CSCI 4000 Assignment 4

Total estimated time for this assignment: **12 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_assignment4.** 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 (`<!-- -->`, `/* */`, `//`).

- Before adding PHP code, all html files must pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning).
- 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>`**.
- 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)

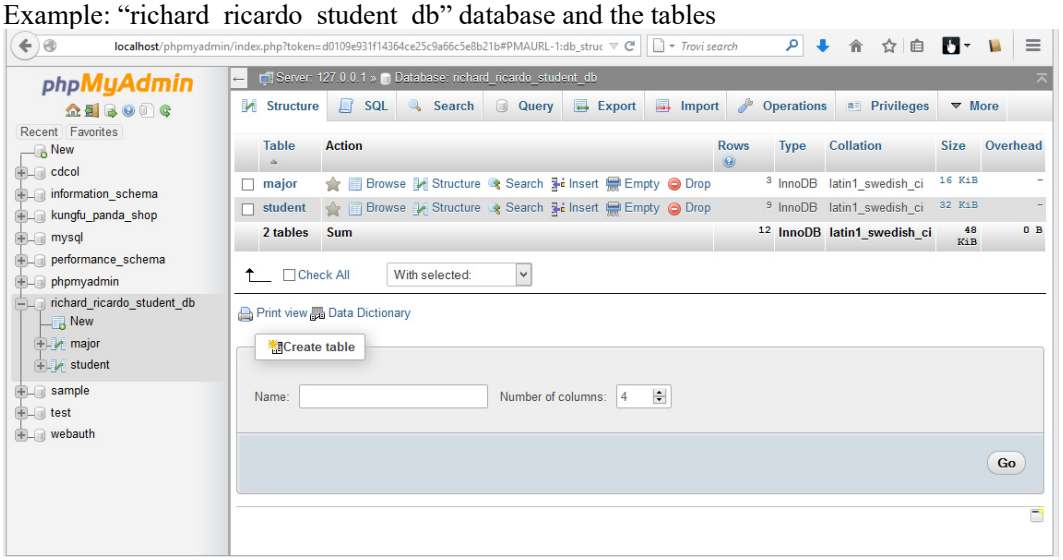
Question 1 – **Database**: PHP Chapter 4, eg009 and knowledge of SQL (20 points) **Estimated time: 3 hours**

- Save question 1 files in folder "**lastname_firstname_assignment4**": (2 points)
  - **create_db.sql**
- Create a text file **create_db.sql,** write sql statements in the file to
  - Create a MySQL database **richard_ricardo_student_db**. (2 points)
  - In the database, create 2 tables. (2 points)
    - major
    - student

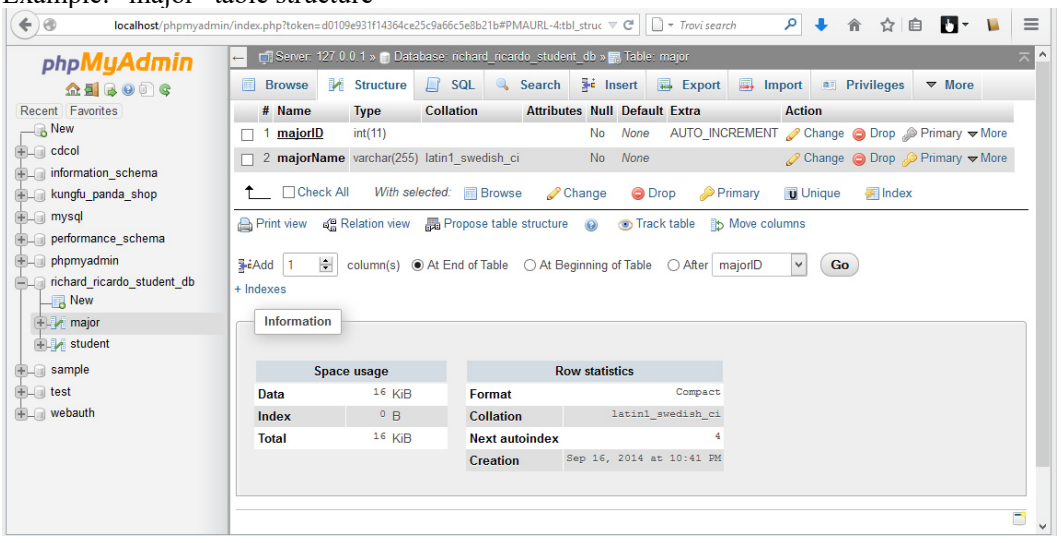
- Create the following fields (columns) for the tables (refer to examples below for details).
  - major table: majorID, majorName (2 points)
  - student table: studentID, majorID, firstName, lastName, gender (2 points)
- Do foreign key referencing, to maintain referential integrity (3 points)
  - One student can only have one major.
  - One major can be used (enrolled) by many students.
  - majorID is the primary key of the major table.
  - studentID is the primary key of the student table.
  - majorID is a foreign key of the student table, for any values of majorID appearing in the student table, these values must also appear in the major table.
- Insert test records to major table and student table. (3 points)
- Create a MySQL database username **richardricardo1** with password **richardisgreat**, with data privileges (select, insert, update, delete) for the **richard_ricardo_student_db** database. (4 points)
- 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.

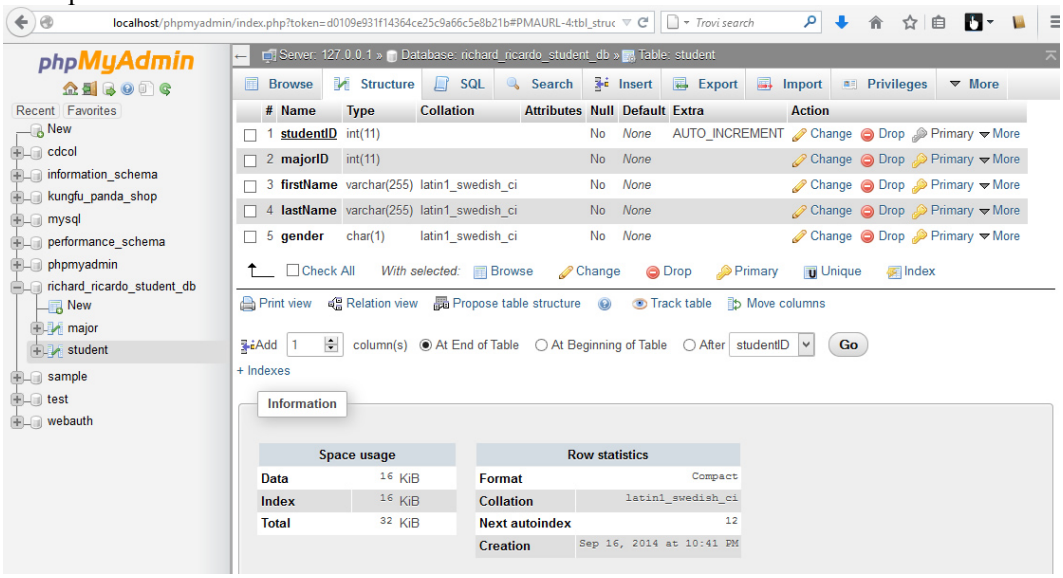
Example: "richard ricardo student db" database and the tables



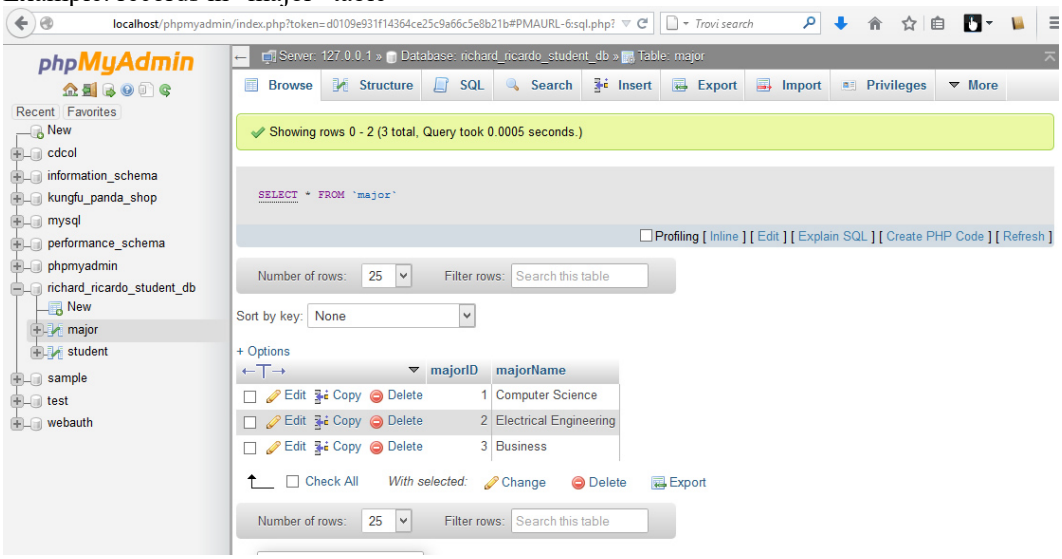
Example: "major" table structure



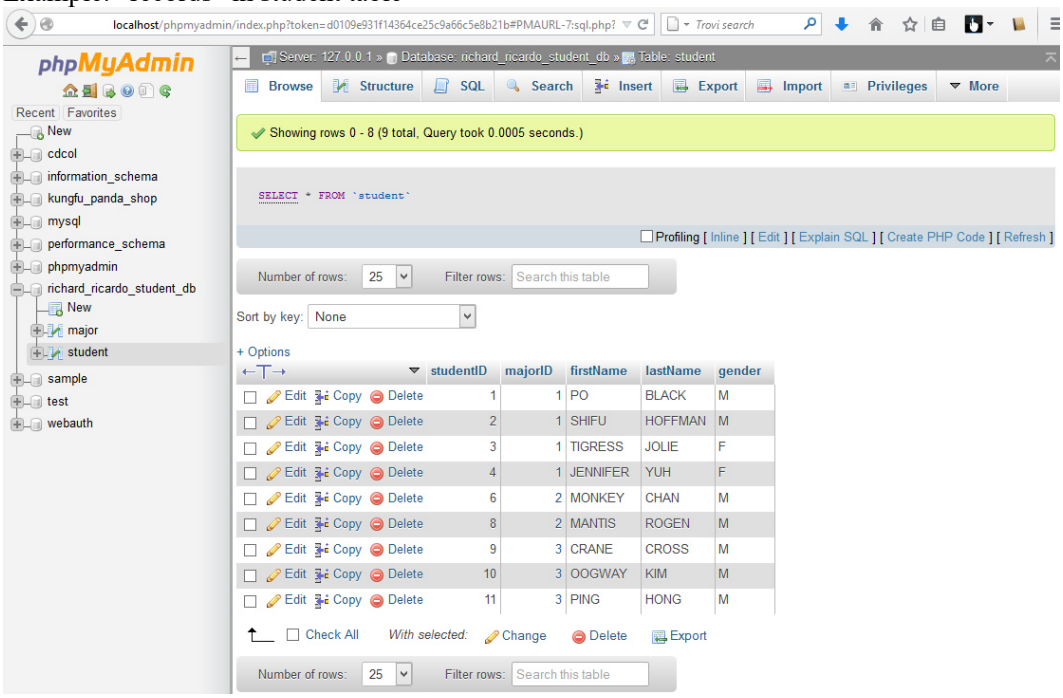
Example: “student” table structure



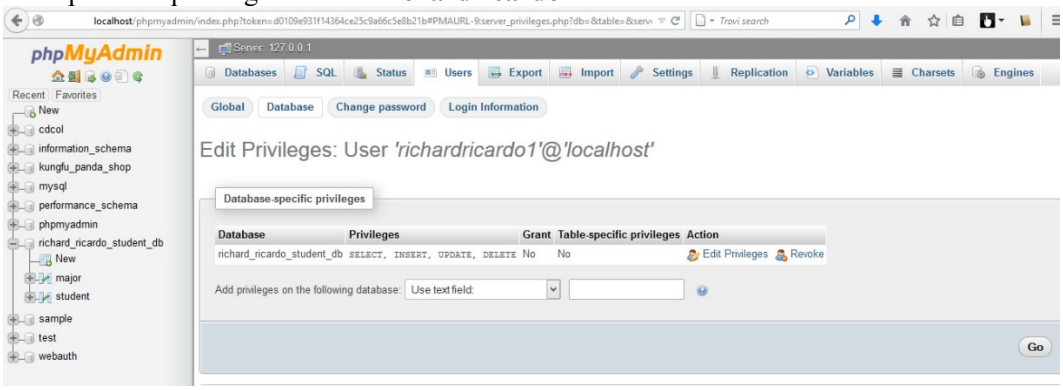
Example: records in “major” table



Example: “records” in student table



Example: data privileges for user **richardricardo1**

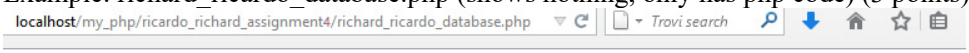


Question 2 – **Display**: PHP Chapter 4, eg009 and knowledge of SQL (35 points) **Estimated time: 4 hours**

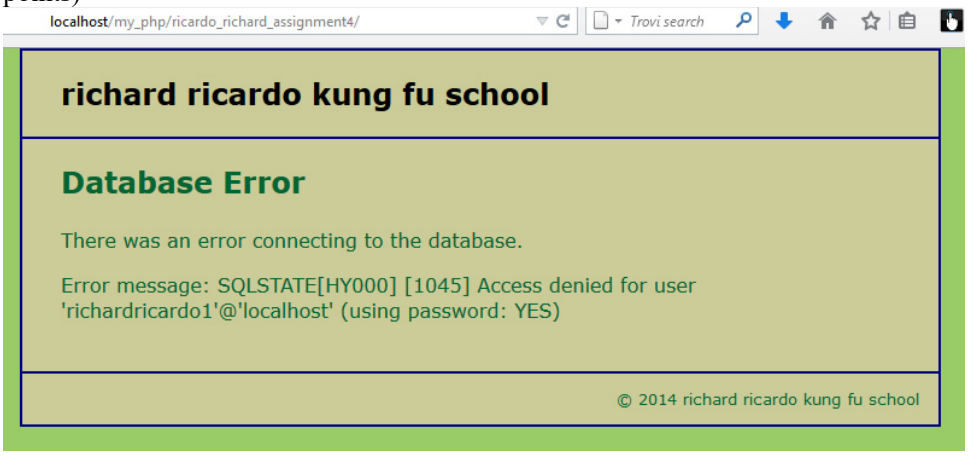
- Save question 2 files in “**lastname_firstname_assignment4**”: (4 points)
  - **index.php**
  - **main.css**
  - **richard_ricardo_database.php**
  - **richard_ricardo_database_error.php**
- Create web page(s) that displays the default information (example shown).
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using “richard ricardo’s kung fu school” as the page title(s) (<title> tag). (2 points)
- Save the first page as index.php. (1 point)

- 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_student_db** MySQL database (created in Q1), using username **richardricardo1** with password **richardisgreat**.
  - 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. (6 points)
- index.php
  - Uses **richard_ricardo_database.php** to connect to the database.
  - The sidebar on the left displays a list of majors from the "major" table (created in Q1).
  - The main portion of the page lists the students from the "student" table (created in Q1), for that major.
  - If a user wants to view the students in another major, the user can click on the link for that major.
  - The display on the main portion of the page changes accordingly as shown in examples.

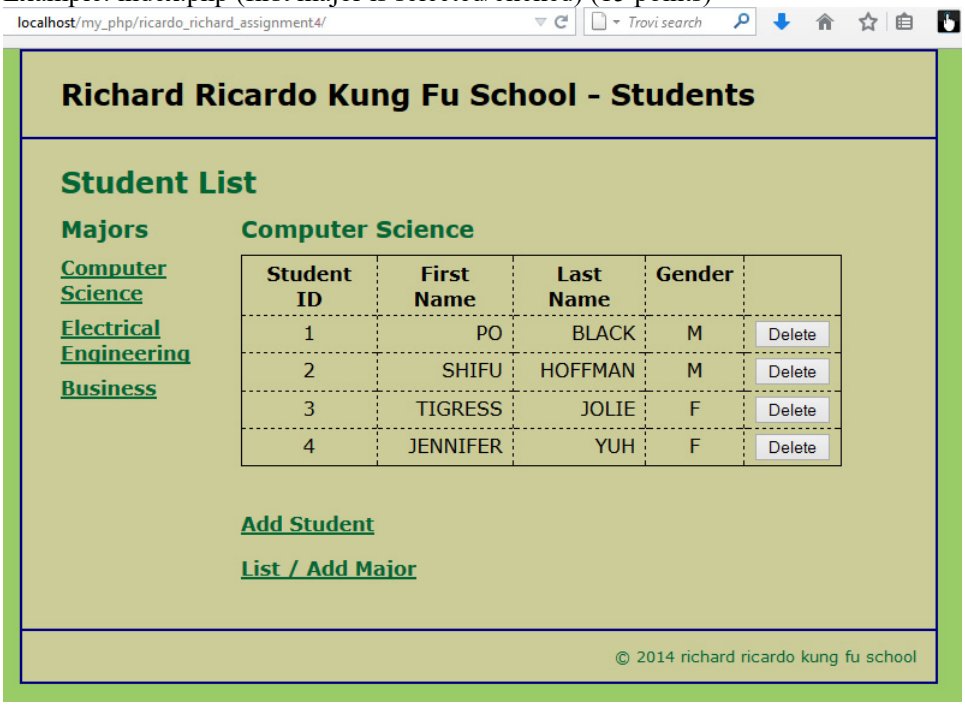
Example: richard_ricardo_database.php (shows nothing, only has php code) (3 points)



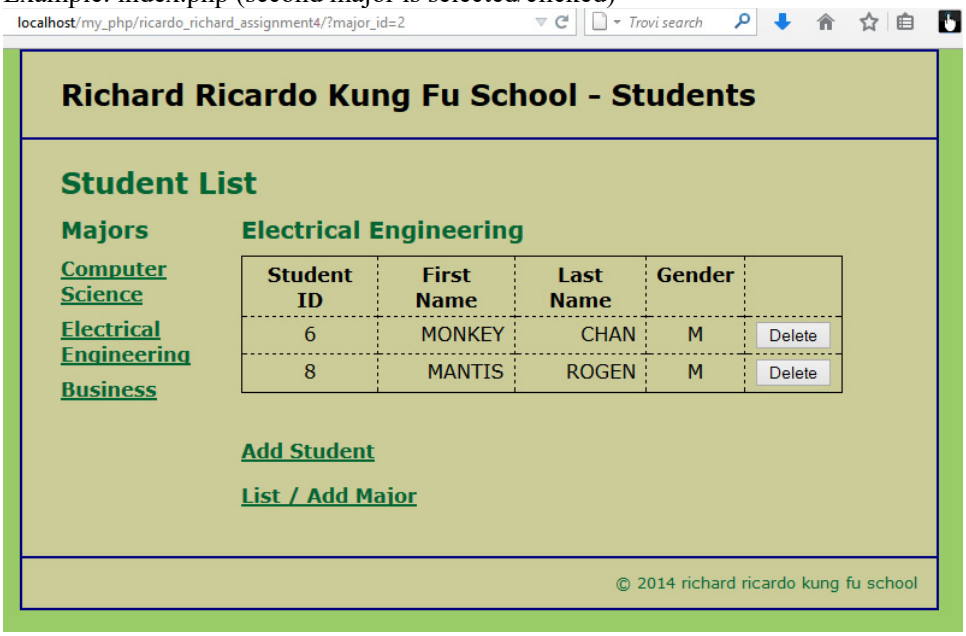
Example: richard_ricardo_database_error.php (shows this if something is wrong with database connection) (4 points)



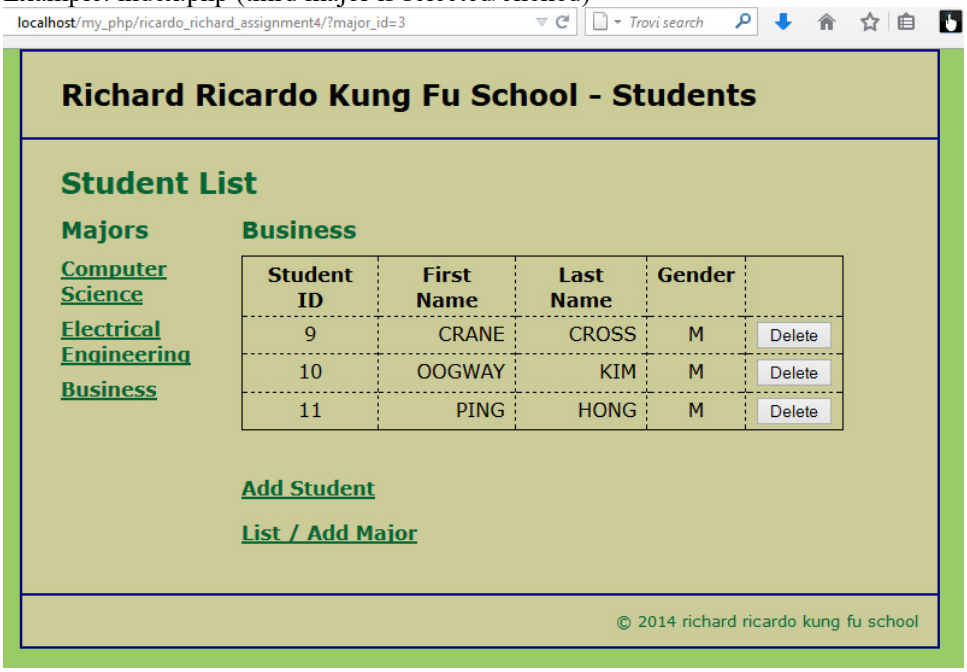
Example: index.php (first major is selected/clicked) (15 points)



Example: index.php (second major is selected/clicked)



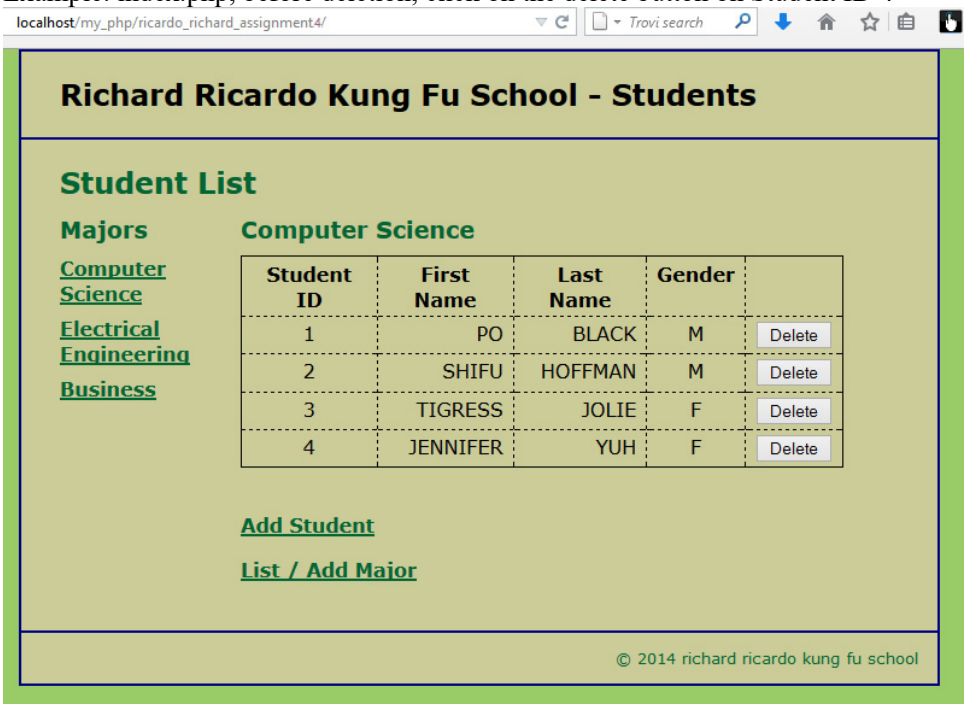
Example: index.php (third major is selected/clicked)



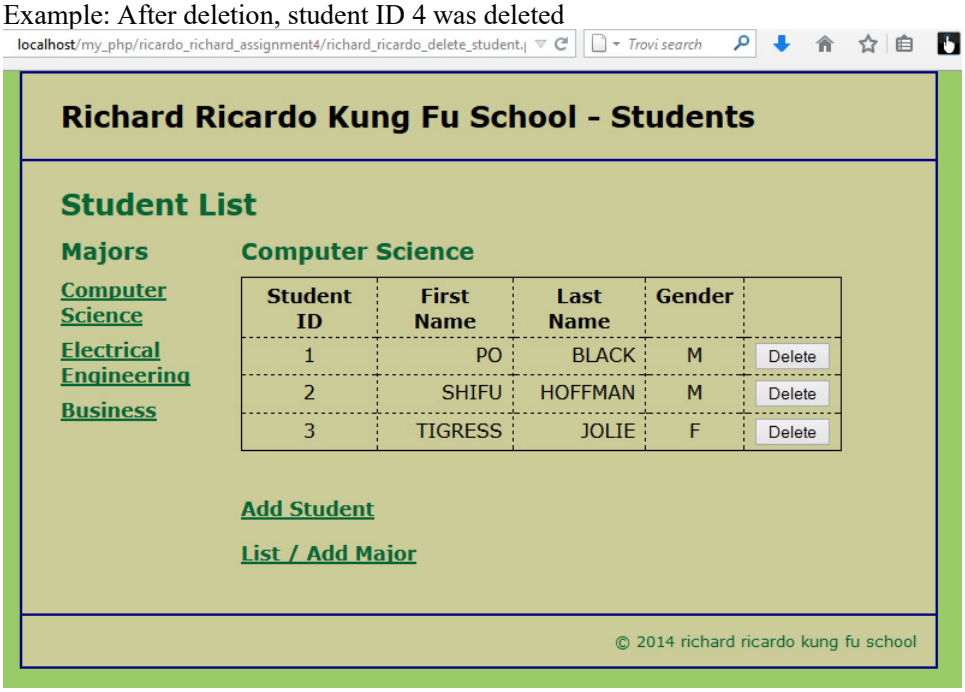
Question 3 – **Delete Student**: PHP Chapter 4, eg009 and SQL (20 points) **Estimated time: 3 hours**

- Save question 3 files in folder "**lastname_firstname_assignment4**". (2 points)
  - **richard_ricardo_delete_student.php**
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "richard ricardo's kung fu school**"** as the page title(s) (<title> tag). (2 points)
- When a user click on a "Delete" link on index.php, (16 points)
  - the user should be directed to **richard_ricardo_delete_student.php**.
  - richard_ricardo_delete_student.php uses **richard_ricardo_database.php** to connect to the database.
  - the selected student record will be deleted from the database table "student".
  - index.php will be shown again with updated records.

Example: index.php, before deletion, click on the delete button on Student ID 4



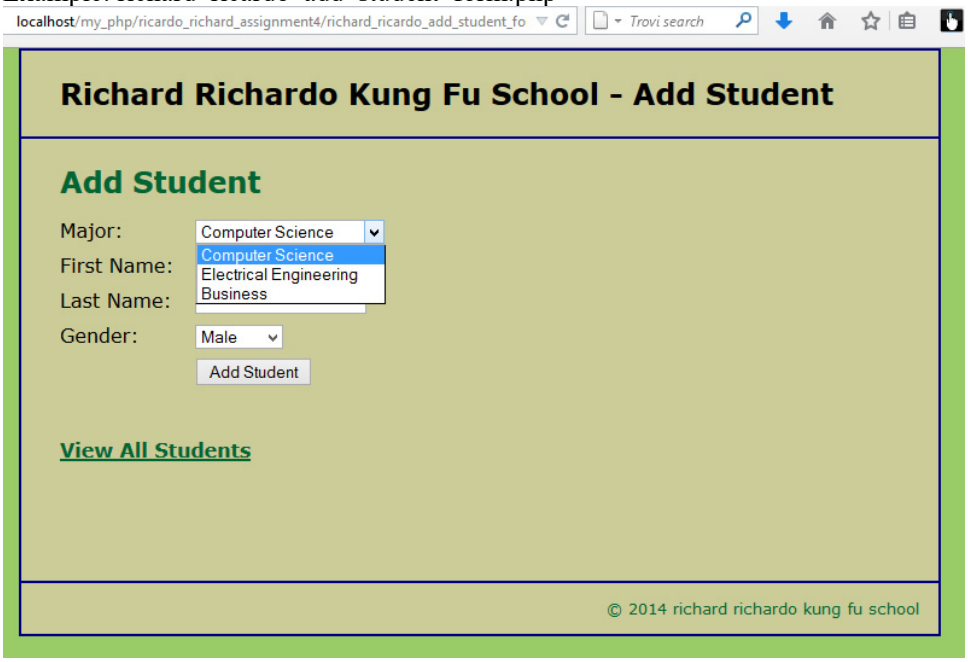
Example: After deletion, student ID 4 was deleted



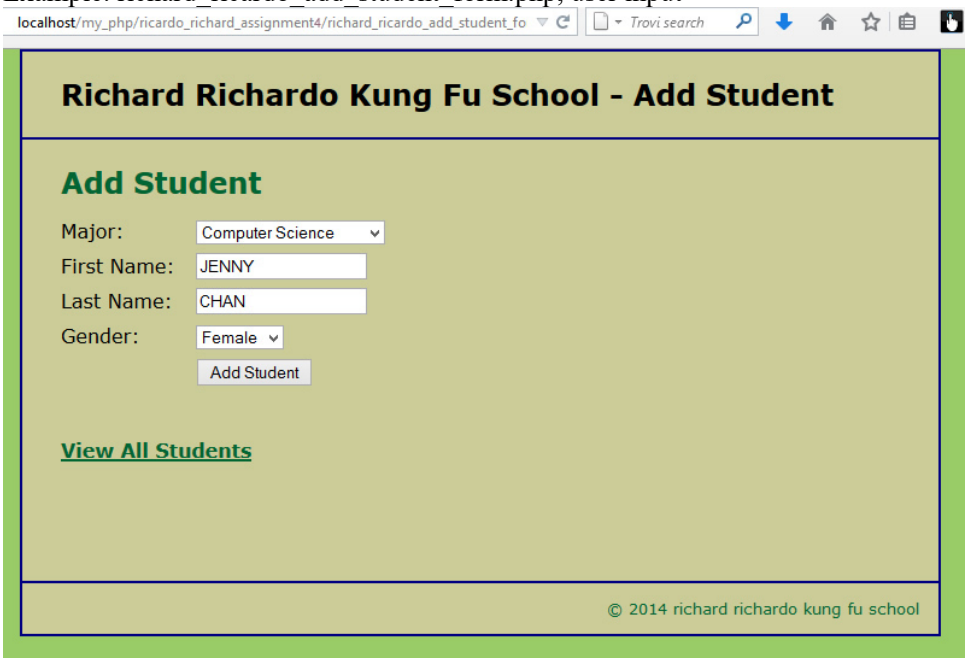
Question 4 – **Add Student**: PHP Chapter 4, eg009 and SQL (25 points) **Estimated time: 4 hours**

- Save question 4 files in folder "**lastname_firstname_assignment4**". (2 points)
  - **richard_ricardo_add_student_form.php**
  - **richard_ricardo_add_student.php**
  - **richard_ricardo_error.php**
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "richard ricardo's kung fu school**"** as the page title(s) (<title> tag). (2 points)
- When a user click on "Add Student" link on index.php, (21 points)
  - the user should be directed to richard_ricardo_add_student_form.php.
  - a link "View All Students" giver the user option to go back to index.php.
  - richard_ricardo_add_student_form.php uses **richard_ricardo_database.php** to connect to the database.
  - A dropdown list will show all majors by retrieve records from "major" database table (created in Q1).
  - After the user keyed in new student data, and press the submit (Add Students) button, richard_ricardo_add_student.php will be used to add student record to the "student" database table, and index.php will be shown again with updated records.
    - richard_ricardo_add_student.php uses **richard_ricardo_database.php** to connect to the database.
  - If the user left any textbox empty, richard_ricardo_error.php page will be shown.

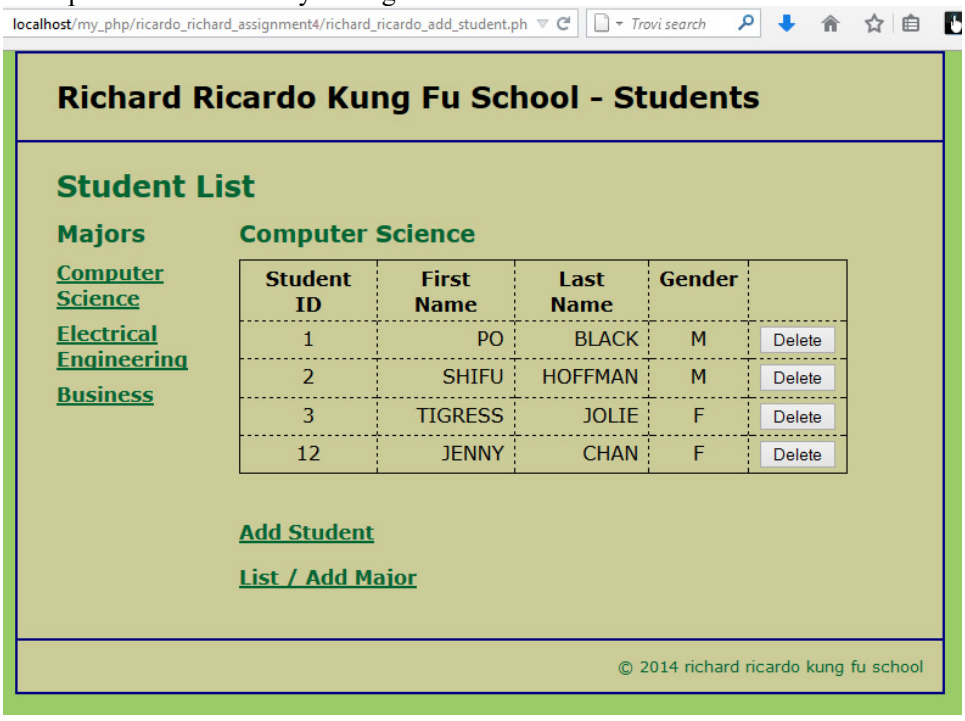
Example: richard_ricardo_add_student_form.php



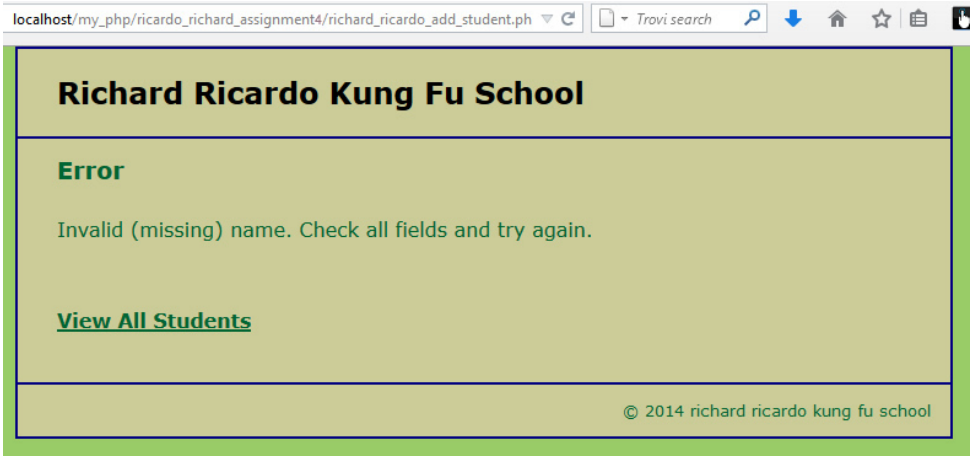
Example: richard_ricardo_add_student_form.php, user input



Example: after successfully adding student record



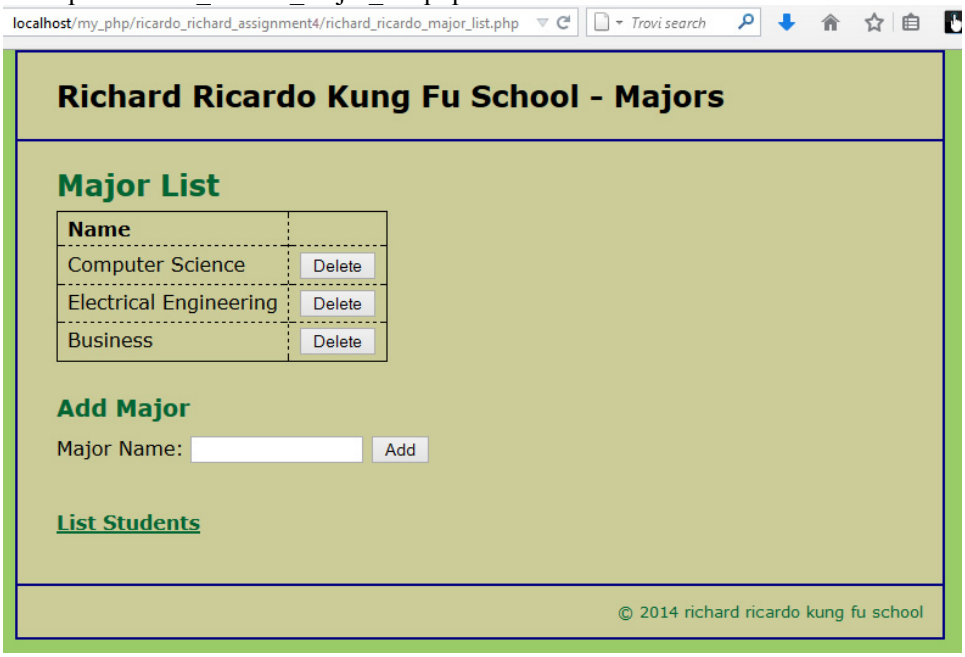
Example: richard_ricardo_error.php page: If any textbox is left empty, richard_ricardo_error.php page will be shown.



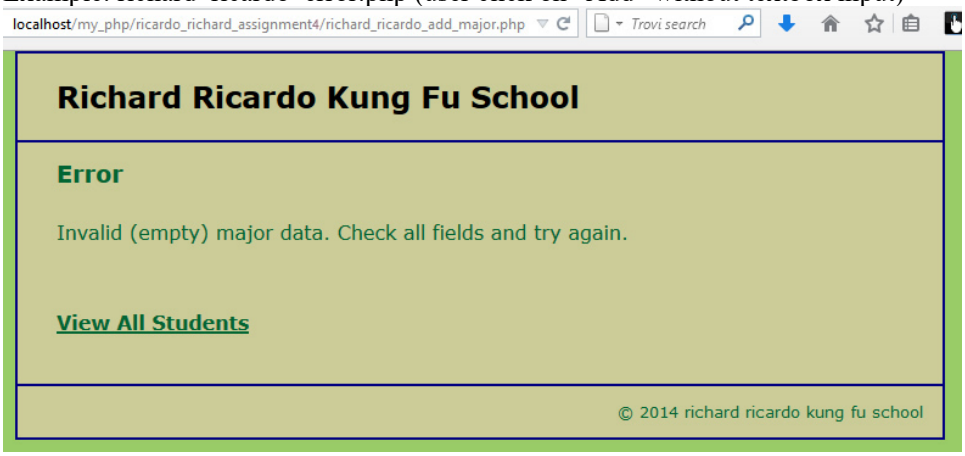
Question 5 Extra Credit – List / Add Major: (20 points) **Estimated time: 4 hours**

- Save question 5 files in folder "**lastname_firstname_assignment4**". (1 point)
  - **richard_ricardo_major_list.php**
  - **richard_ricardo_delete_major.php**
  - **richard_ricardo_add_major.php**
- The initial page and related outputs should look like the examples shown below.
- Create your page(s) using "richard ricardo's kung fu school**"** as the page title(s) (<title> tag). (1 point)
- When a user click on a "List / Add Major" link on index.php, (2 points)
  - the user should be directed to **richard_ricardo_major_list.php**.
  - richard_ricardo_major_list.php uses **richard_ricardo_database.php** to connect to the database.
  - records from the "major" table will be displayed.
- When the user click on "Add" button without keying in anything, richard_ricardo_error.php page (created in Q4) will be shown. (4 points)
- When the user key in some major, and click on "Add" button, (4 points)
  - the user should be directed to richard_ricardo_add_major.php.
  - richard_ricardo_add_major.php uses **richard_ricardo_database.php** to connect to the database.
  - user input will be added to "major" database table.
  - richard_ricardo_major_list.php will be shown again with updated records.
- When the user click on a "Delete" link on richard_ricardo_major_list.php, (8 points)
  - the user should be directed to **richard_ricardo_delete_major.php**.
  - richard_ricardo_delete_major.php uses **richard_ricardo_database.php** to connect to the database.
  - the selected major record will be deleted from the database table "major".
  - richard_ricardo_major_list.php will be shown again with updated records.
  - Due to foreign key referencing, user should not be able to delete any "major" record with "student" record linked to it (Q1).

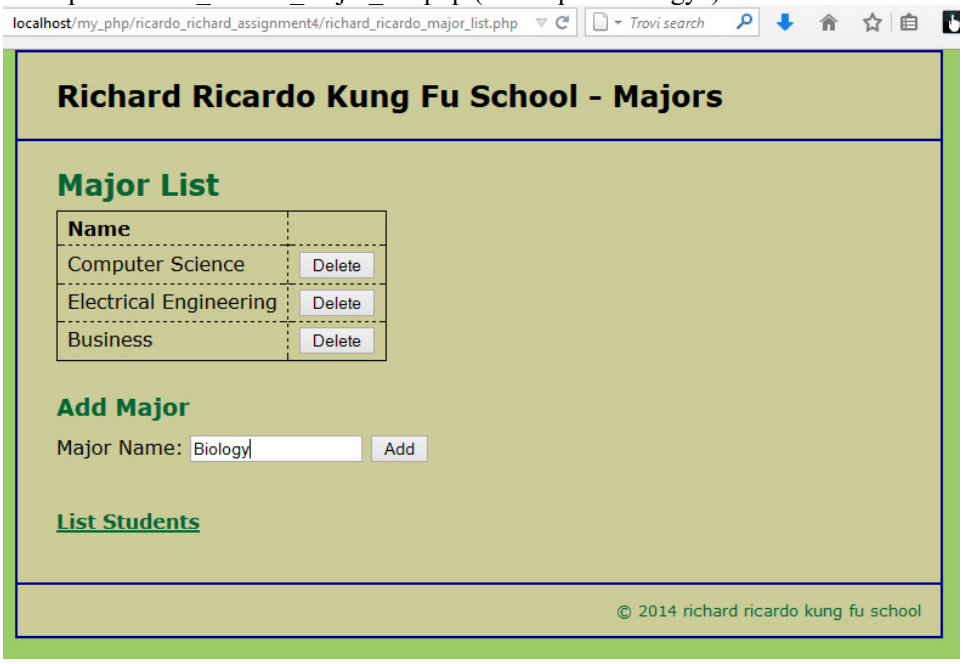
Example: richard_ricardo_major_list.php



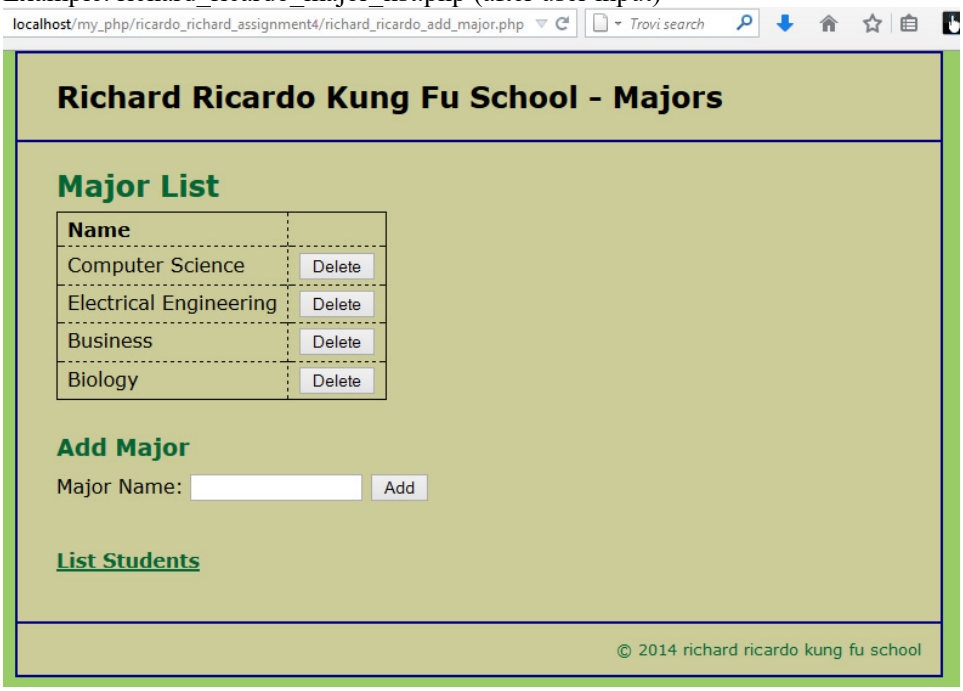
Example: richard_ricardo_error.php (user click on "Add" without textbox input)



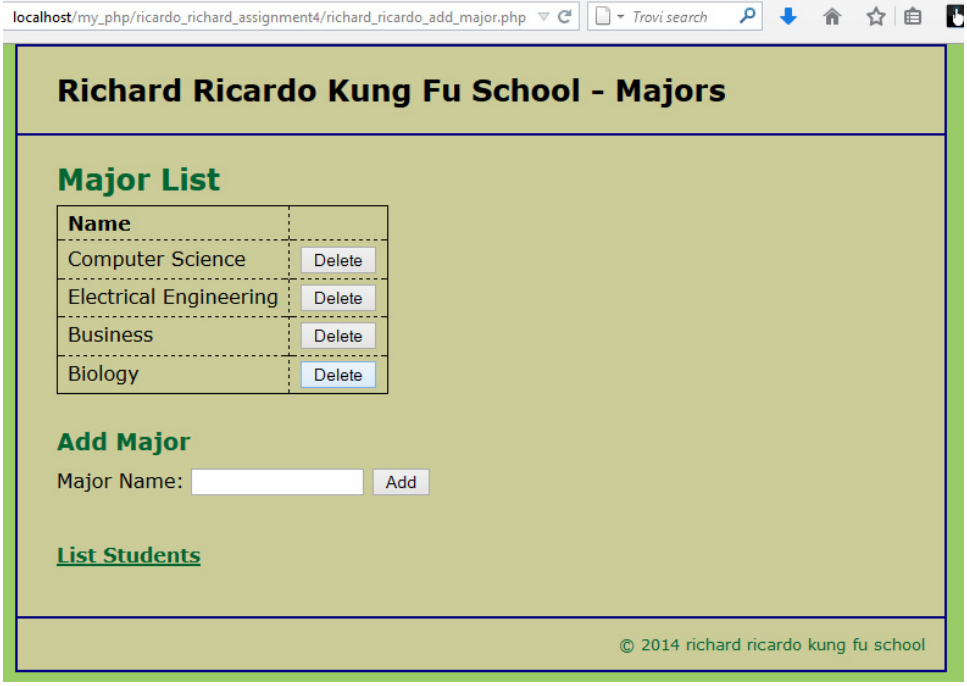
Example: richard_ricardo_major_list.php (user input “biology”)



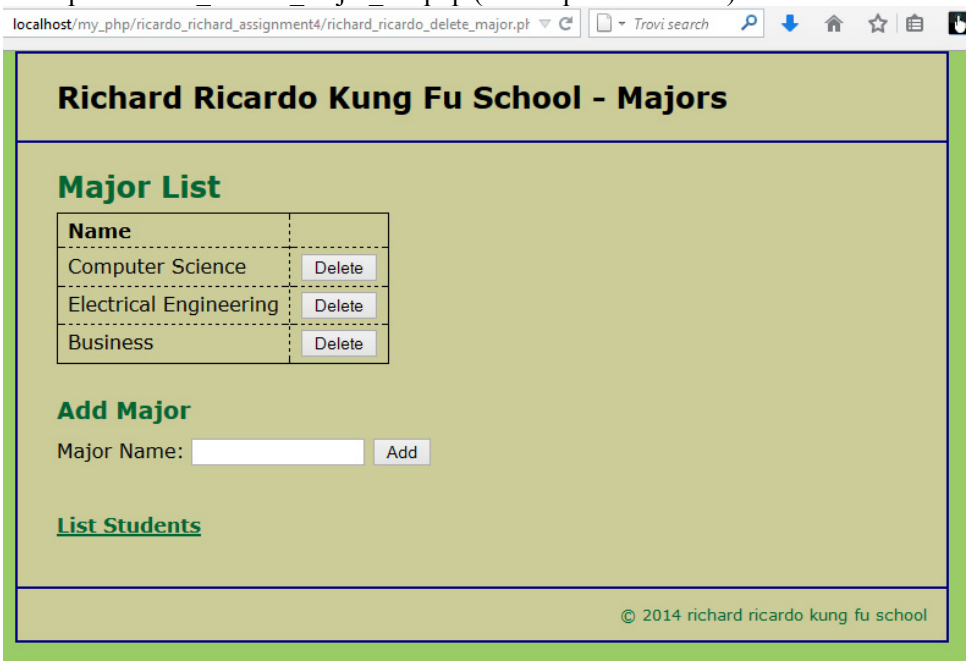
Example: richard_ricardo_major_list.php (after user input)



Example: richard_ricardo_major_list.php (user to click on “Delete” next to “biology”)



Example: richard_ricardo_major_list.php (show updated records)



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. Before adding PHP code, all html files must pass html validation at http://validator.w3.org/ without any **error** (and with only 1 warning). Use the validator's "File Upload" tab to check each file.
    a. If you want to validate a PHP file after adding PHP code, you can temporarily rename the *.php file to *.htm file.
    b. 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)
    c. If any html error is found, 2 points will be deducted for each error. Please validate the files before adding PHP code.
4. All css files must pass css validation at http://jigsaw.w3.org/css-validator/ without any **error**.
5. If your files do not pass the html and css validations, **2 points will be deducted** for **each html or css error** found (no deduction for php error at html validation).
6. 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 Fireworks, GIMP, or Microsoft Expression Design 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_assignment4.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_assignment4**) into a single zip file (or rar file) **lastname_firstname_assignment4.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_assignment4\create_db.sql
- lastname_firstname_assignment4\index.php
- lastname_firstname_assignment4\main.css
- lastname_firstname_assignment4\richard_ricardo_add_major.php (extra credit)
- lastname_firstname_assignment4\richard_ricardo_add_student.php
- lastname_firstname_assignment4\richard_ricardo_add_student_form.php
- lastname_firstname_assignment4\richard_ricardo_database.php
- lastname_firstname_assignment4\richard_ricardo_database_error.php
- lastname_firstname_assignment4\richard_ricardo_delete_major.php (extra credit)
- lastname_firstname_assignment4\richard_ricardo_delete_student.php
- lastname_firstname_assignment4\richard_ricardo_error.php
- lastname_firstname_assignment4\richard_ricardo_major_list.php (extra credit)

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**.