Chapter 7

Database Connectivity in PHP

5.1 Introduction to SQL

SQL (Structured Query Language) is a standardized programming language specifically designed for managing and manipulating relational databases. SQL enables users to perform various operations on the data stored in databases, such as querying, updating, inserting, and deleting data.

- **Database:** A structured collection of data stored in an organized manner.
- **Table:** A collection of related data entries, consisting of rows and columns.
- **Row:** A single data entry in a table.
- **Column:** A single field in a table that holds specific types of data.
- **Primary Key:** A unique identifier for each row in a table.
- **Foreign Key:** A field in one table that uniquely identifies a row in another table, establishing a relationship between the two tables.

5.2 Basic SQL Commands (CRUD)

CRUD stands for Create, Read, Update, and Delete. These are the four basic operations you can perform on data in a database.

1. Create (INSERT)

The INSERT statement is used to add new records to a table. You specify the table name and the columns for which you want to insert values. The values to be inserted are provided in a corresponding order.

Syntax:

INSERT INTO table_name (column1, column2, ...)

VALUES (value1, value2, ...);

Example:

INSERT INTO users (name, age)

VALUES ('Alice', 30);

This command adds a new row to the users table with the name 'Alice' and age 30.

2. Read (SELECT)

The SELECT statement is used to retrieve data from a database. You specify the columns you want to retrieve and the table from which to retrieve them. Conditions can be added to filter the data.

Syntax:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Example:

SELECT name, age

FROM users

WHERE age > 25;

This command retrieves the name and age columns from the users table for all rows where the age is greater than 25.

3. Update (UPDATE)

The UPDATE statement is used to modify existing records in a table. You specify the table name, the columns to update, and the new values. A condition is usually added to specify which records should be updated.

Syntax:

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Example:

UPDATE users

SET age = 31

WHERE name = 'Alice';

This command updates the age of the user named 'Alice' to 31.

4. Delete (DELETE)

The DELETE statement is used to remove records from a table. You specify the table name and add a condition to indicate which records should be deleted.

Syntax:

DELETE FROM table_name

WHERE condition;

Example:

DELETE FROM users

WHERE name = 'Alice';

This command deletes the row from the users table where the name is 'Alice'.

5.3 HTML Forms and Methods

HTML forms are essential for collecting user input and sending it to a server for processing. They provide a user-friendly interface for data entry and can use various methods to transmit this data.

HTML Form

An HTML form is created using the <form> element. Inside the form, various input elements are used to collect user data, such as text fields, checkboxes, radio buttons, and submit buttons.

Syntax:

<form action="URL" method="METHOD">

  <!-- Form elements like input, select, textarea, etc. -->

</form>

Form Elements

1. **Text Input**: Collects a single line of text.

   <input type="text" name="username" placeholder="Enter your username">

2. **Password Input**: Collects a password (hidden text).

   <input type="password" name="password" placeholder="Enter your password">

3. **Radio Button**: Allows the user to select one option from a set.

   <input type="radio" name="gender" value="male"> Male

   <input type="radio" name="gender" value="female"> Female

4. **Checkbox**: Allows the user to select multiple options.

<input type="checkbox" name="vehicle" value="car"> Car

<input type="checkbox" name="vehicle" value="bike"> Bike

5. **Submit Button**: Submits the form data to the server.

<input type="submit" value="Submit">

Form Methods

The method attribute specifies how to send form data to the server. There are two primary methods: GET and POST.

1. GET Method

The GET method appends form data to the URL specified in the action attribute. It is suitable for non-sensitive data and when bookmarking or sharing the URL with the query string is desirable. The data sent is visible in the URL.

Syntax:

<form action="submit_form.php" method="get">

  <!-- Form elements -->

</form>

Example:

<form action="submit_form.php" method="get">

  <input type="text" name="username" placeholder="Enter your username">

  <input type="submit" value="Submit">

</form>

Request Example:

GET /submit_form.php?username=example HTTP/1.1

2. POST Method

The POST method sends form data in the HTTP request body, making it suitable for sensitive data and large amounts of data. The data sent is not visible in the URL.

Syntax:

<form action="submit_form.php" method="post">

  <!-- Form elements -->

</form>

Example:

```
<form action="submit_form.php" method="post">
   <input type="text" name="username" placeholder="Enter your username">
   <input type="submit" value="Submit">
</form>
```

GET	POST Method in PHP
GET is a method that sends information by appending them to the page request.	POST is a method that transfers information via HTTP header.
URL	
The form information is visible in the URL	The form information is not visible in the URL
Information Amount	
Limited amount of information is sent. It is less than 1500 characters.	Unlimited amount of information is sent.
Usage	
Helps to send non-sensitive data	Helps to send sensitive data (passwords), binary data (word documents, images)and uploading files
Security	
Not very secure.	More secure.

5.4 Database Connectivity

1. Establishing a Connection

There are two primary ways to connect PHP to a MySQL database:

- Using the mysqli extension (MySQL Improved)
- Using the PDO (PHP Data Objects) extension

1. **Using mysqli Extension**

The mysqli extension is a relational database driver used in the PHP programming language to provide an interface with MySQL databases.

Example:

```
<?php
$servername = "localhost";
$username = "root"; (username)
$password = ""; (password)
$dbname = "database_name";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Using PDO Extension

PDO is a database access layer providing a uniform method of access to multiple databases.

Example:

```
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "database_name";


try {

   $conn = new PDO("mysql:host=$servername; dbname=$dbname", $username, $password);

   // set the PDO error mode to exception

   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

   echo "Connected successfully";


}

catch(PDOException $e) {

   echo "Connection failed: " . $e->getMessage();

}

?>
```

MySQL Functions

Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value. They are often used with the GROUP BY clause.

- **COUNT()**: Returns the number of rows.

```
SELECT COUNT(*) FROM employees;
```

- **SUM()**: Returns the sum of a set of values.

```
SELECT SUM(salary) FROM employees;
```

- **AVG()**: Returns the average value of a set of values.

```
SELECT AVG(salary) FROM employees;
```

- **MAX()**: Returns the maximum value.

```
SELECT MAX(salary) FROM employees;
```

- **MIN()**: Returns the minimum value.

```
SELECT MIN(salary) FROM employees;
```

String Functions

String functions perform operations on string data types.

- **CONCAT()**: Concatenates two or more strings.

  ```
  SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;
  ```

- **SUBSTRING()**: Extracts a substring from a string.

  ```
  SELECT SUBSTRING(first_name, 1, 3) FROM employees;
  ```

- **LENGTH()**: Returns the length of a string.

  ```
  SELECT LENGTH(first_name) FROM employees;
  ```

- **UPPER()**: Converts a string to uppercase.

  ```
  SELECT UPPER(first_name) FROM employees;
  ```

- **LOWER()**: Converts a string to lowercase.

SELECT LOWER(first_name) FROM employees;

Date and Time Functions

Date and time functions perform operations on date and time data types.

- **NOW()**: Returns the current date and time.

SELECT NOW();

- **CURDATE()**: Returns the current date.

SELECT CURDATE();

- **CURTIME()**: Returns the current time.

SELECT CURTIME();

- **DATE_ADD()**: Adds a specified time interval to a date.

SELECT DATE_ADD(CURDATE(), INTERVAL 7 DAY);

- **DATEDIFF()**: Returns the difference between two dates.

SELECT DATEDIFF('2024-08-06', '2024-01-01');

Numeric Functions

Numeric functions perform operations on numeric data types.

- **ABS()**: Returns the absolute value of a number.

SELECT ABS(-10);

- **ROUND()**: Rounds a number to a specified number of decimal places.

SELECT ROUND(123.456, 2);

Executing DDL with PHP

Executing DDL (Data Definition Language) queries using PHP involves creating, altering, or dropping database structures such as tables. You can use either

MySQLi or PDO (PHP Data Objects) for this purpose. Below are examples of how to execute DDL queries using both MySQLi and PDO.

Using MYSQLI

1. **Connect to the Database**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

2. Execute a DDL Query (e.g., creating a table)

```
<?php
$ddlQuery = "CREATE TABLE Students (
   id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   firstname VARCHAR(30) NOT NULL,
   lastname VARCHAR(30) NOT NULL,
   email VARCHAR(50),
```

```
    reg_date TIMESTAMP
)";

if ($conn->query($ddlQuery) === TRUE) {
    echo "Table Students created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}
?>
```

3. Close the Connection

```
<?php
$conn->close();
?>
```

DML Queries

Executing DML (Data Manipulation Language) queries using PHP involves operations like inserting, updating, and deleting data in the database. You can use either MySQLi or PDO (PHP Data Objects) to perform these operations. Below are examples of how to execute DML queries using both MySQLi and PDO.

Using MySQLi

1. **Connect to the Database**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
```

```
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

2. **Execute an INSERT Query**

```
<?php
$insertQuery = "INSERT INTO Students (firstname, lastname, email)
VALUES ('John', 'Sharma', 'john@example.com')";

if ($conn->query($insertQuery) === TRUE) {
   echo "New record created successfully";
} else {
   echo "Error: " . $insertQuery . "<br>" . $conn->error;
}
?>
```

3. **Execute an UPDATE Query**

```
<?php
$updateQuery = "UPDATE Students SET email='john.doe@example.com' WHERE firstname='John' AND lastname='Doe'";

if ($conn->query($updateQuery) === TRUE) {
   echo "Record updated successfully";
} else {
   echo "Error: " . $updateQuery . "<br>" . $conn->error;
}
?>
```

4. **Execute a DELETE Query**

```
<?php
$deleteQuery = "DELETE FROM Students WHERE firstname='John' AND lastname='Doe'";

if ($conn->query($deleteQuery) === TRUE) {
   echo "Record deleted successfully";
} else {
   echo "Error: " . $deleteQuery . "<br>" . $conn->error;
}
?>
```

5. **Close the Connection**

```
<?php
$conn->close();
?>
```

Using PDO

1. **Connect to the Database**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDatabase";

try {
   $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
   // Set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   echo "Connected successfully";
}
catch(PDOException $e) {
   echo "Connection failed: " . $e->getMessage();
}
?>
```

Execute an INSERT Query

```
<?php
try {
   $insertQuery = "INSERT INTO Students (firstname, lastname, email)
```

```
    VALUES ('John', 'Doe', 'john@example.com')";

    $conn->exec($insertQuery);
    echo "New record created successfully";
}
catch(PDOException $e) {
    echo $insertQuery . "<br>" . $e->getMessage();
}
?>
```

Execute an UPDATE Query

```
<?php
try {
    $updateQuery = "UPDATE Students SET email='john.doe@example.com'
WHERE firstname='John' AND lastname='Doe'";

    $stmt = $conn->prepare($updateQuery);
    $stmt->execute();

    echo $stmt->rowCount() . " record(s) updated successfully";
}
catch(PDOException $e) {
    echo $updateQuery . "<br>" . $e->getMessage();
}
?>
```

Execute a DELETE Query

```
<?php
try {
   $deleteQuery = "DELETE FROM Students WHERE firstname='John' AND lastname='Doe'";

   $stmt = $conn->prepare($deleteQuery);
   $stmt->execute();

   echo $stmt->rowCount() . " record(s) deleted successfully";
}
catch(PDOException $e) {
   echo $deleteQuery . "<br>" . $e->getMessage();
}
?>
```

Close the Connection

```
<?php
$conn = null;
?>
```

Register Form

```
<!DOCTYPE html>
<html>
<head>
   <title>Registration Form</title>
   <script>
      function validateForm() {
```

```
        var firstname = document.forms["registerForm"]["firstname"].value;
        var lastname = document.forms["registerForm"]["lastname"].value;
        var email = document.forms["registerForm"]["email"].value;
        var password = document.forms["registerForm"]["password"].value;
        var confirmPassword = document.forms["registerForm"]["confirmPassword"].value;

        if (firstname == "" || lastname == "" || email == "" || password == "" || confirmPassword == "") {
            alert("All fields must be filled out");
            return false;
        }

        if (password != confirmPassword) {
            alert("Passwords do not match");
            return false;
        }

        return true;
    }
  </script>
</head>
<body>
  <h2>Registration Form</h2>
  <form name="registerForm" action="register.php" method="post" onsubmit="return validateForm()">
    <label for="firstname">First Name:</label>
```

```
    <input type="text" id="firstname" name="firstname" required><br><br>

    <label for="lastname">Last Name:</label>
    <input type="text" id="lastname" name="lastname" required><br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password"
required><br><br>

    <label for="confirmPassword">Confirm Password:</label>
    <input type="password" id="confirmPassword" name="confirmPassword"
required><br><br>

    <input type="submit" value="Register">
  </form>
</body>
</html>
```

Register.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDatabase";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $firstname = $_POST['firstname'];
   $lastname = $_POST['lastname'];
   $email = $_POST['email'];
   $password = $_POST['password'];

   // Hash the password
   $hashed_password = password_hash($password, PASSWORD_DEFAULT);

   // Insert data into database
   $sql = "INSERT INTO Users (firstname, lastname, email, password) VALUES (?,
?, ?, ?)";

   $stmt = $conn->prepare($sql);
   $stmt->bind_param("ssss", $firstname, $lastname, $email,
$hashed_password);
```

```
    if ($stmt->execute() === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $stmt->close();
}

$conn->close();
?>
```

Login and Authentication

Creating a login and authentication system using PHP, HTML, and JavaScript involves several steps.

1. **Database Setup**

```
CREATE DATABASE login_system;

USE login_system;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

2. **Registration Form (register.html)**

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Register</title>
</head>
<body>
    <h2>Register</h2>
    <form action="register.php" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"
required><br><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

3. Registration Processing (register.php)

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hash the password

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

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
}
?>
```

3. **Login Form (login.html)**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
```

```
<body>
   <h2>Login</h2>
   <form action="login.php" method="post">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required><br><br>
      <label for="password">Password:</label>
      <input type="password" id="password" name="password"
required><br><br>
      <button type="submit">Login</button>
   </form>
</body>
</html>
```

4. **Login Processing (login.php)**

```
<?php
session_start();

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
```

```
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $sql = "SELECT * FROM users WHERE username='$username'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
            $_SESSION['username'] = $username;
            echo "Login successful!";
        } else {
            echo "Invalid password.";
        }
    } else {
        echo "No user found.";
    }

    $conn->close();
}
?>
```

5. Session Management and Logout (logout.php)

```
<?php
```

```
session_start();

session_unset();

session_destroy();

header("Location: login.html");

exit();

?>
```

Session and cookies:

Sessions and cookies are two ways to store data on the server and client side, respectively.

1. Using Sessions in PHP

Starting a Session

To use sessions in PHP, you need to start a session at the beginning of your script using session_start().

```
<?php

session_start(); // Start the session


// Set session variables

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

$_SESSION["email"] = "john@example.com";


// Access session variables

echo "Username: " . $_SESSION["username"];

echo "Email: " . $_SESSION["email"];

?>
```

Destroying a Session

To log out a user and destroy the session:

```php
<?php
session_start();
session_unset(); // Unset all session variables
session_destroy(); // Destroy the session
header("Location: login.html"); // Redirect to login page
exit();
?>
```

2. Using Cookies in PHP

Setting Cookies

You can set cookies using the setcookie() function. Cookies are stored on the client side.

```php
<?php
$cookie_name = "username";
$cookie_value = "john_doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day

// Check if the cookie is set
if(isset($_COOKIE[$cookie_name])) {
   echo "Cookie '" . $cookie_name . "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
} else {
   echo "Cookie is not set!";
}
?>
```

Deleting Cookies

To delete a cookie, you set its expiration date to a time in the past.

```
<?php
setcookie("username", "", time() - 3600, "/"); // Expire the cookie

// Check if the cookie is deleted
if(!isset($_COOKIE["username"])) {
   echo "Cookie 'username' is deleted.";
} else {
   echo "Cookie 'username' is not deleted.";
}
?>
```

Example: Login System with Sessions and Cookies

Login Form (login.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Login</title>
</head>
<body>
   <h2>Login</h2>
   <form action="login.php" method="post">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required><br><br>
```

```
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <label for="remember">Remember Me:</label>
        <input type="checkbox" id="remember" name="remember"><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Login Processing (login.php)

```
<?php
session_start();

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $remember = isset($_POST['remember']);

    $sql = "SELECT * FROM users WHERE username='$username'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
            $_SESSION['username'] = $username;
            echo "Login successful!";

            if ($remember) {
                setcookie("username", $username, time() + (86400 * 30), "/"); // 30
days
            }
        } else {
            echo "Invalid password.";
        }
    } else {
        echo "No user found.";
    }

    $conn->close();
}
```

```
?>
```

Accessing Session and Cookie Data

You can access the session and cookie data in other parts of your application.

```
<?php
session_start();

if (isset($_SESSION['username'])) {
    echo "Welcome, " . $_SESSION['username'] . "!";
} else if (isset($_COOKIE['username'])) {
    echo "Welcome back, " . $_COOKIE['username'] . "!";
} else {
    echo "Please log in.";
}
?>
```

Logout (logout.php)

```
<?php
session_start();
session_unset();
session_destroy();
setcookie("username", "", time() - 3600, "/"); // Delete the cookie
header("Location: login.html");
exit();
?>
```

User With logout button

1. Database Setup

Make sure you have the users table set up as described earlier.

2. Registration Form (register.html)

Allow users to register.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register</title>
    <style>
        body { font-family: Arial, sans-serif; }
        form { margin: 50px auto; width: 300px; }
        label, input { display: block; margin: 10px 0; }
    </style>
</head>
<body>
    <h2>Register</h2>
    <form action="register.php" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"
required><br><br>
        <button type="submit">Register</button>
    </form>
</body>
```

```
</html>
```

3. Registration Processing (register.php)

Handle registration.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hash the password

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

```
    if ($conn->query($sql) === TRUE) {
        echo "Registration successful. <a href='login.html'>Login here</a>";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
}
?>
```

4. Login Form (login.html)

A form for logging in.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <style>
        body { font-family: Arial, sans-serif; }
        form { margin: 50px auto; width: 300px; }
        label, input { display: block; margin: 10px 0; }
    </style>
</head>
<body>
    <h2>Login</h2>
    <form action="login.php" method="post">
```

```
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"
required><br><br>
        <label for="remember">Remember Me:</label>
        <input type="checkbox" id="remember" name="remember"><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

5. Login Processing (login.php)

Handle login and session management.

```
<?php
session_start();

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $remember = isset($_POST['remember']);

    $sql = "SELECT * FROM users WHERE username='$username'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
            $_SESSION['username'] = $username;
            if ($remember) {
                setcookie("username", $username, time() + (86400 * 30), "/"); // 30
days
            }
            header("Location: welcome.php");
            exit();
        } else {
            echo "Invalid password.";
        }
    } else {
        echo "No user found.";
    }
```

```
    $conn->close();
}
?>
```

6. Welcome Page (welcome.php)

A page users see after logging in, with a logout button.

```
<?php
session_start();

if (!isset($_SESSION['username']) && !isset($_COOKIE['username'])) {
    header("Location: login.html");
    exit();
}

$username = isset($_SESSION['username']) ? $_SESSION['username'] :
$_COOKIE['username'];
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .container { margin: 50px auto; width: 300px; }
    </style>
```

```
</head>
<body>
  <div class="container">
    <h2>Welcome, <?php echo htmlspecialchars($username); ?>!</h2>
    <form action="logout.php" method="post">
      <button type="submit">Logout</button>
    </form>
  </div>
</body>
</html>
```

7. Logout Processing (logout.php)

Handle logout and redirect to login page.

```
<?php
session_start();
session_unset();
session_destroy();
setcookie("username", "", time() - 3600, "/"); // Delete the cookie
header("Location: login.html");
exit();
?>
```

```
 ('DOMContentLoaded', loadUsers);
```