Purpose

Students will modify their Movie Shopping Cart website to include AJAX, database, and email communication.

Related Project Objectives

This assignment satisfies Project Objectives 1, 2, 3, 4, 5 and 6.

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](https://canvas.unf.edu/courses/54204/pages/accessing-the-course-web-server)
  + Viewing your [ePortfolio in a web browser](https://canvas.unf.edu/courses/54204/pages/accessing-your-eportfolio" \o "Accessing Your ePortfolio)
* Students are required to modify the website (PHP, JavaScript and CSS files) they created for Project 4 to complete the Project 5 assignment successfully. Students that were unable to complete the Project 4 assignment successfully, in its entirety, will need to complete the Project 4 assignment to avoid additional deductions on their Project 5 assignment (the Project 4 assignment will remain available as a reference throughout Project 5).
* Students have the option to request a working copy of the Project 4 assignment from the professor in exchange for an automatic 10-point deduction on their Project 5 assignment grade. Please note, however, that the working copy of the Project 4 assignment offered by the professor is provided as-is - the professor will not modify the project to suit a particular student's needs nor will he explain how the code works.
* Students are required to request a **free** API key from the [Open Movie Database (OMDB) (Links to an external site.)](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.

Required Tools

Students will be required to use one or more of the following tools to earn a passing grade on the project. 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
  + BitVise (Windows)
  + ssh (Mac OS and Linux)
* File transfer tool (must support SFTP via SSH - **DO NOT USE FTP**)
  + BitVise (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, 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 AJAX programming; therefore, students should avoid waiting too long before starting this assignment so they have plenty of time to become comfortable with the AJAX 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 modify the movie shopping cart website they created during Project 4 to add functionality to the website using a combination of AJAX, PHP, HTML, CSS, JavaScript, database, and email communication. This website will require users to create a user account, validate their account, and authenticate using the username and password they created before they can access their shopping cart. Once authenticated, users can search for movies, view information about the movies, and manage their shopping cart - add or remove movies.

**Important Note**

Since students are expected to modify their existing Project 4 website files, the professor has copied each student's files from their project4 folder to their project5 folder on the Course Web Server.

**Preliminary Tasks**

* Log onto the [Course Web Server](https://canvas.unf.edu/courses/54204/pages/accessing-the-course-web-server)
* Create a folder called images in the **project5** folder  
  + Note: This folder should already exist from Project 4.
  + 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 templates in the **project5** folder  
  + Note: This folder should already exist from Project 4.
  + Students will store their HTML template files in the **templates** folder
    - Use relative URLs to access the HTML template files
* 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 and HTML Files**

User Authentication Page (40 points)

**WARNING!!** The username and password data stored in the database is **NOT** encrypted. Students are advised to **NOT** use any of their **REAL** usernames or passwords with this assignment.

* Modify the **User Authentication**PHP file logon.php in the **project5** folder
  + The **logon.php** file consists of eleven (11) functions:  
    - Note: The Project 4 functions that will be modified for Project 5 are colored BLUE.
    - authenticateUser($username, $password)
    - createAccount($username, $password, $displayName, $emailAddress)
    - displayCreateAccountForm()
    - displayForgotPasswordForm()
    - displayLoginForm($message="")
    - displayResetPasswordForm($userId)
    - processPageRequest()
    - resetPassword($userId, $password)
    - sendForgotPasswordEmail($username)
    - sendValidationEmail($userId, $displayName, $emailAddress)
    - validateAccount($userId)
  + The first three (3) statements on the page must be
    - require\_once '/home/common/mail.php'; // Add email functionality  
      require\_once '/home/common/dbInterface.php'; // Add database functionality  
      processPageRequest(); // Call the processPageRequest() function
  + Create the **User Authentication** HTML files in the **project5/templates** folder using the following filenames
    - Note: The Project 4 HTML files that will be modified for Project 5 are colored BLUE. New HTML files are colored RED.
    - create\_form.html - This file consists of HTML statements to display the **create account**form
    - forgot\_form.html - This file consists of HTML statements to display the **forgot password** form
    - logon\_form.html - This file consists of HTML and PHP statements to display the **logon** form
    - reset\_form.html - This file consists of HTML and PHP statements to display the **reset password** form
  + **authenticateUser($username, $password)**function
    - Test whether the user entered valid login credentials
      * Call the **dbInterface.php** function validateUser($username, $password)  
        + The function returns one of two (2) values:

An array containing the user's data: ID, Display Name and Email Address

NULL: The username and password values are invalid

* + - * If the **validateUser** function returns an array,
        + Create a **session**

Store the user's **ID**, **display name**and**email address**values in the session

* + - * + Redirect the browser to the index.php page
      * If the **validateUser**function returns a NULL value,
        + Create an appropriate error message
        + Call the displayLoginForm($message) function

Pass the error message

* + **createAccount($username, $password, $displayName, $emailAddress)** function
    - Store the user's account information in the database
      * Call the **dbInterface.php** function addUser($username, $password, $displayName, $email)  
        + The function returns one of two (2) values:

1 to ∞: The user ID of the new account

0: The provided username already exists

* + - * If the **addUser** function returns a value greater than 0,
        + Call the sendValidationEmail($userId, $displayName, $emailAddress) function

Pass the ID, display name and email address values

* + - * If the **addUser** function returns a value of 0,
        + Create an appropriate error message
        + Call the displayLoginForm($message) function

Pass the error message

* + **displayCreateAccountForm() function**
    - Use a require\_once statement to include the create\_form.html **template** file
  + **displayForgotPasswordForm()**function
    - Use a require\_once statement to include the forgot\_form.html **template** file
  + **displayLoginForm($message="")**function
    - Use a require\_once statement to include the logon\_form.html **template** file
  + **displayResetPasswordForm($userId)**function
    - Use a require\_once statement to include the reset\_form.html **template** file
  + **processPageRequest()**function
    - **Clear** all session variables
    - Test whether any $\_POST data was passed to the page  
      * If the $\_POST['action'] variable is set,  
        + Call the appropriate function (see below)
        + Pass the appropriate values provided in the $\_POST data
        + **create**: createAccount($username, $password, $displayName, $emailAddress) function
        + **forgot**: sendForgotPasswordEmail($username) function
        + **login**: authenticateUser($username, $password) function
        + **reset**: resetPassword($userId, $password) function
    - Test whether any $\_GET data was passed to the page
      * If the $\_GET['action'] variable is set,
        + Call the appropriate function (see below)
        + Pass the appropriate values provided in the $\_GET data
        + **validate**: validateAccount($userId) function
      * If the $\_GET['form'] variable is set,
        + Call the appropriate function (see below)
        + Pass the appropriate values provided in the $\_GET data, when appropriate
        + **create**: displayCreateAccountForm() function
        + **forgot**: displayForgotPasswordForm() function
        + **reset**: displayResetPasswordForm($userId) function
    - If neither $\_POST nor $\_GET data was passed to the page
      * Call the displayLoginForm() function
  + **resetPassword($userId, $password)**function
    - Update the user's password in the database
      * Call the **dbInterface.php** function resetUserPassword($userId, $password)  
        + The function returns one of two (2) values:

True: The user's password was successfully updated

False: Two (2) possible reasons for a False value:

1) The provided user ID does not exist

2) The provided "new" password is the same as the "current" password

* + - * If the **resetUserPassword**function returns true,
        + Create an appropriate success message
        + Call the displayLoginForm($message) function

Pass the success message

* + - * If the **resetUserPassword**function returns false,
        + Create an appropriate error message
        + Call the displayLoginForm($message) function

Pass the error message

* + **sendForgotPasswordEmail($username)**function
    - Send a message to the email address stored in the specified user's account
      * Call the **dbInterface.php** function getUserData($username)  
        + The function returns one of two (2) values:

An array containing the user's data: ID, Display Name and Email Address

NULL: The username value is invalid

* + - * Create an HTML message containing the following information
        + Display myMovies Xpress! in some fashion
        + Include the user's Display Name
        + Provide directions to the user about the password reset process

See the password reset message example at this [website (Links to an external site.)](https://www.eventbrite.com/support/articles/en_US/How_To/how-to-reset-your-password?lg=en_US) for ideas

* + - * + Include the following hyperlink

http://[server\_IP\_address]/~[studentID]/project5/logon.php?form=reset&user\_id=[User's ID]

* + - * Call the **mail.php** function sendMail to send an email to the user's email address
        + Note: Refer to the  **Reference Information**section at the end of the assignment.
  + **sendValidationEmail($userId, $displayName, $emailAddress) function**
    - Send a message to validate the user's account
      * Create an HTML message containing the following information
        + Display myMovies Xpress! in some fashion
        + Include the user's Display Name
        + Provide directions to the user about the account validation process

See the email verification message example at this [website  (Links to an external site.)](http://help.ablecommerce.com/mergedprojects/ablecommerce7/configure/product_reviews/product_reviews.htm)for ideas

* + - * + Include the following hyperlink

http://[server\_IP\_address]/~[studentID]/project5/logon.php?action=validate&user\_id=[User's ID]

* + - * Create a subject for the message
        + Example: "myMovies! Account Validation"
      * Call the **mail.php** function sendMail to send an email to the user's email address  
        + Note: Refer to the  **Reference Information**section at the end of the assignment.
  + **validateAccount($userId) function**
    - Activate the new user's account
      * Call the **dbInterface.php** function activateAccount($userId)  
        + The function returns one of two (2) values:

True: The specified User ID exists, and the account has been activated

False: The specified User ID does not exist

* + - * + If the **activateAccount**function returns true,

Create an appropriate success message

Call the displayLoginForm($message) function

Pass the success message

* + - * + If the **activateAccount**function returns false,

Create an appropriate error message

Call the displayLoginForm($message) function

Pass the error message

* + **create\_form.html functionality**
    - Display the create account 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" onsubmit="return validateCreateAccountForm();" method="post">
        + Four (4) **text fields** configured as required

Display Name

Email Address

Confirm Email Address

Username

* + - * + Two (2) **password fields**configured as required

Password

Confirm Password

* + - * + Three (3) **buttons**

Cancel

The button's **onClick** action should call the JavaScript function confirmCancel([form])

Note: Pass create as the **[form]** value.

 Clear

This button should **reset** the form fields

Create Account

This button should s**ubmit** the form

* + - * + One (1) **hidden field**

The value of the **name** argument **MUST** be action.

The value of the **value** argument **MUST** be create.

* + **forgot\_form.html functionality**
    - Display the forgot password 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

* + - * + Three (3) **buttons**

Cancel

The button's **onClick** action should call the JavaScript function confirmCancel([form])

Note: Pass forgot as the **[form]** value.

 Clear

This button should **reset** the form fields

Submit

This button should s**ubmit** the form

* + - * + One (1) **hidden field**

The value of the **name** argument **MUST** be action.

The value of the **value** argument **MUST** be forgot.

* + **logon\_form.html functionality**  
    - Display the login form using the appropriate **HTML** **and PHP**statements
      * Give the page a descriptive title
      * Display myMovies Xpress! in some fashion (header text, picture, etc.) near the top of the page
      * Determine if the $message variable contains a value
        + If the $message variable contains a value, use **HTML statement(s)** to display the message
      * 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

* + - * + One (1) **hidden field**

The value of the **name** argument **MUST** be action.

The value of the **value** argument **MUST** be login.

* + - * + Two (2) **hyperlinks**

Create Account

Note: Set link to ./logon.php?form=create

Forgot Password

Note: Set link to ./logon.php?form=forgot

* + - * At the bottom of the page, include one (1) hyperlink to your **ePortfolio**
        + Use a relative URL for the **ePortfolio**link
  + **reset\_form.html functionality**  
    - Display the reset password form using the appropriate **HTML and PHP**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" onsubmit="return validateResetPasswordForm();" method="post">
        + Two (2) **password fields**configured as required

Password

Confirm Password

* + - * + Three (3) **buttons**

Cancel

The button's **onClick** action should call the JavaScript function confirmCancel([form])

Note: Pass reset as the **[form]** value.

 Clear

This button should **reset** the form fields

Reset Password

This button should s**ubmit** the form

* + - * + Two (2) **hidden fields**

Hidden Field 1

The value of the **name** argument **MUST** be action

The value of the **value** argument **MUST** be reset

Hidden Field 2

The value of the **name** argument **MUST** be user\_id

The value of the **value** argument **MUST** be the **$userId** passed to the function

Shopping Cart Page (30 points)

* Modify the **Shopping Cart**PHP file index.php in the **project5** folder
  + Remove the following Project 4 functions from the index.php file:
    - readMovieData()
    - writeMovieData($array)
  + The **index.php** file consists of seven (7) functions:
    - Note: The Project 4 functions that will be modified for Project 5 are colored BLUE.
    - addMovieToCart($movieID)
    - checkout($name, $address)
    - createMovieList($forEmail=false)
    - displayCart($forEmail=false)
    - processPageRequest()
    - removeMovieFromCart($movieID)
    - updateMovieListing($order)
  + The first four (4) statements on the page must be
    - session\_start(); // Connect to the existing session  
      require\_once '/home/common/mail.php'; // Add email functionality  
      require\_once '/home/common/dbInterface.php'; // Add database functionality  
      processPageRequest(); // Call the processPageRequest() function
  + Create the **Shopping Cart** HTML files in the project5/templates folder using the following filenames
    - Note: The Project 4 HTML files that will be modified for Project 5 are colored BLUE. New HTML files are colored RED.
    - cart\_form.html - This file consists of HTML and PHP statements to display the contents of the shopping cart
    - movie\_list.html - This file consists of HTML and PHP statements to create a list of movies in the shopping cart
  + **addMovieToCart($movieID)**function
    - Add the specified movie to the shopping cart
      * Call the **dbInterface.php** function movieExistsInDB($imdbId)
        + Pass the movieID value (i.e., tt0133093)
        + This function returns one of two (2) values

1 to ∞: The ID of the movie (the movie exists in the database - no need to add it)

0: The movie does not exist in the database (need to add it)

* + - * If the movie does **NOT** exist in the database,
        + 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

* + - * + Call the **dbInterface.php** function addMovie($imdbId, $title, $year, $rating, $runtime, $genre, $actors, $director, $writer, $plot, $poster)

Pass the appropriate values contained in the **OMDB API** array

This function simply adds the specified movie information to the database (**NOT** to the shopping cart)

The function returns the ID of the movie just added to the database

Note: The movie is not added if it already exists.

* + - * Call the **dbInterface.php** function addMovieToShoppingCart($userId, $movieId)
        + Pass the user's ID value stored in the **session**
        + Pass the movie ID obtained from either the **movieExistsInDB** or **addMovie** functions
        + Note 1: This function does not return a value.
        + Note 2: This function does **NOT** allow duplicate entries (you **cannot** add the **same** movie twice).
      * Call the displayCart() function
  + **checkout($name, $address)**function
    - Send a receipt to the email address stored in the specified user's account
      * Create an HTML message containing the following information
        + Call the **displayCart(true)** function

Do not include the specified elements in the **displayCart** or **createMovieList** functions

* + - * Call the **mail.php** function sendMail to send an email to the user's email address  
        + Note: Refer to the  **Reference Information**section at the end of the assignment.
  + **createMovieList($forEmail=false)** function
    - Retrieve an **array** of movie IDs for the movies stored in the user's shopping cart
      * If the cart order value is stored in the **session**
        + Call the **dbInterface.php** function getMoviesInCart($userId, $order)

Pass the user's ID and cart order values stored in the session

The function returns an array containing the IDs of the movies in the user's shopping cart

* + - * If the cart order value does not exist in the **session**
        + Call the **dbInterface.php** function getMoviesInCart($userId)

Pass the user's ID value stored in the session

The function returns an array containing the IDs of the movies in the user's shopping cart

* + - Use the following PHP statements to create a string containing an HTML table of movies in the shopping cart
      * Note: Do **NOT** modify the PHP statements.
      * ob\_start(); // Create an output buffer  
        require\_once './templates/movie\_list.html';  
        $message = ob\_get\_contents(); // Get the contents of the output buffer  
        ob\_end\_clean(); // Clear the output buffer
    - Return the string containing the HTML code
  + **displayCart($forEmail=false)**function
    - Call the **dbInterface.php** function countMoviesInCart($userId)
      * Pass the user's ID value stored in the session
      * The function returns one of two (2) values
        + An integer greater than or equal to 0 indicating the number of movies in the user's shopping cart
        + False: The specified user ID is invalid
    - Call the createMovieList($forEmail) function
    - Use the following PHP statements to create a string containing an HTML page of the shopping cart
      * Note: Do **NOT** modify the PHP.
      * ob\_start(); // Create an output buffer  
        require\_once './templates/cart\_form.html';  
        $message = ob\_get\_contents(); // Get the contents of the output buffer  
        ob\_end\_clean(); // Clear the output buffer
    - Return the string containing the HTML code
  + **processPageRequest()**function
    - If the display name value is **NOT** stored in the session
      * Call the ./logon.php file
    - Test whether any $\_GET data was passed to the page
      * If the $\_GET['action'] variable is set, call the appropriate function (see below)
        + Note: Pass the appropriate values provided in the $\_GET data
        + **add**

Call the addMovieToCart($movieId) function

Echo the output of the displayCart() function

echo displayCart(); // The echo is very important

* + - * + **checkout**: Call the checkout($name, $address) function
        + **remove**

Call the removeMovieFromCart($movieId) function

Echo the output of the displayCart() function

echo displayCart(); // The echo is very important

* + - * + **update**: Call the updateMovieListing($order) function
    - If $\_GET data was not passed to the page
      * Call the displayCart() function
  + **removeMovieFromCart($movieID)**function
    - Remove the specified movie to the shopping cart
      * Call the **dbInterface.php** function removeMovieFromShoppingCart($userId, $movieId)
        + This function returns one of two (2) values

True: The specified movie was removed from the shopping cart

False: The specified movie does not exist in the shopping cart

* + - * Call the displayCart() function
  + **updateMovieListing($order)**function
    - Updates the movies listed in the shopping cart in the specified order
      * Note: The **updateMovieListing** function should only be called by the JavaScript function **changeMovieDisplay()** (via the **processPageRequest** function) to support AJAX functionality of the shopping cart.
      * Store the **$order** value in the **session**using the cart order value
      * Call the createMovieList(false) function
      * Echo the string returned by the **createMovieList**function
        + Do **not** return the string
  + **cart\_form.html functionality**
    - Display the contents of the shopping cart using the appropriate **HTML** **and PHP**statements
      * Note: Pay attention to the  "Do not include this if **$forEmail** is true" notes
      * Give the page a descriptive title
        + Do not include this if **$forEmail** is true
      * In the top-left corner, display the message Welcome, [display name] (logout)
        + Do not include this if **$forEmail** is true
        + Replace **[display name]** with the display name value stored in the PHP **session** variable
        + 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
      * If 0 movies exist in the shopping cart, display Add Some Movies to Your Cart
      * If at least one (1) movie exists in the array,
        + Display the text [count] Movies in Your Shopping Cart

Where **[count]** is the number of movies in the user's shopping cart

Replace **[count]** with the number of movieID values in the **PHP array**

* + - * + Add one (1) HTML **select**element to allow users to arrange the movies in different ways

Use the following HTML for the select element (do **NOT** modify the HTML)

<select id='select\_order' onchange='changeMovieDisplay();'>

Give the select element an appropriate label

Add the following **options** to the select element:

Movie Title

Set value to **0**

Set as the **default** option

Runtime (shortest -> longest)

Set value to **1**

Runtime (longest -> shortest)

Set value to **2**

Year (old -> new)

Set value to **3**

Year (new -> old)

Set value to **4**

* + - * + Add one (1) HTML **div** element

Use the following HTML for the div element (do **NOT** modify the HTML)

<div id='shopping\_cart'> </div>

Store the list of movies  between the **<div> </div>** elements

* + - * + Below the list of movies or the text Add Some Movies to Your Cart, add two (2) **Button** controls

Do not include the buttons if **$forEmail** is true

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()

* + - * + Add the following HTML code immediately above the **</body>** tag (do **NOT** modify the HTML)

Do not include this if **$forEmail** is true

<div id='modalWindow' class='modal'>  
 <div id='modalWindowContent' class='modal-content'>  
 </div>  
</div>

* + **movie\_list.html functionality**
    - Generate a list of movies in the specified order using the appropriate **HTML and PHP**statements
      * Note: Pay attention to the  "Do not include this if **$forEmail** is true" notes
      * Create an HTML table
      * For each movieID value contained in the array,
        + Call the **dbInterface.php** function getMovieData($movieId)

Pass the movieID value

The function returns one of two (2) values

An array containing the information about the specified movie

The following four (4) keys are needed for each row contained in the array:

ID

Title

Year

Poster

NULL: The movie ID value is invalid

* + - * + Create a table **row** containing the following four (4) cells:

Movie image

Create an HTML image using the URL value listed for the **Poster**key in the array

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 in the array

A hyperlink of the text View More Info

Do not include this if **$forEmail** is true

Use the following HTML statement (exactly as shown)

Note: The [movie\_id] is the **ID** key in the array (not the IMDB\_ID key)

<a href="javascript:void(0);" onclick='displayMovieInformation([movie\_id]);'>View More Info</a>

A hyperlink of the letter X

Do not include this if **$forEmail** is true

The link should call the JavaScript function confirmRemove([title],[movie\_id])

Note: The [movie\_id] is the **ID** key in the array (not the IMDB\_ID key)

* + - * Close the HTML table

Movie Search Page (10 points)

* Modify the Movie Search PHP file search.php in the **project5** folder
  + The **search.php** file consists of three (3) functions
    - Note: The Project 4 functions that will be modified for Project 5 are colored BLUE.
    - displaySearchForm()
    - displaySearchResults($searchString)
    - processPageRequest()
  + The first two (2) statements on the page must be
    - session\_start(); // Connect to the existing session  
      processPageRequest(); // Call the processPageRequest() function
  + Modify the **Movie Search** HTML files in the project5/templates folder using the following filenames
  + Note: The Project 4 HTML files that will be modified for Project 5 are colored BLUE.
    - search\_form.html - This file consists of HTML and PHP statements to display the **search** form
    - results\_form.html - This file consists of HTML and PHP statements to display the **search results** form
  + **displaySearchForm()**function
    - Use a require\_once statement to include the search\_form.html **template** file
  + **displaySearchResults($searchString)**function
    - Call the OMDB API using the $searchString to obtain the search result data
      * Note: The resulting array will be used in the results\_form.html file
      * $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
    - Use the require\_once statement to include the results\_form.html **template** file
  + **processPageRequest() function**
    - If the display name value is **NOT** stored in the session
      * Call the ./logon.php file
    - 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
  + **results\_form.html functionality**
    - Display the search form using the appropriate **HTML and PHP**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 PHP **session**variable
        + 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
        + Replace **[count]** with the number of search result values in the **PHP array**
        + If at least one (1) **search result**value exists in the **PHP array**,

Use **HTML statement(s)** to display the search results

Create an HTML table

Loop through each search result value in the **PHP array**

For each search result value in the **PHP array**,

Create an HTML table row containing the following three (3) cells:

* Movie image
  + Create an HTML image using the URL listed for the **Poster** key
    - See the **Search Results**example in the **Examples** section at the end of the assignment
    - Combine the **HTML image** statement and the **PosterURL** value
  + 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 **Search Results** example in the **Examples** section at the end of the assignment
    - Combine the **HTML text** statement and the **Title** and **Year** values
  + Create an HTML hyperlink of the Title and Year
    - The hyperlink should call the relevant page on IMDB
    - https://www.imdb.com/title/[movie\_id]/
      * Replace **[movie\_id]** with the imdbID key
* An HTML hyperlink of the plus sing +
  + The hyperlink should call the JavaScript function addMovie([movie\_id])
    - Replace **[movie\_id]** with the **imdbID**value

Below the complete list of search results or the text 0 Movies Found, add one (1) HTML button

Cancel

The button's **onClick** action should call the ./index.php page

* + **search\_form.html functionality**
    - Display the search form using the appropriate **HTML** **and PHP**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 PHP **session**variable
        + 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

Movie Info Page (10 points)

**Important Note**: An example of the Movie Info page output is provided in the Examples section at the end of this assignment.

* Create the **Movie Info** PHP file using the filename movieinfo.php in the **project5** folder
  + Note: The **movieinfo.php** page should only be called by the JavaScript function **displayMovieInformation**() to support AJAX functionality of the shopping cart in the **index.php** file.
  + The movieinfo.php page consists of two (2) functions
    - createMessage($movieId)
    - processPageRequest()
  + The first three (3) statements on the page must be
    - session\_start(); // Connect to the existing session  
      require\_once '/home/common/dbInterface.php'; // Add database functionality  
      processPageRequest(); // Call the processPageRequest() function
  + Create the Movie Info HTML file using the filename movie\_info.html in the **project5/templates** folder
    - The **movie\_info.html** file consists of HTML and PHP statements to display the pop-up modal window
  + **createMessage($movieId)**function
    - Call the **dbInterface.php** function getMovieData($movieId)
      * Pass the movieId value
      * The function returns one of two (2) values
        + An array containing the information about the specified movie
        + NULL: The movie ID value is invalid
    - Use the following PHP statements to create a string containing an HTML page of the pop-up modal window
      * Note: Do **NOT** modify the PHP.
      * ob\_start(); // Create an output buffer  
        require\_once './templates/movie\_info.html';  
        $message = ob\_get\_contents(); // Get the contents of the output buffer  
        ob\_end\_clean(); // Clear the output buffer
    - Echo the string containing the message created by the HTML statements above
      * Do **NOT** return the string
  + **processPageRequest()**function
    - If the display name value is **NOT** stored in the session
      * Call the ./logon.php file
    - Test whether any $\_GET data was passed to the page
      * If the $\_GET['movie\_id'] variable is set, call the createMessage($movieId) function
        + Pass the value provided in the $\_GET data
    - If $\_GET data was not passed to the page
      * Call the createMessage($movieId) function
        + Pass 0 as the **movieId** value
  + **movie\_info.html functionality**
    - Generate a message containing information about the specified movie using the appropriate **HTML** **and PHP**statements
      * Create a string containing a message using the following HTML code
        + Do **NOT** **modify** the HTML code

<div class='modal-header'>  
 <span class='close'>[Close]</span>  
 <h2>**Title** (**Year**) Rated **Rating** **Runtime**<br />**Genre**</h2>  
</div>  
<div class='modal-body'>  
 <p>Actors: **Actors**<br />Directed By: **Director**<br />Written By: **Writer**</p>  
</div>  
<div class='modal-footer'>  
 <p>**Plot**</p>  
</div>

* + - * If the **getMovieData** function returned a NULL value,
        + Replace the **Plot** field with the text Invalid Movie ID!
      * If the **getMovieData** function returned an array,
        + Replace the fields (i.e., **Title**, **Year**, **Rating**, etc.) with the appropriate data in the array

**External CSS File** (0 points)

* Modify the site.css file located in the **public\_html/css/** folder to control the presentation of the **User Authentication**,**Shopping Cart**and**Search**pages.
* Add the following code to the site.css file to support the **Movie Info** page

/\* The following CSS code blocks produce a pop-up modal window to display additional movie information \*/  
  
/\* The Modal (background) \*/  
.modal {  
 display: none; /\* Hidden by default \*/  
 position: fixed; /\* Stay in place \*/  
 z-index: 1; /\* Sit on top \*/  
 padding-top: 100px; /\* Location of the box \*/  
 left: 0;  
 top: 0;  
 width: 100%; /\* Full width \*/  
 height: 100%; /\* Full height \*/  
 overflow: auto; /\* Enable scroll if needed \*/  
 background-color: rgb(0,0,0); /\* Fallback color \*/  
 background-color: rgba(0,0,0,0.4); /\* Black w/ opacity \*/  
}  
  
/\* Modal Content \*/  
.modal-content {  
 position: relative;  
 background-color: #fefefe;  
 margin: auto;  
 padding: 0;  
 border: 1px solid #888;  
 width: 80%;  
 box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2),0 6px 20px 0 rgba(0,0,0,0.19);  
 animation-name: animatetop;  
 animation-duration: 0.4s  
}  
  
/\* The Close Button \*/  
.close {  
 color: white;  
 float: right;  
 font-size: 28px;  
 font-weight: bold;  
}  
  
.close:hover,  
.close:focus {  
 color: #000;  
 text-decoration: none;  
 cursor: pointer;  
}  
  
.modal-header {  
 padding: 2px 16px;  
 background-color: #5cb85c;  
 color: white;  
}  
  
.modal-body {padding: 2px 16px;}  
  
.modal-footer {  
 padding: 2px 16px;  
 background-color: #5cb85c;  
 color: white;  
}

**External JavaScript File** (10 points)

* Modify the external **JavaScript** file with the filename script.js in the **project5** folder
  + Import the **script.js** file into the  following HTML **template** files
    - cart\_form.html
    - create\_form.html
    - forgot\_form.html
    - reset\_form.html
    - results\_form.html
    - search\_form.html
  + 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
    - **confirmCancel**(form)
      * Input
        + form - The form being cancelled

create - The Create Account form

forgot - The Forgot Password form

reset - The Reset Password form

search - The Movie Search form

* + - * Process
        + Prompt the user to confirm they wish to cancel the specified form

If the user clicks the **Cancel** button, return false

If the user clicks the **OK** button,

For the **create**, **forgot** or **reset** forms

Call the logon.php page

window.location.replace("./logon.php");

For the **search**form

Call the index.php page

window.location.replace("./index.php");

* + - * + Return true
      * Output
        + Return false if the user clicks the **Cancel** button; otherwise, return true
    - **changeMovieDisplay**()
      * Input
        + None
      * Process
        + Get the currently selected **option** of the HTML **select** element

Note: Should be a value between 0 [Movie Title] and 4 [Year (new -> old)]

* + - * + Create a new **XMLHttpRequest** object

Refer to this w3schools.com [AJAX example (Links to an external site.)](https://www.w3schools.com/xml/ajax_php.asp) for assistance

* + - * + Define a function for the XMLHttpRequest.onreadystatechange event handler

Include the following statement (do **NOT** modify the code)

document.getElementById("shopping\_cart").innerHTML= this.responseText;

* + - * + Call the XMLHttpRequest.open() function using the following parameters (do **NOT** modify the parameters)
        + xhttp.open("GET", "./index.php?action=update&order=" + [selected value], true);

Replace [selected value] with the value of the **selected option** (0 to 4)

* + - * + Call the XMLHttpRequest.send() function
      * Output
        + None
    - **confirmCheckout**()
      * Input
        + None
      * Process
        + Prompt the user to confirm they wish to checkout

If the user clicks the **Cancel** button, return false

If the user clicks the **OK** button,

Call the index.php page

window.location.replace("./index.php?action=checkout");

Return true

* + - * Output
        + Return false if the user clicks the **Cancel** button; otherwise, return true
    - **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, movie\_id**)**
      * Input
        + title - The title of the movie to remove from the shopping cart (see the Search Results example)
        + movie\_id - The ID of the movie to remove from the shopping cart
      * 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
    - **displayMovieInformation**(movie\_id)
      * Input
        + movie\_id - The ID of the movie for which to display additional information
      * Process
        + Create a new **XMLHttpRequest** object

Refer to this w3schools.com [AJAX example (Links to an external site.)](https://www.w3schools.com/xml/ajax_php.asp) for assistance

* + - * + Define a function for the XMLHttpRequest.onreadystatechange event handler

Include the following statement (do **NOT** modify the code)

document.getElementById("modalWindowContent").innerHTML= this.responseText;  
showModalWindow();

* + - * + Call the XMLHttpRequest.open() function using the following parameters (do **NOT** modify the parameters)
        + xhttp.open("GET", "./movieinfo.php?movie\_id=" + [movie\_id], true);

Replace [movie\_id] with the specified **movie\_id** value

* + - * + Call the XMLHttpRequest.send() function
      * Output
        + None
    - **forgotPassword**()
      * Input
        + None
      * Process
        + Call the logon.php page

Include the **action**parameters

window.location.replace("./logon.php?action=forgot");

Return true

* + - * Output
        + Return true
    - **showModalWindow**()
      * Use the following code for the function (do **NOT** modify the code)

function showModalWindow()  
{  
 var modal = document.getElementById('modalWindow');  
 var span = document.getElementsByClassName("close")[0];  
  
 span.onclick = function()   
 {   
 modal.style.display = "none";  
 }  
  
 window.onclick = function(event)   
 {  
 if (event.target == modal)   
 {  
 modal.style.display = "none";  
 }  
 }  
   
 modal.style.display = "block";  
}

* + - **validateCreateAccountForm**()
      * Input
        + None
      * Process
        + If the values in **any** of the form fields (except Display Name) contain one (1) or more **spaces**

Display an appropriate error message

* + - * + If the values in the Email Address and Confirm Email Address fields are not **equal**

Display an appropriate error message

* + - * + If the values in the Password and Confirm Password fields are not **equal**

Display an appropriate error message

* + - * Output
        + Return true if **all** of the above tests pass; otherwise, return false
    - **validateResetPasswordForm**()
      * Input
        + None
      * Process
        + If the values in the Password and/or Confirm Password fields contain one (1) or more **spaces**

Display an appropriate error message

* + - * + If the values in the Password and Confirm Password fields are not **equal**

Display an appropriate error message

* + - * Output
        + Return true if **all** of the above tests pass; 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 Project 5 to open your **User Authentication**page (logon.php)  
    - Use a relative URL for the **User Authentication** link

**Reference Information**

Note: Student can use the **sendMail()** function to send an email to a user's email address.

* Call the **mail.php** function sendMail to send an email to a user's email address
  + $result = sendMail([mail\_id], [email\_address], [display\_name], [subject], [message]);
* Parameter information
  + Replace **[mail\_id]** with your unique **Email ID** (available in the Grades section on Canvas)
  + Replace **[email\_address]** with the user's **Email Address**
  + Replace **[display\_name]** with the user's **Display Name**
  + Replace **[subject]** with a string containing the **subject** of the message
    - Example: "myMovies! Password Reset Request"
  + Replace **[message]** with the **HTML message** to send in the email
* Warnings
  + Students should only send messages to an existing and valid email address used by the student
  + The sendMail function will only allow one (1) email to be sent by the same Email ID 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
  + 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
* Return value information
  + The **sendMail** function will return one (1) of the following codes to indicate its success or failure

|  |  |
| --- | --- |
| **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) |

**Examples**

Note: These examples are for reference only. These examples are included to help students understand the contents of the Pop-up Modal Window, as well as the Movie Data and Search Results data provided by the OMDB API.

Pop-up Modal Window

* Students can download a [Modal Window ExamplePreview the document](https://canvas.unf.edu/courses/54204/files/6467221/download?wrap=1) to see how the HTML, JavaScript, and CSS files work together.
* The modal window example has been tested on the **latest versions** of the following browsers
  + Chrome, Chromium, Edge, Firefox, Internet Explorer, Opera, and Safari

Movie Data Request : movieID = tt0133093

{"Title":"The Matrix","Year":"1999","Rated":"R","Released":"31 Mar 1999","Runtime":"136 min","Genre":"Action, Sci-Fi","Director":"Lana Wachowski, Lilly Wachowski","Writer":"Lilly Wachowski, Lana Wachowski","Actors":"Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving","Plot":"Thomas A. Anderson is a man living two lives. By day he is an average computer programmer and by night a hacker known as Neo. Neo has always questioned his reality, but the truth is far beyond his imagination. Neo finds himself targeted by the police when he is contacted by Morpheus, a legendary computer hacker branded a terrorist by the government. Morpheus awakens Neo to the real world, a ravaged wasteland where most of humanity have been captured by a race of machines that live off of the humans' body heat and electrochemical energy and who imprison their minds within an artificial reality known as the Matrix. As a rebel against the machines, Neo must return to the Matrix and confront the agents: super-powerful computer programs devoted to snuffing out Neo and the entire human rebellion.","Language":"English","Country":"USA","Awards":"Won 4 Oscars. Another 34 wins & 47 nominations.","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@.\_V1\_SX300.jpg","Ratings":[{"Source":"Internet Movie Database","Value":"8.7/10"},{"Source":"Rotten Tomatoes","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","Production":"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://images-na.ssl-images-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@.\_V1\_SX300.jpg"},{"Title":"The Matrix Reloaded","Year":"2003","imdbID":"tt0234215","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BYzM3OGVkMjMtNDk3NS00NDk5LWJjZjUtYTVkZTIyNmQxNDMxXkEyXkFqcGdeQXVyNTAyODkwOQ@@.\_V1\_SX300.jpg"},{"Title":"Matrix Revolutions","Year":"2003","imdbID":"tt0242653","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BNzNlZTZjMDctZjYwNi00NzljLWIwN2QtZWZmYmJiYzQ0MTk2XkEyXkFqcGdeQXVyNTAyODkwOQ@@.\_V1\_SX300.jpg"},{"Title":"The Matrix Revisited","Year":"2001","imdbID":"tt0295432","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BMTIzMTA4NDI4NF5BMl5BanBnXkFtZTYwNjg5Nzg4.\_V1\_SX300.jpg"},{"Title":"Armitage III: Dual Matrix","Year":"2002","imdbID":"tt0303678","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BOTUwOTY3Mjg1MF5BMl5BanBnXkFtZTcwODI2MTAyMQ@@.\_V1\_SX300.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/MV5BMGZiNzdmYWUtZTY0ZS00ZGU4LWE1NDgtNTNkZWM3MzQ0NDY4L2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyMjIzMDAwOTc@.\_V1\_SX300.jpg"},{"Title":"Return to Source: Philosophy & 'The Matrix'","Year":"2004","imdbID":"tt0439783","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BODIwNDQ3MTYtMWZiYS00MDYyLWI4ZGEtZjBkODU4NTgyNDFkXkEyXkFqcGdeQXVyMjM3ODA2NDQ@.\_V1\_SX300.jpg"},{"Title":"Making 'The Matrix'","Year":"1999","imdbID":"tt0365467","Type":"movie","Poster":"N/A"},{"Title":"Sex Files: Sexual Matrix","Year":"2000","imdbID":"tt0224086","Type":"movie","Poster":"N/A"}],"totalResults":"81","Response":"True"}

**Submission**

[Submission Details](https://canvas.unf.edu/courses/54204/assignments/494306/submissions/19644)

Grade: 100 (100 pts possible)

Graded Anonymously: no

**Comments:**

No Comments