Points of Interest Website Design and Analysis

By Matthew Dear

Task A

For 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

The functions.php script will contain five functions, that can be called by any page, this is done to reduce code repetition. The title function will take one argument a 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 company name and then call the links function and pass the user type as an argument. Links will then 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, a foreach loop will then iterate though it and output them to the page. If user type does not equal one it will follow the same process but, the awaiting approval link will not be included in the array. The back button function will output a button to the page that uses the JavaScript history function to return you to the previous page in your browser history. The database connection function will create a new PDO connection object for the database and set the error modes for better error reporting. Once created the PDO object will be returned to the requesting page. This entire process will be surrounded by a try catch block to stop any connection issues from completely breaking the site. The final function called footer will be used to echo out the HTML footer for the website.

login.php

This page will start with the PHP session start function that will either start a new session or resume an existing one. It then uses the PHP include function to link it to the functions and userDAO scripts to allow their functionality to be used. It will then call the custom title function from functions.php passing in the isadmin session variable if set otherwise, it will pass in 0 by default to make sure that users not logged in still get a navigation menu. Then using an if statement it checks if the gatekeeper session variable has been set, if it has the uses name will be output to the page. The custom back button function is then called completing the header section of the page. A try catch block is then opened and the custom database connection 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 a POST request are stored in the un and pw variables. An if statement is then used to validate the un and pw variables, using custom regular expressions, before continuing and to defend against cross site scripting attacks. If invalid an error will be output to the page the try catch block closed and the custom footer function called. If valid then a new usersDAO is created passing in the conn variable and hard coded table name as arguments, the retuned usersDAO is then stored in the variable usersDAO. Once created the verifyLogin function will be called on it passing in the un and pw variables as arguments and the returned usersDTO will be stored in the variable usersDTO. The gatekeeper session variable is the set to the value of the un variable and the isadmin session variable is set using the isadmin attribute of the usersDTO variable. Then the PHP 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 up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php and poiDAO.php. A new poiDAO is then created passing in the conn variable and the hard-coded table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Once created the find regions 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 the regions variable before continuing. If invalid an error will be output to the page the try catch block closed and the custom footer function called. If valid then a foreach loop will iterate thought the regions array outputting the regions to the page, before closing the try catch block and calling the custom footer function.

addPoiForm.php

The logic of this page is the same as login.php up until the end of the header. The only difference is the included file, for this page is functions.php. An if statement is then 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. 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 countries 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 index.php up until the PDO connection is stored in the conn variable. An if statement is then 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 it is then the name, type, country, region, and description sent by the addPoiForm page using a POST request are then store in variables with the same name. A username variable is then created to store the value of the gatekeeper session variable. An if statement is then used to validate all variables using custom regular expressions before continuing. If invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a new poiDTO is created passing eight arguments null, name, type, country, region, description, and username variables. The returned poiDTO is then stored in the variable poiDTO. Then a new poiDAO is created by passing in the conn variable and the hard-coded table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Once created the add function will be called on poiDAO passing the poiDTO variable as the only argument, the returned poiDTO will then be stored in the returnedPOIDTO variable. The returnedPOIDTO will then be used to output the new POI to the page, before closing the try catch block and calling the custom footer function.

Task B

For task b the following pages/scripts will be needed: index.php, functions.php, and regionResult.php. Index.php will communicate with regionResults.php useing a GET request, a GET request has been used in this situation because information is being display from the database.

regionResults.php

The logic of this page is the same as index.php up until the PDO connection is stored in the conn variable. First the region sent by the index page using a GET request is store in a variable with the same name. An if statement is then used to validate the region variable using custom regular expressions before continuing. If invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a new poiDAO is created passing the conn variable and the hard-coded table name as arguments, the retuned poiDAO is then stored in the variable poiDAO. Once created the findByRegion function will be called on it passing in the region as its only argument, the returned array will be stored in the variable pois. An if statement is then used to validate that the pois variable is not null. If it is an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a foreach loop will iterate thought the indexed array outputting the POI's to the page. For each POI, a form is then output to the page with two hidden text inputs one being POI ID the other the POI region and a submit input before closing the form. This form will allow a user to recommend that POI. Then two query string links our output to the page the first to the reviewResults page with the POI ID included allowing users to see reviews for that POI, and the second to the addReviewForm page with the POI ID and name included allowing users to add a review for that POI. Finally, after closing the try catch block the custom footer function will be called.

Task C

For 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 and two hidden values, one containing the POI ID and the other the POI region. The reason a hidden value has been used is to pass information received from one page to another.

addRecommendation.php

The logic of this page is the same as index.php up until the PDO connection is stored in the conn variable. Then the POI ID and region sent by the regionResults page using a POST request are stored in variables with the same names. An if statement is then used to validate the POI ID and region variables using a custom regular expression before continuing. If invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a new poiDAO is created passing the conn variable and the hard-coded table name as arguments, the returned poiDAO is then stored in the variable poiDAO. Once created the addRecommendation function will be called on it passing in the POI ID and region as arguments.

Task D

For 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 login.php up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php and reviewResults.php. Then the POI ID sent by the regionResults page using a GET request is stored in a variable with the same name. An if statement is then used to validate the POI ID variable using a custom regular expression before continuing. If invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a new reviewsDAO is created passing the conn variable and the hard-coded table name as arguments, the retuned reviewsDAO is then stored in the reviewsDAO variable. Once created the findByPoiIdAndApproved function will be called on it passing in the POI ID as its only argument, the returned array will be stored in the variable reviews. Then a new poiDAO is created passing in the conn variable and hard coded table name as arguments, the returned poiDAO is then stored in the poiDAO variable. Once created the findByid function will be called on it passing in the POI ID as its only argument, the returned POI 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, the try catch block closed, and the custom footer function called. 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, the try catch block closed, and the custom footer function called. If both are not null, then a foreach loop will iterate thought the indexed array outputting the reviews to the page. Finally, after closing the try catch block the custom footer function will be called.

Task E

For 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 login.php up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php and poiDAO.php. An if statement is then 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. Then the POI ID and POI name sent by the regionResults page using a GET request are stored in variables with the same names. An if statement is then used to validate the POI ID and POI name variables using a custom regular expression before continuing. If either are invalid an error will be output to the page, the try catch block closed, and the custom footer function called. 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 login.php up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php and reviewsDAO.php. An if statement is then 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. Then the POI ID and review sent by the addReviewForm page using a POST request are stored in variables with the same name. An if statement is then used to validate the POI ID and review variables using a custom regular expression before continuing. If either are invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If they are valid then a new reviewsDTO is created passing null, POI ID, review, and 0 (this sets the review to unapproved) as the arguments, the returned reviewsDTO is then stored in the variable reviewsDTO. Then a new reviewsDAO is created by passing in the conn variable and the hard-coded table name, the returned reviewsDAO is then stored in the variable reviewsDAO. Once created the addReview function will be called on it passing in the reviewsDTO as its only argument, the returned review is then stored in the variable returnedReviewDTO. The returnedReviewDTO will then be used to output the new review to the page, before closing the try catch block and calling the custom footer function.

Task F

For 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 login.php up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php, reviewsDAO.php, and poiDAO.php. A new reviewsDAO is created passing in the conn variable table name as arguments, the returned reviewsDAO is then stored in the variable reviewsDAO. Once created 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 an error will be output to the page, the try catch block closed, and the custom footer function called. If reviews are not null, then a foreach loop will iterate thought the indexed array outputting the reviews to the page. For each review, a form is then output to the page with one hidden text input containing the review ID and a submit input before closing the form. This form will allow a review to be approved. Finally, after closing the try catch block the custom footer is called.

approveReview.php

The logic of this page is the same as login.php up until the PDO connection is stored in the conn variable. The only difference is the included files, for this page they are functions.php and reviewsDAO.php. Then the review ID sent by the reviewResultsAdmin page using a POST request are stored in variables with the same name. An if statement is then used to validate the review ID using a custom regular expression before continuing. If invalid an error will be output to the page, the try catch block closed, and the custom footer function called. If valid then a reviewsDAO is created by passing the conn variable and the hard-coded table name as arguments, the returned reviewsDAO is then stored in the variable reviewsDAO. Once created the approveReview function will be called on it and the review ID will be passed in as the only argument.

Task G

For 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 private attributes, below I will talk about the usersDTO.php. This script will have 4 private attributes: id, username, password, and isadmin. It will have a public constructor function that takes 4 arguments and stores them in their corresponding attributes. There will be public 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

This script will have 2 private attributes: table and conn. Table to store the database table name and conn to store the PDO connection object. It will have a public construct function that will allow a new poiDAO object to be created, taking in the conn and table variables as arguments and storing them in the corresponding attributes. There will be a public findRegions function that runs a prepared SQL select statement on the database 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 then be returned to the page. It will also include a public findById function that runs a prepared SQL select statement on the database using the conn, table and poiId variables to pull out a specific database row. That row will then be used to create a new poiDTO object that is returned to the page. There will be a public addRecommendation function that runs a prepared SQL update statement on the database using the conn, table, and poiId variables to add one to the recommended value in the database. Then using the PHP header function will send the user to the regionResults.php page, using the region variable to complete the query string. Finally, there will be a public add. It will execute a prepared SQL insert statement on the database 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 will work in the same as the ones for poiDAO.php. There will be a public findByPoiIdAndApproved function that will run a prepared SQL select statement on the database using the conn, table, and poiId variables to find all reviews that have been approved for that POI. A while loop will then iterate thought the returned rows and add them to an indexed array that will then be returned to the page. There will also be a public findByUnapproved function that will run a prepared SQL select statement on the database 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 then be returned to the page. There will be a public addReview function that will run a prepared SQL insert statement on the database 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 requesting page. Finally, there will be a public approveReview function that will execute a prepared SQL update statement on the database using the conn, table, and reviewId variables, to approve that specific review. Once completed the PHP header function will return the user to the reviewResultsAdmin.php page.

usersDAO.php

The attributes and constructor for this script will work in the same as the ones for poiDAO.php. It will have a public verifyLogin function that will run a prepared SQL select statement on the database using the conn, table, un, and pw variables. It will retrieve the matching database record and use the fetched row to create a new usersDTO object that will be returned to the page.