School of Media Arts and Technology

BSc Software Engineering

Matthew Dear

Q10232711

Points of Interest Website Design and Analysis

Assignment 2

Developing for the Internet

Tutor: Nick Whitelegg

28th April 2020

Task A

The following pages/scripts will be needed: login.php, index.php, functions.php, addPOIForm.php, and addPOI.php. Login.php will communicate with addPOI.php using the gatekeeper session variable and addPoiForm.php will communicate with addPOI.php using a POST request. A POST request has been used in this situation because the database is being updated.

functions.php

This script will contain functions that can be called by any page to reduce code repetition. The `title` function will take one argument user type, this will be a 1 if the user is an admin otherwise it will be a 0. It will output the logo and title before calling the links function and passing the user type as an argument. `Links` will use an if statement to check if the input user type is equal to 1. If it is, an associative array containing the admin links will be created and a foreach loop will iterate though it outputting them to the page. If not, the same process will be followed with non admin links. The `backButton` function will output a button to the page that uses the JavaScript history function to return you to the previous page. The `databaseConnection` function will create a PDO object for the database, set the error modes for better error reporting, and return it to the page. The final function `footer` will be used to echo out the website's footer.

login.php

Will start with the `session_start` function that will either create a new session or resume an existing one. It then uses the `include` function to link it to the functions and userDAO scripts to allow their functionality to be used. It will then call the `title` function passing the isadmin session variable if set, otherwise it will pass in 0 by default. Then an if statement is used to check the gatekeeper session variable has been set, if set the users name will be output to the page. The `backButton` function is then called completing the header section of the page. A try catch block is then opened and the `databaseConnection` function is called returning a PDO object that will be stored in the conn variable. Then the username and password sent by the login page using POST will be stored in the un and pw variables. An if statement is then used to validate the un and pw variables, using regular expressions to defend against cross site scripting attacks. If invalid an error will be output to the page. If valid then a usersDAO is created passing the conn and table name as arguments, the retuned usersDAO is then stored in the variable usersDAO. Then the `verifyLogin` function is called on it passing un and pw as arguments and the returned usersDTO will be stored in the variable usersDTO. The gatekeeper session variable is then set to the value of un and the isadmin session variable is set using the isadmin attribute of the usersDTO. Then the `header` function will be used to send the user to the index page.

index.php

The logic of this page is the same as login.php until the conn variable is created. The only difference is the included files are functions.php and poiDAO.php. A poiDAO is then created passing the conn variable and table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Then the `findRegions` function will be called on it and the returned array will be stored in the variable regions. An if statement is then used to validate that regions is not null, if regions is null an error will be output to the page. If regions is not null a foreach loop will iterate thought the indexed array outputting them to the page. Before, closing the try catch block and calling the `footer` function.

addPoiForm.php

The logic of this page is the same as login.php until the end of the header. The only difference is the included file for this page is functions.php. An if statement is used to check if the gatekeeper session

variable is set, if not the header function is used to send the user to the login page. If set a form is output to the page with a text input for the POI (Point of Interest) name, description, type, and region. Then a select input is populated with a list of regions and a submit input is added before closing the form. Finally, the custom `footer` function is called.

addPoi.php

The logic of this page is the same as addPoiForm.php until the gatekeeper variable check. Then the name, type, country, region, and description sent by the addPoiForm page using POST are stored in variables. A variable is then created to store the value of the gatekeeper session variable. An if statement is then used to validate all variables using regular expressions and If any are invalid an error will be output to the page. If all are valid then a poiDTO is created passing null, name, type, country, region, description, and username as arguments and the returned poiDTO is then stored in the variable poiDTO. Then a poiDAO is created by passing conn and the table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Then the `add` function will be called on it passing the poiDTO as an argument, the returned poiDTO is then be stored in the returnedPOIDTO variable. The returnedPOIDTO will then be used to output the POI to the page, before closing the try catch block and calling the `footer` function.

Task B

The following pages/scripts will be needed: index.php, functions.php, and regionResult.php. Index.php will communicate with regionResults.php using a GET request, a GET request has been used in this situation because the database is not being updated.

regionResults.php

The logic of this page is the same as index.php until the conn variable is created. Then the region sent by the index page using GET is store in the variable region. An if statement is then used to validate region using a regular expression and if invalid an error will be output to the page. If valid then a poiDAO is created passing conn and the table name as arguments, the retuned poiDAO is then stored in the variable poiDAO. Then the `findByRegion` function will be called on it passing the region as an argument, the returned array will be stored in the variable pois. An if statement is then used to validate that pois is not null. If it is null an error will be output to the page. If it is not null, then a foreach loop will iterate thought the indexed array outputting the pois to the page. Then for each POI, a form that uses POST to send information to the addRecommendation page is output. With two hidden text inputs: POI ID and POI region followed by a submit input before closing the form. Then two query string links are output to the page the first to the reviewResults page with the POI ID included, the second to the addReviewForm page with the POI ID and name included. Before closing the try catch block and calling the `footer` function.

Task C

The following pages/scripts will be needed: functions.php, regionResult.php and addRecommendation.php. RegionResults.php will communicate with addRecommendation.php using a POST request sending two hidden values POI ID and POI region. The reason a hidden value has been used is to pass information received from one page to another without the user seeing it.

addRecommendation.php

The logic of this page is the same as index.php until the conn variable is created. Then the POI ID and region sent by the regionResults page using POST are stored in variables. An if statement is then used to validate the POI ID and region using a regular expression. If invalid an error will be output to

the page. If valid then a poiDAO is created passing conn and the table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Then the `addRecommendation` function will be called on it passing the POI ID and region as arguments.

Task D

The following pages/scripts will be needed: functions.php, regionResult.php and reviewResults.php. The regionResults.php page will communicate with reviewResults.php using a GET request.

reviewResults.php

The logic of this page is the same as index.php until the conn variable is created. The only difference is the included files are functions.php and reviewResults.php. Then the POI ID sent by the regionResults page using GET is stored in a variable. An if statement is then used to validate poiID using a regular expression before continuing. If invalid an error will be output to the page. If valid then a reviewsDAO is created passing conn and the table name as arguments, the retuned reviewsDAO is then stored in the reviewsDAO variable. Then the `findByPoiIdAndApproved` function will be called on it passing the poiID as an argument, the returned array will then be stored in the variable reviews. Then a poiDAO is created passing conn and the table name as arguments, the returned poiDAO is then stored in the poiDAO variable. Then the `findByid` function will be called on it passing the poiID as an argument, the returned poiDTO is then stored in the variable poi. An if statement is then used to validate that poi is not null, if poi is null an error will be output to the page. Then an else if statement is used to validate that reviews is not null, if reviews is null an error will be output to the page. If both are not null, then a foreach loop will iterate thought the indexed array outputting the reviews to the page before, closing the try catch block and calling the footer function.

Task E

The following pages/scripts will be needed: functions.php, regionResult.php, addReviewForm.php, and addReview.php. The regionResults.php page will communicate with addReviewForm.php using a GET request and the addReviewForm.php page will communicate with addReview.php using a POST reguest and one hidden value containing the POI ID.

addReviewForm.php

The logic of this page is the same as addPoiForm.php until the gatekeeper variable check. The only difference is the included files are functions.php and poiDAO.php. Then the POI ID and POI name sent by the regionResults page using GET are stored in variables. An if statement is then used to validate the POI ID and POI name variables using regular expressions. If either are invalid an error will be output to the page. If valid a form is output to the page with a text input for review, a hidden input for POI ID, and a submit input is added before closing the form. Finally, the custom footer function is called.

addReview.php

The logic of this page is the same as addReviewForm.php until the gatekeeper variable check. The only difference is the included files are functions.php and reviewsDAO.php. Then the POI ID and review sent by the addReviewForm page using POST are stored in variables. An if statement is then used to validate the POI ID and review variables using regular expressions before continuing. If either are invalid an error will be output to the page. If they are valid then a reviewsDTO is created passing null, POI ID, review, and 0 as the arguments, the returned reviewsDTO is then stored in the variable reviewsDTO. Then a reviewsDAO is created by passing conn and the table name as

arguments, the returned reviewsDAO is then stored in the variable reviewsDAO. Then the `addReview` function will be called on it passing the reviewsDTO as an argument, the returned review is then stored in the variable returnedReviewDTO. The returnedReviewDTO will then be used to output the review to the page, before closing the try catch block and calling the footer function.

Task F

The following pages/scripts will be needed: functions.php, reviewResultsAdmin.php, and approveReview.php. The reviewResultsAdmin.php page will communicate with approveReview.php using a POST request and one hidden value containing the review ID.

reviewResultsAdmin.php

The logic of this page is the same as index.php until the conn variable is created. The only difference is the included files are functions.php, reviewsDAO.php, and poiDAO.php. A reviewsDAO is created passing conn and the table name as arguments, the returned reviewsDAO is then stored in the variable reviewsDAO. Then the `findByUnapproved` function will be called on it, the returned array will then be stored in the review's variable. An if statement is then used to validate that the reviews variable is not null. If it is null an error will be output to the page. If reviews is not null, then a foreach loop will iterate thought the indexed array outputting the reviews to the page. For each review, a form that uses POST to send information to the approveReview page is output. With one hidden text input containing the review ID and a submit input before closing the form. Before closing the try catch block and calling the `footer` function.

approveReview.php

The logic of this page is the same as index.php until the conn variable is created. The only difference is the included files are functions.php and reviewsDAO.php. Then the review ID sent by the reviewResultsAdmin page using POST is stored in a variable. An if statement is then used to validate reviewID using a regular expression. If invalid an error will be output to the page. If valid then a reviewsDAO is created passing conn and the table name as arguments, the returned reviewsDAO is then stored in the variable reviewsDAO. Then the `approveReview` function will be called on it and the reviewID will be passed in as the only argument.

Task G

The following scripts will be needed: poiDTO.php, poiDAO.php, reviewsDTO.php, reviewsDAO.php, usersDTO.php, and usersDAO.php. These scripts will be called by other pages to request information from or update the database. All DAO's will be linked to their corresponding DTO's using the `include` function.

poiDTO.php, reviewsDTO.php, and usersDTO.php

These scripts are all similar the only difference is the number of attributes, below I will talk about the usersDTO.php script. It has 4 attributes: id, username, password, and isadmin. It will have a constructor function that takes 4 arguments and stores them in their corresponding attributes. There will be getter and setter functions for each attribute. Getters return the value contained in an attribute and setters update the value in an attribute. Finally, for diagnostic purposes they will all contain a display function that will output the attributes of a specific instance of an object to the page.

poiDAO.php

Will have 2 attributes: table and conn. Table to store the database table name and conn to store the PDO object. It will have a construct function that will allow a poiDAO object to be created, taking in a PDO object and table name as arguments and storing them in the corresponding attributes. There will be a `findRegions` function that runs a prepared SQL select statement using the conn and table variables to find all the regions in the database. (Prepared SQL statements are used to guard against SQL injection attacks.) A while loop will then iterate thought the returned rows and add them to an indexed array that will be returned to the page. It will also include a `findById` function that runs a prepared SQL select statement using the conn, table and poiID variables to pull out a specific database row. That row will then be used to create a poiDTO object that is returned to the page. There will also be an `addRecommendation` function that runs a prepared SQL update statement using the conn, table, and poiID variables to add 1 to the recommended value in the database. Then the `header` function will send the user to the regionResults page, using the region variable to complete the query string. Finally, there will be an `add` function that will execute a prepared SQL insert statement using the conn, table, and poiObj variables, populating the table columns with the POI objects attributes. The POI object will then have its ID attribute set to the ID from the database before being returned to the page.

reviewsDAO.php

The attributes and constructor for this script work the same as the ones for poiDAO.php. Then there will be a `findByPoiIdAndApproved` function that will run a prepared SQL select statement using the conn, table, and poiID variables to find all approved reviews for that POI. A while loop will then iterate thought the returned rows and add them to an indexed array that will be returned to the page. There will also be a `findByUnapproved` function that will run a prepared SQL select statement using the conn and table variables to find all reviews pending approval. A while loop will then iterate thought the returned rows and add them to an indexed array that will be returned to the page. There will also be an `addReview` function that will run a prepared SQL insert statement using the conn, table, and reviewObj variables, populating the table columns with the review objects attributes. The review object will then have its ID attribute set to the ID from the database before being returned to the page. Finally, there will be an `approveReview` function that will execute a prepared SQL update statement using the conn, table, and reviewID variables, to approve that specific review. Once completed the `header` function will return the user to the reviewResultsAdmin page.

usersDAO.php

The attributes and constructor for this script work the same as the ones for poiDAO.php. Then there will be a `verifyLogin` function that will run a prepared SQL select statement using the conn, table, username, and password variables. It will retrieve the matching database record and use the fetched row to create a usersDTO object that will be returned to the page.