CSE 3100: Web Programming Laboratory

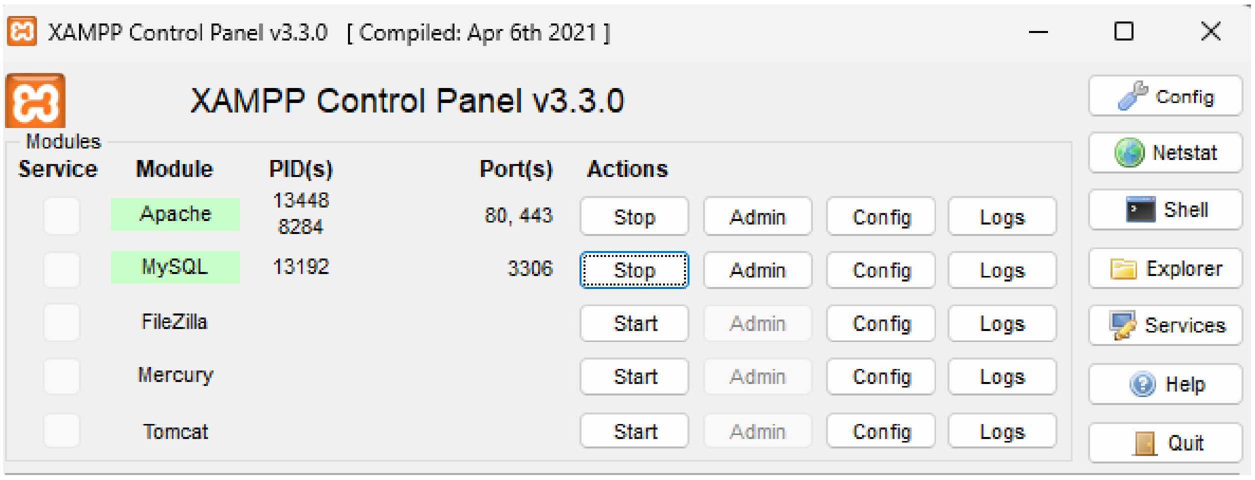
# Lab 5: Database Management, Sessions, Cookies in PHP

**Farhan Sadaf**
**Lecturer,**
**Dept of CSE, KUET**
**Email: farhansadaf@cse.kuet.ac.bd**

**Kazi Saeed Alam**
**Assistant Professor,**
**Dept of CSE, KUET**
**Email: saeed.alam@cse.kuet.ac.bd**

## PHP & MySQL

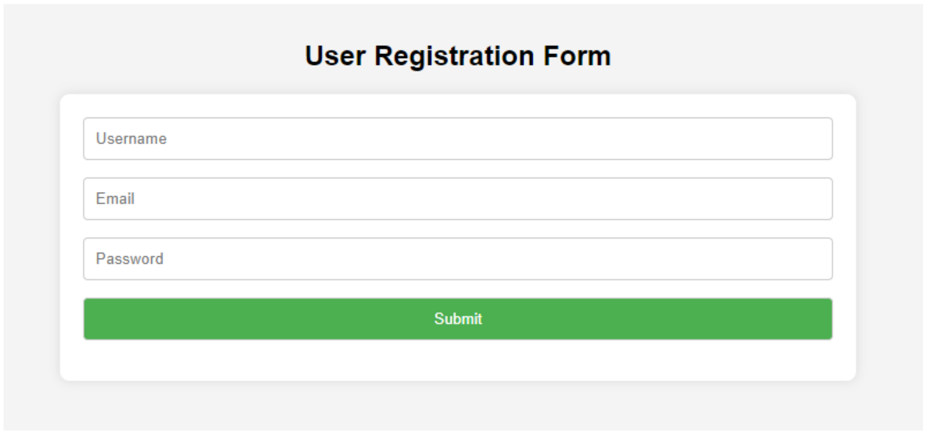
PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. In this lab, you will be creating a database system and learn about CRUD (Create, Read, Update, Delete) operations using MySQL in PHP. To set it up, first open your XAMPP Control Panel and turn MySQL services on along with the Apache server.


XAMPP Control Panel v3.3.0 [ Compiled: Apr 6th 2021 ]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

| Service | Module | PID(s) | Port(s) | Actions | | | |
|---|---|---|---|---|---|---|---|
| | Apache | 13448 8284 | 80, 443 | Stop | Admin | Config | Logs |
| | MySQL | 13192 | 3306 | Stop | Admin | Config | Logs |
| | FileZilla | | | Start | Admin | Config | Logs |
| | Mercury | | | Start | Admin | Config | Logs |
| | Tomcat | | | Start | Admin | Config | Logs |

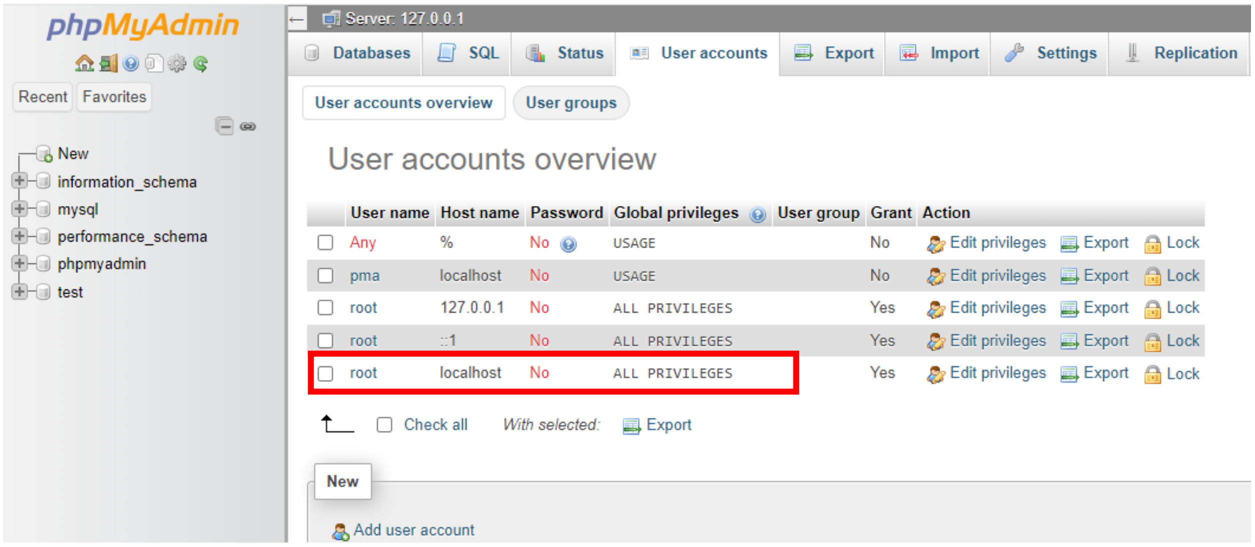
## Registration Form

So far, you have already learnt how to create and process forms using HTML and CSS. Now open ‘**reg.php**’ and ‘**reg.css**’ and try to understand the structure of the form. In the browser the ‘reg.php’ should appear like this:



## Step 1: Database Connection

Now paste **'http://localhost/phpmyadmin'** URL in your browser to open phpMyAdmin, an open-source web-based administration tool for managing MySQL database. It is written in PHP and provides a graphical user interface (GUI) to interact with the MySQL database. You will now interact with this GUI to manage our database. First, you need to open **User accounts** section if you want to create a new user account otherwise, you can use the available **root** user. Today you will use the user's name = 'root' and Host name = 'localhost' for your database connection.



User accounts overview

| | User name | Host name | Password | Global privileges | User group | Grant | Action |
|---|---|---|---|---|---|---|---|
| ☐ | Any | % | No | USAGE | | No | Edit privileges Export Lock |
| ☐ | pma | localhost | No | USAGE | | No | Edit privileges Export Lock |
| ☐ | root | 127.0.0.1 | No | ALL PRIVILEGES | | Yes | Edit privileges Export Lock |
| ☐ | root | ::1 | No | ALL PRIVILEGES | | Yes | Edit privileges Export Lock |
| ☐ | root | localhost | No | ALL PRIVILEGES | | Yes | Edit privileges Export Lock |

The **mysqli_connect** function is used to open a new connection to the MySQL server. It establishes a connection and returns a MySQLi object that can be used for further database operations. The function can have several optional parameters of which we will use 4 parameters for now.

**Syntax**: mysqli_connect(hostname, username, password, **database name**);

- You can create the database using phpMyAdmin GUI or SQL query in your code. If you follow the latter, you can omit the database name from the connection command. Now add the following code in your 'reg.php' file.
- If you are encountering an error with mysqli_connect, it's important to troubleshoot and identify the specific issue. You can use mysqli_connect_error() to check any error occurred.
- Finally, mysqli_close is used to close a database connection. This function takes connection resource returned by mysql_connect function. It returns TRUE on success or FALSE on failure.

```php
<?php
if (isset($_POST['submit'])) {

    $connection = mysqli_connect('localhost', 'root', '');

    if (!$connection) {
        die("Connection failed: " . mysqli_connect_error());
    }

    echo "Connected successfully";


    mysqli_close($connection);
}
?>
```

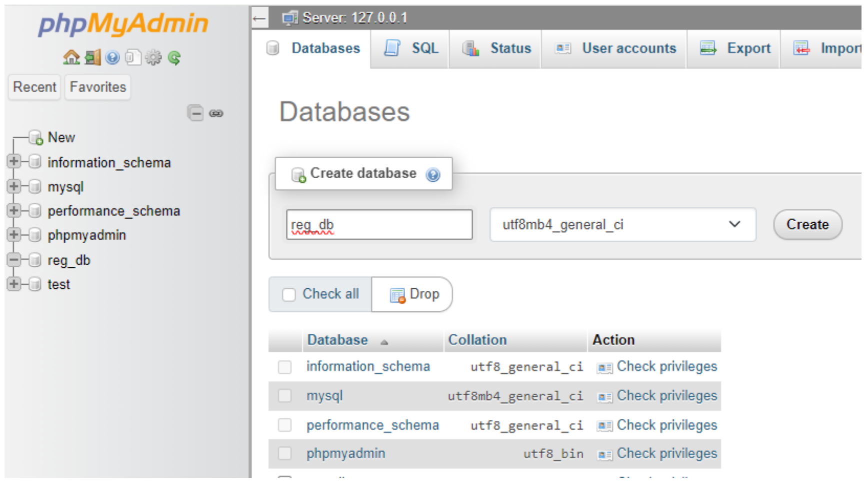
## Step 2: Database Creation

You can use either SQL query or phpMyAdmin GUI to create a Database.

Using SQL Query:

```php
// Create a new database
$sql = 'CREATE DATABASE test_db';
$retval = mysqli_query($conn, $sql);
if (!$retval) {
    die('Could not create database: ' . mysqli_error($conn));
}
```

Using phpMyAdmin GUI: (We will follow this one)



reg_db utf8mb4_general_ci

| Database | Collation | Action |
|---|---|---|
| information_schema | utf8_general_ci | Check privileges |
| mysql | utf8mb4_general_ci | Check privileges |
| performance_schema | utf8_general_ci | Check privileges |
| phpmyadmin | utf8_bin | Check privileges |

The following task is to create database tables.

Using SQL Query in the same way shown before:

```
CREATE TABLE `reg_db`.`reg_table` (`id` INT(11) NOT NULL AUTO_INC
REMENT , `username` VARCHAR(255) NOT NULL , `email` VARCHAR(255)
NOT NULL , `password` VARCHAR(255) NOT NULL , `submit_time` TIMES
TAMP NOT NULL DEFAULT CURRENT_TIMESTAMP , PRIMARY KEY (`id`)) ENG
INE = InnoDB;
```

Using phpMyAdmin GUI: (We will follow this one)

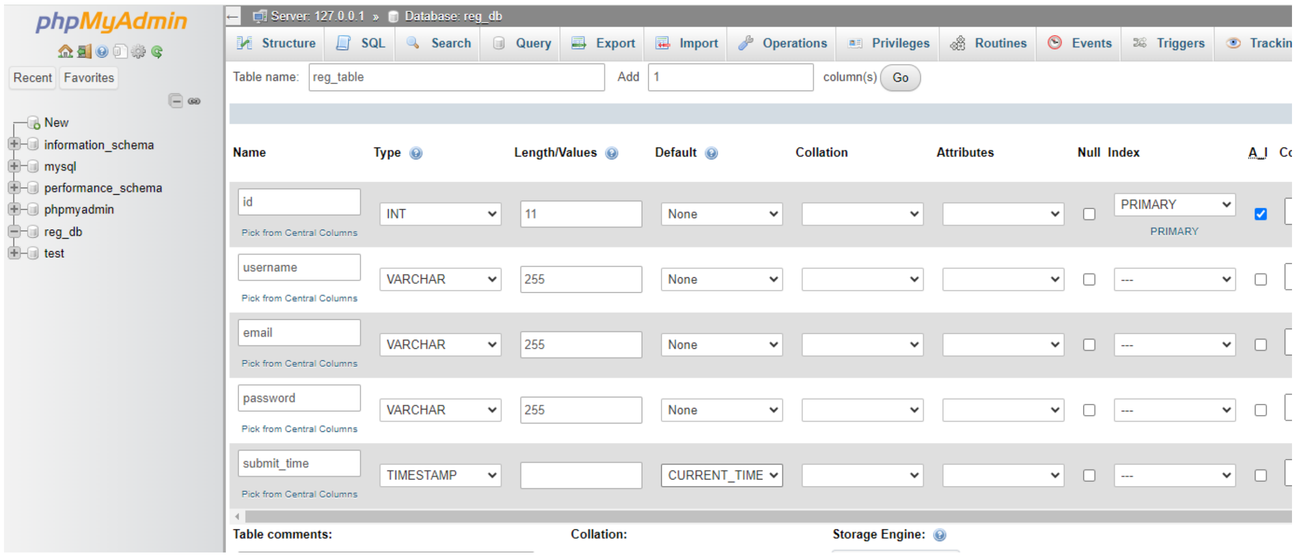


Table name: reg_table Add 1 column(s)

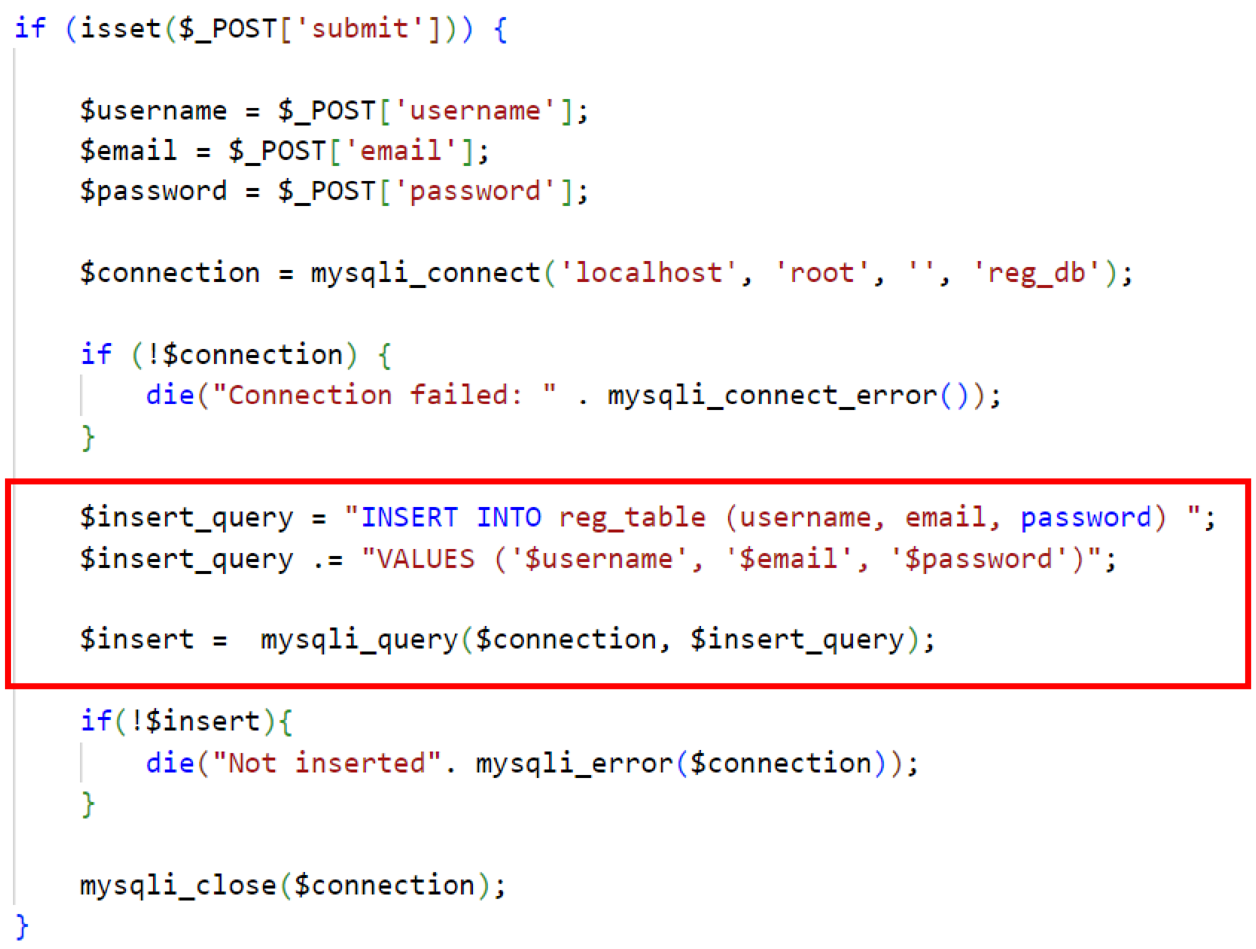
| Name | Type | Length/Values | Default | Collation | Attributes | Null | Index | A_I |
|---|---|---|---|---|---|---|---|---|
| id | INT | 11 | None | | | ☐ | PRIMARY (PRIMARY) | ☑ |
| username | VARCHAR | 255 | None | | | ☐ | --- | ☐ |
| email | VARCHAR | 255 | None | | | ☐ | --- | ☐ |
| password | VARCHAR | 255 | None | | | ☐ | --- | ☐ |
| submit_time | TIMESTAMP | | CURRENT_TIME | | | ☐ | --- | ☐ |

Table comments: Collation: Storage Engine:

Note that, you need to add Primary Key (Unique Identifier) to your database table. Here, you will use the attribute 'id' as the primary key. The other attributes can be chosen as required. For example, you can have 'CURRENT_TIME' as the default value for the 'submit_time' attribute.

## Step 3: Insert Data

Now you need to process the data given by the user in the registration form. You can store the data using the Superglobals $_POST or $_GET depending on the task. After that you have to add the insert query from MySQL to insert the data whenever any user submits the form.



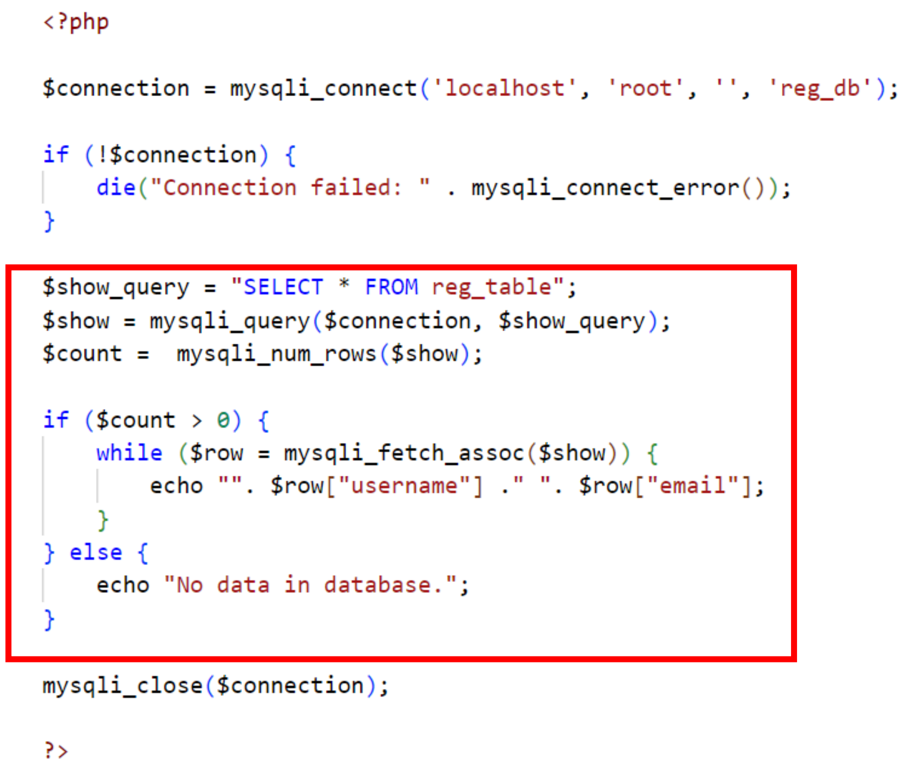
```
if (isset($_POST['submit'])) {

    $username = $_POST['username'];
    $email = $_POST['email'];
    $password = $_POST['password'];

    $connection = mysqli_connect('localhost', 'root', '', 'reg_db');

    if (!$connection) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $insert_query = "INSERT INTO reg_table (username, email, password) ";
    $insert_query .= "VALUES ('$username', '$email', '$password')";

    $insert =  mysqli_query($connection, $insert_query);

    if(!$insert){
        die("Not inserted". mysqli_error($connection));
    }

    mysqli_close($connection);
}
```

**H.W #1:** ***Now try to add constraints: restrict empty strings, apply unique username, maintain specific password structure etc.***

**H.W #2:** ***Now try to add specific messages whenever a data is inserted/updated/deleted. (you will see update and delete shortly)***

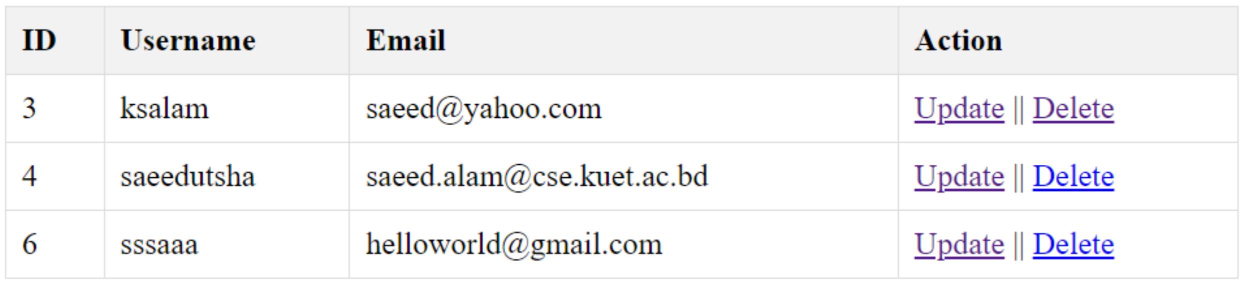
## Step 4: Read/Show Data

Now open **‘view.php’, ‘table.html’** and **‘table.css’** where you will show the data inserted previously. You can use your own data format structure or style to show the data. You need to add the contents of ‘table.html’ into ‘view.php’ and modify ‘view.php’ so that it can show the data from the database.



```php
<?php

$connection = mysqli_connect('localhost', 'root', '', 'reg_db');

if (!$connection) {
    die("Connection failed: " . mysqli_connect_error());
}

$show_query = "SELECT * FROM reg_table";
$show = mysqli_query($connection, $show_query);
$count =  mysqli_num_rows($show);

if ($count > 0) {
    while ($row = mysqli_fetch_assoc($show)) {
        echo "". $row["username"] ." ". $row["email"];
    }
} else {
    echo "No data in database.";
}

mysqli_close($connection);

?>
```

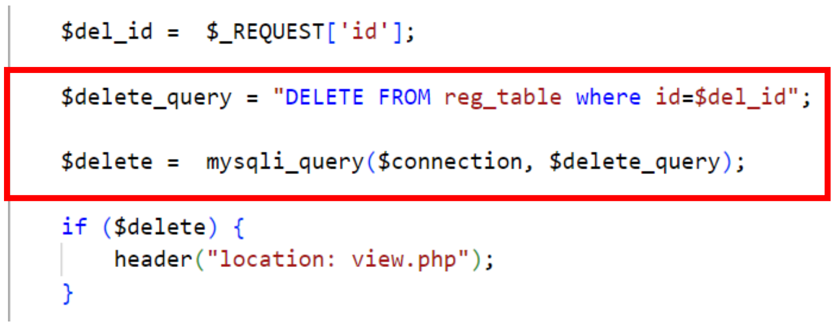
The table should have the following appearance:



| ID | Username | Email | Action |
|---|---|---|---|
| 3 | ksalam | saeed@yahoo.com | Update \|\| Delete |
| 4 | saeedutsha | saeed.alam@cse.kuet.ac.bd | Update \|\| Delete |
| 6 | sssaaa | helloworld@gmail.com | Update \|\| Delete |

## Step 5: Delete Data

At this point, open **‘delete.php’** and modify it so that whenever an user clicks the ‘delete’ button of any row in the table created before, it will delete that particular row associated with it.

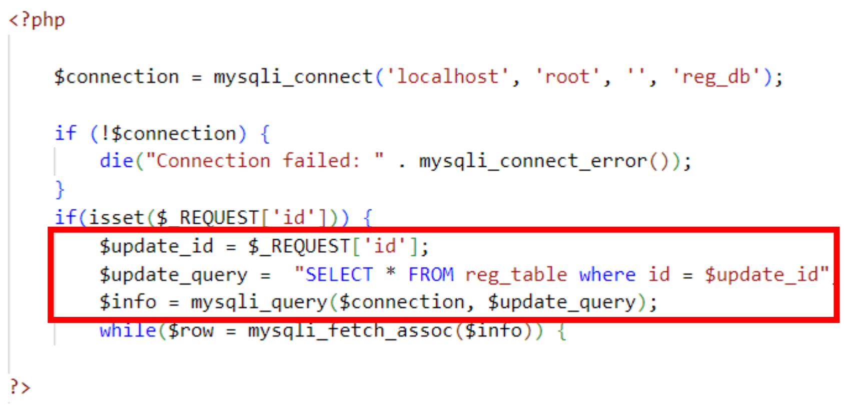


```
$del_id =  $_REQUEST['id'];

$delete_query = "DELETE FROM reg_table where id=$del_id";

$delete =  mysqli_query($connection, $delete_query);

if ($delete) {
    header("location: view.php");
}
```

**Tip**: The part after **‘?’** in **‘localhost/delete.php?id=1’** can be accessed using the superglobal **$_REQUEST[‘id’]**.

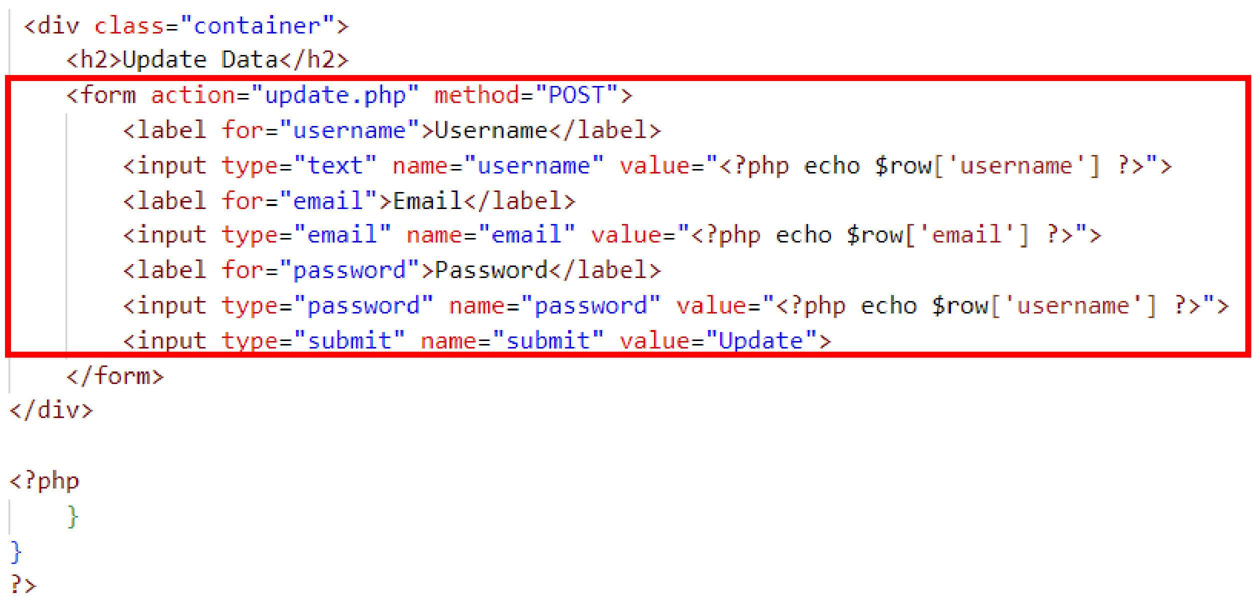
## Step 6: Update Data

To update the data, you need to first visualize the existing data. If any user clicks ‘Update’ in the previously created table, it will redirect user to a page where user will have an update form. You can use **‘edit.php’** to obtain this. First understand the structure and then try to access the id from ‘view.php’ and use it in ‘edit.php’ so that you can see the data to be updated.



```
<?php

    $connection = mysqli_connect('localhost', 'root', '', 'reg_db');

    if (!$connection) {
        die("Connection failed: " . mysqli_connect_error());
    }
    if(isset($_REQUEST['id'])) {
        $update_id = $_REQUEST['id'];
        $update_query =  "SELECT * FROM reg_table where id = $update_id"
        $info = mysqli_query($connection, $update_query);
        while($row = mysqli_fetch_assoc($info)) {

?>
```

Here, first the id is accessed and put into the SQL query to retrieve.

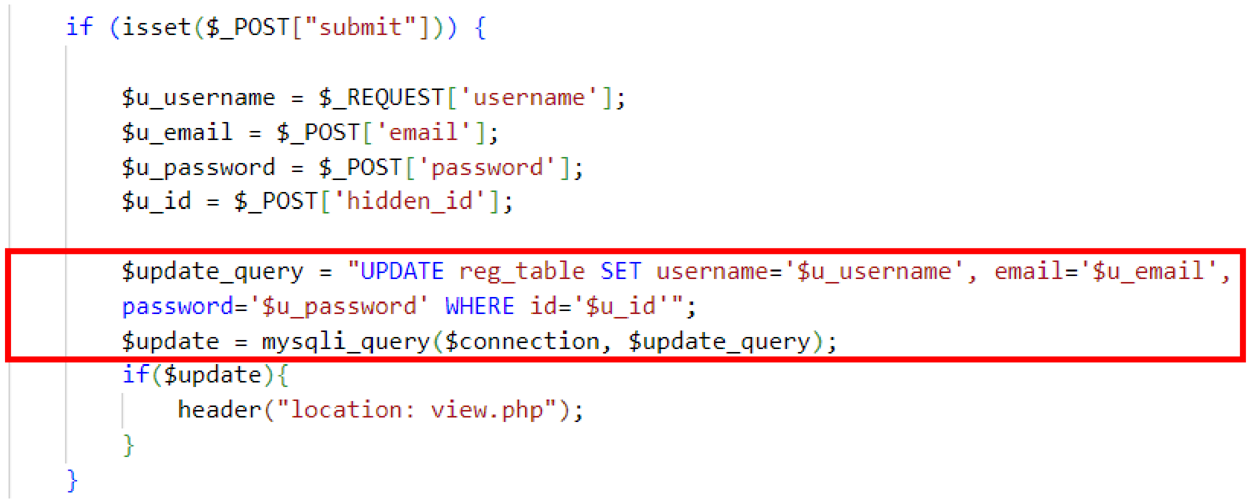


```
 <div class="container">
    <h2>Update Data</h2>
    <form action="update.php" method="POST">
        <label for="username">Username</label>
        <input type="text" name="username" value="<?php echo $row['username'] ?>">
        <label for="email">Email</label>
        <input type="email" name="email" value="<?php echo $row['email'] ?>">
        <label for="password">Password</label>
        <input type="password" name="password" value="<?php echo $row['username'] ?>">
        <input type="submit" name="submit" value="Update">
    </form>
</div>

<?php
    }
}
?>
```

Then, the data is shown in a HTML form.

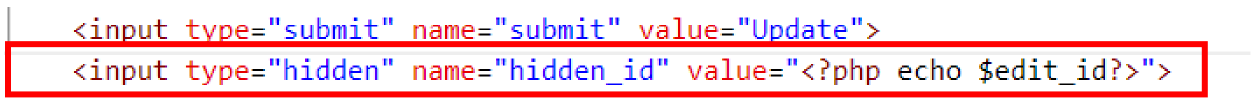
Tip: ***You have to be careful while writing both HTML and PHP in the same file. Specially with the starting and closing tags of PHP.***

So, now the user can see what data is already stored for any particular id. Now the user can give new values of the attributes to update. At this point, open the **'update.php'** file to modify it to achieve the update operation.



```
    if (isset($_POST["submit"])) {

        $u_username = $_REQUEST['username'];
        $u_email = $_POST['email'];
        $u_password = $_POST['password'];
        $u_id = $_POST['hidden_id'];

        $update_query = "UPDATE reg_table SET username='$u_username', email='$u_email',
        password='$u_password' WHERE id='$u_id'";
        $update = mysqli_query($connection, $update_query);
        if($update){
            header("location: view.php");
        }
    }
```

As you will need the **id** to update the data, you need to send the **id** as hidden input in the form as following:



```
    <input type="submit" name="submit" value="Update">
    <input type="hidden" name="hidden_id" value="<?php echo $edit_id?>">
```

Finally, the updated data will be appeared in the table.

# PHP – Cookies

A cookie is often used to identify a user. Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users –

- Server script sends a set of cookies to the browser. For example, name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this –

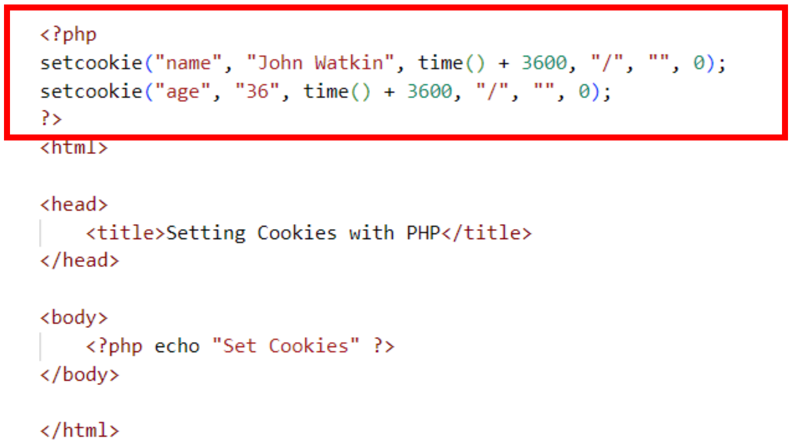
```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
SetCookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
           path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

A PHP script will then have access to the cookie in the environmental variables **$_COOKIE.**

## Setting Cookies with PHP:

PHP provided setcookie() function to set a cookie. This function requires upto six arguments and **should be called before <html> tag**. For each cookie this function has to be called separately.
**setcookie(name, value, expiration, path, domain, security);**

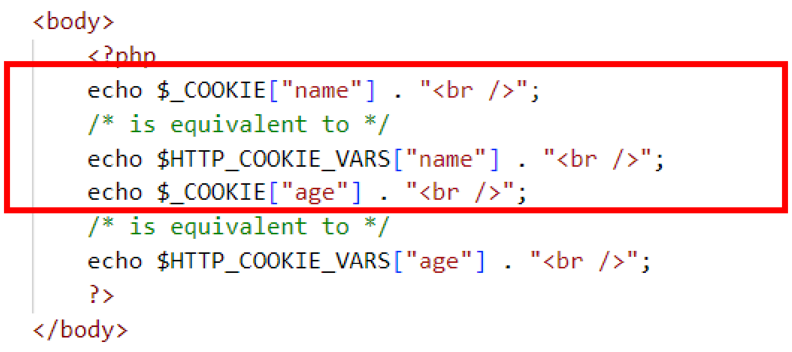
Following example will create two cookies name and age these cookies will be expired after one hour.



```
<?php
setcookie("name", "John Watkin", time() + 3600, "/", "", 0);
setcookie("age", "36", time() + 3600, "/", "", 0);
?>
<html>

<head>
    <title>Setting Cookies with PHP</title>
</head>

<body>
    <?php echo "Set Cookies" ?>
</body>

</html>
```

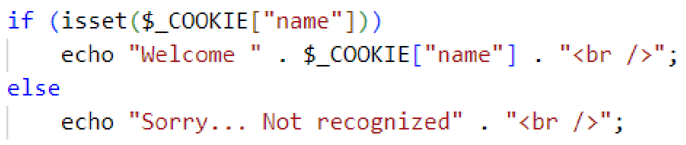
## Accessing Cookies with PHP:

Superglobal $_COOKIE[] can be used to access the cookie.



```
<body>
    <?php
    echo $_COOKIE["name"] . "<br />";
    /* is equivalent to */
    echo $HTTP_COOKIE_VARS["name"] . "<br />";
    echo $_COOKIE["age"] . "<br />";
    /* is equivalent to */
    echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>
</body>
```

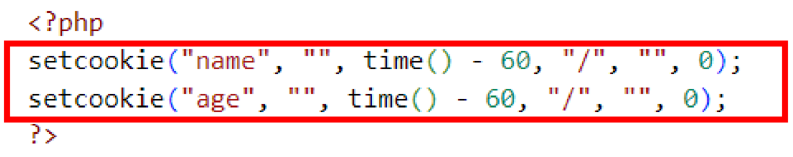
You can use isset() function to check if a cookie is set or not.



```
if (isset($_COOKIE["name"]))
    echo "Welcome " . $_COOKIE["name"] . "<br />";
else
    echo "Sorry... Not recognized" . "<br />";
```

## Deleting Cookies with PHP:

You can set the cookie with a date that has already expired –



```
<?php
setcookie("name", "", time() - 60, "/", "", 0);
setcookie("age", "", time() - 60, "/", "", 0);
?>
```

# PHP – Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the **server** where registered session variables and their values (i.e. username, shopping items, etc.) are stored. This data will be available to all pages on the site during that visit. However, session information will be deleted after the user has left the website.

- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID.
- The UID is either stored in a cookie or is propagated in the URL.

## Working with a PHP Session:

A PHP session is easily started by making a call to the session_start() function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page before the <html> tag.

Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a session.

Make use of isset() function to check if session variable is already set or not.

```
<?php
session_start();
if (isset($_SESSION['counter'])) {
    $_SESSION['counter'] += 1;
} else {
    $_SESSION['counter'] = 1;
}
$msg = "You have visited this page " . $_SESSION['counter'];
$msg .= "in this session.";
?>

<html>

<head>
    <title>Setting up a PHP session</title>
</head>

<body>
    <?php echo ($msg); ?>
</body>

</html>
```

The above example starts a session then register a variable called **counter** that is **incremented** each time the page is visited during the session.

## Destroying a PHP Session:

A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

```
<!-- Here is the example to unset a single variable - -->
<?php
unset($_SESSION['counter']);
?>
<!-- Here is the call which will destroy all the session variables - -->
<?php
session_destroy();
?>
```

**H.W #3:** ***Now based on the registration system, create a login page which will check username and password stored in the database and allows user to log in or not. If logged in, it will show a welcome page.***

**H.W #4:** ***Apply php Session and Cookies functionalities in your login system. What differences can you observe for session vs cookies.***