CSCI 5060 Assignment 6

Total estimated time for this assignment: **6 hours** (if you are a good programmer)

The main programming language for this assignment should be **PHP** (mixed with HTML and CSS). If you do not use **PHP**, you will get **0 points**. For this assignment, you must use **PDO** (PHP Data Object) to interface with the MySQL database. If you do not use PDO, you will receive 0 points for almost all questions.

When you see "Richard Ricardo" or "richardricardo" in the examples and screen captures, change it to **<your name>**.
When you see "Richard" or "richard" in the example screen captures, change it to **<your first name>**.
If you do not put **<your name> / <your first name>** in the above mentioned fields, you will get **0 points** for the question(s).

No two students should submit webpages with exactly the same code, content, layout, or color combination. If found, **both** students will get **0 points**.

Please change the provided example's layout and color combination (color scheme). **If you use (copy) ANY of the provide example's color combination, 10 points will be deducted (-10 points).**

Create a folder on your hard disk, name the folder **lastname_firstname_assignment6.** Save all the files from this assignment in this folder.

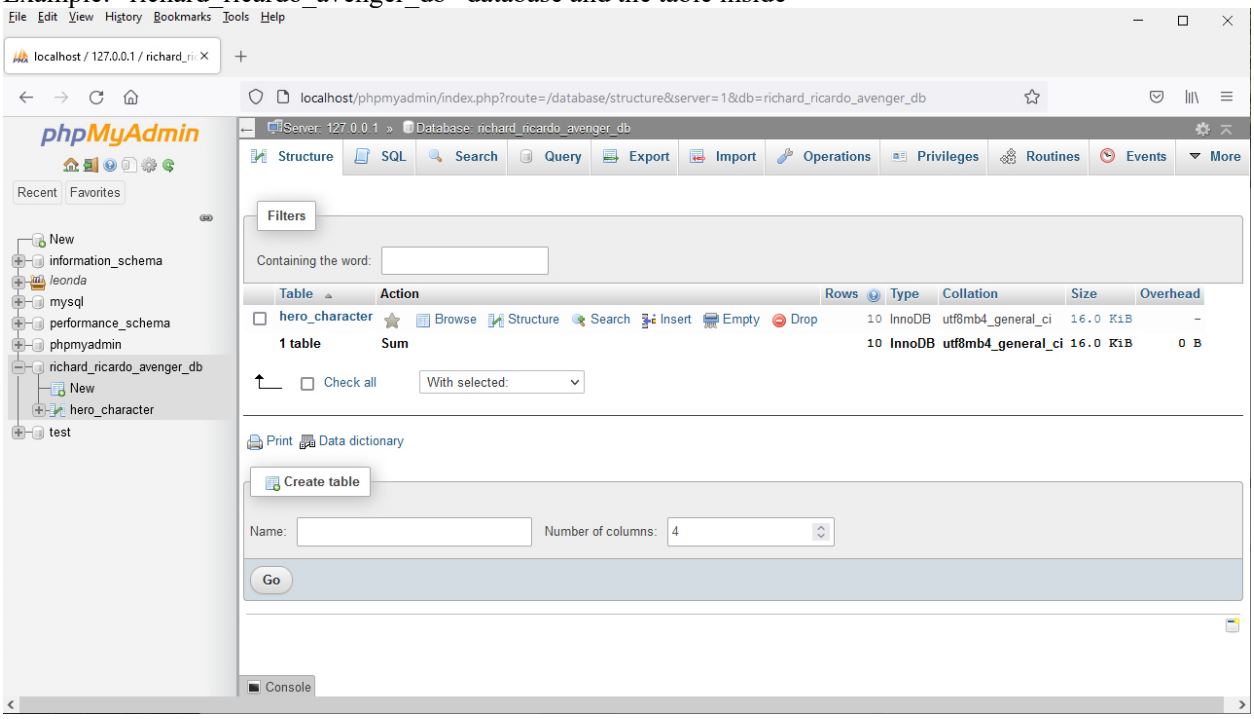
Use XAMPP web server solution stack package to help debugging PHP code. It will make your debugging process easier. All php files must not produce any error, or any warning **(-2 points for each error, each warning**). Your program must run. A program that does not run will get at most **50% of the possible points**. All files must begin with a **comment** that identifies the author, the course code, and the program date **(- 2 points each question** if found missing). All html, css and php files must be clearly **documented (commented)**. Points will be taken off **(-2 points each question)** for insufficient comments (`<!-- -->, /* */, //`).

- When you view page source in a web browser, **`<!DOCTYPE html>`** must be at the top of every page. In other words, all pages must be written in HTML5. **(-20 points** if not)
  - You **can** put php code before **`<!DOCTYPE html>`**.
  - You **cannot** put html code before **`<!DOCTYPE html>`**.
- Before adding PHP code, all html files must pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning).
- After adding PHP code, the generated html code (Firefox web browser > right-click > view page source) must also pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning).
- All css files must pass css validation at http://jigsaw.w3.org/css-validator/ without any **error**. **(-2 points for each error/warning**, only 1 warning is allowed for html validator)

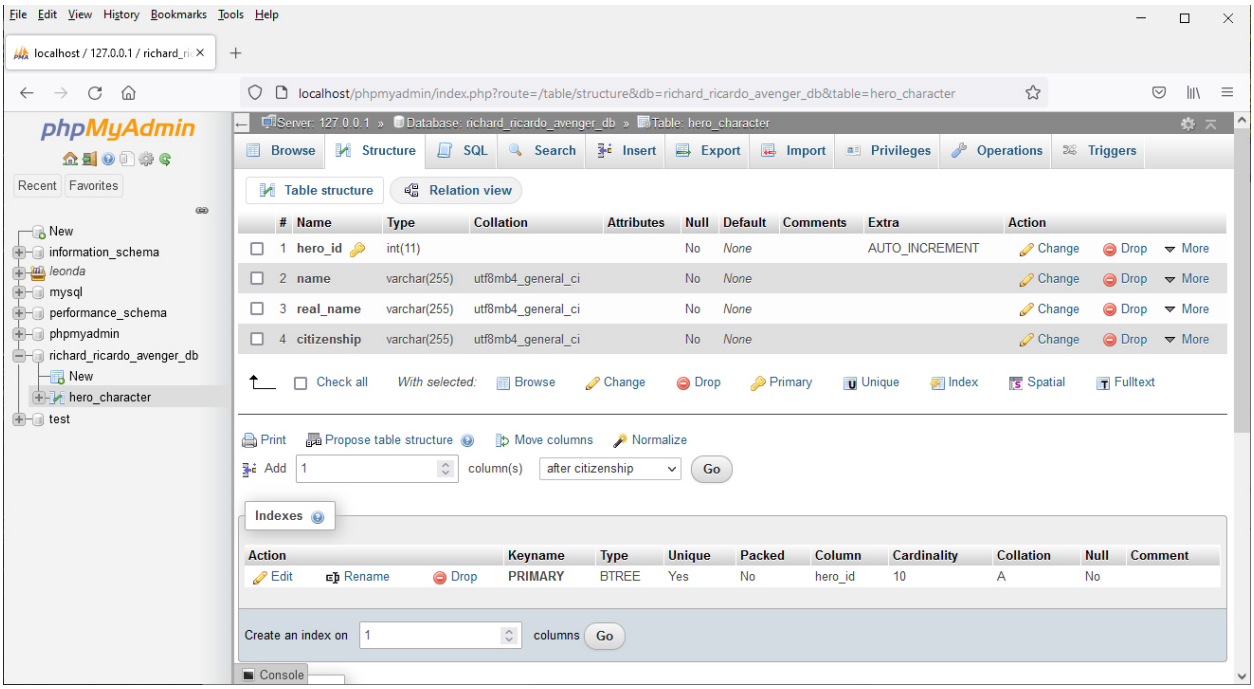
Question 1 – **Database**: PHP Chapter 4, eg008 and knowledge of SQL (10 points) **Estimated time: 1 hour**

- You created part the requested sql file in Assignment 3, Q1. You can copy create_db.sql from Assignment 3 and update the sql file, instead of creating a new file.
- Save question 1 files in folder "**lastname_firstname_assignment6**": (1 point)
  - **create_db.sql**
- Create a text file **create_db.sql,** write sql statements in the file to
  - Create a MySQL database **richard_ricardo_avenger_db**. (1 point)
  - In the database, create 1 table.
    - hero_character (1 point)
  - Create the following fields (columns) for the table (refer to examples below for details).
    - hero_character table: hero_id, name, real_name, citizenship (3 points)
    - hero_id is the primary key of the character table (1 point)
  - Insert test records to the hero_character table. (3 points)
  - Create a MySQL database username **richarduser** with password **richardpowerability**, with data privileges (select, insert, update, delete) for the **richard_ricardo_avenger_db** database. (1 point)
  - All above must be done by SQL statements in the text file create_db.sql. (0 points if not)
- Load **create_db.sql** in XAMPP > phpMyAdmin to create the above mentioned database.
- Note: In the real world, do NOT put sql files in a website folder. Keep it offline and safe.

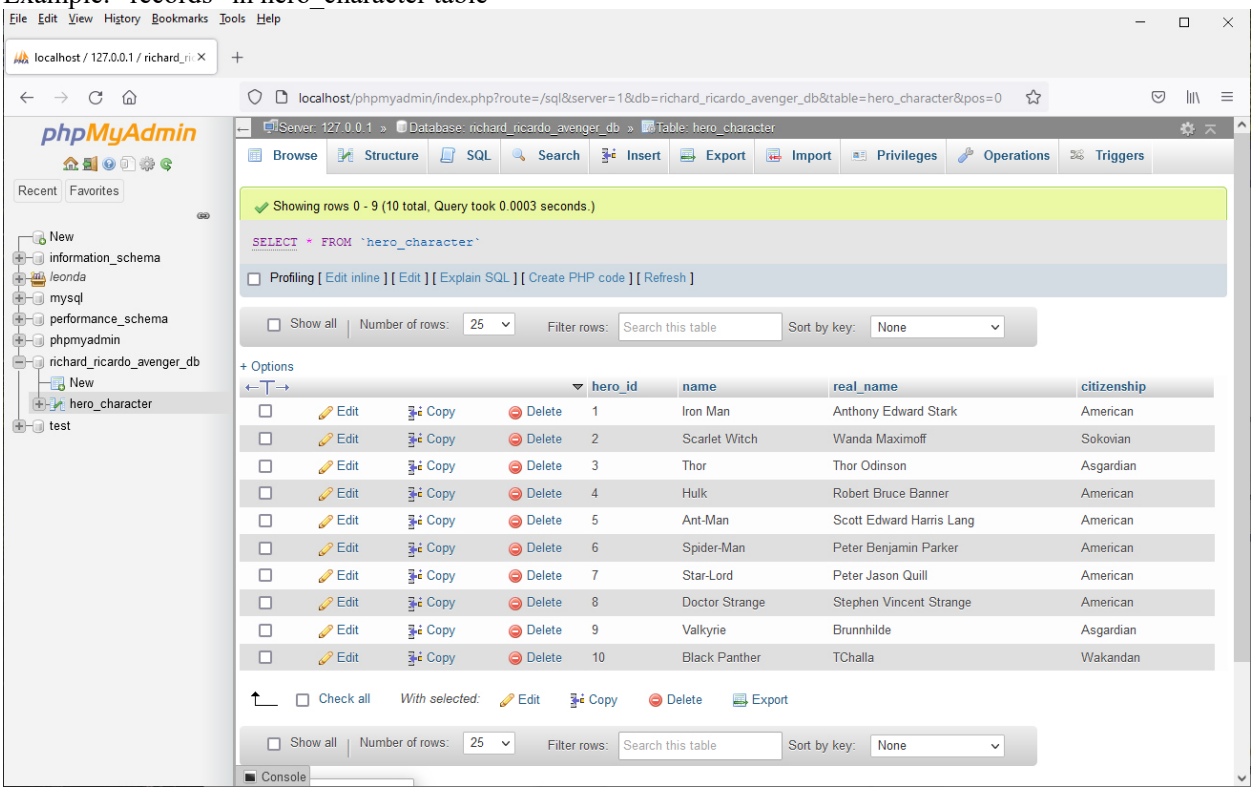
Example: "richard_ricardo_avenger_db" database and the table inside



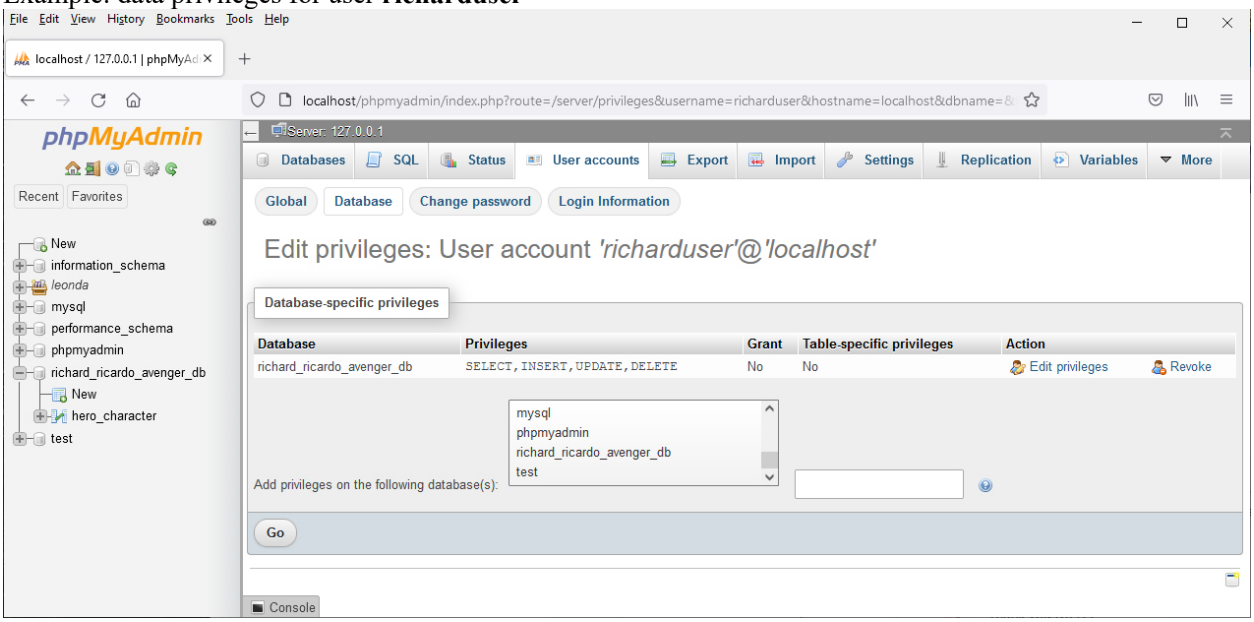
Example: “hero_character” table structure



Example: “records” in hero_character table



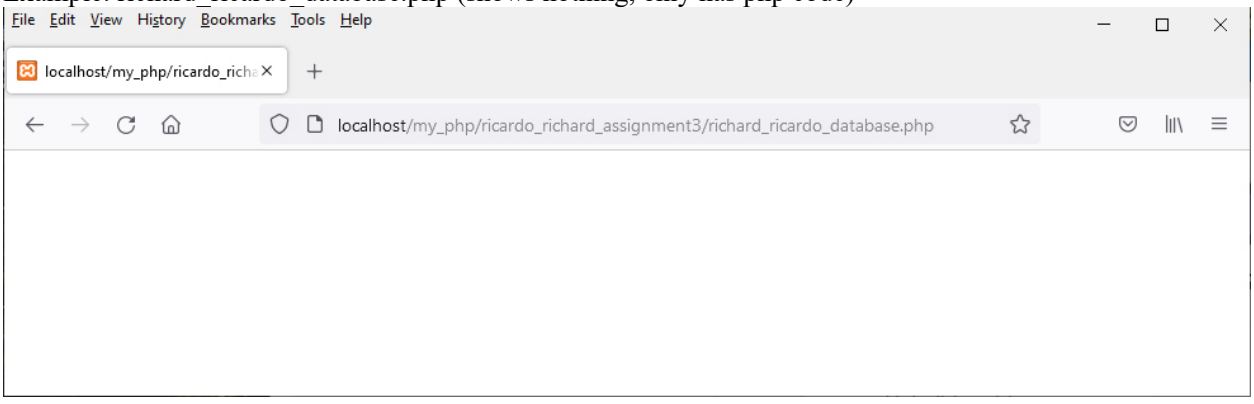
Example: data privileges for user **richarduser**



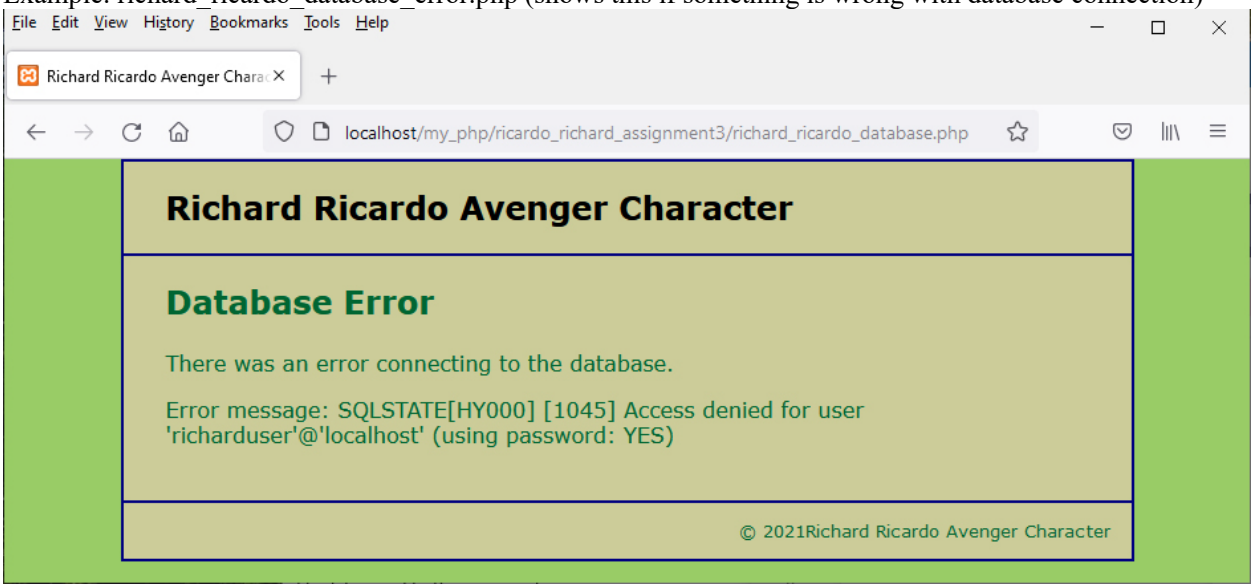
Question 2 – **Connect**: PHP Chapter 4, eg008/9 and knowledge of SQL (5 points) **Estimated time: 0 hours**

- You can copy your files from Assignment 3. Technically you do not need to do anything for this question if you finished Assignment 3.
- Save question 2 files in "**lastname_firstname_assignment6**": (5 points, -5 points if there is any error)
  - **main.css**
  - **richard_ricardo_database.php**
  - **richard_ricardo_database_error.php**
- Create web page(s) that displays the default information (example shown).
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "Richard Ricardo Avenger Character**"** as the page title(s).
- Create file richard_ricardo_database.php (php code only, no html code) to connect to database
  - Use PDO (PHP Data Object) to interface with your MySQL database.
  - Connect to the **richard_ricardo_avenger_db** MySQL database (created in Q1), using username **richarduser** with password **richardpowerability**.
  - Use try {} catch () {} exception handling to detect if the connection is successful.
  - If the connection is not successful, show an error page **richard_ricardo_database_error.php**.
- Create a css file named **main.css** to format all pages by creating your own layout (no two students should have the same layout). You should use **the same css file** to format all questions.

Example: richard_ricardo_database.php (shows nothing, only has php code)



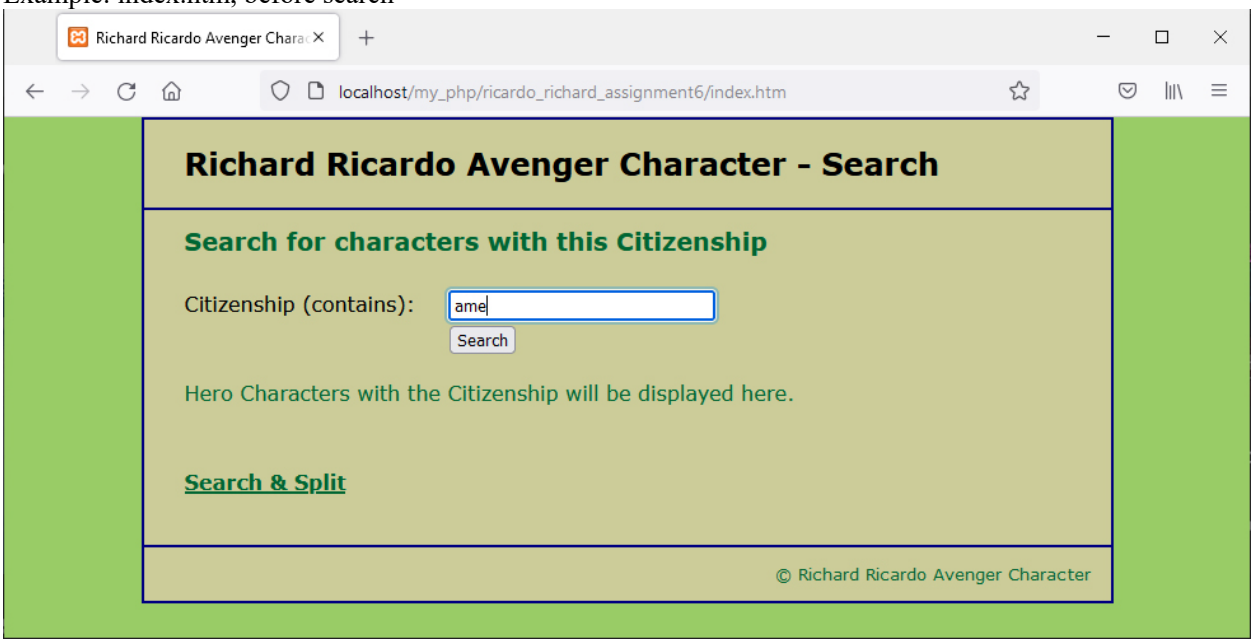
Example: richard_ricardo_database_error.php (shows this if something is wrong with database connection)



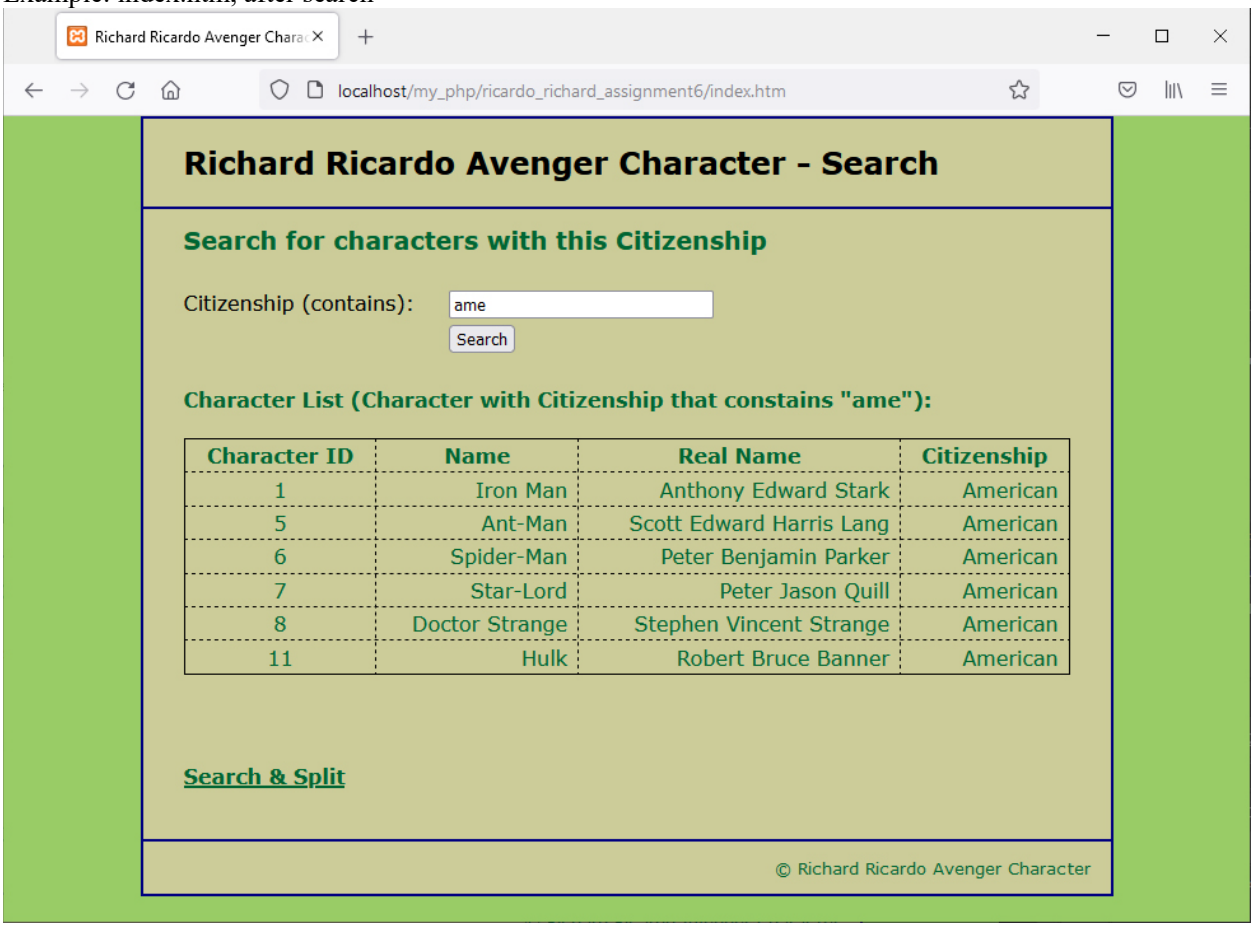
Question 3 – **AJAX Search**: PHP Chapter 4, eg008/9, SQL & AJAX (40 points) **Estimated time: 2 hours**

- Save question 3 files in folder "**lastname_firstname_assignment6**". (2 points)
  - **index.htm**
  - **richardricardo_search.php**
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "Richard Ricardo Avenger Character**"** as the page title(s) (<title> tag). (2 points)
- The index.htm web page should (36 points, no points if not using ajax)
  - Allow user to enter (part of) a Citizenship.
  - Have a search button.
  - Display search results without re-loading the page (use ajax instead).
  - Create an `XMLHttpRequest` object in a JavaScript function, which is triggered by user clicking the button.
  - The `XMLHttpRequest` object uses **richardricardo_search.php** to search the database.
  - **richardricardo_search.php** retrieves all the data from the `hero_character` database table, with `citizenship` value that contains the user input.
  - Use html table (or css) to format the output table (into rows and columns).
  - The `XMLHttpRequest` object displays all retrieved data (without reloading **index.htm**).

Example: index.htm, before search



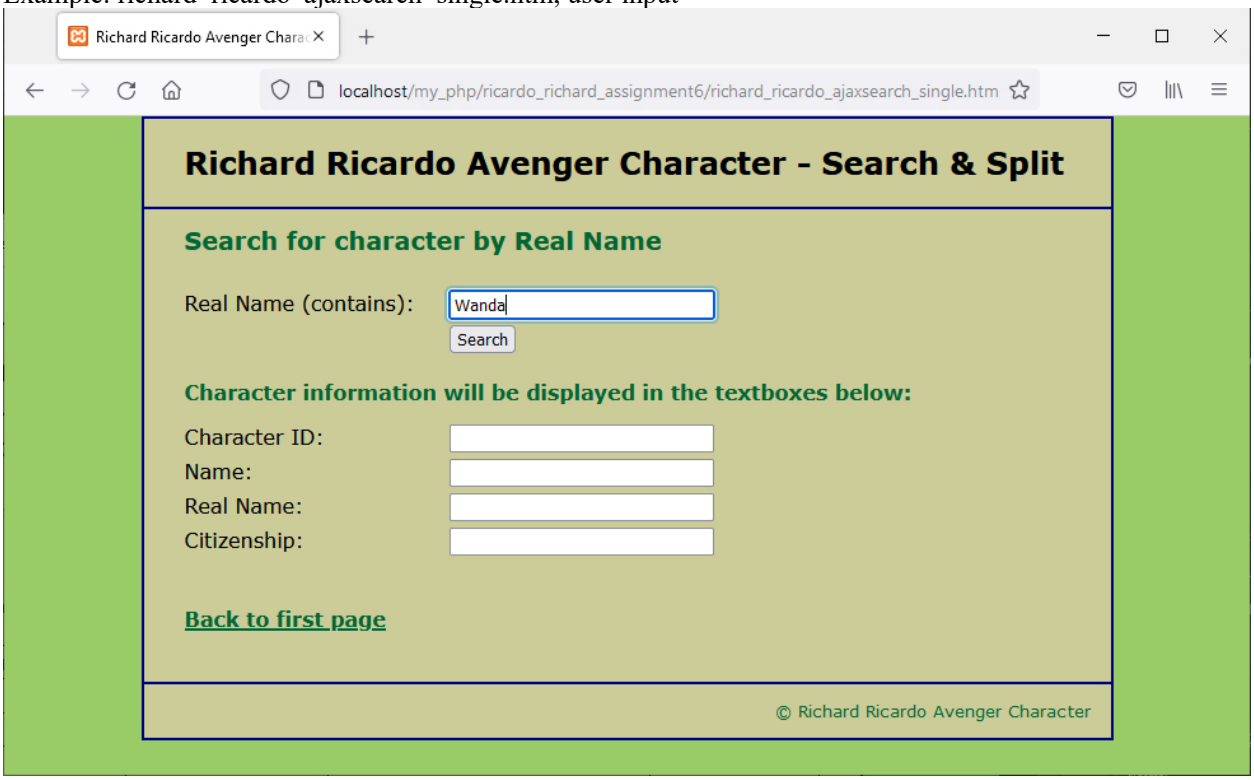
Example: index.htm, after search



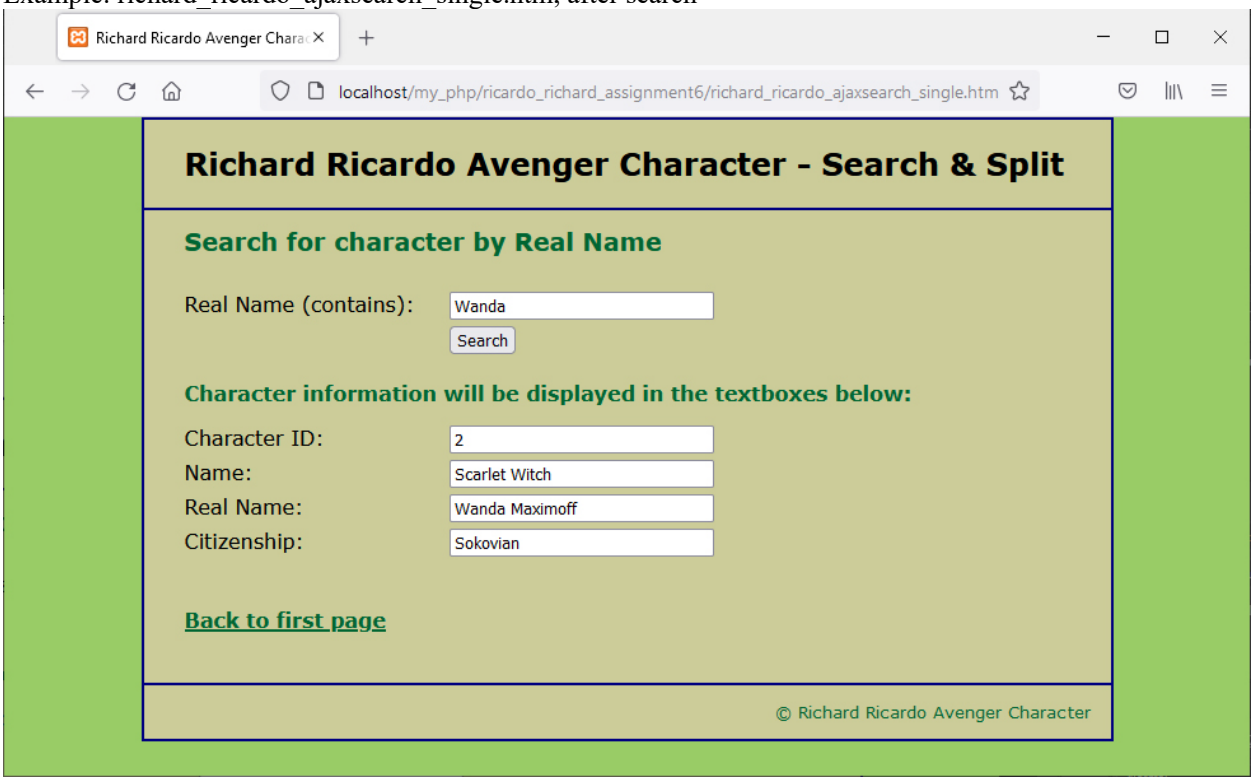
Question 4 – **AJAX Search Split**: PHP Ch 4, eg008/9, SQL, AJAX (45 points) **Estimated time: 3 hours**

- Save question 4 files in folder "**lastname_firstname_assignment6**". (2 points)
  - **richard_ricardo_ajaxsearch_single.htm**
  - **richardricardo_search_single.php**
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "Richard Ricardo Avenger Character**"** as the page title(s) (<title> tag). (2 points)
- When a user click on "Search & Split" link on index.htm, the user should be directed to richard_ricardo_ajaxsearch_single.htm. (2 points)
- The richard_ricardo_ajaxsearch_single.htm web page should (39 points, no points if not using ajax)
  - Allow user to enter part of a `real_name`.
  - Have a search button.
  - Display search results without re-loading the page (use ajax instead).
  - Have four **read-only** textboxes for output.
  - Create an `XMLHttpRequest` object in a JavaScript function, which is triggered by user clicking the button.
  - The `XMLHttpRequest` object uses **richardricardo_search_single.php** to search the database.
  - **richardricardo_search_single.php** retrieves a row from the `hero_character` database table, with the `real_name` value contains the user input.
  - The `XMLHttpRequest` object uses the four textboxes to display the retrieved data. Do not recreate the four textboxes. Put data into the existing textboxes. (-10 points if recreating textboxes)
  - You were not taught how to split retrieved data. You need to do some research on this topic.
    - Hint: ajaxRequest.responseText.split("").

Example: richard_ricardo_ajaxsearch_single.htm, user input



Example: richard_ricardo_ajaxsearch_single.htm, after search



Important:

1. If you do not put **<your name> / <your first name>** in the above mentioned fields (as shown in the examples), you will get **0 points** for the question(s).
2. **No two students** should submit webpages with exactly the same code, content, layout, or color combination. If found, both students will get **0 points**.
3. When you view page source in a web browser, **`<!DOCTYPE html>`** must be at the top of every page. In other words, all pages must be written in HTML5. **(-20 points** if not)
   - You **can** put php code before **`<!DOCTYPE html>`**.
   - You **cannot** put html code before **`<!DOCTYPE html>`**.
4. Before adding PHP code, all html files must pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning).
5. After adding PHP code, the generated html code (Firefox web browser > right-click > view page source) must also pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning).
6. All css files must pass css validation at http://jigsaw.w3.org/css-validator/ without any **error/warning**.
7. If your files do not pass the html and css validations, **2 points will be deducted** for **each html or css error/warning** found (1 warning allowed for html validator).
8. Document (comment) your HTML files (`<!-- -->`), CSS files (`/* */`), and PHP files (`/* */` OR `//`). **Points will be taken off** for insufficient comments (`<!-- -->, /* */, //`).

Submission instructions:

- You need to test all document(s).
- Do screen capture(s) of the **input** and the related **output(s)**. Use any graphic editing software (e.g. Microsoft Paint, Adobe Photoshop, or GIMP etc.) to cut out the browser output (from the screen capture), paste them into a word document.
- Provide **2 different test cases** for each question. In other words, for **each question**, you may need to have **2 input** screen captures and **2 related output** screen captures.
- Do NOT need to do screen capture(s) of html validation results and css validation results for this assignment.
- Save the word document as a pdf file.

You need to submit the following:
1. A pdf file containing the screen capture(s) of the web browser input and output pages, name the file **lastname_firstname_assignment6.pdf**.
2. All html file(s), php file(s), css file(s), and other related files (e.g. image files). Zip your file folder (**lastname_firstname_assignment6**) into a single zip file (or rar file) **lastname_firstname_assignment6.zip**. In the above example, the zip file should contain the following files and subfolders. If there is any image, there should be a \images\ subfolder.

- lastname_firstname_assignment6\create_db.sql
- lastname_firstname_assignment6\index.htm
- lastname_firstname_assignment6\main.css
- lastname_firstname_assignment6\richard_ricardo_ajaxsearch_single.htm
- lastname_firstname_assignment6\richard_ricardo_database.php
- lastname_firstname_assignment6\richard_ricardo_database_error.php
- lastname_firstname_assignment6\richardricardo_search.php
- lastname_firstname_assignment6\richardricardo_search_single.php

Please submit the above mentioned **two files** (.pdf and .zip) to D2L digital dropbox.

Grading guidelines (programming questions):
Your programs will be judged on several criteria, which are shown below.

- Correctness (50%): Does the program compile (run) correctly? Does the program do what it's supposed to do?
- Design (20%): Are operations broken down in a reasonable way (e.g. classes and methods)?
- Style (10%): Is the program **indented** properly? Do variables have **meaningful names**?
- Robustness (10%): Does the program handle erroneous or unexpected input gracefully?
- Documentation (10%): Do all program files begin with a **comment** that identifies the author, the course code, and the program date? Are all the classes, methods and data fields clearly **documented (commented)?** Are unclear parts of code **documented (commented)?** (Some items mentioned may not apply to some languages)

A program that does not compile (run) will get at most **50% of the possible points**.