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

COS 216 Practical Assignment 4

- Date Issued: **3 May 2021**
- Date Due: **31 May 2021** before **08:00**
- Submission Procedure: **Upload to the web server** (`wheatley`) **+ ClickUP**
- This assignment consists of **6 tasks** for a total of **100 marks**.

1 Introduction

During this practical assignment you will be designing and developing a web site that will showcase a Video Game Database similar to what can be seen from RAWG (`https://rawg.io`). Each assignment will build of the other in attempt to have a fully functional video game database listing website at the end of all the practicals. **NB: It is important that you do not miss any practicals or you will fall behind.**

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 video game
- secure your api from malicious attacks
- implementing 'friend' functionality

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/`) 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 utilise any text editor or IDE, upon an OS of your choice, again, as long as you do not make use of any tools to generate your assignment.
6. All written code should contain comments including your name, surname and student number at the top of each file.
7. Your assignment must work on the `wheatley` web server, as you will be marked off there.
8. **You may not use external libraries to perform security operations (You may use PHP built-in functionality).**
9. **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 clickUP in a 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 your practical assignment works on the web server before the deadline. No late submissions will be accepted, so make sure you upload in good time. The server will not be accepting any uploads and updates to files from the stipulated deadline time until the end of the marking week (Friday at 3pm).

The deadline is on Sunday but we will allow you to upload until Monday 8am. After this NO more submissions will be accepted.

Note, `wheatley` is currently available from anywhere. But do not rely that outside access from the UP network will always work as intended. You must therefore make sure that you `ftp` your assignment to the web server. Also make sure that you do this in good time. A snapshot of the web server will be taken just after the submission was due and only files in the snapshot will be marked.

NB: You must also submit a ReadMe.txt file.
It should detail the following:

- how to use your website
- default login details (username and password) for a user you have on your API
- any functionality not implemented
- bonus features that you have implemented

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
Friend Functionality	
Views	4
Send Request	3
Accept/Reject Request	6
Unfriend	2
Upload	
Does not work on `wheatley`	**-10**
Not uploaded to clickUP	**-100**
Bonus	5
Total	**100**

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 in JS 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.

NB: 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 video game information and settings.). You will need to store this in a 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, you should 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 (the user should not have to 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.
Note: You should choose your own way of doing this, but remember that **only registered users** can update their preferences.

Once these are updated they should reflect on the "Trending" page, by already having this filter applied. For example if the user chooses his/her preference for Genre to only display Sports video games, then the "Trending" page should only show Sports video games by applying the filter for Genre. You will need to restructure your database for this. Make sure that the correct user's 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.

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 video game. You should have a way of rating a video game 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 video game.**

You should now update the "Top Rated" page to incorporate your own rating scale and provide a registered user the opportunity to rate video game 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 Metacritic score, rating value, API key, etc. Remember to make use of the correct database relationships (Primary and Foreign Keys).

Task 5: Friend Functionality .... (15 marks)

For this task you will need to implement friend functionality in your website. **Note**: This functionality will be used in the Homework Assignment, so try to complete during this practical so that you do not fall behind.

You will need to implement the following functionality:

- HTML views that display a list of your Friends and Friend Requests (this includes received requests as well as pending requests that you have sent). This can be done on a settings page, profile page, etc.
- Be able to Send a Friend Request to a user. The request should be visible in your Pending Requests once it has been sent.
- Be able to Accept/Reject Friend Requests from other users. If accepted, the user should appear in your Friends list. If rejected, the request should simply be removed from your Received Requests and removed from the other user's Pending Requests).
- Be able to Unfriend a user. This will remove a user from your Friends list.

You will also need to modify your database design. You should create a new database table to store the Friend data. You may structure it as you like to make the Friend functionality as simple as possible, but remember to make use of the correct database relationships (Primary and Foreign Keys).

Note: The Friend functionality simply needs to be done using SQL statements to query your database. Also, when accepting a Friend request, this automatically makes both users friends (i.e. the user who accepted the request does not need to send a request to the other user and wait for it to be accepted).

Task 6: 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. You can also add a better method of rating a video game by creating a CSS-based 10 star bar where users can click on the number of stars to give their rating. You can take it a step further by also allowing half stars and quarter stars.