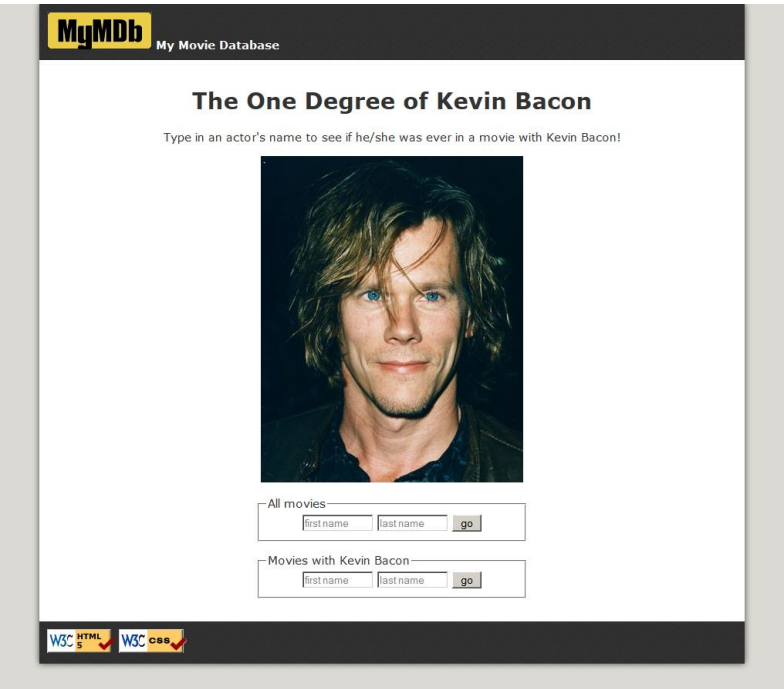
General tutorial instruction

- ✓ *This is the **SECOND** and also the **LAST** assignment of this course. It is worth 25% of your total mark.*
- ✓ *You should work **IN PAIR** on this one. (If you cannot find a partner, you must work **ALONE**, not a group of more than two members)*
- ✓ *Do not copy anything from your friends' work or available source code, not yours.*
- ✓ *If you have any question about the assignment, please post yours to the course forum.*

Assignment 2 PHP and SQL

Assignment Requirement

In this assignment, you will build following pages:



The Six Degrees of Kevin Bacon is a game based upon the theory that every actor can be connected to actor Kevin Bacon by a chain of movies no more than 6 in length. (Most, but not all, can reach him in 6 steps. 12% of all actors cannot reach him at all.)

Your task for this assignment is to write the HTML and PHP code for a web site called **MyMDb** that mimics part of the popular IMDb movie database site. Your site will show the movies in which another actor has appeared with Kevin Bacon. The site will also show a list of all movies in which the other actor has appeared. The front search page ~~**mymdb.php**~~ **index.php** has a form where the user can type an actor's name. When the form is submitted, the results are shown on **search.php** (as the right screenshot above).

You will submit the following files:

- **search-all.php**, the page showing search results for all films by a given actor.
- **search-kevin.php**, the page showing search results for all films with the given actor and Kevin Bacon.
- **bacon.css**, the CSS styles for both pages
- **common.php**, any common code that is shared between pages ("optional")

Since this is the last assignment, one of its themes is to tie together much of the material you have learned. You will write some HTML/CSS and JavaScript and a large amount of PHP and SQL code.

Appearance Constraints (both pages)

Your **index.php** and search pages must both match certain appearance criteria listed below. Beyond these, any other aspects of the page are up to you, so long as it does not conflict with what is required.

- All pages must begin with the content from top.html and end with the content from bottom.html, including the "favicon", MyMDb logo, W3C validator button links, and the two forms to search for movies. It is especially important not to modify the provided form query parameters' names such as first_name.
- The main section of content must be a centered area that is narrower than the overall page body. This main section should have a different background color than the overall body behind it, to make it stand out.
- Every page should have a descriptive level-1 heading explaining the contents of the page.
- The top and bottom banner areas containing the MyMDb logo and W3C images should have a common color scheme and/or background image that make them stand out from other content on the page.
- The site should have a consistent color and font scheme used throughout. Your CSS should be structured such that it is easy to change the color/font scheme by modifying colors and fonts in a single place in the file.
- All content should have reasonable sizing, padding, and margins such that content does not awkwardly bump into other content on the page. Content should also be aligned in reasonable ways for easy viewing.
- Any query results should be shown in tables with captions describing the tables, and headings describing each column. The rows of the table should alternate in background color, also called "zebra striping." Borders should be collapsed.

Front Page, index.php

The initial page, **index.php**, allows the user to search for actors to match against Kevin Bacon. This file is already provided to you on the course web site; you may modify it in any way you like, or you may leave it as-is. If you modify it, turn in your modified version. The forms on the page contain two text boxes that allow the user to search for an actor by first/last name.

- firstname: for the actor’s firstname
- lastname: for the actor’s lastname

Movie Search Page, search-all.php and search-kevin.php

The two search pages perform queries on the imdb database to show a given actor's movies. Query the database using PHP's PDO library as taught in class. Connect to the database using your ID and the MySQL password.

The database has the following relevant tables. (The roles table connects actors to movies.)

Table	Columns
actors	id, first_name, last_name, gender
movies	id, name, year
roles	actor_id, movie_id, role

Your page should perform the following two queries. For each query, you will need to use a join between several tables of the database.

1. **search-all.php** – **A query is to find a complete list of movies** in which the actor has performed, displaying them in an HTML table. If the actor doesn't exist in the database, don't show a table, and instead show a message such as, "Actor Borat Sagdiyev not found." If the actor is found in the database, you may assume that any actor in the actors table has been in at least one movie.
   Hint: You will need to join the actors, movies, and roles tables, and only retain rows where the various IDs from the tables (actor ID, movie ID) match each other and the name or ID of your actor. Our solution's query joins 3 tables in the FROM clause and contains one condition in its WHERE clause.
2. **search-kevin.php** – **A query to find all movies in which the actor performed** with Kevin Bacon. These movies should be displayed as a second HTML table, with the same styling as the first. This is the hard query and should be done last. If the actor has not been in any movies with Kevin Bacon, don't show a table, and instead show a message such as, "Borat Sagdiyev wasn't in any films with Kevin Bacon."

   This query is bigger and tougher because you must link a pair of performances, one by the submitted actor and one by Kevin Bacon that occurred in the same film. Do not directly write any actor's ID number anywhere in your PHP code, not even Kevin Bacon's.

   Hint: You will need to join a pair of actors (yours and Kevin Bacon), a corresponding pair of roles that match those actors, and a related movie that matches those two roles. Our query joins 5 tables in the FROM clause and contains 3 conditions in its WHERE clause.

The data in both tables should be sorted in descending order by year, breaking ties by movie title. The tables have three columns: A number for each movie, starting at 1; the title; and the year. The columns must have styled headings, such as bolded. The rows of the table must have alternating background colors, sometimes called "zebra striping." (Use PHP to apply styles to alternating rows.) For example:

No.	Title	Year
1	Private War, A	2005
2	Reefer Madness	2004
3	Blind Horizon	2004

Databases

MySQL (or MariaDB) is required for this assignment. Thus, make sure that your install the DBMS or the whole web development package XAMPP or LAMP in your computer.

Next, we need to download the table information and then put it into MySQL by creating a new database and adding the tables to it:

- Download imdb.sql (main one) imdb_small.sql (one for testing purpose)
- Use phpmyadmin to create 2 two databases imdb_small and imdb; then import two given sql files

Or

- Use the command line to create databases and import the data with following instruction:

```
CREATE DATABASE imdb_small;
use imdb_small;
SOURCE imdb_small.sql;

CREATE DATABASE imdb;
use imdb;
SOURCE imdb.sql;  # this will take a while
```

Developing Your Program

Because the database is large, you may not want to test on the big one first. The smaller database **imdb_small** with fewer records will allow for more efficient testing. When you think your code works, switch your PHP and/or JavaScript code to refer to **imdb**.

Use the MySQL console / or phpmyadmin console to develop your queries before writing PHP SQL code. Example tests are *376249* (Brad Pitt) or *770247* (Julia Roberts). If your query takes too long, press Ctrl-C to abort it / or close the web-based window of phpmyadmin.

Enable exceptions on your PDO object to spot mistakes, as shown in the slides. Print out your SQL queries while debugging, so you can see the actual query your code is making. Many PHP SQL bugs come from improper SQL query syntax, such as missing quotes, improperly inserted variables, etc.

Your code should follow style guidelines similar to those on our past homework specs. Minimize redundant HTML/PHP code. Use functions and include files to break up code. Avoid global variables.

After you finished your pages and before submission, don't forget to change the database from "**imdb_small**" (for test) to "**imdb**". To check your work, you can search "*Kevin Spacey*" in your page. The output should be: 75 movies by Kevin Spacey, and 2 movies by Kevin Spacey and Kevin Bacon together.

Implementation and Grading

- ✓ Your HTML (including PHP output) should pass the W3C HTML validator. Your CSS code should pass the W3C CSS validator and should avoid redundant or poorly written rules.
- ✓ Your code should follow style guidelines similar to those on our past assignment specs. Avoid global variables, and use descriptive names. Place descriptive comments at the top of each file, each function, and on complex code. Also place a comment next to every single SQL query you perform that explains what that query is searching for. Use parameters and return values properly. Show proper separation of content, presentation, and behavior between HTML, CSS, and PHP.

- ✓ With so much in common between the two search pages, it is important to find ways to avoid redundancy. You should use the PHP include function with shared common content included by various pages. Also use functions as appropriate to capture structure and repeated code and HTML content.
- ✓ For full credit, do not directly write any actor's ID number anywhere in your PHP code, not even Kevin Bacon's. For example, don't write a line like:

```
$bacon_id = 22591;     # Kevin Bacon's actor id (BAD, don't do this)
```

- ✓ Properly use whitespace and indentation. Use good variable and method names. Avoid lines of code more than 100 characters wide.
- ✓ Please do not place a solution to this assignment online on a publicly accessible (un-passworded) web site.

It is **not acceptable** to perform query filtering in PHP. Your SQL queries must filter the data down to only the relevant rows and columns. For example, a bad algorithm would be to query all of the actor's movies, then query all of Bacon's movies, then use PHP to loop over the two looking for matches. Each of the two queries above should be done with a single SQL query to the database.

Submission

You must submit a single zip file containing all required files to the portal by the due date. The zip file name must be of the form student1id-student2id-wpg2017-2.zip, where student-id is your student identifier (the remaining bits of the file name must not be changed!). For example, if your student ids are 0912345678 and 0912345679 then your zip file must be named **0912345678-0912345679-wpg2017-2.zip**. Only one of you should submit your assignment.

IMPORTANT: Failure to name the file as shown will result in no marks being given!