

**LAPORAN HASIL PRAKTIKUM  
PEMROGRAMAN WEB & MOBILE**



**NAMA : ELNATAN KENINGATKO**  
**NIM : 193020503038**  
**KELAS : A**  
**MODUL : II (Form Handling)**

**JURUSAN TEKNIK INFORMATIKA**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS PALANGKA RAYA**  
**2021**

## BAB I

### 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

Variabel superglobal PHP \$\_GET dan \$\_POST digunakan untuk mengumpulkan data-form. 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>
```

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>
```

Jika field nama diinputkan dengan Tono dan email

diinputkan dengan [tono@mail.com](mailto:tono@mail.com) maka output yang akan tampil adalah sebagai berikut:

Welcome Budi

Your email address is [tono@mail.com](mailto: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>
```

dengan file “welcome\_get.php” sebagai berikut:

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

### 1.2.1. GET vs. POST

GET dan POST membuat sebuah array (contoh array(kunci => nilai, kunci2 => nilai2, kunci3 => nilai3, ...)). Array ini menyimpan pasangan kunci/nilai, dimana kuncikunci adalah nama-nama dari form control dan nilai-nilai adalah data input dari user. Method GET diakses menggunakan `$_GET` dan method POST diakses menggunakan `$_POST`. Kedua variabel ini adalah variabel superglobal, yang selalu bisa diakses, tanpa memperhatikan lingkup dan bisa diakses dari fungsi, class atau file yang berbeda tanpa harus melakukan teknik khusus. `$_GET` adalah sebuah array dari variabel yang dikirimkan ke skrip melalui parameter URL. `$_POST` adalah

sebuah array dari variabel yang dikirimkan ke skrip melalui method HTTP POST.

#### 1.2.2. 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!

#### 1.2.3. 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.

#### 1.2.4. Validasi Form PHP

Pertimbangkan keamanan ketika memproses form PHP!

##### PHP Form Validation Example

\* required field.

Name:  \*

E-mail:  \*

Website:

Comment:

Gender: ☐ Female ☐ Male \*

**Gambar 1.1.** Form PHP.

Form HTML yang akan kita gunakan pada modul ini, mengandung bermacam-macam 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

Kode HTML untuk membentuk Form tersebut adalah sebagai berikut:

#### 1.2.5. 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>`

#### 1.2.6. 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`

### 1.2.7. 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.

### 1.2.8. 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/%3cnama\\_folder%3e/test\\_form.php/%22%3E%3Cscript%3Ealert\('hacked'\)%3C/script](http://localhost/%3cnama_folder%3e/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script)

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.

#### 1.2.9. 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:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/scr
ipt&gt;">
```

dengan cara ini, percobaan penyalahgunaan akan gagal.

### 1.2.10. 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 ekstra) dari data input user (dengan fungsi `trim()`).
2. Membuang backslash (\) satu garis miring dari data input user (dengan fungsi `stripslashes()`).

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

```
<?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;
}

?>
```

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.

#### 1.2.11. 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
// 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"]);
        }
    }
?>

```

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>

```

#### 1.2.12. 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";
}

```

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

#### 1.2.13. 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";
    }

```

#### 1.2.14. 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
```

## BAB II

### PEMBAHASAN

Berikut ini adalah penjelasan mengenai source code program web untuk menginputkan username dan password menggunakan form dan penanganan input data dengan kriteria sebagai berikut.

1. Username yang diinputkan tidak boleh lebih dari tujuh karakter.
2. Password yang diinputkan harus terdiri dari huruf kapital, huruf kecil, angka dan karakter khusus.
3. Jumlah karakter password tidak boleh kurang dari sepuluh karakter.

#### **193020503038.php**

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Handling</title>
</head>
<body>
  <form
    action="193020503038_1.php" method="post">

    Username: <input type="text" name="user" id="user">
    <br><br>

    Password: <input type="Password" name="pass" id="pass">
    <br><br>

    <button type="submit">Submit</button>

  </form>
</body>
</html>
```

Blok code di atas merupakan source code dari file PHP pertama yang diberi nama 193020503038.php. berikut ini merupakan penjelasan tentang blok code di atas.

```
<!DOCTYPE html>
```

Blok code di atas merupakan deklarasi pada dokumen HTML5 yang berfungsi untuk memberikan informasi kepada web browser tentang versi dokumen HTML yang bersangkutan.

```
<html>
...
[baris-baris code program]
...
</html>
```

Blok code di atas merupakan tag dengan elemen level tertinggi yang menyertakan setiap halaman HTML.

```
<head>
  <title>Form Handling</title>
</head>
```

Blok code di atas merupakan tag `<head></head>` yang menyimpan informasi meta, seperti judul dan *charset* halaman dan tag `<title></title>` yang berfungsi membuat judul halaman yang nantinya akan ditampilkan di browser. Pada bagian ini dibuat judul halaman dengan nama “Form Handling”.

```
<body>
...
[baris-baris code program]
...
</body>
```

Blok code di atas merupakan tag yang melampirkan semua konten yang muncul pada suatu halaman.

```
<form
  action="193020503038_1.php" method="post">
```

Blok code di atas merupakan penggunaan *form* yang digunakan untuk membuat elemen *form* berupa *check box*, *radio button*, *menu*, *text box*, *text area* dan *button*. Atribut yang pertama yaitu *action*, yang berfungsi untuk menjelaskan kemana data *form* akan dikirimkan. Di dalam blok code tersebut terlihat bahwa data *form* akan dikirimkan ke file PHP bernama “193020503038\_1.php”. Kemudian, atribut kedua yaitu *method*, yang berfungsi untuk menjelaskan bagaimana data ision *form* akan dikirimkan oleh web browser. Terlihat bawa nilai dari atribut *method* tersebut berupa *post*.

```
Username: <input type="text" name="user" id="user">
<br><br>
Password: <input type="Password" name="pass" id="pass">
<br><br>
```

Blok code di atas adalah bagian tag *input* yang paling banyak digunakan di dalam *form*. `<input type="text">` digunakan untuk inputan berupa text pendek, sedangkan `<input type="Password">` digunakan juga untuk inputan text pendek namun text yang diinput tidak akan terlihat. Terdapat atribut *name* agar *form* dapat diproses oleh web server nantinya. Nilai dari atribut *name* inilah yang akan menjadi *variable form*. Atribut *id* digunakan untuk menyimpan nilai yang telah diinputkan oleh *user*.

```
<button type="submit">Submit</button>
```

Blok code di atas merupakan tag *button* yang digunakan untuk membuat tombol yang bertuliskan “Submit”. `type="submit"` merupakan tipe tombol untuk memproses form.

### 193020503038\_1.php

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

```

    $x = false;

    if (!preg_match("/[0-7]*/", $user)) {
        echo "Username tidak boleh lebih dari 7 karakter<br>";
        $x = true;
    }

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

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

    if (!preg_match("/[0-9]/", $pass)) {
        echo "Password harus terdiri dari karakter khusus (simbol atau
angka)<br>";
        $x = true;
    }

    if (!preg_match("/[{\n,10}]/", $_pass)) {
        echo "Password tidak boleh kurang dari 10 karakter";
        $x = true;
    }

    if ($x == false) {
        echo "Selamat Datang".<br>";
        echo "Login Berhasil";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Login Berhasil</title>
</head>
<body>
    <br><br>
    <a href="193020503038.php">Logout
</body>
</html>

```



Blok code di atas merupakan source code dari file PHP pertama yang diberi nama 193020503038\_1.php. berikut ini merupakan penjelasan tentang blok code di atas.

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

Blok code di atas merupakan fungsi if yang digunakan untuk memeriksa apakah *form* sudah disubmit menggunakan `$_SERVER["REQUEST_METHOD"]`. Jika `REQUEST_METHOD` adalah `POST`, maka *form* telah disubmit dan seharusnya tervalidasi.

Terdapat variable baru yang diberi nama “user” dan “pass”. `$_POST` digunakan untuk memanggil data yang telah diinputkan agar bisa ditampilkan di file *action*. Kemudian kedua variable baru tersebut diberi fungsi *strlen* yang digunakan untuk menghitung jumlah variable agar validasi data form dapat dilakukan. Lalu, terdapat juga variable *x* yang diberi fungsi *Boolean* dengan nilai awal yaitu *false*.

```
    if (!preg_match("/[0-7]+/", $user)) {
        echo "Username tidak boleh lebih dari 7 karakter<br>";
        $x = true;
    }

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

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

    if (!preg_match("/[0-9]/", $pass)) {
        echo "Password harus terdiri dari karakter khusus (simbol atau angka)<br>";
```

```

    $x = true;
}

if (!preg_match("/[n,10}]/", $_pass)) {
    echo "Password tidak boleh kurang dari 10 karakter";
    $x = true;
}

if ($x == false) {
    echo "Selamat Datang". "<br>";
    echo "Login Berhasil";
}
}
?>

```

Blok code di atas merupakan bagian validasi *username*. Bagian tersebut dilakukan dengan cara menggunakan fungsi *preg\_match* dan *if*. Fungsi *preg\_match* digunakan untuk mencari *string* dengan pola tertentu pada variable *user*. Salah satu contohnya yaitu pada saat memeriksa *username*, pola yang digunakan untuk memeriksanya yaitu *"/[0-7]\*/"* yang berarti karakter yang dapat digunakan berjumlah maksimal tujuh karakter. Fungsi *if* digunakan agar pada saat variable *user* dicek dan tidak memiliki kesalahan maka nilai variable *x* akan bernilai *true*. Jika variable *x* bernilai *false*, maka akan muncul tulisan “Username tidak boleh lebih dari 7 karakter”. Sehingga program web akan dijalankan ulang.

Begitu juga pada saat memeriksa *password*, yang membedakannya hanya pada pola yang digunakan dan output yang ditampilkan pada saat fungsi *if* bernilai *true*. Beberapa pola yang digunakan untuk memeriksa password yaitu memeriksa huruf capital, huruf kecil, simbol atau angka dan tidak boleh kurang dari sepuluh karakter.

Yang terakhir terdapat fungsi *if* dengan perintah *(\$x == false)* yang berarti jika variable *x* bernilai *false*, maka akan muncul output dengan tulisan “Selamat Datang” dan “Login Berhasil”.

```

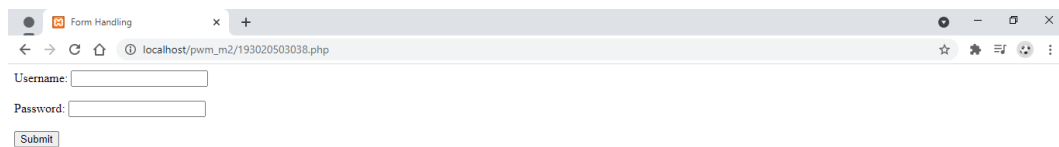
<!DOCTYPE html>
<html>
<head>
    <title>Login Berhasil</title>
</head>
<body>

```

```
<br><br>
<a href="193020503038.php">Logout
</body>
</html>
```

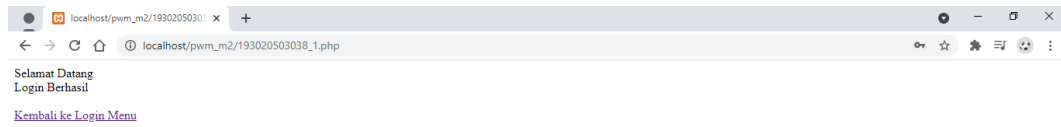
Blok code di atas merupakan dokumen HTML singkat yang digunakan untuk menambahkan tag `<a>` yang berartikan *hyperlink* dengan kalimat “Logout”. Hyperlink tersebut mengacu pada file PHP bernama “193020503038.php”. Sehingga, apabila login berhasil dilakukan akan terdapat *hyperlink* yang ketika ditekan akan kembali ke menu login awal.

Kemudian, pada saat program web dijalankan akan menampilkan output sebagai berikut.

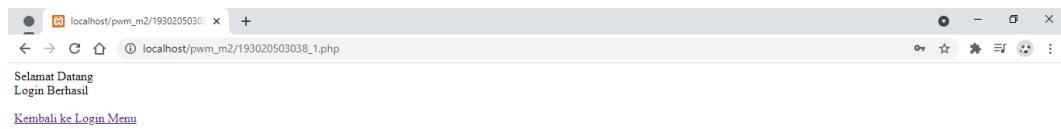


Activate Windows  
Go to Settings to activate Windows.

**Gambar 2.1.** Tampilan Awal Program Web.



**Gambar 2.2.** Input Username dan Password.



**Gambar 2.3.** Login Berhasil.

### **BAB III**

### **KESIMPULAN**

Suatu aplikasi banyak menerima masukan/input data dari pengguna, misalkan ketika registrasi kesuatu website atau ketika login pada halaman dan sebagainya. Untuk memungkinkan hal tersebut diperlukan suatu antarmuka pada klien dalam bentuk form HTML dan suatu mekanisme untuk penanganan data yang dikirim oleh pengguna pada sisi server. 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.

## **DAFTAR PUSTAKA**

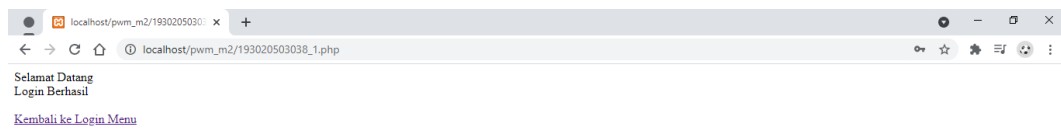
Tanpa Tahun. *MODUL PRAKTIKUM PEMROGRAMAN WEB I*. Palangka Raya:  
Jurusan Teknik Informatika, Universitas Palangka Raya.

## LAMPIRAN



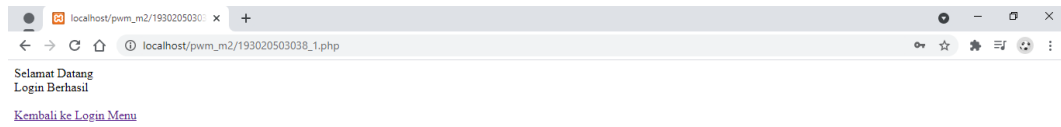
A screenshot of a web browser window. The title bar shows 'Form Handling'. The address bar displays 'localhost/pwm\_m2/193020503038.php'. The page content includes a 'Username:' label followed by a text input field, a 'Password:' label followed by a text input field, and a 'Submit' button below them.

**Gambar 2.1.** Tampilan Awal Program Web.



A screenshot of a web browser window. The title bar shows 'localhost/pwm\_m2/193020503038\_1.php'. The page content displays 'Selamat Datang' and 'Login Berhasil'. Below this, there is a blue hyperlink that reads 'Kembali ke Login Menu'.

**Gambar 2.2.** Input Username dan Password.



Activate Windows  
Go to Settings to activate Windows.

**Gambar 2.3.** Login Berhasil.