LAPORAN AKHIR

PEMROGRAMAN DAN WEB I

Nama : **Sugeng Wahyu Nugroho**

NIM : **193010503005**

Kelas : **C**

Modul : **II (FORM HANDLING)**

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKARAYA

2021

BAB I

PENDAHULUAN

1.1. Tujuan

- Mahasiswa mampu membuat handling yang mampu mengolah data dari form HTML.
- Mahasiswa mampu membuat batasan-batasan untuk menangani inputan dari form HTML.

1.2.Latar Belakang

1.2.1. Pengertian Variabel SuperHloba dalam PHP

Variabel $_GET dan $_POST (dan juga $_REQUEST) di dalam PHP termasuk ke dalam kelompok variabel yang dikenal dengan ‘Variabel SuperGlobal’.

Variabel SuperGlobals adalah variabel khusus di dalam PHP yang bisa diakses dari halaman PHP manapun tanpa perlu mendefinisikannya terlebih dahulu, dan untuk mengakses variabel ini kita juga tidak perlu menggunakan keyword global (sebagaimana variabel global pada umumnya)

Selain variabel $_GET, $_POST dan $_REQUEST, PHP masih memiliki beberapa variabel superglobal lainnya seperti $_COOKIE, $_SESSION, dan $_SERVER. Ciri khusus untuk variabel global di dalam PHP, diawali dengan tanda $_. Namun pada tutorial ini kita hanya fokus kepada variabel $_GET, $_POST dan $_REQUEST.

Variabel $_GET, $_POST dan $_REQUEST merupakan tipe data array, sehingga untuk mengakses nilainya, kita menggunakan cara akses array yakni dengan menggunakan kurung siku seperti: $_GET[‘nama’]

dimana nama adalah nilai dari atribut name pada objek form yang akan diakses.

Contoh berikut menunjukkan form HTML sederhana dengan dua field input dan tombol submit:

```
<html>
	<body>
		<form action="welcome.php" method="post">
			Name: <input type="text" name="name"><br>
			E-mail: <input type="text" name="email"><br>
			<input type="submit">
		</form>

	</body>
</html>
```

Gambar 1 *contoh form HTML sederhana*

Ketika user mengisi form, dan menekan tombol click, data form dikirim untuk memproses file PHP dengan nama “welcome.php”. Data form dikirimkan dengan method HTTP POST. Untuk menampilkan data yang sudah disubmit bisa dilakukan dengan mencetak data tersebut menggunakan perintah echo. File “welcome.php” adalah sebagai berikut:

```
<html>
	<body>
		Welcome <?php echo $_POST["name"]; ?><br>
		Your email address is: <?php echo $_POST["email"];
	?> </body>
</html>
```

Gambar 2 *contoh form HTML + PHP*

Jika field nama diinputkan dengan Tono dan email diinputkan dengan tono@mail.com maka output yang akan tampil adalah sebagai berikut: Welcome Budi Your email address is tono@mail.com Hasil yang sama juga akan tampil dengan menggunakan method get sebagai berikut:

```
<html>
	<body>

		<form action="welcome_get.php" method="get">
			Name: <input type="text" name="name"><br>
			E-mail: <input type="text" name="email"><br>
			<input type="submit">
		</form>
	</body>
</html>
```

Gambar 3 *contoh form dengan methode get*

1.2.2. Perbedaan Variabel global $_GET, $_POST dan $_REQUEST

Jika form dikirim menggunaan method=get maka di dalam PHP mengaksesnya dengan variabel $_GET, namun jika form dibuat menggunakan method=post, mengaksesnya dengan variabel $_POST.

Bagaimana jika pada saat memproses form tidak mengetahui dengan pasti apakah form dikirim dengan GET atau POST? PHP menyediakan variabel $_REQUEST sebagai salah satu solusinya. Variabel $_REQUEST menampung nilai form yang dikirim dengan method=get, maupun method=post secara bersamaan.

1.2.3. Kapan sebaiknya menggunakan GET?

Informasi dikirim dari sebuah form dengan method GET bisa dilihat oleh semua orang (semua nama dan nilai variabel ditampilkan di URL). GET juga memiliki batas pada jumlah informasi yang dikirim. Batasannya adalah sekitar 2000 karakter. Namun, karena variabel ditunjukkan di URL, ia memungkinkan untuk dilakukan bookmark halaman. Dalam beberapa kasus, hal ini sangat bermanfaat. GET bisa digunakan untuk mengirimkan data yang tidak sensitif.

Ingat! GET tidak boleh digunakan untuk mengirimkan password atau informasi sensitif lainnya!

1.2.4. Kapan Menggunakan POST?

Informasi yang dikirim dari sebuah form dengan method POST tidak bisa dilihat oleh siapapun (semua nama-nama atau nilai-nilai tertanam didalam body request HTTP) dan tidak memiliki batasan jumlah informasi yang akan dikirim. POST juga mendukung fungsionalitas lanjutan seperti dukungan untuk input biner multi-part ketika sedang melakukan upload file ke server. Namun, karena variabel tidak ditampilkan di URL, tidak

mungkin untuk dilakukan bookmark halaman (data tidak ter-bookmark). Developer lebih baik menggunakan POST untuk mengirimkan data form.

1.2.5. Validasi Form PHP

Gambar 4 *contoh form dengan validasi*

Form HTML yang akan kita gunakan pada modul ini, mengandung bermacammacam field input, misalnya text field yang harus diisi dan text field yang opsional, tombol pilihan (radio button), dan tombol submit. Rule atau aturan validasi untuk form diatas adalah sebagai berikut:

Field	Rule Validasi
Name	Dibutuhkan. + Harus hanya mengandung huruf dan spasi
E-mail	Dibutuhkan. + Harus mengandung sebuah alamat email yang valid dengan @ dan .
Website	Opsional. Jika ada, harus mengandung URL yang valid.
Comment	Opsional. Field input multi-line (text area).
Gender	Dibutuhkan. Harus memilih salah satu

Gambar 5 *aturan validasi*

Kode HTML untuk membentuk Form tersebut adalah sebagai berikut:

- ***Text Field***

  Field nama, email dan website adalah elemen-elemen text input, dan field komentar adalah textarea yaitu sebagai berikut:

  Name: <input type="text" name="name">

  E-mail: <input type="text" name="email">

  Website: <input type="text" name="website">

Comment: <textarea name="comment" rows="5" cols="40"></textarea>

- ***Radio Button***

  Field jenis kelamin adalah radio button yaitu sebagai berikut:

  Gender:

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

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

- ***Form Element***

  Kode HTML untuk membentuk form pada gambar diatas adalah sebagai berikut:

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

  Ketika form disubmit, data pada form dikirim dengan method "post". $_SERVER["PHP_SELF"] adalah variabel super global yang mengembalikan nama file dari skrip yang sedang dieksekusi. Sehingga kode form diatas mengirim data pada form ke halaman itu sendiri. Sedangkan fungsi htmlspecialchars() adalah fungsi yang mengkonversikan karakter-karakter spesial ke entitas HTML. Sebagai contoh, fungsi tersebut akan mengkonversikan karakter < dan > menjadi < dan >. Fungsi ini mencegah injeksi yang bisa dilakukan dengan HTML atau javascript (Cross-site Scripting Attack) pada form tersebut.

1.2.6. Catatan Penting pada Keamanan Form PHP

Variabel $_SERVER["PHP_SELF"] bisa digunakan oleh hacker! Jika PHP_SELF digunakan pada halaman web, user bisa memasukkan skrip dengan terlebih dahulu memasukkan garis miring (/) kemudian beberapa perintah Cross Site Scripting (XSS) untuk dieksekusi. XSS adalah tipe kelemahan keamanan komputer yang secara tipikal ditemukan dalam aplikasi web.

Asumsikan kita memiliki halaman web dengan nama "test_form.php", dan form hanya kita deklarasikan sebagai berikut:

<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">

Kemudian user memasukkan URL pada address bar dengan alamat sebagai berikut:

http://localhost/<nama_folder>/test_form.php/%22%3E%3Cscript%3Eal ert('hacked')%3C/scr ipt %3E

yang jika ditranslasikan akan menjadi:

<form method="post" action="test_form.php/"><script>alert('hacked')</script>

Kode ini menambah tag script dan perintah alert atau peringatan, ketika halaman dibuka, kode javascript tersebut akan dieksekusi, maka user akan melihat kotak peringatan dengan tulisan "hacked".

Berhati-hatilah dengan kemungkinan penambahan kode javascript pada tag <script>!
Hacker bisa mengarahkan user ke file pada server yang lain, dan file itu bisa mengandung kode yang bisa merubah variabel global atau melakukan submit form pada alamat web yang berbeda untuk mencuri data user.

Bagaimana menghindari penyalahgunaan $_SERVER["PHP_SELF"]?

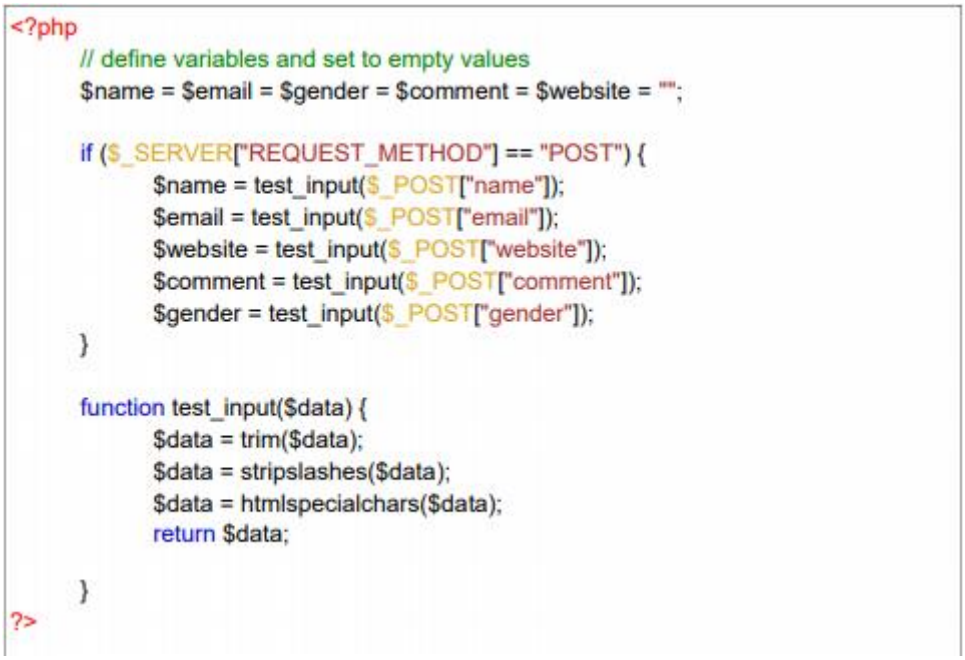
Caranya adalah dengan menggunakan fungsi htmlspecialchars(). Fungsi tersebut akan mengkonversikan karakter khusus ke entitas HTML. Ketika user memasukkan URL dengan tag script seperti contoh sebelumnya, maka akan ditranslasikan sebagai berikut:

<formmethod="post"action="test_form.php/"><script>aler t('hacked')</script>">

dengan cara ini, percobaan penyalahgunaan akan gagal.

- **Memvalidasi data Form dengan PHP**

  Hal pertama yang akan kita lakukan adalah memasukkan semua variabel melalui fungsi htmlspecialchars(). Kemudian ada juga dua hal ketika user melakukan submit form: 1. Membuang karakter-karakter yang tidak dibutuhkan (seperti spasi extra, tab extra, dan baris baru yang ektra) dari data input user (dengan fungsi trim()). 2. Membuang backslash (\) tatu garis miring dari data input user (dengan fungsi stripslashed()). Langkah



```php
<?php
    // define variables and set to empty values
    $name = $email = $gender = $comment = $website = "";

    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $name = test_input($_POST["name"]);
        $email = test_input($_POST["email"]);
        $website = test_input($_POST["website"]);
        $comment = test_input($_POST["comment"]);
        $gender = test_input($_POST["gender"]);
    }

    function test_input($data) {
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;

    }
?>
```

Gambar 5 *fungsi untuk pemeriksaan*

berikutnya adalah membuat fungsi yang akan melakukan pemeriksaan kebenaran data yang diinputkan oleh user. Contohnya adalah sebagai berikut:

Ingat bahwa pada permulaan skrip, adalah pemeriksaan apakah form sudah disubmit menggunakan $_SERVER[“REQUEST_METHOD”]. Jika REQUEST_METHOD adalah POST, maka form telah disubmit dan seharusnya tervalidasi. Jika belum tersubmit, lewati langkah validasi dan tampilkan form kosong. Namun pada contoh diatas semua field input adalah opsional. Skrip bekerja baik bahkan jika user tidak melakukan entri data.

- **Field yang dibutuhkan**

Kode program berikut terdapat tambahan variabel baru yaitu: $nameErr, $emailErr, $genderErr. Variabel-variabel error ini akan menangani pesan error untuk field yang dibutuhkan. Percabangan dengan if else juga akan ditambahkan untuk setiap variabel $_POST. Fungsinya untuk memeriksa apakah variabel $_POST kosong, hal ini dilakukan dengan menggunakan fungsi empty(). Jika kosong, maka pesan error disimpan dalam variabel error yang berbeda, dan jika tidak kosong, ia akan mengirim data input user melalui fungsi test_input():

```php
<?php
    // define variables and set to empty values
    $nameErr = $emailErr = $genderErr = $websiteErr = "";
    $name = $email = $gender = $comment = $website = "";

    if ($_SERVER["REQUEST_METHOD"] == "POST")
        { if (empty($_POST["name"])) {
            $nameErr = "Name is required";
        } else {
            $name = test_input($_POST["name"]);
        }

        if (empty($_POST["email"])) {
            $emailErr = "Email is required";
        } else {
            $email = test_input($_POST["email"]);
        }

        if (empty($_POST["website"])) {
            $website = "";
        } else {
            $website = test_input($_POST["website"]);
        }

        if (empty($_POST["comment"])) {
            $comment = "";
        } else {
            $comment = test_input($_POST["comment"]);
        }

        if (empty($_POST["gender"])) {
            $genderErr = "Gender is required";
        } else {
            $gender = test_input($_POST["gender"]);
        }
    }
?>
```

Gambar 6 *fungsi menangani pesan error*

Setelah kode diatas ditambahkan, beberapa skrip ditambahkan pada setiap field yang dibutuhkan pada form, fungsinya untuk menampilkan pesan error jika field yang dibutuhkan tidak diisi. Form HTMLnya adalah sebagai berikut:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">

        Name: <input type="text" name="name">
        <span class="error">* <?php echo
        $nameErr;?></span> <br><br>
        E-mail:
        <input type="text" name="email">
        <span class="error">* <?php echo $emailErr;?></span>
        <br><br>
        Website:
        <input type="text" name="website">
        <span class="error"><?php echo $websiteErr;?></span>
        <br><br>
        Comment: <textarea name="comment" rows="5" cols="40"></textarea>
        <br><br>
        Gender:
        <input type="radio" name="gender" value="female">Female
        <input type="radio" name="gender" value="male">Male
        <span class="error">* <?php echo $genderErr;?></span>
        <br><br>
        <input type="submit" name="submit" value="Submit">

</form>
```

Gambar 6 *fungsi menampilkani pesan error*

- **Validasi Nama**

Kode berikut menunjukkan cara sederhana untuk memeriksa apakah field nama hanya mengandung huruf dan spasi. Jika nilai dari nama tidak valid, maka pesan error akan disimpan didalam variabel $nameErr:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
        $nameErr = "Only letters and white space allowed";
}
```

Gambar 7 *fungsi memeriksa nama*

Fungsi preg_match() mencari string berdasarkan pola, mengembalikan nilai true jika polanya ada, false jika polanya tidak ada.

- **Validasi Email**

Cara paling mudah dan paling aman untuk memeriksa apakah sebuah alamat email memiliki pola yang sesuai adalah dengan menggunakan fungsi filter_var(). Kode dibawah memeriksa apakah alamat email yang dimasukkan menggunakan pola yang sesuai atau tidak, jika tidak, maka pesan error akan disimpan kedalam variabel $emailErr:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        { $emailErr = "Invalid email format";
}
```

Gambar 8 *fungsi memeriksa email*

- **Validasi URL**

Kode program berikut menunjukkan cara untuk memeriksa apakah sintaks alamat URL valid atau tidak. Ekspresi reguler ini mengizinkan keberadaan tanda pisah pada URL. Jika sintaks alamat URL tidak valid, maka pesan error akan disimpan kedalam variabel $websiteErr:

```
$website = test_input($_POST["website"]);

if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
        9+&@#\/%?=~_|!:,.;]*[-a-z0-

9+&@#\/ %=~_|]/i",$website)) {

$websiteErr = "Invalid URL";

}
```

Biasanya, jika user salah menginputkan nilai, maka halaman yang tampil adalah halaman yang sama dengan field yang sudah terisi dengan nilai field yang sudah diinput sebelumnya. Untuk menunjukkan nilai dalam field input setelah user menekan tombol submit, ada beberapa skrip PHP yang perlu ditambahkan didalam atribut value pada field input name, email, dan website. Khusus untuk field textarea, akan skrip tersebut akan ditambahkan antara tag <textarea> dan tag </textarea>. Skrip yang singkat

akan mengeluarkan nilai dari variabel $name, $email, $website dan $comment. Untuk radio button atau tombol radio, akan ditambahkan kode yang membuat salah satu pilihan terpilih.

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;? ></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo
"checked";?> value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo
"checked";?> value="male">Male
```

Gambar 9 *penambahan script PHP*

BAB II

PEMBAHASAN

Buatlah program web untuk menginputkan username dan password menggunakan form dan penanganan input data dengan kriteria sebagai berikut:

2.1. Username yang diinputkan tidak boleh lebih dari tujuh karakter!

Pada soal nomor satu, praktikan diperintahkan membuat sebuah handling yang mengendalikan agar apabila user memasukan username, dan karakternya lebih dari tujuh maka program akan menyatakan bahwa tidak boleh melebihi tujuh karakter.

```php
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $user = $_request["user"];
        $pass = $_request["pass"];
        $_user = strlen($user);
        $_pass = strlen($pass);
        $x = false;

        if ($_user > 7) {
            echo "Username harus kurang dari 7 karakter<br>";
            $x = true;
        }
```

Gambar 2.1 *Penulisan Handling pada Program*

Pada baris awal-awal, merupakan sebuah script untuk pembuatan elemen form. Apabila di submit maka akan dikirim menggunakan method *"POST"*. Lalu selanjutnya itu merupakan bentuk validasi data form dengan PHP yang jelas adalah memasukkan masing-masing variabel. Dimaksudkan agar apabila user mengklik dan memasukkan nama user dan password akan langsung dikirim ke server. Fungsi *strlen* berfungsi untuk menghitung panjang string. Untuk handling ini tidak terlepas dengan yang namanya percabangan *if*, dan untuk variable user apabila lebih dari 7, maka akan ditampilkan *"username harus kurang dari 7 karakter"*. Apabila benar berarti bernilai *true.*

2.2. Password yang dinputkan harus terdiri dari huruf kapital, huruf kecil, angka dan karakter khusus.

```
    if (!preg_match("/[A-Z]/", $pass)) {
        echo "Password harus terdapat huruf kapital<br>";
        $x = true;
    }

    if (!preg_match("/[a-z]/", $pass)) {
        echo "Password harus terdapat huruf kecil<br>";
        $x = true;
    }

    if (!preg_match("/[0-9]/", $pass)) {
        echo "Password harus terdapat karakter khusus (ang
        $x = true;
    }

    if ($_pass < 10) {
        echo "Password harus lebih dari 10 karakter<br>";
        $x = true;
    }

    if ($x == false) {
        echo "Username dan Password memenuhi syarat";
    }
}
```

Gambar 2.2 *Penulisan Handling pada Program password*

Kode *"preg_match"* digunakan untuk memeriksa apakah field nama hanya mengadung huruf dan spasi, jika nilai tidak valid maka pesan error akan disimpan didalam variable *$nameErr*. apabila salah maka akan muncul pesan *"Pesan harus terdapat huruf kecil"*.

Begitu juga dengan *if* yang kedua akan tetapi harus berupa karakter khusus atau angka. Apabila salah akan muncul pesan error *"Password harus terdapat karakter khusus(angka)"*.

2.3. Jumlah karakter password tidak boleh kurang dari 10 karakter.

Selanjutnya password harus lebih dari 10 karakter, apabila kurang dari 10 karakter, maka akan muncul pesan error *"Password harus lebih dari 10 karakter"*.

Variable x yang bernilai *false*, dan apabila bernilai *salah*, maka akan dinyatakan benar dan muncul pesan *"Username dan Password memenuhi syarat"*

BAB III

KESIMPULAN

Error handling sangat berguna bagi user/pengguna khsusunya karena dengan menggunakan Handling bisa memudahkan pengguna dalam menggunakan suatu aplikasi atau web terutama pada saat melakukan register atau submit akun. Sehingga pengguna tidak kebingungan apa saja yang tidak bisa dilakukan dan apa saja yang bisa dilakukan untuk memasukkan value kedalam form akun.

DAFTAR PUSTAKA

[1] A. Muhardian, “Belajar PHP: Menggunakan Percabangan untuk Membuat Logika Program,” *www.petanikode.com*, 2015. https://www.petanikode.com/php-percabangan/ (accessed Apr. 06, 2021).

[2] N. Huda, “PHP Dasar: Manipulasi String | Jago Ngoding,” *jagongoding.com*, 2020. https://jagongoding.com/web/php/dasar/manipulasi-string/ (accessed Apr. 06, 2021).

[3] Andre, “Tutorial Form PHP: Pengertian Variabel SuperGlobal $_GET, $_POST | Duniailkom,” *www.duniailkom.com*, 2014. https://www.duniailkom.com/tutorial-form-php-pengertian-variabel-superglobals-_get-_post-dan-_request/ (accessed Apr. 05, 2021).

[4] Jurusan Teknik Informatika, *MODUL PRAKTIKUM PEMROGRAMAN WEB I*. 2021.

LAMPIRAN

```
<html>
	<body>
		<form action="welcome.php" method="post">
			Name: <input type="text" name="name"><br>
			E-mail: <input type="text" name="email"><br>
			<input type="submit">
		</form>

	</body>
</html>
```

Gambar 1 *contoh form HTML sederhana*

```
<html>
	<body>
		Welcome <?php echo $_POST["name"]; ?><br>
		Your email address is: <?php echo $_POST["email"];
	?> </body>
</html>
```

Gambar 2 ***contoh form HTML + PHP***

```
<html>
	<body>

		<form action="welcome_get.php" method="get">
			Name: <input type="text" name="name"><br>
			E-mail: <input type="text" name="email"><br>
			<input type="submit">
		</form>
	</body>
</html>
```

Ingat! GET tidak boleh digunakan untuk mengirimkan password atau informasi sensitif lainnya!

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Submit

Gambar 4 ***contoh form dengan validasi***

Field	Rule Validasi
Name	Dibutuhkan. + Harus hanya mengandung huruf dan spasi
E-mail	Dibutuhkan. + Harus mengandung sebuah alamat email yang valid dengan @ dan .
Website	Opsional. Jika ada, harus mengandung URL yang valid.
Comment	Opsional. Field input multi-line (text area).
Gender	Dibutuhkan. Harus memilih salah satu

Gambar 5 *aturan validasi*

Berhati-hatilah dengan kemungkinan penambahan kode javascript pada tag <script>!
Hacker bisa mengarahkan user ke file pada server yang lain, dan file itu bisa mengandung kode yang bisa merubah variabel global atau melakukan submit form pada alamat web yang berbeda untuk mencuri data user.

```php
<?php
	// define variables and set to empty values
	$name = $email = $gender = $comment = $website = "";

	if ($_SERVER["REQUEST_METHOD"] == "POST") {
		$name = test_input($_POST["name"]);
		$email = test_input($_POST["email"]);
		$website = test_input($_POST["website"]);
		$comment = test_input($_POST["comment"]);
		$gender = test_input($_POST["gender"]);
	}

	function test_input($data) {
		$data = trim($data);
		$data = stripslashes($data);
		$data = htmlspecialchars($data);
		return $data;

	}
?>
```

Gambar 5 *fungsi untuk pemeriksaan*

```php
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
	$nameErr = "Only letters and white space allowed";
}
```

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;? ></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo
"checked";?> value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo
"checked";?> value="male">Male
```

Gambar 9 *penambahan script PHP*

```php
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $user = $_request["user"];
        $pass = $_request["pass"];
        $_user = strlen($user);
        $_pass = strlen($pass);
        $x = false;

        if ($_user > 7) {
            echo "Username harus kurang dari 7 karakter<br>";
            $x = true;
        }
```

Gambar 2.1 *Penulisan Handling pada Program*

```
    if (!preg_match("/[A-Z]/", $pass)) {
        echo "Password harus terdapat huruf kapital<br>";
        $x = true;
    }

    if (!preg_match("/[a-z]/", $pass)) {
        echo "Password harus terdapat huruf kecil<br>";
        $x = true;
    }

    if (!preg_match("/[0-9]/", $pass)) {
        echo "Password harus terdapat karakter khusus (ang
        $x = true;
    }

    if ($_pass < 10) {
        echo "Password harus lebih dari 10 karakter<br>";
        $x = true;
    }

    if ($x == false) {
        echo "Username dan Password memenuhi syarat";
    }
}
```

Gambar 2.2 *Penulisan Handling pada Program password*