WEB CREATIVE
SOLUTIONS

BENGKEL MUDAH PHP MySQL

" Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime."

1. Introduction

- PHP stands for PHP : Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use

1.1 What is a PHP File?

- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"
- PHP is compatible with almost all servers used today (Apache IIS, etc.)
- PHP is FREE to download from the official PHP resource : www.php.net

2. Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends width **?>**. A scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

However, for maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Bengkel Mudah PHP MySQL" to the browser :

```
<html>
<body>
<?php
echo "Bengkel Mudah PHP MySQL";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to the output text with PHP : **echo** and **print**. In the example above we have to used the echo statement to the output the text “Bengkel Mudah PHP MySQL”.

Note : The file must have the .php extension. In file with the .html extension, the PHP code will not be executed.

2.1 Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>
<?php
// This is a comment
/*
This is
a comment
block
/*
?>
</body>
</html>
```

2.2 Variables in PHP

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is set it can be used over and over again in your script.

All variables in PHP start with a $ sign symbol.

The correct way of setting a variable in PHP :

```
$var_name = value;
```

New PHP programmers often forget the $ sign at the beginning of the variable. In that case it will not work.

Let’s try creating a variable with a string, and a variable with a number :

```
<?php
$txt = “JKA”;
$number = 1517;
?>
```

2.3 PHP is a Loosely Typed Language

In PHP a variable does not need to be declared before being set.

In the example above, you see that you do not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on how they are set.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP the variable is declared automatically when you use it.

2.4 Variable Naming Rules

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- A variable name should not contains spaces. If a variable name is more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)

2.5 Strings in PHP

String variables are used for values that contains character strings

In this tutorial we are going to look at some of the most common functions and operators used to manipulate strings in PHP.

After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the string "PHP" to a string variable called $txt :

```
<?php
$txt = "PHP";
echo $txt;
?>
```

The output of the code above will be :

```
PHP
```

Now, lets try to use some different functions and operators to the manipulate our string.

2.6 The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenation two variables together, use the dot (.) operator :

```
<?php
$txt1 = “PHP”;
$txt2 = “MySQL”;
echo $txt1 . “ “ .$txt2;
?>
```

The output of the code above will be :

```
PHP MySQL
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

2.7 Using the strlen() function

The strlen() function is used to find the length of a string.

Let’s find the length of our string “Bengkel Mudah PHP MySQL”.

```
<?php
echo strlen(“Bengkel Mudah PHP MySQL”);
?>
```

The output of the code above will be :

```
23
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

2.8 Using the strops() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string :

```
<?php
echo strpos("Hello world!", "world");
?>
```

The output of the code above will be :

```
6
```

As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

3. PHP Operators

This section lists the different operators used in PHP.

3.1 Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus(division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

3.2 Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%=y

3.3 Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=returns true

3.4 Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x<10 && y>1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

4. Conditional Statement

4.1 The If…Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if…else statement.

Syntax

```
if (condition)
   code to be executed if condition is true;
else
   code to be executed if condition is false;
```

Example

The following example will output “Have a nice weekend!” if the current date is Friday, otherwise it will output “Have a nice day!” :

```
<html>
<body>
<?php
$d = date(“D”);
if ($d==”Fri”)
   echo “Have a nice weekend!”;
else
   echo “Have a nice day!”;
?>
</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces :

```
<html>
<body>
<?php
$d = date(“D”);
if ($d==”Fri”)
{
   echo “Have a nice weekend!”;
}
else
{
   echo “Have a nice day!”;
}
?>
</body>
</html>
```

4.2 The Elseif Statement

If you want to execute some code if one of several conditions are true use the elseif statement.

Syntax

```
if (condition)
   code to be executed if condition is true;
elseif (condition)
   code to be executed if condition is true;
else
   code to be executed if condition is false;
```

Example

The following example will output “Have a nice weekend!” if the current day is Friday, and “Have a nice Sunday!” if the current day is Sunday. Otherwise it will output “Have a nice day!” :

```
<html>
<body>
<?php
$d = date(“D”);
if ($d==”Fri”)
   echo “Have a nice weekend!”;
elseif ($d==”Sun”)
   echo “Have a nice Sunday!”;
else
   echo “Have a nice day!”;
?>
</body>
</html>
```

5. Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements :

- **while** – loops through a block of code if and as long as a specified condition is true
- **do…while** – loops through a block of code once, and then repeats the loop as long as a special condition is true
- **for** – loops through a block of code a specified number of times
- **foreach** – loops through a block of code for each element in an array

5.1 The for Statement

The for statement is the most advanced of the loops in PHP.

In it’s simplest form, the for statement is used when you know how many times you want to execute a statement or a list of statements.

Syntax

```
for (init; cond; incr)
{
   code to be executed;
}
```

Parameters :

- **init :** Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
- **cond :** Is evaluated at the beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
- **incr :** Is mostly used to increment a counter, but can any code to be executed at the end of each loop.

Note : Each of the parameters can be empty or have multiple expressions separated by commas.

- **cond :** All expressions sepearated by a comma are evaluated but the result is taken from the last part. This parameter being empty means the loop should be run indefinitely. This is useful when using a conditional break statement inside the loop for ending the loop.

Example

The following example prints the text “Hello World!” five times :

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
{
   echo “Hello World!”;
}
?>
</body>
</html>
```

5.2 The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

Syntax

```
while (condition)
code to be executed;
```

Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs :

```
<html>
<body>
<?php
$i = 1;
while ($i<=5)
{
   echo “The number is “ .$i. “<br/>”;
   $i++;
}
?>
</body>
</html>
```

5.3 The do…while Statement

The do…while statement will be execute a block of code **at least once** – it then will repeat the loop **as long as** condition is true.

Syntax

```
Do
{
   code to be executed;
}
while (condition)
```

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 5 :

```
<html>
<body>
<?php
$i = 0;
do
{
   $i++;
   echo “The number is “ .$i. “<br/>”;
}
while ($i<5);
?>
</body>
</html>
```

6. PHP Functions

A function is a block of code that can be executed whenever we need it.

Creating PHP functions :

- All functions start with the word “function()”
- Name the function – It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a “{“ – The function code starts after the opening curly brace
- Insert the function code
- Add a “}” – The function is finished by a closing curly brace

Example

A simple function that writes my name when it is called :

```
<html>
<body>
<?php
function writeMyName()
{
   echo “Ahmad Afiq Salehin Bin Ahmad Supiee”;
}
writeMyName();
?>
</body>
</html>
```

```
<html>
<body>
<?php
function writeMyName()
{
   echo “Ahmad Afiq Salehin Bin Ahmad Supiee”
}
echo “Hello world!<br/>”;
echo “My name is “;
writeMyName();
echo “.<br/>That’s right, “;
writeMyName();
echo “ is my name.”;
?>
</body>
</html>
```

The output of the code above will be :

```
Hello world!
My name is Ahmad Afiq Salehin Bin Ahmad Supiee.
That's right, Ahmad Afiq Salehin Bin Ahmad Supiee is my name.
```

6.1 PHP Functions – Adding parameters

Our first function (writeMyName()) is a very simple function. It only writes a static string.

To add more functionally to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like : writeMyName(). The parameters are specified inside the parentheses.

Example 1

The following example will write different first names, but the same last name :

```
<html>
<body>
<?php
function writeMyName($fname)
{
   echo $fname . " Ahmad Supiee.<br/>";
}
echo "My name is ";
writeMyName("Ahmad Afiq Salehin");
echo "My name is ";
writeMyName("Norleza");
echo "My name is ";
writeMyName("Mohd Suffian");
?>
</body>
</html>
```

The output of the code above will be :

```
My name is Ahmad Afiq Salehin Ahmad Supiee.
My name is Norleza Ahmad Supiee.
My Name Mohd Suffian Ahmad Supiee.
```

Example 2

The following function has two parameters :

```
<html>
<body>
<?php
function writeMyName($fname, $punctuation)
{
   echo $fname . " Ahmad Supiee" . $punctuation . "<br/>";
}
echo "My name is ";
writeMyName("Ahmad Afiq Salehin", ".");
echo "My name is ";
writeMyName("Norleza", "!");
echo "My name is ";
writeMyName("Mohd Suffian", "…");
?>
</body>
</html>
```

The output of the code above will be :

```
My name is Ahmad Afiq Salehin Ahmad Supiee.
My name is Norleza Ahmad Supiee!
My name is Mohd Suffian Ahmad Supie…
```

6.2 PHP Functions – Return values

Functions can also be used to return values.

```
<html>
<body>
<?php
function add($x,$y)
{
   $total = $x+$y;
   return $total;
}
echo "1+16=" . add(1,16);
</body>
</html>
```

The output of the code above will be :

```
1+16=17
```

7. PHP Form Handling

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Form example :

```
<html>
<body>
<form action=”welcome.php” method=”post”>
Name : <input type=”text” name=”name” />
Age : <input type=”text” name=”age” />
<input type=”submit” />
</form>
</body>
</html>
```

The example HTML page above contains two input fields and a submit button. When the user fills in this form and click on the submit button, the form data is sent to the “welcome.php” file.

The “welcome.php” files looks like this :

```
<html>
<body>
Welcome <?php echo $_POST[‘name’];?>.<br/>
You are <?php echo $_POST[‘age’];?> years old.
</body>
</html>
```

A sample output of the above script may be :

```
Welcome Afiq.
You are 22 years old.
```

8. The $_GET Variable

The $_GET variable is an array of variable names and values sent by the HTTP GET method.

The $_GET variable is used to collect values from a form with method=”get”. Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser’s address bar) and it has limits on the amount of information to send (max. 100 characters).

Example

```
<form action=”welcome.php” method=”get”>
Name : <input type=”text” name=”name” />
Age : <input type=”text” name=”age” />
<input type=”submit” />
</form>
</body>
</html>
```

When the user clicks the “Submit button, the URL sent could look something like this:

```
http://www.w3schools.com/welcome.php?name=Afiq&age=22
```

The “welcome.php” file can now use the $_GET variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_GET array) :

```
Welcome <?php echo $_GET[‘name’]; ?>.<br />
You are <?php echo $_GET[‘age’]; ?> years old!
```

8.1 Why use $_GET?

Note : When using the $_GET variable all variable names and values are displayed in the URL. So this method should not be used when sending passwords or other sensitive information! However, because the variables are displayed in the URL, it it possible to bookmark the page. This can be useful in some cases.

Note : The HTTP GET method is not suitable on large variable values; the value cannot exceed 100 characters.

9. The $_POST Variable

The $_POST variable is an array of variable names and values sent by the HTTP POST method. The $_POST variable is used to collect values from a form with method=”post”. Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

Example

```
<form action=”welcome.php” method=”post”>
Enter your name : <input type=”text” name=”name” />
Enter your age : <input type=”text” name=”age” />
<input type=”submit” />
</form>
```

When the user clicks the “Submit” button, the URL will not contain any form data, and will look something like this :

```
http://www.w3schools.com/welcome.php
```

The “welcome.php” file can now use the $_POST variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_POST array) :

```
Welcome <?php echo $_POST[‘name’];?>.<br/>
You are <?php echo $_POST[‘age’];?> years old!
```

9.1 Why use $_POST?

- Variables sent with HTTP POST are not shown in the URL
- Variables have no length limit

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

10. The $_REQUEST Variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Example

```
Welcome <?php echo $_REQUEST[‘name’];?>.<br/>
You are <?php echo $_REQUEST[‘age’];?> years old!
```

11. The $_REQUEST Variable

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

11.1 Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note : The session_start() function must appear BEFORE the <html> tags :

```
<?php session_start();?>
<html>
<body>
</body>
</html>
```

The code above will register the user's session with the server, aloow you to start saving user information, and assign a UID for that user's session.

11.2 Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable :

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
// retrieve session data
echo "Pageviews = ".$_SESSION['views'];
?>
</body>
</html>
```

Output :

```
Pageviews = 1
```

In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1 :

```
<?php
session_start();
if(isset($_SESSION['views']))
   $_SESSION['views']=$_SESSION['views']+1;
else
   $_SESSION['views']=1;
echo "Views = ".$_SESSION['views'];
?>
```

11.3 Destroying a Session

If you wish to delete some session data, you can use the unset().

The unset() function is used to free the specified session variable :

```
<?php
unset($_SESSION['views']);
?>
```

12. PHP Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie to. With PHP, you can both create and retrieve cookie values.

12.1 How to Create a Cookie?

The setcookie() function is used to set a cookie.

Note : The setcookie() function must appear BEFORE the <html> tag.

Syntax

```
setcookie(name, value, expire, path, domain);
```

Example 1

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour :

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>
<html>
…..
```

Note : The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Example 2

You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

```
<?php
$expire = time()+60*60*24*30;
setcookie(“user”, “Alex Porter”, $expire);
?>
<html>
…..
```

In the example above the expiration time is set to month (60 sec * 60 min *60 hours * 30 days).

12.2 How to Retrieve a Cookie Value?

The PHP $_COOKIE variable is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named “user” and display it on a page :

```
<?php
// Print a cookie
echo $_COOKIE[‘user’];
// A way to view all cookies
print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set :

```
<html>
<body>
<?php
if(isset($_COOKIE[‘user’]))
   echo “Welcome “ .$_COOKIE[‘user’]. ”!<br/>”;
else
   echo “Welcome guest!<br/>”;
?>
</body>
</html>
```

12.3 How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is the past.

Delete example :

```
<?php
// set the expiration date to one hour ago
setcookie(“user”, “”, time()-3600);
?>
```

SAMPLE CODING

DATABASE CONNECTION

```
<?php
$host = "host";
$username = "username";
$password = "password";
$db_name = "db_name";

$connect = new mysqli($host, $username, $password, $db_name);
if($connect->connect_errno)
{
	echo "Failed to connect to MySQL : ".$connect->error;
}
?>
```

CREATE / INSERT RECORD

```
<?php
$sql_insert_record = "INSERT INTO table (att1, att2, att3) VALUES ('Att1', 'Att2', 'Att3')";
if($result_insert_record = $connect->query($sql_insert record))
{
	echo "Success!";
}
else
{
	echo "Failed!";
}
?>
```

READ / VIEW RECORD

```
<?php
$sql_view_record = "SELECT * FROM table";
if($result_view_record = $connect->query($sql_view_record))
{
	$rows_view_record = $result_view_record->fetch_array();
	$total_view_record = $result_view_record->num_rows;
}
?>
```

UPDATE / EDIT RECORD

```
<?php
$sql_edit_record = "UPDATE table SET att1 = 'Att1', att2 = 'Att2', att3 = 'Att3' WHERE id =
".$_GET['id']."";
if($result_edit_record = $connect->query($sql_edit_record))
{
	echo "Success!";
}
else
{
	echo "Failed!";
}
?>
```

DELETE / REMOVE RECORD

```
<?php
$sql_delete_record = "DELETE FROM table WHERE id = ".$_GET['id']."";
if($result_delete_record = $connect->query($sql_delete_record))
{
	echo "Success!";
}
else
{
	echo "Failed!";
}
?>
```

LOGOUT

```
<?php
unset($_SESSION['id']);
?>
```

SECURE PAGE

```
<?php
if(!isset($_SESSION['id']))
{
	echo "<script language=javascript>alert('Sila log masuk terlebih
dahulu.');window.location='login.php';</script>";
}
?>
```

PRACTICE

EXERCISE

1. Create 'bengkel' database :

pentadbir	pentadbir_id, int (100), AI pentadbir_nama, varchar (1000), NULL pentadbir_username, varchar (1000), NULL pentadbir_password, varchar (1000), NULL
pelajar	pelajar_id, int (100), AI pelajar_nama, varchar (1000), NULL pelajar_email, varchar (1000), NULL pelajar_notelefon, varchar (1000), NULL pelajar_alamat, varchar (1000), NULL pelajar_daerah, varchar (1000), NULL pelajar_poskod, varchar (1000), NULL pelajar_negeri, varchar (1000), NULL pelajar_username, varchar (1000), NULL pelajar_password, varchar (1000), NULL pelajar_gambar, varchar (1000), NULL
kandungan	id, int (100) kandungan, text, NULL

2. Open config.php file :

```
<?php
$host = "localhost";
$username = "root";
$password = "";
$db_name = "bengkel";

$connect = new mysqli($host, $username, $password, $db_name);
if($connect->connect_errno)
{
	echo "Failed to connect to MySQL : ".$connect->error;
}
?>
```

3. Open register.php file :

```
<?php
session_start();
include('config.php');

if(isset($_POST['daftar']))
{
	$nama = $connect->real_escape_string(stripslashes($_POST['nama']));

	$sql_daftar = "INSERT INTO pelajar (pelajar_nama) VALUES ('".$nama."')";
	if($result_daftar = $connect->query($sql_daftar))
	{
		echo "Berjaya!";
	}
	else
	{
		echo "Gagal!";
	}
}
?>
```

4. Open login.php file :

```
<?php
$sql_login_pentadbir = "SELECT * FROM pentadbir WHERE pentadbir_username = '".$username."'
AND pentadbir_password = '".$password."'";
if($result_login_pentadbir = $connect->query($sql_login_pentadbir))
{
	$rows_login_pentadbir = $result_login_pentadbir->fetch_array();
	if($total_login_pentadbir = $result_login_pentadbir->num_rows)
	{
		$_SESSION['pentadbir_id'] = $rows_login_pentadbir['pentadbir_id'];
		echo "<script language=javascript>window.location='admin/index.php';</script>";
	}
	else
	{
		echo "<script language=javascript>alert('Log masuk tidak berjaya. Sila cuba
lagi.');window.location='login.php';</script>";
	}
}
?>
```

5. Open admin/index.php file :

```
<?php
if(!isset($_SESSION['pentadbir_id']))
{
	echo "<script language=javascript>alert('Sila log masuk terlebih
dahulu.');window.location='../login.php';</script>";
}

if($_GET['action']=="logout")
{
	unset($_SESSION['pentadbir_id']);
	echo "<script language=javascript> window.location='../login.php';</script>";
}

$sql_pentadbir = "SELECT * FROM pentadbir WHERE pentadbir_id = '".$_SESSION['pentadbir_id']."'";
if($result_pentadbir = $connect->query($sql_pentadbir))
{
	$rows_pentadbir = $result_pentadbir->fetch_array();
	$total_pentadbir = $result_pentadbir->num_rows;
}
?>
```

6. Open admin/student-list.php file :

```
<?php
$sql_pelajar = "SELECT * FROM pelajar";
if($result_pelajar = $connect->query($sql_pelajar))
{
	$rows_pelajar = $result_pelajar->fetch_array();
	$total_pelajar = $result_pelajar->num_rows;
	$num_pelajar = 0;
}
?>
```

```
<?php if($total_pelajar>0) { do { ?>
<tr bgcolor="#FFFFFF">
	.....
</tr>
<?php } while($rows_pelajar = $result_pelajar->fetch_array()); } ?>
```

7. Open admin/view-student.php file :

```
<?php
$sql_pelajar = "SELECT * FROM pelajar WHERE pelajar_id = ".$_GET['id']."";
if($result_pelajar = $connect->query($sql_pelajar))
{
	$rows_pelajar = $result_pelajar->fetch_array();
	if(!$total_pelajar = $result_pelajar->num_rows)
	{
		echo "<script language=javascript>window.location='student-list.php';</script>";
	}
}
else
{
	echo "<script language=javascript window.location='student-list.php';</script>";
}
?>
```

8. Open admin/edit-student.php file :

```
<?php
$sql_edit_pelajar = "SELECT * FROM pelajar WHERE pelajar_id != '".$_GET['id']."' AND
pelajar_username = '".$username."'";
if($result_edit_pelajar = $connect->query($sql_edit_pelajar))
{
	if($total_edit_pelajar = $result_edit_pelajar->num_rows)
	{
		echo "<script language=javascript>window.location='edit-
student.php?id=".$_GET['id']."';</script>";
	}
	else
	{
		$sql_edit_pelajar = "UPDATE pelajar SET pelajar_nama = '".$nama."' WHERE
pelajar_id = '".$_GET['id']."'";
		if($result_edit_pelajar = $connect->query($sql_edit_pelajar))
		{
			echo "<script language=javascript>window.location='edit-
student.php?id=".$_GET['id']."';</script>";
		}
		else
		{
			echo "<script language=javascript>window.location='edit-
student.php?id=".$_GET['id']."';</script>";
		}
	}
}
?>
```

9. Open admin/student-list.php file :

```
<?php
if(($_GET['action']=="delete") && ($_GET['id']!=NULL))
{
	$sql_padam_pelajar = "DELETE FROM pelajar WHERE pelajar_id = '".$_GET['id']."'";
	if($result_padam_pelajar = $connect->query($sql_padam_pelajar))
	{
		echo "<script language=javascript>window.location='student-list.php';</script>";
	}
	else
	{
		echo "<script language=javascript>window.location='student-list.php';</script>";
	}
}
?>
```

10.Open student/index.php :

```
// Select student where login (same as admin/index.php file)
```

11.Open student/edit-profile.php :

```
// Update student where login session
```

12.Upload picture

```
<?php
$gambar = $connect->real_escape_string(stripslashes($_FILES['gambar']['name']));
$target = "../images/student/";
$target = $target.basename($gambar);

$sql_edit_pelajar = "UPDATE pelajar SET pelajar_gambar = '".$gambar."' WHERE pelajar_id =
'".$_SESSION['pelajar_id']."'";
if($result_edit_pelajar = $connect->query($sql_edit_pelajar))
{
	move_uploaded_file($_FILES['gambar']['tmp_name'], $target);
	echo "<script language=javascript>window.location='edit-profile.php';</script>";
}
else
{
	echo "<script language=javascript>window.location='edit-profile.php';</script>";
}
?>
```

BONUS!!

JQUERY DATATABLES

```
<style type="text/css" title="currentStyle">
@import "../js/dataTables/css/demo_page.css";
@import "../js/dataTables/css/demo_table_jui.css";
@import "../js/dataTables/themes/smoothness/jquery-ui-1.8.4.custom.css";
</style>
<script type="text/javascript" language="javascript" src="../js/dataTables/jquery.js"></script>
<script type="text/javascript" language="javascript"
src="../js/dataTables/jquery.dataTables.js"></script>
<script type="text/javascript" charset="utf-8">
$(document).ready(function() {
	oTable = $('#example').dataTable({
		"bJQueryUI": true,
		"sPaginationType": "full_numbers",
		"aoColumnDefs": [{ 'bSortable': false, 'aTargets': [ 0,4 ] }],
	});
} );
</script>
```

```
<table width="750" border="0" cellspacing="1" cellpadding="5" bgcolor="#CCCCCC" class="display"
id="example">
.....
</table>
```

JQUERY WYSIWYG EDITOR

```
<script src="../js/ckeditor/ckeditor.js"></script>
```

```
<script type="text/javascript">
CKEDITOR.replace( 'kandungan', {
	filebrowserBrowseUrl: '../js/ckeditor/kcfinder/browse.php?type=files',
	filebrowserImageBrowseUrl: '../js/ckeditor/kcfinder/browse.php?type=images',
	filebrowserFlashBrowseUrl: '../js/ckeditor/kcfinder/browse.php?type=flash',
	filebrowserUploadUrl: '../js/ckeditor/kcfinder/upload.php?type=files',
	filebrowserImageUploadUrl: '../js/ckeditor/kcfinder/upload.php?type=images',
	filebrowserFlashUploadUrl: '../js/ckeditor/kcfinder/upload.php?type=flash'
});
</script>
```

PHP MAIL FUNCTION

1. Open contact-us.php file :

```
<?php
$to = "afiq_salehin@yahoo.com";
$subject = ucwords($tajuk);
$message = "
<html>
<head>
<title>".ucwords($tajuk)."</title>
</head>
<body>
        <p>".$kandungan."</p>
</body>
</html>
";
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
$headers .= 'To: Ahmad Afiq Salehin Bin Ahmad Supiee <afiq_salehin@yahoo.com>' . "\r\n";
$headers .= 'From: '.ucwords($nama).' <'.$email.'>' . "\r\n";
mail($to,$subject,$message,$headers);
?>
```

" Thank You & Good Luck "