Candidate 2 evidence

Fitness for Purpose

The results of testing show that the completed project now meets all of the original requirements, functional as well as end-user requirements. Users can register with the site, login using their registration details and search for cosmetic products. All inputs to the site have been validated as per the requirements of an AH project. Testing with the persona and test cases showed that participants were able to successfully complete all tasks given to them. I carried out all of the tests listed in my test plan and produced evidence for each of the test cases in the plan. I am confident that my solution meets all of the requirements listed in the requirements specification.

Original Requirements (Functional and End-user)	Completed
Store details of customers and products in an external database	✓
Validate all user inputs to the website	✓
Easy to navigate with a fully functioning navigation bar	✓
Clutter free layout with palatable colour scheme	✓
Pages formatted using an external style sheet	✓
Search for product by brand or by product name	✓
Display formatted search results	✓
Session variables used to store a customer's login details across pages of the website	✓
Responsive layout that changes depending on the screen size being used	✓

Maintainability

I believe that my solution is maintainable. I have used meaningful variable names for each HTML input element and also for the PHP variables and the database fields. I have added comment lines on each of my HTML pages to explain the purpose of each input element. Here is a screen-shot showing some of the commentary used in the product.html script.

```
<!--drop down menu so cuustoers can view products-->

<form name="products" method="get" action="products.php" >
<!--the line above sends the infromation inserted by the user to the php page to be validated -->

<select name="brand">

<!--options for the drop down bar -->

    <option value="Anastasia Beverley Hills">Anastasia Beverley Hills</option>
    <option value="Urban Decay">Urban Decay</option>
    </select>

<!--submit button to activate query -->
<input type="submit" name="submit" value="submit"/>
```

On my PHP pages, I have used lots of white space to split the code into several sections that each perform one server-side side. An internal comment at the start of each section makes it easy for other developers to read the code and understand what each section of code is doing. I have also used internal comments to explain the purpose of each PHP variable. This is a screen-shot from the products.php script used to process the search feature of the website.

```
//connect to the database server and select database
$link=mysqli_connect($servername,$username,$password,$password);

//check for successful connection
if (mysqli_connect_error())
   {
   echo "Failed to connect to MySQL: " . mysqli_connect_error();
   }


//assign value submitted by HYML form to PHP variable

$brand = $_GET['brand'];

//create query to fetch brand details from database

$query = "SELECT * FROM product WHERE brand='$brand'";
```

It would have been possible to make use of functions to perform some of the processing on each page. For example, I could have used a function to validate the form input on the registration form but instead, I simply added the validation code to the body of the page. Because of this, my solution is not modular.
I used external CSS to style each page of the website. This means that if any corrective, adaptive or perfective maintenance is required in the future, it will be easier and much less repetitive for developers to edit or change the formatting details because they are held in a single file.

Robustness

The registration page of my website is robust as every single input is validated using server-side code. When creating a new account, the user must provide each one of the required details otherwise an error message will appear telling the user that the detail is required. The code used to do this validation is part of the registration.php script. Here is a small section of that code.

```
//check surname has been entered
if empty($_POST["surname"])) }
    echo "Surname is required";
}

//check address has been entered
if (empty{$_POST["address"])) {
    echo "Address is required";
}
```

When I was work on the implementation, I realised that it was taking me longer that I expected to create working code. In the end, I decided to focus on building a working website. Although I managed to get the login and search features of my website to work correctly, I didn't have time to add similar validation code to the login.php and product.php scripts.