Requirements:

* Add all files to a folder and submit the compressed folder as a zip file, called **484-php-<yournames>.zip**including html, php, .sql files and a text file for mysqldump output.
* Include a text file called README.txt in the folder as well with your notes, explaining what parts of the assignment is done and functional.
* Honor indentation and write clean code.
* Add comments for clarity.
* Develop in XAMPP stack.
* Sharing the source code and copying code from other resources will count as cheating and will result in 0 credit.

Goal:

Building a "Sign up"/"Log in" page for a website.

Grading Criteria:

* Instructions must be followed.
* The functionality of the website is of great importance
* CSS/Styling is not the primary goal of the assignment; you can keep it to the required minimum as long as your pages are clean and organized.

Database, final.sql  : (20 pt)

* Create the following user on your database server:
  + username: webphp
  + password: password
  + **This is the user you will use to connect to the DB from your script.**
* Investigate book.sql (file on canvas) which we used to create “books” DB. Use it as a guide to create a new DB, table, and columns as described here.
* Using mysql command line utility, create a mysql Database called "finaldb". Create a table called "auth" in "finaldb" database.
* Create the following columns in “auth” table: userid, username, password, firstName, lastName, email, phone. Please use those exact names for the columns.
* Set “userid” field as the “primary key” of the table as an auto-incremented field. Set its type to integer. The type of all other fields should be " varchar(60)".
* Insert 3 records in the database.
* Include all the mysql commands you used to create the DB, the table and to insert records in a file called final.sql

home.html: (15 pt)

* This page is the start page of the project. It includes 2 separate forms. "Log in" form, to be used by returning (registered) users and "Sign up" form, to be used by new users. Each form has a separate submit button and different action. Both use "Post" method.
* "Log in" form has:
  + An h1 element saying "Registered Users:"
  + A label element and then a text input type called "username".
  + A label element and password input type called "password".
  + A submit button named " login", with value "Log in". The user-entered values are sent to a file called login.php on the server.
  + Already registered users should be able to use this section to authenticate and log into the website.

* "Sign up" form has
  + An h1 element saying "New Users:"
  + A label element and then a text input type for "First name", named "firstname".
  + The same for "lastname", "phone", "email", "username", a label and text input for each.
  + A label element and password input type called "password"
  + A label element and password input type called "Retype password"
  + A submit button named " singup", with value "sign up". The user-entered values are sent to a file called signup.php on the server.
  + New users register in this form
* Password matching: (10 pt)

Using javascript make sure the user enters the same string in "password" and "Retype password".

* Add an empty <span id="nomatch"> element right after the "Retype password" input.
* Register an event listener for "keyup" event for the “Retype Password” input, which happens every time a user releases a key, so when typing inside the input box, it will happen after each character is typed.
* The event calls a function called "passMatch" which compares the values of "password" and "Retype password" input boxes and if they don't match, it adds a text inside the <span> element with red font color, saying "passwords don't match!". If the values match, the function removes the text inside the <span> element.

File signup.php (25 pt)

* Using php $\_POST array, processes the input values entered in "Sign up" form of “home.html” page and sent to “signup.php”.
* This page:
* Makes a connection to the DB
* Inserts a new record in the DB with the received information. (For simplicity you don't need to validate the input info)
* If the submission is not successful an error message should be displayed on this page. If successful a success message should be displayed such as "you are registered now".
* For simplicity insert the entered password as plain text into the DB (no need for encryption).
* The script should have a function called showTable, which once called, queries the DB and displays a **table** of all registered users' firstname and lastname.
* Call function showTable at the end to display the table content at the bottom on the page. (you can use this function for debugging purposes as well)

File login.php (25 pt)

* Using php $\_POST array, accesses the input values in "Log in" form.
* The page:
* Connects to the DB
* Using a select query looks for a record in the DB which matches the entered value in "username" input box.
* If the record is found, checks if the password entered by the user matches the password stored in the record.
* If it is not a match displays an error message "Incorrect username or password."
* If it is a match, display a success message such as " Welcome back <username>! Login was successful". Then display the user's firstname, lastname, email and phone in a table with headers.

Database Dump dbdump.txt:  (5 pt)

* Use mysqldump utility from the command line to dump the database you created in a text file called dbdump.txt. the format of the command to do so looks like this:
  + mysqldump -u username -p [database name] [table name] > [filename]
  + Refer to online resources for more information.
  + [https://mariadb.com/kb/en/mariadb/mysqldump/ (Links to an external site.)Links to an external site.](https://mariadb.com/kb/en/mariadb/mysqldump/)