COMP 3512 Assignment #2

Due Wed, April 6th at noon
Version 1.0, Feb 28

Overview

This assignment is a group project that expands your first assignment in functionality and scope. It is broken into several milestones with different dates. The milestones are in place to ensure your group is progressing appropriately.

Some of the specific details for some of the milestones and pages will develop over time; that is, this assignment specification is a living document that will grow over the next several weeks.

Composition

You can work in groups of two or three for this assignment.

If working in a group, each member needs to take responsibility for and complete an appropriate amount of the project work.

Hosting

You will be using JavaScript, PHP, and MySQL in this assignment. Eventually this will mean your assignment will need to reside on a working host server. In the tutorials (and the labs, if you are working on them), you make use of XAMPP on your local development machine as both host server and as a development environment. While easy, it means no one but you can ever see your work. If you ever want to use your assignment as a portfolio piece, it needs to be on a working host server.

For this assignment, these are your hosting options:

- **Heroku** for your site code. It has a free tier and is a popular hosting option that integrates nicely with github. It requires that at least one person register with heroku and install its CLI software on their computer. That person then uses a few command line instructions to copy software from github repo to the heroku servers.
- **Google Cloud Platform** for your database. You will set up a mySQL instance on a Compute Engine (a virtualized server). You should have enough credits for this not to be a problem.

Functionality

Your second assignment will replicate some of the functionality from assignment 1, but is **not** a single-page application; instead multiple pages in PHP are needed. Some of these pages also use JavaScript but others don't.

Visual Design: Your pages will be marked on a desktop computer, but your site needs to be usable at mobile sizes. As in assignment 1, I'll be viewing your site in Chrome at Mobile L and Laptop L sizes.

Header: Each page will have a header at the top that will contain some type of logo and a navigation menu. For smaller browser sizes, you will instead need a "hamburger" menu, that is, a responsive navigation bar. There are many examples online of the necessary CSS and JavaScript for this to work. If you make use of CSS+JavaScript you find online, please be sure to document this in the About page. The header/hamburger menu should have the following links/options:

- Home (not logged-in version)
- Home (logged-in version)
- About
- Browse
- Favorites (only available if user is logged in)
- Login/Logout. If user is not logged in, then the option should be **Login**; if user is already logged in, then option should be **Logout**.
- Registration (only available if user isn't logged in)

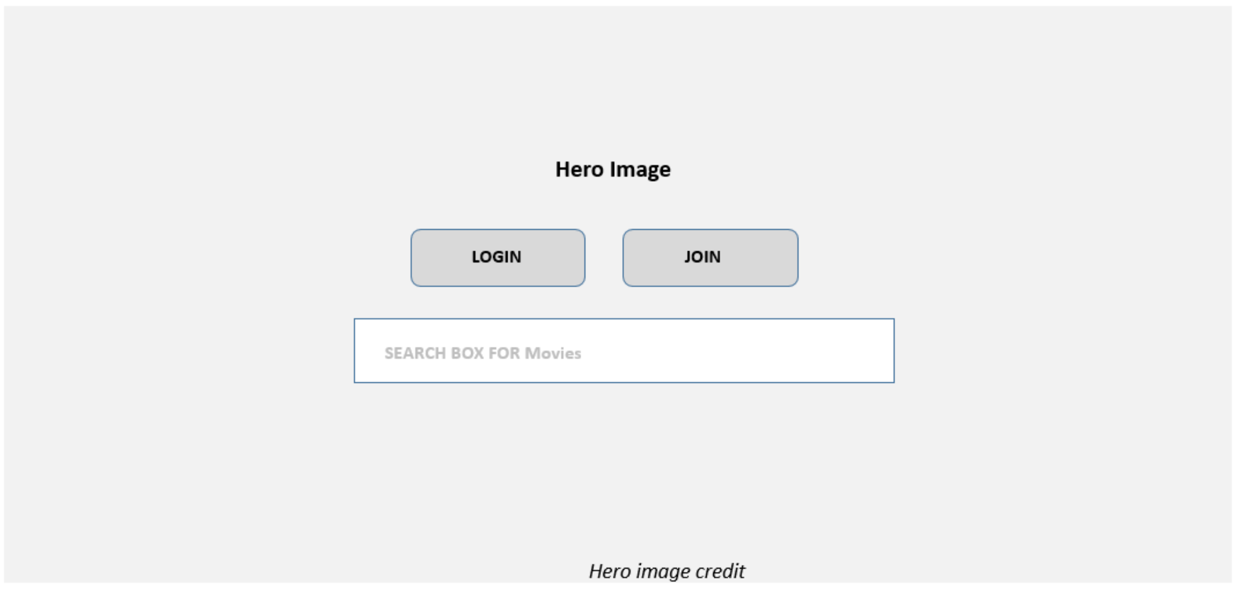
Note: the sketches in this assignment specification are meant to show functionality, not design. Here I've shown content as boxes, but you could do them as rectangles, circles, icons, simple links, etc. As with assignment 1, just following these rough sketches for your own site design will result in, at best, mediocre marks. **Make your pages look nicer than the sketches!**

Home Page (Not Logged-In Version)

The main page for the assignment. This file **must** be named `index.php`. This must have the functionality shown in the image below. Again, a reminder that you probably shouldn't make your home page **look** like this – it just has to have the shown *functionality*.

The search box will behave like it did for assignment 1, but the data for the title search will come from an API that you create yourself, instead of from the TMDB API.

The viewing of favorites has moved into the logged-in version of Home.



Home Page (Logged-In Version)

Display the user info (first name, last name, city, country). Also display movies they have put on their Favorite list (if none yet, be able to handle that).

Also display 10-15 recommended movies the user "may" like. How to do this? Ideally, if you had several additional months to work on this assignment, you would create a recommendation engine using some type of Machine Learning algorithm. But given the time constraints, your list will be based on movies that the user has already favorited: find 10-15 movies with the same release year as one or more of the favorited movies **and** with an average rating that is the same (within +/- 0.25) as the chosen movies as well.

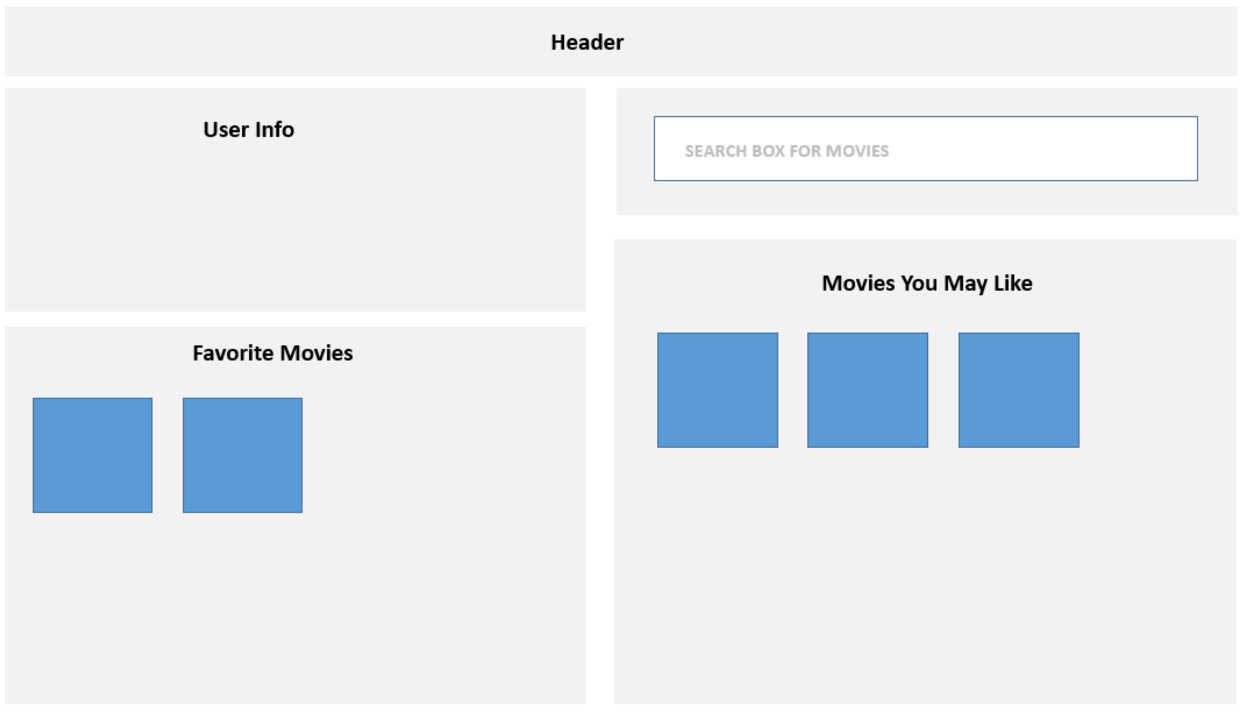
If your Movie You May Like "algorithm" has fewer than 10-15 images, fill it up with the most highly-rated movies – movies with the highest average rating.

It's possible that a user hasn't favorited *any* movies yet; in that case, show 10-15 of the most highly-rated moves.

Each of these movie poster images should be links to the `single-movie.php` page with the movie id in the query string.

The search box should behave as if the user wishes to perform a title search.

Important: unlike assignment one, where favorited movies were stored in local storage, in this assignment, things are different here. See the section "Favoriting" in the details for the Browse/Search page.



About Page

Provide the following information on that page in a nicely formatted way:

- university
- class name
- semester w/ year
- web technologies used
- team members with working links to their GitHub profiles
- working link to the assignment source code on GitHub

In addition, as you did for milestone 1 in assignment 1, display how many days, hours, and minutes are left until milestone 5 is due (it's due April 6 @ 9:00 PM). This is not a countdown timer - just how much time is left at the time this page is output by the server. PHP must be used to accomplish this. Make sure to check your work here – last semester, a surprising number of students slipped up here.

Browse Page

This page should be named `browse-movies.php`. It displays a list of movies with the same similar filter functionality as the first assignment. Most of the functionality on this page should all work via JavaScript: that is, you should be able to make use of most of your assignment 1 JavaScript code here.

You can use the same visual design as you used in assignment 1, or decide to strike off in a new direction. As long as the same functionality is present in the page, feel free to get creative!

As with the first assignment, movies should be sortable by title, year, and rating.

Clicking on the title or the poster image will take the user to the **Movie Details** page with the movie id in the query string (e.g., `single-movie.php?id=17`).

Favoriting

Things are different here than they were in assignment one.

First off, the favoriting feature must only be visible if the user is logged in.

Next, the actions taken when (un)favoriting occurs are quite different: we now need to redirect to a PHP page that adds/removes the movie to/from the session list, and then redirects back to the Browse page. This is old-fashioned and inefficient as it results in two redirects (two request/response cycles). It's good experience for you, though.

Favorites Page

Will display list of logged-in user's favorited movies (implemented via PHP sessions). If none yet, be able to handle that with a message. This page must be named `favorites.php`. The user should be able to remove movies singly or all at once from this list. This page will be entirely in PHP. The list should display a small version of the movie poster and its title. They both should be links to the appropriate single movie page.

Login/Logout Page

Login

If the user is logged out, display a login screen with email and password fields, plus a login button and a link or button for signing up if the user doesn't have an account yet. Some of the information below about hashing algorithms will be explained in class when we cover security.

Your database has a table named Users. It contains the following fields: `id, firstname, lastname, city, country, email, password, salt, password_sha256`.

The actual password for each user is **mypassword**, but that's not what is stored in the database. Instead, the actual password has been subjected to a bcrypt hash function with a cost value of 12. The resulting digest from that hash function is what has been stored in the password field. That means to check for a correct login, you will have to perform something equivalent to the following:

```
$digest = password_hash( $_POST['pass'], PASSWORD_BCRYPT, ['cost' => 12] );
if ($digest == $password_field_from_database_table && emails also match) {
   // we have a match, log the user in
}
```

For successful matches, you will need to preserve in PHP session state the log-in status of the user and the user's id. As well, after logging in, redirect to the **Home (Logged-In Version)** page.

Logout

If the user is logged in, then modify your session state information so your program knows a user is no longer logged in. After logging out, redirect to **Home (Logged-Out Version)** page.

Registration Page

This page displays a registration form. The content shown in the sketch below should be present, but you should present things much more nicely if you plan on having a decent mark for this portion of the assignment!

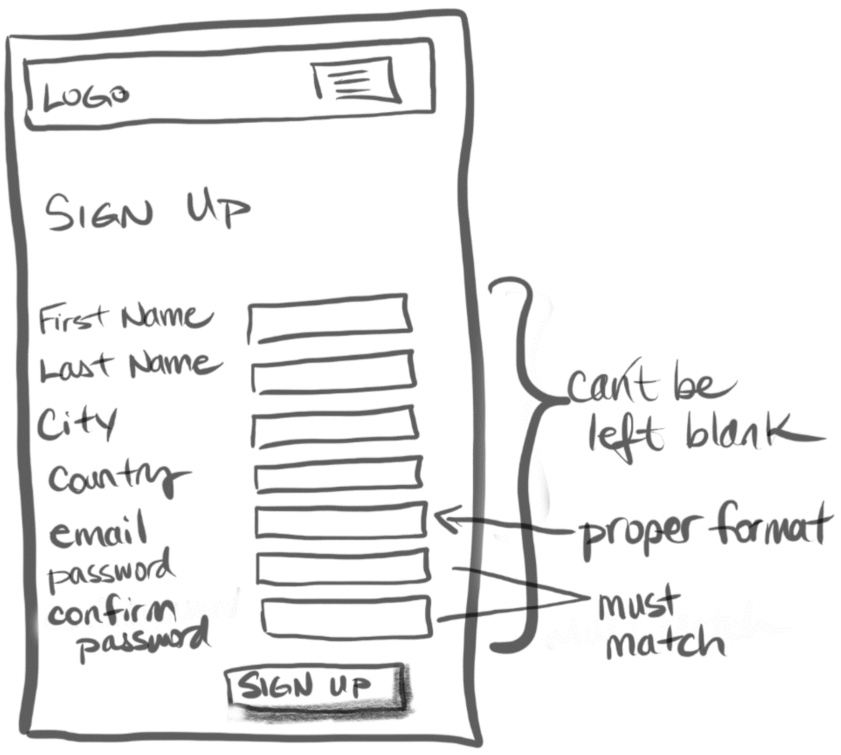
Perform the following client-side validation checks using JavaScript:

- first name, last name, city, and country can't be blank,
- email must be in proper format (use regular expressions), and
- the two passwords must match and be at least 8 characters long.

Be sure to display any error messages nicely and follow the many best practices shown in this article and this one as well. If you have the textbook, there is some very useful information in section 9.5 (Forms in JavaScript).

Once the user clicks Sign Up/Register button, you will have to check to ensure that the email doesn't already exist in the users table. If it does, return to the registration form (with the user's data still there) and display any appropriate error message. If the email is new, then add a new record to the Users table, log them in, and redirect to the **Home (Logged-In Version)** page.

You will **not** store the password as plain text in the database. Instead, you will save the bcrypt hash with a cost value of 12.



Movie Details Page

This page should be named `single-movie.php`. It should display the single movie indicated by the passed query string parameter. It should have the same functionality as the Movie Details View in the first assignment (but no speak or close buttons).

Again, you can use the same layout as in the first assignment, or go in a different direction.

In the first assignment, you fetched the data using JavaScript from an API. You won't be doing that here. In this assignment, you are going to program like it was still 2013: that is, your PHP page will retrieve the data from your database and then programmatically construct the page.

You will need to add favoriting functionality to this page, as you have in the Browse/Search page.