IAT352 PA3 Project Report

RewardX

Leo Ruan

301298088

Functionalities

Visitors have access to the login page, register page, home page, content unit page (named partners) and rewards page. Visitors will not have access to the interface to post reviews and adding favourite stores.

Members have access user's own dashboard. The dashboard displays the profit picture retrieved from http://avatars.adorable.io/, their username, a QR code (right now it's just a placeholder. As this function is not required, I will implement it in my own time as my personal project), their points left and a button to log out. Users can also use the star icon next to the partnered stores to save their favourite stores, they have access to the interface to post a new review and the points are given to the store will be reflected in realtime to the store's total satisfaction score. Users have access to all their favourite stores added by going into the dashboard page and there is a fast link to the store detail page from their dashboard.

All the stores in the partner's page have their hours displayed with the open status dynamically. There is code handling the open status of each store.

The rewards page is visible to all users but only members can redeem products if they have enough points. If a user has enough points, he/she will be redirected to a page to enter their shipping information and I plan to save shipping information in the database (Order table) for the future implementation of admin to access and handle orders manually.

User Guide

The Partners page contains the content units, not the Rewards page.

For Access to points, use the following user or add points manually from PHPMyAdmin -> LEO_RUAN -> Users -> Look for the account registered -> StorePoints Column -> Modify the points: (For testing the functionalities of redeeming rewards As right now there is no way of adding reward points.)

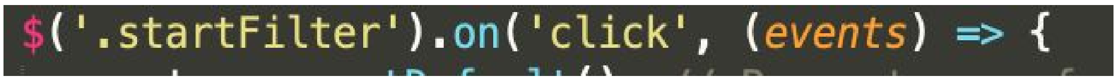
Email: test@test.com

Password: test

Added Logics and Functionalities for PA3

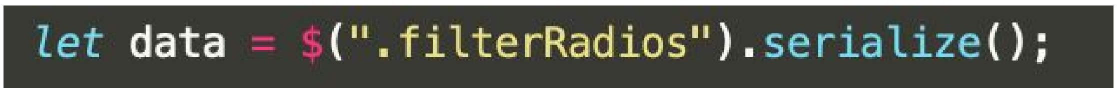
Filter AJAX

1. Filter AJAX is built on the original filter page, I added new JavaScript (JQuery) to handle button click. This click event will fire every time the button with class named startFilter is called. I pass in the events to the callback so that I can use preventDefault to prevent page from reloading. :



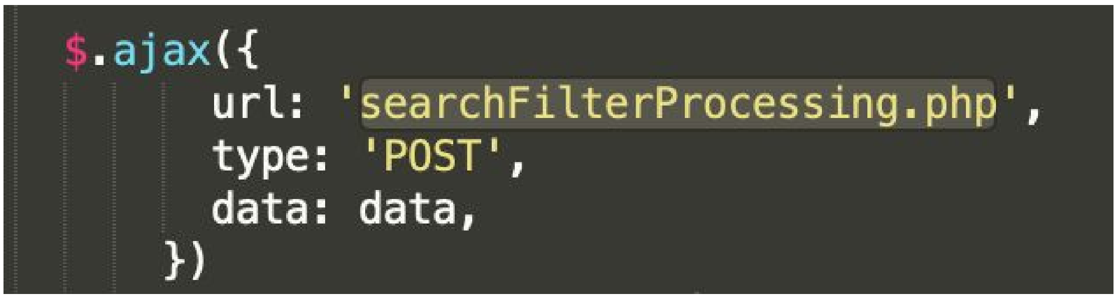
```
$('.startFilter').on('click', (events) => {
```

2. After the event is fired, I simply use the JQuery build-in function serialize() on the form as the filter is within a form tag to get all the information and save it to a variable data.



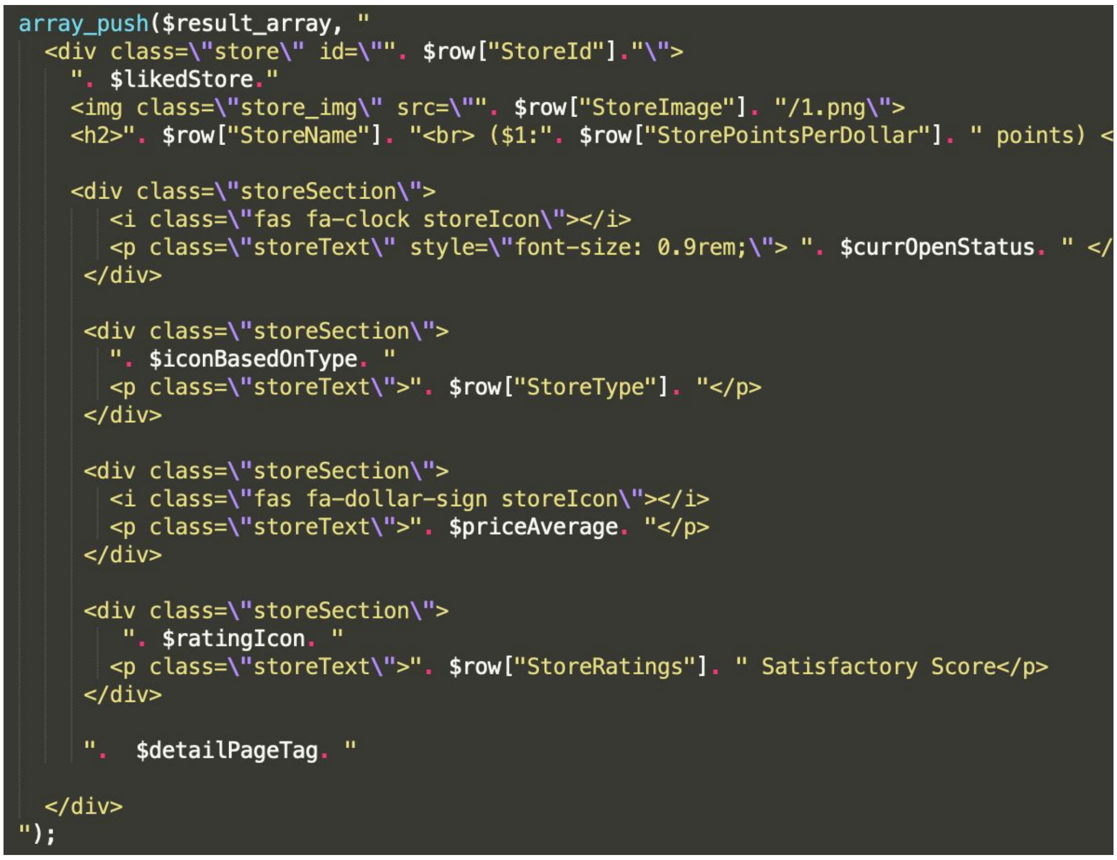
```
let data = $(".filterRadios").serialize();
```

3. Then, using JQuery ajax function to pass the sterilized data to searchFilterProcessing.php for processing with post method.



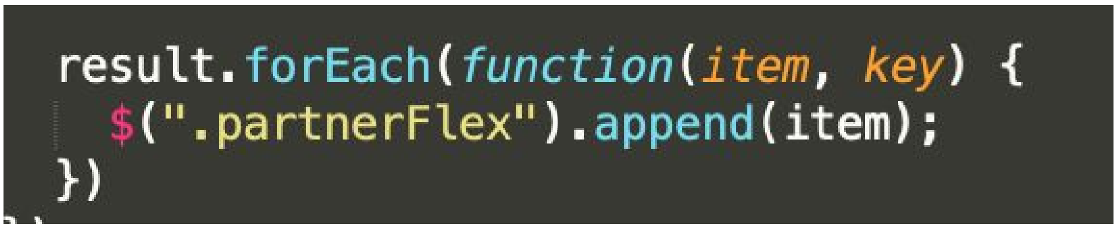
```
$.ajax({
        url: 'searchFilterProcessing.php',
        type: 'POST',
        data: data,
    })
```

4. In searchFilterProcessing.php I build the query like how I did in PA2 (See below). But instead of echoing the html elements directly, to avoid doing the same process again in JavaScript when the data returns to the front-end, I created an array and push in the html structures as a string so that in frontend JavaScript I can just process the array.



```
array_push($result_array, "
  <div class=\"store\" id=\"". $row["StoreId"]."\">
    ". $likedStore."
    <img class=\"store_img\" src=\"". $row["StoreImage"]. "/1.png\">
    <h2>". $row["StoreName"]. "<br> ($1:". $row["StorePointsPerDollar"]. " points) <

    <div class=\"storeSection\">
      <i class=\"fas fa-clock storeIcon\"></i>
      <p class=\"storeText\" style=\"font-size: 0.9rem;\"> ". $currOpenStatus. " </
    </div>

    <div class=\"storeSection\">
      ". $iconBasedOnType. "
      <p class=\"storeText\">". $row["StoreType"]. "</p>
    </div>

    <div class=\"storeSection\">
      <i class=\"fas fa-dollar-sign storeIcon\"></i>
      <p class=\"storeText\">". $priceAverage. "</p>
    </div>

    <div class=\"storeSection\">
       ". $ratingIcon. "
      <p class=\"storeText\">". $row["StoreRatings"]. " Satisfactory Score</p>
    </div>

    ".  $detailPageTag. "

  </div>
");
```

5. After building out the the array, I will use json_encode() function to send back the array to the frontend.
6. In the frontend, I first clear all the stores currently in the container by using the empty function, then I parse the data passed back from php, use a for loop to loop through the array and append the item to the container.

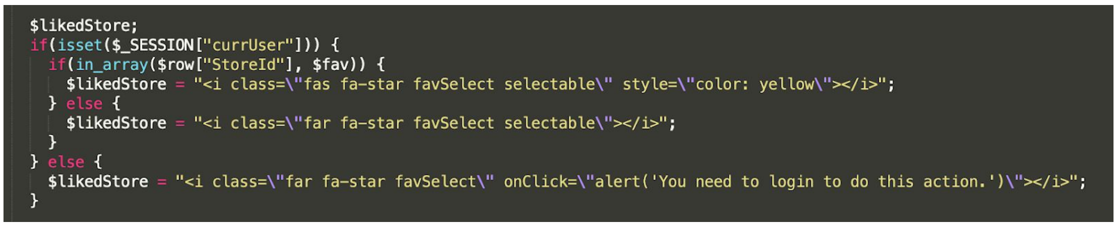


```
result.forEach(function(item, key) {
  $(".partnerFlex").append(item);
})
```

7. (The searching follows the same logics besides the fact that the query building is a bit different.)

Mark as Favorite AJAX

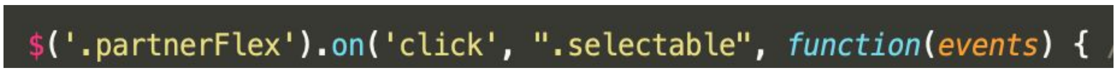
1. I gave the icon tag that allows user to save content unit an extra class called selectable, so when the page first loaded, I will detect if there is a user currently logged in, if there is, I will give the a tag the selectable class, otherwise a onclick event will be bind with the icon tag that alerts whenever it is clicked.



```
$likedStore;
if(isset($_SESSION["currUser"])) {
  if(in_array($row["StoreId"], $fav)) {
    $likedStore = "<i class=\"fas fa-star favSelect selectable\" style=\"color: yellow\"></i>";
  } else {
    $likedStore = "<i class=\"far fa-star favSelect selectable\"></i>";
  }
} else {
  $likedStore = "<i class=\"far fa-star favSelect\" onClick=\"alert('You need to login to do this action.')\"></i>";
}
```

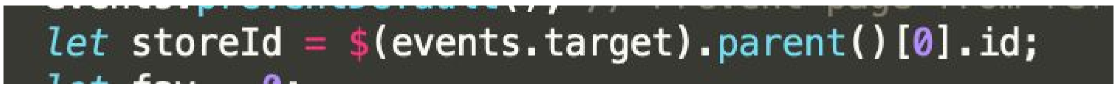
2. So in the JS file, Detect click event as well. This time the click event is not bind to the icon tag however, the tag is being changed dynamically, so the click event is not going to be attached to the newly created icon tags. So I use the jquery on method to bind events on the content unit container. (The reason

why is not being bound to the store content unit card is that the content unit card can be dynamically removed and added as well because of filter).

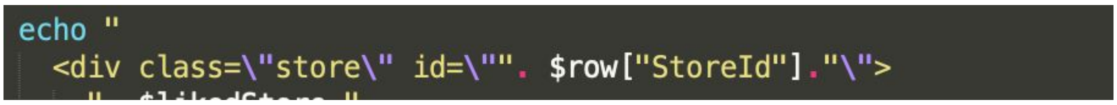


```
$('.partnerFlex').on('click', ".selectable", function(events) {
```

3. Then I get the id of the store where the click event happens by access the events.target created by JQuery, the id of the card is attached to the parent of the icon tag, so using the parent method provided by JQuery and I can call .id to retrieve the only id attached to the tag.

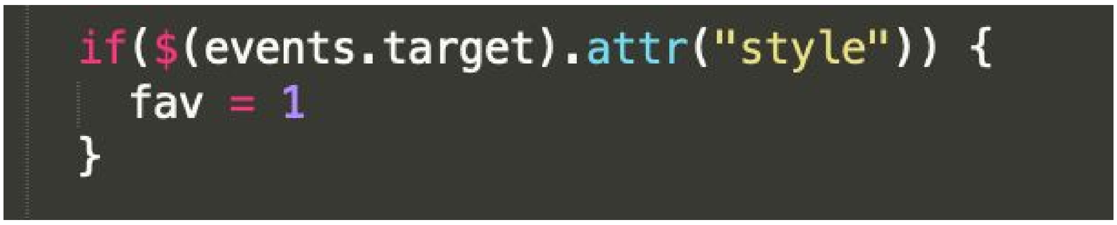


```
let storeId = $(events.target).parent()[0].id;
```



```
echo "
  <div class=\"store\" id=\"". $row["StoreId"]."\">
```

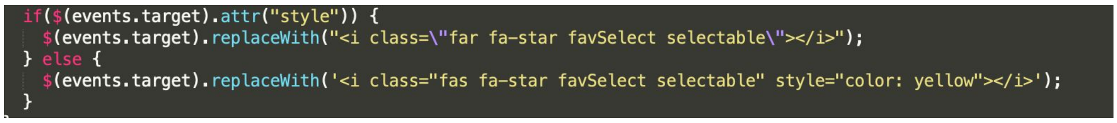
4. Then I check if there is a style attribute attached to the icon tag to detect whether the store is already liked or not and set this information to a variable.



```
if($(events.target).attr("style")) {
  fav = 1
}
```

5. After getting the information on step 4, I created a JavaScript Object with the store id and favorite status, send this data via ajax.
6. The php processing is similar to what is done in PA2 (See below), after the query is run, I detect if there is any error, if not I will echo 1 else 0.
7. Back in the frontend JavaScript, I check if the query failed, if it does, I will alert accordingly. Otherwise the query succeed, I use replaceWith function in

JQuery to display different icon tags and styles if needed.

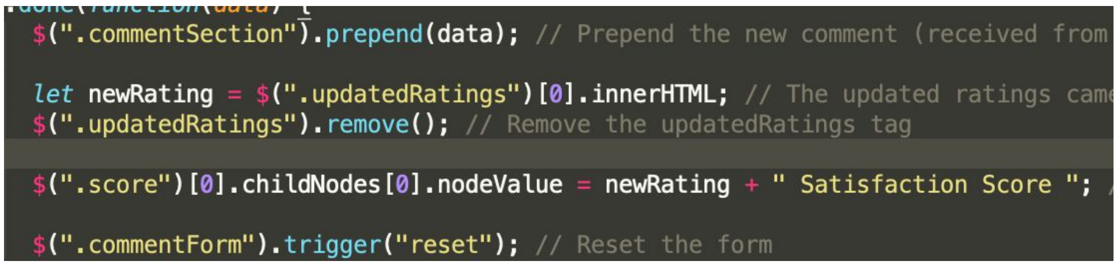


```
if($(events.target).attr("style")) {
  $(events.target).replaceWith("<i class=\"far fa-star favSelect selectable\"></i>");
} else {
  $(events.target).replaceWith('<i class="fas fa-star favSelect selectable" style="color: yellow"></i>');
}
```

8. Same logic is used for saving as favorite from detailed content unit view page.

Updating Comments

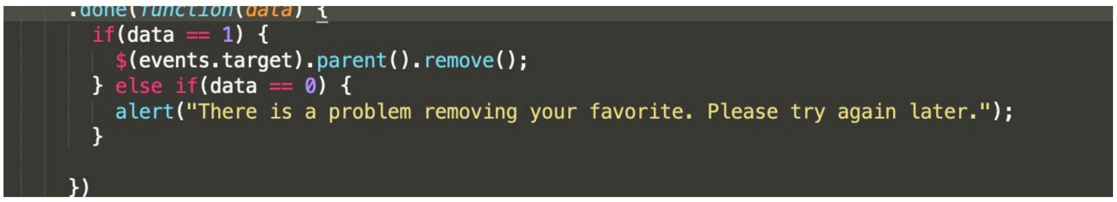
1. A JS function will be attached to the form's submit button. An id will be added to the icon tag for the references of the store id.
2. In the js function after the click event is triggered, preventDefault() will be called. Then serialize the data, reformatted the serialized data with the id of the store
3. Do the AJAX call. Then update the comment database in the other php page (Same logics as below). Then echo the string of the HTML with a paragraph tag containing the updated rating.
4. In the js file for the front-end, prepend the data to the comment, retrieve the new updated total rating from the p tag and remove the p tag. Next update the score, by accessing nodeValue of childNode because the score is contained in a span tag. Lastly use the trigger function with reset.



```
.done(function(data) {
  $(".commentSection").prepend(data); // Prepend the new comment (received from

  let newRating = $(".updatedRatings")[0].innerHTML; // The updated ratings came
  $(".updatedRatings").remove(); // Remove the updatedRatings tag

  $(".score")[0].childNodes[0].nodeValue = newRating + " Satisfaction Score "; /

  $(".commentForm").trigger("reset"); // Reset the form
```

Updating Dashboard

1. I attached event to all the remove buttons on the page. The button will have an id attached with the id of the store when constructing favorites.
2. When a remove button is clicked, I retrieve the id of the store by accessing events.target's id attribute.
3. Then I created a data variable with the removedId retrieved from events.target then send the data through the AJAX call.
4. The backend process is the same as PA3 (See Below). If the deleting query worked, the result will be returned.
5. In the frontend JavaScript I check if the returning result is true or not, if it is true, I remove the $(event.target).parent() by using the remove function provided by JQuery. Else I sent an alert saying there is an error.



```
.done(function(data) {
  if(data == 1) {
    $(events.target).parent().remove();
  } else if(data == 0) {
    alert("There is a problem removing your favorite. Please try again later.");
  }

})
```

User Logics

I started by creating the User table in the database. The columns I needed for the user at this stage of the project are 1. An auto-increment user-id. 2. User's name. 3. User's email address. 4. User's password (hashed). 5. User's favourite stores' id. 6.(For a bonus feature, not implemented yet) Is this user an admin user. 7. The points the user currently has. 8.(For a bonus feature, not implemented yet) Is this user a merchandise owner.

I also created a separate PHP file for saving constant variables to initialize the DB connection.

User Register

1. Building on the login form created from the last assignment, I changed from saving the password, username and email to the database. I started with calling the session_start() function as after registering the session will be set.
2. Then I require the constant variables I saved in the other file to use mysqli_connect to establish a connection to the database.
3. After the connection is established, I first check if the user entered email address already exists in the database. To do that, I use the query SELECT * FROM Users WHERE UserEmail = entered email. After the results have been retrieved I check the length of the results by using the function mysqli_num_rows. If this function returns more than one, that means the same email already exists in the database and I will redirect back to the register page with a URL parameter of the reasons for failure in a form as a code.
4. Else if the email is unique, I use the INSERT INTO query to pass in necessary information including a hash of the password by using this function: hash('sha512', the password entered). After insertion, I check if the result is true if it is, I use the SELECT statement pass in the email entered to retrieve to the id of the user then set the “currUser” session to the id of this user. Free the results then redirect them to the dashboard page.

User Login

1. After user input, I get the user input password and email then use the query SELECT * FROM Users Where UserEmail = “emailEntered” and

UserPassword = hash('sha512', “passwordEntered”). A better way to use hash would be using password_hash() when registering and password_verify() to check if the password can be verified against the entered password. To do this the process password is more secure but it will require more time to complete the task.

2. If the query returns a result, I will set the session of currUser to the user id, free the results and redirect the user to the dashboard page.
3. If the query returns nothing, I will redirect to Login.php with a URL parameter failed = 1 and in the login page, I will use a conditional statement to check if there is a URL parameter being passed to the page by checking $_GET[“failed”].

Stores (Content Unit)

I also have a table created manually from MyPHPAdmin, the table contains an auto-increment id, store name, store address, the relative path of store images, store hours formatted for reading, store ratings, store points per dollar, stores price average (-1 for stores that the average based on the users), links to the store, types of the store and the store description.

Displaying all stores and filtered stores

1. I check if there are any $_POST contents if not, I construct the query SELECT * FROM Store.

2. Else if the $_POST contains the input named “nameSearch”, that means the user used the name search function and I will add “WHERE StoreName LIKE '%$_POST["nameSearch"]%;” to the basic query “SELECT * FROM Store”.
3. If $_POST does not contain “nameSearch” and it contains more than 1 contents, I will check if a checkbox is set by using the isset() function and build the query dynamically. For instance, if the checkbox “food” is checked, I will add “StoreType =Shop OR " to the basic query. After all the checkboxes are checked, I will use the substr function to remove the last few characters added by the checkbox (the OR in the example above). Then I will check if the open checkbox is selected and if it is set the checkOpen to true else it defaults to false
4. After building the query, I first set up the timezone and the current day and time by using the build-in function date(). Then I use a while loop to retrieve all the rows and iterate through them.
5. The first thing to do in the while loop is to check the opening status. I get the hours in the database and break them by a space into an array by using the function explode(). Then I use the day variable defined in step 4 as the condition for my switch statement to switch between days (Monday to Sunday). In each case of the day, I read the start time and end time from the hour’s in the relative index (0 is Sunday, 1 is Monday, 2 is Tuesday etc.) array and the time will be something like 10:00am_10:00pm, so I use explode one more time to break this time into two starting time and ending time. I then check if the time of the store is closed, if it is, I save the word Closed to a variable later to use displaying the time. If it is not closed, I reformate the start

time and end time by substr so that they are readable by build-in strtotime function. After reformatting them, I compare them with the current time defined in step 4 and set the variable for displaying later to either open or close. I also declared a variable called openNow and if it is open I set the openNow to true.

6. Then I check if the price average is -1 or not if it is, I save “Based on individuals” to a string variable.
7. Next, I check if the open checkbox is checked if it is, I check if the openNow variable (defined in step 5) is false, I use continue statement to move to the next iteration of the while loop and skip the displaying of the store.
8. Then I construct the anchor tag for moving to the detailed page. I am passing two URL parameters to the href, first is the store Id, second is the open status to avoid running the time check again in the storeDetail page.
9. Then I check if the store id exists in the user’s favourite store list (retrieved and saved to an array at the start of the page by checking session and select from the database with the same id), if it is, I will display the star with yellow if not regular star will be displayed. There is also an anchor tag created based on if the store is liked or not if its liked, the URL parameter I pass will be liked = 1 else the liked = 0.
10. Then there are two condition statements to display icons dynamically based on the ratings of the store and the type of the store.
11. Finally, I use echo to display the stored information in a div with the $row and texts I constructed from the steps above.

12. If the query returns 0 results, there will be a paragraph tag saying there is no matching records found.

User Adding Favoritouts

1. The icon of the star on each store will redirect user to editFav page if a member clicks on it.
2. In the editFav page, I will retrieve the user information by their id in the session, then I process the user favourites from the database (it is a type of text) by using explode.
3. Then I check from the URL parameter if liked is 1 or 0
4. If the store is liked, I use the unset function to remove the favourite id passed from the URL parameter then use join function to turn them back to a string separated by a comma. Then I use the UPDATE clause to SET the fav Id with the user. And I check if the page is accessed from the detail page (there will be a URL parameter) it is from the detail page, I redirect back to the detail page otherwise I redirect to the partners page.
5. If the store is not liked, the same logic will be used to update the database.

Accessing store details

1. As mentioned when creating all stores, the anchor tag will pass in the URL parameters with the store id and the open status of the store.
2. I use the query to select the store with the store id from the URL parameter retrieved by $_GET['id'].

3. I reformat the time by adding days of the week in front of each time and use switch statements based on the open from the URL parameter to set open or close.
4. Then I display all the required information by getting the row's column name and echo.

Comment

The comment table contains an auto-increment id, a user name, a user rating, a user comment, the store id the comment belongs to and the user id who posted the comment.

1. If there is no comment, the length of the result retrieved from the query string SELECT * FROM Comments WHERE BelongStore = the store id retrieved from the URL parameter will be 0 then I will use a paragraph tag to display that there is no comment for the store.
2. If there are comments, the length of the result from the same query will be more than 0. Then a while loop will be used with mysqli_fetch_assoc() to display comments dynamically by echo
3. If there is no currUser in the session, there will be no inputs for users to enter reviews.
4. If there is a currUser, there will be two input boxes, one more the ratings (range from 0 to 10, scores over 10 or less than 0 will be changed from 0 or 10 from the code by using an if statement). A text area for entering detailed comments. After a form is submitted, a POST request is sent to another page newComment.php.

5. In newComment.php, I first get the current user and run the query to get the user name of currUser. SELECT UserName from Users WHERE UserId = currUser id. Then I set the user name to a variable.
6. Next, I ran the INSERT INTO query into the Comments table with the user id(from the session), store id (from URL parameter), ratings and comments (from POST) and user name from DB.
7. After insertion, I will retrieve all the comments that are posted to the store id then calculate the count of the stores (by COUNT(*)) and the total rate scores of all the comments (by running a while loop). Then I did the calculation by dividing total scores by the count of total comment count. The result will be the new rating scores for the store.
8. Last I use the UPDATE clause to update the Store table's rating and redirect back to detailStore.php.

Rewards

The reward section is **not content units.** Rewards are added to be part of the functional project.

Displaying Rewards

Ideally, rewards will be added by an admin, but right now all the rewards are added in the MyPHPAdmin interface. In the reward table, the columns are an auto-increment id, the name of the reward, the description of the reward item, the image sources of the reward item, the points required to redeem item and the quality left.

1. I start by running the query SELECT * FROM Rewards WHERE RewardQuatity > 0, so that any items that do not have any inventories, will not be displayed.
2. Use the while loop to display the rewards dynamically. Each item contains an anchor tag with a href to a different PHP page (redeemReward.php) to handle the reward redeem. There is also a URL parameter passed to the redeemReward.php page with the reward item's id.
3. In redeemReward.php, I check if there are enough points for redeeming the item, if not a URL parameter of fail will be passed back to the rewards page. Otherwise, a form will display getting the user's shipping information.
4. After shipping information has been entered, the form will redirect to shippConfirm.php, I will update the reward's quantity left as well as the points the users have. Then I will display a message saying that an email will be received once the order is processed. (The process will be set by admin in milestone 3 as well as the order table)