Due Nov 4 by 11:59pm **Points** 100
Available Oct 15 at 12am - Nov 4 at 11:59pm 21 days

This assignment was locked Nov 4 at 11:59pm.

Module 4 Assignment

Purpose

Students will create a Movie Shopping Cart website using a combination of PHP, HTML, CSS and JavaScript.

Related Module Objectives

This assignment satisfies Module Objectives 1, 2, 3, 4 and 5.

Possible Points

This assignment is worth a maximum of 100 points.

Important Notes

- Students may refer to the following pages in case they forget how to perform the following tasks:
  - Access the **Course Web Server**
  - Viewing your **ePortfolio in a web browser**
- Students are required to request a **free** API key from the **Open Movie Database (OMDB) (http://www.omdbapi.com/apikey.aspx)** to complete the assignment successfully. While the OMDB API key is free, students are limited to **1,000** requests **per day** (an average of 1 request every 90 seconds). Students should avoid infinite loops and unnecessary API requests to avoid delays in the development and testing of their websites. As a recommendation, students should avoid adding more than **five** (**5**) movies to their shopping cart at any time - **one** (**1**) API request is required for **each movie** in the shopping cart **each time** the contents of the shopping cart are displayed.

- Students will expand upon their Module 4 assignment in Module 5; therefore, students are advised to carefully follow the Module 4 assignment instructions and logically organize their code in order to avoid having to make extensive modifications to their website to complete the Module 5 assignment.

Required Tools

Students will be required to use one or more of the following tools to earn a passing grade on the module assignment. Each of the tools listed below can be downloaded for free or already exist in the indicated operating system.

- Web browser (Chrome or Firefox recommended)
- Basic text editor
  - Notepad++ (Windows)
  - TextEdit in plain-text mode (Mac OS)
  - pico or vi (Linux)
- Secure Shell (SSH) client
  - PuTTy (Windows)
  - ssh (Mac OS and Linux)
- File transfer tool (must support SFTP via SSH - **DO NOT USE FTP**)
  - WinSCP (Windows)
  - CyberDuck (Mac OS)
  - sftp (Linux)

Warnings

- Unlike previous assignments, students will need to perform most of their coding (especially the PHP code) directly on the Course Web Server. This is required because PHP requires a web server in order to run. Please note, however, that this does not mean that students must edit their PHP files directly on the server.
- Unlike previous assignments, which involved client-side code only, students will need to manage server-side (PHP) and client-side (HTML, CSS and Javascript) code to make their website function properly. Many students struggle, at first, with server-side programming, as well as coordinating server-side and client-side functionality; therefore, students should avoid waiting too long before starting this assignment so they have plenty of time to become comfortable with the PHP language, server-side programming.
- Students must complete this assignment without the assistance of third-party development tools or frameworks such as jQuery or Bootstrap. Assignments that appear to be the product of third-party development tools or frameworks (professor's discretion) will receive **0** points.

Directions

- Review the requirements listed in the Assignment Requirements section
- Create a website that meets all of the stated requirements
- Complete the assignment before the due date (refer to the Course Schedule)
  - Note: Students will not submit anything to Canvas.

Assignment Requirements

Project Description

Students will create a movie shopping cart website using a combination of PHP, HTML, CSS and JavaScript. This website will require users to authenticate using a username and password before they can access their shopping cart. Once authenticated, users can search for movies, add movies to their shopping cart, and manage their shopping cart - add or remove movies.

Preliminary Tasks

- Log onto the **Course Web Server**
- Create a folder called images in the **module4** folder
  - Students are free to add pictures to the website
    - Store any pictures used in the website in the **images** folder
      - Use relative URLs to access the pictures
- Create a folder called data in the **module4** folder
  - In the **data** folder, create two (2) files
    - cart.db
    - credentials.db
  - Edit the **credentials.db** file
    - On a single line, create a user account that will be used to log onto and and interact with the website
      - The user account will contain the following data: **username**, **password**, **display name**, and **email address**
      - Separate each of the values using a **comma**
        - Example: jdoe,1234,John Doe,j.doe@gmail.com
      - **WARNING!!** The data stored in the credentials.db file is **NOT** encrypted. Students are advised to **NOT** use any of their **REAL** usernames or passwords with this assignment.
    - Be sure to save the file
- For the pages created for the website in this assignment
  - Students will include hyperlinks in the web pages
    - Hyperlinks to pages outside the web server (i.e., Wikipedia.org) should open the linked pages in a new browser **tab** or **window**

    - All other hyperlinks should open the linked pages in the same browser **tab** or **window**
- An automatic 10-point (10%) **penalty** will be assessed for a **disorganized** page.

PHP Files

User Authentication Page (10 points)

- Create a **User Authentication** page using the filename logon.php in the **module4** folder
  - The **logon.php** page consists of three (3) functions:
    - authenticateUser($username, $password)
    - displayLoginForm($message="")
    - processPageRequest()
  - The first statement on the page must be
    - ```
      processPageRequest(); // Call the processPageRequest() function
      ```
- **authenticateUser($username, $password) function**
  - Test whether the user entered valid login credentials
    - Read the data values from the credentials.db file
      - The data values include the **username, password**, **display name**, and **email address**
        - The data values are separated by commas
    - Convert the data values obtained from the credentials.db file into an array containing the four (4) values
    - Compare the **username** and **password** values obtained from the credentials.db file to the values stored in the **$username** and **$password** variables
      - If the strings match,
        - Create a **session**
        - Store the **display name** and **email address** values obtained from the credentials.db file in the session
        - Redirect the browser to the index.php page
      - If the strings do not match,
        - Create an appropriate error message
        - Call the displayLoginForm($message="") function
          - Pass the error message
- **displayLoginForm($message="") function**
  - Display the login form using the appropriate **HTML** statements
    - Give the page a descriptive title
    - Display myMovies Xpress! in some fashion (header text, picture, etc.) near the top of the page
    - Provide directions to the user (in plain text)
    - Create a **form** that includes the following **controls**:
      - Note: Use the form tag <form action="./logon.php" method="post">.
      - One (1) **text field** configured as required
        - Username
      - One (1) **password field** configured as required
      ```

        - Password
    - Two (2) **buttons**
        - Clear
            - This button should **reset** the form fields
        - Login
            - This button should s**ubmit** the form
- At the bottom of the page, include one (1) hyperlink to your **ePortfolio**
    - Use a relative URL for the **ePortfolio** link

- **processPageRequest() function**
    - Test whether any $_POST data was passed to the page
        - **Clear** all session variables
        - If $_POST data does not exist, call the displayLoginForm() function
        - If $_POST data does exist, call the authenticateUser($username, $password) function
            - Pass the appropriate values provided in the $_POST data

Shopping Cart Page (30 points)

- Create a **Shopping Cart** page using the filename index.php in the **module4** folder
    - The **index.php** page consists of five (5) functions
        - addMovieToCart($movieID)
        - checkout($name, $address)
        - displayCart()
        - processPageRequest()
        - removeMovieFromCart($movieID)
    - The first three (3) statements on the page must be
        - ```
          session_start(); // Connect to the existing session
          require_once '/home/mail.php'; // Add email functionality to the progr
          am
          processPageRequest(); // Call the processPageRequest() function
          ```

- **addMovieToCart($movieID) function**
    - Add the specified movie to the shopping cart
        - Read the **movieID** values from the cart.db file into an array
        - Add the **$movieID** value to the array
        - Write the array values to the cart.db file
            - Overwrite the contents of the cart.db file
    - Call the displayCart() function
- **checkout($name, $address) function**
    - Call the sendMail function to send an email to the user's email address
        - ```
          $result = sendMail([mail_id], [email_address], [display_name], [subject], [message]);
          ```
        - Warnings

        - The sendMail function will only allow one (1) email to be sent every 60 seconds
        - Allow up to 60 seconds for the message to arrive in your inbox
            - Ensure you are monitoring the inbox of the email address passed to the sendMail function
        - The sendMail function will return an error if an invalid **[mail_id]** value is provided
        - All of the values passed to the sendMail function are stored in a database
            - Derogatory or offensive material will be dealt with in accordance to the Course Syllabus and UNF Student Conduct policies
    - Parameter information
        - Replace **[mail_id]** with your unique **Email ID** (available in the Grades section on Canvas)
        - Replace **[email_address]** with the **$address** value
        - Replace **[display_name]** with the **$name** value
        - Replace **[subject]** with a string containing the **subject** of the message
            - Example: "Your Receipt from myMovies!"
        - Replace **[message]** with a string containing the **HTML message**
            - The HTML message should contain nearly all the same information as the **displayCart()** function
                - Do not include the following elements in the HTML message:
                    - Descriptive title
                    - The text Welcome, [display name] (logout)
                    - A hyperlink of the letter X for each movie in the cart
                    - The Add Movie and Checkout buttons
    - Return value information
        - The **sendMail** function will return one (1) of the following codes:

Code	Description
0	The email message was sent successfully
1 to 59	The time (in seconds) that remains before the next email can be sent
-1	An error occurred while sending the email message (email not sent)
-2	An invalid [mail_id] value was provided (email not sent)
-3	An error occurred while accessing the database (email not sent)

- **displayCart() function**
    - Display the contents of the shopping cart using the appropriate **HTML** statements
        - Read the **movieID** values from the cart.db file into an array
        - Give the page a descriptive title
        - In the top-left corner, display the message Welcome, [display name] (logout)
            - Replace **[display name]** with the display name value stored in the **session**
            - Create a hyperlink using the word **logout**
                - The link should call the JavaScript function confirmLogout()
        - Display myMovies Xpress! in some fashion (header text, picture, etc.) near the top of the page
        - Display the text [count] Movies in Your Shopping Cart

- Where **[count]** is the number of movieID values in the array
- If no movieID values exist in the array, display Add Some Movies to Your Cart
- If at least one (1) movieID value exists in the array,
  - Create an HTML table
  - For each movieID value in the array,
    - Call the OMDB API to obtain the data for the movieID using the statements below

```
$movie = file_get_contents('http://www.omdbapi.com/?apikey=[your_api_key]&i=[movie_id]&type=movie&r=json');
$array = json_decode($movie, true);
```

    - Replace **[your_api_key]** with your OMDB API KEY (see the Important Notes section)
    - Replace **[movie_id]** with the movieID value
  - Create a table row containing the following three (3) cells:
    - Movie image
      - Create an HTML image using the URL listed for the **Poster** key (see the Movie Data Request example)
      - Set the height argument to **100**
    - Movie title and year in the format title (year)
      - Use the values listed for the **Title** and **Year** keys (see the Movie Data Request example)
        - Create a hyperlink of the Title and Year
          - The link should call the relevant page on IMDB

```
https://www.imdb.com/title/[movie_id]/
```

    - A hyperlink of the letter X
      - The link should call the JavaScript function confirmRemove([title],[movie_id])
- Below the list of movies or the text Add Some Movies to Your Cart, add two (2) **Button** controls
  - Add Movie
    - The button's **onClick** action should call the ./search.php page
  - Checkout
    - The button's **onClick** action should call the JavaScript function confirmCheckout()

- **processPageRequest() function**
  - Test whether the $_GET['action'] value was passed to the page
    - If $_GET['action'] value does not exist, call the displayCart() function
    - If $_GET['action'] value does exist,
      - If $_GET['action'] value is add, call the addMovieToCart($movieID) function
        - Pass the $_GET['movie_id'] value to the function
      - if $_GET['action'] value is checkout, call the checkout($name, $address) function
        - Pass the **display name** and **email address** values stored in the **session**
      - If $_GET['action'] value is remove, call the removeMovieFromCart($movieID) function
        - Pass the $_GET['movie_id'] value to the function

- **removeMovieFromCart($movieID) function**
  - Remove the specified movie to the shopping cart
    - Read the **movieID** values from the cart.db file into an array
    - Search the array for the **$movieID** value
      - If found, remove the **$movieID** value from the array
    - Write the array values to the cart.db file
      - Overwrite the contents of the cart.db file
  - Call the displayCart() function

Movie Search Page (30 points)

- Create a **Movie Search** page using the filename search.php in the **module4** folder
  - The **search.php** page consists of three (3) functions
    - displaySearchForm()
    - displaySearchResults($searchString)
    - processPageRequest()
  - The first two (2) statements on the page must be
    - ```
      session_start(); // Connect to the existing session
      processPageRequest(); // Call the processPageRequest() function
      ```

- **displaySearchForm() function**
  - Display the search form using the appropriate **HTML** statements
    - Give the page a descriptive title
    - In the top-left corner, display the message Welcome, [display name] (logout)
      - Replace **[display name]** with the display name value stored in the **session**
      - Create a hyperlink using the word **logout**
        - The link should call the JavaScript function confirmLogout()
    - Display myMovies Xpress! in some fashion (header text, picture, etc.) near the top of the page
    - Provide directions to the user (in plain text)
    - Create a **form** that includes the following **controls**:
      - Note: Use the form tag <form action="./search.php" method="post"> to create the form.
    - One (1) **text field**
      - Keyword
      - The text field should be configured to require user input
    - Three (3) **buttons**
      - Cancel
        - The button's **onClick** action should call the ./index.php page
      - Clear
        - This button should **reset** the form fields
      - Search
        - The button should **submit** the form
- **displaySearchResults($searchString) function**
      ```

- Display the search results using the appropriate **HTML** statements
  - Call the OMDB API using the $searchString to obtain the search result data
    - ```
      $results = file_get_contents('http://www.omdbapi.com/?apikey=[your_api_key]&s='.urlencode([keyword]).'&type=movie&r=json');
      $array = json_decode($results, true)["Search"];
      ```
    - Replace **[your_api_key]** with your OMDB API KEY (see the Important Notes section)
    - Replace **[keyword]** with the $searchString value
  - Give the page a descriptive title
  - In the top-left corner, display the message Welcome, [display name] (logout)
    - Replace **[display name]** with the display name value stored in the **session**
    - Create a hyperlink using the word **logout**
      - The link should call the JavaScript function confirmLogout()
  - Display myMovies Xpress! in some fashion (header text, picture, etc.) near the top of the page
  - Display the text [count] Movies Found
    - Where [count] is the number of search results in the **$array**
  - If at least one (1) search result exists in the **$array**,
    - Create an HTML table
    - For each search result in the **$array**,
      - Create a table row with the following three (3) cells:
        - Movie image
          - Create an HTML image using the URL listed for the **Poster** key (see the Search Results example)
        - Movie title and year in the format title (year)
          - Use the values listed for the **Title** and **Year** keys (see the Search Results example)
            - Create a hyperlink of the Title and Year
              - The link should call the relevant page on IMDB
                - ```
                  https://www.imdb.com/title/[movie_id]/
                  ```
        - A hyperlink of the plus sign +
          - The link should call the JavaScript function addMovie(movieID)
            - Pass the value listed for the **imdbID** key (see the Search Results example)
  - One (1) **button**
    - Cancel
      - The button's **onClick** action should call the ./index.php page
- **processPageRequest() function**
  - Test whether any $_POST data was passed to the page
    - If $_POST data does not exist, call the displaySearchForm() function
    - If $_POST data does exist, call the displaySearchResults($searchString) function
      - Pass the appropriate value provided in the $_POST data

External CSS File (0 points)

- Modify the site.css file located in the **public_html** folder to control the presentation of the **User Authentication**, **Shopping Cart**,and **Movie Search** pages.
  - Have fun using CSS to format the pages, forms and controls

External JavaScript File (30 points)

- Create an external **JavaScript** file using the filename script.js in the **module4** folder
- Import the **script.js** file into the HTML generated by the index.php and search.php files
- Add the following functions to the **script.js** file:
  - **addMovie**(movieID)
    - Input
      - movieID - The imdbID of the movie to add to the shopping cart (see the Search Results example)
    - Process
      - Call the ./index.php page
        - Include the movie_id parameter and movieID value
          - `window.location.replace("./index.php?action=add&movie_id=" + movieID);`
      - Return true
    - Output
      - Return true

  - **confirmCheckout**()
    - Input
      - None
    - Process
      - Prompt the user to confirm they wish to checkout from **myMovies Xpress!**
        - Note: It does not matter if 0 movies exist in the shopping cart.
        - If the user clicks the **Cancel** button, return false
        - If the user clicks the **OK** button,
          - Call the ./index.php page
            - Include the action parameter and checkout value
              - `window.location.replace("./index.php?action=checkout");`
          - Return true
        - Output
          - Return true if user clicks the **OK** button; otherwise, return false

  - **confirmLogout**()
    - Input
      - None
    - Process

- Prompt the user to confirm they wish to logout of **myMovies Xpress!**
  - If the user clicks the **Cancel** button, return false
  - If the user clicks the **OK** button,
    - Call the ./logon.php page
      - Include the action parameter and logoff value
        - `window.location.replace("./logon.php?action=logoff");`
    - Return true
- Output
  - Return true if user clicks the **OK** button; otherwise, return false

- **confirmRemove**(title, movieID)
  - Input
    - title - The title of the movie to remove from the shopping cart (see the Search Results example)
    - movieID - The imdbID of the movie to remove from the shopping cart (see the Search Results example)
  - Process
    - Prompt the user to confirm they wish to remove the selected movie
      - Display the movie title in the prompt
    - If the user clicks the **Cancel** button, return false
    - If the user clicks the **OK** button,
      - Call the ./index.php page
        - Include the movie_id parameter and movieID value
          - `window.location.replace("./index.php?action=remove&movie_id=" + movieID);`
      - Return true
  - Output
    - Return true if user clicks the **OK** button; otherwise, return false

ePortfolio Page (0 points)

- Update your **ePortfolio** webpage (index.html) in the **public_html** folder
  - Create a hyperlink using the existing text Module 4 to open your **User Authentication** page (login.php)
    - Use a relative URL for the **User Authentication** link

Examples

Note: These examples are for reference only. These examples are included to help students understand the contents of the Movie Data and Search Results data provided by the OMDB API.

Movie Data Request : movieID = tt0133093

{"Title":"The Matrix","Year":"1999","Rated":"R","Released":"31 Mar 1999","Runtime":"136 min","Genre":"A
ction, Sci-Fi","Director":"Lana Wachowski, Lilly Wachowski","Writer":"Lilly Wachowski, Lana Wachowsk
i","Actors":"Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving","Plot":"Thomas A. Anders
on is a man living two lives. By day he is an average computer programmer and by night a hacker known a
s Neo. Neo has always questioned his reality, but the truth is far beyond his imagination. Neo finds hi
mself targeted by the police when he is contacted by Morpheus, a legendary computer hacker branded a te
rrorist by the government. Morpheus awakens Neo to the real world, a ravaged wasteland where most of hu
manity have been captured by a race of machines that live off of the humans' body heat and electrochemi
cal energy and who imprison their minds within an artificial reality known as the Matrix. As a rebel ag
ainst the machines, Neo must return to the Matrix and confront the agents: super-powerful computer prog
rams devoted to snuffing out Neo and the entire human rebellion.","Language":"English","Country":"US
A","Awards":"Won 4 Oscars. Another 34 wins & 47 nominations.","Poster":"https://images-na.ssl-images-am
azon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0O
TY@._V1_SX300.jpg","Ratings":[{"Source":"Internet Movie Database","Value":"8.7/10"},{"Source":"Rotten T
omatoes","Value":"87%"},{"Source":"Metacritic","Value":"73/100"}],"Metascore":"73","imdbRating":"8.
7","imdbVotes":"1,373,591","imdbID":"tt0133093","Type":"movie","DVD":"21 Sep 1999","BoxOffice":"N/A","P
roduction":"Warner Bros. Pictures","Website":"http://www.whatisthematrix.com","Response":"True"}

Search Results : keyword = matrix

{"Search":[{"Title":"The Matrix","Year":"1999","imdbID":"tt0133093","Type":"movie","Poster":"https://im
ages-na.ssl-images-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEy
XkFqcGdeQXVyNjU0OTQ0OTY@._V1_SX300.jpg"},{"Title":"The Matrix Reloaded","Year":"2003","imdbID":"tt02342
15","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BYzM3OGVkMjMtNDk3NS00N
Dk5LWJjZjUtYTVkZTIyNmQxNDMxXkEyXkFqcGdeQXVyNTAyODkwOQ@@._V1_SX300.jpg"},{"Title":"Matrix Revolution
s","Year":"2003","imdbID":"tt0242653","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/
images/M/MV5BNzNlZTZjMDctZjYwNi00NzljLWIwN2QtZWZmYmJiYzQ0MTk2XkEyXkFqcGdeQXVyNTAyODkwOQ@@._V1_SX300.jp
g"},{"Title":"The Matrix Revisited","Year":"2001","imdbID":"tt0295432","Type":"movie","Poster":"http
s://images-na.ssl-images-amazon.com/images/M/MV5BMTIzMTA4NDI4NF5BMl5BanBnXkFtZTYwNjg5Nzg4._V1_SX300.jp
g"},{"Title":"Armitage III: Dual Matrix","Year":"2002","imdbID":"tt0303678","Type":"movie","Poster":"ht
tps://images-na.ssl-images-amazon.com/images/M/MV5BOTUwOTY3Mjg1MF5BMl5BanBnXkFtZTcwODI2MTAyMQ@@._V1_SX3
00.jpg"},{"Title":"Sex and the Matrix","Year":"2000","imdbID":"tt0274085","Type":"movie","Poster":"N/
A"},{"Title":"Buhera mátrix","Year":"2007","imdbID":"tt0970173","Type":"movie","Poster":"https://images
-na.ssl-images-amazon.com/images/M/MV5BMGZiNzdmYWUtZTY0ZS00ZGU4LWE1NDgtNTNkZWM3MzQ0NDY4L2ltYWdlL2ltYWdl
XkEyXkFqcGdeQXVyMjIzMDAwOTc@._V1_SX300.jpg"},{"Title":"Return to Source: Philosophy & 'The Matrix'","Ye
ar":"2004","imdbID":"tt0439783","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/image
s/M/MV5BODIwNDQ3MTYtMWZiYS00MDYyLWI4ZGEtZjBkODU4NTgyNDFkXkEyXkFqcGdeQXVyMjM3ODA2NDQ@._V1_SX300.jpg"},
{"Title":"Making 'The Matrix'","Year":"1999","imdbID":"tt0365467","Type":"movie","Poster":"N/A"},{"Titl
e":"Sex Files: Sexual Matrix","Year":"2000","imdbID":"tt0224086","Type":"movie","Poster":"N/A"}],"total
Results":"81","Response":"True"}