

Code Snippets and Mockup Webpage :

Machala 101036

## **Introduction**

In any web application, ensuring the security and integrity of user data is essential. From managing database connections to validating user input, every aspect of the application's functionality plays a crucial role in maintaining a secure and reliable environment. This document presents a series of code snippets designed to address key aspects of security and functionality within PHP applications.

These snippets collectively contribute to the security and functionality of my PHP application by handling database connections, session management, password validation, and user input validation. Each snippet addresses specific requirements, and provides solutions to common challenges faced in web development.

The code snippets below serve as essential tools for the security and reliability of PHP applications. From validating password complexity to checking query lengths, these snippets empower us developers to build secure, resilient applications that inspire trust and confidence among users.

Before we look at each code snippet and its corresponding functionality in detail: here is a Mockup of my one page website I have used for this project:



## Sharing Art. Finding Expression

### About Us

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras auctor pharetra sodales. Sed finibus venenatis dignissim. Etiam volutpat sed eros ac suscipit. In sed gravida sapien. Etiam eu orci nunc. Sed interdum ex a malesuada dignissim. In finibus auctor malesuada. (Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

### Services

#### Service 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### Service 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

### Gallery



### Testimonials

Lorem ipsum dolor sit amet, consectetur adipiscing elit dolor sit amet

- John Doe

Lorem ipsum dolor sit amet, consectetur adipiscing elit dolor sit amet

- John Doe

Lorem ipsum dolor sit amet, consectetur adipiscing elit dolor sit amet

- John Doe

### How Can We Help?

First Name

Last Name

Create a Username

Phone Number

Email Address

Your Query

Create a Password

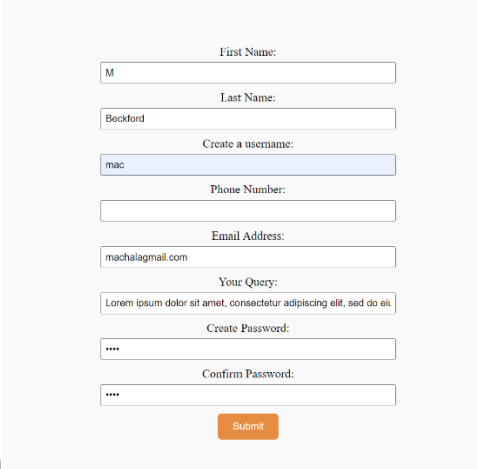
Confirm Password

submit



## 1. illustrating instances of incorrect data input on the form and the resulting error codes

a)



First Name: M

Last Name: Beckford

Create a username: mac

Phone Number:

Email Address: machalagmail.com

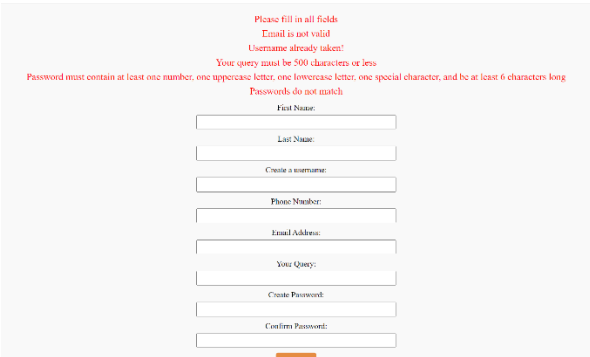
Your Query: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod

Create Password: \*\*\*\*

Confirm Password: \*\*\*\*

Submit

b)



Please fill in all fields  
Email is not valid  
Username already taken!  
Your query must be 500 characters or less  
Password must contain at least one number, one uppercase letter, one lowercase letter, one special character, and be at least 6 characters long  
Passwords do not match

First Name:

Last Name:

Create a username:

Phone Number:

Email Address:

Your Query:

Create Password:

Confirm Password:

### Image Explanations:

- a) **Incorrect Data Input on Form:** In the first image, I intentionally Each input incorrect data into each of the fields, such as invalid email addresses, passwords that do not meet complexity requirements, or queries exceeding the character limit. These errors show why it's crucial to have strong checks in place to make sure the information users enter is correct and reliable.
- b) **Resulting Error Codes:** In response to the incorrect data input, the PHP application generates corresponding error codes. These error messages indicate the specific issues encountered during data validation and provide valuable feedback to the user. By understanding these error codes, developers can identify and address potential vulnerabilities in their applications, enhancing overall security and user experience.

These images offer tangible examples of how the code snippets outlined in this document can effectively validate user input and handle errors, ultimately contributing to a more secure and reliable PHP application.

## 2.Html Contact form

```
<h2>How Can We Help <br></h2>

<div class="error_messages">
  <?php check_signup_errors(); ?>
</div>

<form class="form" action="includes/signup.inc.php" method="post">
```

```

<label class="label" for="firstname">First Name:</label>
<input type="text" id="firstname" name="firstname" >

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

<label class="label" for="username">Create a username:</label>
<input type="text" id="username" name="username" >

<label class="label" for="phone">Phone Number:</label>
<input type="text" id="phone" name="phone">

<label class="label" for="email">Email Address:</label>
<input type="text" id="email" name="email" >

<label class="label" for="usersQuery">Your Query:</label>
<input type="text" id="usersQuery" name="usersQuery">

<label class="label" for="pwd">Create Password:</label>
<input type="password" id="pwd" name="pwd">

<label class="label" for="confirm_password">Confirm Password:</label>
<input type="password" id="confirm_password" name="confirm_password">

<input type="submit" value="Submit">
</form>
</div>

```

### HTML Form for Contact Session:

This HTML snippet represents a contact form section within a webpage.

It includes input fields for the user to provide their first name, last name, username, phone number, email address, query, password, and confirm password.

Each input field is accompanied by a corresponding label for clarity.

The form is set to submit user input to the signup.inc.php file using the POST method upon submission.

### 3.Php to check errors

```

<?php

declare(strict_types=1);

function check_signup_errors(){

```

```

if(isset($_SESSION['errors_signup'])){
    $errors = $_SESSION['errors_signup'];

    echo "<br>";

    foreach ($errors as $error) {
        echo '<p class="form-error">' . $error . '</p>';
    }
    unset($_SESSION['errors_signup']);
}else if (isset($_GET['signup']) && $_GET['signup'] === "success") {

    echo '<br>';
    echo '<p class="form-success"> Query Successfully submitted!</p>';
}
}

```

### PHP Form Error Handling and Success Message Display:

This PHP snippet checks for any errors that occurred during form submission and displays them to the user.

It calls the `check_signup_errors()` function, which checks if there are any errors stored in the session and displays them as error messages.

Additionally, it checks if the signup was successful (via a GET parameter) and displays a success message if applicable.

This snippet ensures that users receive feedback on the outcome of their form submission, whether it was successful or if there were errors that need attention.

## 4.PHP Database Connection

```

<?php
$host = "localhost";
$dbname = "user_query";
$dbusername = "root";
$dbpassword = "";

try {

    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $dbusername,
$dbpassword);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed " . $e->getMessage());
}
?>

```

### PHP Database Connection:

This snippet establishes a connection to a MySQL database using PDO (PHP Data Objects), a secure and consistent way to access databases in PHP.

It specifies the host, database name, database username, and password needed for the connection.

Inside the try-catch block, it attempts to create a new PDO instance with the provided database credentials.

If the connection fails, it catches any PDOException that occurs and terminates the script, displaying the error message.

### 5. PHP Function to regenerate session id

```
function regenerate_session_id(){  
  
    regenerate_session_id();  
    $_SESSION["last_regeneration"] = time();  
}
```

#### PHP Function to Regenerate Session ID:

This function is responsible for regenerating the session ID periodically to enhance session security.

It calls the session\_regeneration\_id() function to regenerate the session ID.

It updates the session variable \$\_SESSION["last\_regeneration"] with the current timestamp to track the last time the session ID was regenerated.

### 6. Php function to create user in database

```
}function create_user(object $pdo, string $firstname , string $lastname, string  
$username, string $phone, string $email, string $usersQuery, string $pwd, string  
$confirm_password)  
{  
    set_user($pdo, $firstname , $lastname, $username, $phone, $email, $usersQuery,  
    $pwd, $confirm_password);  
}
```

#### PHP Function to Create User in Database:

This function facilitates the creation of a new user record in the database.

It calls another function, `set_user()`, passing the provided user details as parameters.

This function abstracts away the database interaction logic, allowing for cleaner and more modular code organization.

## 7. PHP Function to Validate Password Complexity

```
function validate_password(object $pdo, string $password): bool {  
    $result = preg_match('/^(?=.*\d)(?=.*[A-Z])(?=.*[a-z])(?=.*\W){6,}$/', $password);  
    return (bool)$result;  
}
```

PHP Function to Validate Password Complexity: This function validates the complexity of a password string according to specified criteria, including at least one digit, one uppercase letter, one lowercase letter, one special character, and a minimum length of six characters. It uses a regular expression to match the password against the defined criteria and returns a boolean value indicating whether the password meets the requirements.

## 8. PHP Function to Check Query Length:

```
function is_usersQuery_exceeded(object $pdo, string $usersQuery): bool {  
    $result = strlen($usersQuery) > 500;  
    return (bool)$result;  
}
```

## PHP Function to Check Query Length

This function checks whether the length of a user's query exceeds a specified limit (in this case, 500 characters). It returns true if the length of the query exceeds the limit and false otherwise, allowing for validation of user input before storing it in the database.

## 9. PHP Function to Confirm Password Match

```
function confirm_password_match(object $pdo, string $password, string  
$confirm_password): bool {  
    $result = $password === $confirm_password;  
    return (bool)$result;  
}
```

## PHP Function to Confirm Password Match:

This function compares two password strings (the original password and its confirmation) to ensure they match. It returns true if the passwords match and false otherwise, enabling validation of password input during user registration or password reset processes.