### National University of Computer & Emerging Sciences, Karachi Computer Science Department



|  |  |
| --- | --- |
| **Course Code: SE-3003** | **Course : Web Engineering Lab** |

**Spring 2023, Lab Manual – 09**

**LLO 02: Web application using PHP and MySQL**

**Contents:**

* Intro to Web Engineering
* Technologies
* Tools
* Introduction to MySQL
* PHP and MySQL Connection
* PHP CRUD
* PHP Signup & Login
* PHP Session and Cookies

**Introduction to Web Engineering**

Web Engineering is the application of systematic and quantifiable approaches (concepts methods, techniques tools) to cost ‐ effective requirements analysis, design, implementation, testing, operation, and maintenance of **high quality Web applications.**

**Technologies to be studied**

* HTML
* CSS
* JavaScript
* Bootstrap
* JQuery
* PHP
* MySQL [Database]
* Laravel [PHP FRAMEWORK]

**Tools – IDEs**

* Visual Studio Code
* Adobe Dreamweaver
* Visual Studio
* XAMPP

**9.1 Introduction to MySQL:**

## SQL stands for Structured Query Language. SQL is a standard programming language specifically designed for storing, retrieving, managing or manipulating the data inside a relational database management system (RDBMS). SQL became an ISO standard in 1987.

## SQL is the most widely-implemented database language and supported by the popular relational database systems, like MySQL, SQL Server, and Oracle. However, some features of the SQL standard are implemented differently in different database systems.

## MySQL Capabilities:

## You can create a database.

## You can create tables in a database.

## You can query or request information from a database.

## You can insert records in a database.

## You can update or modify records in a database.

## You can delete records from the database.

## You can set permissions or access control within the database for data security.

## You can create views to avoid typing frequently used complex queries.

## The list does not end here, you can perform many other database-related tasks with SQL.

## MySQL Structure:

## MySQL database stores data into tables like other relational database. A table is a collection of related data, and it is divided into rows and columns.

Each row in a table represents a data record that are inherently connected to each other such as information related to a particular person, whereas each column represents a specific field such as *id*, *first\_name*, *last\_name*, *email*, etc. The structure of a simple MySQL table that contains person's general information may look something like this:

+----+------------+-----------+----------------------+

| id | first\_name | last\_name | email |

+----+------------+-----------+----------------------+

| 1 | Peter | Parker | peterparker@mail.com |

| 2 | John | Rambo | johnrambo@mail.com |

| 3 | Clark | Kent | clarkkent@mail.com |

| 4 | John | Carter | johncarter@mail.com |

With SQL you can perform any database-related task, such as creating databases and tables, saving data in database tables, query a database for specific records, deleting and updating data in databases.

Look at the following standard SQL query that returns the email address of a person whose first name is equal to 'Peter' in the *persons* table:

SELECT email FROM persons WHERE first\_name="Peter"

If you execute the SQL query above it will return the following record:

peterparker@mail.com

# **9.2 PHP & MySQL Connection:**

In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server. PHP offers two different ways to connect to MySQL server: **MySQLi** (Improved MySQL) and **PDO** (PHP Data Objects) extensions.

While the PDO extension is more portable and supports more than twelve different databases, MySQLi extension as the name suggests supports MySQL database only. MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server. Both PDO and MySQLi offer an object-oriented API, but MySQLi also offers a procedural API which is relatively easy for beginners to understand.

## Connecting to MySQL Database Server:

In PHP you can easily do this using the mysqli\_connect() function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi extensions:

#### **Syntax: MySQLi, Procedural way**

$link = mysqli\_connect("hostname", "username", "password", "database");

The *hostname* parameter in the above syntax specify the host name (e.g. localhost), or IP address of the MySQL server, whereas the *username* and *password* parameters specifies the credentials to access MySQL server, and the *database* parameter, if provided will specify the default MySQL database to be used when performing queries.

The following example shows how to connect to MySQL database server using MySQLi in procedural way.

**Syntax:**

<?php

/\* Attempt MySQL server connection. Assuming you are running MySQL

server with default setting (user 'root' with no password) \*/

$link = mysqli\_connect("localhost", "root", "");

// Check connection

if($link === false){

die("ERROR: Could not connect. " . mysqli\_connect\_error());

}

// Print host information

echo "Connect Successfully. Host info: " . mysqli\_get\_host\_info($link);

?>

Note: The default username for MySQL database server is root and there is no password. However to prevent your databases from intrusion and unauthorized access you should set password for MySQL accounts.

## Closing the MySQL Database Server Connection

The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends. However, if you want to close it earlier you can do this by simply calling the PHP mysqli\_close() function.

**SYNTAX:**

<?php

/\* Attempt MySQL server connection. Assuming you are running MySQL

server with default setting (user 'root' with no password) \*/

$link = mysqli\_connect("localhost", "root", "");

// Check connection

if($link === false){

die("ERROR: Could not connect. " . mysqli\_connect\_error());

}

// Print host information

echo "Connect Successfully. Host info: " . mysqli\_get\_host\_info($link);

// Close connection

mysqli\_close($link);

?>

# **9.3 PHP CRUD Operations:**

### CRUD is an acronym, it stands for:

**C  –** Create OR Insert data to MySQL Database.

**R –** Read Database Records.

**U –** Update Selected MySQL Records

**D –** Delete Selected Record From MySQL Database.

**Step 1: Setup the Environment & Create Database:**

1. Install Apache and MySQL server on your local machine.
2. Create a new database in MySQL using the following SQL command:

CREATE DATABASE **demo**;

USE **demo**;

## Step 2: Creating the Database Table:

Execute the following SQL query to create a table named *employees* inside your MySQL database. We will use this table for all of our future operations.

CREATE TABLE employees (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

salary INT(10) NOT NULL

);

## Step 3: Creating the Config File:

After creating the table, we need to create a PHP script in order to connect to the MySQL database server. Let's create a file named "config.php" and put the following code inside it.

We'll later include this config file in other pages using the PHP require\_once() function.

<?php

/\* Database credentials. Assuming you are running MySQL

server with default setting (user 'root' with no password) \*/

define('DB\_SERVER', 'localhost');

define('DB\_USERNAME', 'root');

define('DB\_PASSWORD', '');

define('DB\_NAME', 'demo');

/\* Attempt to connect to MySQL database \*/

$link = mysqli\_connect(DB\_SERVER, DB\_USERNAME, DB\_PASSWORD, DB\_NAME);

// Check connection

if($link === false){

die("ERROR: Could not connect. " . mysqli\_connect\_error());

}

?>

## Step 4: Creating the Landing Page:

First, we will create a landing page for our CRUD application that contains a data grid showing the records from the *employees* database table. It also has action icons for each record displayed in the grid, that you may choose to view its details, update it, or delete it.

We'll also add a create button on the top of the data grid that can be used for creating new records in the *employees* table. Create a file named "index.php" and put the following code in it:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Dashboard</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

table tr td:last-child{

width: 120px;

}

</style>

<script>

$(document).ready(function(){

$('[data-toggle="tooltip"]').tooltip();

});

</script>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<div class="mt-5 mb-3 clearfix">

<h2 class="pull-left">Employees Details</h2>

<a href="create.php" class="btn btn-success pull-right"><i class="fa fa-plus"></i> Add New Employee</a>

</div>

<?php

// Include config file

require\_once "config.php";

// Attempt select query execution

$sql = "SELECT \* FROM employees";

if($result = mysqli\_query($link, $sql)){

if(mysqli\_num\_rows($result) > 0){

echo '<table class="table table-bordered table-striped">';

echo "<thead>";

echo "<tr>";

echo "<th>#</th>";

echo "<th>Name</th>";

echo "<th>Address</th>";

echo "<th>Salary</th>";

echo "<th>Action</th>";

echo "</tr>";

echo "</thead>";

echo "<tbody>";

while($row = mysqli\_fetch\_array($result)){

echo "<tr>";

echo "<td>" . $row['id'] . "</td>";

echo "<td>" . $row['name'] . "</td>";

echo "<td>" . $row['address'] . "</td>";

echo "<td>" . $row['salary'] . "</td>";

echo "<td>";

echo '<a href="read.php?id='. $row['id'] .'" class="mr-3" title="View Record" data-toggle="tooltip"><span class="fa fa-eye"></span></a>';

echo '<a href="update.php?id='. $row['id'] .'" class="mr-3" title="Update Record" data-toggle="tooltip"><span class="fa fa-pencil"></span></a>';

echo '<a href="delete.php?id='. $row['id'] .'" title="Delete Record" data-toggle="tooltip"><span class="fa fa-trash"></span></a>';

echo "</td>";

echo "</tr>";

}

echo "</tbody>";

echo "</table>";

// Free result set

mysqli\_free\_result($result);

} else{

echo '<div class="alert alert-danger"><em>No records were found.</em></div>';

}

} else{

echo "Oops! Something went wrong. Please try again later.";

}

// Close connection

mysqli\_close($link);

?>

</div>

</div>

</div>

</div>

</body>

</html>

## Step 5: Creating the Create Page:

In this section we'll build the **C**reate functionality of our CRUD application.

Let's create a file named "create.php" and put the following code inside it. It will generate a web form that can be used to insert records in the *employees* table.

<?php

// Include config file

require\_once "config.php";

// Define variables and initialize with empty values

$name = $address = $salary = "";

$name\_err = $address\_err = $salary\_err = "";

// Processing form data when form is submitted

if($\_SERVER["REQUEST\_METHOD"] == "POST"){

// Validate name

$input\_name = trim($\_POST["name"]);

if(empty($input\_name)){

$name\_err = "Please enter a name.";

} elseif(!filter\_var($input\_name, FILTER\_VALIDATE\_REGEXP, array("options"=>array("regexp"=>"/^[a-zA-Z\s]+$/")))){

$name\_err = "Please enter a valid name.";

} else{

$name = $input\_name;

}

// Validate address

$input\_address = trim($\_POST["address"]);

if(empty($input\_address)){

$address\_err = "Please enter an address.";

} else{

$address = $input\_address;

}

// Validate salary

$input\_salary = trim($\_POST["salary"]);

if(empty($input\_salary)){

$salary\_err = "Please enter the salary amount.";

} elseif(!ctype\_digit($input\_salary)){

$salary\_err = "Please enter a positive integer value.";

} else{

$salary = $input\_salary;

}

// Check input errors before inserting in database

if(empty($name\_err) && empty($address\_err) && empty($salary\_err)){

// Prepare an insert statement

$sql = "INSERT INTO employees (name, address, salary) VALUES (?, ?, ?)";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "sss", $param\_name, $param\_address, $param\_salary);

// Set parameters

$param\_name = $name;

$param\_address = $address;

$param\_salary = $salary;

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

// Records created successfully. Redirect to landing page

header("location: index.php");

exit();

} else{

echo "Oops! Something went wrong. Please try again later.";

}

}

// Close statement

mysqli\_stmt\_close($stmt);

}

// Close connection

mysqli\_close($link);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Create Record</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<h2 class="mt-5">Create Record</h2>

<p>Please fill this form and submit to add employee record to the database.</p>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label>Name</label>

<input type="text" name="name" class="form-control <?php echo (!empty($name\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $name; ?>">

<span class="invalid-feedback"><?php echo $name\_err;?></span>

</div>

<div class="form-group">

<label>Address</label>

<textarea name="address" class="form-control <?php echo (!empty($address\_err)) ? 'is-invalid' : ''; ?>"><?php echo $address; ?></textarea>

<span class="invalid-feedback"><?php echo $address\_err;?></span>

</div>

<div class="form-group">

<label>Salary</label>

<input type="text" name="salary" class="form-control <?php echo (!empty($salary\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $salary; ?>">

<span class="invalid-feedback"><?php echo $salary\_err;?></span>

</div>

<input type="submit" class="btn btn-primary" value="Submit">

<a href="index.php" class="btn btn-secondary ml-2">Cancel</a>

</form>

</div>

</div>

</div>

</div>

</body>

</html>

The same "create.php" file will display the HTML form and process the submitted form data. It will also perform basic validation on user inputs (*line no-11 to 37*) before saving the data.

## Step 6: Creating the Read Page:

Now it's time to build the **R**ead functionality of our CRUD application.

Let's create a file named "read.php" and put the following code inside it. It will simply retrieve the records from the *employees* table based the id attribute of the employee.

<?php

// Check existence of id parameter before processing further

if(isset($\_GET["id"]) && !empty(trim($\_GET["id"]))){

// Include config file

require\_once "config.php";

// Prepare a select statement

$sql = "SELECT \* FROM employees WHERE id = ?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "i", $param\_id);

// Set parameters

$param\_id = trim($\_GET["id"]);

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

$result = mysqli\_stmt\_get\_result($stmt);

if(mysqli\_num\_rows($result) == 1){

/\* Fetch result row as an associative array. Since the result set

contains only one row, we don't need to use while loop \*/

$row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

// Retrieve individual field value

$name = $row["name"];

$address = $row["address"];

$salary = $row["salary"];

} else{

// URL doesn't contain valid id parameter. Redirect to error page

header("location: error.php");

exit();

}

} else{

echo "Oops! Something went wrong. Please try again later.";

}

}

// Close statement

mysqli\_stmt\_close($stmt);

// Close connection

mysqli\_close($link);

} else{

// URL doesn't contain id parameter. Redirect to error page

header("location: error.php");

exit();

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>View Record</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<h1 class="mt-5 mb-3">View Record</h1>

<div class="form-group">

<label>Name</label>

<p><b><?php echo $row["name"]; ?></b></p>

</div>

<div class="form-group">

<label>Address</label>

<p><b><?php echo $row["address"]; ?></b></p>

</div>

<div class="form-group">

<label>Salary</label>

<p><b><?php echo $row["salary"]; ?></b></p>

</div>

<p><a href="index.php" class="btn btn-primary">Back</a></p>

</div>

</div>

</div>

</div>

</body>

</html>

## Step 7: Creating the Update Page:

Similarly, we can build the **U**pdate functionality of our CRUD application.

Let's create a file named "update.php" and put the following code inside it. It will update the existing records in the *employees* table based the id attribute of the employee.

<?php

// Include config file

require\_once "config.php";

// Define variables and initialize with empty values

$name = $address = $salary = "";

$name\_err = $address\_err = $salary\_err = "";

// Processing form data when form is submitted

if(isset($\_POST["id"]) && !empty($\_POST["id"])){

// Get hidden input value

$id = $\_POST["id"];

// Validate name

$input\_name = trim($\_POST["name"]);

if(empty($input\_name)){

$name\_err = "Please enter a name.";

} elseif(!filter\_var($input\_name, FILTER\_VALIDATE\_REGEXP, array("options"=>array("regexp"=>"/^[a-zA-Z\s]+$/")))){

$name\_err = "Please enter a valid name.";

} else{

$name = $input\_name;

}

// Validate address address

$input\_address = trim($\_POST["address"]);

if(empty($input\_address)){

$address\_err = "Please enter an address.";

} else{

$address = $input\_address;

}

// Validate salary

$input\_salary = trim($\_POST["salary"]);

if(empty($input\_salary)){

$salary\_err = "Please enter the salary amount.";

} elseif(!ctype\_digit($input\_salary)){

$salary\_err = "Please enter a positive integer value.";

} else{

$salary = $input\_salary;

}

// Check input errors before inserting in database

if(empty($name\_err) && empty($address\_err) && empty($salary\_err)){

// Prepare an update statement

$sql = "UPDATE employees SET name=?, address=?, salary=? WHERE id=?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "sssi", $param\_name, $param\_address, $param\_salary, $param\_id);

// Set parameters

$param\_name = $name;

$param\_address = $address;

$param\_salary = $salary;

$param\_id = $id;

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

// Records updated successfully. Redirect to landing page

header("location: index.php");

exit();

} else{

echo "Oops! Something went wrong. Please try again later.";

}

}

// Close statement

mysqli\_stmt\_close($stmt);

}

// Close connection

mysqli\_close($link);

} else{

// Check existence of id parameter before processing further

if(isset($\_GET["id"]) && !empty(trim($\_GET["id"]))){

// Get URL parameter

$id = trim($\_GET["id"]);

// Prepare a select statement

$sql = "SELECT \* FROM employees WHERE id = ?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "i", $param\_id);

// Set parameters

$param\_id = $id;

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

$result = mysqli\_stmt\_get\_result($stmt);

if(mysqli\_num\_rows($result) == 1){

/\* Fetch result row as an associative array. Since the result set

contains only one row, we don't need to use while loop \*/

$row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

// Retrieve individual field value

$name = $row["name"];

$address = $row["address"];

$salary = $row["salary"];

} else{

// URL doesn't contain valid id. Redirect to error page

header("location: error.php");

exit();

}

} else{

echo "Oops! Something went wrong. Please try again later.";

}

}

// Close statement

mysqli\_stmt\_close($stmt);

// Close connection

mysqli\_close($link);

} else{

// URL doesn't contain id parameter. Redirect to error page

header("location: error.php");

exit();

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Update Record</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<h2 class="mt-5">Update Record</h2>

<p>Please edit the input values and submit to update the employee record.</p>

<form action="<?php echo htmlspecialchars(basename($\_SERVER['REQUEST\_URI'])); ?>" method="post">

<div class="form-group">

<label>Name</label>

<input type="text" name="name" class="form-control <?php echo (!empty($name\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $name; ?>">

<span class="invalid-feedback"><?php echo $name\_err;?></span>

</div>

<div class="form-group">

<label>Address</label>

<textarea name="address" class="form-control <?php echo (!empty($address\_err)) ? 'is-invalid' : ''; ?>"><?php echo $address; ?></textarea>

<span class="invalid-feedback"><?php echo $address\_err;?></span>

</div>

<div class="form-group">

<label>Salary</label>

<input type="text" name="salary" class="form-control <?php echo (!empty($salary\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $salary; ?>">

<span class="invalid-feedback"><?php echo $salary\_err;?></span>

</div>

<input type="hidden" name="id" value="<?php echo $id; ?>"/>

<input type="submit" class="btn btn-primary" value="Submit">

<a href="index.php" class="btn btn-secondary ml-2">Cancel</a>

</form>

</div>

</div>

</div>

</div>

</body>

</html>

## Step 8: Creating the Delete Page:

Finally, we will build the **D**elete functionality of our CRUD application.

Let's create a file named "delete.php" and put the following code inside it. It will delete the existing records from the *employees* table based the id attribute of the employee.

<?php

// Process delete operation after confirmation

if(isset($\_POST["id"]) && !empty($\_POST["id"])){

// Include config file

require\_once "config.php";

// Prepare a delete statement

$sql = "DELETE FROM employees WHERE id = ?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "i", $param\_id);

// Set parameters

$param\_id = trim($\_POST["id"]);

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

// Records deleted successfully. Redirect to landing page

header("location: index.php");

exit();

} else{

echo "Oops! Something went wrong. Please try again later.";

}

}

// Close statement

mysqli\_stmt\_close($stmt);

// Close connection

mysqli\_close($link);

} else{

// Check existence of id parameter

if(empty(trim($\_GET["id"]))){

// URL doesn't contain id parameter. Redirect to error page

header("location: error.php");

exit();

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Delete Record</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<h2 class="mt-5 mb-3">Delete Record</h2>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="alert alert-danger">

<input type="hidden" name="id" value="<?php echo trim($\_GET["id"]); ?>"/>

<p>Are you sure you want to delete this employee record?</p>

<p>

<input type="submit" value="Yes" class="btn btn-danger">

<a href="index.php" class="btn btn-secondary">No</a>

</p>

</div>

</form>

</div>

</div>

</div>

</div>

</body>

</html>

## Step 9: Creating the Error Page:

At the end, let's create one more file "error.php". This page will be displayed if request is invalid i.e. if id parameter is missing from the URL query string or it is not valid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Error</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

.wrapper{

width: 600px;

margin: 0 auto;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="container-fluid">

<div class="row">

<div class="col-md-12">

<h2 class="mt-5 mb-3">Invalid Request</h2>

<div class="alert alert-danger">Sorry, you've made an invalid request. Please <a href="index.php" class="alert-link">go back</a> and try again.</div>

</div>

</div>

</div>

</div>

</body>

</html>

After a long journey finally we've finished our CRUD application with PHP and MySQL. You can run it through browser.

**9.4 PHP Signup & Login:**

User authentication is very common in modern web application. It is a security mechanism that is used to restrict unauthorized access to member-only areas and tools on a site.

Following are the steps to create login and signup pages, using PHP and MySql.

## Building the Registration System

In this section we'll build a registration system that allows users to create a new account by filling out a web form. But, first we need to create a table that will hold all the user data.

### **Step 1: Creating the Database Table**

Execute the following SQL query to create the *users* table inside your MySQL database.

CREATE TABLE users (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

### **Step 2: Creating the Config File**

After creating the table, we need create a PHP script in order to connect to the MySQL database server. Let's create a file named "config.php" and put the following code inside it.

<?php

/\* Database credentials. Assuming you are running MySQL

server with default setting (user 'root' with no password) \*/

define('DB\_SERVER', 'localhost');

define('DB\_USERNAME', 'root');

define('DB\_PASSWORD', '');

define('DB\_NAME', 'demo');

/\* Attempt to connect to MySQL database \*/

$link = mysqli\_connect(DB\_SERVER, DB\_USERNAME, DB\_PASSWORD, DB\_NAME);

// Check connection

if($link === false){

die("ERROR: Could not connect. " . mysqli\_connect\_error());

}

?>

### **Step 3: Creating the Registration Form For Signup:**

Let's create another PHP file "register.php" and put the following example code in it. This example code will create a web form that allows user to register themselves.

This script will also generate errors if a user tries to submit the form without entering any value, or if username entered by the user is already taken by another user.

<?php

// Include config file

require\_once "config.php";

// Define variables and initialize with empty values

$username = $password = $confirm\_password = "";

$username\_err = $password\_err = $confirm\_password\_err = "";

// Processing form data when form is submitted

if($\_SERVER["REQUEST\_METHOD"] == "POST"){

// Validate username

if(empty(trim($\_POST["username"]))){

$username\_err = "Please enter a username.";

} elseif(!preg\_match('/^[a-zA-Z0-9\_]+$/', trim($\_POST["username"]))){

$username\_err = "Username can only contain letters, numbers, and underscores.";

} else{

// Prepare a select statement

$sql = "SELECT id FROM users WHERE username = ?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "s", $param\_username);

// Set parameters

$param\_username = trim($\_POST["username"]);

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

/\* store result \*/

mysqli\_stmt\_store\_result($stmt);

if(mysqli\_stmt\_num\_rows($stmt) == 1){

$username\_err = "This username is already taken.";

} else{

$username = trim($\_POST["username"]);

}

} else{

echo "Oops! Something went wrong. Please try again later.";

}

// Close statement

mysqli\_stmt\_close($stmt);

}

}

// Validate password

if(empty(trim($\_POST["password"]))){

$password\_err = "Please enter a password.";

} elseif(strlen(trim($\_POST["password"])) < 6){

$password\_err = "Password must have atleast 6 characters.";

} else{

$password = trim($\_POST["password"]);

}

// Validate confirm password

if(empty(trim($\_POST["confirm\_password"]))){

$confirm\_password\_err = "Please confirm password.";

} else{

$confirm\_password = trim($\_POST["confirm\_password"]);

if(empty($password\_err) && ($password != $confirm\_password)){

$confirm\_password\_err = "Password did not match.";

}

}

// Check input errors before inserting in database

if(empty($username\_err) && empty($password\_err) && empty($confirm\_password\_err)){

// Prepare an insert statement

$sql = "INSERT INTO users (username, password) VALUES (?, ?)";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "ss", $param\_username, $param\_password);

// Set parameters

$param\_username = $username;

$param\_password = password\_hash($password, PASSWORD\_DEFAULT); // Creates a password hash

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

// Redirect to login page

header("location: login.php");

} else{

echo "Oops! Something went wrong. Please try again later.";

}

// Close statement

mysqli\_stmt\_close($stmt);

}

}

// Close connection

mysqli\_close($link);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Sign Up</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

body{ font: 14px sans-serif; }

.wrapper{ width: 360px; padding: 20px; }

</style>

</head>

<body>

<div class="wrapper">

<h2>Sign Up</h2>

<p>Please fill this form to create an account.</p>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label>Username</label>

<input type="text" name="username" class="form-control <?php echo (!empty($username\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $username; ?>">

<span class="invalid-feedback"><?php echo $username\_err; ?></span>

</div>

<div class="form-group">

<label>Password</label>

<input type="password" name="password" class="form-control <?php echo (!empty($password\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $password; ?>">

<span class="invalid-feedback"><?php echo $password\_err; ?></span>

</div>

<div class="form-group">

<label>Confirm Password</label>

<input type="password" name="confirm\_password" class="form-control <?php echo (!empty($confirm\_password\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $confirm\_password; ?>">

<span class="invalid-feedback"><?php echo $confirm\_password\_err; ?></span>

</div>

<div class="form-group">

<input type="submit" class="btn btn-primary" value="Submit">

<input type="reset" class="btn btn-secondary ml-2" value="Reset">

</div>

<p>Already have an account? <a href="login.php">Login here</a>.</p>

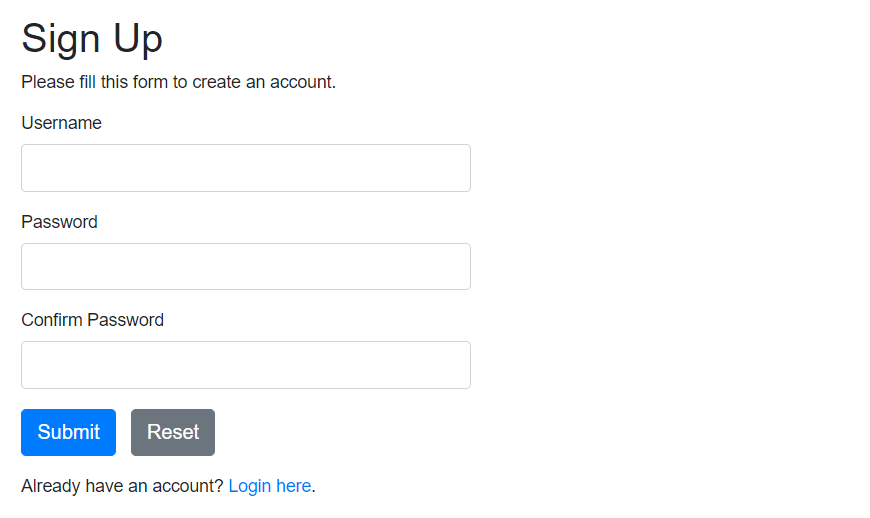
</form>

</div>

</body>

</html>

— The output of the above example (i.e. signup form) will look something like this:



In the above example, we have used the PHP's inbuilt password\_hash() function to create a password hash from the password string entered by the user (*line no-78*). This function creates a password hash using a strong one-way hashing algorithm. It also generates and applies a random salt automatically when hashing the password; this basically means that even if two users have the same passwords, their password hashes will be different.

At the time of login we'll verify the given password with the password hash stored in the database using the PHP password\_verify() function, as demonstrated in the next example.

NOTE: Password salting is a technique which is widely used to secure passwords by randomizing password hashes, so that if two users have the same password, they will not have the same password hashes. This is done by appending or prepending a random string, called a salt, to the password before hashing.

## Step 4: Creating the Login page:

In this section we'll create a login form where user can enter their username and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the username and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

Let's create a file named "login.php" and place the following code inside it.

<?php

// Initialize the session

session\_start();

// Check if the user is already logged in, if yes then redirect him to welcome page

if(isset($\_SESSION["loggedin"]) && $\_SESSION["loggedin"] === true){

header("location: welcome.php");

exit;

}

// Include config file

require\_once "config.php";

// Define variables and initialize with empty values

$username = $password = "";

$username\_err = $password\_err = $login\_err = "";

// Processing form data when form is submitted

if($\_SERVER["REQUEST\_METHOD"] == "POST"){

// Check if username is empty

if(empty(trim($\_POST["username"]))){

$username\_err = "Please enter username.";

} else{

$username = trim($\_POST["username"]);

}

// Check if password is empty

if(empty(trim($\_POST["password"]))){

$password\_err = "Please enter your password.";

} else{

$password = trim($\_POST["password"]);

}

// Validate credentials

if(empty($username\_err) && empty($password\_err)){

// Prepare a select statement

$sql = "SELECT id, username, password FROM users WHERE username = ?";

if($stmt = mysqli\_prepare($link, $sql)){

// Bind variables to the prepared statement as parameters

mysqli\_stmt\_bind\_param($stmt, "s", $param\_username);

// Set parameters

$param\_username = $username;

// Attempt to execute the prepared statement

if(mysqli\_stmt\_execute($stmt)){

// Store result

mysqli\_stmt\_store\_result($stmt);

// Check if username exists, if yes then verify password

if(mysqli\_stmt\_num\_rows($stmt) == 1){

// Bind result variables

mysqli\_stmt\_bind\_result($stmt, $id, $username, $hashed\_password);

if(mysqli\_stmt\_fetch($stmt)){

if(password\_verify($password, $hashed\_password)){

// Password is correct, so start a new session

session\_start();

// Store data in session variables

$\_SESSION["loggedin"] = true;

$\_SESSION["id"] = $id;

$\_SESSION["username"] = $username;

// Redirect user to welcome page

header("location: welcome.php");

} else{

// Password is not valid, display a generic error message

$login\_err = "Invalid username or password.";

}

}

} else{

// Username doesn't exist, display a generic error message

$login\_err = "Invalid username or password.";

}

} else{

echo "Oops! Something went wrong. Please try again later.";

}

// Close statement

mysqli\_stmt\_close($stmt);

}

}

// Close connection

mysqli\_close($link);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Login</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

body{ font: 14px sans-serif; }

.wrapper{ width: 360px; padding: 20px; }

</style>

</head>

<body>

<div class="wrapper">

<h2>Login</h2>

<p>Please fill in your credentials to login.</p>

<?php

if(!empty($login\_err)){

echo '<div class="alert alert-danger">' . $login\_err . '</div>';

}

?>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label>Username</label>

<input type="text" name="username" class="form-control <?php echo (!empty($username\_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $username; ?>">

<span class="invalid-feedback"><?php echo $username\_err; ?></span>

</div>

<div class="form-group">

<label>Password</label>

<input type="password" name="password" class="form-control <?php echo (!empty($password\_err)) ? 'is-invalid' : ''; ?>">

<span class="invalid-feedback"><?php echo $password\_err; ?></span>

</div>

<div class="form-group">

<input type="submit" class="btn btn-primary" value="Login">

</div>

<p>Don't have an account? <a href="register.php">Sign up now</a>.</p>

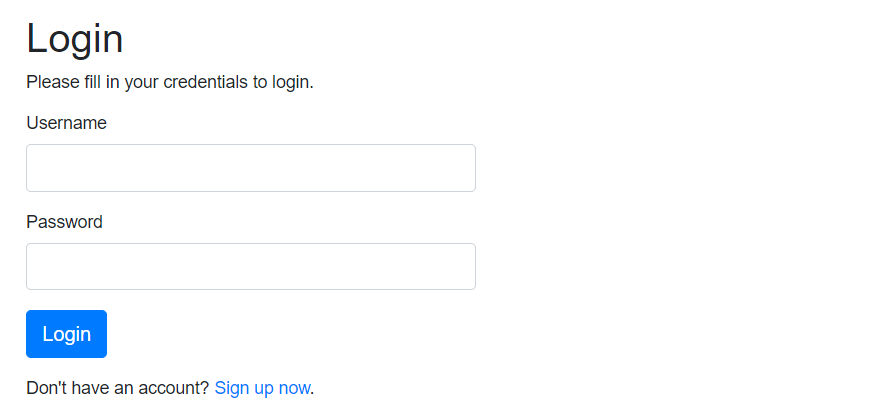
</form>

</div>

</body>

</html>

— The output of the above example (i.e. login form) will look something like this:



# **9.5 PHP Cookies & Session:**

## What is a Cookie

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visit the website next time.

Note: Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

## Setting a Cookie in PHP

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set. The basic syntax of this function can be given with:

setcookie(*name*, *value*, *expire*, *path*, *domain*, *secure*);

The parameters of the setcookie() function have the following meanings:

| **Parameter** | **Description** |
| --- | --- |
| name | The name of the cookie. |
| value | The value of the cookie. Do not store sensitive information since this value is stored on the user's computer. |
| expires | The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0. |
| path | Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain. |
| domain | Specify the domain for which the cookie is available to e.g www.example.com. |
| secure | This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists. |

NOTE: If the expiration time of the cookie is set to 0, or omitted, the cookie will expire at the end of the session i.e. when the browser closes.

Here's an example that uses setcookie() function to create a cookie named username and assign the value value John Carter to it. It also specify that the cookie will expire after 30 days (30 days \* 24 hours \* 60 min \* 60 sec).

<?php

// Setting a cookie

setcookie("username", "John Carter", time()+30\*24\*60\*60);

?>

## Accessing Cookies Values

The PHP $\_COOKIE superglobal variable is used to retrieve a cookie value. It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name. The individual cookie value can be accessed using standard array notation, for example to display the username cookie set in the previous example, you could use the following code.

<?php

// Accessing an individual cookie value

echo $\_COOKIE["username"];

?>

The PHP code in the above example produce the following output.

John Carter

It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:

<?php

// Verifying whether a cookie is set or not

if(isset($\_COOKIE["username"])){

echo "Hi " . $\_COOKIE["username"];

} else{

echo "Welcome Guest!";

}

?>

## Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past, as shown in the example below:

<?php

// Deleting a cookie

setcookie("username", "", time()-3600);

?>

**What is a Session**

Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.

Also every time the browser requests a URL to the server, all the cookie data for a website is automatically sent to the server within the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

NOTE: The session IDs are randomly generated by the PHP engine which is almost impossible to guess. Furthermore, because the session data is stored on the server, it doesn't have to be sent with every browser request.

## Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP session\_start() function. It will create a new session and generate a unique session ID for the user.

The PHP code in the example below simply starts a new session.

<?php

// Starting session

session\_start();

?>

The session\_start() function first checks to see if a session already exists by looking for the presence of a session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

Note: You must call the session\_start() function at the beginning of the page i.e. before any output generated by your script in the browser, much like you do while setting the cookies with setcookie() function.

## Storing and Accessing Session Data

You can store all your session data as key-value pairs in the $\_SESSION[] superglobal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

<?php

// Starting session

session\_start();

// Storing session data

$\_SESSION["firstname"] = "Peter";

$\_SESSION["lastname"] = "Parker";

?>

To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling session\_start() and then pass the corresponding key to the $\_SESSION associative array.

<?php

// Starting session

session\_start();

// Accessing session data

echo 'Hi, ' . $\_SESSION["firstname"] . ' ' . $\_SESSION["lastname"];

?>

The PHP code in the example above produce the following output.

Hi, Peter Parker

## Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $\_SESSION associative array, as shown in the following example:

<?php

// Starting session

session\_start();

// Removing session data

if(isset($\_SESSION["lastname"])){

unset($\_SESSION["lastname"]);

}

?>

However, to destroy a session completely, simply call the session\_destroy() function. This function does not need any argument and a single call destroys all the session data.

<?php

// Starting session

session\_start();

// Destroying session

session\_destroy();

?>

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of session.gc\_maxlifetime variable in the PHP configuration file (php.ini).

***Tasks***

1. Implement a basic PHP CRUD system using MySQL database and PHP. Test the system by performing CRUD operations on the database.
2. Add login functionality to the CRUD system using session management. Create a login form that allows users to enter their username and password. If the credentials match with the database, the user should be redirected to the dashboard. Otherwise, an error message should be displayed.
3. Implement a user registration system using PHP and MySQL. Create a registration form that allows users to enter their details such as name, email, and password. Once the user submits the form, validate the inputs and store them in the database.
4. Add cookies functionality to the CRUD system. Use cookies to remember the user's preferences or last visited page.
5. Create a user profile page that displays the user's information such as name, email, and profile picture. Allow the user to edit their profile details and upload a new profile picture.
6. Implement a search functionality to the CRUD system using PHP and MySQL. Create a search form that allows users to search for data in the database. The search results should be displayed in a table format and should include pagination