# **PHP and MySQL**

## Description

The goal of this project is to learn server-side web programming using PHP and a relational database system (MySQL). On top of your pervious website

## Platform

Did this project on windows PC/laptop. You need to install the [XAMPP](https://www.apachefriends.org/) web server, which includes the Apache http web server, PHP, MySQL (MariaDB), and PHPMyAdmin. It can be installed on Windows, Linux, and OS X. You will test the project on your PC/laptop using the Mozilla Firefox web browser.

users (username, password, fullname, email)

posts (id, replyto, postedby, datetime, message)

Primarykeys: users.username and posts.id.

Foreignkeys: posts.postedby->users.username and posts.replyto->posts.id.  
To create the database, start the Apache Web Server and the MySQL Database on your PC using the XAMPP manager console. Run [phpMyAdmin](http://localhost/phpmyadmin/) on your browser, create a new database with name board by clicking on New. After you create it, select this database (under the name board), go to the SQL tab, and cut and paste the SQL code in createDB.sql and push Go. This will create your schema. You can test your setup on your web browser by using the URL address <http://localhost/PA4/board.php>

The PA4 directory contains the file board.php that uses the PDO extension of PHP to insert a new user and to query the users table using MySQL.

## Project Requirements

Writen two PHP scrips *login.php* and *board.php*. The *login.php* script generates a form that has two text windows for username and password and a "*Login*" button. *The board.php* has a *"Logout*" button, a textarea to write a message, *a "New Post"* button, and a list of messages. The board script prints all the messages in the database as a flat list ordered by date/time (newest first, oldest last). Note: messages should not be organized based on their reply to attributes. For each posted message, it prints:

* The message ID.
* The username and the fullname of the person who posted the message.
* The date and time when this message was posted.
* If this is a reply to a message, the ID of this message.
* The message text.
* A button "Reply" to reply to this message.

From the login script, if the user enters a wrong username/password and pushes "Login", it should go to the login script again. If the user enters a correct username/password and pushes "Login", it should go to the board script. From the board script, if the user pushes "Logout", it should logout and go to the login script. The board script must always make sure that only authorized used (users who have logged-in properly) can view and post messages. From the board script, if the user fills out the textarea and pushes the "New Post" button, it will insert the new message in the database (with null replyto attribute) and will go to the board script again. If the user fills out the textarea and pushes the "Reply" button, it will insert the message in the database -- but this time you need to set the replyto value and will go to the board script again.

Hints: Each Reply button must have an action that submits the form to *board.php* with a different replyto value. You may use a [form button](http://www.w3schools.com/tags/tag_button.asp) with type="submit" and formaction="board.php?replyto=12345" to reply to a message with ID 12345.  
Use [md5](http://us2.php.net/manual/en/function.md5.php) to encode passwords in PHP. Use [uniqid](https://www.php.net/manual/en/function.uniqid.php)to generate a unique id in PHP. Use the MySQL function [now()](http://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html) to return the current date and time.