

**LAPORAN HASIL PRAKTIKUM
PEMROGRAMAN WEB & MOBILE I**



NAMA : AXEL BERKATI
NIM : 193010503007
KELAS : A
MODUL : II (FORM HANDLING)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2021

BAB 1

TUJUAN DAN LANDASAN TEORI

1.1. Tujuan

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

1.2. Landasan Teori

Superglobals variabel pada PHP yaitu variabel \$ _GET dan variabel \$ _POST digunakan untuk mengumpulkan data formulir. Kita dapat membuat dan menggunakan formulir di PHP. Untuk mendapatkan data formulir, kita perlu menggunakan variabel PHP superglobals \$ _GET dan \$ _POST(Sutiono, 2020).

Permintaan formulir atau form dapat berupa get atau post. Untuk mengambil data dari get request, kita perlu menggunakan \$ _GET, untuk post request harus menggunakan \$ _POST.

Situs web saat ini telah menyediakan banyak fungsionalitas yang dapat digunakan untuk menyimpan, memperbarui, mengambil, dan menghapus data dalam database. Form adalah dokumen yang berisi banyak data, yang dimana datanya dapat diisi pengguna atau pengguna dapat memilih datanya, Biasanya data tersebut akan disimpan di database suatu sistem.

Contoh di bawah ini menampilkan formulir HTML sederhana dengan dua input masukan dan sebuah tombol submit yang dapat dilihat pada kode dibawah ini.

```
<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>
```

Ketika pengguna mengisi formulir di atas dan mengklik tombol submit, maka data formulir akan dikirim untuk diproses ke file PHP bernama “welcome.php”. Data formulir dikirim dengan metode HTTP POST. Untuk menampilkan data yang dikirimkan, Anda cukup memanggil semua variabelnya pada “Welcome.php” yang dapat terlihat seperti kode dibawah ini.

```
<html>
<body>

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

</body>
</html>
```

Maka akan menghasilkan output dengan hasil yaitu Welcome John Your email address is john.doe@example.com.

1.2.1. Metode GET Request

GET Request adalah permintaan atau request formulir secara default di PHP. Data yang melewati permintaan sebuah metode get akan terlihat di browser. Yang akan terlihat pada browser disini adalah URL dari input user sehingga tidak diamankan. Anda dapat mengirim data dalam jumlah terbatas melalui permintaan get. Kita akan buat sebuah contoh sederhana dari metode get request di PHP.

Mari kita lihat contoh sederhana untuk menerima data dari get request di PHP:

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

Kemudian kita dapat buat file PHP nya dengan kode berikut:

```
<?php
$name=$_GET["name");//receiving name field value in $name variable
echo "Welcome, $name";
?>
```

1.2.2. Metode POST Request

Post request banyak digunakan untuk submit form yang memiliki banyak data seperti upload file, upload gambar, form login, form pendaftaran dan lainnya. Data yang dikirimkan melalui permintaan post maka data input dari user ini tidak terlihat di browser . Jadi URL yang dimasukkan saat mengirimkan sebuah data dari form akan lebih diamankan. Anda dapat mengirim data dalam jumlah besar melalui permintaan post.

Mari kita lihat contoh sederhana untuk menerima data dari permintaan post di PHP.

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text"
name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password"
name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>
</td></tr>
</table>
</form>
```

Kemudian kita dapat buat file PHP nya dengan kode berikut

```
<?php
$name=$_POST["name"];
$password=$_POST["password"];
echo "Welcome: $name, your password is: $password";
?>
```

1.2.3. Perbedaan Metode GET dan POST

GET dan POST membuat sebuah array (contoh array(kunci => nilai, kunci2 => nilai2, kunci3 => nilai3, ...))(Informatika, 2021). Array ini akan menampung pasangan kunci (key) / nilai (value), di mana kunci adalah nama dari form controll yang ada dan nilai adalah data masukan dari pengguna. Baik GET dan POST diperlakukan sebagai \$ _GET dan \$ _POST. Ini adalah variabel superglobals, yang artinya selalu dapat diakses, apa pun cakupannya – dan Anda dapat mengaksesnya dari fungsi, kelas, atau file apa pun tanpa harus melakukan sesuatu yang khusus.

- \$ _GET adalah array variabel yang diteruskan ke skrip saat ini melalui parameter URL.
- \$ _POST adalah array variabel yang diteruskan ke skrip saat ini melalui metode HTTP POST.

1.2.3.1. Kapan menggunakan GET?

Method get digunakan ketika mengirimkan data dari form yang sifatnya sensitif, karena inputan form akan ditampilkan dibagian url dari website, anda tidak disarankan menggunakan method get ketika membuat form yang datanya rahasia seperti form login, form register, dan form transaksi(Samsudin, 2018). Informasi yang dikirim dari formulir dengan metode GET dapat dilihat oleh semua orang (semua nama dan nilai variabel ditampilkan di URL). GET juga memiliki batasan jumlah informasi yang akan dikirim. Batasannya sekitar 2000 karakter.

Namun, karena variabel ditampilkan di URL, dimungkinkan untuk menandai halaman. Ini dapat berguna dalam beberapa kasus. GET dapat digunakan untuk mengirim data non-sensitif. Catatan penting yaitu metode GET TIDAK PERNAH digunakan untuk mengirim sandi atau informasi sensitif lainnya.

1.2.3.2.Kapan menggunakan POST?

Informasi yang dikirim dari formulir dengan metode POST tidak terlihat oleh orang lain (semua nama / nilai disematkan di dalam body suatu permintaan HTTP) dan tidak memiliki batasan jumlah informasi yang akan dikirim. Selain itu, POST mendukung fungsionalitas lanjutan seperti dukungan untuk input biner multi-bagian saat mengunggah file ke server. Namun, karena variabel tidak ditampilkan di URL, maka tidak mungkin untuk menandai suatu halaman. Pengembang program saat ini lebih memilih POST untuk mengirim data form.

Pikirkan KEAMANAN dari suatu data saat kita memproses form di PHP! Halaman ini tidak berisi validasi formulir apa pun, ini hanya menunjukkan bagaimana Anda dapat mengirim dan mengambil data formulir. Namun, selanjutnya kita akan menunjukkan bagaimana memproses formulir PHP dengan mempertimbangkan keamanan dengan sebuah validasi. Validasi data formulir yang tepat penting untuk melindungi formulir Anda dari peretas dan pengirim spam.

1.2.4. Form Validation

Form yang harus diisi akan memeriksa apakah form tersebut harus diisi atau tidak dengan cara yang benar. Sebagian besar kasus kita saat akan mengharuskan mengisi kolom form maka dapat menggunakan simbol * untuk input wajib. Validasi berarti memeriksa masukan yang dikirimkan oleh pengguna. Ada dua jenis validasi yang tersedia di PHP. Jenis validasi disini adalah sebagai berikut

- Client-Side Validation yaitu Validasi yang dilakukan di browser web mesin klien.
- Server-Side Validation yaitu Setelah data terkirim dari pengguna, maka data akan dikirim ke server dan melakukan pemeriksaan validasi di mesin server.

Beberapa Aturan Validasi yang biasanya harus digunakan yaitu :

Tabel 1.1. Aturan Validasi

Field	Validation Rules atau Aturan Validasi
Nama	Harus (Required) diisi huruf dan spasi
Email	Harus (Required) diperlukan @ dan . (titik). Email disini yaitu Sintaks yang akan memverifikasi apakah alamat Email yang diberikan berbentuk baik atau tidak. Jika tidak, itu akan menampilkan pesan kesalahan.
Website	Tidak Harus (Optional) membutuhkan URL yang valid, kalau tidak ada tidak masalah. Valid URL disini adalah Sintaks yang akan memverifikasi apakah URL yang diberikan valid atau tidak. Ini harus mengizinkan beberapa kata kunci seperti https, ftp, www, a-z, 0-9, .. dan lainnya ..
Radio	Harus (Required) dapat dipilih setidaknya sekali
Check Box	Harus (Required) dapat dicentang setidaknya sekali
Drop Down menu	Harus (Required) dapat dipilih setidaknya sekali

Catatan Besar dan sangat penting tentang Form Validation di PHP yaitu variabel `$ _SERVER ["PHP_SELF"]` dapat digunakan oleh peretas. Jika `PHP_SELF` ini digunakan di halaman Anda, maka pengguna dapat memasukkan garis miring (/) dan kemudian memasukkan beberapa perintah Cross Site Scripting (XSS) untuk dijalankan dan dapat meretas

situs anda. Cross Site Scripting (XSS) adalah jenis kerentanan keamanan komputer yang biasanya ditemukan di aplikasi Web. XSS memungkinkan penyerang memasukkan skrip sisi klien ke halaman Web yang dilihat oleh pengguna lain. Untuk memperjelas penyerangan ini maka kita dapat membuat sebuah ilustrasi untuk memudahkannya.

Asumsikan kita memiliki formulir berikut di halaman bernama “test_form.php”

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

Sekarang, jika pengguna memasukkan URL normal di bilah alamat seperti : “http://www.example.com/test_form.php”, kode di atas akan diterjemahkan menjadi berikut

```
<form method="post" action="test_form.php">
```

Sejauh ini sudah lebih baik. Namun, pertimbangkan bahwa pengguna memasukkan URL berikut di browser:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

Dalam hal ini, kode di atas akan diterjemahkan menjadi berikut ini

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

Kode peretasan ini digunakan dengan menambahkan tag skrip dan perintah peringatan alert. Dan saat halaman dimuat, kode JavaScript akan dijalankan (pengguna akan melihat kotak peringatan). Ini hanyalah contoh sederhana dan tidak berbahaya bagaimana variabel PHP_SELF dapat dieksploitasi. Ketahuilah bahwa kode JavaScript apa pun dapat ditambahkan di dalam tag <script> .

Seorang peretas dapat mengarahkan pengguna ke file yang berada di server lain, dan file itu dapat menyimpan kode berbahaya yang dapat mengubah variabel global atau mengirimkan formulir ke alamat lain untuk menyimpan data pengguna. Cara Menghindari Eksploitasi \$ _SERVER ["PHP_SELF"] adalah dengan menggunakan fungsi htmlspecialchars () yang telah dijelaskan sebelumnya. Fungsi htmlspecialchars () mengubah karakter khusus menjadi entitas HTML.

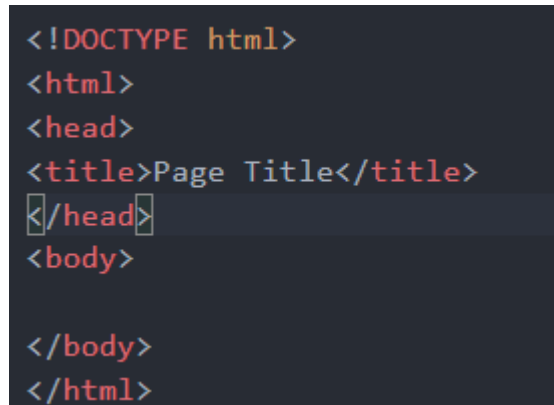
BAB II PEMBAHASAN

2.1. Langkah Kerja

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

- a. Username yang diinputkan tidak boleh lebih dari 7 karakter.
- b. Password yang diinputkan harus terdiri dari huruf kapital, huruf kecil, angka dan karakter khusus
- c. Jumlah karakter password tidak boleh kurang dari sepuluh karakter.

2.2. Pembahasan

A screenshot of a code editor showing the basic structure of an HTML document. The code is as follows:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

</body>
</html>
```

Gambar 2.1. Deklarasi dokumen HTML

Deklarasi ini akan memberitahu web browser bahwa kita menuliskan instruksi dalam versi HTML. Penulisannya sendiri ditandai dengan adanya DOCTYPE atau Document Type Declaration (DTD) pada baris pertama text editor.

Setelah mendeklarasikan dokumen HTML, kita bisa memulai proses pembuatan form dengan tag `<html>`.

```
<head>
<title>Praktikum Modul 2</title>
</head>
```

Gambar 2.2. Isi pada tag <head>

Tag ini berfungsi untuk membuat kepala dokumen. Pada isi tag <head> yang dibuat diatas di isi tag <title> dengan nama praktikum modul 2. Tag <title> ini sendiri yaitu untuk membuat judul halaman.

```
<body>

<form action="" method="post">
  <input type="text" name="nm" value="" placeholder="Username">
  <br>
  <input type="text" name="p" value="" placeholder="Password">
  <br>

  <input type="submit" name="s" value="LOGIN">
</form>

<?php
if(isset($_POST['s'])){
  $n = $_POST['nm'];
  $p = $_POST['p'];
  if(empty($n)){
    echo "Tolong Masukkan Username anda";
  }
  if(strlen($n)>7){
    echo "username tidak boleh lebih dari 7 karakter";
  }
  if(empty($p)){
    echo "Tolong masukan password anda";
  }
  if(strlen($p)<10){
    echo "Password tidak boleh kurang dari 10 karakter";
  }
  if ( !preg_match("#[0-9]+#", $p) ) {
    echo "Password harus terdiri dari angka<br>";
  }

  if ( !preg_match("#[a-z]+#", $p) ) {
    echo "Password harus terdiri dari huruf kecil<br>";
  }

  if ( !preg_match("#[A-Z]+#", $p) ) {
    echo "Password harus terdiri dari huruf besar<br>";
  }

  if ( !preg_match("/[\\'\"^$%&*()]{@#~?><},|=_+!-|/"/, $p) ) {
    echo "Password harus terdiri dari special character<br>";
  }
}
?>

</body>
```

Gambar 2.3. Gambar isi tag <form>

Selanjutnya terdapat Tag <body>, yaitu wadah untuk meletakkan seluruh komponen inti form Yang akan dibuat.

```
<form action="" method="post">
  <input type="text" name="nm" value="" placeholder="Username">
  <br>
  <input type="text" name="p" value="" placeholder="Password">
  <br>

  <input type="submit" name="s" value="LOGIN">
</form>
```

Gambar 2.4. Gambar isi tag <form> Badan Form Login

Sebuah form dalam HTML harus berada di dalam tag form, yang diawali dengan <form> dan diakhiri dengan </form> (Andre, 2013). Tag form akan membutuhkan beberapa atribut untuk dapat berfungsi dengan seharusnya.

Atribut pertama adalah action, yang berfungsi untuk menjelaskan kemana data form akan dikirimkan. Biasanya nilai dari atribut action ini adalah alamat dari sebuah halaman PHP yang digunakan untuk memproses isi data form.

Atribut kedua adalah method, yang berfungsi untuk menjelaskan bagaimana data isian form akan dikirim oleh web browser. Nilai dari atribut method ini bisa berupa get atau post.

Pada potongan code diatas, dibuat form untuk menginputkan username dalam bentuk inputan text, dimana inputan ini akan dimasukan ke dalam variabel nm. Kemudian, dibuat juga inputan untuk menginputkan password dengan inputan text, dimana inputan ini akan dimasukan kedalam variabel p. Kemudian, dibuat sebuah inputan dalam bentuk submit, yang di beri nama login.

```

<?php
if(isset($_POST['s'])){
    $n = $_POST['nm'];
    $p = $_POST['p'];
    if(empty($n)){
        echo "Tolong Masukkan Username anda";
    }
    if(strlen($n)>7){
        echo "username tidak boleh lebih dari 7 karakter";
    }
    if(empty($p)){
        echo "Tolong masukan password anda";
    }
    if(strlen($p)<10){
        echo "Password tidak boleh kurang dari 10 karakter";
    }
    if ( !preg_match("#[0-9]+#", $p) ) {
        echo "Password harus terdiri dari angka<br>";
    }

    if ( !preg_match("#[a-z]+#", $p) ) {
        echo "Password harus terdiri dari huruf kecil<br>";
    }

    if ( !preg_match("#[A-Z]+#", $p) ) {
        echo "Password harus terdiri dari huruf besar<br>";
    }

    if ( !preg_match("/[\'^$%&*()]{@#~?><>,|_+!-}/", $p) ) {
        echo "Password harus terdiri dari special character<br>";
    }
}
?>

```

Gambar 2.5. Program PHP

Sebelum masuk ke validasi form pada gambar di atas, terdapat sintaks `<?php.` sintaks ini berfungsi sebagai kode wajib dalam program PHP agar program dapat dijalankan dan selalu diakhiri dengan `?>.` Potongan coding diatas merupakan validasi form dalam PHP, yang dimana dimana akan memeriksa isi dari form yang telah dibuat. Pada potongan coding diatas, terdapat kondisi if, yang berisi `isset()`, yang dimana akan menghasilkan nilai true, jika variabel `$_POST['s']` tersedia untuk diproses atau tidak. Jika tersedia, maka masuk ke isi dari kondisi if, yaitu isi dari variabel `$_POST['nm']` dicopy ke variabel `$n`, isi dari variabel `$_POST['p']` dicopy ke variabel `$p`.

Kemudian, dibuat kondisi if untuk memeriksa apakah variabel \$n telah di inputkan atau tidak menggunakan fungsi EMPTY. Jika tidak, maka akan ditampilkan Tolong masukan username anda.

Kemudian, dibuat juga kondisi if untuk memeriksa jumlah karakter pada variabel \$n menggunakan fungsi STRLEN, dimana jika jumlah karakter pada variabel \$n lebih dari 7, maka akan ditampilkan bahwa username tidak boleh lebih dari 7 karakter.

Kemudian, dibuat juga kondisi untuk memeriksa apakah variabel \$ telah di inputkan atau tidak menggunakan fungsi EMPTY. Jika tidak, maka akan ditampilkan Tolong masukan password anda.

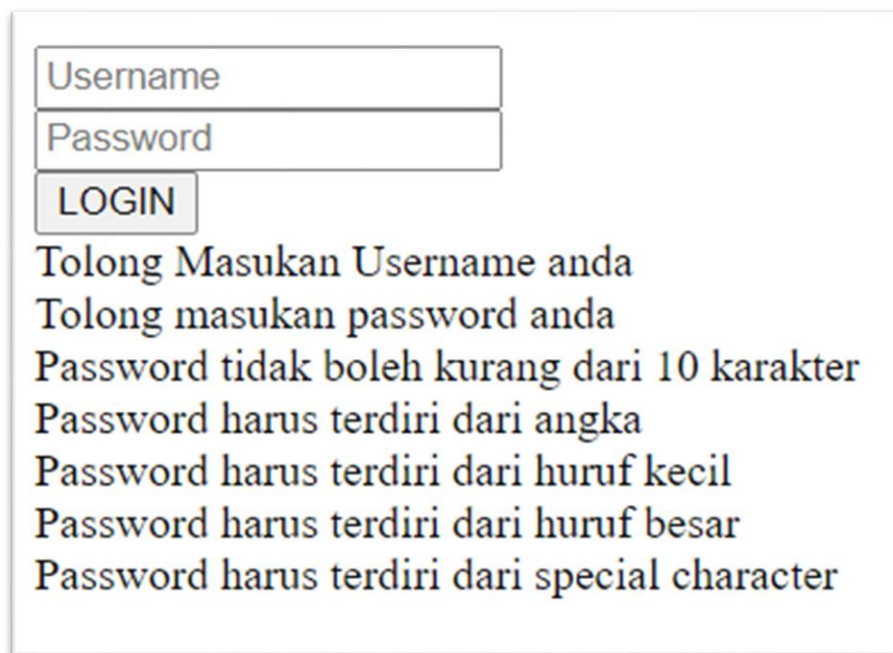
Kemudian, dibuat juga kondisi if untuk memeriksa jumlah karakter pada variabel \$p menggunakan fungsi STRLEN, dimana jika jumlah karakter pada variabel \$n kurang dari 10, maka akan ditampilkan bahwa password tidak boleh kurang dari 7 karakter.

Kemudian, dibuat juga kondisi if untuk memeriksa karakter pada variabel \$p, apakah terdapat angka dari 0-9, menggunakan fungsi PREG_MATCH. Jika tidak terdapat angka dari 0-9, maka akan ditampilkan bahwa password harus terdiri dari angka.

Kemudian, dibuat juga kondisi if untuk memeriksa karakter pada variabel \$p, apakah terdapat huruf kecil dari a-z didalamnya, menggunakan fungsi PREG_MATCH. Jika tidak terdapat huruf kecil a-z, maka akan ditampilkan bahwa password harus terdiri dari huruf kecil.

Kemudian, dibuat juga kondisi if untuk memeriksa karakter pada variabel \$p, apakah terdapat huruf besar dari A-Z didalamnya, menggunakan fungsi PREG_MATCH. Jika tidak terdapat huruf besar A-Z, maka akan ditampilkan bahwa password harus terdiri dari huruf besar.

Kemudian, dibuat juga kondisi if untuk memeriksa karakter pada variabel \$p, apakah terdapat special karakter didalamnya, menggunakan fungsi PREG_MATCH. Jika tidak terdapat spesial karakter didalamnya, maka akan ditampilkan bahwa password harus terdiri dari spesial karakter.



A screenshot of a web application's login interface. It features two input fields: 'Username' and 'Password', both of which are empty. Below these fields is a 'LOGIN' button. Underneath the button, there are several lines of text in blue, indicating required actions and password rules: 'Tolong Masukan Username anda', 'Tolong masukan password anda', 'Password tidak boleh kurang dari 10 karakter', 'Password harus terdiri dari angka', 'Password harus terdiri dari huruf kecil', 'Password harus terdiri dari huruf besar', and 'Password harus terdiri dari special character'.

Username

Password

LOGIN

Tolong Masukan Username anda

Tolong masukan password anda

Password tidak boleh kurang dari 10 karakter

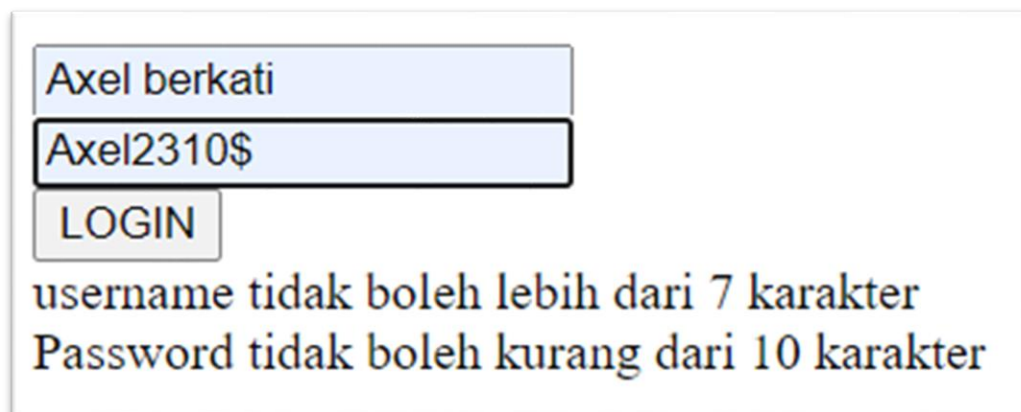
Password harus terdiri dari angka

Password harus terdiri dari huruf kecil

Password harus terdiri dari huruf besar

Password harus terdiri dari special character

Gambar 2.6. Output jika Username dan password tidak diinput



A screenshot of the same login interface as in Gambar 2.6, but with the input fields filled. The 'Username' field contains the text 'Axel berkati' and the 'Password' field contains 'Axel2310\$'. The 'LOGIN' button is still present. Below the button, there are two lines of text in blue: 'username tidak boleh lebih dari 7 karakter' and 'Password tidak boleh kurang dari 10 karakter'.

Axel berkati

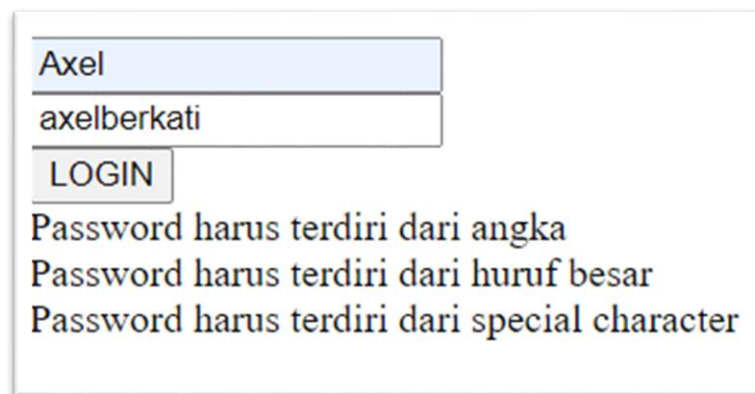
Axel2310\$

LOGIN

username tidak boleh lebih dari 7 karakter

Password tidak boleh kurang dari 10 karakter

Gambar 2.7. Output jika inputan username lebih dari 7 karakter dan password lebih 10 karakter



Axel

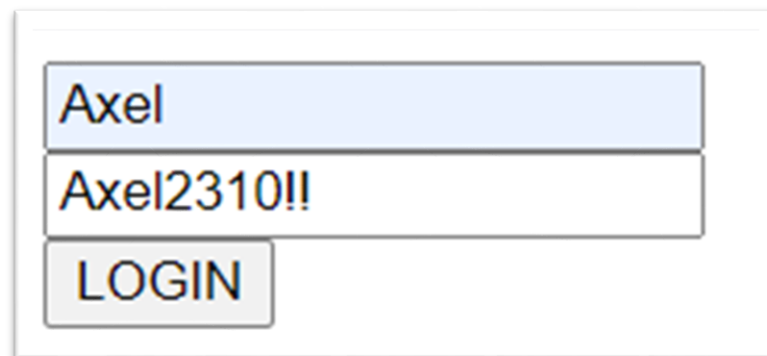
axelberkati

LOGIN

Password harus terdiri dari angka
Password harus terdiri dari huruf besar
Password harus terdiri dari special character

This screenshot shows a login form with a username field containing 'Axel' and a password field containing 'axelberkati'. Below the password field, there are three lines of red text indicating validation errors: 'Password harus terdiri dari angka', 'Password harus terdiri dari huruf besar', and 'Password harus terdiri dari special character'. A 'LOGIN' button is located below the password field.

Gambar 2.8. Output jika inputan password tidak ada angka, huruf besar, dan special karakter



Axel

Axel2310!!

LOGIN

This screenshot shows the same login form as Gambar 2.8, but with a valid password 'Axel2310!!' entered in the password field. The 'LOGIN' button is still present, and there are no validation error messages displayed.

Gambar 2.9. Output jika inputan pada username dan password sesuai dengan validasi form

BAB III

KESIMPULAN

Dari praktikum yang telah dilaksanakan, dapat disimpulkan bahwa Form handling ialah suatu mekanisme untuk menangani suatu masukan dari form yang dikirim oleh pengguna. Form handling berhubungan dengan metode yang dikirim dan bagaimana menangani pengiriman data berdasarkan metode yang digunakan.

Permintaan formulir atau form dapat berupa get atau post. Untuk mengambil data dari get request, kita perlu menggunakan \$ _GET, untuk post request harus menggunakan \$ _POST. Method get digunakan ketika mengirimkan data dari form yang sifatnya sensitif, karena inputan form akan ditampilkan dibagian url dari website, anda tidak disarankan menggunakan method get ketika membuat form yang datanya rahasia seperti form login, form register, dan form transaksi. Sedangkan, Method post dapat digunakan ketika data yang diinputkan form bersifat sensitif, seperti form login, register, ataupun form transaksi, hal tersebut dikarenakan method post tidak menampilkan inputan form, dibagian url dari web.

DAFTAR PUSTAKA

- Informatika, D. T. (2021). *MODUL PRAKTIKUM PEMROGRAMAN WEB I*
Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya.
- Andre. (2013, May 5). *Belajar HTML Dasar: Cara Membuat Form di HTML (tag form)* / *Duniaikom*. 2013. <https://www.duniaikom.com/belajar-html-cara-membuat-form-di-html-tag-form/>
- Samsudin, A. (2018, September 17). *Tutorial Belajar PHP Part 27 – Penanganan Form di PHP* / *Warung Belajar*. 2018.
<https://www.warungbelajar.com/penanganan-form-di-php.html>
- Sutiono. (2020, October 28). *Form Handling PHP: Contoh dan Source Code* - *DosenIT.com*. 2020. <https://dosenit.com/php/form-handling-php-contoh-dan-source-code>

LAMPIRAN

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

</body>
</html>
```

Gambar 2.1. Deklarasi dokumen HTML

```
<head>
<title>Praktikum Modul 2</title>
</head>
```

Gambar 2.2. Isi pada tag <head>

```
<body>
<form action="" method="post">
  <input type="text" name="nm" value="" placeholder="Username">
  <br>
  <input type="text" name="p" value="" placeholder="Password">
  <br>
  <input type="submit" name="s" value="LOGIN">
</form>
<?php
if(isset($_POST['s'])){
  $n = $_POST['nm'];
  $p = $_POST['p'];
  if(empty($n)){
    echo "Tolong Masukkan Username anda";
  }
  if(strlen($n)>7){
    echo "username tidak boleh lebih dari 7 karakter";
  }
  if(empty($p)){
    echo "Tolong masukan password anda";
  }
  if(strlen($p)<10){
    echo "Password tidak boleh kurang dari 10 karakter";
  }
  if ( !preg_match("#[0-9]+#", $p) ) {
    echo "Password harus terdiri dari angka<br>";
  }
  if ( !preg_match("#[a-z]+#", $p) ) {
    echo "Password harus terdiri dari huruf kecil<br>";
  }
  if ( !preg_match("#[A-Z]+#", $p) ) {
    echo "Password harus terdiri dari huruf besar<br>";
  }
  if ( !preg_match("/[!@#$%^&*(){}|~?><.,_|-!-/", $p) ) {
    echo "Password harus terdiri dari special character<br>";
  }
}
?>
</body>
```

Gambar 2.3. Gambar isi tag <form>

```
<form action="" method="post">
    <input type="text" name="nm" value="" placeholder="Username">
    <br>
    <input type="text" name="p" value="" placeholder="Password">
    <br>

    <input type="submit" name="s" value="LOGIN">
</form>
```

Gambar 2.4. Badan Form Login

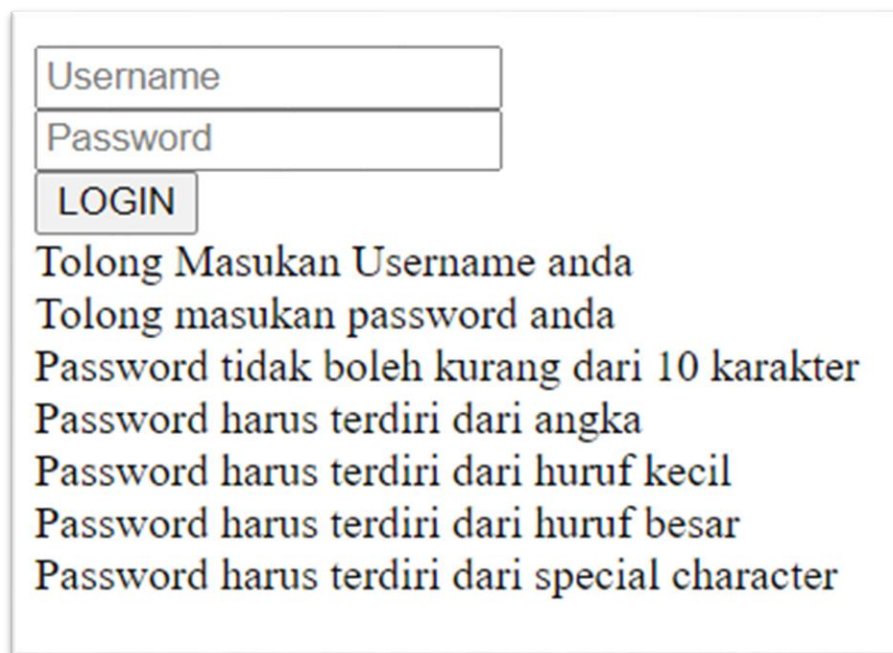
```
<?php
if(isset($_POST['s'])){
    $n = $_POST['nm'];
    $p = $_POST['p'];
    if(empty($n)){
        echo "Tolong Masukan Username anda";
    }
    if(strlen($n)>7){
        echo "username tidak boleh lebih dari 7 karakter";
    }
    if(empty($p)){
        echo "Tolong masukan password anda";
    }
    if(strlen($p)<10){
        echo "Password tidak boleh kurang dari 10 karakter";
    }
    if ( !preg_match("#[0-9]+#", $p) ) {
        echo "Password harus terdiri dari angka<br>";
    }

    if ( !preg_match("#[a-z]+#", $p) ) {
        echo "Password harus terdiri dari huruf kecil<br>";
    }

    if ( !preg_match("#[A-Z]+#", $p) ) {
        echo "Password harus terdiri dari huruf besar<br>";
    }

    if ( !preg_match("/[\'^!$%&*(){@#~?><>,|_+!-]/", $p) ) {
        echo "Password harus terdiri dari special character<br>";
    }
}
?>
```

Gambar 2.5. Program PHP



A screenshot of a web application's login interface. It features two input fields: 'Username' and 'Password', both of which are empty. Below these fields is a 'LOGIN' button. Underneath the button, there are several lines of text in blue, which serve as error messages or instructions for the user.

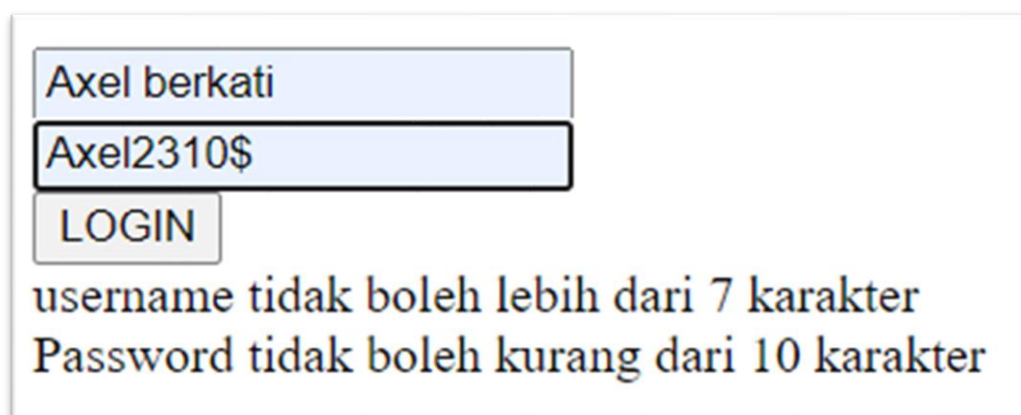
Username

Password

LOGIN

Tolong Masukan Username anda
Tolong masukan password anda
Password tidak boleh kurang dari 10 karakter
Password harus terdiri dari angka
Password harus terdiri dari huruf kecil
Password harus terdiri dari huruf besar
Password harus terdiri dari special character

Gambar 2.6. Output jika Username dan password tidak diinput



A screenshot of the same login interface as in Gambar 2.6, but with the input fields filled. The 'Username' field contains the text 'Axel berkati' and the 'Password' field contains 'Axel2310\$'. The 'LOGIN' button is still present. Below the button, there are two lines of text in blue, indicating validation errors for the username and password lengths.

Axel berkati

Axel2310\$

LOGIN

username tidak boleh lebih dari 7 karakter
Password tidak boleh kurang dari 10 karakter

Gambar 2.7. Output jika inputan username lebih dari 7 karakter dan password lebih 10 karakter

Axel

axelberkati

LOGIN

Password harus terdiri dari angka
Password harus terdiri dari huruf besar
Password harus terdiri dari special character

This screenshot shows a login form with a username field containing 'Axel' and a password field containing 'axelberkati'. Below the fields is a 'LOGIN' button. Three lines of error messages are displayed: 'Password harus terdiri dari angka', 'Password harus terdiri dari huruf besar', and 'Password harus terdiri dari special character'.

Gambar 2.8. Output jika inputan password tidak ada angka, huruf besar, dan special karakter

Axel

Axel2310!!

LOGIN

This screenshot shows the same login form, but the password field now contains 'Axel2310!!'. The 'LOGIN' button is still present, and no error messages are visible, indicating a successful login.

Gambar 2.9. Output jika inputan pada username dan password sesuai dengan validasi form