**ASSIGNMENT 3**

**GAGAN KAPILA-056-214425**

* **HOOKS**

WordPress hook is a feature that allows you to manipulate a procedure without modifying the file on WordPress core. A hook can be applied both to action (action hook) and filter (filter hook).

* **Purpose of Hooks**

The primary purpose of hooks is to automatically run a function. In addition, this technique also has the ability to modify, extend, or limit the functionality of a theme or plugin.

Here is an example of a hook in WordPress:

function mytheme\_enqueue\_script()

{wp\_enqueue\_script( 'my-custom-js', 'custom.js', false );}

add\_action( 'wp\_enqueue\_scripts', 'mytheme\_enqueue\_script' );

The example above shows that the hook is created to connect the mytheme\_enqueue\_script function with the wp\_enqueue\_scripts action. This hook triggers a new action on your site, therefore, it is called an action hook.

Hooks are often used in making plugin components of an application. It is not only used in content management systems (CMS) like WordPress but is also commonly used in e-commerce and intranet sites within an enterprise.

Furthermore, as we mentioned above, hooks are divided into two categories: action and filter. Action hook is used to add a process, while filter hook functions to change or modify the value of a process.

* **How to Use WordPress Hooks?**

Using hooks in WordPress does require a bit of knowledge about HTML and PHP. However, even if you are a complete beginner, creating both action and filter hooks might not be as difficult as you think.

You only need to go to your post page, then switch to the text editor. When you are there, you can paste the hooks that you have copied from other sites or created yourself.

* **Creating an Action Hook**

To add an action hook, you must activate the add\_action () function in the WordPress plugin. This function can be activated by writing the patterns below in your functions.php file :

add\_action('function name of target\_hook', 'The\_name\_of\_function\_you\_want\_to\_use' ,'priority\_scale')

As we see above, hooks use priority scale to function properly. This scale is an automatic ordinal value based on a scale from 1 to 999. It defines the order precedence for functions associated with that particular hook.

A lower value of priority means that the function will be run earlier, while the higher one will be run later. The scale will show the output sequence of the functions installed when using the same target\_hooks.

The default value of priority\_scale is 10. You can arrange the scale according to the number of your target\_hooks.

Here is an example of an action hook:

<?php

add\_action( 'wp\_print\_footer\_scripts', 'hostinger\_custom\_footer\_scripts' );

function webhost\_custom\_footer\_scripts(){

?>

<script>//fill the footer scripts right here</script>

<?php

}

?>

Note the pattern in the example above:

<?php is the place where you put the hook to function

add\_action is the command for creating the action hook

wp\_print\_footer\_scripts is the target\_hook that you link to a new function

Hostinger\_custom\_footer\_scripts is the function installed and linked to the target\_hook

<script> represents the text you want to show on the target\_hook (in this case, it is the wp\_print\_footer\_scripts)

* **Creating a Filter Hook**

You can create a filter hook by utilizing apply\_filters() function. The hook filter is used to modify, filter, or replace a value with a new one.

Just like using an action hook, it also has a function that filters a value with the associated filter hook functions (apply\_filter).

What is more, it has the function to add a hook filter to be associated with another function (add\_filter).

Here is an example of a filter hook:

$score = 100;

echo "Current score is : ". apply\_filters( 'change\_score', $score );

$score = 100 is the initial score value

echo “Current score is: ” represents the script that you’re showing

apply\_filters is the command to create a filter hook

‘change\_score’, $score is the function to be filtered

This is the filter:

add\_filter( 'change\_score', 'function\_change\_score' );

function function\_change\_score( $score ){

? $score+=100;

? return $score;

}

Add\_filter is created to connect the filter hook with a new function

‘Change\_score‘ is the target hook that will be modified

‘Function\_change\_score‘ is a new function that will affect the initial value

? $score+=100; is the code to add more value to the initial value ($score)

? return $score; is the code to show the new value at the end

The result should look like this:

Current score: 200

**SOME INFROMATIONS ABOUT TEMPLATE TAGS**

Template tags are basically PHP functions, so any PHP function defined by a WordPress plugin or theme can be used as a template tag. To use a theme function as a template tag, the function should be defined in the theme’s functions.php file. Template tags are PHP functions, so they can also be used inside other PHP functions and template tags. In the example below, we have defined a function that displays some text.

Example- function donation\_request() { $this\_article = wp\_title('',true); echo '

Hi, if you enjoyed reading '.$this\_article.' please consider [donating](http://www.example.com/donate/).'; }

**Functions:**

Action Hooks are a very useful tool in WordPress and they are used to perform functions (actions) in specific places of a theme or plugin. Many themes and plugins, such as Total, use action hooks as an easy way for users to modify the output of the project or to add their own custom code. o my\_project\_remove\_menu\_page() Custom function to remove useless elements from WordPress admin dashboard. o add\_action() Function add\_action() calls the hook. The key moment is admin\_menu. which means that the function is called when WordPress gets ready to display the admin menu. Parameters to add to the function :

o First parameter (the handle) : Name given to your file (e.g. : your project's name).

o Second parameter (file address) : Using get\_template\_directory\_uri() supplies the path to the theme. To declare a CSS file in the theme's root, the use get\_stylesheet\_uri() alone.

o Array() : The array is used to have WordPress scripts and stylesheets declared first.

o Version number : the version number is used to indicate the version number of the files. When you update the file, simply increase the version number (e.g. : 1.1) so the browser's files in cache won't be used.

o Boolean : A boolean is used last in scripts declaration. If true, the script is placed at the end of the page, if false the script is placed at the beginning of the page.

o jQuery : In order to use the latest version of jQuery, since WordPress does it another way for older Explorer browsers, we have declared another version.