Criterion C - Development

<u>Ingenuity</u>

- Using the image address in the database to display the product
  - Success Criteria: 9
- Skincare quiz
  - Success Criteria: 7
- Showing the rating of a product using stars on the product display page
  - Success Criteria: 9
- Display of home page and ordering products by rating
  - Success Criteria: 4, 5, 8, 9, 10

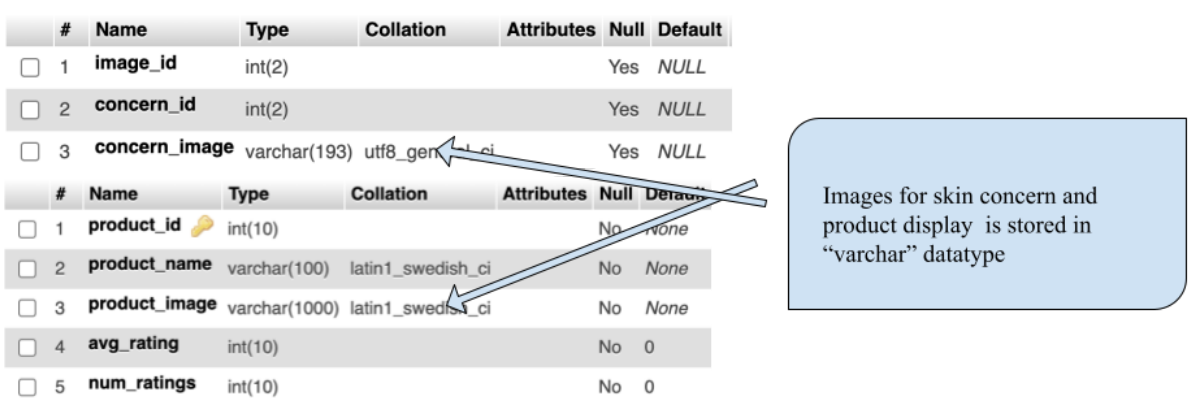
<u>Complexity</u>

- Making the rating functionality work with stars as buttons
  - Success Criteria: 13
- Searching and sorting for products
  - Success Criteria: 14, 15, 16
- Product recommendations based on quiz responses
  - Success Criteria: 8, 10
- Remove concern functionality
  - Success Criteria: 6

I. Using the image address in the database to display the product

Success Criteria: 9

The product display allows images to be displayed using the image address or image link with VARCHAR rather than using the "image" datatype.



#	Name	Type	Collation	Attributes	Null	Default
1	image_id	int(2)			Yes	NULL
2	concern_id	int(2)			Yes	NULL
3	concern_image	varchar(193)	utf8_gen		Yes	NULL

#	Name	Type	Collation	Attributes	Null	Default
1	product_id	int(10)			No	None
2	product_name	varchar(100)	latin1_swedish_ci		No	None
3	product_image	varchar(1000)	latin1_swedish_ci		No	None
4	avg_rating	int(10)			No	0
5	num_ratings	int(10)			No	0

Fig. 1: Showing database structure using varchar datatype for two tables containing images

Using VARCHAR datatype rather than image made my database structure more efficient and editable. It was also much easier to gather the image address than to reformat the image to fit the correct data type.

image_id	concern_id	concern_image
1	1	https://gladskin.com/cdn/shop/articles/2021-06-09_...
2	1	https://post.medicalnewstoday.com/wp-content/uploa...
3	1	https://us.123rf.com/450wm/maemodnit/maemodnit2112...
4	1	https://www.cerave.com/-/media/project/loreal/bran...
5	1	https://cdn2.stylecraze.com/wp-content/uploads/202...
6	2	https://c.ndtvimg.com/2022-08/ispjd1o_oily-skin-ho...
7	2	https://cdn-prod.medicalnewstoday.com/content/imag...
8	2	https://ntg-catalog.imgix.net/products/6811261/681...
9	2	https://img.buzzfeed.com/buzzfeed-static/static/20...
10	2	https://images.ctfassets.net/mdcr7mahi0vp/3UFj8VCe...
11	3	https://upload.wikimedia.org/wikipedia/commons/thu...
12	3	https://health.clevelandclinic.org/wp-content/uplo...
13	3	https://images.ctfassets.net/mdcr7mahi0vp/2DPefIFR...
14	3	https://m4b6f3p8.rocketcdn.me/app/uploads/2021/04/...
15	4	https://www.epiphanydermatology.com/wp-content/upl...

Usage of URL in database rather than Image datatype with Binary Data shown in **concern_image** and **product** tables

product_id	product_name	product_image	avg_rating	num_ratings
1	CeraVe Foaming Facial Cleanser	https://m.media-amazon.com/images/I/31j+VyDf+zL._S...	4	4
2	CeraVe Hydrating Facial Cleanser	https://www.cerave.com/-/media/project/loreal/bran...	4	4
3	CeraVe Moisturizing Cream	https://www.cerave.com/-/media/project/loreal/bran...	4	2
4	CeraVe PM Facial Moisturizing Lotion	https://www.cerave.com/-/media/project/loreal/bran...	5	1
5	CeraVe AM Facial Moisturizing Lotion with Sunscree...	https://m.media-amazon.com/images/I/41K+11aLxaL._S...	4	1
6	The Ordinary Natural Moisturizing Factors + HA	https://static.thcdn.com/images/large/original//pr...	0	0
7	The Ordinary Niacinamide 10% + Zinc 1%	https://i5.walmartimages.com/asr/a0e77f91-ad22-4f7...	5	1
8	The Ordinary Squalane Cleanser	https://theordinary.com/dw/image/v2/BFKJ_PRD/on/de...	5	1
10	The Ordinary Hyaluronic Acid 2% + B5	https://m.media-amazon.com/images/I/41EFcn3vhRL.jp...	0	0
11	The Ordinary Multi-Peptide + HA Serum	https://media.ulta.com/i/ulta/2600597?w=1020&h=102...	5	1
12	The Ordinary "Buffet" + Copper Peptides 1%	https://static.thcdn.com/images/large/original//pr...	0	0
13	The Ordinary Azelaic Acid Suspension 10%	https://media.ulta.com/i/ulta/2551154?w=1020&h=102...	0	0
14	The Ordinary Natural Moisturizing Factors + PhytoC...	https://www.sephora.com/productimages/sku/s2644169...	0	0
15	The Ordinary Multi-Peptide Eye Serum	https://media.ulta.com/i/ulta/2606218?w=1020&h=102...	0	0

Fig. 2: Showing database entries using image address in concern_image and product table

Instead of converting the file to binary data, I am able to copy and paste the link into the database instantly. Using the URL allows for a broader range of possible images because when converting images to binary data, not all formats, such as jpg, png, or webp, can be converted. Furthermore, when fetching images from the database, they are well organized and can be recalled with a simple “echo” statement.

```
while($eyes_row = mysqli_fetch_assoc($eyes_result)){
    echo"<div class = product_display>";
        echo"<form action = 'rate_comment.php'>
                <button id = 'rating_button'> <a href='rate_comment.php?product_id=" . $eyes_row['product_id'] . "'>" .
                    $eyes_row['product_name'] . "</a></button>";
        echo "</form>";
        echo "<img src=" . $eyes_row['product_image'] . " width=100px height=100px><br>";
```

Use of echo statement to call image

Image size is dictated by the *width* and *height* parameters

Fig. 3: Code showing the use of an echo statement to call a product image

II. Skincare quiz

Success Criteria: 7

I created the skincare quiz so that products can be recommended to the user based on quiz responses. Concerns like dry and oily skin are easily recognizable, but terms like blackheads and large pores may not be familiar to someone just getting into skincare. I used images of people with that concern that corresponded with the checkboxes in the quiz so that users could compare their own skin to the photo.

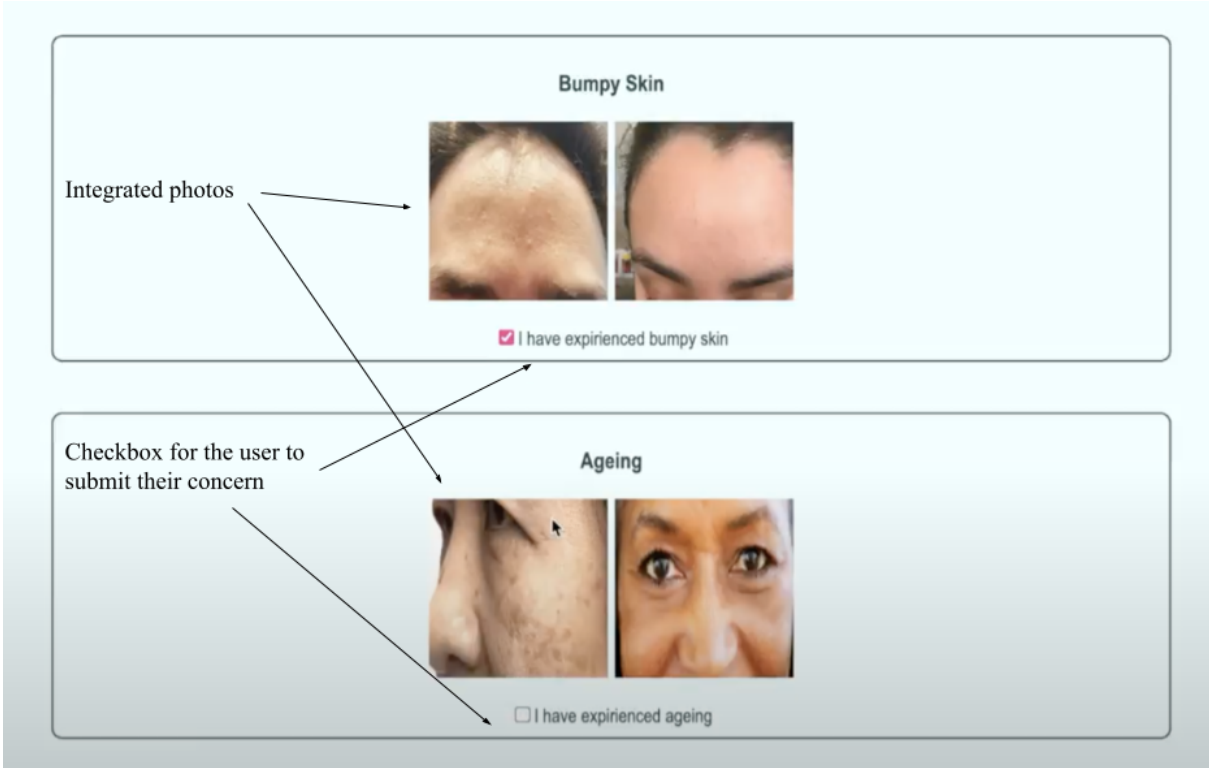


Fig. 4: Use of pictures in the skincare quiz

I used checkboxes since they allow for multiple options to be selected while allowing some options not to be selected if the user doesn't experience certain concerns.

```
//Bumpy Skin
echo"<div id = 'bumpy_skin' class = 'concern_display'>";
    $bumpy_skin_sql = "SELECT * FROM concern_image WHERE concern_id = 5";
    $bumpy_skin_result = mysqli_query($conn, $bumpy_skin_sql);
    $bumpy_skin_row = mysqli_fetch_assoc($bumpy_skin_result);

    echo"<h3>Bumpy Skin</h3>";

    while($bumpy_skin_row = mysqli_fetch_assoc($bumpy_skin_result)) {
        echo"<img src=". $bumpy_skin_row['concern_image'] ." width= 150px height=150px class = 'image'>";
    }

    echo"<br><br><input type='checkbox' id='bumpy_skin' name='bumpy_skin' value='bumpy_skin'>I have expirienced bumpy
        skin
</div>";
```

Code to fetch the concern image

Div to separate each section of the quiz for organization

Calls image from database and displays it with set width and height parameters

Checkbox input for user to check - Asks user if they have experienced the concern (Bumpy Skin)

Fig. 5: Showing code structure for the one section of the quiz (bumpy skin)

III. Showing the rating of a product using stars on the product display page

Success Criteria: 9

Having a self-explanatory, good-looking interface for product display and review was very important to my client. I used stars to show the rating of a product on the product display page since that is a common method of showing ratings on most commerce websites.

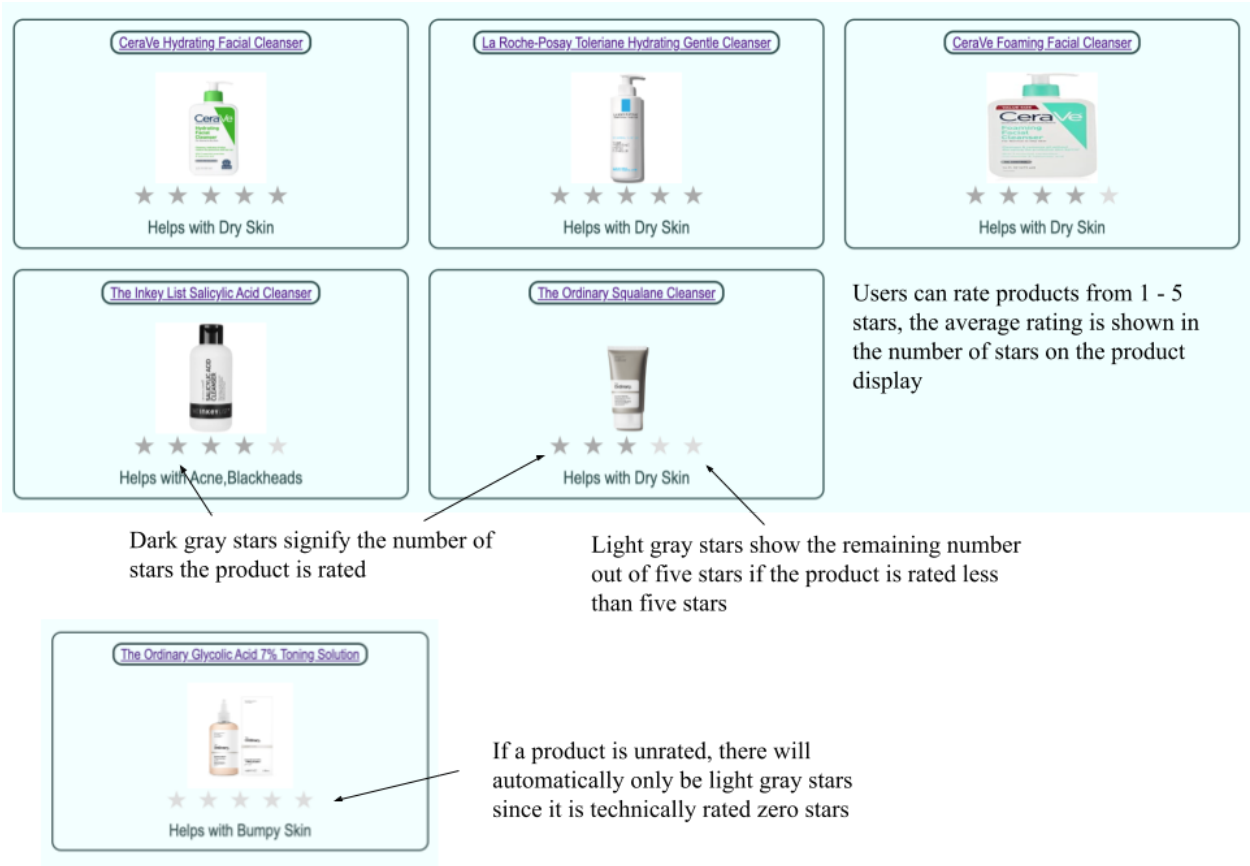


Fig. 6: Product display with stars showing product rating

In order to implement star icons, I used Scalable Vector Graphics (SVG) icons and inline CSS to display stars on the page. I used SVGs because they are the most reliable method of displaying icons without using complex CSS and eliminating additional HTML requests.

```
$all_products_sql = "SELECT product.product_id, product.product_name, product.product_image, product.avg_rating,
                     GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns FROM product_concern
                     INNER JOIN product ON product_concern.product_id = product.product_id
                     INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
                     GROUP BY product.product_id, product.product_name, product.product_image
                     ORDER BY product.avg_rating DESC;";
$all_products_result = mysqli_query($conn, $all_products_sql);
//while loop to display all products
while ($all_products_row = mysqli_fetch_assoc($all_products_result)) {
```

Step 1: **SQL statement** selects all product information including the **average rating** of the product

Step 2: **While loop** allows database information to be stored in a **row variable** based on the result

Fig. 7: Code of the star display - showing steps 1 - 2 of displaying star icons

Step 4: **For loop** loops five times since that is the total number of stars

Step 3: The number of stars a product has is based on the average rating (**rounded** to the nearest whole number) so that it can be displayed in star form

Step 5: For each iteration of the loop, the **if statement** checks how many stars should be checked or unchecked.

```
$product_num_stars = (round($all_products_row['avg_rating']));

for ($x = 1; $x <= 5; $x++) {  // Loop through a fixed number of stars (5)
    if ($x <= $product_num_stars) {
        //Checked stars used for the number / 5
        echo"<svg xmlns='http://www.w3.org/2000/svg' width='30' height='30' viewBox='0 0 100 100' class =
            'star-icon-checked'>
            <path d='M50 5.9l7.1 21.6h22.9l-18.4 13.4 7 21.5-18.4-13.3-18.4 13.3 7-21.5-18.4-13.4h22.9z'/>
        </svg>";
    } else {
        //Unchecked stars used for remaining / 5
        echo"<svg xmlns='http://www.w3.org/2000/svg' width='30' height='30' viewBox='0 0 100 100' class =
            'star-icon-unchecked'>
            <path d='M50 5.9l7.1 21.6h22.9l-18.4 13.4 7 21.5-18.4-13.3-18.4 13.3 7-21.5-18.4-13.4h22.9z'/>
        </svg>";
    }
```

Step 6: The **if statement** first checks how many stars a product has, calling the $product_num_stars variable which is pre-rounded. It then echos the amount of stars using the 'star-icon-checked' SVG.

Step 7: The remaining number (5 - the number of stars) is echoed in unchecked stars which appear as light gray

Fig. 8: Code of the star display - showing steps 3 - 7 of displaying star icons

---

IV. Display of home page and ordering products by rating

Success Criteria: 4, 5, 8, 9, 10

I designed the homepage to be easily understood and utilized by anyone. There are labels for the different elements, such as the user's concerns and skincare routine. It is important for products to be ordered by rating so that the user can see the most popular products first.

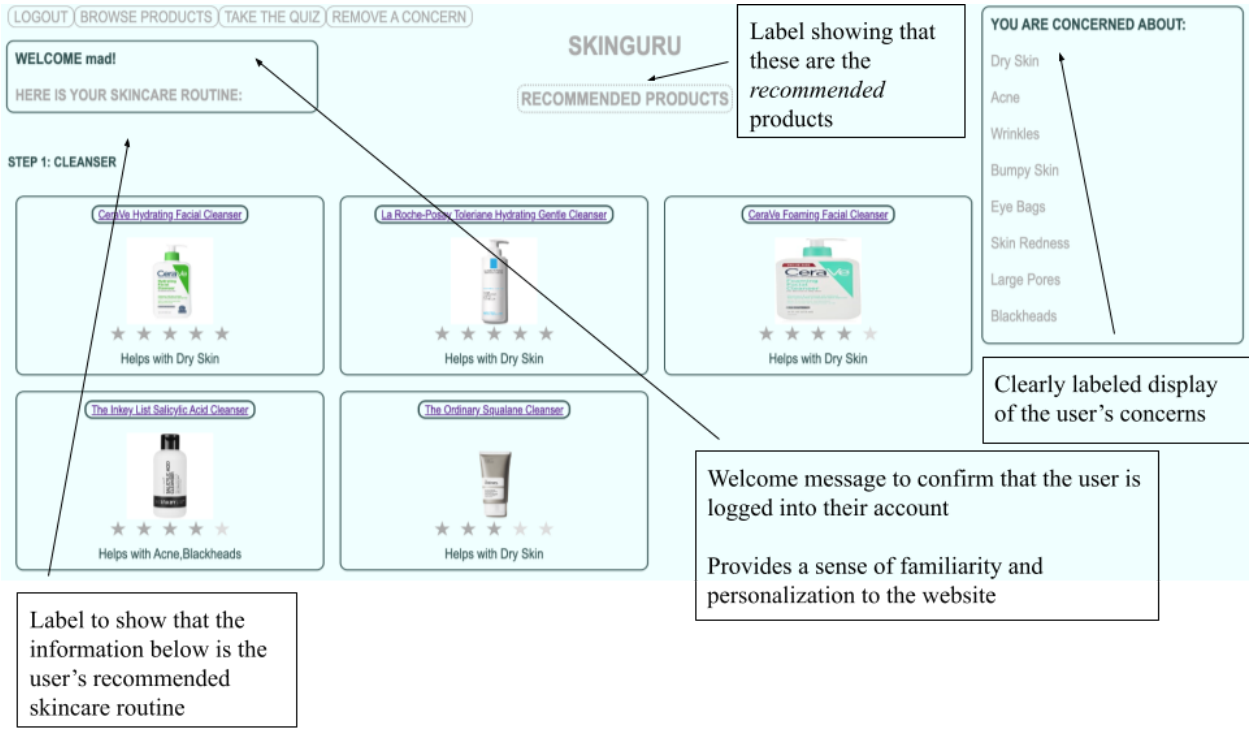


Fig. 9: Design considerations of the home page

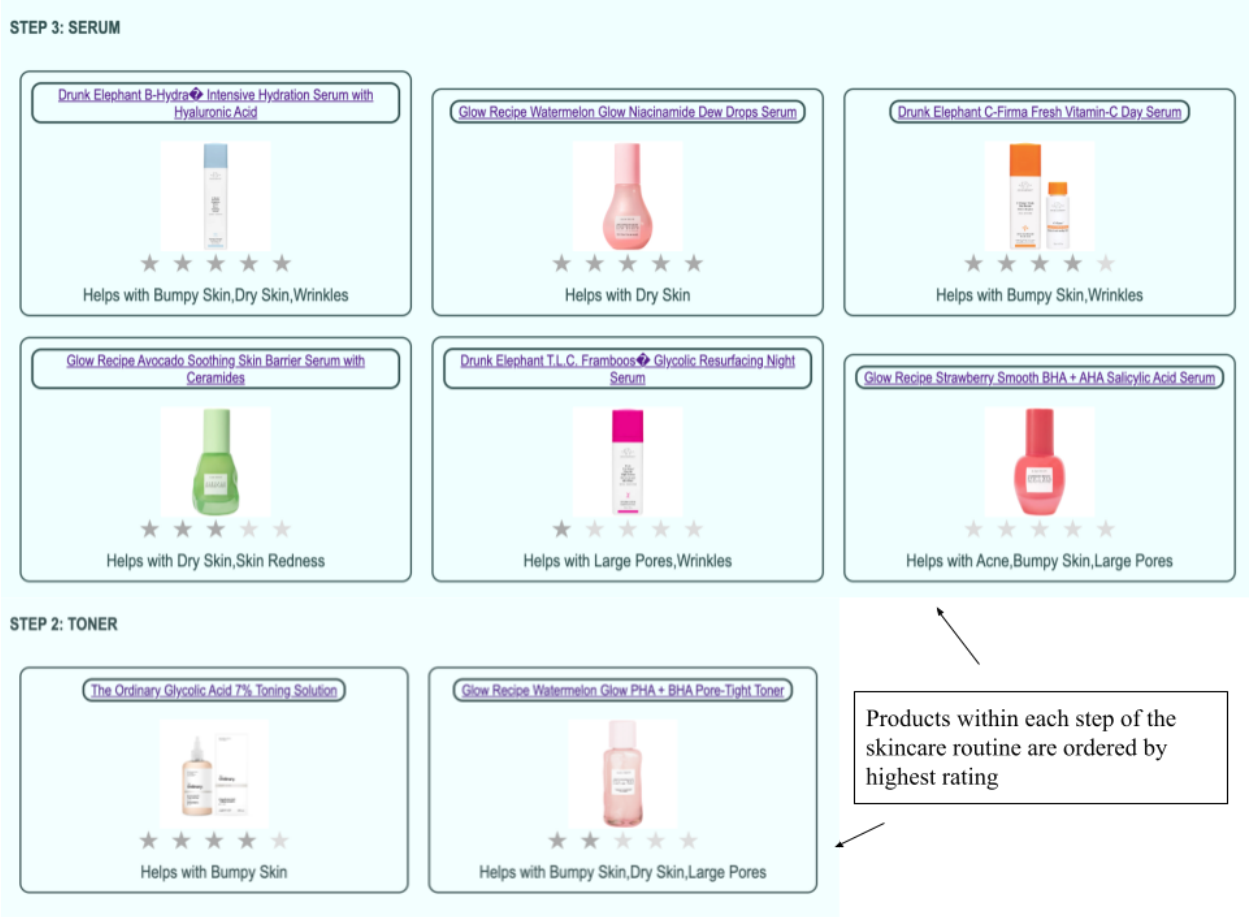


Fig. 10: Showing how products are displayed by highest rating within the skincare routine

I also displayed products in order of highest rating throughout the rest of the website. For example, on the 'Browse Products' page, all products are listed and ordered by highest rating.

```
$all_products_sql = "SELECT product.product_id, product.product_name, product.product_image, product.avg_rating,
                     GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns FROM product_concern
                     INNER JOIN product ON product_concern.product_id = product.product_id
                     INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
                     GROUP BY product.product_id, product.product_name, product.product_image
                     ORDER BY product.avg_rating DESC;";
$all_products_result = mysqli_query($conn, $all_products_sql);
//while loop to display all products
while ($all_products_row = mysqli_fetch_assoc($all_products_result)) {
```

Step 1: Tables and columns are selected for **INNER JOIN** and joined together

Step 2: Tables are **grouped by** product_id, product_name, and product_image

Step 3: Use of **ORDER BY** query to order products based on average rating in descending order

Fig. 11: Example of product ordering by average rating from the browse products page

---

V. Making the rating functionality work with stars as buttons

Success Criteria: 13

I used clickable stars (checkboxes) instead of a dropdown so that it is self-explanatory and fun for the user to click the buttons.

Fig. 12: Showing rating format in the website

Coding the rating submission was simple; however, I used complex CSS to make the checkboxes look like stars. The 'rate_comment.php' page changes based on the product that is selected. To do this, I passed the product_id between the page where the product is being displayed and the rate_comment page.

```php
echo"<div class = product_display>";
        //takes user to rate comment form if clicked
        echo"<form action = 'rate_comment.php'>
                <button id = 'rating_button'> <a href='rate_comment.php?product_id=" . $all_products_row['product_id'] .
                        "'>" . $all_products_row['product_name'] . "</a></button>";
        echo "</form>";
```

The link on the button is structured as a form with the **action** being 'rate_comment.php' - *allows the information to be passed between pages*

The button within the form contains a link to 'rate_comment.php'

The '?' passes the product_id variable which is taken by selecting the product_id from that specific row in the main SQL **select** query

```php
if (isset($_GET['product_id'])) {
    $product_id = $_GET['product_id'];
```

rate_comment.php:
product_id variable is called

made into a variable called '$product_id' so that the product id can be called in future queries

Fig. 13: Showing passing of product_id to rate_comment.php from product display

Once the product ID is identified on the rate_comment.php page, I use a query to select product information.

```php
$product_info_sql = "SELECT product.product_name, product.product_image, product.avg_rating,
                     GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns,
                     product_concern.product_id
                     FROM product_concern
                     INNER JOIN product ON product_concern.product_id = product.product_id
                     INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
                     WHERE product.product_id = '$product_id'
                     GROUP BY product.product_name, product.product_image, product_concern.product_id";
$product_info_result = mysqli_query($conn, $product_info_sql);
$product_info_row = mysqli_fetch_assoc($product_info_result);
```

SELECT statement retrieves the product name, image, average rating, and concerns that the product addresses

WHERE statement distinguishes the product and targets the query to only apply to that product

Fig. 14: Showing query to get information about a specific product

Once the product information is selected from the query, it can be applied to the rate_comment page for that specific product.

```php
echo"<div id = product_display>";
    echo "<h3 id = 'product_name'>" . $product_info_row['product_name'] . "</h3><br><br>";
    echo "<img src=" . $product_info_row['product_image'] . " width=300px height=300px id = 'product_image'><br>";
    echo "ADDRESSES:  " . $product_info_row['concerns'] . "<br><br>";
    echo "CURRENT RATING: " . $product_info_row['avg_rating'] . "/5";
echo"</div>";
```

Fig. 15: Showing the display of relevant information from the passed variables

```php
echo"<div id = rate>";
    echo"<form action = rate_comment.php method = 'get'>";
        echo "<input type='hidden' name='submit_rating' value='true'>";
        echo "<input type='hidden' name='product_id' value='$product_id'>";

            echo"<label class='star-container'>
                <input class='star' type='checkbox' name='rating[]' value='1' >
                <span class='star-label' id='star_1'>★</span>
            </label><label class='star-container'>
                <input class='star' type='checkbox' name='rating[]' value='2' >
                <span class='star-label' id='star_2'>★</span>
            </label><label class='star-container'>
                <input class='star' type='checkbox' name='rating[]' value='3' >
                <span class='star-label' id='star_3'>★</span>
            </label><label class='star-container'>
                <input class='star' type='checkbox' name='rating[]' value='4' >
                <span class='star-label' id='star_4'>★</span>
            </label><label class='star-container'>
                <input class='star' type='checkbox' name='rating[]' value='5' >
                <span class='star-label' id='star_5'>★</span>
            </label>";

        echo "<input type = 'submit' name = 'submit_rating' value = 'SUBMIT RATING' id = 'rating_submit'>";
    echo"</form>";
echo"</div>";
```

1) Rating form with **star containers (CSS)** as checkboxes

2) **Value** shows how many stars the product is being rated

Fig. 16: Steps 1 - 2 | Showing how the star checkboxes are used to submit the rating

```php
if (isset($_GET['submit_rating'])) {
    if (isset($_GET['rating']) && is_array($_GET['rating'])) {
        $checkedCount = count($_GET['rating']);
        echo "<b id = rating_confirmation>You gave this product: " . $checkedCount . "</b>";
```

3) When 'submit_rating' button is pressed, the code will check how many stars were checked

4) $checked_count variable is the number / 5 that the user rated the product

5) Getting relevant variables to make an SQL database entry

```php
    $product_id = $product_info_row['product_id'];
    $user_id = $_SESSION['user_id'];

    $rating_sql = "INSERT INTO rating (product_id, user_id, rating) VALUES ('$product_id', '$user_id', '$checkedCount')";
    $rating_result = mysqli_query($conn, $rating_sql);

    if (!$rating_result) {
        echo "Rating insertion failed: " . mysqli_error($conn);
    }

    // Update total ratings count
    $total_ratings_sql = "UPDATE product SET num_ratings = num_ratings + 1 WHERE product_id = '$product_id'";
    $total_ratings_result = mysqli_query($conn, $total_ratings_sql);
```

6) **Total ratings count** is updated so that the average rating can be calculated

Fig. 17: Steps 3 - 6 | Showing preparation and execution of SQL statements

```php
$sum_sql = "SELECT SUM(rating) AS total_sum FROM rating WHERE product_id = '$product_id'";
$sum_result = mysqli_query($conn, $sum_sql);
$sum_row = mysqli_fetch_assoc($sum_result);

$num_ratings_sql = "SELECT num_ratings FROM product WHERE product_id = '$product_id'";
$num_ratings_result = mysqli_query($conn, $num_ratings_sql);
$num_ratings_row = mysqli_fetch_assoc($num_ratings_result);

if ($num_ratings_row['num_ratings'] > 0) {
    $avg_rating = $sum_row['total_sum'] / $num_ratings_row['num_ratings'];
    $update_rating_sql = "UPDATE product SET avg_rating = '$avg_rating' WHERE product_id = '$product_id'";
    $update_rating_result = mysqli_query($conn, $update_rating_sql);
    if (!$update_rating_result) {
        echo "Updating average rating failed: " . mysqli_error($conn);
    }
} else {
    $avg_rating = 0; // To avoid division by zero
}
```

7) Calculating average rating by taking the **sum** of all ratings and dividing it by the total number of ratings

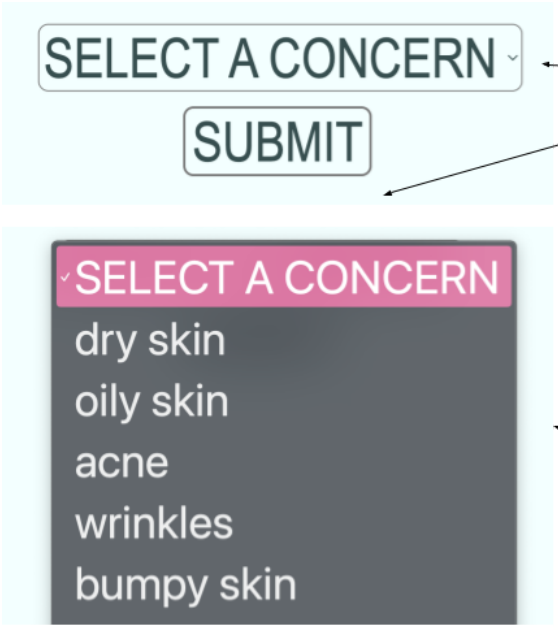
8) Rating is updated

Fig. 18: Steps 7 - 8 | Calculation and updating of rating

VI. Searching and sorting for products

Success Criteria: 14, 15, 16

Recommended products can be helpful, but users may want to search for a product themselves to check out the reviews or compare it to other products. This is why I needed to put a searching and sorting option. I will use the 'filter by concern' page as an example of sorting.



User is prompted to select a concern from the dropdown and submit it

The form appears this way on the website

```
echo" <div id = 'form'>
    <form action = 'filter_by_concern.php'>
        <select name='dynamic' id = 'select'>";
            echo "<option value='select a concern'>SELECT A CONCERN</option>";
            echo "<option value='dry skin'>dry skin</option>";
            echo "<option value='oily skin'>oily skin</option>";
            echo "<option value='acne'>acne</option>";
            echo "<option value='wrinkles'>wrinkles</option>";
            echo "<option value='bumpy skin'>bumpy skin</option>";
            echo "<option value='ageing'>ageing</option>";
            echo "<option value='eye bags'>eye bags</option>";
            echo "<option value='skin redness'>skin redness</option>";
            echo "<option value='large pores'>large pores</option>";
            echo "<option value='blackheads'>blackheads</option>";
        echo "</select>
        <input type = submit value = 'SUBMIT' name = 'concern_submit' id ='submit'>
    </form>
</div>";
```

Fig. 19: Select a concern form and code

```php
if(isset($_GET['concern_submit'])) {
    $selected_concern = $_GET['dynamic'];
```

Checks if form has been submitted, then **passes the selected concern as a variable** which is saved as $selected_concern

SQL query **selects** product information but then **filters it by concern** by saying that the product can only be selected if it addresses that specific 'selected_concern'

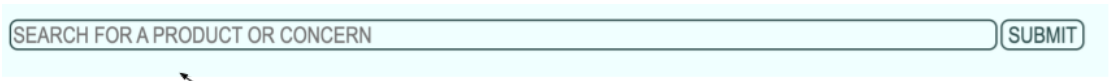
```php
$product_type_sql = "SELECT product.product_id, product.product_name, product.product_image, product.avg_rating,
                     GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns FROM product_concern
                     INNER JOIN product ON product_concern.product_id = product.product_id
                     INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
                     WHERE skin_concern.concern LIKE '%$selected_concern%'
                     GROUP BY product.product_id, product.product_name, product.product_image
                     ORDER BY product.avg_rating DESC;";
```

```php
while ($product_type_row = mysqli_fetch_assoc($product_type_result)) {
    echo"<div class = product_display>";
```

Once the query is complete, the **while loop** will **output** all the products that address that concern

Fig. 20: Code showing sorting by concern

Additionally to sorting by brand or concern, if a user knows exactly what they are looking for, they have the option to search for a product.



Form where the user can submit a search

```php
echo" <div id = 'form'>
    <form action = 'search_product.php'>
        <input type= 'search' id='search' name='search' placeholder = 'SEARCH FOR A PRODUCT OR CONCERN'>
        <input type = submit value = 'SUBMIT' name = 'search_submit' id ='submit'>
    </form>
</div>";
```

Fig. 21: Showing the search form on the website and in code

```php
if(isset($_GET['search_submit'])) {
    $search = $_GET['search'];
    echo"<div id = 'all_products'>";

        echo"<h2>RESULTS 4 " . $search . ": </h2><br>";
```

RESULTS 4 sunscreen:

Once search is submitted, the entry is saved in the '$search' variable

Echoes what the user searched for at the top of the page. For example, if the user searches 'sunscreen', it will show that they searched for sunscreen. This is helpful in case the user mistypes their search.

```php
$search_sql = "SELECT product.product_id, product.product_name, product.product_image, product.avg_rating,
                GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns FROM product_concern
                INNER JOIN product ON product_concern.product_id = product.product_id
                INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
                WHERE product.product_name LIKE '%$search%' OR skin_concern.concern LIKE '%$search%'
                GROUP BY product.product_id, product.product_name, product.product_image
                ORDER BY product.avg_rating DESC;";
```

SQL statement only selects products with names or concerns matching what the user searched for.

I used 'LIKE' so that if the search was incompletely typed, the query would still select products or concerns with the same letters instead of showing no results.

Fig. 22: Showing how the search query works

VII. Product recommendations based on quiz responses

Success Criteria: 8, 10

The skincare quiz is a good way to collect information about a person's skin in order to recommend products. As seen previously, the user selects concerns that they experience and then the website displays these concerns on the home page. Products recommended to the user will show up in the skincare routine and these products will selectively address the concerns that the user submitted.

Here is an example of a user who only submitted two concerns, dry skin and acne:

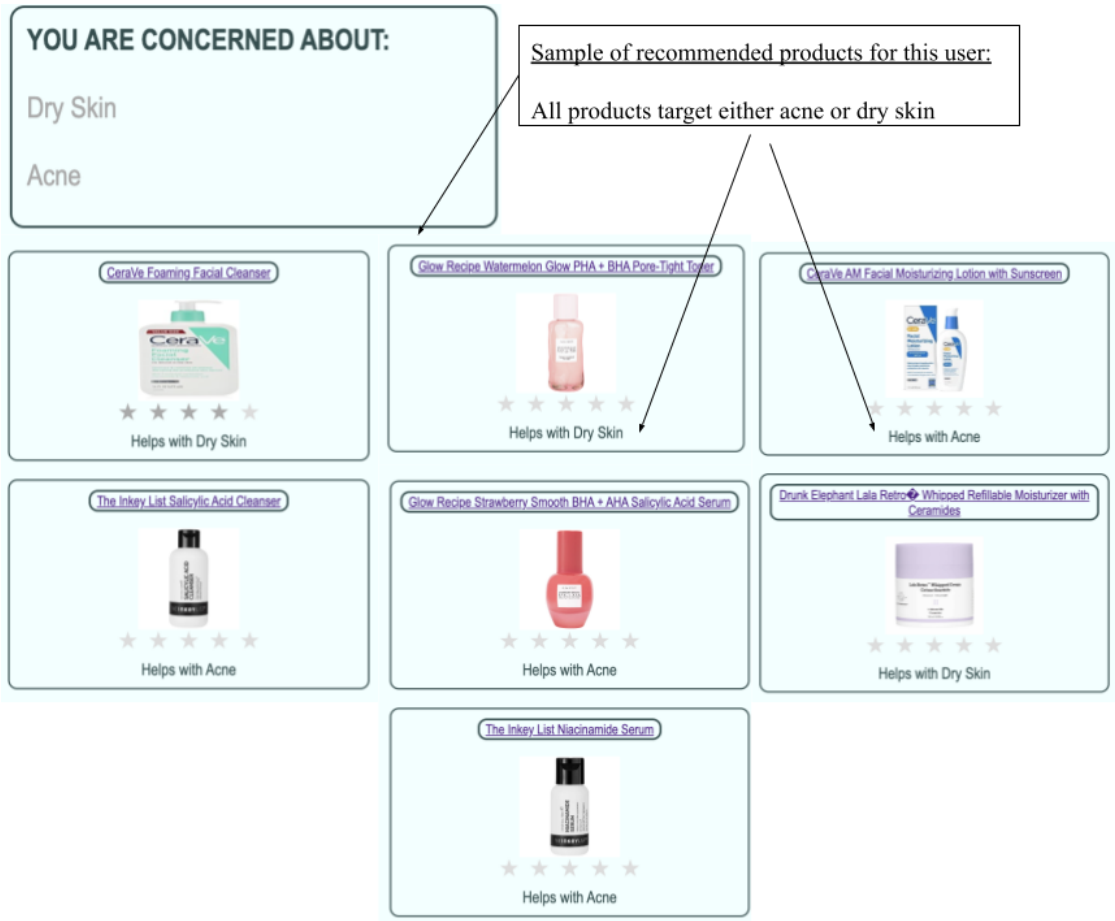


Fig. 23: Showing how products are only recommended based on the user's concerns

Example of SELECT query (toner):

- **GROUP_CONCAT** of distinct user concerns ensures each concern is only selected once.
- Once the concerns are grouped, '**INNER JOIN user_concern ON product_concern**' joins the user and product concerns to ensure that the products selected only contain those specific concerns.
- Once the user and product concerns are selected, '**INNER JOIN product ON product_concern**' actually selects the products with these concerns
- Finally, '**INNER JOIN skin_concern ON product_concern**' connects the concern IDs with the concern names, allowing them to be called in future queries

```
$toner_sql = "SELECT product.product_name, product.product_image,
        GROUP_CONCAT(DISTINCT user_concern.concern_id) AS concern_ids,
        GROUP_CONCAT(DISTINCT skin_concern.concern) AS concerns, product_concern.product_id
        FROM product_concern
        INNER JOIN user_concern ON product_concern.concern_id = user_concern.concern_id
        INNER JOIN product ON product_concern.product_id = product.product_id
        INNER JOIN skin_concern ON product_concern.concern_id = skin_concern.concern_id
        WHERE product.product_name LIKE '%ton%' AND user_concern.user_id = '$user_id'
        GROUP BY product.product_name, product.product_image, product_concern.product_id
```

Fig. 24: Showing how the query selects products based on user concerns

VIII. Remove concern functionality

Success Criteria: 6

In order to make it easy for users to remove concerns, I ensured that only their current selected concerns would appear on the website's ‘remove concern’ page.

I will use the previous example of the user who only selected dry skin and acne.

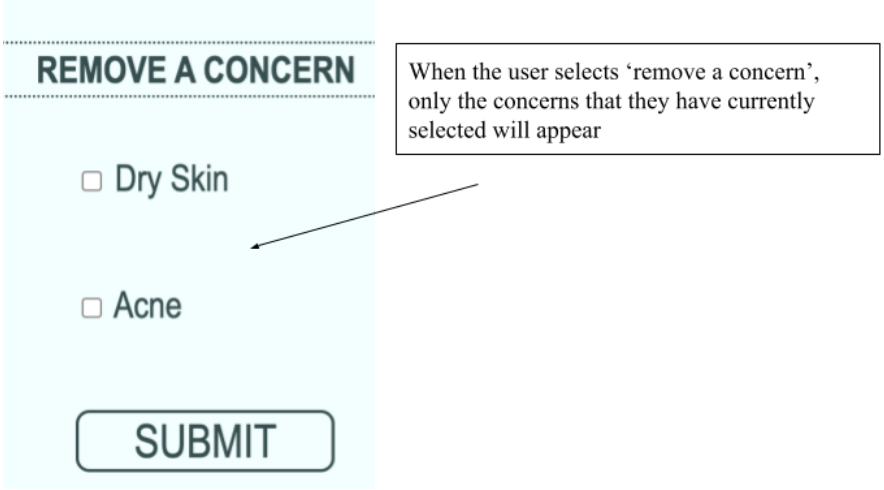


Fig. 25: Showing that only the user's concerns will appear when they are trying to remove a concern

```
echo"<form action = 'remove_concern.php' method = 'get' id = 'remove_concern'>";
    $user_id = $_SESSION['user_id'];
    $select_concern_sql = "SELECT user_concern.user_id, skin_concern.concern, skin_concern.concern_id FROM user_concern
        INNER JOIN skin_concern ON user_concern.concern_id = skin_concern.concern_id WHERE user_concern.user_id =
        '$user_id'";
```

Within the form, there is an SQL query selecting only the concerns that the user has

INNER JOIN skin_concern ON user_concern joins the user's concern IDs with the corresponding concern names so that they can be called in future queries

Once the form is submitted, the IF statement will activate the other IF statements.

Each concern has an ID and each nested IF statement checks if a specific concern ID 'isset', then a query will delete that specific concern from the 'user_concern' table

```
if(isset($_GET['remove_concern_submit'])) {

    if (isset($_GET['1'])) {
        $user_id = $_SESSION['user_id'];
        $remove_concern_dry_skin_sql = "DELETE FROM user_concern WHERE user_id = $user_id AND concern_id = '1'";
        if (mysqli_query($conn, $remove_concern_dry_skin_sql)){
            header("refresh: 1; url = 'home.php'");
        } else {
            echo "nope";
        }
    }
```

Fig. 26: Showing remove concern code

Word count: 691

Works Cited

"How to Create a Simple Star Rating with CSS." *Www.w3schools.com*, www.w3schools.com/howto/howto_css_star_rating.asp.

Stack Overflow. "Stack Overflow - Where Developers Learn, Share, & Build Careers." *Stack Overflow*, 2022, stackoverflow.com/.

"Star Rating Using HTML CSS and JavaScript." *GeeksforGeeks*, 22 Sept. 2023, www.geeksforgeeks.org/star-rating-using-html-css-and-javascript/. Accessed 6 Feb. 2024.

W3Schools. "HTML Links." *W3schools.com*, 2019, www.w3schools.com/html/html_links.asp.

w3schools. "SQL INNER JOIN Keyword." *W3schools.com*, 2019, www.w3schools.com/sql/sql_join_inner.asp.