UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 216 Practical Assignment 4

- Date Issued: **6 April 2020**
- Date Due: **19 April 2020** before **11:00**
- Submission Procedure: **Upload to the web server** (`wheatley`) **+ CS Web**
- This assignment consists of **5 tasks** for a total of **85 marks**.

1 Introduction

During this practical assignment you will be designing and developing a web site that will showcase a Music Database similar to what can be seen from ALLMUSIC (`https://www.allmusic.com`). Each assignment will build of the other in attempt to have a fully functional Music database listing website at the end of all the practicals.

The specific PHP web page for this assignment will showcase the following functionality:

- implementing the "update", "login" types of the API;
- ability to set and save user preferences;
- ability to rate a favourite music / album;
- secure your api from malicious attacks.

2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be able to give you the solutions.
3. You must produce all of the source files yourself; you may not use any tool to generate source files or fragments thereof automatically.
4. Your assignment will be viewed using Brave Web Browser (`https://brave.com/`) (the version in the labs or newer) so be sure to test your assignment in this browser. Nevertheless, you should take care to follow published standards and make sure that your assignment works in as many browsers as possible.
5. **You may not use external libraries to perform security operations (You may use PHP built-in functionality).**
6. **Server-side scripting should be done using an Object Oriented approach.**

3 Submission Instructions

You are required to upload all your source files (e.g. HTML5 documents, any images, etc.) to the web server (`wheatley`) and the CS Web in an compressed (zip) archive. Make sure that you test your submission to the web server thoroughly. All the menu items, links, buttons, *etc.* must work and all your images must load. Make sure that practical assignment works on the web server before the deadline. No late submissions will be accepted, so make sure you upload in good time. From the stipulated deadline time to after the last demonstration of the assignment, the web server will not be accepting updates to the files.

Note, `wheatley` is currently available from within the Hatfield campus, and on the DMZ (accessible anywhere). You must, therefore, not rely on always having access off-campus and should make sure that you are on campus when you `ftp` your assignment to the web server.

4 Online resources

Databases - `http://www.smartwebby.com/PHP/database_table_create.asp`

PHP Sessions - `http://www.w3schools.com/php/php_sessions.asp`

PHPMyAdmin - `http://www.phpmyadmin.net/home_page/index.php`

Timestamps - `https://en.wikipedia.org/wiki/Unix_time`

Cookie - `https://www.w3schools.com/js/js_cookies.asp`

5 Rubric for marking

Login API type	
HTML	2
Security	4
API + Validation	4
Cookie/Local DOM Storage	
Theme	10
API Key	3
JS	12
Update API type	
API + MYSQL	15
Filtering	5
Authorization + Validation	10
Rate API type	
API + MYSQL	5
Filtering	10
Authorization + Validation	5
Upload	
Does not work on `wheatley`	**-10**
Not uploaded to CS web	**-50**
Bonus	5
Total	**85**

6 Assignment Instructions

NOTE: For this practical you will need to fully integrate your PHP API to your website. This means that you should no longer query external API's from your website like you did in Practical 2, but query the information from your PHP API instead. You will also need to secure your API and prevent XSS and SQL attacks. Bonus marks will be given for extra security measures incorporated. Your API should perform error checking and provide meaningful error messages. It is good practice to perform validation of input on both client and server side.

Task 1: Login .... (10 marks)

Implement the login validation and verification from the previous practical. You will need to implement the "login" type for the API as well as your PHP website.

Important: Once a user has successfully logged in their API key needs to be returned, and any API requests need to use this API key (for retrieving music / album information and settings.). You will need to store this in the Cookie or local DOM storage as seen in Task 2. Ensure that your API only accepts valid requests.
You will also implement logout functionality which simply clears and removes the cookie/session.

Task 2: Cookie or Local DOM Storage .... (25 marks)

Storing the API key in the cookie or local DOM storage makes it easier to retrieve the key to make the requests to your PHP API. Once a user has logged in create either a cookie or local DOM storage with the API key that was returned during the login process. You will also use the cookie or local DOM storage for some CSS styling preferences. These preferences must be saved and remembered each time a user loads your site. That is you should store this in a Cookie, however you may also include this as a User preference and sync to your database. Many websites have a themed CSS styling, where a user is allowed to choose a theme (mostly light or dark). You need to implement this and at least have functionality to change between a theme in the footer of the webpage. You must have a light and dark theme. When a theme is changed it should dynamically be updated (should not reload the page).

Task 3: "update" PHP API type .... (30 marks)

For this task you will need to implement the "update" type for the API. This feature simply allows a user to change their preferences. These preferences **are** the filter types you have chosen in Practical 1. You should choose your own way of doing this, but remember that only registered users can update their preferences. You should at least have the user preference for (Genre, and Year). Once these are updated they should reflect on the "Trending" page, by already having this filter applied. For example if the user choose his/her preference for Genre to only display rock music / album the "Trending" page should only show rock music / album by applying the filter for Genre. You will need to restructure your database for this, make sure that the correct users preferences are updated.

In order to display this functionality you will need to either create a settings page or from the 'Trending' page have a button to save preferences based on the current filters. You need to explain your choice.

Task 4: "rate" PHP API type .... (20 marks)

For this task you will need to implement the "rate" type for the API. This feature simply allows a user to rate a music / album. You should have a way of rating a music / album in your HTML, this can be through a slider with a Modal pop-up or a simple dropdown element. Only registered users can rate a music / album. You should now update the "Top Rated" page to incorporate your own rating scale and provide a registered user the opportunity to rate music / album from your website in order to display that your PHP API works. You will also need to modify your database design. You should create a new database table with the Billboard Ranking, rating value, API key, remember to make use of the correct database relationships (Primary and Foreign Keys).

Task 5: Bonus .... (5 marks)

In order to receive any marks here you need to have all the tasks and functionality implemented.

You may add additional 'nice to have' features and depending on the level of difficulty marks will be given.