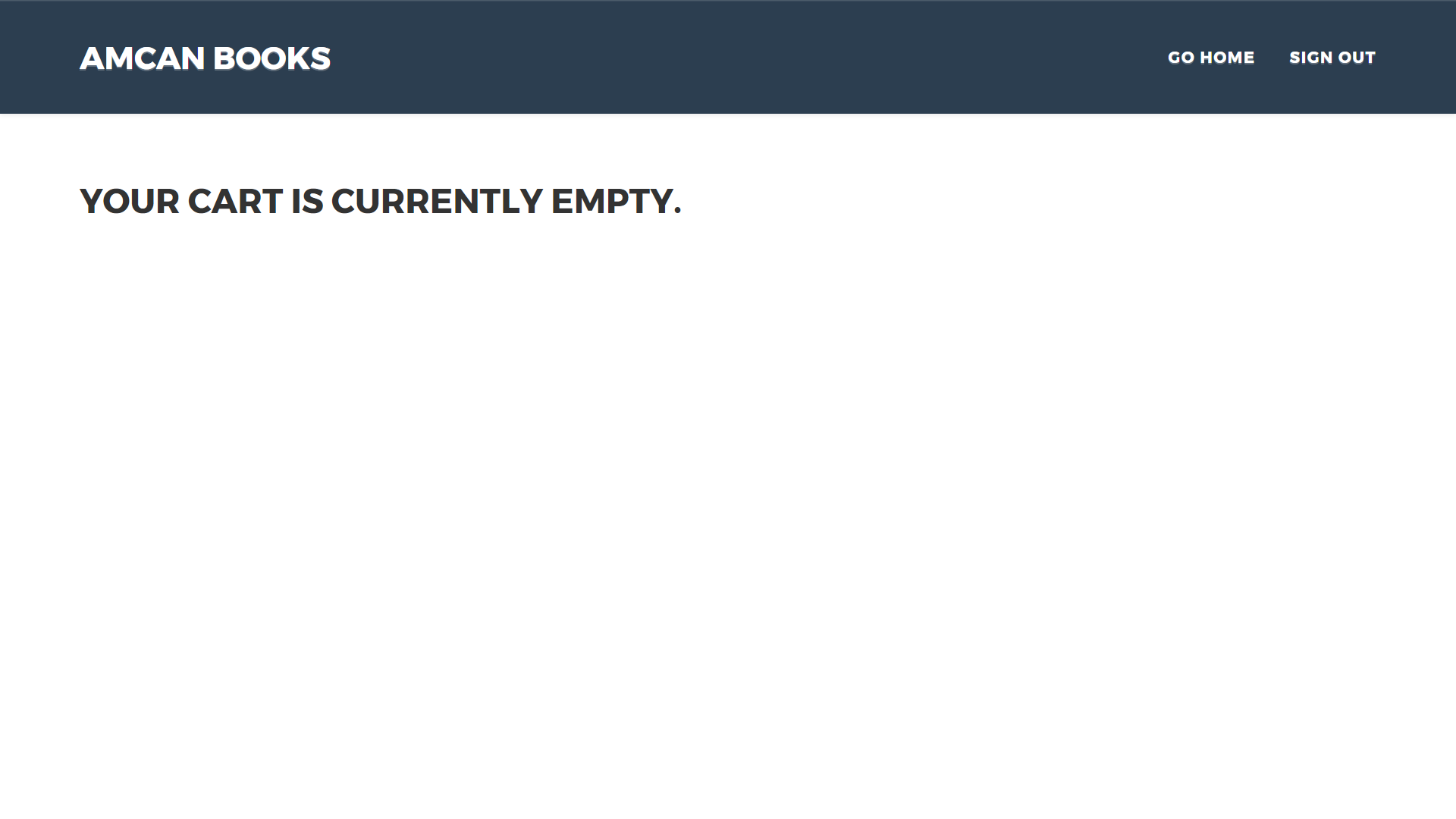
The input boxes have a set quantity of 1 in each of them by default however once the user changes the numbers they must press “update cart” of which upon doing so the quantities are set and the totals update.

The other button is the “Buy Now” button which allows the user to proceed with the purchase by taking them to a paypal form on a different page.

## Error Handling

The error handling on this page consists of the standard redirect if not logged in code, and code to display a message to the user if the cart has nothing in it.

The code below is from an if statement, the statement contains all the code that generates the cart page as we see in the image above, however if this page is visited and the cart is empty the user is displayed the following message.



The code following is from the else statement at the end of the previously stated if statement.

**else**

# Or display a message.

{ **echo** '<br><br><div class="container"><h2>Your cart is currently empty.</h2></div>' ; }

## Internal Documentation

Comments have been used on this page to inform the user of what each stage of the cart they are working on. The code following shows this.

# Check if form has been submitted for update.

**if** ( $\_SERVER['REQUEST\_METHOD'] == 'POST' )

{

# Update changed quantity field values.

**foreach** ( $\_POST['qty'] **as** $item\_id => $item\_qty )

{

# Ensure values are integers.

$id = (**int**) $item\_id;

$qty = (**int**) $item\_qty;

# Change quantity or delete if zero.

**if** ( $item\_qty == 0 ) { **unset** ($\_SESSION['cart'][$item\_id]); }

**elseif** ( $item\_qty > 0 ) { $\_SESSION['cart'][$item\_id]['quantity'] = $item\_qty; }

}

}

# Initialize grand total variable.

$total = 0;

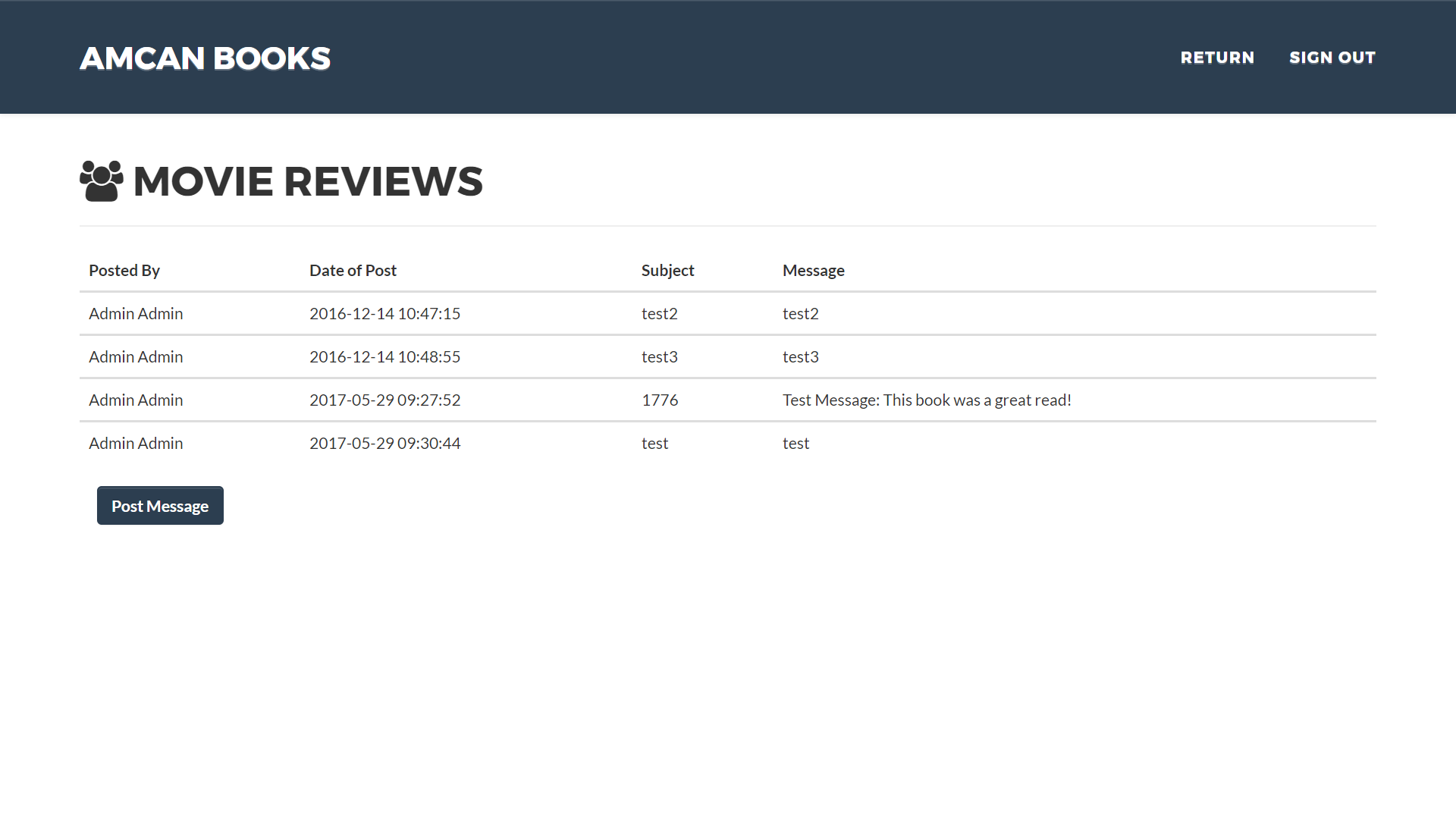
# Display the cart if not empty.

**if** (!**empty**($\_SESSION['cart']))

{

We see here that the programmer is being informed of what each section of code is doing if certain requirements are met. Due to this being complex code these comments greatly aid a programmer’s readability of this code as without them it would take much longer to understand this all.

# See Reviews / forum.php



## Source Code Listing

The main features of this page are the way it can display all the previously made reviews that have been submitted onto the database from the display.php page.

# Display body section, retrieving from 'forum' database table.

$q = "SELECT \* FROM forum" ;

$r = **mysqli\_query**( $link, $q ) ;

**if** ( **mysqli\_num\_rows**( $r ) > 0 )

{

**echo** '<table class="table">';

**echo** '<thead><th> Posted By </th><th> Date of Post </th><th> Subject </th><th id="msg"> Message </th></tr></thead>';

**while** ( $row = **mysqli\_fetch\_array**( $r, **MYSQLI\_ASSOC** ))

{

**echo** '<tbody><tr><td>' . $row['first\_name'] .' '. $row['last\_name'].'</td>

<td>' . $row['post\_date'].'</td>

<td>' . $row['movie'] . '</td>

<td>' . $row['comment'] . '</td> </tr>';

}

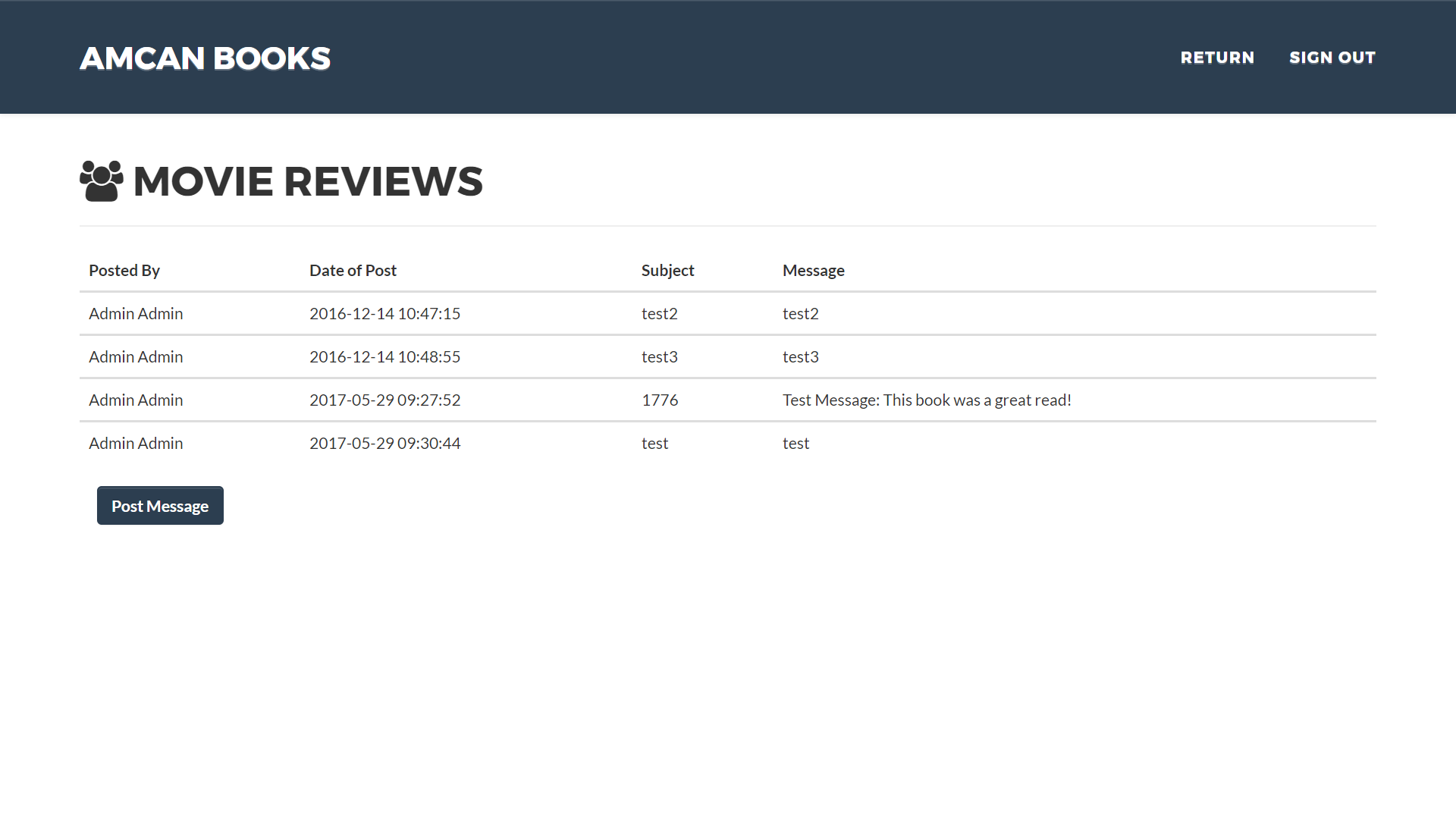
}

**else** { **echo** '<p>There are currently no messages.</p>' ; }

The code above here is the code that displays the body of the page which consists of the user’s first then last name, the post date and time, the book and the comment left with it. All this is done by creating a table and then using a loop to generate each row with the data from the database. The code above also generates and error message if the review page is empty allowing the user to be informed that there are no reviews on the database at that moment.

## User Interface Code Listings

On this page, there in not many ways in which the user can interact with it due to it being a place to simply display the reviews from the database. However, the user can still interface with the navigation bar at the top which consists of two buttons, one of which takes the user to goodbye.php which signs the user out of the session the other of which takes the user back to display.php to further use the website as a whole.



<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav navbar-right"**>

<li class=**"hidden"**>

<a href=**"#page-top"**></a>

</li>

<li class=**"page-scroll"**>

<a href=**"../display.php"**>**Return**</a>

</li>

<li class=**"page-scroll"**>

<a href=**"goodbye.php"**>**Sign Out**</a>

</li>

</ul>

</div>

## Error Handling

The error handling on this page consists of the standard redirect if not logged in code, and code to display a message to the user if the database has nothing in it.

# Redirect if not logged in.

**if** ( !**isset**( $\_SESSION[ 'user\_id' ] ) ) { **require** ( 'login\_tools.php' ) ; load() ; }

The is also an if statement that displays every review however when there are no reviews the page can’t be left blank because that does not aid the user in any way, therefore and else statement has been added to display an error message to the user informing them of the issue.

}

**else** { **echo** '<p>There are currently no messages.</p>' ; }

## Internal Documentation

The code below is from the main body of the forum page.

{

**echo** '<tbody><tr><td>' . $row['first\_name'] .' '. $row['last\_name'].'</td>

<td>' . $row['post\_date'].'</td>

<td>' . $row['movie'] . '</td>

<td>' . $row['comment'] . '</td> </tr>'; #this part displays firstname, lastname, date of review, the book, and the comment made

}

The comments here show programmers what each of these table cells output which can be very useful for in the future maybe more or less information will be placed in the table, so knowing what is currently there by briefly reading the comment allows them to make their own judgement.

# Sign Out / goodbye.php



## Source Code Listing

The main feature of this page is to reset and end the session so afterwards a new session can be created.

# Clear existing variables.

$\_SESSION = **array**() ;

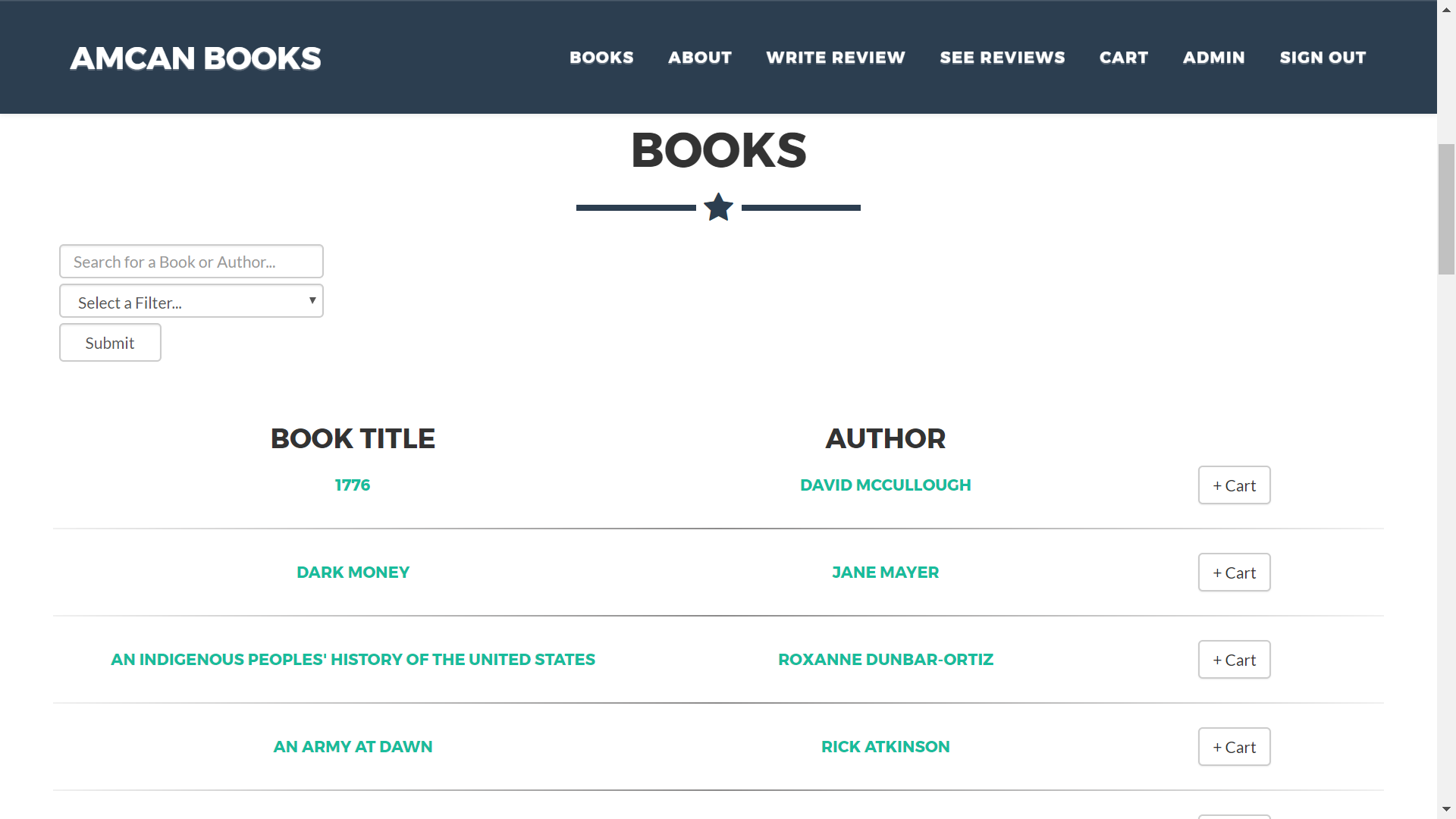
# Destroy the session.

**session\_destroy**() ;

The code above resets the session by clearing out all the variables saved by the session and then destroys the session meaning it can no longer be accessed by the user, this means that they can not go back and undo this as the session no longer exists.

## User Interface Code Listings

There is no way for the user to interact with this page, however the page must be accessed through the sign out button on the navigation bar of which is on every page that has access to the session.



## Error Handling

The error handling on this page consists of the standard redirect if not logged in code, and code to display a message to the user if the cart has nothing in it.

# Redirect if not logged in.

**if** ( !**isset**( $\_SESSION[ 'user\_id' ] ) ) { **require** ( 'login\_tools.php' ) ; load() ; }

## Internal Documentation

The following code is examples of comments I have put on the goodbye.php page.

# Set page title and display header section.

$page\_title = 'Goodbye' ;

**include** ( '' ) ;

# Clear existing variables.

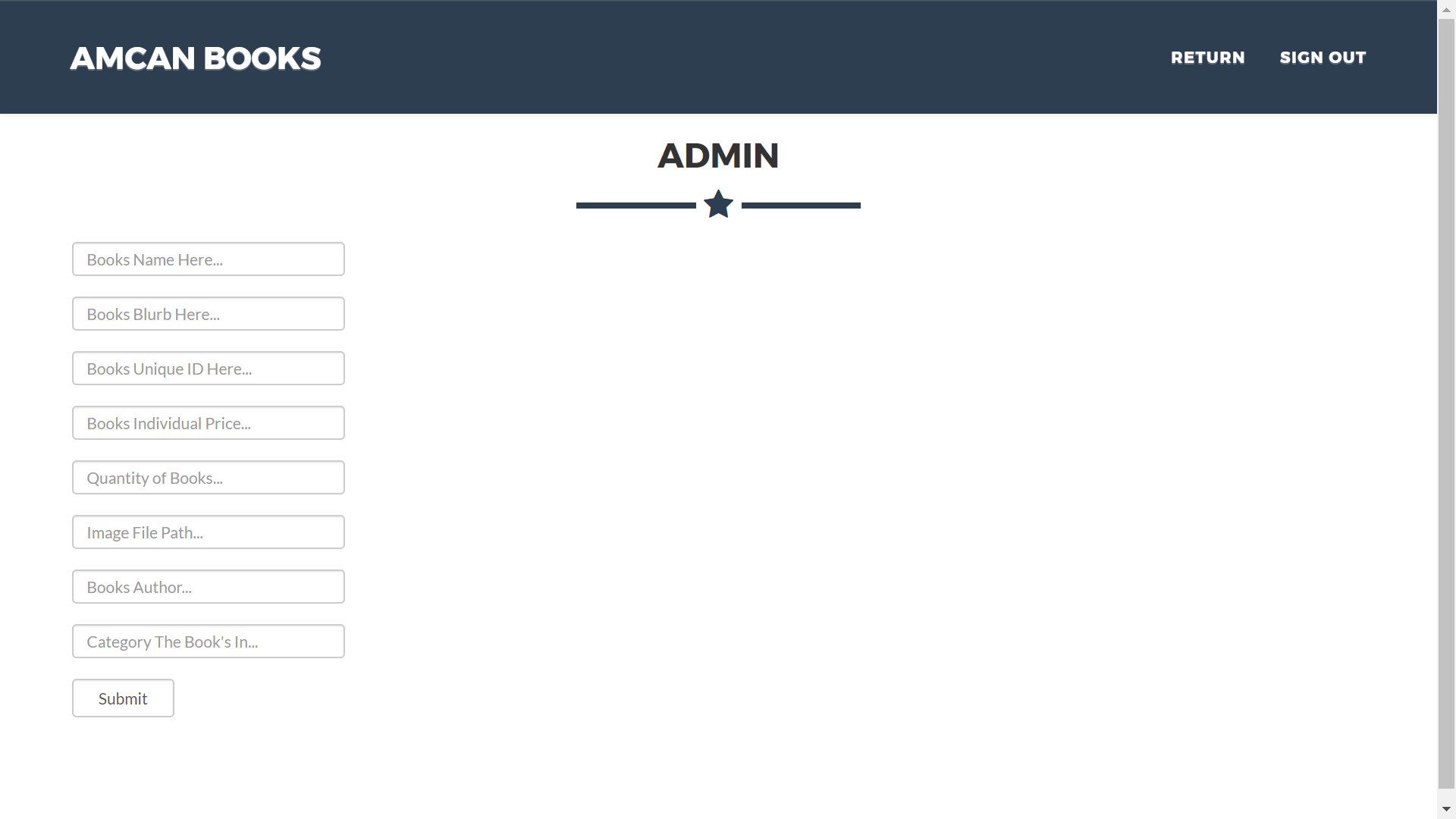
$\_SESSION = **array**() ;

# Destroy the session.

**session\_destroy**() ;

The comments here describe what each section of the page does, the first section is a simple “set title for page”, the second is as stated previously the code which resets and destroys the sessions completely. The benefit of having the comments here is that everyone can easily tell how the session is being ended without understanding the advanced code.

# Admin Page / admin.php



## Source Code Listing

The main feature of this page is the ability to add a new book to the database, this is done by the user’s/admin’s input which is then put it into a query and then submitted to the database to be ran.

<?php

**include**('includes/connect\_db.php');

**if**(**isset**($\_POST['submit']))

{

$item\_name = $\_POST["item\_name"];

$blurb = $\_POST["blurb"];

$item\_id = $\_POST["item\_id"];

$item\_price = $\_POST["item\_price"];

$item\_qty = $\_POST["item\_name"];

$image = $\_POST["image"];

$author = $\_POST["author"];

$category = $\_POST["category"];

$insert\_query = "INSERT INTO shop (item\_name,blurb,item\_id,item\_price,item\_qty,image,author,category) VALUES ('$item\_name','$blurb','$item\_id','$item\_price','$item\_qty','$image','$author','$category');";

}

$result = **mysqli\_query**($link, $insert\_query);

?>

Here we can see the variables being set with the output from the user within certain input-boxes, then the variables with their content are put into a variable and then sent off as a query to the database to be ran. Upon the query being ran they database will update and will then show the new book on display.php. Unfortunately this doesn’t work as of yet but I will discuss this at the end in the project management stage.

## User Interface Code Listings

The following is the user inputs for the admin to enter a new book to the store.

<form>

<input type=**"text"** name=**"item\_name"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Books Name Here..."** />

<input type=**"text"** name=**"blurb"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Books Blurb Here..."** />

<input type=**"text"** name=**"item\_id"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Books Unique ID Here..."** />

<input type=**"text"** name=**"item\_price"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Books Individual Price..."** />

<input type=**"text"** name=**"item\_qty"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Quantity of Books..."** />

<input type=**"text"** name=**"image"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Image File Path..."** />

<input type=**"text"** name=**"author"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Books Author..."** />

<input type=**"text"** name=**"category"** class=**"form-control input"** style=**"height:30px;width:20%;min-width:175px;margin-bottom:1.5%"** placeholder=**"Category The Book's In..."** />

<input type=**"submit"** name=**"submit"** class=**"form-control input"** value=**"Submit"** style=**"width:90px"** />

</form>

The code here consists of inputs-boxes which has a value pre-defined in it, this then upon the pressing of the submit button will fill these values with the users input and then set the php as stated above with the value.

## Error Handling

The admin page has no error prevention built into it but I will discuss this in the project management part of this report.

## Internal Documentation

This page has comments to ensure that anybdy who needs to try and understand what each aspect of this code does will understand it without reading into the code.

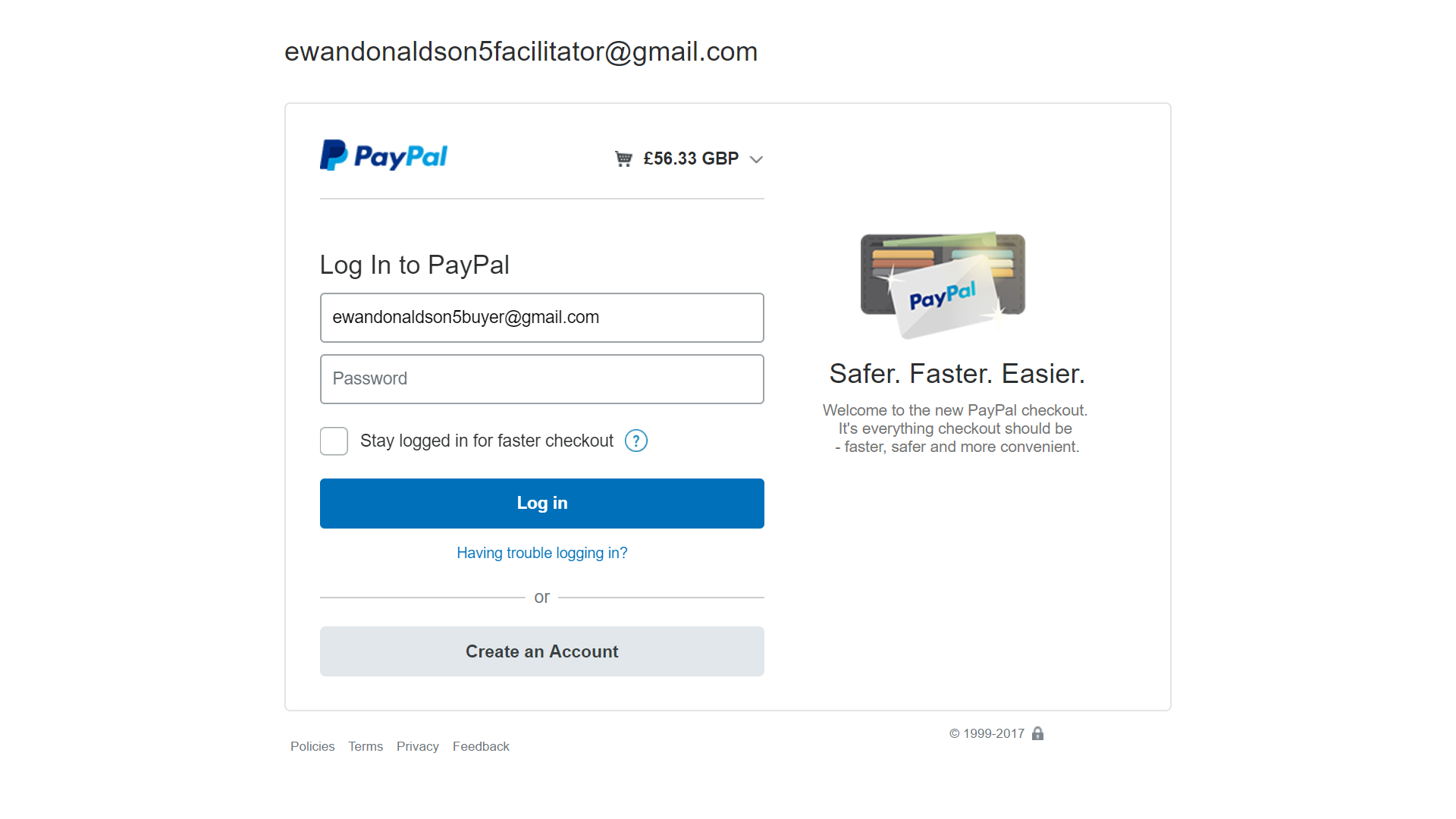
//$insert\_query = "INSERT INTO shop(item\_name,blurb,item\_id,item\_price,item\_qty,image,author,category) VALUES ('test9', 'test9', '471', '54.86', '56', 'test9', 'test9', 'test9'

);";

$insert\_query = "INSERT INTO shop (item\_name,blurb,item\_id,item\_price,item\_qty,image,author,category) VALUES ('$item\_name','$blurb','$item\_id','$item\_price','$item\_qty','$image','$author','$category');";

This comment above the code is an example of what the data would be like within the query this is useful as it can show the programmer an example of what sort of data types would be entered into each field.

# PayPal Payment / <https://www.sandbox.paypal.com/cgi-bin/webscr>



## Source Code Listing

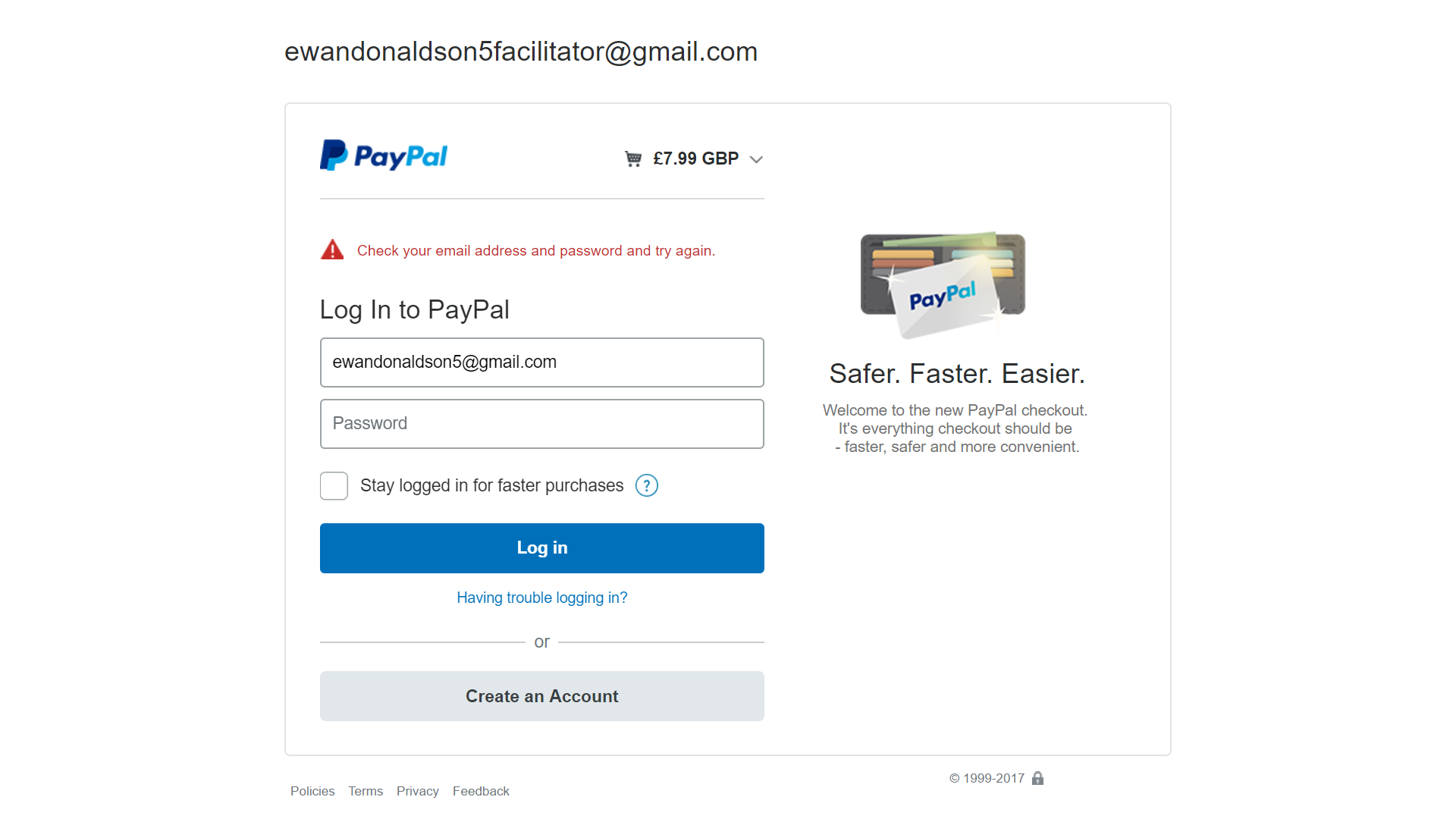
This page is used to pay for the cart through the use of a paypal form. This has been setup using a sandbox, testing, account. I have used a test business account [ewandonaldson5facilitator@gmail.com](mailto:ewandonaldson5facilitator@gmail.com) which is used to receive the payment and a personal account [ewandonaldson5buyer@gmail.com](mailto:ewandonaldson5buyer@gmail.com) which is used to test the purchasing of books.

## User Interface Code Listings

The user interfaces with this paypal form through the use of input-boxes, checkboxes and buttons. The input-boxes are for the user to enter in their email and password, the checkbox is to give the user the ability to stay logged in and there are buttons to login, to register and to got to “having trouble logging in”. These all have specific functions that grant the user access to various parts of paypal.

## Error Handling

The error prevention on this page consist of an error message that is displayed to the user upon entering an email and password that do not match or exist. The error message is in a red font with an error symbol to make it very clear to the user, the benefit of this is that they will be able to clearly see what the error is before making a second attempt.

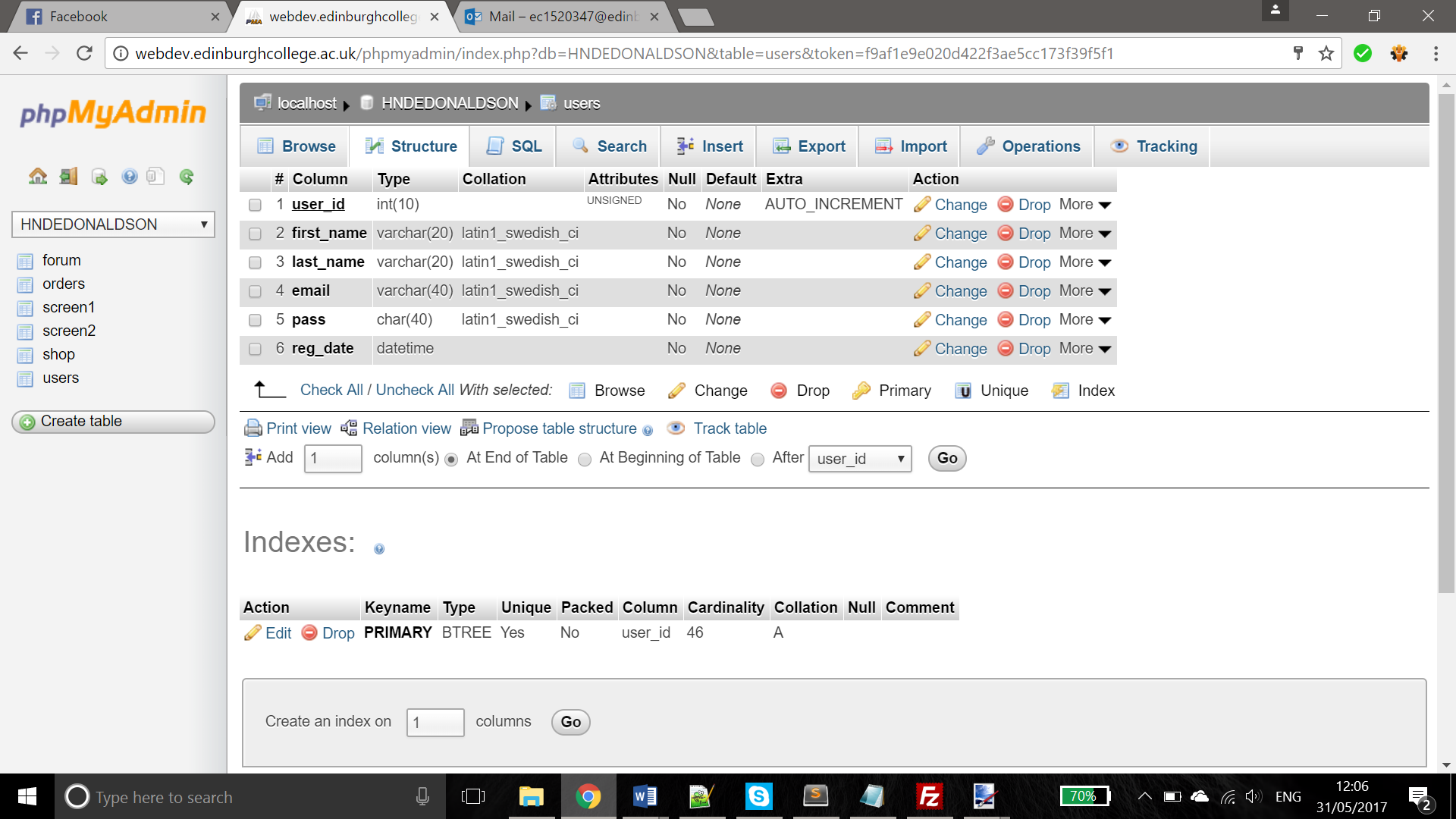


## Internal Documentation

No internal comments exist for this page due to it being a page that I do not have on my server but a page that paypal has control over. Due to this I am unable to put comments on this code however I do not have a need to see this code as I have no desire/requirement to edit it in any way.

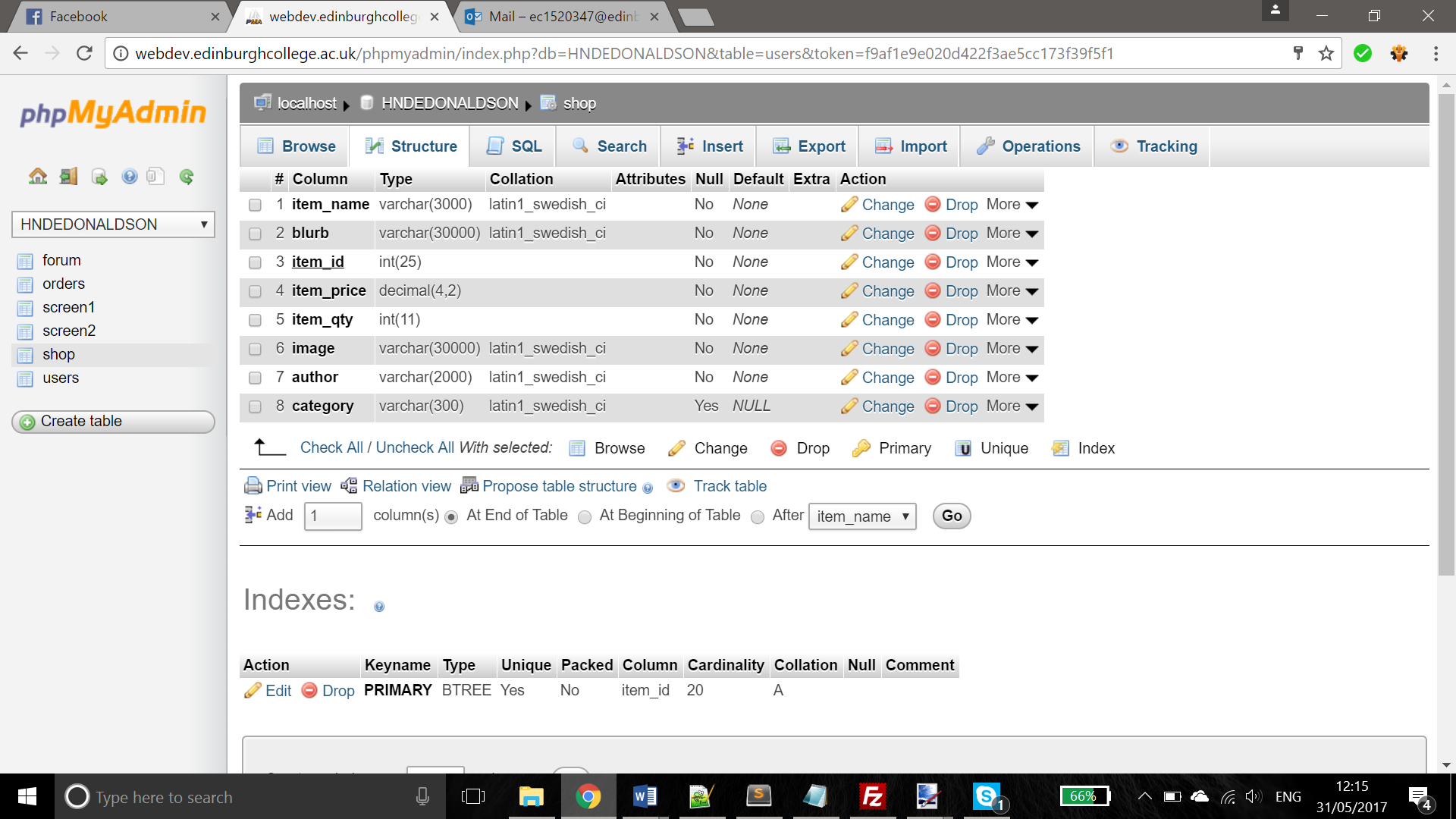
# Database Structure / Content

## Users Table



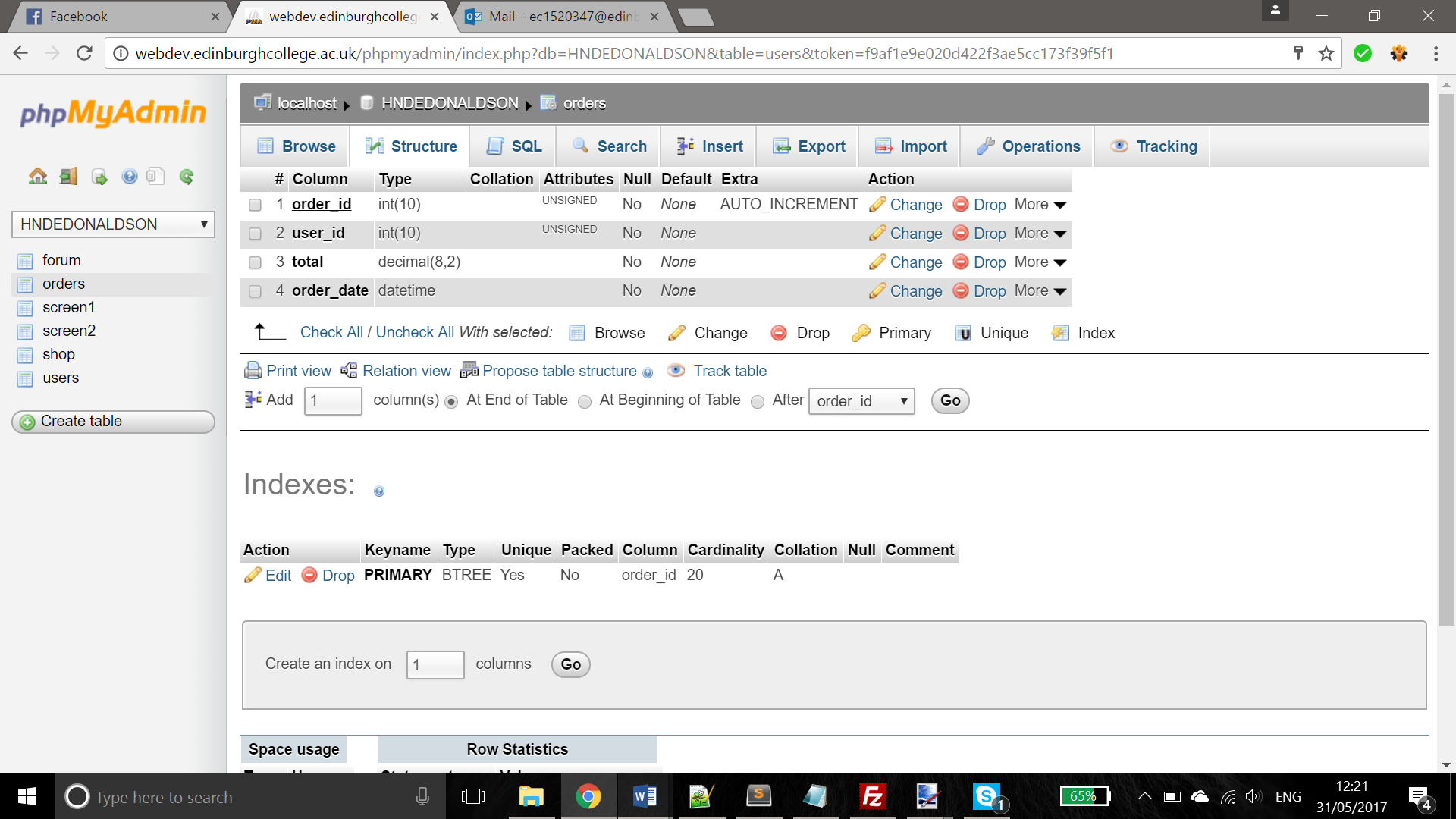
|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Description |
| user\_id | Int | 3 | Identification of the user |
| first\_name | Varchar | 20 | Firstname of the user |
| last\_name | Varchar | 20 | Last name of the user |
| email | Varchar | 40 | Emails of the user |
| pass | Char | 40 | Password for the user, of which is encrypted |
| reg\_date | Datetime | n/a | Registered date and time of account |

## Shop table



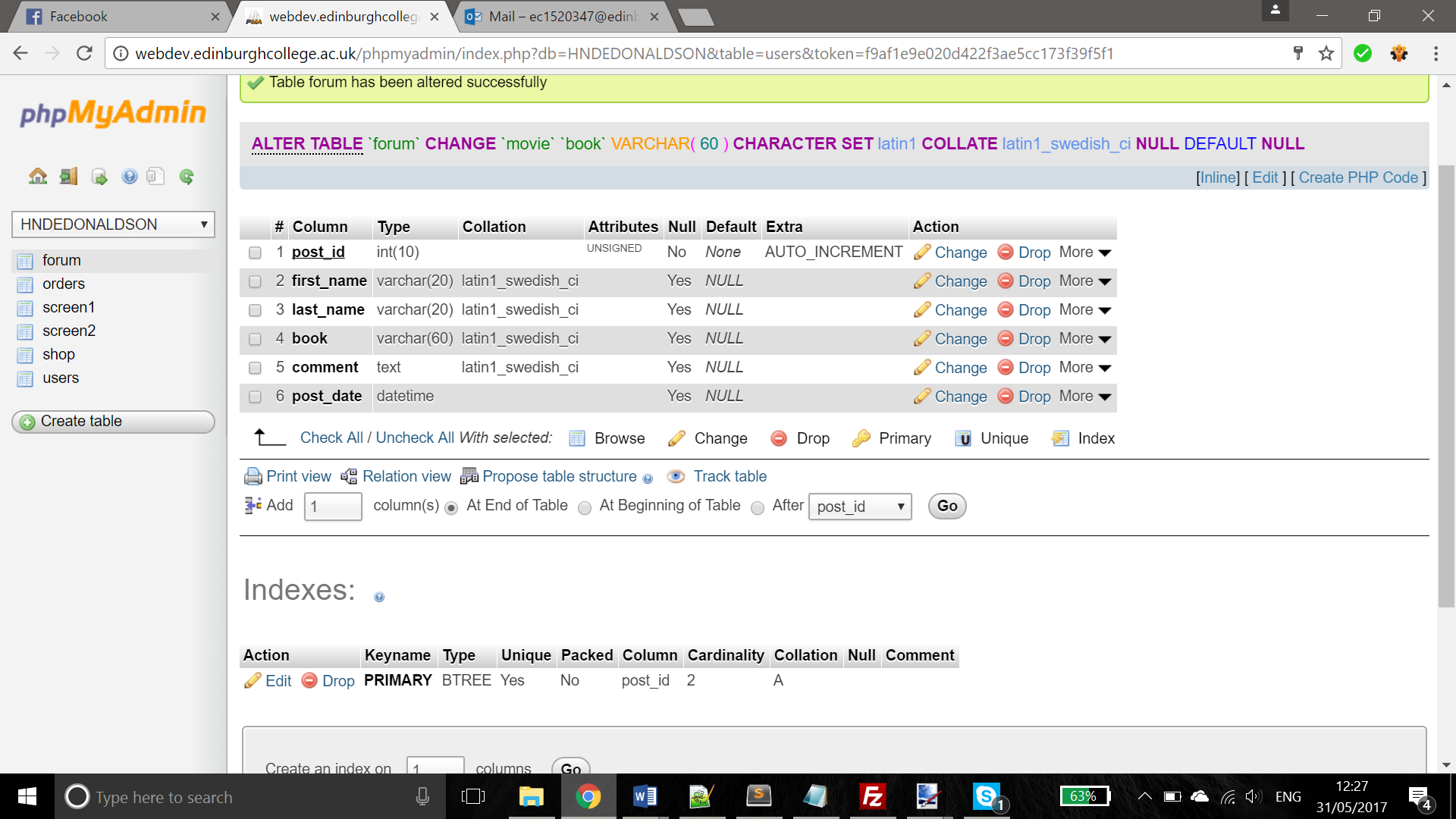
|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Description |
| Item\_name | Varchar | 3000 | Book name |
| Blurb | Varchar | 30000 | Blurb/summary of the book |
| Item\_id | Int | 25 | Unique id of the car |
| Item\_price | Decimal | (4,2) | Price of the book |
| Item\_qty | Int | 11 | Quantity of the book in store |
| Image | Varchar | 30000 | Image file path for the book |
| Author | Varchar | 2000 | The name of the author of the book |
| Category | Varchar | 300 | The category of book the books fall under |

## Orders Table



|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Description |
| Order\_id | Int | 10 | The unique order number |
| User\_id | Int | 10 | The unique number for the user |
| Total | Decimal | (8,2) | Total cost of the order |
| Order\_date | Datetime |  | Date and time of the order |

## Forum Table



|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Description |
| Post\_id | Int | 10 | Unique id of the post |
| First\_name | Varchar | 20 | Users first name |
| Last\_name | Varchar | 20 | Users last name |
| Book | Varchar | 60 | The book the user has put |
| Comment | Text |  | The comment the user has left |
| Post\_date | Datetime |  | The date and time of the post |

# Use of Unfamiliar Constructs and Libraries

<!-- Bootstrap Core CSS -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"css/bootstrap.css"**>

<link rel=**"stylesheet"** type=**"text/css"** href=**"css/bootstrap.min.css"**>

<link rel=**"stylesheet"** type=**"text/css"** href=**"css/bootstrap-theme.css"**>

<link rel=**"stylesheet"** type=**"text/css"** href=**"css/bootstrap-theme.min.css"**>

<link href=**"vendor/bootstrap/css/bootstrap.min.css"** rel=**"stylesheet"**>

<link rel=**"stylesheet"** href=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"**>

<!-- Theme CSS -->

<link href=**"css/freelancer.min.css"** rel=**"stylesheet"**>

<!-- Custom Fonts -->

<link href=**"vendor/font-awesome/css/font-awesome.min.css"** rel=**"stylesheet"** type=**"text/css"**>

<link href=**"https://fonts.googleapis.com/css?family=Montserrat:400,700"** rel=**"stylesheet"** type=**"text/css"**>

<link href=**"https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic"** rel=**"stylesheet"** type=**"text/css"**>

<link href=**'http://fonts.googleapis.com/css?family=Oswald'** rel=**'stylesheet'** type=**'text/css'**>

<script src=**"https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"**></script>

<script src=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"**></script>

</head>

These pieces of code show the links to different online and internally saved sources which assist me with styling scaling and general styles within my website. The main library used was bootstrap which is a downloadable source of code which is used to assist with scaling of a page as well as a few styles. Bootstrap consists of multiple files, of which you can choose which you use, to work on your website. Online fonts are also accessed which allow me to get the website presentable, because if I had used default fonts like times new roman for example it may have looked too simplistic and formal which is not the foundational upon which I want my site to be based around. Google maps was used in my website which is a third-party API (application program interface) which had allowed me to embed a google maps into the webpage using an i-frame.

<div style=**"margin-left:5%"**>

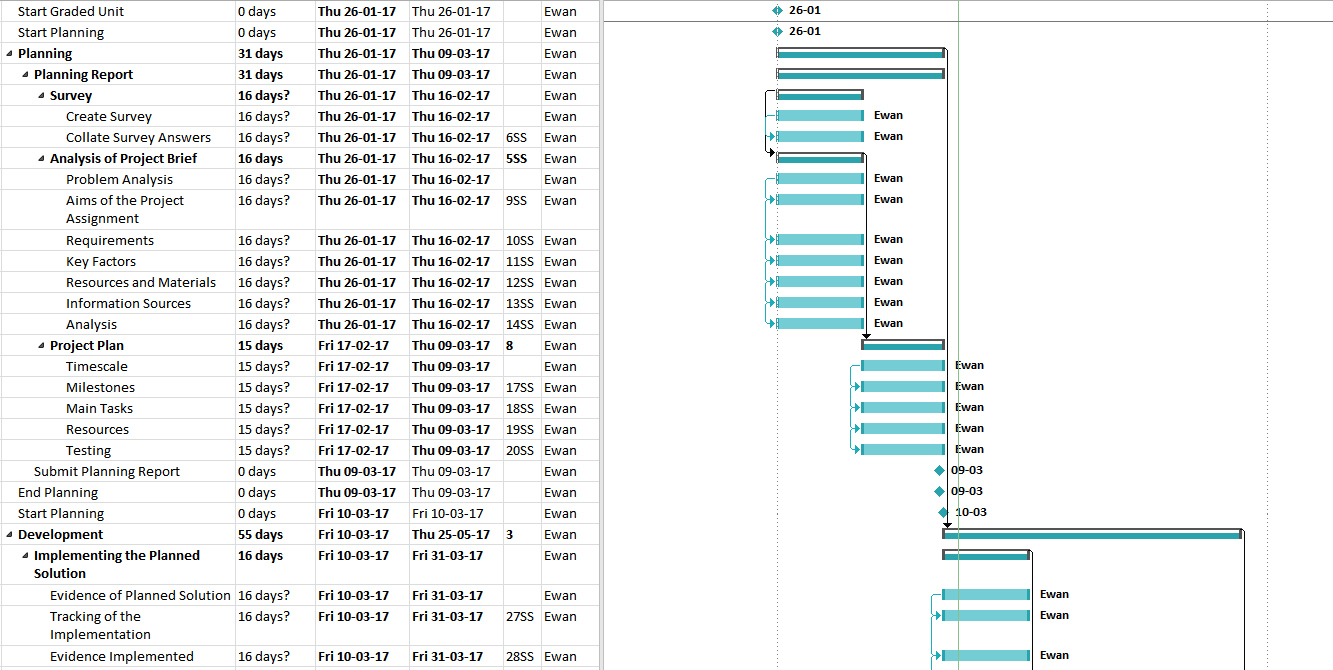
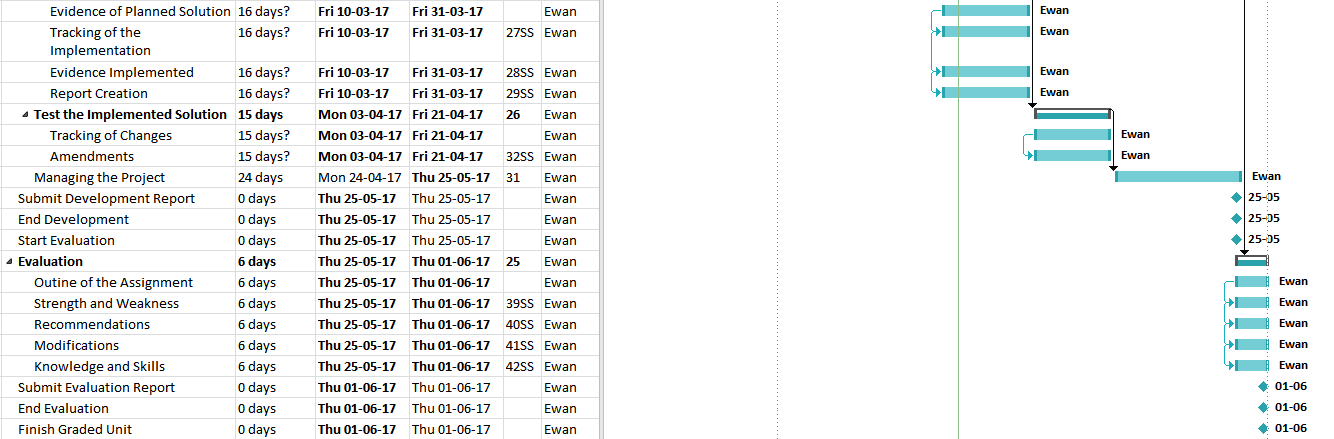
<iframe src=**"https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d1119.5003057725348!2d-4.281098707145652!3d55.8626557959889!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x0%3A0x0!2zNTXCsDUxJzQ1LjYiTiA0wrAxNic0OS45Ilc!5e0!3m2!1sen!2suk!4v1496072204729"** width=**"95%"** height=**"450"** frameborder=**"0"** style=**"border:0"** allowfullscreen></iframe>

</div>

Paypal was also used within the website which is also a third-party API, this grants me that ability to embed a link into my website and to them link it to the paypal form which would be generated from the page the user was previously on, cart.php. Once there the user is taken through stages to make the purchases and then a conformation of the purchase is created and display whilst still on a paypal site.

Whilst creating my website some bootstrap online free to use templates were used. The navigation bar styles were taken from a bootstrap template then altered slightly to assist with the screen.

# Project Management

The following image is the gantt chart from the planning stage.

I followed the time scale of this gantt chart as well as possible although there were some causes for additional time to be taken such as strikes within the college which therefore meant deadlines had to be stretched.

If I had more time I would ensure that the admin page would work fully as of now it currently doesn’t send the querys to the database to be ran for a reason of which is unknown. Also if I had more time I would add some error handling in the admin page such as if the user\_id already exits it would inform the admin using the page so they can either change their chosen user\_id or view the record with that id and remove it if possible.

# Testing

I have decided to test certain aspects of my code using normal, extreme and exceptional tests. Each of these test types are a solid way of testing how an input works for a user. Normal data refers to typical data that should definitely work, extreme data refers to data that should be on the limits of how the parameters but should still work and exceptional data refers to data that is out with the parameters and should return an error but not crash the website. 3 of each test type should be used to ensure a fair set of results are outputted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function being tested | Test Type | Test Data | expected reSult | actual result |
| Registers Password and Confirm Password Input Boxes | Normal | Pass and Pass | Should work | Does Work |
| Normal | Phrase and Phrase | Should work | Does Work |
| Normal | Password123 and Password123 | Should work | Does Work |
| Extreme | Pass@l21 and Pass@l21 | Should work | Does Work |
| Extreme | A@@{3 and A@@{3 | Should work | Does Work |
| Extreme | 123876 and 123876 | Should work | Does Work |
| Exceptional | Leave them empty | Displays error message | Displays an error message |
| Exceptional | A forty-one long character string | Displays error message | Displays an error message |
| Exceptional | Put in a space | Displays error message | It allows it |

From this test set we can see that the password input boxes are very good with handling certain functions however they did allow a password that had a space in it which I did not expect therefore there is a flaw in the input boxes which would be fixed later.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function being tested | Test Type | Test Data | expected reSult | actual result |
| Search Function | Normal | geo | Should display multiple books | Displays multiple books |
| Normal | da | Should display multiple books | Displays multiple books |
| Normal | cook | Should display multiple books | Displays multiple books |
| Extreme | David | Should display multiple books | Displays multiple books |
| Extreme | Geography | Should display multiple books | Displays multiple books |
| Extreme | Leave it blank | Should display all books | Displays all books |
| Exceptional | @ | Display no books | Displays no books |
| Exceptional | % | Display no books | Displays all books |
| Exceptional | 1234567 | Display no books | Displays no books |

From this test data, we can tell that the search function works almost as expected but when a percent symbol is put in it searches for all the books, the reason for this is because in php a percentage symbol also means all therefore all books are searched.

For this filter, I will test it by presenting the expected output with each option being chosen, and when a new book with a new category has been added, what will happen when you select that category.

|  |  |  |  |
| --- | --- | --- | --- |
| Function being tested | Test Data | expected reSult | actual result |
| Filter Dropdown | History | Should display all history books with corresponding popups | Displays all books from the history category |
| Cooking | Should display all Cooking books with corresponding popups | Displays all books from the Cooking category |
| Best Sellers | Should display all Best Sellers books with corresponding popups | Displays all books from the Best Sellers category |
| Geography | Should display all Geography books with corresponding popups | Displays all books from the Geography category |
| 2017 books | Should display the new book added | Displays new book added |
| Test8 | Should display the book in category test8 | Displays test8 book |

From this test data, we can tell that the filter works as well as desired, there are no flaws with it as there are no errors arising from it, this could be due to the fact it is not driven through the user of user inputs.

# User Guide

|  |  |
| --- | --- |
| Page | Guide |
|  | Login.php This page has the user input their username and password then press login to submit their inputs.  They can also click on “create an account” to go to register page. |
|  | Register.php This has user inputs for first name, last name, username, password and password confirmation.  Then the user clicks the submit to submit their entries.  Go to “Login Here” to go to login.php |
|  | Display.php This has the main page content such as the books with the filter and the search function. The about section, and location on the map.  There is also the navigation bar to move throughout the pages. |
|  | Admin.php This has all the inputs the admin requires to input add another book to the database. |
|  | Added.php This page appears once the user has clicked the +cart page on the display.php page. This tells the user they have added the book to the cart and give them options to add another book or to continue with the purchase. |
|  | Cart.php This page has the listings off all books the user has added to the cart, with this they can type an amount of each book then update their cart. From here they can click the paypal buy now button to pay for their cart. |
|  | Paypal form This page allows the user to sign into paypal and to purchase the books. Upon payment they are shown a message with the completion of the purchase. |
|  | Goodbye.php This page is used to sign out the user and end the session, the user presses sign out on any navbar to do so. |