Development Log

Error
Solution
Iterative Testing

Date: 19/03/2021

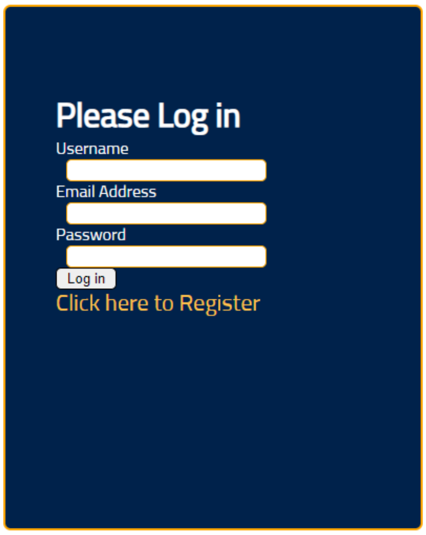
Task: Creating a login page

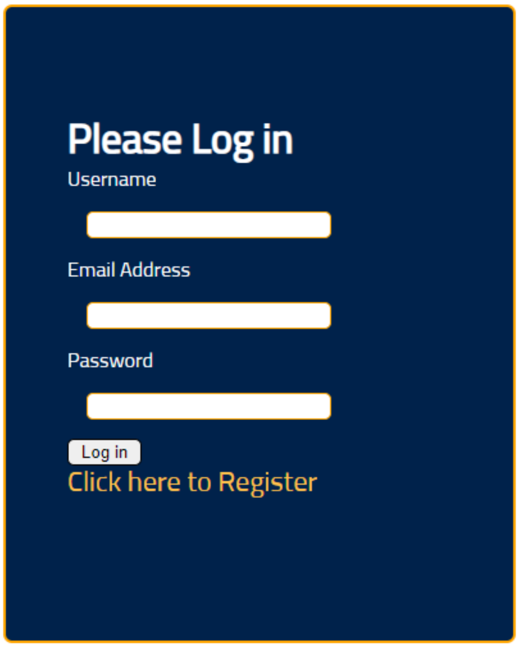
```
<div id="account">
            <form id="loginForm" action="DoLogin.php" method="post">
                <h1>Please Log in</h1>
                <p id="usernameLabel">Username</p>
                <p><input class="textInput" type="text" name="username"
/></p>
                <p id="usernameLabel">Email Address</p>
                <p><input class="textInput" type="text" name="email"
/></p>
                <p id="passwordLabel">Password</p>
                <p><input class="textInput" type="password"
name="password" /></p>
                <p><input class="standardButton" type="submit"
value="Log in" /></p>
                <a href="Register.php" target="_blank">Click here to
Register</a>
            </form>
            <p>
                <?php
                if (isset($_REQUEST["error"])) {
                    echo $_REQUEST["error"];
                }
                ?>
            </p>
        </div>
```

As a first step, I created a login page for my project using HTML and CSS. One problem that I have encountered is that when I opened my project at home on my own laptop, the margins and paddings were different and the inputs were very close to each other. For example, I made my login div to be at the centre of the page by using margin in CSS, but when I opened my project on my laptop the login div was moved to the side. So I have solved this problem by setting the margins as a percentage, so it changes dynamically based on the screen size.
In addition, I have implemented a very basic processing page for the login using PHP.

```
<?php
$valid = True;
$vals = array();
foreach (["username", "email"] as $val) {
    if (!isset($_POST[$val]) || strlen($_POST[$val]) == 0) {
        $valid = False;
    }
    $vals[$val] = $_POST[$val];
}
$p = http_build_query($vals);
if ($valid == True) {
    header("Location: Welcome.php");
    exit;
} else {
    header("Location: index.php?error=Failed Validation&" . $p);
    exit;
}
```

Another error I have encountered when using PHP is that when I wanted to check the length of the inputs I used sizeof() function instead of strlen() function. So I fixed the error by researching online about the method of checking input lengths in PHP.





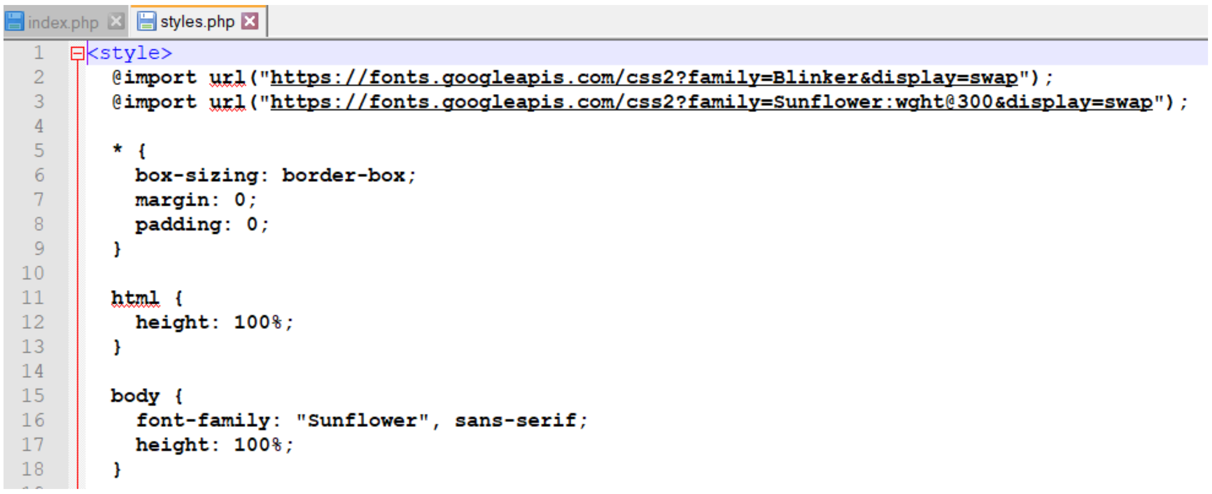
Date: 22/03/2021

Task: Solving CSS issue

While I was working on the style of the login page, I realized that none of the changes I made on the CSS file is being applied to the web page. The first thing I did to solve the problem is that I checked if I correctly linked the styles.css file with my PHP file, but I found out that it was correctly linked and there was no issue with that. After a little research, I found out that the issue was browser caching. This means that the browser requests the stylesheet file once from the server and saves it so it doesn't need to request it every time we reload the page, which can improve the performance. Therefore, some of the changes weren't applied properly. As a result, I discovered that I need to change the extension of my styles file from .css to .php and then link it to my HTML using PHP with the following code:

```
<?php
session_start();

require_once("styles.php");
}
?>
```

To test if this solution works, I imported a Google font to my styles file and then applied it to the elements, and it worked perfectly fine.



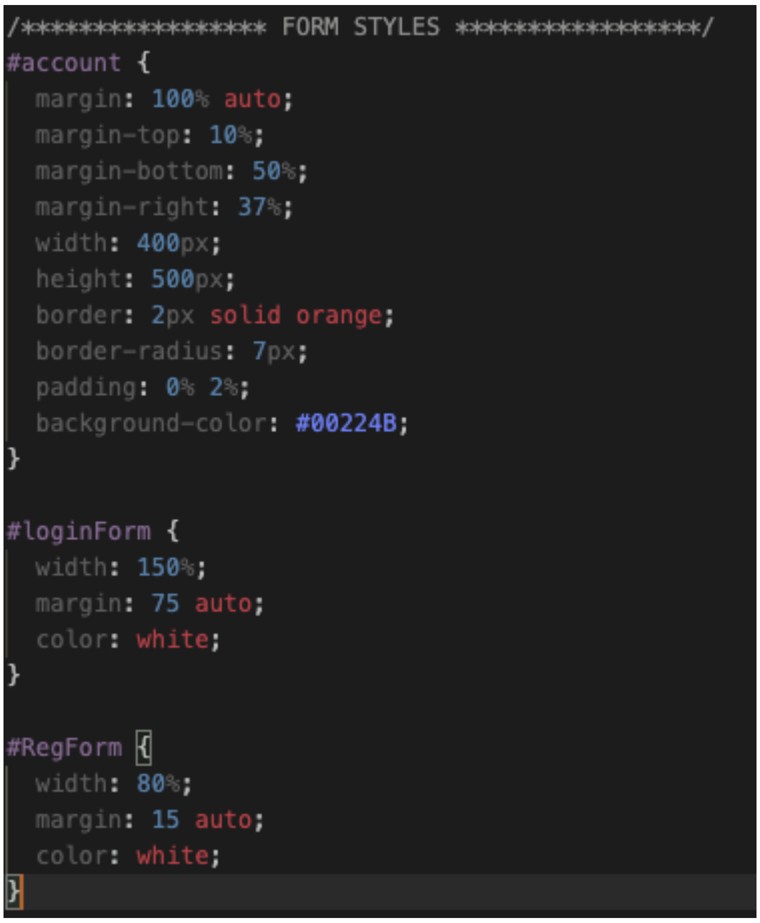
index.php styles.php

```
<style>
  @import url("https://fonts.googleapis.com/css2?family=Blinker&display=swap");
  @import url("https://fonts.googleapis.com/css2?family=Sunflower:wght@300&display=swap");

  * {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
  }

  html {
    height: 100%;
  }

  body {
    font-family: "Sunflower", sans-serif;
    height: 100%;
  }
```

Date: 24/03/2021

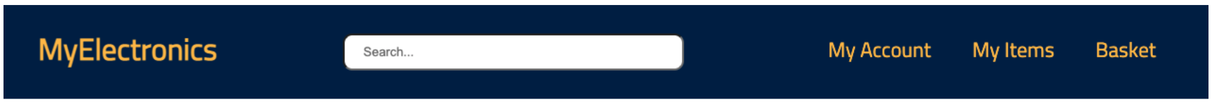
Task: Creating Registration and a Welcome page

After I have done the login page, I realised that making a registration page is not very difficult to implement as it's almost similar to the login page, but it only needs a few more inputs from the user. Therefore, I created another file and called it Register.php, then I reused all the code I have written for my login page to build my registration page. However, I added a couple more inputs such as first name and last name. After I added those inputs, some of the elements moved outside the div. Therefore, I needed to set a separate id for the registration form rather than having the same id as the login form. Then I changed the paddings and margins of those elements until they were placed properly inside the div, also I was reloading the page every time I made a change to make sure that it was working properly.



```
/***************** FORM STYLES *****************/
#account {
  margin: 100% auto;
  margin-top: 10%;
  margin-bottom: 50%;
  margin-right: 37%;
  width: 400px;
  height: 500px;
  border: 2px solid orange;
  border-radius: 7px;
  padding: 0% 2%;
  background-color: #00224B;
}

#loginForm {
  width: 150%;
  margin: 75 auto;
  color: white;
}

#RegForm {
  width: 80%;
  margin: 15 auto;
  color: white;
}
```

Furthermore, I have started working on the welcome page. The first thing I did for the welcome page was the header and the navigation bar. As I'm new to the language I did not have the knowledge to build a header and a navigation bar. Therefore, I have researched the ways, methods and HTML tags that can be used to build such a thing. Later on, I figured out how to build the header and the navigation bar after putting together all the information I got from my research. I learned that to build a header and a navigation bar I will need to use the header, nav, ul and li tags. This is my final result:



Date: 26/03/2021

Task: Adding validations to the login form

To make sure that the user types the correct thing for each input, I needed to implement some validations on those inputs, specifically, email address and password.

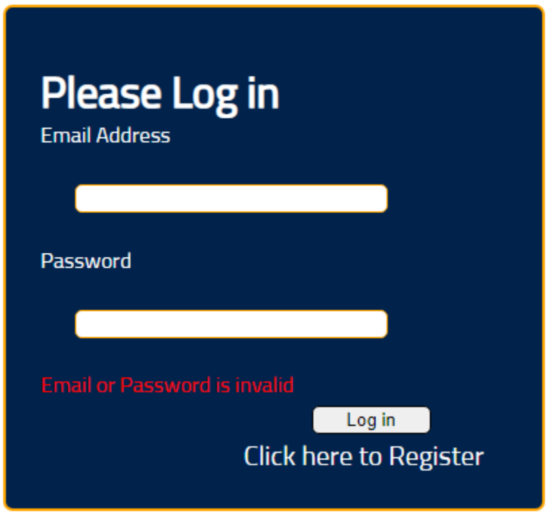
At the start, I knew how to check the length of the inputs and whether the user typed something or not, but I wanted to make it more advanced by validating the email and making sure that the user types it correctly in the right format. In addition, for the password I wanted the user to include at least one upper case letter. at least one lower case letter, and number. However, I did not know how to implement those validations using PHP. After a little research online, I discovered that I need to use a function called “preg_match” which allows me to perform a regular expression match. Then I used an if statement to check if the user typed something then I checked the inputs against the validation rules that I have set using preg_match, but it wasn’t working for some reason. This is how my code looked:

```
if (!isset($_POST["password"]) || $isUppercase || $isLowercase || $atLeastOneNumber ||
strlen($_POST["password"]) < 8) {
        $errors[] = "password";
}
```

However, after asking and investigating, I realised that I need to add a “!” before each statement which at the beginning I thought I only needed to put at the start of the statements. As a result, this how my code looked after:

```
if (!isset($_POST["password"]) || !$isUppercase || !$isLowercase || !$atLeastOneNumber ||
strlen($_POST["password"]) < 8) {
        $errors[] = "password";
}
```

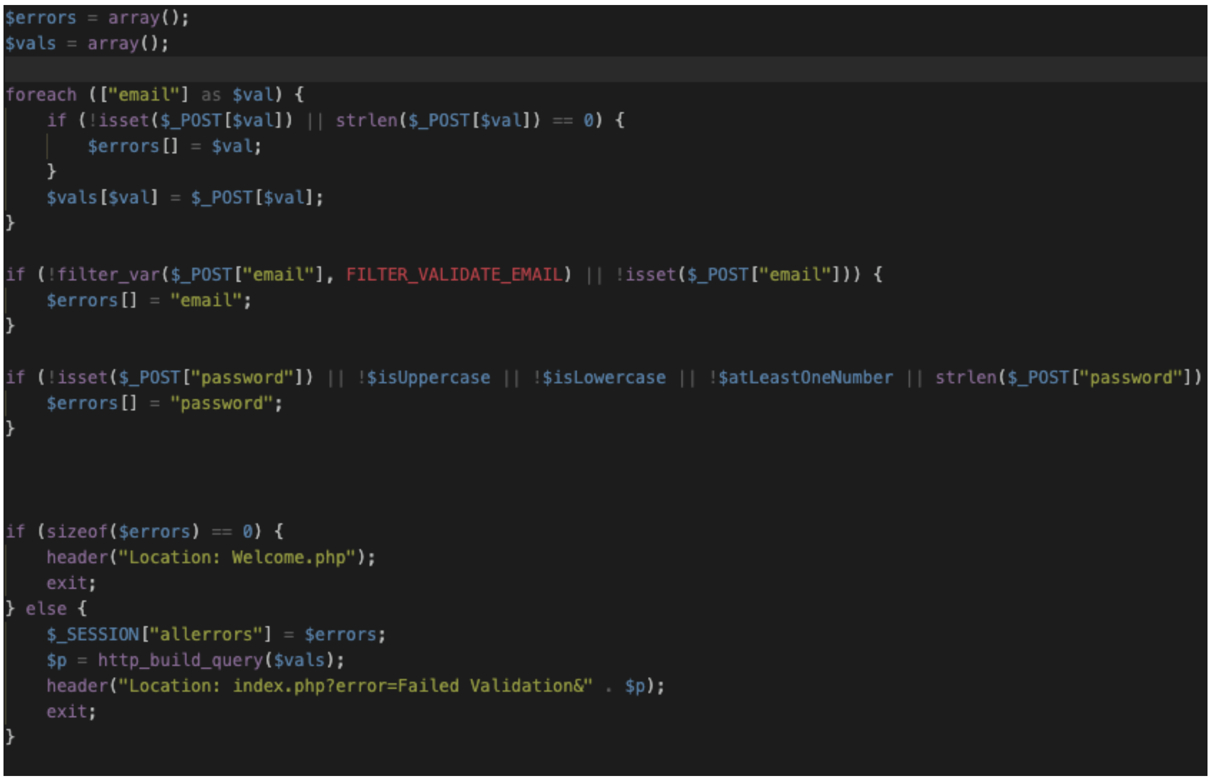
Finally, to check if it was working properly I tried to print a statement informing me whether the input I typed in was valid or not valid.


Please Log in
Email Address
Password
Email or Password is invalid

Date: 29/03/2021

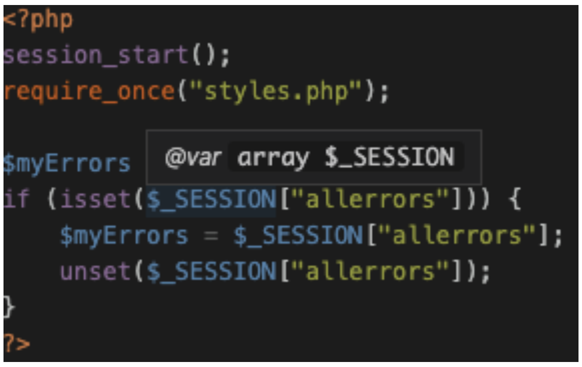
Task: Printing a message to the user if an input is invalid

On this day, I wanted to make my login form more sophisticated by making it inform the user which input is invalid if they typed in something incorrectly. In the beginning, my only issue was that I wasn’t sure how to implement this feature, but after asking and researching I discovered that I will need to use “session_start()”. This allows me to store user information across multiple pages. Therefore, I can call it by assigning a session variable using “$_SESSION[]”. However, before I used these functions, I needed to store all the errors in an array in my process login file. This works by checking which inputs are incorrect using the validation rules and then if it doesn’t meet those rules I can add it to the errors array.



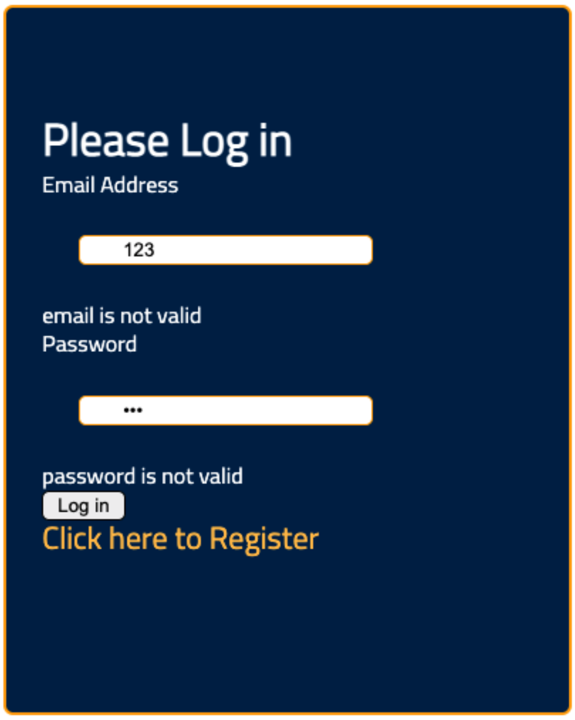
```
$errors = array();
$vals = array();

foreach (["email"] as $val) {
    if (!isset($_POST[$val]) || strlen($_POST[$val]) == 0) {
        $errors[] = $val;
    }
    $vals[$val] = $_POST[$val];
}

if (!filter_var($_POST["email"], FILTER_VALIDATE_EMAIL) || !isset($_POST["email"])) {
    $errors[] = "email";
}

if (!isset($_POST["password"]) || !$isUppercase || !$isLowercase || !$atLeastOneNumber || strlen($_POST["password"])
    $errors[] = "password";
}



if (sizeof($errors) == 0) {
    header("Location: Welcome.php");
    exit;
} else {
    $_SESSION["allerrors"] = $errors;
    $p = http_build_query($vals);
    header("Location: index.php?error=Failed Validation&" . $p);
    exit;
}
```

The code in the index.php file:



```
<?php
session_start();
require_once("styles.php");

$myErrors
if (isset($_SESSION["allerrors"])) {
    $myErrors = $_SESSION["allerrors"];
    unset($_SESSION["allerrors"]);
}
?>
```

However, after calling the session variable in my index.php page, I needed to store it in a new array so I could print the message to the user, but I found out that those variables are being stored in a 2D array and I wasn't able to tell how. Later on, I found out that when you assign the session variable to the new array in the index.php page you will need to assign it as a variable array and not as a value inside the array. Finally, I ran a few tests by typing in incorrect values for my inputs and seeing if it prints anything, and it was working perfectly. This is my final result. As a final step, I applied all this code on my registration page as well.


Please Log in

Email Address

123

email is not valid

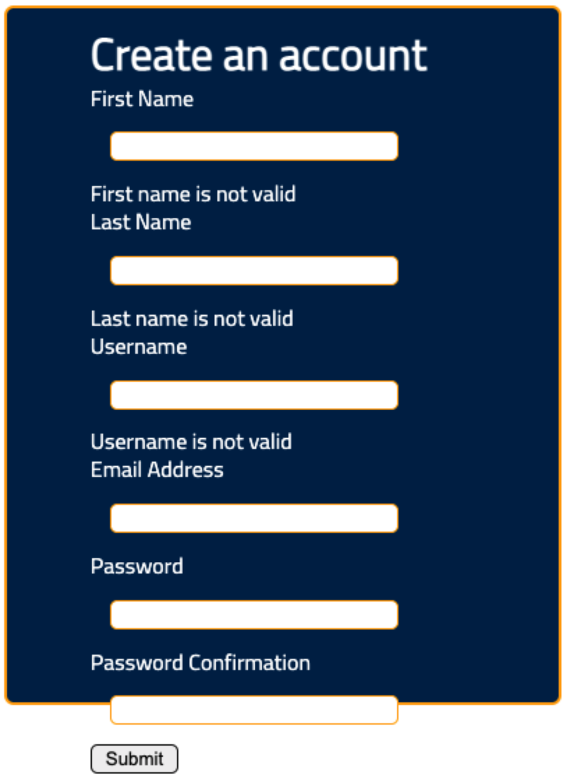
Password

•••

password is not valid

Log in

Click here to Register


Create an account

First Name

First name is not valid

Last Name

Last name is not valid

Username

Username is not valid

Email Address

Password

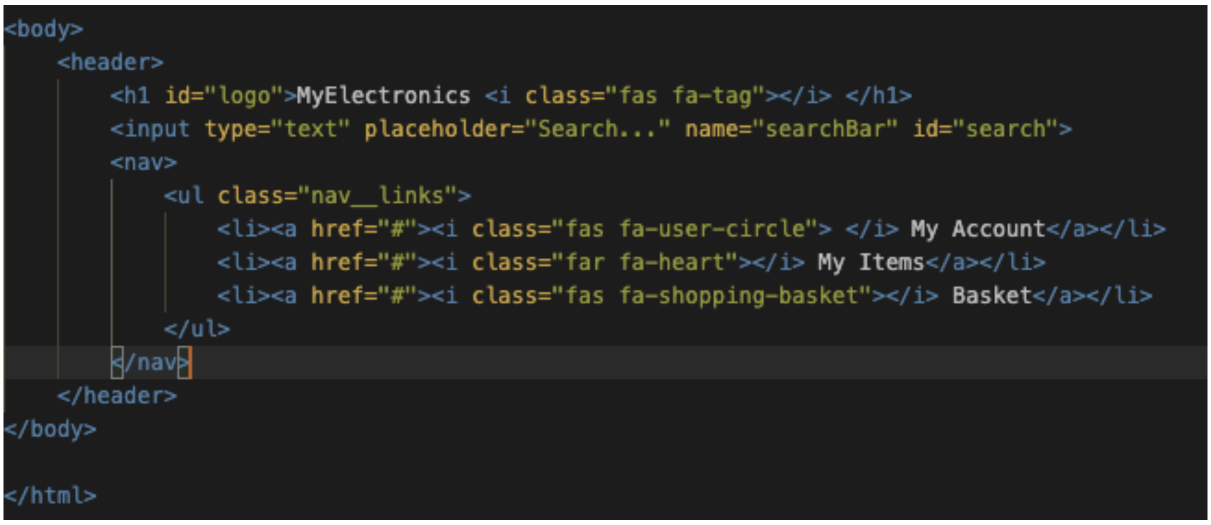
Password Confirmation

Submit

Date: 31/03/2021

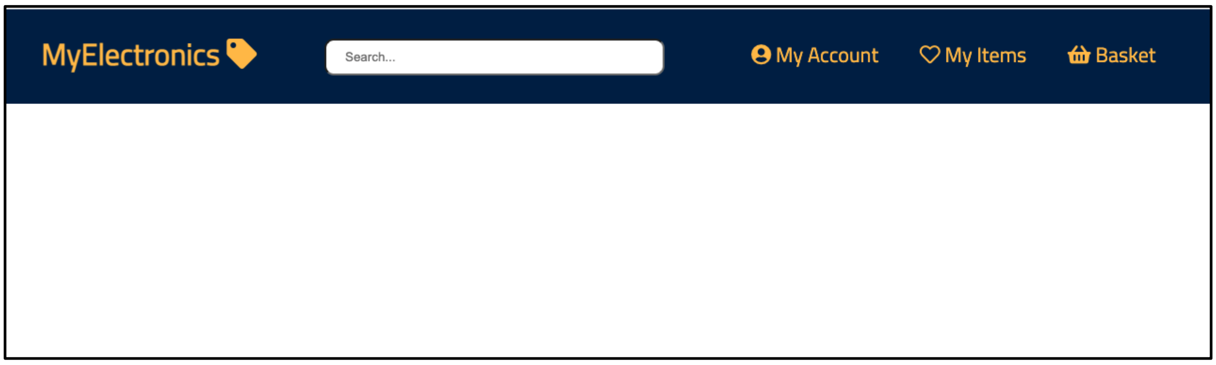
Task: Adding icons to the Welcome Page

I wanted to make the Welcome page more appealing to the user by adding icons and a logo in the header and the navigation bar. However, I have a very little experience with designing etc. But I did not know how to add the designs and icons I wanted by coding them. So after researching, I discovered that I could download already made icons and then import them into my code so I can add them to my web page. After downloading the icons and all the designs from a website called Font Awesome, I added them to my project file, then I used their website to find the exact line of code I needed for a specific icon. However, I wasn't able to find out whether I would need to use CSS to position these icons in the correct place or I can just add those lines using HTML. After a few tries and tests by adding the lines of code in different places in my code, I discovered that I can simply add them next to the a tag inside the list item which was much easier than I thought.



```
<body>
    <header>
        <h1 id="logo">MyElectronics <i class="fas fa-tag"></i> </h1>
        <input type="text" placeholder="Search..." name="searchBar" id="search">
        <nav>
            <ul class="nav__links">
                <li><a href="#"><i class="fas fa-user-circle"> </i> My Account</a></li>
                <li><a href="#"><i class="far fa-heart"></i> My Items</a></li>
                <li><a href="#"><i class="fas fa-shopping-basket"></i> Basket</a></li>
            </ul>
        </nav>
    </header>
</body>

</html>
```

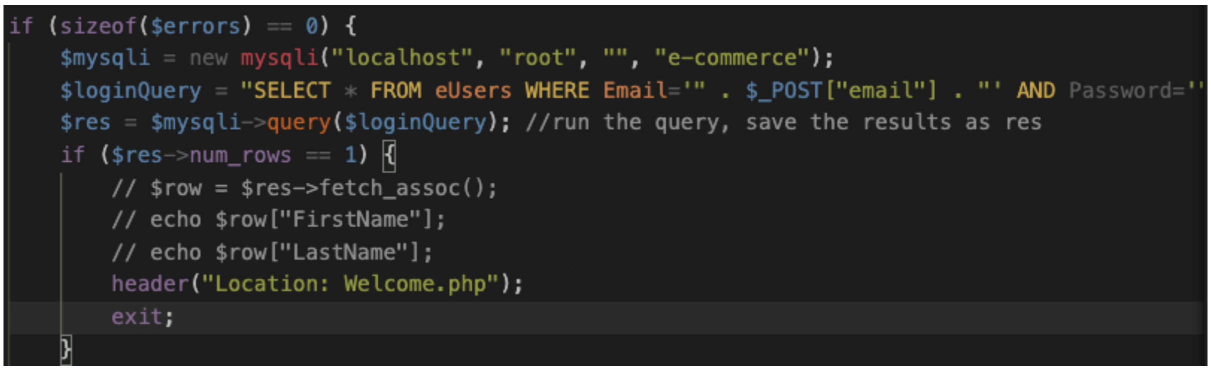
In the end, this is my final result:



Date: 19/04/2021

Task: Creating a database table and Linking MySQL with PHP

I have started creating the database, and as a first step, I have created a users table using MySQL. The first time I created the table the website crashed and the table got deleted. However, I closed the website and opened it again and I repeated the process and it worked. After creating the table, I needed to decide what columns I will need in my table, so I used my design document which has the database planning to help me add the correct columns. While I was adding the columns, I realised that the number of columns in the table and the number of inputs in the registration form must be the same because when a new user signs up he/she will need to enter all the details required so the database table can't be empty and left blank. However, I only had 5 inputs in the registration form which were First name, last name, email, password and password confirmation, so I realised that I have completely forgotten to add the other inputs that I have planned to add in the registration form such as phone number and address details. So I decided to go back a step and complete my registration form and add all the other inputs required.
In addition, I have linked the database with PHP using a class called mysqli, and then I have tested it by printing the line of the query MySQL makes when a user signs in. This is a screenshot of the test and it shows that the link was working.

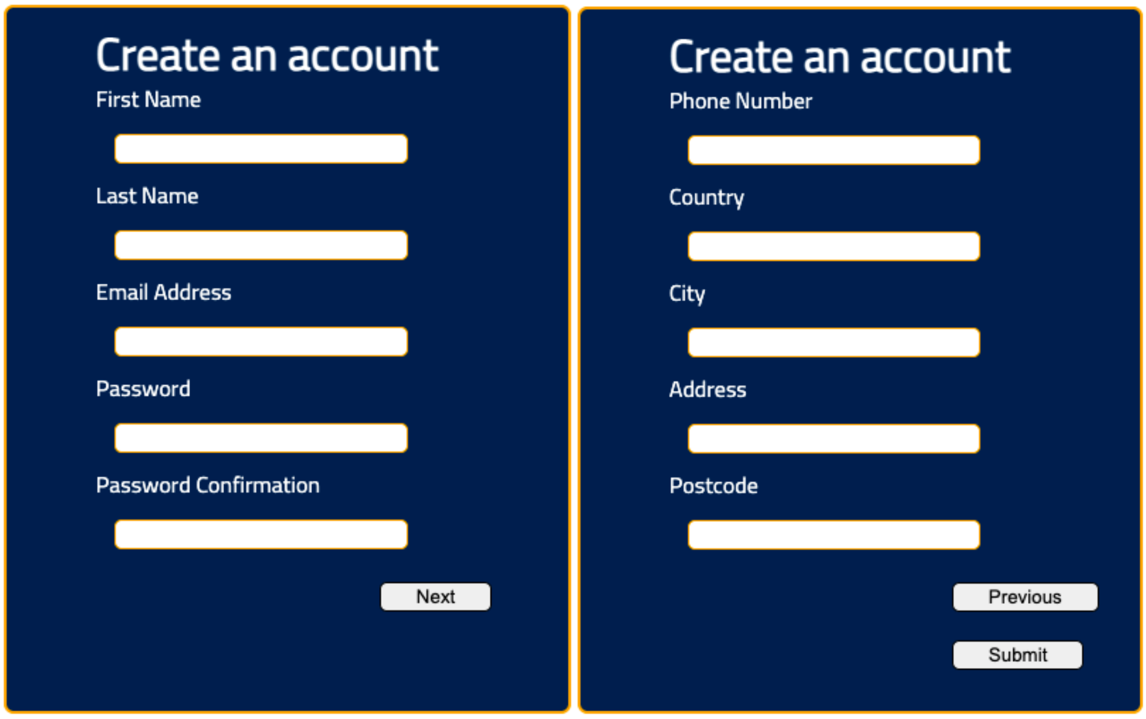


```
if (sizeof($errors) == 0) {
    $mysqli = new mysqli("localhost", "root", "", "e-commerce");
    $loginQuery = "SELECT * FROM eUsers WHERE Email='" . $_POST["email"] . "' AND Password='
    $res = $mysqli->query($loginQuery); //run the query, save the results as res
    if ($res->num_rows == 1) {
        // $row = $res->fetch_assoc();
        // echo $row["FirstName"];
        // echo $row["LastName"];
        header("Location: Welcome.php");
        exit;
    }
```

Date:21/04/2021

Task: Completing the Registration Form

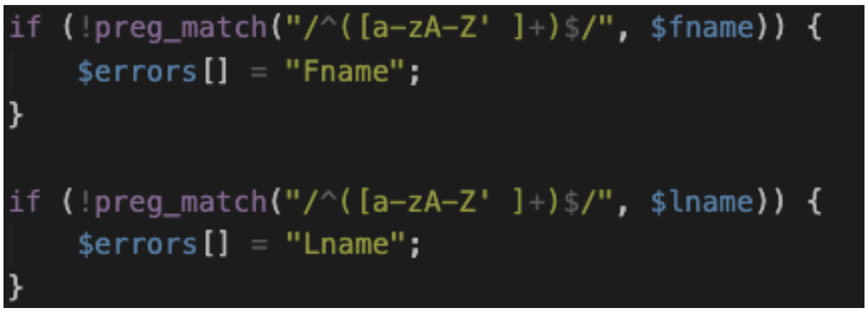
I started by adding the additional inputs that I have planned for my reg form to have which are phone number, country, city, address and postcode. I wanted the inputs to be displayed to the user in 2 parts instead of being in one big div which might be confusing to the user. The first part is the name, email and password and then the second part is the inputs I mentioned above. However, I knew that I had to use CSS and JS to hide and show the divs, but I wasn't sure whether I should have created another form for the second part, or created another div inside the form. So after I did my research and asked people I discovered that I had to create another div inside the form for part 2. In addition, I was advised to use jQuery as it's more efficient than JS for this particular task I want to implement.
After I downloaded jQuery and added it to my project, I used a function called “toggle” inside the function I created to hide and show the divs. However, this solution wasn't working and I wasn’t able to find out the reason, but after several tests, I figured out that I need to change the input type from “sumbit” to “button” for my function to work. So I changed the type of the input but it still wasn’t working and I completely had no idea why. After a while, I realized that I did not import jQuery to the PHP file, and that's why my function wasn't working.
Furthermore, I created another input with type input in the div for part 2 to enable the user to go back to part 1 and I called the input “Prevoius”. Finally, I did a few tests to verify that the solution was working by just clicking the “Next” and “Previous” buttons several times to ensure that it shows and hides the divs as required.



Date:23/04/2021

Task: Validating additional inputs and connecting the database

To make sure that the code is unbreakable, I needed to validate the additional inputs. So I researched online on how to validate the phone number and the postcode, however, for country, city and address I used the same validation as the first name and last name as they only accept letters. After that, I wanted to test the validations by entering correct and incorrect information, but I realised that the validation of the first name, last name, country, city and address inputs weren't working even though I entered the correct information, it didn't take me to the welcome page. I did not know what the reason was, therefore, I decided to change the validation rule by using a different regex expression by searching online, and it worked properly. This is a screenshot of the new expression:



```
if (!preg_match("/^([a-zA-Z' ]+)$/", $fname)) {
    $errors[] = "Fname";
}

if (!preg_match("/^([a-zA-Z' ]+)$/", $lname)) {
    $errors[] = "Lname";
}
```

Finally, after I validated all the inputs, I linked the registration form with the database. The process was almost similar to the login form, however, I only needed to change the SELECT query to an INSERT query so when the user registers for the first time his/her details get added to the database. I tested it by registering with a few different emails and names to make sure that it's being added correctly to the database.

Date:26/04/2021

Task: Prepared statements and User Object Class

I researched prepared statements and why they are useful as well as how I should use them. I found out that they prevent SQL Injections and make the queries more secure. However, I wasn't sure how to implement it, so I kept researching until I figured out how to use prepared statements for the INSERT query and I implemented it in my registration form. After that, I tested it by creating an account and checking if it gets added to the database. As a result, the account I created got inserted into the database and it worked properly.
On the other hand, I knew it would be slightly different for the login form as we use a select query for the login. Therefore, I searched how to use prepared statements with a select query, and I discovered that it's pretty much the same method as the insert query plus that I had to add a function called get_result() to store the results of the query, so we can check it for the user to sign in.

```
$mysqli = new mysqli("localhost", "root", "", "e-commerce");
$regQuery = "INSERT INTO `eusers` (`UserType`, `FirstName`, `LastName`,
`Email`, `Password`, `phoneNumber`, `Country`, `City`, `Address`,
`Postcode`, `loginDate`) VALUES ('1', ?, ?, ?, ?, ?, ?, ?, ?, ?,
current_timestamp())";
$stmt = $mysqli->prepare($regQuery);
$stmt->bind_param("sssssssss", $fname, $lname, $email, $hashedPassword,
$pNum, $country, $city, $address, $postcode);
$stmt->execute();
$stmt->close();
```

Furthermore, I created a User object class to make it easier for me to retrieve data about users by just calling functions such as getFirstName() etc. I included several variables such as user id, user type, first name, last name and email. Afterwards, I created a setter and a getter function for each variable. Finally, I wanted to test the user class, so when I made a login query in my login process file, I made a user object called myUser which stores information about the user. After that, I printed the first name and the last name using the getters that I had created earlier in my class, and it worked perfectly.

```
class User {
    private $userID;
    private $firstName;
    private $lastName;
    private $email;
    private $userType;
    public function __construct($id, $ut, $fn, $ln, $em) {
        $this->userID = $id;
        $this->userType = $ut;
        $this->firstName = $fn;
        $this->lastName = $ln;
        $this->email = $em;
    }
    function getUserId()
    {
        return $this->userID;
    }
    function setUserId($id)
    {
        $this->userID = $id;
    }
```

Login process:

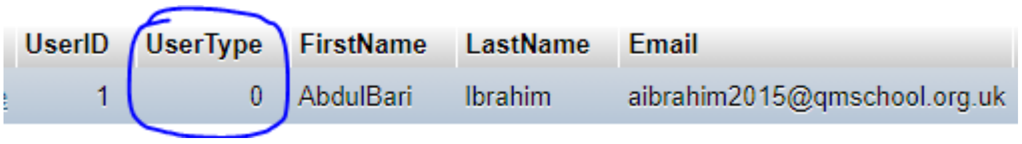
```
$mysqli = new mysqli("localhost", "root", "", "e-commerce");
$stmt = $mysqli->prepare("SELECT * FROM eUsers WHERE Email=?");
$stmt->bind_param("s", $_POST["email"]);
$stmt->execute();
$res = $stmt->get_result();
$stmt->close();
 if ($res->num_rows == 1) {
     $row = $res->fetch_assoc();
      if (password_verify($_POST["password"], $row["Password"])) {
          $myUser = new User($row["UserID"], $row["UserType"],
$row["FirstName"], $row["LastName"], $row["Email"]);
            $_SESSION["myUser"] = $myUser;
            header("Location: /MyEcommerce/Welcome.php");
            exit;
        }
```

Date:28/04/2021

Task: Starting the user management system

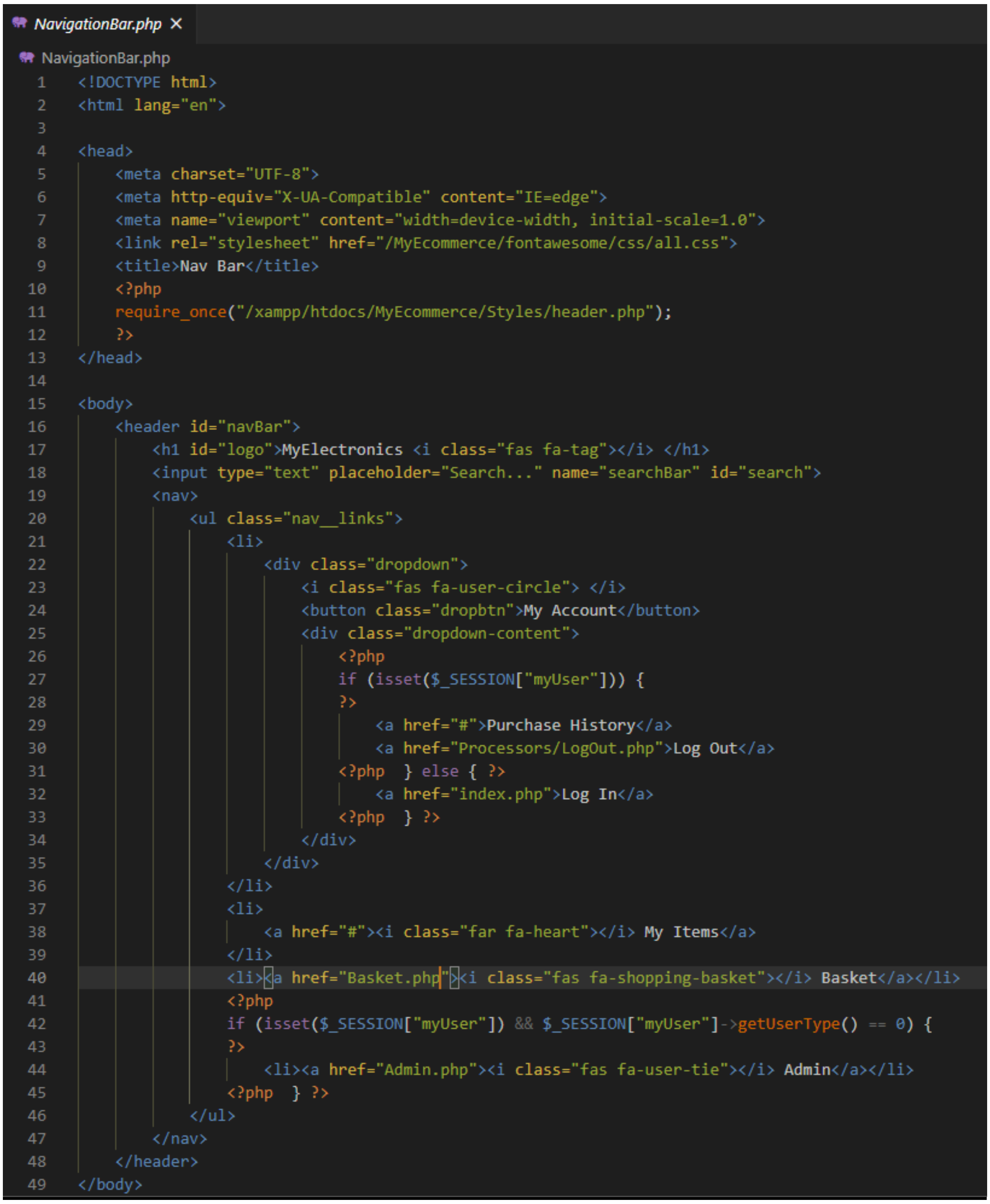
I started thinking about how I can make a user management system (e.g. customer account, admin account). One way I thought about is to make a radio button in the registration form to enable the user to select what type of account they will use. However, this way isn't too secure as a customer can sign in as an admin and can delete, edit or add information that can harm the website. After asking and searching, I discovered that I will need to create a field in the database called User Type. This will store an integer which is either 0 = admin or

1 = customer, after that, in my SQL query I had to set the value of this field as 1 because all users will have a customer account unless the admin changes the value of the user type manually to 1. This way is more secure and professional.
To test this, I created a href link in the navigation bar called "Admin", then I made an IF statement to check if the user type of the person that has logged is 0 or 1, and if the user type was 0 then the Admin link would appear in the header when the user logs in and if it was 1 then it would be a customer account so it won't appear in the header. However, the first time I tested it worked when I logged in as an admin, but when I logged in as a customer the admin link was still there so I knew there was an error. After searching for the error I found out that the parameters of the user object class weren't in the same order as the new object's arguments that I declared in my login process file. So I fixed the error by writing them in the same order and it worked.



UserID	UserType	FirstName	LastName	Email
1	0	AbdulBari	Ibrahim	aibrahim2015@qmschool.org.uk

In addition, I made a template for the navigation bar/header by creating a separate PHP file that contains code only for the header, and then I just used a PHP function called "include" on any page that I wanted the header to appear in.



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/MyEcommerce/fontawesome/css/all.css">
    <title>Nav Bar</title>
    <?php
    require_once("/xampp/htdocs/MyEcommerce/Styles/header.php");
    ?>
</head>

<body>
    <header id="navBar">
        <h1 id="logo">MyElectronics <i class="fas fa-tag"></i> </h1>
        <input type="text" placeholder="Search..." name="searchBar" id="search">
        <nav>
            <ul class="nav__links">
                <li>
                    <div class="dropdown">
                        <i class="fas fa-user-circle"> </i>
                        <button class="dropbtn">My Account</button>
                        <div class="dropdown-content">
                            <?php
                            if (isset($_SESSION["myUser"])) {
                            ?>
                                <a href="#">Purchase History</a>
                                <a href="Processors/LogOut.php">Log Out</a>
                            <?php  } else { ?>
                                <a href="index.php">Log In</a>
                            <?php  } ?>
                        </div>
                    </div>
                </li>
                <li>
                    <a href="#"><i class="far fa-heart"></i> My Items</a>
                </li>
                <li><a href="Basket.php"><i class="fas fa-shopping-basket"></i> Basket</a></li>
                <?php
                if (isset($_SESSION["myUser"]) && $_SESSION["myUser"]->getUserType() == 0) {
                ?>
                    <li><a href="Admin.php"><i class="fas fa-user-tie"></i> Admin</a></li>
                <?php  } ?>
            </ul>
        </nav>
    </header>
</body>
```

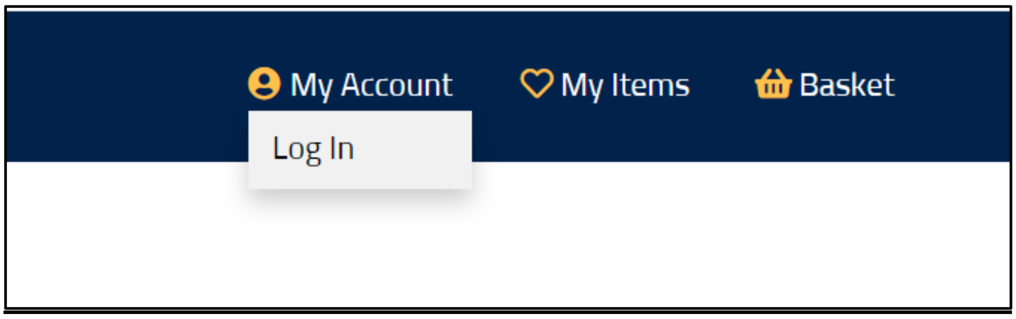
Then I used this line of code to add it to any page I want:

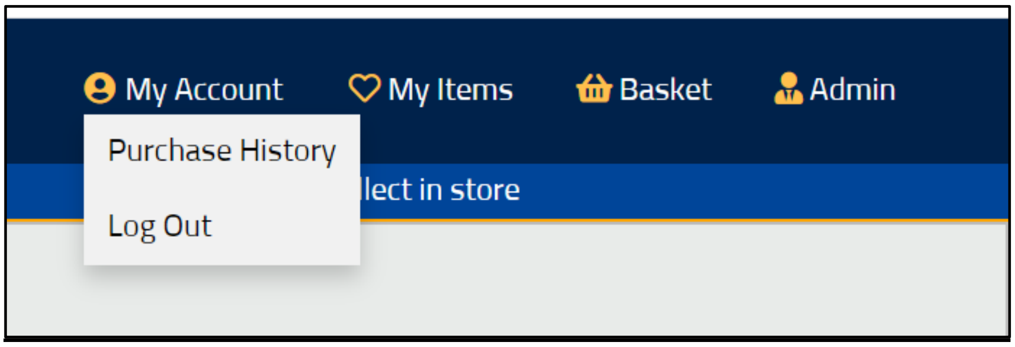
```
include "NavigationBar.php";
```

Date:30/04/2021

Task: Adding a dropdown menu to the My Account link

Initially, I started by searching for ways of adding a dropdown menu to the My Account link in the header, after a while, I discovered that I needed to create a div inside my "li" tag for the dropdown menu, also I had to change the type of My Account from a <a> tag to a <button> tag. This is because I wanted this feature to work in a way that when the user hovers over the button, the dropdown menu appears, and this feature couldn't be done using a <a> tag. I tested this by using the website on a couple of different devices and checking if the dropdown menu appears properly and with the right sizes, and it was all fine.



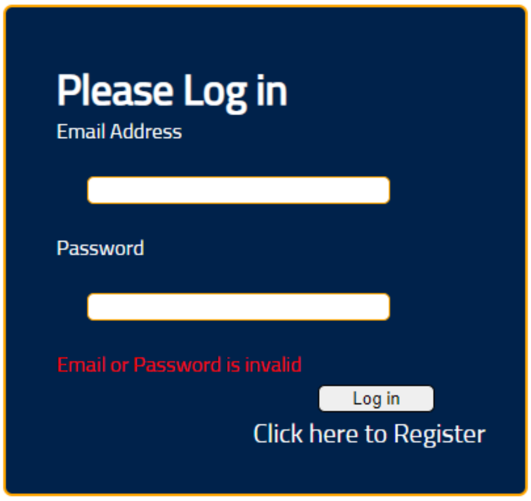


Date:05/05/2021

Task: Improving styles

On this day I have only focused on improving the styles and making general changes to the website. I started by changing the id for the login form and the registration form, in which at the beginning they had the same id, but I realised that the login form has fewer inputs than the registration form, so the div was too big for it. Therefore, I changed the id of the login form and I copied the CSS code for the registration form and pasted the CSS code for the login form, then I just changed the margins and paddings and made them smaller to suit the login form. Furthermore, I needed to change the colour of the error messages that appear to the user when the inputs are invalid to a red colour, but the problem was that I was printing those messages with a <p> tag using the PHP echo function, and so I didn't know how to change colour as I kept getting an error every time I tried to set a class or id for the <p> tag. After asking, I figured out that instead of a double speech mark I only had to use a single speech mark when declaring the class/id inside an echo function. To test it, I typed in incorrect information and then clicked submit, and all my error messages were in the colour red, so it worked properly.

```
echo "<p class='errorMessage'>Email or Password is invalid</p>";
```


Please Log in
Email Address
Password
Email or Password is invalid

Date:07/05/2021

Task: Fixing the login form

As I was testing different parts of the website, I noticed that the login form wasn't printing an error message when I logged in with an email and a password that did not exist in the database, instead the system took me to a blank page. However, after going through my code for a while, I discovered that I needed to add an "else" after the IF statement that I used to check the number of rows in the "DoLogin.php" file. Inside the else, I added the username and the password to my errors list and then stored it in the session so I can use it in the login.php file to print an error message, after that, I redirected the user back to the login page. I tested this by typing an email and a password that does not exist in the database, and it printed an error, which means it was working correctly.

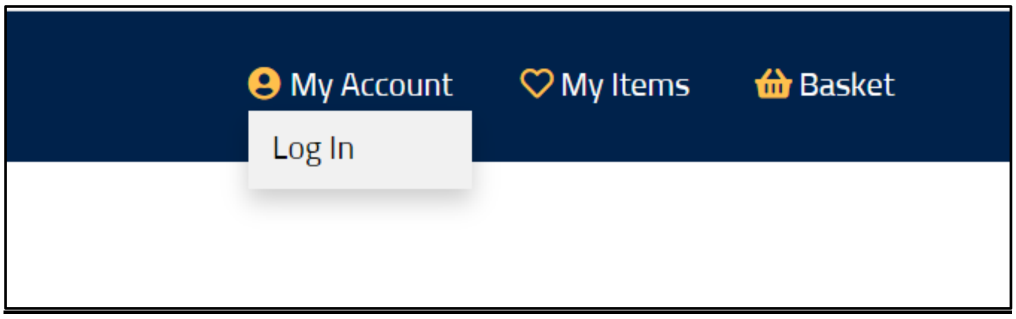
```
else {
        $errors[] = "email";
        $errors[] = "password";
        $_SESSION["allerrors"] = $errors;
        header("Location: /MyEcommerce/index.php");
        exit;
    }
```

Date:17/05/2021

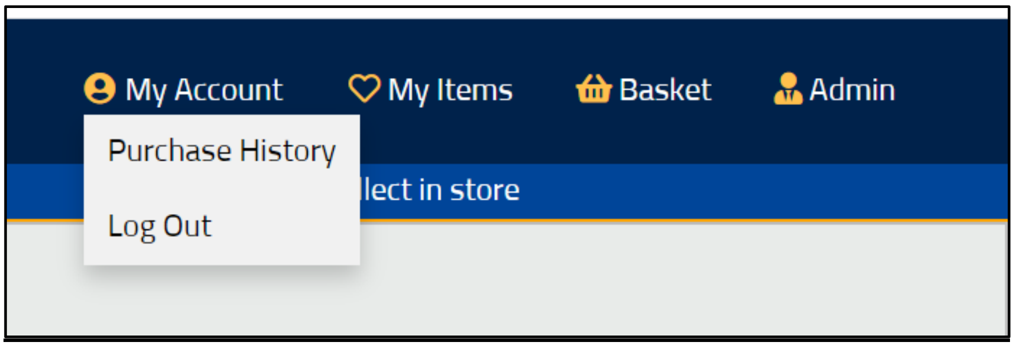
Task: Start building the Admin page

On this day, I began planning and designing the Admin page, but as I was including the Navigation bar in the page, I was alerted that before the page loads to the user, I need to check if the page is loading to the right user which is only Admin. This is because anyone can copy the URL of the admin page and access it, but if the page detects that the user is not an admin, then it will redirect them to the index page.
So in my utilities.php file, I made a function consisting of an IF statement to check the user type using the information stored in the session when the user logs in. I completed the if statement then I tested it by logging in and then going to the admin page, copying the link of the admin page, then logging out and finally accessing the page using the link we copied. I found out that I could still access the page and it did not redirect me. After going through my code, I discovered that there was some issue with the logic in the if statement, in which at the beginning I checked if the user was in the session, then redirect them which is incorrect, therefore, I changed the logic and made if the user was not in the session or user type is not equal to zero then redirect to index page. Finally, I tested it the same way I did before and it worked well.

If the user doesn't log in or if the account type is a user then the admin link won't appear:



Otherwise:



Date:19/05/2021

Task: Organising my project file

As the number of scripts inside my project file is increasing, I decided to organise them and put them in folders, so when I want to access a file it would be more efficient and quicker instead of getting confused between so many files in one folder. Therefore, I created a file of the classes and objects (e.g. UserObject), and also created a file for styles and processors. One problem I encountered was linking the files in the code. Before creating folders for the files, all I had to do when I wanted to include a file was just write the name of the file + .php, but after putting different files in different folders, I had to specify the exact path of the file. However, this wasn't a major issue, the issue was that in each case I had to use a different path. For example, inside the function require_once() I had to recall the whole path starting from the root which is xampp, but in other cases, I only had to start the path from the project file (/MyEcommerce…). This process took me a while as I had to change all the links in my project. Eventually, I tested the website by using all the features I made to make sure that everything was working properly, and the links were successfully working.

Date:21/05/2021

Task: Creating a Slideshow in the Welcome Page

On this day, I started working on a slideshow on the welcome page. The purpose of the slideshow is to give the user a quick overview of the offers and discounts available on the website. I began the process by researching ways of creating a slideshow just to get a hint on how to start. Firstly, I created a div which was the container of the slideshow, then inside that, I created other 3 divs, and each div represented a different image.
After that, I added some CSS for the slideshow to adjust its size and margins etc. I wanted to make the slideshow automatic, this means that I wanted the image to change every (e.g. 1seconds), therefore, I used Javascript to adjust the display of the image and change it to either "none" or "block". I used a function called Setimeout() to change the image every 1 second, and this function requires two arguments, one is the function for displaying the images and the second argument is the amount of time (e.g. 2 seconds = 2000).
After adding the arguments, I logged in to my website and went to the welcome page to test the slideshow, unfortunately, the slideshow wasn't changing images. I went back to my code to figure out what was the problem, so after a while, I discovered that inside the Setimeout() function, I called my function ShowSlides inside speech marks, therefore it wasn't calling the function and as a result, it wasn't changing image, so I removed the speech marks and I tested it again and it was working successfully.

<u>Screenshots.</u>

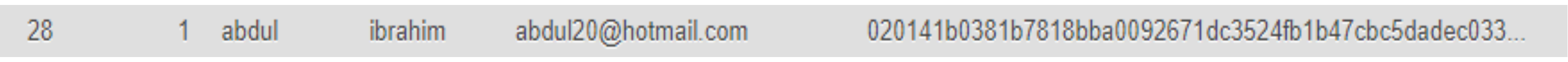
Date:24/05/2021

Task: Hashing passwords pt.1

To start with, I searched how to hash passwords in PHP, and I found a function called hash() which takes the following parameters: Name of the algorithm, password. I discovered that the algorithm sha256 is one of the best and most secure algorithms to use, and it uses 256 bits. Therefore, in my database, I changed the length of the password field to 256.
This is the line of code I used to produce a hashed password, then I simply replaced the variable hashed password with the password in the prepared statement.

```
$hashedPassword = hash("sha256", $password);
...
$stmt->bind_param("ssssissss", $fname, $lname, $email, $hashedPassword,
$pNum, $country, $city, $address, $postcode);
```

I tested this by creating a new account and checking the database to see if the password has been hashed, and it successfully worked.



28	1	abdul	ibrahim	abdul20@hotmail.com	020141b0381b7818bba0092671dc3524fb1b47cbc5dadec033...

However, I was getting an error when I was trying to log in, and I found out that when I log in, the system checks the password that was entered by me, and not the hashed password. Therefore, the passwords didn't match and so I couldn't log in. To solve this I created a function in my Utilities.php file and I called it createHashedPassword() that takes in a password and hash it and returns the hashed password. Therefore, I could use it in the register and login process files by requiring the utility file once, this will allow me to check the hashed password in DoLogin.php.

```
function createHashedPassword($password) {
    $hash = hash("sha256", $password);
    return $hash;
}
```

```
$hashedPassword = createHashedPassword($password);
```

I tested this by logging in with the account that I created earlier, and it successfully logged me in.

Date:20/06/2021

Task: Hashing and Salting passwords pt.2

After a while, I learned from my searches that to make the passwords even more secure and unbreakable, I needed to salt them. This means adding a random text to a hashed password so when a hacker gets access to one password, he/she won't be able to access other

accounts with the same password even if they have the same hash, this is due to the unique salt each password has. To salt a password, inside my createHashedPassword() function, I generated a random text using this line of code md5(uniqid(rand() true)). Then I concatenated it with the hashed password. After that, I tested the program by creating a new account and then logging in with it, and the result was I couldn't log in. I started searching for the problem and why it is not functioning properly and after a while, I discovered that whenever I call the createdHashedPassword() function it creates a new salt and adds it to the hashed password. For example, when I call it in the DoRegister file, it salts the hashed password and stores it in the database, however, when I call the function again in the process login file, it creates a new salt, therefore, it doesn't match the one in the database, and as a result, it won't allow me to log in.
After a long period of searching for new ways of hashing and salting, I discovered a built-in function in PHP called password_hash(). This function hashes and salts a given password by taking 2 parameters: the actual password and the hashing algorithm. Therefore, I realised I don't need my createdHashedPassword in the utilities file anymore, so I deleted it. In addition, I found out that to use this function to check the password in the DoLogin file, I needed to use another built-in function called password_verify(). However, to use this, I needed to remove the password argument from the prepared statement (bind-param), and after checking the number of rows found, I needed to make another if statement that checks the hashed and salted password against the actual password. To test this, I created another account, then I logged in using that account and I successfully gained access to the account. Finally, I checked the database if the password was salted and hashed correctly by creating another account with the same password, then I verified if the hashes were different and they were different, so the salting and hashing were successful.

Date:28/06/2021

Task: Creating a Basket Object

To begin with, as adding an item in the basket is the same as adding a user to the Users table in the database, I thought it would be efficient to create a basket object, so every time the user adds a new item to the basket, I can just make a new instance of that item the same way I did for the user. This will allow me to use the information of the item that is in the basket such as name, price and quantity of a product in other processes such as checkout just by calling the getters functions, or even make adjustments using the setters functions. Finally, I was not able to test the basket object as I didn't create a basket table in the database, also I didn't add any products.

```
class Basket {
    private $productID;
    private $productName;
    private $productPrice;
    private $productImage;
    private $productQuantity;
    public function __construct($id, $name, $price, $image, $quantity){
        $this->productID = $id;
        $this->productName = $name;
        $this->productPrice = $price;
        $this->productImage = $image;
        $this->productQuantity = $quantity;
    }
    function getProductID()
    {
        return $this->productID;
    }

    function setProductID($id)
    {
        $this->productID = $id;
    }
```

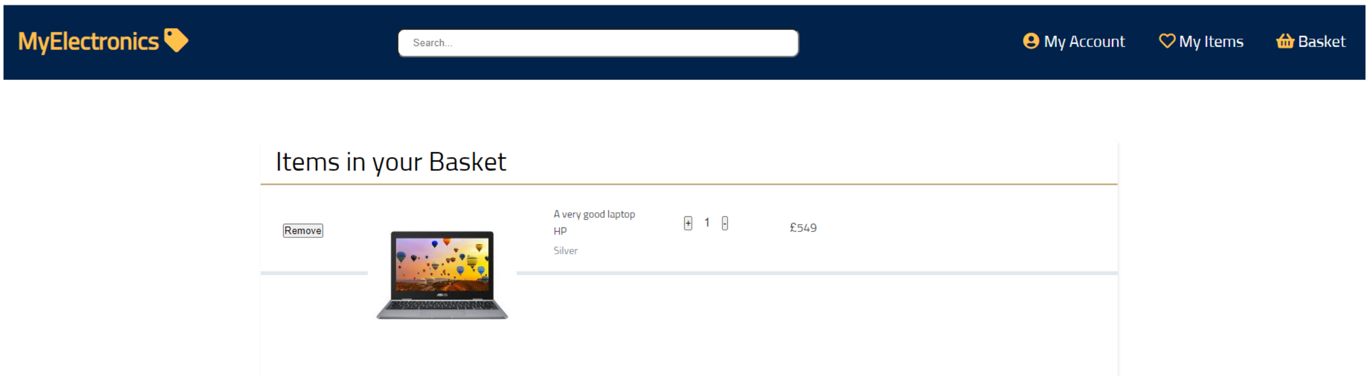
Date:30/06/2021

Task: Building the Basket page

I started by doing a little bit of research on how basket pages are structured and how the div tags should be structured in my html code. After a while, I got a good understanding of the structure that should be used, therefore, I began to code my div tags and this is the result.

```
<div class="basket">
        <h1 id="title">Items in your Basket</h1>
        <hr>
        <div class="item">
            <div id="remove">
            </div>
            <div id="image">
            </div>
            <div id="description">
            </div>
            <div id="quantity">
            </div>
            <div id="price">
            </div>
        </div>
    </div>
```

The next step was to create a styles file for the basket page and then connect it to the basket.php file. To test that the connection works properly, I added a little bit of style on the title, then I refreshed the basket page, and the style was successfully added.
After that, I went ahead and started styling the whole page. In this process, I did not encounter any issues or errors, however, I used a couple of online resources to help me add the correct CSS code to achieve the required style for the page. As I was adding the CSS code, I was consistently visiting the page and refreshing it to ensure that the changes are being added correctly. One issue I faced at the end was making the horizontal line that separates the items in the basket. I used a CSS feature called:nth-child() to add the horizontal line, however, the line wasn't appearing in the correct position as half of the first item in the basket was under the line.



Date:05/07/2021

Task: Completing the Basket Design

I found out that the size of the picture was too big, therefore the line wasn't appearing correctly. I adjusted the picture and here is the result:

After that, I started to work on the quantity button. In the beginning, I created an add button and a subtract button, and in between them, I created an input to show the quantity. I wanted to create an on click function that either decreases or increases the quantity. However, as I was researching ways to implement this, I discovered that there is an input type called the number which already has this feature. Therefore, I removed the button and changed the type of the input from text to number. I tested it to make sure that it changes the quantity by clicking the up and down buttons, and it was working fine. The next step was to change the total price based on the quantity. As I only had one div tag for the price of the product, I realised that I had to create another div for the total price that changes based on the quantity. I wasn't sure how to implement this, however, after asking and receiving some help, I found an efficient way to do this. I knew that in the database, each item will have an id, therefore, my updateQuantity() function takes an id as a parameter and concatenates it with the id of the input which is "amount" as well as the id of the price which is "price". As a result, if the id of the was for example 24, then it will amount24 and price24, the function will retrieve the price and quantity of that item and change the total price by multiplying those two. I tested this feature by changing the quantity of the item I have in my basket, and the total price was getting updated successfully. Finally, I did not encounter any errors or issues during this process.

Date:09/07/2021

Task: Start working on Basket page functionality

To begin with, I created a function in the utilities file that checks if the user is in the session, so only logged in users can access the basket page and if they try to access the page and are not logged in they will be redirected to the login page. I tested this by accessing the page while logged in and the result was that it redirected me to the correct page which is the basket page, then I logged out and tried to click on the basket and it redirected me to the login page, therefore, it was successful. In this process, I did not encounter any issues or errors.

Date:16/07/2021

Task: Showing products on the Welcome page & Add to basket button

Firstly, I started by loading the products on the welcome page for the user to browse through them and then decide whether they want to add them to the basket or not. I did this by creating a function in the ProductObject file that was previously called BasketObject, and I changed the name as the code the file contains is not only for the basket, it can also be applied to the welcome page. I called the function loadAllProducts(), this function goes through the products table in the database and appends each product to an array and then returns that array.

```
public static function loadAllProducts() {
        $mysqli = new mysqli("localhost", "root", "", "e-commerce");
        $stmt = $mysqli->prepare("SELECT * FROM eProducts");
        $stmt->execute();
        $res = $stmt->get_result();
        $stmt->close();
        $products = [];
        while ($row=$res->fetch_assoc()) {
            $myProduct = new Product($row["ProductID"],
$row["ProductName"], $row["ProductPrice"], $row["ProductImage"],
$row["ProductDescription"], $row["ProductStock"]);
            $products[] = $myProduct;
        }
        return $products;
    }
```

After this, in the welcome page I added all the HTML code for how I want the product to look(hard code - Prototype), then I moved it to the ProductObject file which then I put in a function so I can just call the function whenever I need to print a product.

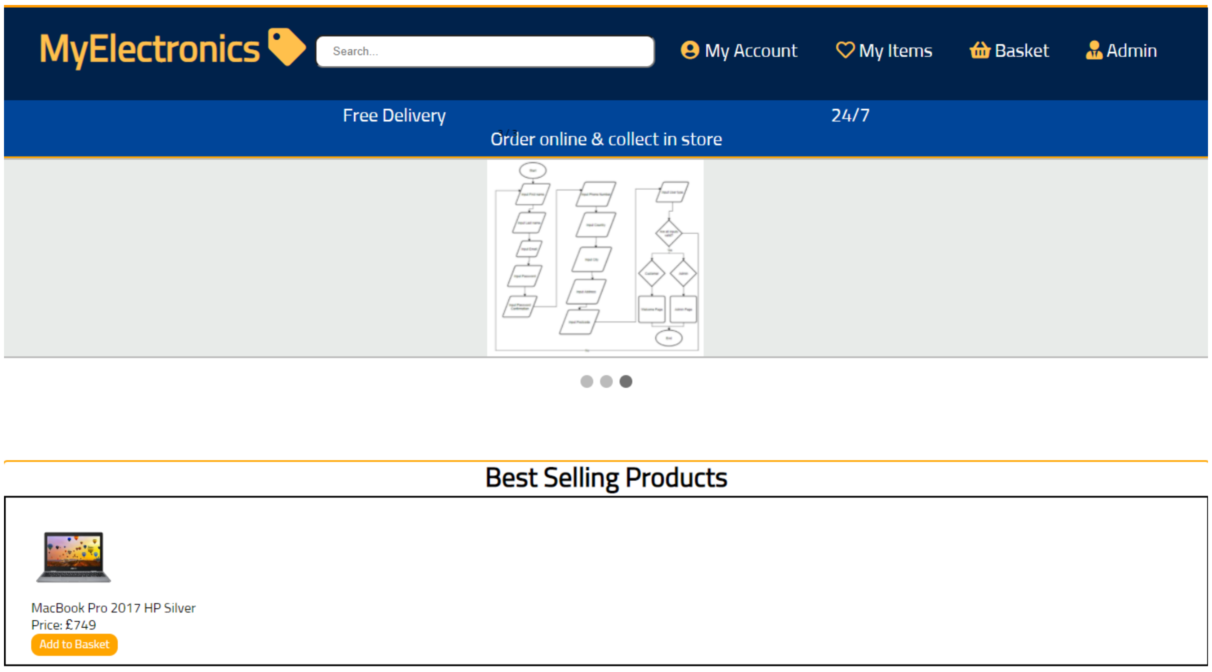
```
function printBestSellingProduct() {
        $html = '<div id="best-selling">
```

```
            <div id="item1">
                <div id="image'.$this->productID.'">
                    <img src="/MyEcommerce/Images/C223NA-
GJ0014_1_Classic.png" style="width:50%">
                </div>

                <div id="name'.$this->productID.'">
                    <span>'.$this->productName.'</span>
                    <span>HP</span>
                    <span>Silver</span>
                </div>

                <div id="itemPrice">
                    <input type="hidden" id="price1" value="549">
                    <span>Price: £'.$this->productPrice.'</span>
                </div>

                <div>
                    <button id="basket-btn"
onclick="addToBasket('.$this->productID.')">Add to Basket</button>
                </div>
            </div>
        </div>';
        return $html;
    }
```

Going back to the welcome page, I started a php code to call the function loadAllProducts() so I can loop through the array it produces then calling the print function on each product as you can see in the code below:

```
<div class="products">
        <h1 id="title" style="text-align:center">Best Selling
Products</h1>
        <?php
            $theProducts = Product::loadAllProducts();
            for ($i = 0; $i < sizeof($theProducts); $i++) {
                echo $theProducts[$i]->printBestSellingProduct();
            }
        ?>
    </div>
```

During this process, I did not encounter any issues or problems with the code. Finally, to test this I added a product to the products table in the database, then I refreshed the welcome page and the product appeared successfully.



Next, I started thinking about the button that will allow the user to add a product to the basket and after a little bit of research, I found out that I can use a built-in function in PHP that refreshes the page whenever the user clicks on add to basket button.

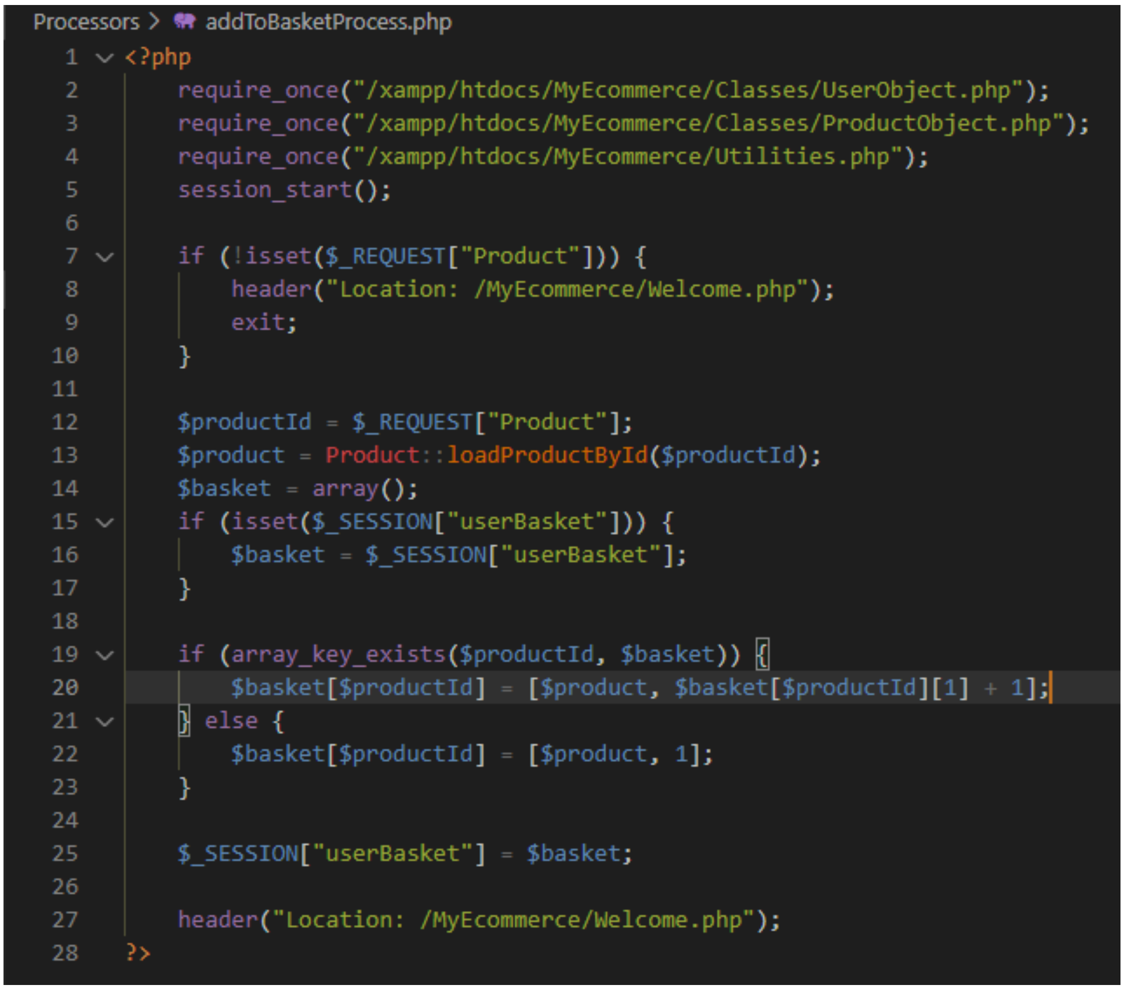
```
 function addToBasket(id) {
            window. location.href =
"Processors/addToBasketProcess.php?Product="+id;
        }
```

After I added this function to my code, I realized that I need a function that loads a product based on a given id. Therefore, I created the function in the ProductObject class and it takes an id as a parameter and I called it loadProductById().

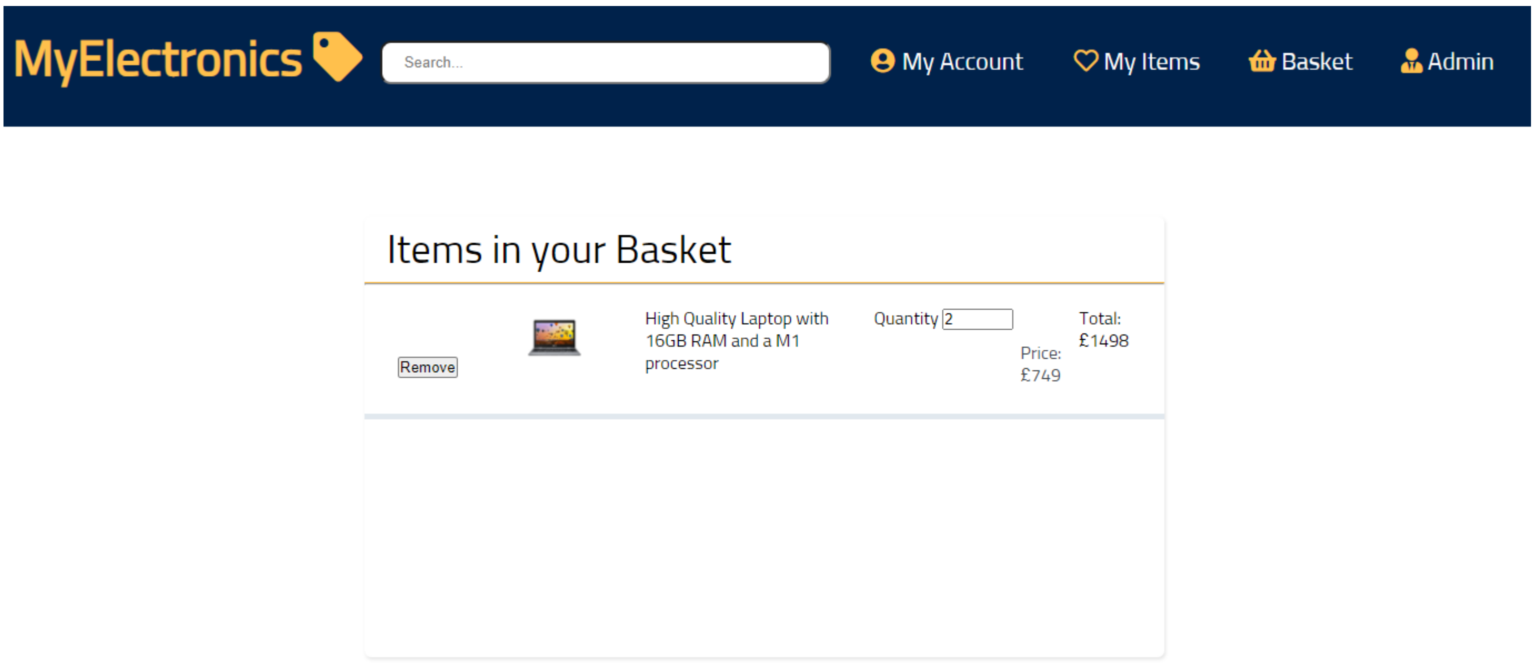
```
public static function loadProductById($id) {
        $mysqli = new mysqli("localhost", "root", "", "e-commerce");
        $stmt = $mysqli->prepare("SELECT * FROM eProducts WHERE
ProductID = ?");
        $stmt->bind_param("i", $id);
        $stmt->execute();
        $res = $stmt->get_result();
        $stmt->close();
        if ($res->num_rows == 1) {
            $row = $res->fetch_assoc();
            $myProduct = new Product($row["ProductID"],
$row["ProductName"], $row["ProductPrice"], $row["ProductImage"],
$row["ProductDescription"], $row["ProductStock"]);
            return $myProduct;
        }
        return null;
```

```
    }
```

The next step was to create a PHP file that carried out the process of adding a product to the basket, so I made the file and I called it addToBasketProcess.php. Firstly, this file allows the product id to get passed through the basket page, then it uses that id to load the product from the database using the loadProductById() function. Then it checks if there is a basket in the session, and if there is a basket it adds to a blank array. The basket is an associative array that holds the product id as a key and the quantity of that product as a value. There was a small issue which was that I struggled a little bit as I didn't know how to search through an associative array, however, after I researched I discovered that I can use a built-in function called array_key_exists that checks if a key exists in an associative array. So I made an if statement that checks if the basket array already contains a product with the id, if yes then it updates the array and adds 1 to the quantity, if not it adds the product with a quantity of 1.
During this, I did not encounter any problems apart from knowledge issues as PHP is a new language, so I had to research every time I was stuck.
Finally, to test this I went to the welcome page, and I clicked on the add to basket page, and then the welcome page was refreshed, after this, I went to the basket page and the product was added successfully, then I repeated the process to see if the quantity changes and it did increase, however, the quantity button in the basket page wasn't working properly as it didn't change the price as I was changing the quantity. I could only increase the quantity by going to the welcome page and clicking add to basket every time I want to do it.


Processors > addToBasketProcess.php

```
<?php
    require_once("/xampp/htdocs/MyEcommerce/Classes/UserObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Classes/ProductObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Utilities.php");
    session_start();

    if (!isset($_REQUEST["Product"])) {
        header("Location: /MyEcommerce/Welcome.php");
        exit;
    }

    $productId = $_REQUEST["Product"];
    $product = Product::loadProductById($productId);
    $basket = array();
    if (isset($_SESSION["userBasket"])) {
        $basket = $_SESSION["userBasket"];
    }

    if (array_key_exists($productId, $basket)) {
        $basket[$productId] = [$product, $basket[$productId][1] + 1];
    } else {
        $basket[$productId] = [$product, 1];
    }

    $_SESSION["userBasket"] = $basket;

    header("Location: /MyEcommerce/Welcome.php");
?>
```



Items in your Basket

High Quality Laptop with 16GB RAM and a M1 processor

Price: £749

Total: £1498

Date:04/08/2021

Task: Fixing quantity button & Adding Categories table in DB

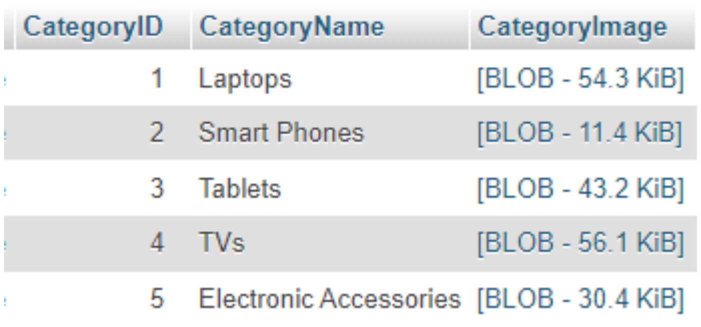
After going through the code a few times, I figured out why the quantity button wasn't working. The reason was that I set the IDs for the quantity and price divs to "amount1" and "price1". This is a problem because the function that I created which is updateTotal(id) takes an id of a product and concatenates it with the words "amount" and "price" to identify the value of the quantity and price so it can calculate the total price.

```
function updateTotal(id) {
            var a = document.getElementById("amount" + id).value;
            var b = document.getElementById("price" + id).value;
            var c = a * b;
            document.getElementById("total" + id).innerHTML = "Total: "
+ "£" + c;
        }
```

Therefore, when I set the ID of the divs as amount1 or price1, it only considers the product with ID 1. As a result, instead of giving the divs a specific id, I used a variable $id that gets the id of the product from the basket that is currently in the session.

```
$basket = array();
            if (isset($_SESSION["userBasket"])) {
                $basket = $_SESSION["userBasket"];
            }
            foreach($basket as $id=>$details) {
                $product = $details[0];
```
As you can see the underlined variable called $id, I used for the ID of the divs:
```
id="amount<?=$id?>"
...
id="price<?=$id?>"
```
I tested this by going to the basket page and changing the quantity of the product using the quantity input, and the price was changed successfully.

Next, I spent a little time on the database and I created a new table called Categories using my Data dictionary. I used websites that are similar to mine to identify what are the most important categories in an e-commerce website just to start with. This was my final result:

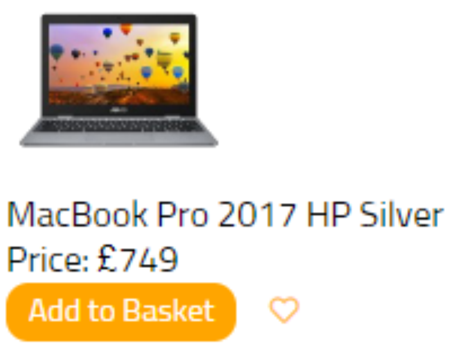


CategoryID	CategoryName	CategoryImage
1	Laptops	[BLOB - 54.3 KiB]
2	Smart Phones	[BLOB - 11.4 KiB]
3	Tablets	[BLOB - 43.2 KiB]
4	TVs	[BLOB - 56.1 KiB]
5	Electronic Accessories	[BLOB - 30.4 KiB]

Date:09/08/2021

Task: Adding functionality to the My Items section

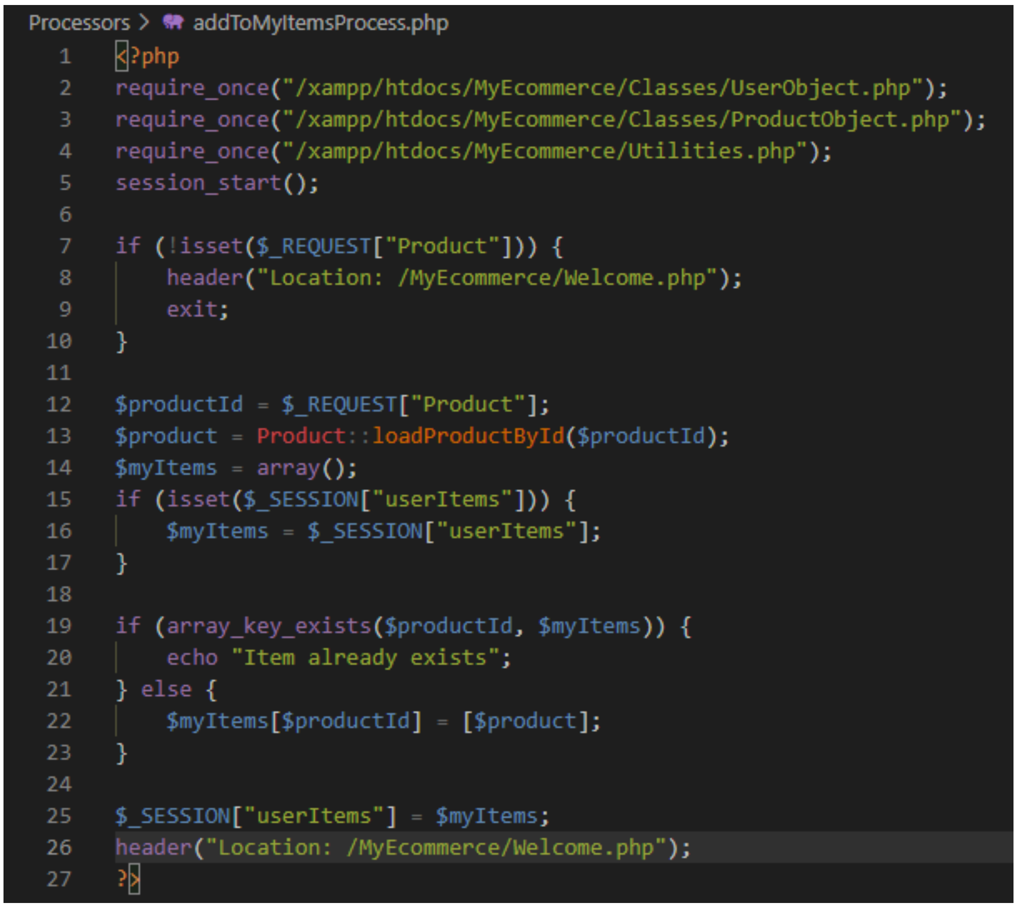
To start with, I made the file for my items page and added the directory of the file to the My Items button in the header so when the user clicks on it, it takes them to that page, then I created a button to allow the user to add a product to the basket in the welcome page, and I added the code for the button in my ProductObject class inside the print product function.
```
<div>
                    <button id="myItems-btn"
onclick="addToMyItems('.$this->productID.')"><i class="far fa-
heart"></i></button>
                </div>
```



Then I started thinking about how to add the product to the My Items section when the user clicks on the button, and after a while, I realised I can use the same method that I used for the basket button, except I only need to change some of the variables. So I began by adding this function to the Welcome page:

```
function addToMyItems(id) {
            window.location.href =
"Processors/addToMyItemsProcess.php?Product="+id;
        }
```

After that, I created the file addToMyItemsProcess.php that holds the process and adds the product to the My Items page.



```
Processors > addToMyItemsProcess.php
<?php
require_once("/xampp/htdocs/MyEcommerce/Classes/UserObject.php");
require_once("/xampp/htdocs/MyEcommerce/Classes/ProductObject.php");
require_once("/xampp/htdocs/MyEcommerce/Utilities.php");
session_start();

if (!isset($_REQUEST["Product"])) {
    header("Location: /MyEcommerce/Welcome.php");
    exit;
}

$productId = $_REQUEST["Product"];
$product = Product::loadProductById($productId);
$myItems = array();
if (isset($_SESSION["userItems"])) {
    $myItems = $_SESSION["userItems"];
}

if (array_key_exists($productId, $myItems)) {
    echo "Item already exists";
} else {
    $myItems[$productId] = [$product];
}

$_SESSION["userItems"] = $myItems;
header("Location: /MyEcommerce/Welcome.php");
?>
```

As it's shown in the picture, the code is very similar to the basket process code, however, I changed the name of the array to myItems and the name of the item in the session to userItems. One problem I encountered is that I didn't know how to output an error or stop the

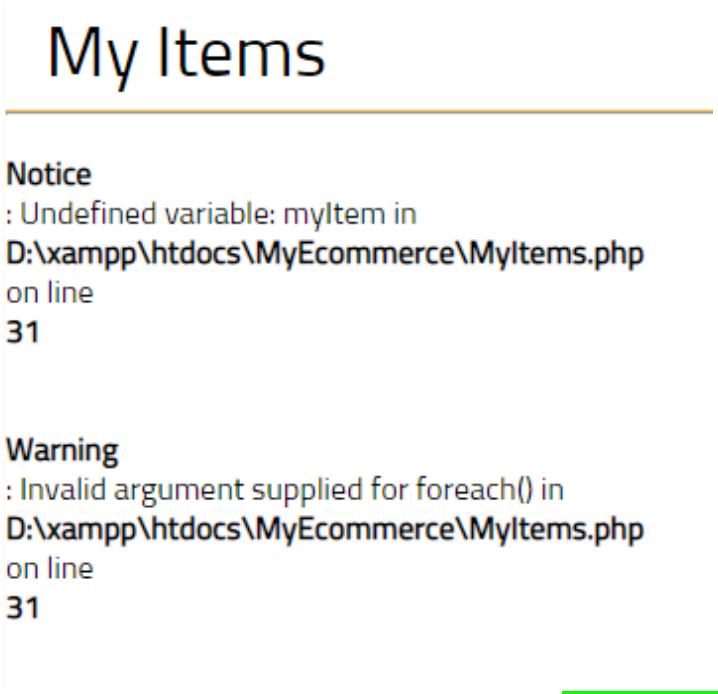
process when the product is already on the My Items page, therefore, I used the echo function temporarily.
In my items file that I created earlier, I started by checking if there is an item in the session, and if there is one, I assigned it to the array myItems. Then I looped through the array to get the id of each product, so I can print the information of that product on the page.

```
 $myItems = array();
            if (isset($_SESSION["userItems"])) {
                $myItems = $_SESSION["userItems"];
            }
            foreach($myItems as $id=>$details) {
                $product = $details[0];
```

After that I used the same html code for the basket to print the product, however, I removed the quantity and total price as it's not needed in the My Items section.

```
<div class="item" id="item<?=$id?>">
                    <div id="remove">
                        <button id="removeButton">Remove</button>
                    </div>


                    <div id="image<?=$id?>">
                        <img src="/MyEcommerce/Images/C223NA-
GJ0014_1_Classic.png" style="width:50%">
                    </div>


                    <div id="description<?=$id?>">
                        <span><?=$product->getProductDes()?></span>
                    </div>


                    <div id="itemPrice">
                        <input type="hidden" id="price<?=$id?>"
value="">
                        <span>Price: £<?=$product-
>getProductPrice()?></span>
                    </div>


                    <div id="move">
                        <button id="moveButton"
onclick="addToBasket(<?=$product->getProductID()?>)">Move to
Basket</button>
                    </div>
                </div>
```

Finally, to test this, I logged in to my account, and then clicked on the heart button to add the product to My Items. The result was that the welcome page refreshed, however, when I went to the My Items page, there was an error:



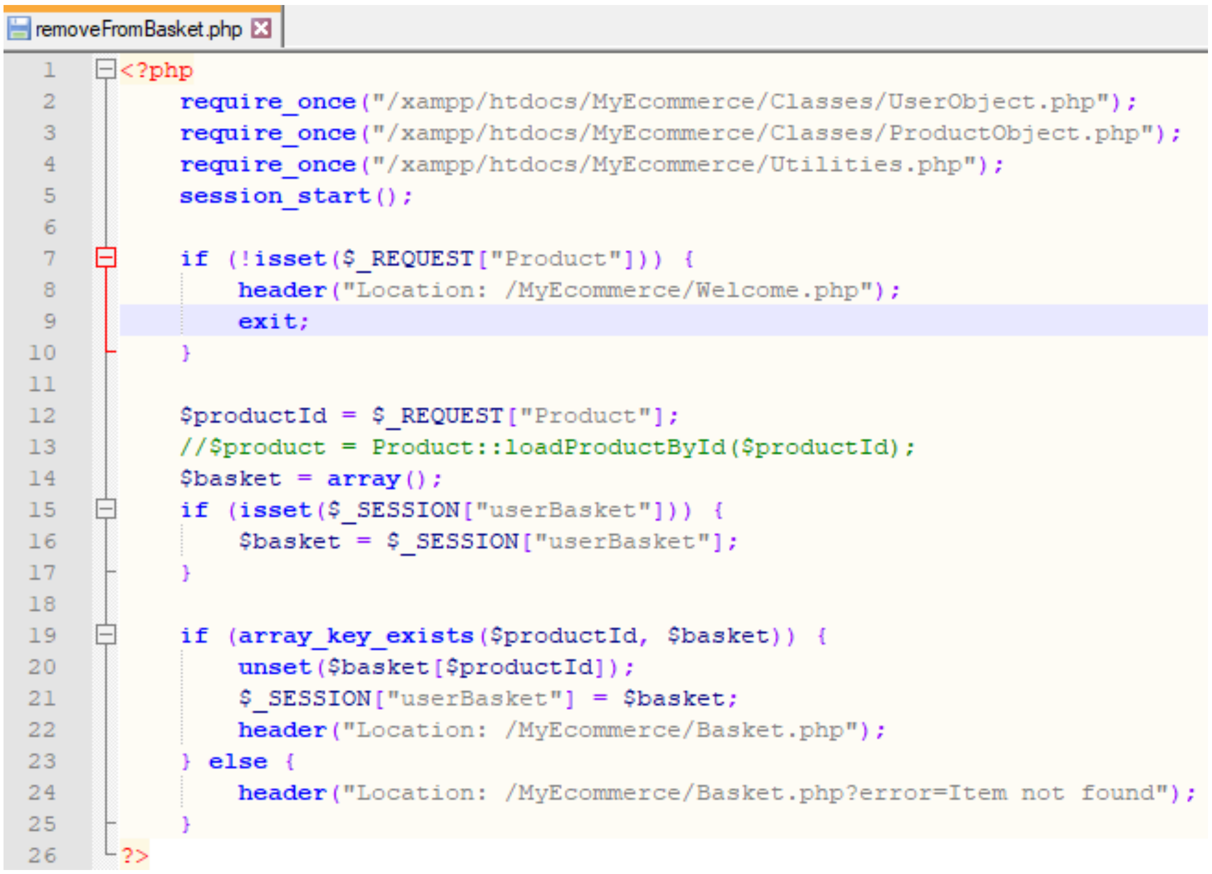
```
My Items

Notice
: Undefined variable: myItem in
D:\xampp\htdocs\MyEcommerce\MyItems.php
on line
31

Warning
: Invalid argument supplied for foreach() in
D:\xampp\htdocs\MyEcommerce\MyItems.php
on line
31
```

So I went to the MyItems.php file, and then I found out that in my foreach loop I didn't spell the name of the array correctly so I fixed it and tested it again the product appeared perfectly on the My Items page.

Date:01/09/2021

Task: Adding remove button in the Basket & MyItems

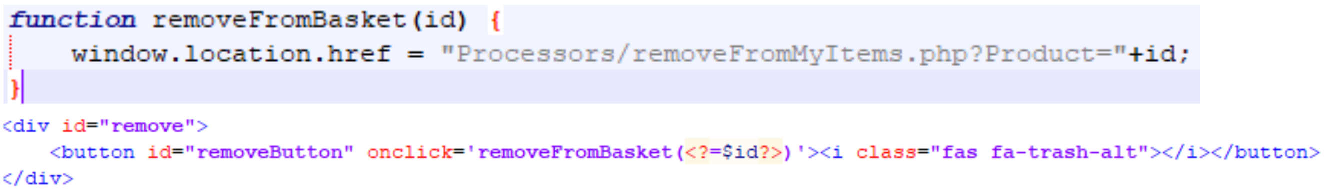
In the beginning, I was a little bit unsure about how to start, however, after research I found out that I can simply use the same method for adding products to the basket and my items, however, instead of adding a product to the associative arrays for the basket and my items, I can just delete that item using the unset() function. So I created a PHP file called removeFromBasket.php that processes deleting an item.



removeFromBasket.php

```php
<?php
    require_once("/xampp/htdocs/MyEcommerce/Classes/UserObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Classes/ProductObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Utilities.php");
    session_start();

    if (!isset($_REQUEST["Product"])) {
        header("Location: /MyEcommerce/Welcome.php");
        exit;
    }

    $productId = $_REQUEST["Product"];
    //$product = Product::loadProductById($productId);
    $basket = array();
    if (isset($_SESSION["userBasket"])) {
        $basket = $_SESSION["userBasket"];
    }

    if (array_key_exists($productId, $basket)) {
        unset($basket[$productId]);
        $_SESSION["userBasket"] = $basket;
        header("Location: /MyEcommerce/Basket.php");
    } else {
        header("Location: /MyEcommerce/Basket.php?error=Item not found");
    }
?>
```

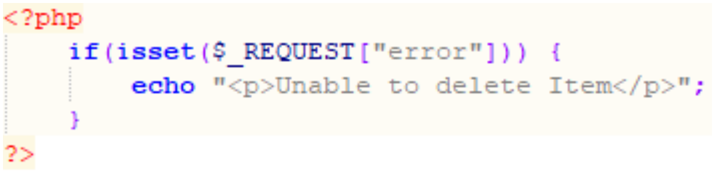
After that, in the basket.php I created a function called removeFromBasket(id). This function is called when the user clicks on the remove button.



```
function removeFromBasket(id) {
    window.location.href = "Processors/removeFromMyItems.php?Product="+id;
}
```

```html
<div id="remove">
    <button id="removeButton" onclick='removeFromBasket(<?=$id?>)'><i class="fas fa-trash-alt"></i></button>
</div>
```

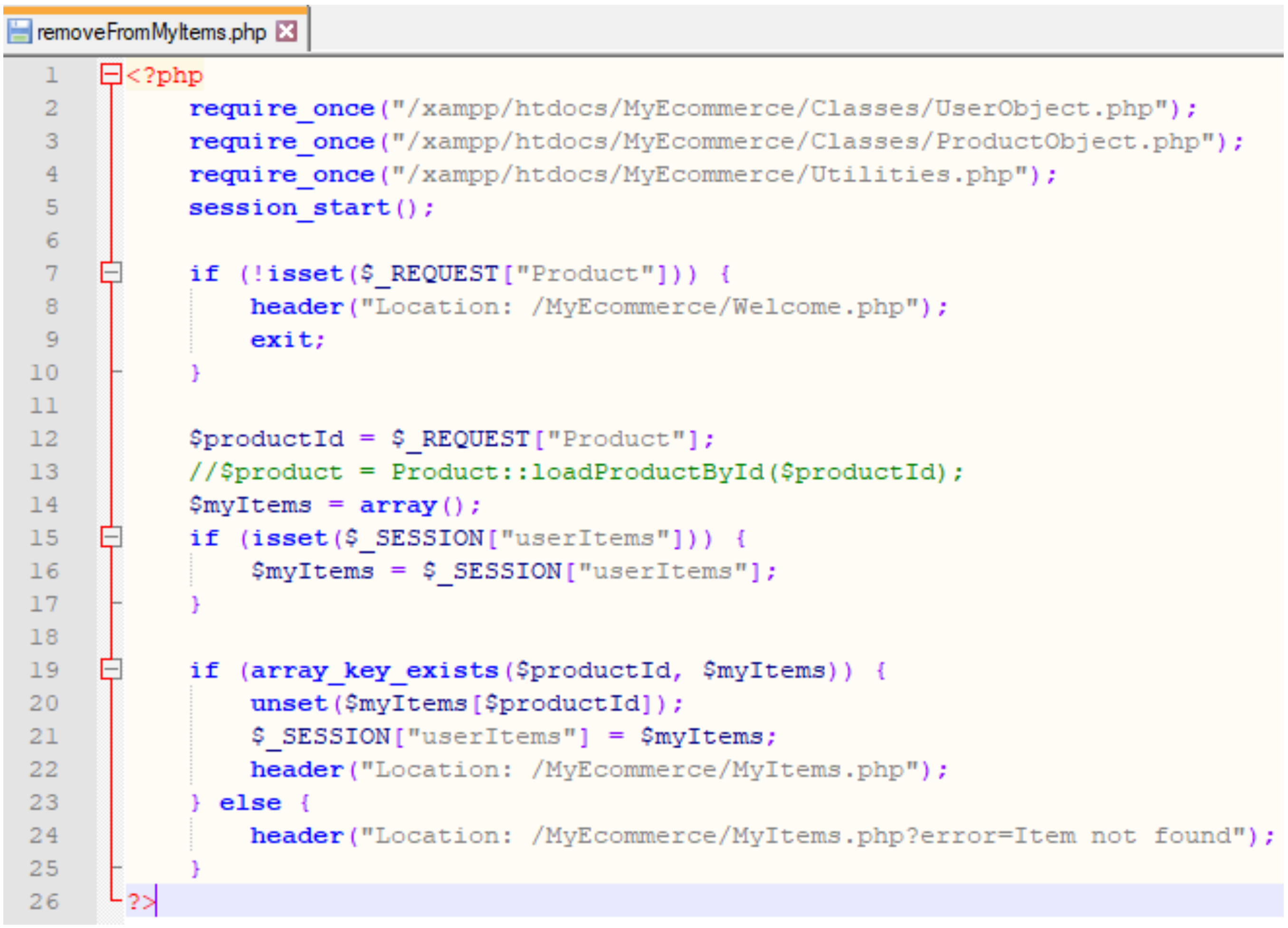
To handle any errors that might occur, as you can see in the picture above, I have another statement that produces an error message. So to display this error I used the isset() function and the <p> tags on the basket page.



```php
<?php
    if(isset($_REQUEST["error"])) {
        echo "<p>Unable to delete Item</p>";
    }
?>
```

Finally, to test this, I added an item to the basket and then clicked the remove button, and the item was successfully removed from the basket. I did not encounter any issues or problems with the basket.

However, I wanted to implement this feature for my Items as well, so I used the same function and the same process file on my items page. So when I tested it I got an error saying unable to delete the item. After a little while, I discovered that I need to use a separate process file to access the userItems in the session and not the userBasket, as the removeFromBasket process file used the userBasket item in the session, therefore, it was unable to delete the item from my items. After that, I added an item to my items and then removed it and it was working perfectly.



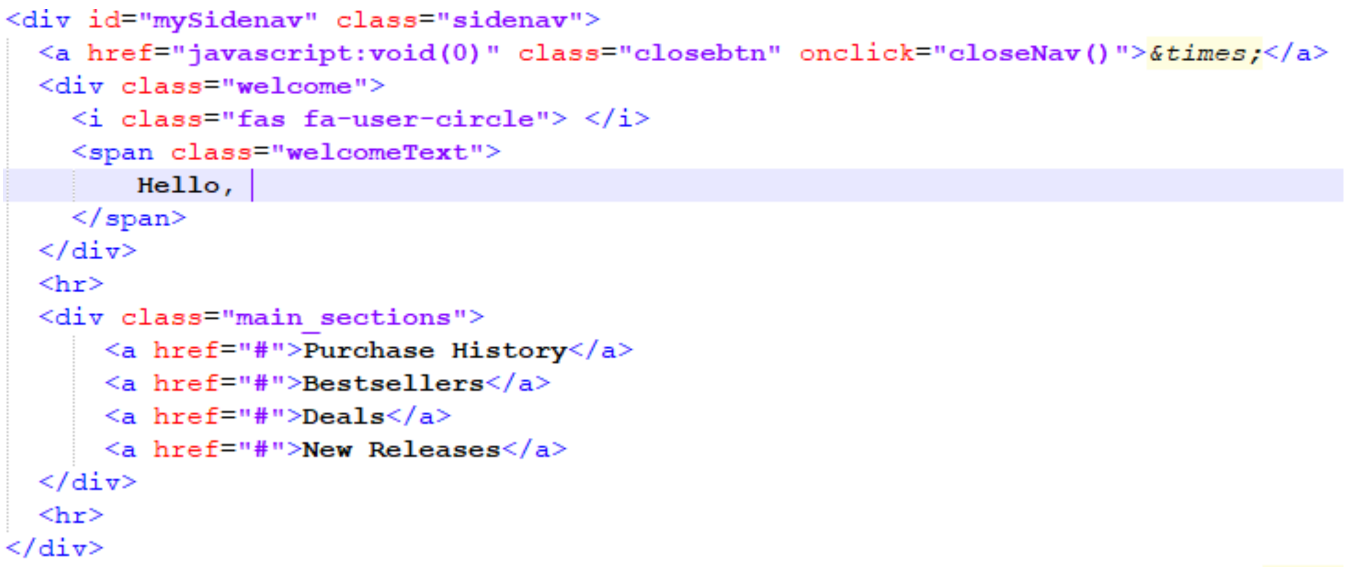
removeFromMyItems.php

```php
<?php
    require_once("/xampp/htdocs/MyEcommerce/Classes/UserObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Classes/ProductObject.php");
    require_once("/xampp/htdocs/MyEcommerce/Utilities.php");
    session_start();

    if (!isset($_REQUEST["Product"])) {
        header("Location: /MyEcommerce/Welcome.php");
        exit;
    }

    $productId = $_REQUEST["Product"];
    //$product = Product::loadProductById($productId);
    $myItems = array();
    if (isset($_SESSION["userItems"])) {
        $myItems = $_SESSION["userItems"];
    }

    if (array_key_exists($productId, $myItems)) {
        unset($myItems[$productId]);
        $_SESSION["userItems"] = $myItems;
        header("Location: /MyEcommerce/MyItems.php");
    } else {
        header("Location: /MyEcommerce/MyItems.php?error=Item not found");
    }
?>
```

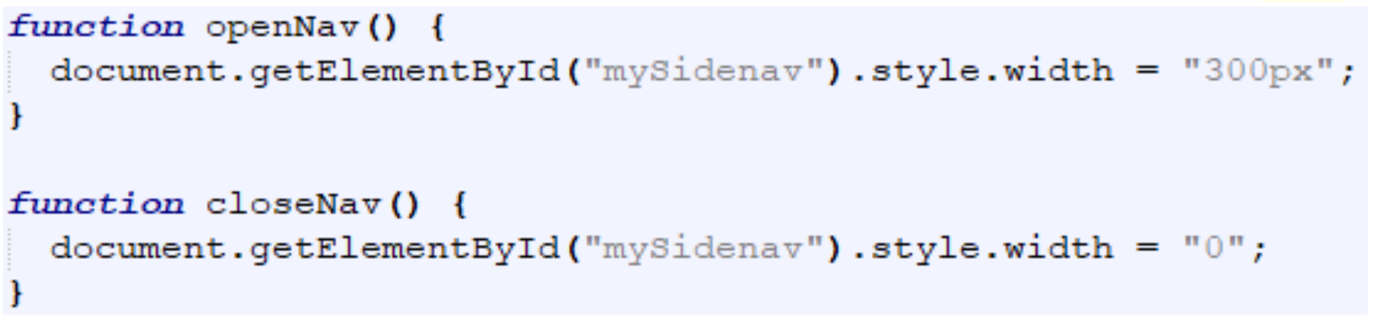
Date:10/09/2021

Task: Creating a Sidebar navigation

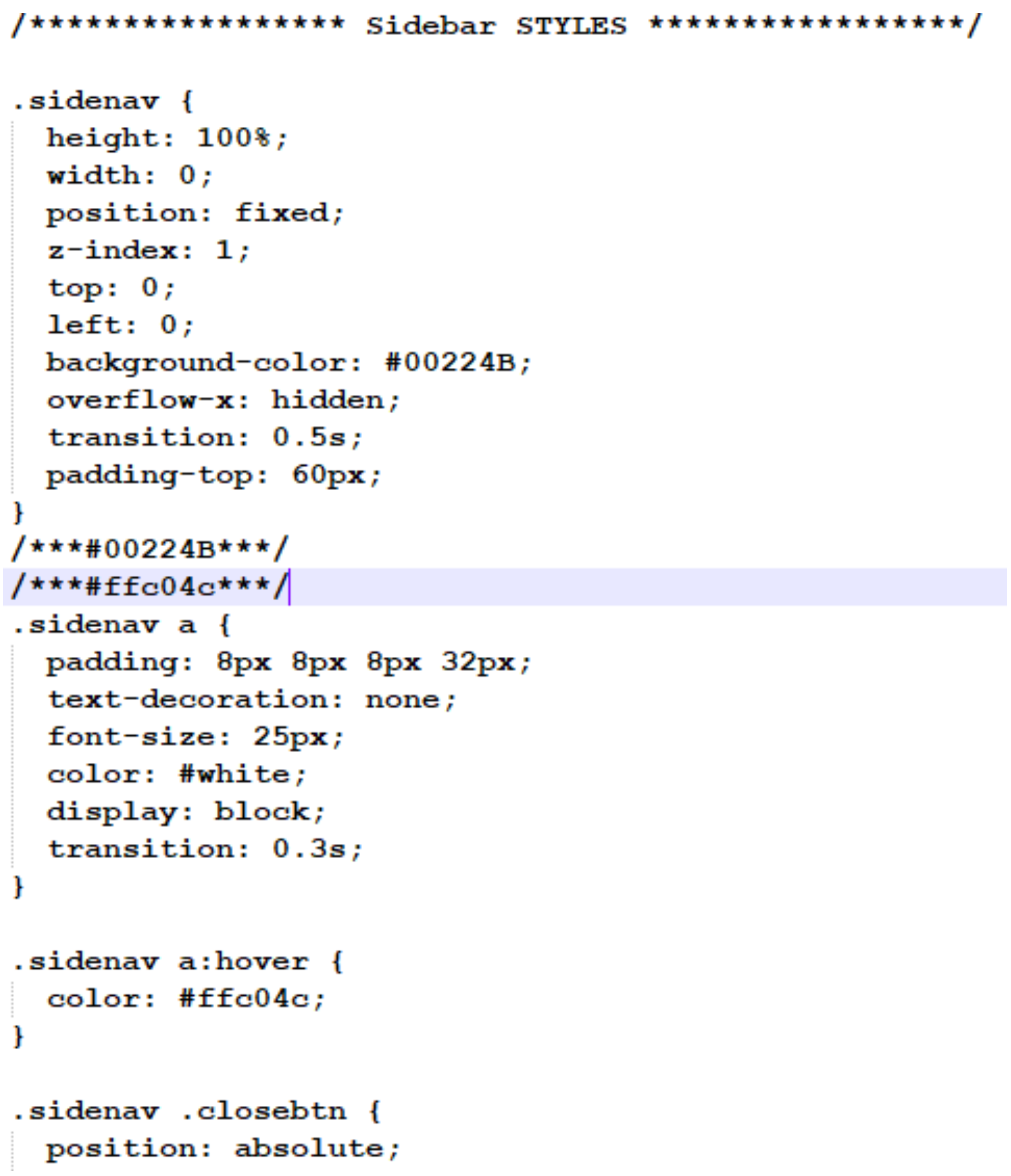
I started by doing a little research on how to create sidebar navigation, fortunately, I found several resources that explain the process clearly. So after I got a general idea of how to create a sidebar, I headed to the navigationBar.php file to start coding the sidebar there. The reason I coded it in the navigationBar.php file was that I wanted the sidebar to appear on every single page of the website. This is the code for the sidebar:



```
<div id="mySidenav" class="sidenav">
  <a href="javascript:void(0)" class="closebtn" onclick="closeNav()">&times;</a>
  <div class="welcome">
    <i class="fas fa-user-circle"> </i>
    <span class="welcomeText">
        Hello,
    </span>
  </div>
  <hr>
  <div class="main_sections">
      <a href="#">Purchase History</a>
      <a href="#">Bestsellers</a>
      <a href="#">Deals</a>
      <a href="#">New Releases</a>
  </div>
  <hr>
</div>
```

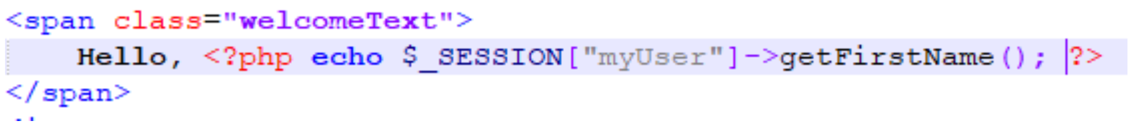


```
function openNav() {
  document.getElementById("mySidenav").style.width = "300px";
}

function closeNav() {
  document.getElementById("mySidenav").style.width = "0";
}
```

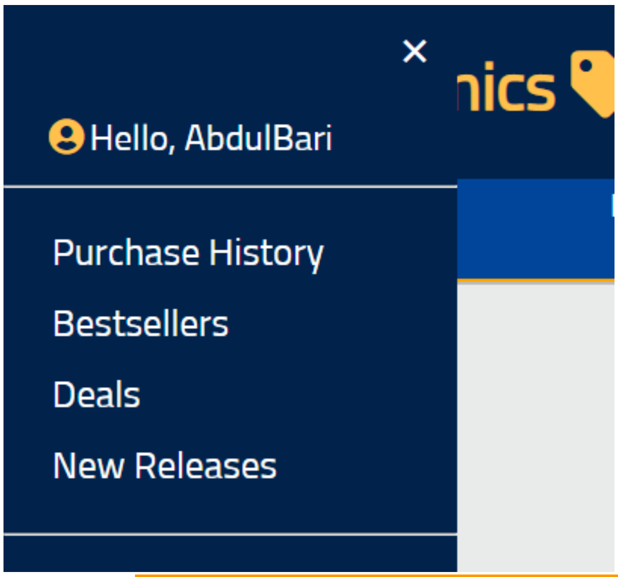


```
/***************** Sidebar STYLES *****************/

.sidenav {
  height: 100%;
  width: 0;
  position: fixed;
  z-index: 1;
  top: 0;
  left: 0;
  background-color: #00224B;
  overflow-x: hidden;
  transition: 0.5s;
  padding-top: 60px;
}
/***#00224B***/
/***#ffc04c***/
.sidenav a {
  padding: 8px 8px 8px 32px;
  text-decoration: none;
  font-size: 25px;
  color: #white;
  display: block;
  transition: 0.3s;
}

.sidenav a:hover {
  color: #ffc04c;
}

.sidenav .closebtn {
  position: absolute;
  top: 0;
  right: 10px;
  font-size: 36px;
  margin-left: 50px;
}

.welcome {
    background-color: #00224B;
}

.welcome i {
    margin-top: 5px;
    margin-bottom: 20px;
    padding-top: 10px;
    padding-left: 30px;
    font-size: 24px;
    color: #ffc04c;
}

.welcomeText {
    color: white;
    font-size: 24px;
}

.main_sections {
    margin-top: 20px;
    margin-bottom: 20px;
}
```

As you can see in the first screenshot, there is a text saying Hello. Next to that text, I wanted to print the name of the user if the user was logged in, and if not just print Sign in. So as I did something similar previously, I knew that I needed to use the $_SESSION to get the user's name using the object class. This is why I did:



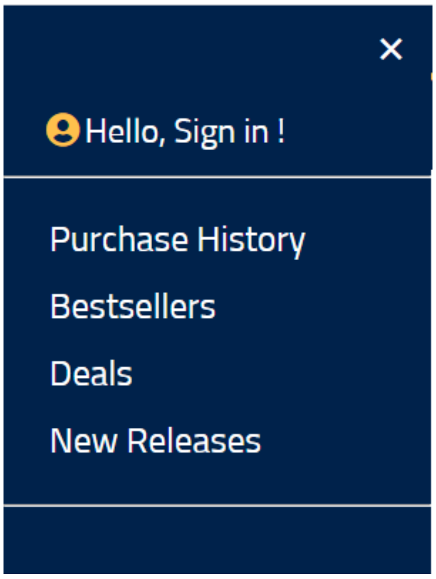
```
<span class="welcomeText">
    Hello, <?php echo $_SESSION["myUser"]->getFirstName(); ?>
</span>
```



However, when I tested this by logging out to see if it says (Hello, Sign in) as the user is not logged in, I encountered an error. The error was that the index page wasn't loading at all, so after a little while of researching, I found out that I needed to use an if statement to check if the item in the session is set by using the in-built function isset(), and if it's set its means there is a user in the session, which means the user is logged in, so just print the user's name, however, if it's not set, it means the user is not logged in so print Hello, Sign in.

```
<span class="welcomeText">
    Hello,
    <?php
        if(isset($_SESSION["myUser"])) {
            echo $_SESSION["myUser"]->getFirstName();
        } else {
            echo "Sign in";
        }
    ?>
    !
</span>
```

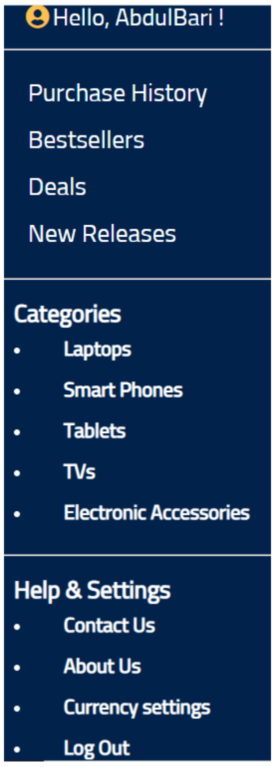
To test this, I logged out and it said Hello, Sign in, which means it was successful.



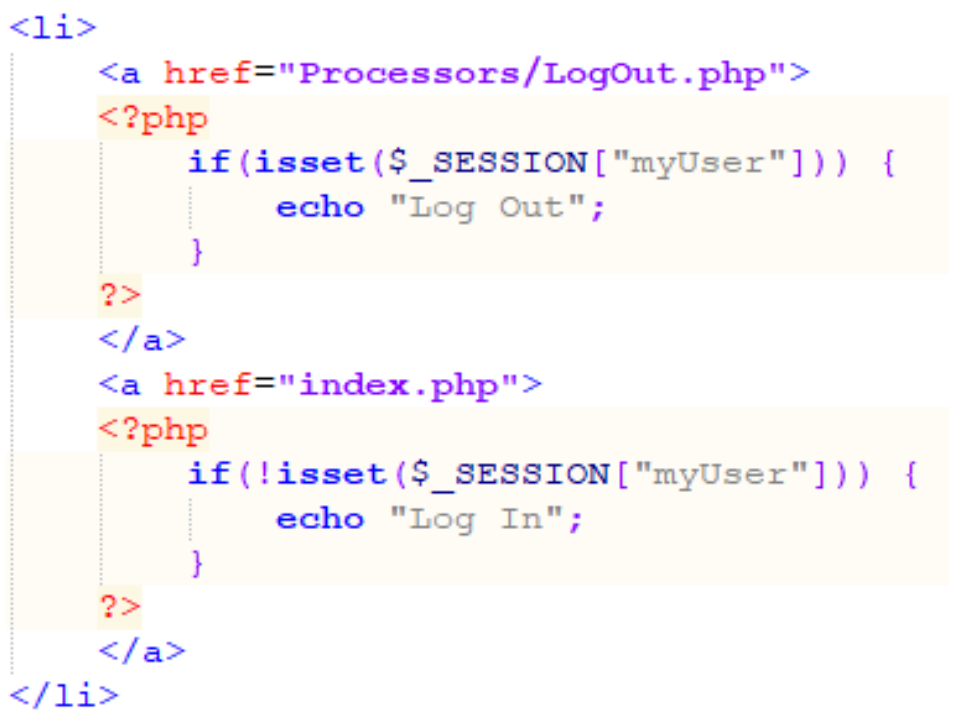
Date:15/09/2021

Task: Completing the sidebar navigation

I completed the sidebar navigation by adding several links to the main sections of the website as shown in the screenshot:



As you can see at the bottom of the sidebar I added a link that says “Log Out”. When the user is logged in it allows the user to log out and when the user is signed out it turns into “Log In" and takes the user to the index page. I implemented this feature using the same way I did for the header of the sidebar “Hello, Sign in". This was done by using the built-in function isset(). I did not encounter any issues with this.

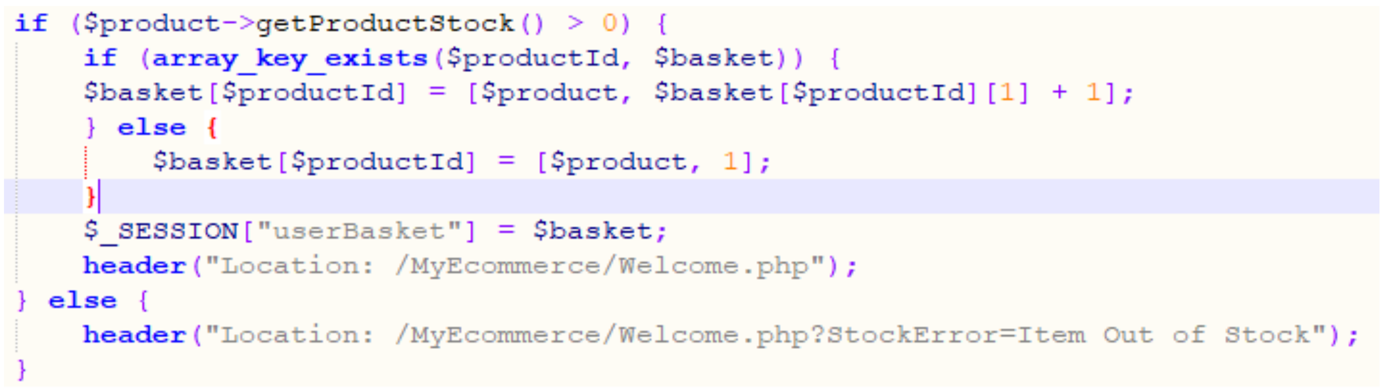


```
<li>
    <a href="Processors/LogOut.php">
    <?php
        if(isset($_SESSION["myUser"])) {
            echo "Log Out";
        }
    ?>
    </a>
    <a href="index.php">
    <?php
        if(!isset($_SESSION["myUser"])) {
            echo "Log In";
        }
    ?>
    </a>
</li>
```

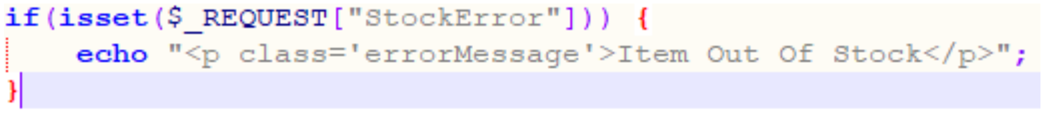
Date:17/09/2021

Task: Product Stock error handling

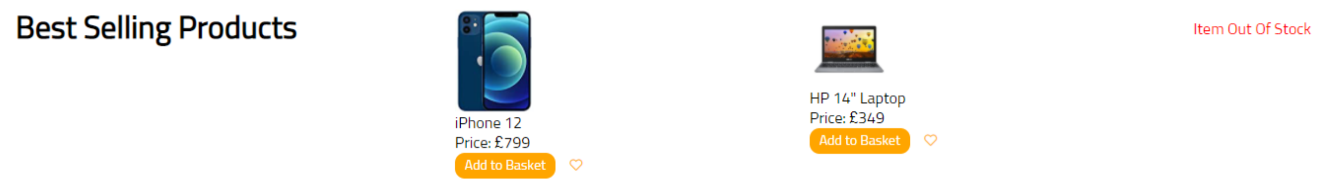
As I was going through the website and checking my work, I remembered that I did not add any error handling for product stock. This means that when a user adds a product to the basket the system does not check if there is actually any available stock for that product. So to solve this, in the addToBasketProcess.php file, where I check if the item already exists in the basket, I added another if statement that checks if the stock of the product is greater than 0 using the getProductStock() function I created in my Product Object class if it's greater than 0 it then checks if it already exists in the basket, and continues, as usual, However, if the stock is less than 0 then it prints an error message. I did not encounter any issues or problems during the implementation of this feature. For this feature, I changed the stock of a product to zero in my database, then I went to the website and added the product into the basket, then I received an error message which indicated that the error handling was working perfectly. However, one thing I was struggling with is the positioning of the error message due to some CSS issues.



```
if ($product->getProductStock() > 0) {
    if (array_key_exists($productId, $basket)) {
    $basket[$productId] = [$product, $basket[$productId][1] + 1];
    } else {
        $basket[$productId] = [$product, 1];
    }
    $_SESSION["userBasket"] = $basket;
    header("Location: /MyEcommerce/Welcome.php");
} else {
    header("Location: /MyEcommerce/Welcome.php?StockError=Item Out of Stock");
}
```



```
if(isset($_REQUEST["StockError"])) {
    echo "<p class='errorMessage'>Item Out Of Stock</p>";
}
```



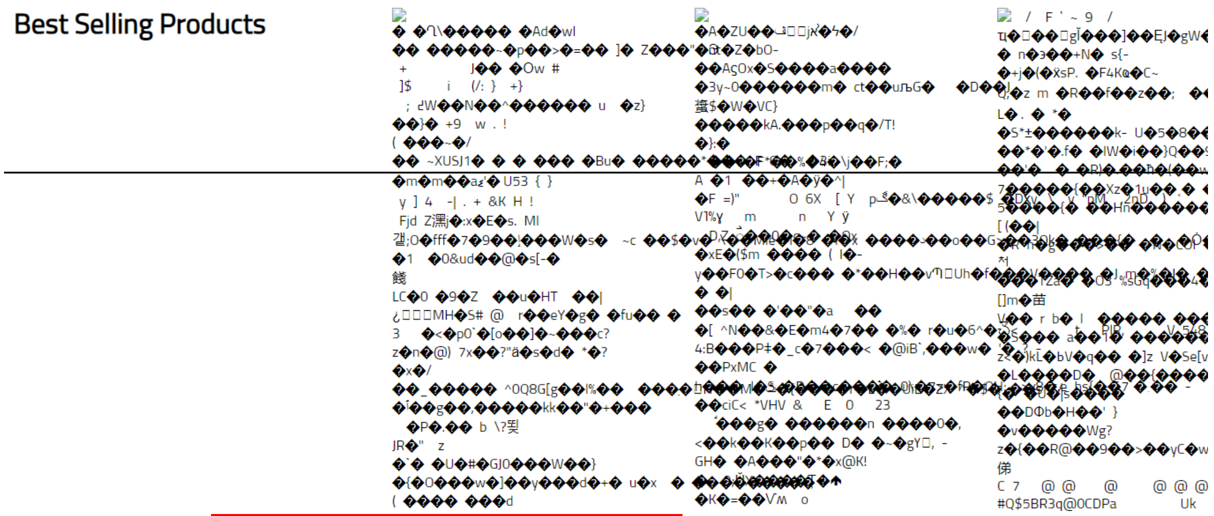
Date:24/09/2021

Task: Loading product photos from the database

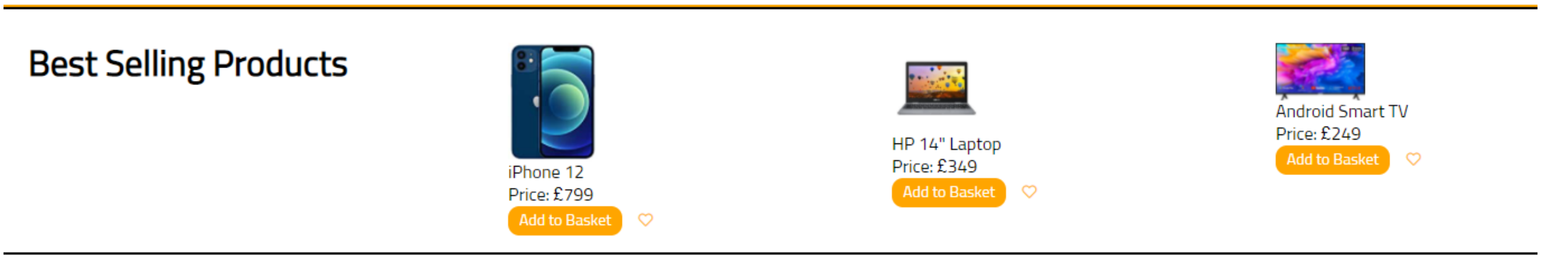
I was writing the directory of the pictures manually in img src in my HTML code just to verify that the styles and the code works properly. So, I decided to make it automated and load it from the database as I was already storing a picture for each product. So in the product object class where I made the printBestSellingProducts() function, instead of writing the directory of the image in img src, I used the variable $productImage in my class:

```
<div id="image'.$this->productID.'">
    <img src="'.$this->productImage.'" alt="'.$this->productName.'" style="width:40%">
</div>
```

However, I encountered an error as I refreshed the welcome page,



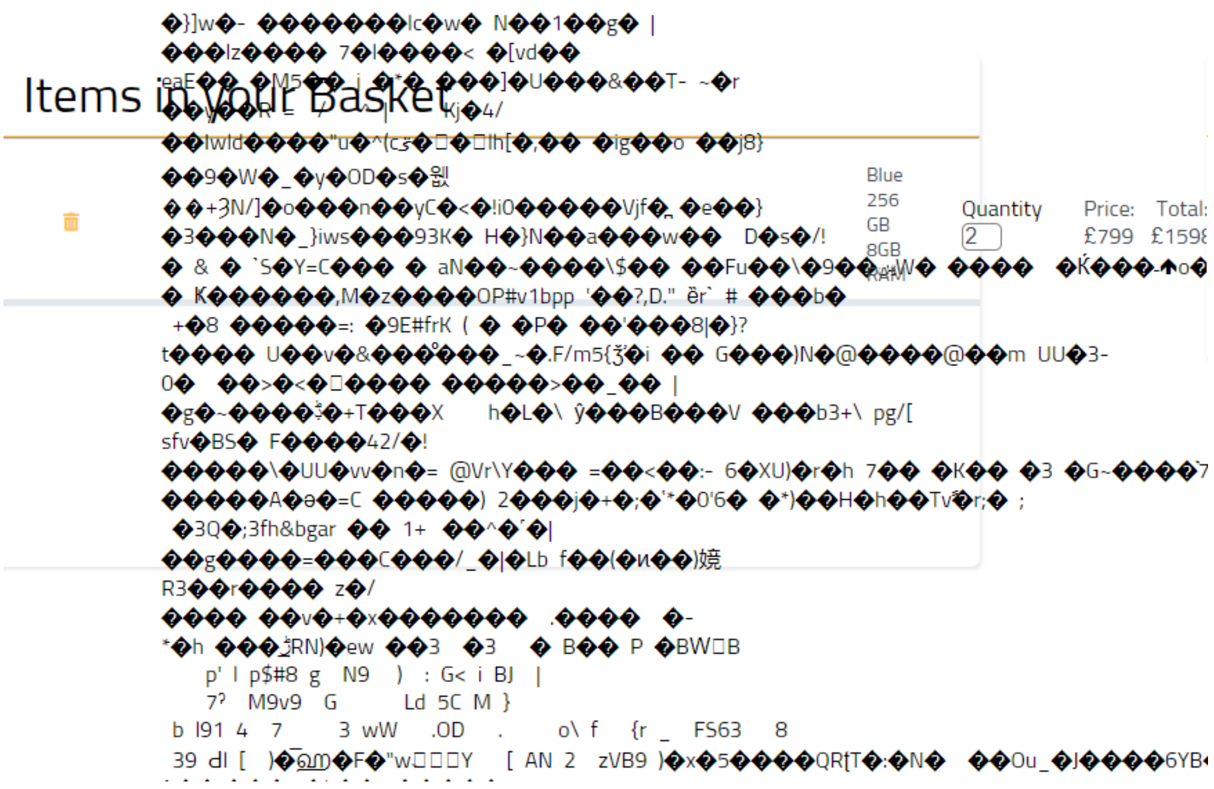
As we can see, the pictures weren't loading properly. I researched the problem and I found out that the pictures were appearing in this unusual way because it was stored as BLOB data type in the database, therefore I had to encode the pictures to make them appear

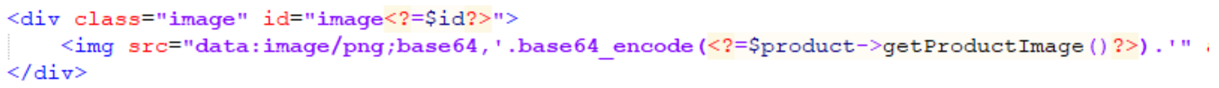


correctly.
So I searched for ways of encoding the pictures. I discovered that I could use base64_encode. This is the code I ended up using:

```
<div id="image'.$this->productID.'">
                    <img src="data:image/png;base64,'.base64_encode($this->productImage).' "
alt=" '.$this->productName.' " style="width:40%">
                </div>
```

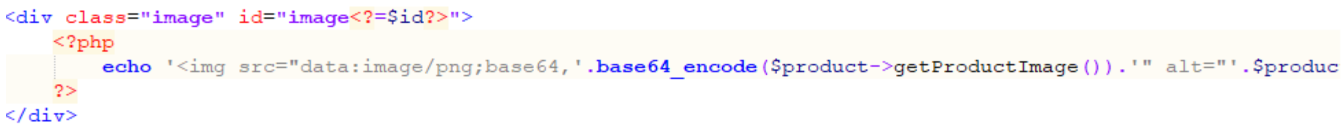
I refreshed the welcome page to check if it is working properly and this was the result:
I added the same code for img src in the Basket page as well as the My Items page so when the user adds a product the pictures appear there as well, but I faced the same problem before despite adding the exact identical line of code apart from the $productImage variable which I changed it into getProductImage().





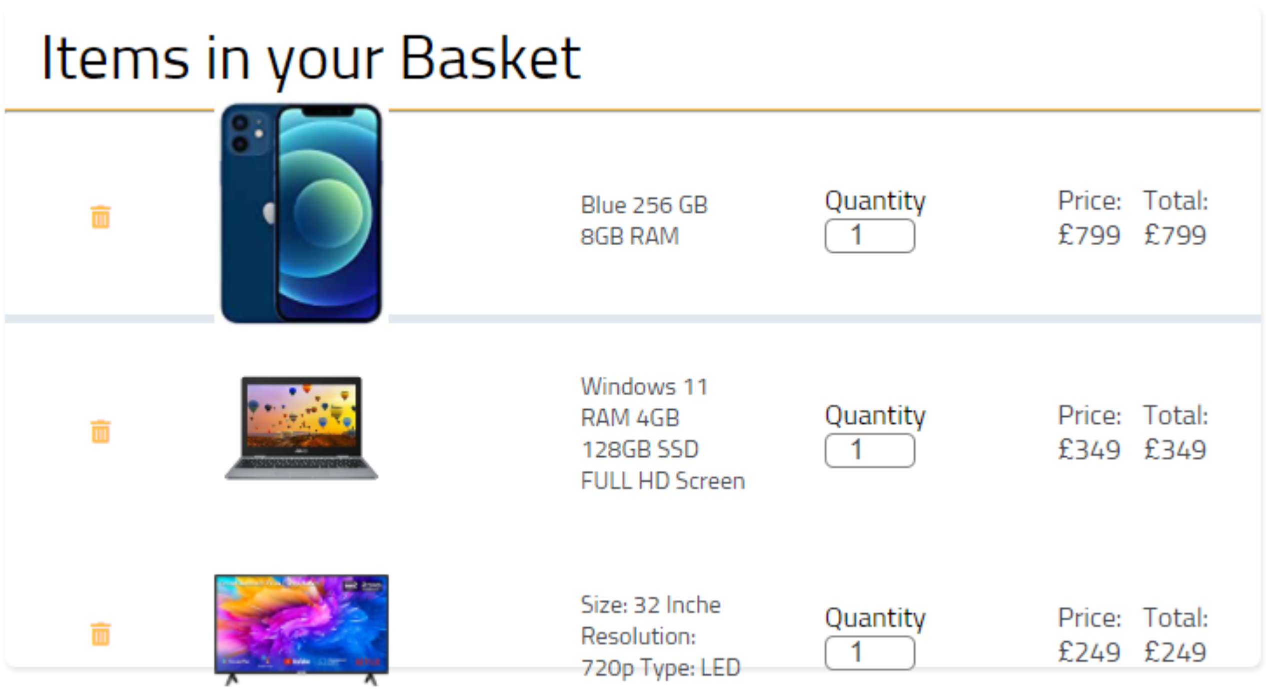
```
<div class="image" id="image<?=$id?>">
    <img src="data:image/png;base64,'.base64_encode(<?=$product->getProductImage()?>).'" ,
</div>
```

After trying and researching several ways of solving the problem, I discovered that I can copy the line of code that is in the Product Object Class and echo it on the basket page.



```
<div class="image" id="image<?=$id?>">
    <?php
        echo '<img src="data:image/png;base64,'.base64_encode($product->getProductImage()).'" alt="'.$produc
    ?>
</div>
```

To test this, I added a product to the basket and refreshed the page and this was the result:



Items in your Basket

	Blue 256 GB 8GB RAM	Quantity 1	Price: £799	Total: £799
	Windows 11 RAM 4GB 128GB SSD FULL HD Screen	Quantity 1	Price: £349	Total: £349
	Size: 32 Inche Resolution: 720p Type: LED	Quantity 1	Price: £249	Total: £249

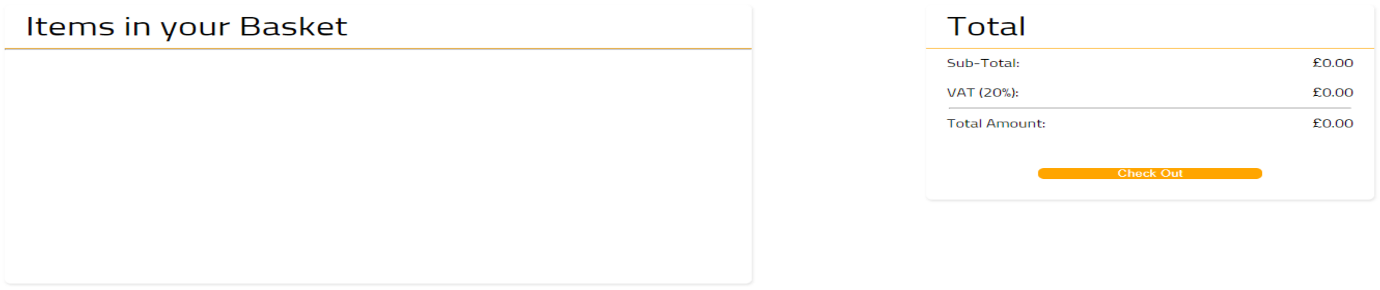
Date:29/09/2021

Task: Building a Total Price section on the Basket Page

I started by creating another div in the basket page called totalPrice that shows the user the total price of all the products added up plus the VAT, then use these 2 values to calculate the total amount of the whole transaction. At the bottom of the div, it allows the user to click on a button to be redirected to the payment page. During the phase of designing the div, the only CSS problem I encountered was that I did not know how to place the Items div and the totalPrice div next to each other, so after a few attempts and trying different methods I discovered that I could use float property in CSS to place them next to each other, so I added float: left for the items div and right for the totalPrice div, then I tested it and it worked successfully. The Next step was to add up the total price of all the items in the basket and display them in the totalPrice div. I implemented this using PHP. As the basket is stored in the session, I used that to loop through the associative array(an array that stores the id and price of each product in the basket) and stored the total price of each product to a variable called total. To calculate the VAT, I multiplied the total price by 0.2 and stored it in a variable called vat. Finally, to calculate the total amount, I created another variable to store the sum of the total price and the vat.

```php
<?php
    $total = 0;
    $vat = 0;
    $totalAmount = 0;
    foreach($basket as $id=>$details) {
        $product = $details[0];
        $total = $total + $product->getProductPrice() * $details[1];
    }
    $vat = $total * 0.2;
    $totalAmount = $total + $vat;
?>
```

So using these variables, I displayed the information required for each label in the totalPrice div, and this was the final result:


Items in your Basket

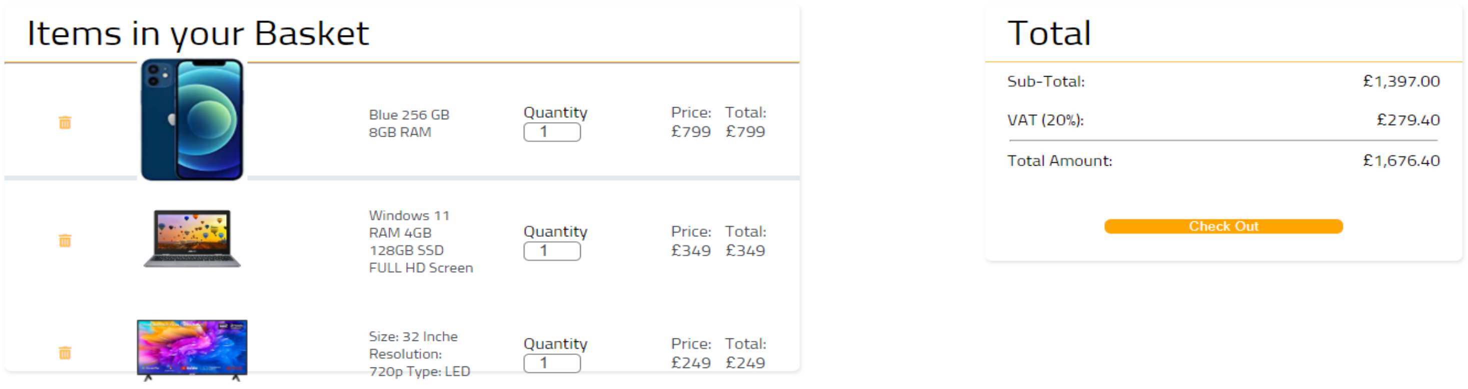
Total

Sub-Total: £0.00
VAT (20%): £0.00
Total Amount: £0.00

One issue I faced was that I struggled a little bit to display the prices in 2 decimal places, however, after I searched for ways to do it, I figured out I could use the built-in function number_format() to specify the decimal points.

```
<span class="costs" >Sub-Total:<span style="margin-left:180px;"> <?php echo "£" . number_format($total, 2); ?>
<span class="costs">VAT (20%):<span style="margin-left:190px;"> <?php echo "£" . number_format($vat, 2); ?></s
```

To test this, I added a few products to the basket and this was the result:

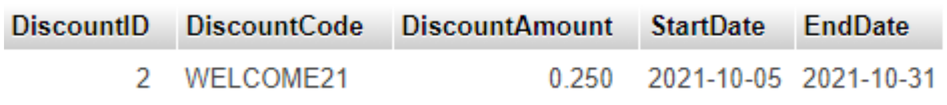

Items in your Basket

	Product	Quantity	Price:	Total:
Blue 256 GB 8GB RAM		1	£799	£799
Windows 11 RAM 4GB 128GB SSD FULL HD Screen		1	£349	£349
Size: 32 Inche Resolution: 720p Type: LED		1	£249	£249

Total

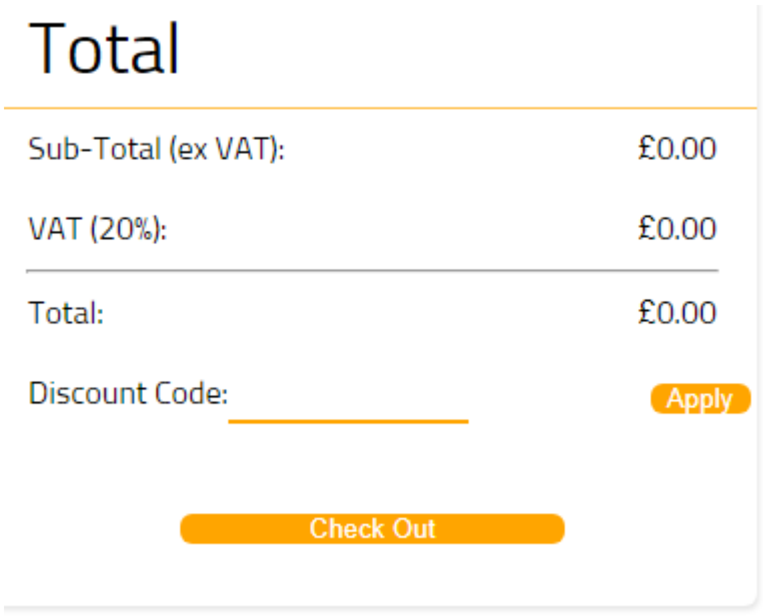
Sub-Total: £1,397.00
VAT (20%): £279.40
Total Amount: £1,676.40

After doing some research, I was able to identify the fields that I needed for the database which were the following:



DiscountID	DiscountCode	DiscountAmount	StartDate	EndDate
2	WELCOME21	0.250	2021-10-05	2021-10-31

The next step was to create an input that allows the user to enter a discount code. I created this input in the total price div on the basket page.



Total

Sub-Total (ex VAT):	£0.00
VAT (20%):	£0.00
Total:	£0.00
Discount Code:	Apply

Check Out

Date:08/10/2021

Task: Building a discount system pt.2

I created a PHP file called discountProcess.php. So when the user types in a discount code and clicks apply, this page processes the whole transaction and checks whether the discount is valid or not and if it exists in the database, etc. If everything is correct then it applies the discount and redirects the user to the basket page, otherwise, it will produce an error message.
Once I created the page, I needed to load the discount code from the database. I was struggling a little bit because this was my first time loading data from a database without having an object for that table, therefore it was quite confusing at the beginning. This is because I was using prepared statements, so this was my query:

```
$mysqli = new mysqli("localhost", "root", "", "e-commerce");
    $stmt = $mysqli->prepare("SELECT * FROM `ediscounts` WHERE
`StartDate` <= CURDATE() AND `EndDate`>= CURDATE() AND `DiscountCode` =
?");
```

As we can see the 'DiscountCode' variable is set to a question mark, which means that this will be whatever discount code the user puts in the discount code input. Therefore, I wasn't sure how to retrieve the discount code that was inputted by the user in the basket page in the discountProcess file. I carried out some research, then I realized that I'd come across this kind of problem earlier in my project and I solved it by using the <from> tag which has the attributes action and method which can be used to post data.

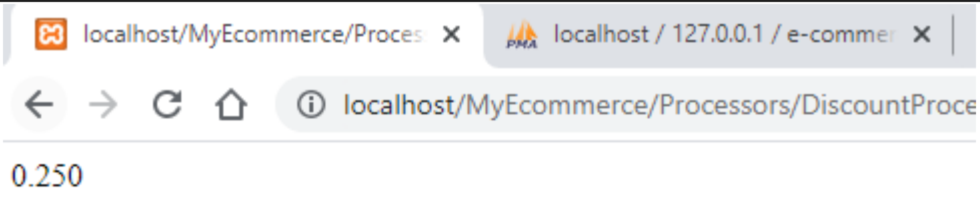
```
<form action="Processors/DiscountProcess.php" method="post">
...
<span class="costs">Discount Code: <input class="textInput" type="text"
name="discountCode" /><span style="margin-left:90px;"><input
type="submit" class="applyBtn" value="Apply"></input></span></span>;
```

Then I used the variable $_POST["name of the input"] in the discountProcess.php file to get the data from the basket page and bind it to the variable from the database.

```
$mysqli = new mysqli("localhost", "root", "", "e-commerce");
    $stmt = $mysqli->prepare("SELECT * FROM `ediscounts` WHERE
`StartDate` <= CURDATE() AND `EndDate`>= CURDATE() AND `DiscountCode` =
?");
    $stmt->bind_param("s", $_POST["discountCode"]);
    $stmt->execute();
    $res = $stmt->get_result();
    $stmt->close();
```

To test that this works properly when I typed in any discount code and clicked apply I printed the discount percentage associated with that discount code.

```
if ($res->num_rows == 1) {
        $row = $res->fetch_assoc();
        $discountAmount = $row["DiscountAmount"];
        echo $discountAmount;
    }
```


localhost/MyEcommerce/Proces
localhost / 127.0.0.1 / e-comme
localhost/MyEcommerce/Processors/DiscountProce

```
0.250
```

Date:15/10/2021

Task: Building a discount system pt.3

At this point, the discount table, the input, and the link between the table and the code were all ready. Therefore, I started thinking about calculating the discount amount and subtracting it from the total and finally displaying it to the user. However, before implementing that, I noticed that the basket is growing massively. This means that there are a lot of features and codes associated with the basket. Therefore, I decided to create a basket object that includes all the code and functions for the basket, this will be very beneficial in the long term because if I wanted to update anything or make any changes or modifications for the basket,

or even add any functionality, I would be able to do all that just in one place which is the basket object. This will increase efficiency and make troubleshooting easier.
I went ahead and created the file and called it BasketObject.php. The file contained 3 private attributes that are the most important for the basket: products, which is an array that holds a product's id and quantity, discount code, and discountPercentage.

```
class Basket {

    private $products;
    private $discountCode;
    private $discountPercentage;

    public function __construct($ps) {
        $this->products = $ps;
        $this->discountCode = null;
        $this->discountPercentage = 0;
    }
```

I set the discountCode as null(empty string) and the percentage as 0 because they will be given values in the discountProcess.php file when the user inputs a discount code.
Until this point, I did not encounter any problems.
After that, I created a function called printBasket(). I used the prototype that is on the Basket page and added it to this function. So, in the basket page instead of having a long HTML code, I could simply call this function and it prints the basket.
Basket.php:

```
<body>
    <?php include "NavigationBar.php"; ?>
    <?php
        $basket = new Basket(array(), array());
        if (isset($_SESSION["userBasket"])) {
            $basket = $_SESSION["userBasket"];
        }
        $basket->printBasket();
    ?>
```

BasketObject.php:

```
function printBasket() {
        $basket = $this->products;
        $html = '<div class="basket">
        <h1 id="title">Items in your Basket</h1>';

        if(isset($_REQUEST["error"])) {
            $html .= '<p>Unable to delete this Item</p>';
        }

        $html .= '<hr>';
        foreach($basket as $id=>$details) {
            $product = $details[0];
            $html .= '<div class="item" id="item'.$id.'">
                <div id="remove">
                    <button id="removeButton"
onclick="removeFromBasket('.$id.')"><i class="fas fa-trash-
alt"></i></button>
                </div>

                <div class="image" id="image'.$id.'">
                    <img
src="data:image/png;base64,'.base64_encode($product-
>getProductImage()).'" alt="'.$product->getProductName().'"
style="width:50%">

                </div>

                <div class="description" id="description'.$id.'">
                    <span>'.$product->getProductDes().'</span>
                </div>

                <div class="quantity" id="quantity'.$id.'">
                    <label for="_quantity">Quantity</label>
                    <input onclick="updateTotal('.$id.')"
id="amount'.$id.'" type="number" min="1" value="'.$details[1].'"
style="width:50%">
                </div>

                <div id="itemPrice">
                    <input type="hidden" id="price'.$id.'"
value="$product->getProductPrice()">
                    <span>Price: £'.$product-
>getProductPrice().'</span>
                </div>

                <div class="total "id="total'.$id.'">
                    <label for="_total">Total:</label>
                    <span>£'.$product->getProductPrice() *
$details[1].'</span>
                </div>
            </div>';
        }
```

When I tested this, I faced several syntax errors due to the big change in the code, and the most common ones were errors like:

```
Parse error: syntax error, unexpected '>' in D:\xampp\htdocs\MyEcommerce\Classes\BasketObject.php on line 78
```

This is because now it is a PHP code inside an HTML code inside a PHP code, whereas before it was only a PHP code inside an HTML code, and so there is a small difference in syntax.
This is how it used to be:

```
<div class="description" id="description<?=$id?>">
                    <span><?=$product->getProductDes()?></span>
                </div>
```

This is how it suppose look:

```
<div class="description" id="description'.$id.'">
                    <span>'.$product->getProductDes().'</span>
                </div>
```

As we can see that every time I used the variable $id or the products class to call a function, instead of using <?=...?> I had to change it to PHP concatenation '.   .'. I had to do this for every div inside the printBasket() function to fix the problem.
To test this, I added a product to the basket to see whether it looks like before or if there were any further errors. The result was good and the basket and all its products appeared successfully.
Next, I added a couple of other functions such as addToBasket() and removeFromBasket() which simply contained the code that was written in addToBasketProcess.php and removeFromBasketProcess.php files.

```
function addToBasket($product) {
        $productId = $product->getProductID();
        if (array_key_exists($productId, $this->products)) {
            $this->products[$productId] = [$product, $this-
>products[$productId][1] + 1];
        } else {
            $this->products[$productId] = [$product, 1];
        }
    }

    function removeFromBasket($product) {
        $productId = $product->getProductID();
        if (array_key_exists($productId, $this->products)) {
            unset($this->products[$productId]);
            return true;
        } else {
            return false;
        }
    }
```

addToBasketProcess.php (after creating the above functions):

```
if ($product->getProductStock() > 0) {
        $basket->addToBasket($product);
        $_SESSION["userBasket"] = $basket;
        header("Location: /MyEcommerce/Welcome.php");
    } else {
        header("Location: /MyEcommerce/Welcome.php?StockError=Item Out
of Stock");
    }
```

removeFromBasket.php (after creating the above functions):

```
if ($basket->removeFromBasket($product)) {
        $_SESSION["userBasket"] = $basket;
        header("Location: /MyEcommerce/Basket.php");
    } else {
        header("Location: /MyEcommerce/Basket.php?error=Item not
found");
    }
```

One problem I faced after testing all these functions was with the removeFromBasket function. The problem was in the else {} statement because if the item can't be deleted the user must be redirected to the basket page and receive an error message. So, I wasn't able to implement this inside the removeFromBasket() function and it wasn't going to work if I didn't add any else statements. Therefore, the solution was using boolean variables. When an item with a specific product id was found, then unset() it (delete it) and return true, otherwise, return false. So in the removeFromBasketProcess.php if the function returns true then redirect to the basket page and update the basket, if it returns false then produce an error message. I tested the function again and the product was getting deleted successfully.

Date:20/10/2021

Task: Building a discount system pt.4

The next step was to create 2 functions, the first one addDiscountCode() which links the private variables discountPercentage and discountCode in the class with the actual values from the database in the discountProcess.php file. The function should take 2 parameters, code and percentage and assign those parameters to the private variables. I called this function in the discountProcess.php file and passed the code and the percentage from the database as arguments.

BasketObject.php

```
function addDiscountCode($code, $percentage) {
    $this->discountCode = $code;
    $this->discountPercentage = $percentage;
}
```

discountProcess.php

```php
$row = $res->fetch_assoc();
$discountAmount = $row["DiscountAmount"];
$basket = new Basket(array());
if (isset($_SESSION["userBasket"])) {
    $basket = $_SESSION["userBasket"];
}
$basket->addDiscountCode($_POST["discountCode"], $discountAmount);
$_SESSION["userBasket"] = $basket;
header("Location: /MyEcommerce/Basket.php");
```

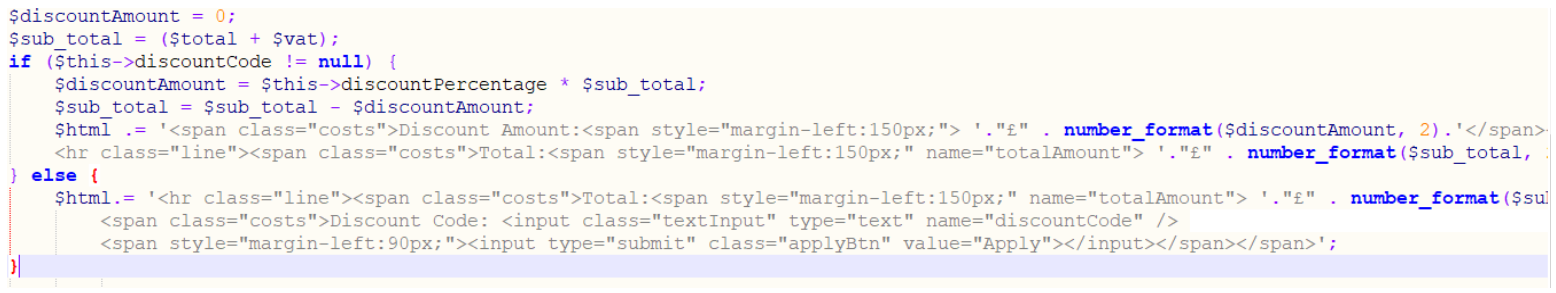
The second function I called it getTotal() which basically calculates the total price of the products as well as VAT and returns an array that holds the total price(ex vat) and vat.

```php
function getTotal() {
    $total = 0;
    $vat = 0;
    $totalAmount = 0;
    foreach($this->products  as $id=>$details) {
        $product = $details[0];
        $total = $total + $product->getProductPrice() * $details[1];
    }
    $vat = $total * 0.2;
    $totalAmount = $total + $vat;
    return [$totalAmount, $vat];
}
```

After that, I had to calculate the discount and display it in the total price div. To do that, firstly I used the above function to get the sub-total (ex vat) and the vat separately to display them in the basket page before any discounts were added.

```php
$totals = $this->getTotal();
$total = $totals[0];
$vat = $totals[1];
$html .= '</div>
    <form action="Processors/DiscountProcess.php" method="post">
    <div class="totalPrice">
    <h1 id="title">Total</h1>
    <span class="costs" >Sub-Total (ex VAT):<span style="margin-left:130px;"> '."£" . number_format($total, 2).'</span></span>
    <span class="costs">VAT (20%):<span style="margin-left:190px;"> '."£" . number_format($vat, 2).'</span></span>';
```

Next, I created a new variable called discount amount to hold the amount that will be discounted from the total price. Then I created an if statement to check if the private attribute discount code variable is not null, if it is not, then multiply the private attribute discount percentage by the sub-total(ex vat). Then subtract the discount amount from the sub-total. Finally, I used all these variables to display the information on the basket page. However, in the else statement where the discount code is not valid or null then simply don't make any changes and keep the total price div the same.

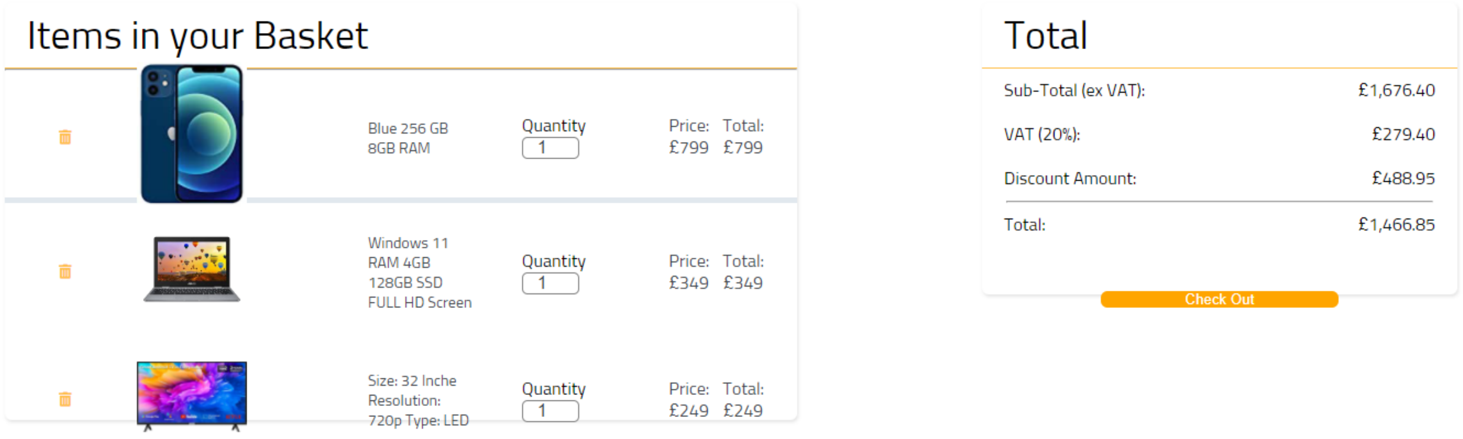


```
$discountAmount = 0;
$sub_total = ($total + $vat);
if ($this->discountCode != null) {
    $discountAmount = $this->discountPercentage * $sub_total;
    $sub_total = $sub_total - $discountAmount;
    $html .= '<span class="costs">Discount Amount:<span style="margin-left:150px;"> '."£" . number_format($discountAmount, 2).'</span>
    <hr class="line"><span class="costs">Total:<span style="margin-left:150px;" name="totalAmount"> '."£" . number_format($sub_total,
} else {
    $html.= '<hr class="line"><span class="costs">Total:<span style="margin-left:150px;" name="totalAmount"> '."£" . number_format($sul
        <span class="costs">Discount Code: <input class="textInput" type="text" name="discountCode" />
        <span style="margin-left:90px;"><input type="submit" class="applyBtn" value="Apply"></input></span></span>';
}
```

During this process, I only encountered small and simple errors such as syntax and mathematical errors when calculating the discount. However, there weren't any major errors.

To test this, I added products to the basket and applied a discount code and it worked successfully. This was the final result:

Valid discount Code:



Items in your Basket

Blue 256 GB
8GB RAM
Quantity 1
Price: £799 Total: £799

Windows 11
RAM 4GB
128GB SSD
FULL HD Screen
Quantity 1
Price: £349 Total: £349

Size: 32 Inche
Resolution:
720p Type: LED
Quantity 1
Price: £249 Total: £249

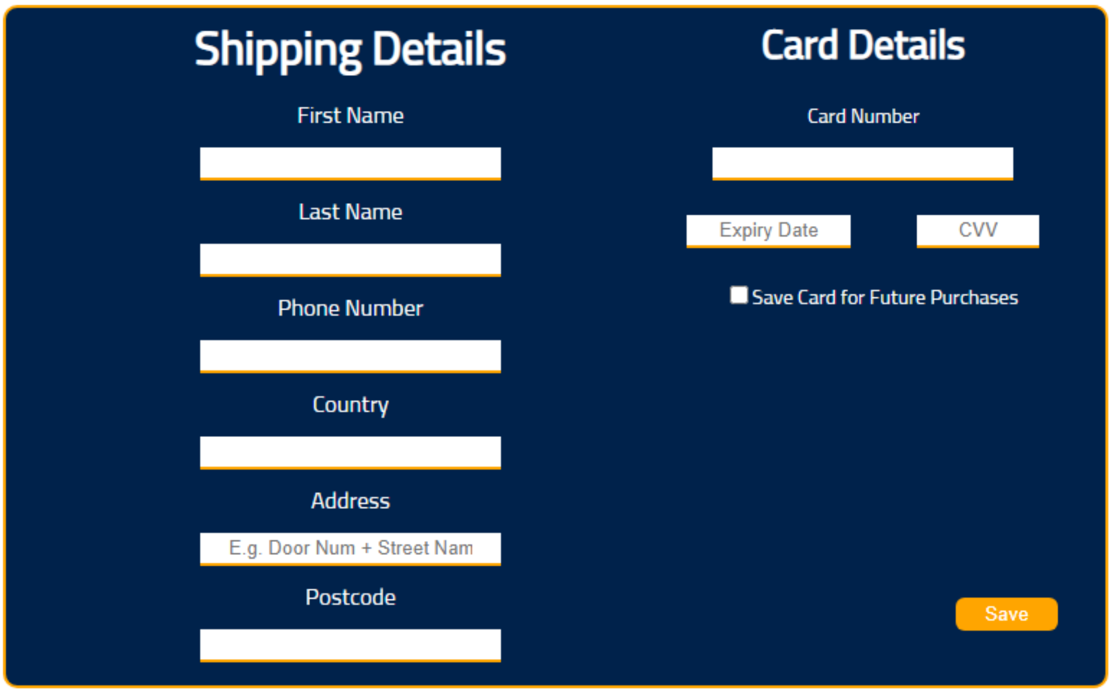
Total

Sub-Total (ex VAT):	£1,676.40
VAT (20%):	£279.40
Discount Amount:	£488.95
Total:	£1,466.85

Date:03/11/2021

Task: Building the checkout page

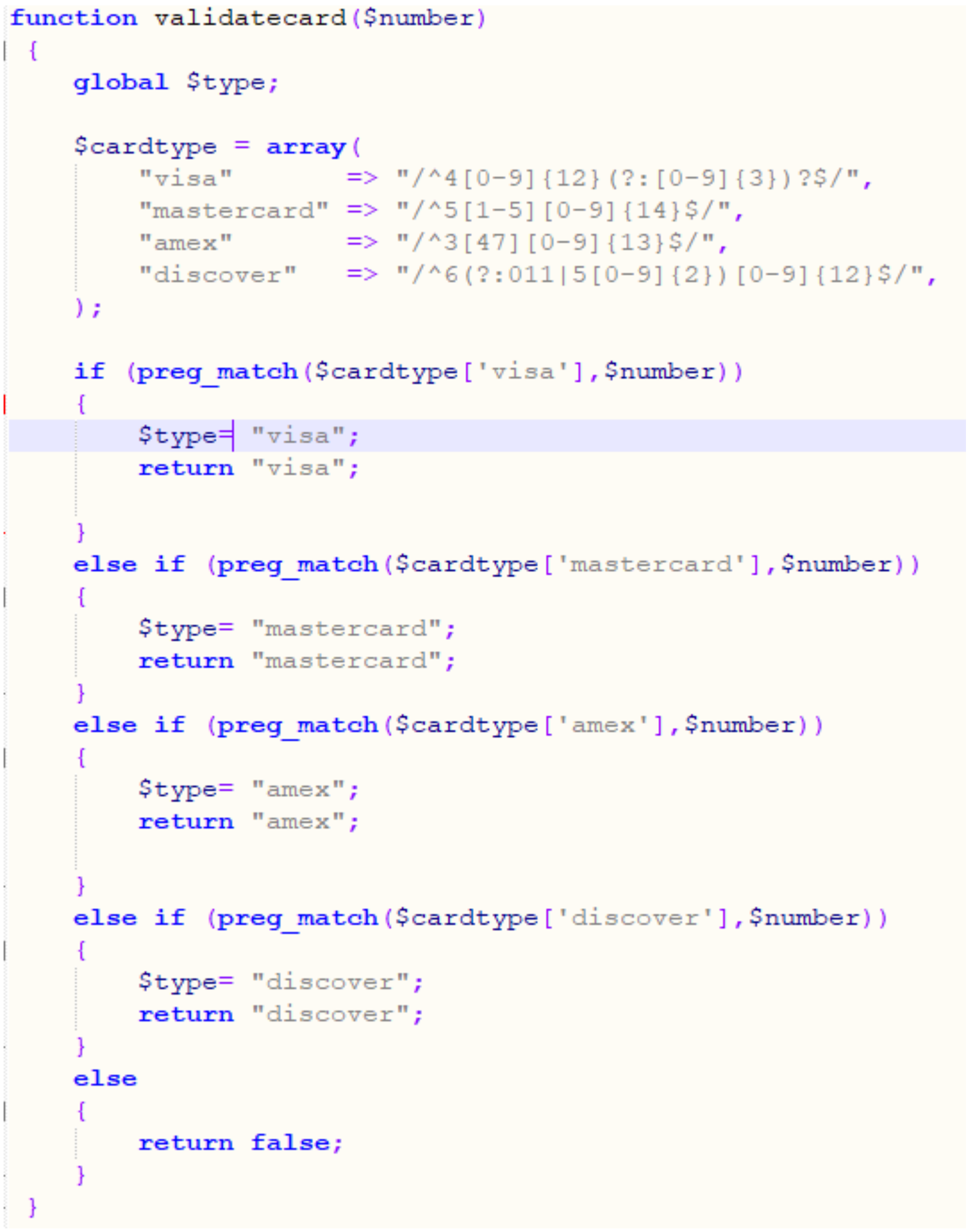
I started by setting up all the PHP files that I will need for this module of the project which is the checkout system. I created the following files: Checkout.php which is the main page, DoCheckout.php (processor) and checkoutStyles.php. Building the page was an easy task as I have done similar tasks several times before, therefore it did not take a long time and I did not encounter any issues. This was the final result:



Date:05/11/2021

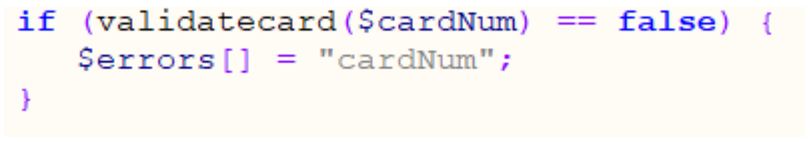
Task: Validating inputs

As the majority of the fields on the checkout page are similar to the fields on the register page, I used the code that I have produced in the DoRegister.php file to validate the inputs in the checkout page. However, as you can see in the screenshot, there are a few extra new fields: card number, expiration date and CVV, therefore, I needed to add different types of validation for these new fields but I was unsure how to implement it. After a little bit of research, I found out that I can create a function that takes one parameter Card Number, and then inside the function, I can create an array that holds the regex for the different types of card (e.g. visa, MasterCard). Then after that, I can use the built-in function preg_match() to validate the Card Number according to their type. The function returns false if the card number is not valid.



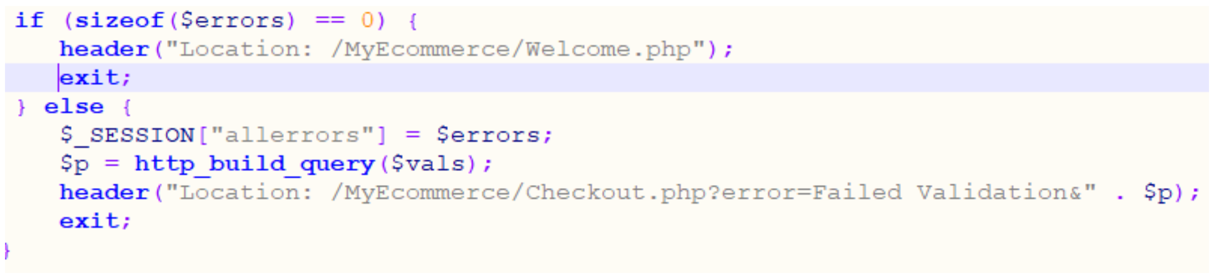
```
function validatecard($number)
 {
    global $type;

    $cardtype = array(
        "visa"       => "/^4[0-9]{12}(?:[0-9]{3})?$/",
        "mastercard" => "/^5[1-5][0-9]{14}$/",
        "amex"       => "/^3[47][0-9]{13}$/",
        "discover"   => "/^6(?:011|5[0-9]{2})[0-9]{12}$/",
    );

    if (preg_match($cardtype['visa'],$number))
    {
        $type= "visa";
        return "visa";

    }
    else if (preg_match($cardtype['mastercard'],$number))
    {
        $type= "mastercard";
        return "mastercard";
    }
    else if (preg_match($cardtype['amex'],$number))
    {
        $type= "amex";
        return "amex";

    }
    else if (preg_match($cardtype['discover'],$number))
    {
        $type= "discover";
        return "discover";
    }
    else
    {
        return false;
    }
 }
```

Outside the function, I created an if-statement that checks if the function returns false, if it does then it adds the error to the errors array.



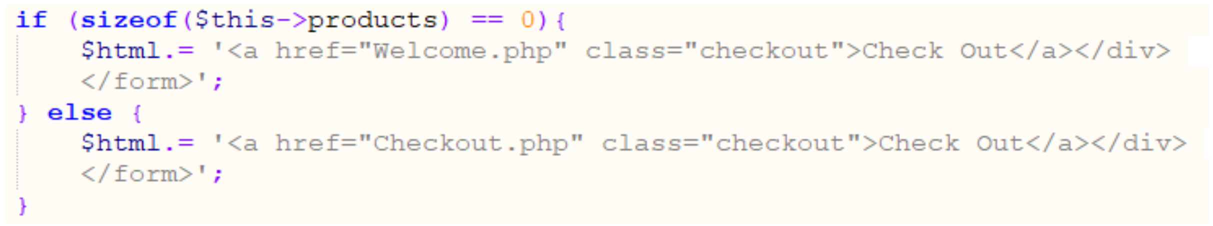
```
if (validatecard($cardNum) == false) {
    $errors[] = "cardNum";
}
```

To test this, I have inputted correct data into the fields to see whether it directs to the welcome page and it successfully did, after that I inputted invalid data and I received an error, which indicates that it works perfectly.



```
if (sizeof($errors) == 0) {
   header("Location: /MyEcommerce/Welcome.php");
   exit;
} else {
   $_SESSION["allerrors"] = $errors;
   $p = http_build_query($vals);
   header("Location: /MyEcommerce/Checkout.php?error=Failed Validation&" . $p);
   exit;
}
```

One last thing I did which I forgot to do when I was building the total price div is error handling for the Checkout button. So, in the basket object file, I made an if statement that checks whether there is a product in the basket or not, if there is then allow the user to be redirected to the checkout page, if not then redirect to the welcome page.



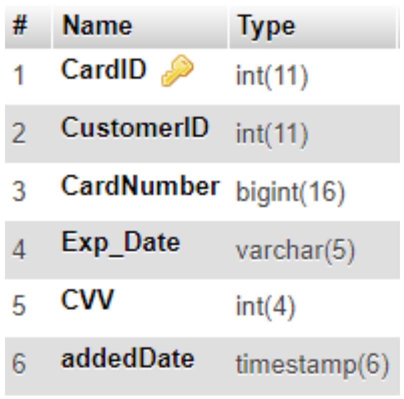
```
if (sizeof($this->products) == 0){
    $html.= '<a href="Welcome.php" class="checkout">Check Out</a></div>
    </form>';
} else {
    $html.= '<a href="Checkout.php" class="checkout">Check Out</a></div>
    </form>';
}
```

I tested this by adding products to the basket and clicking the checkout button and it successfully redirected me to the checkout page, then I removed the product and it redirected me to the welcome page, and so it was working perfectly.

Date:08/11/2021

Task: Creating payment methods table

The next step was to create the link between the database and the checkout page. However, before that, I needed to create a table in order to store the card details if the user checked the save card checkbox. I wanted to use my Design folder to add the fields in the data, however, when I had a look at the data dictionary, I realised that I don't need to store all the data (e.g. first name, address) again as its already stored in the users' table, but I will, however, store it in the orders table. These are the fields I ended up using:



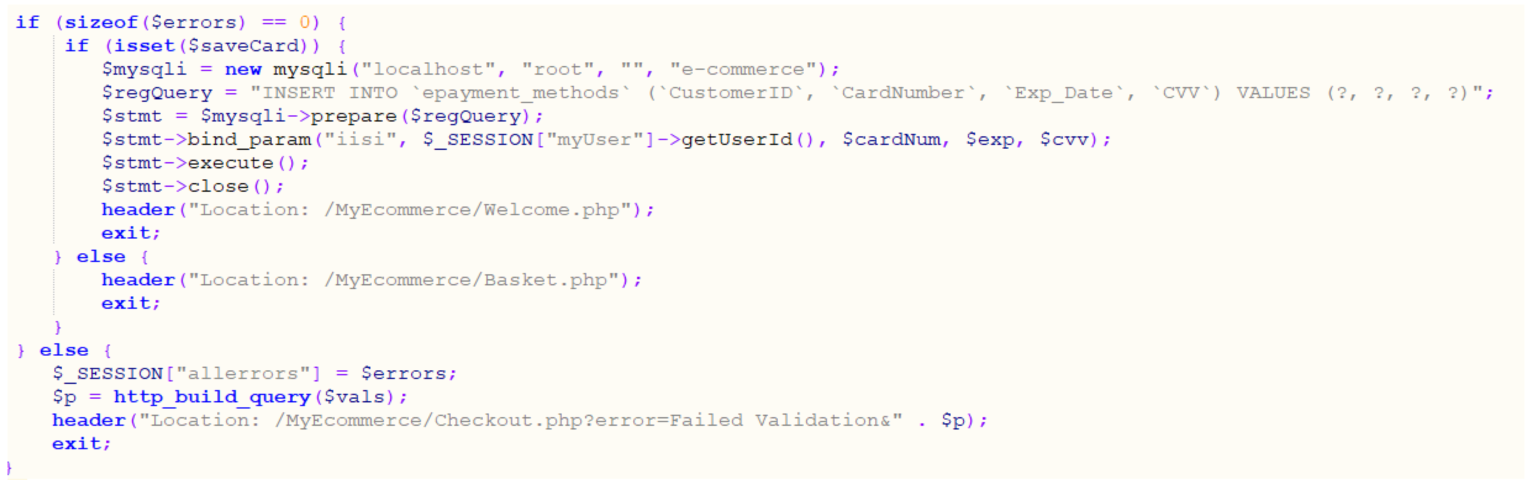
#	Name	Type
1	CardID 🔑	int(11)
2	CustomerID	int(11)
3	CardNumber	bigint(16)
4	Exp_Date	varchar(5)
5	CVV	int(4)
6	addedDate	timestamp(6)

One Error I was faced with is regarding the Exp_Date field. Initially, I set the type of the field as Date. When I was adding a record to the table manually to test if all data appears correctly, I was getting all zeros(00-00-0000) in the exp_date field as I was entering the expiry date as (mm/yy) only because this is how it appears on a credit/debit card. However, the solution was very simple which was just changing the type from Date to VARCHAR as shown in the screenshot above.

Date:17/11/2021

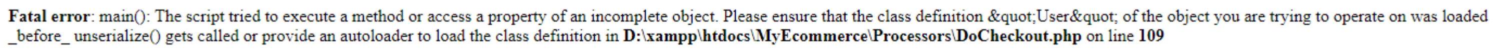
Task: Linking the payment methods table with the code

The next step was to make the checkout form actually function. Therefore, I started by checking if there were no errors in terms of the validation of inputs, if that is true then check if the user has checked the save card checkbox, if yes then create a link between the database and the table and use prepared statements to insert the data into the table.
This was the final code:



```
if (sizeof($errors) == 0) {
    if (isset($saveCard)) {
        $mysqli = new mysqli("localhost", "root", "", "e-commerce");
        $regQuery = "INSERT INTO `epayment_methods` (`CustomerID`, `CardNumber`, `Exp_Date`, `CVV`) VALUES (?, ?, ?, ?)";
        $stmt = $mysqli->prepare($regQuery);
        $stmt->bind_param("iisi", $_SESSION["myUser"]->getUserId(), $cardNum, $exp, $cvv);
        $stmt->execute();
        $stmt->close();
        header("Location: /MyEcommerce/Welcome.php");
        exit;
    } else {
        header("Location: /MyEcommerce/Basket.php");
        exit;
    }
} else {
    $_SESSION["allerrors"] = $errors;
    $p = http_build_query($vals);
    header("Location: /MyEcommerce/Checkout.php?error=Failed Validation&" . $p);
    exit;
}
```

One error I was faced with during this process is the CustomerID parameter. The customer Id is a foreign key from the users' table, therefore, I had to use the User object to retrieve the user id and I did that using the session as shown in the above screenshot. However, I was receiving an error:



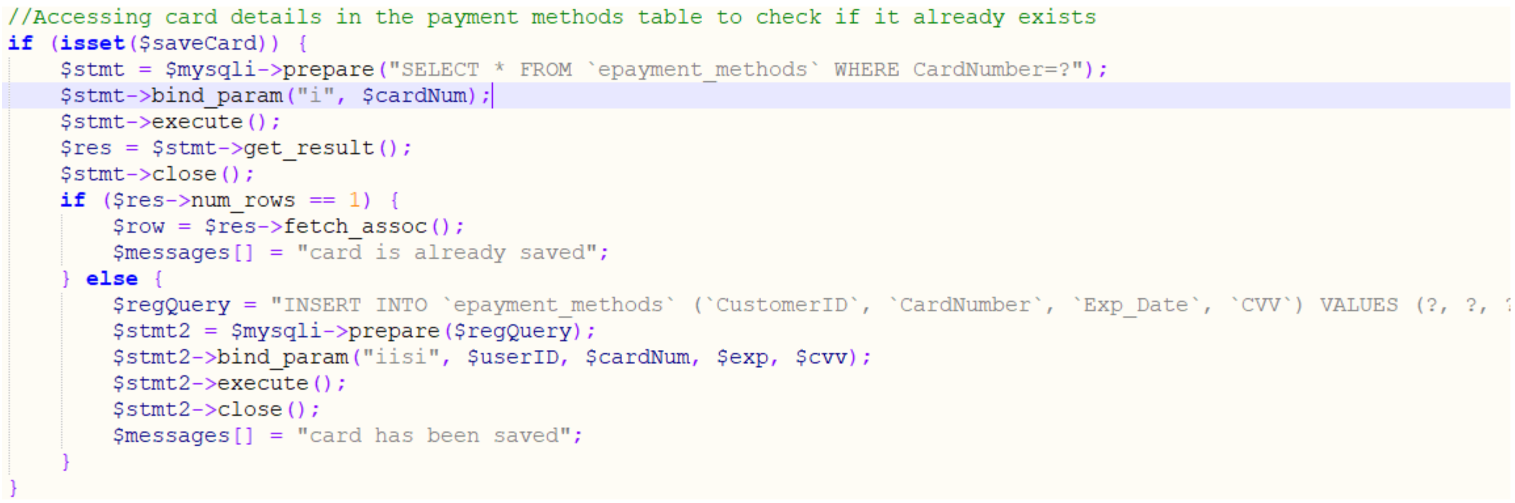
Fatal error: main(): The script tried to execute a method or access a property of an incomplete object. Please ensure that the class definition &quot;User&quot; of the object you are trying to operate on was loaded _before_ unserialize() gets called or provide an autoloader to load the class definition in **D:\xampp\htdocs\MyEcommerce\Processors\DoCheckout.php** on line **109**

In the beginning, I thought that the problem was to do with the session. For example, maybe I have forgotten to start the session on one of the pages, but this was not the issue. After hours of researching, I found out that I forgot to require the user object file in the DoCheckout.php. Therefore, I added the requrie_once() function to include the user object file, and I tested the code by filling in the checkout form. The data was successfully added to the payment methods table in the DB.

Date:01/12/2021

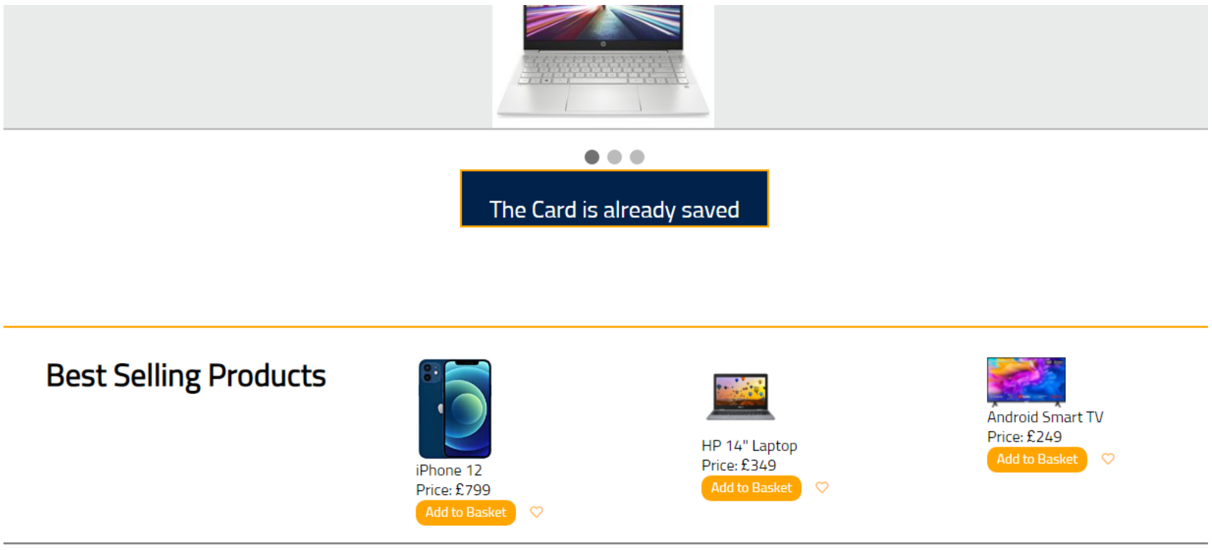
Task: Fixing the save card checkbox

Although adding the card details was working perfectly, however, there was a problem with the checkbox that saves the card. The problem was that even if the card is already saved in the database the system would store it again which causes repetition of data. I fixed this by accessing the database and checking whether that particular card number already exists or not before I added anything to the database. If any record with that exact card number is found and the save card checkbox was set to true, then print a message saying that the card is already saved. Else just redirect the user to the welcome page. However, if no record is found and the save card check box is set to true then add it to the database straight away. This is the code I added in the checkout process file:



```
//Accessing card details in the payment methods table to check if it already exists
if (isset($saveCard)) {
    $stmt = $mysqli->prepare("SELECT * FROM `epayment_methods` WHERE CardNumber=?");
    $stmt->bind_param("i", $cardNum);
    $stmt->execute();
    $res = $stmt->get_result();
    $stmt->close();
    if ($res->num_rows == 1) {
        $row = $res->fetch_assoc();
        $messages[] = "card is already saved";
    } else {
        $regQuery = "INSERT INTO `epayment_methods` (`CustomerID`, `CardNumber`, `Exp_Date`, `CVV`) VALUES (?, ?, 
        $stmt2 = $mysqli->prepare($regQuery);
        $stmt2->bind_param("iisi", $userID, $cardNum, $exp, $cvv);
        $stmt2->execute();
        $stmt2->close();
        $messages[] = "card has been saved";
    }
}
```

I tested this by using a card number that already exists in the database, and after clicking save/submit, this is the message the user receives:



The Card is already saved

Date:06/12/2021

Task: Adding orders to the Orders table

The first step was to create the orders table in the database. I used the data dictionary in the design section to assist me with creating the table. I deviated a little bit from the original design of the table. This is because, in the design section, I intended to add customer details such as contact information or address in the orders table, however, when I was creating the table I came to realise that I don't need to store all the information again as I'm already storing the customer id in the orders table, which can give me access to the contact details and address of that customer.

Adding to the orders table was not very complicated. In the checkout process file, after the system does all the checks regarding saving the card and after printing any messages informing the user about the submission of an order etc. I linked the orders table with the code, then I used an INSERT query that adds the customer id using the user object class, the payment id from the payment methods table (after checking if that payment method exists in the database), if not then it only stores the card number (for invoicing purposes), also it adds the total price of the order into the database, and finally the timestamp.

I did not encounter any issues with this process as I was already familiar with using insert queries. I tested this feature by creating and submitting an order and then checking the orders table to see if the correct information was inserted, and the test was successful.