<https://darshanbhavsar.wordpress.com/2015/02/03/magento-2-directory-structure/>

<https://www.silksoftware.com/magento-module-creator/> for creating modules in magento1 and magento2

<https://validator.prestashop.com/generator> :-- prestashop basic module generator:- [manitak@chetu.com/12345678](mailto:manitak@chetu.com/12345678)

https://darshanbhavsar.wordpress.com/2015/01/12/magento-2-installation/

# Magento 2 Directory Structure

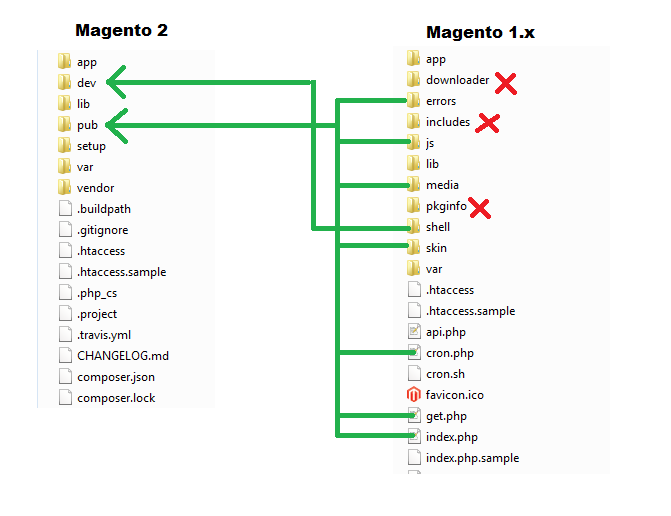
[FEBRUARY 3, 2015](https://darshanbhavsar.wordpress.com/2015/02/03/magento-2-directory-structure/)[DARSHANB](https://darshanbhavsar.wordpress.com/author/darshanb/)

I have gone through the directories of Magento 2 and i found that directory structure has changed significantly. It is important to understand these difference between Magento 1.x and Magento 2. In this post i will describe significant changes of directory and file structure.

### Root Folder Structure

First let’s compare root folder structure of Magento 1.x and Magento 2. There are major changes applied at root level. Very first thing you will identify is that some new folders are introduced and some of them are removed.

* media and errors folder are moved to pub folder
* skin and js folder are moved to pub/static folder. Now it is easy to maintain static content via CDN.
* js, css, images are again divided into pub/static/adminhtml and pub/static/frontend for backend and frontend area respectively.
* dev folder contains various tools for developers such as migration tools and tests. shell is moved to dev folder
* downloader, includes and pkginfo folder are no more with magento 2.
* setup directory is for installation process
* get.php, cron.php and index.php files are moved to pub folder.



There are 2 index.php files one in Magento root folder and another in pub folder. You can configure your server to run your application through any of this index.php file. Using pub/index.php is a good option as it will stop access of application files and it is good for security.

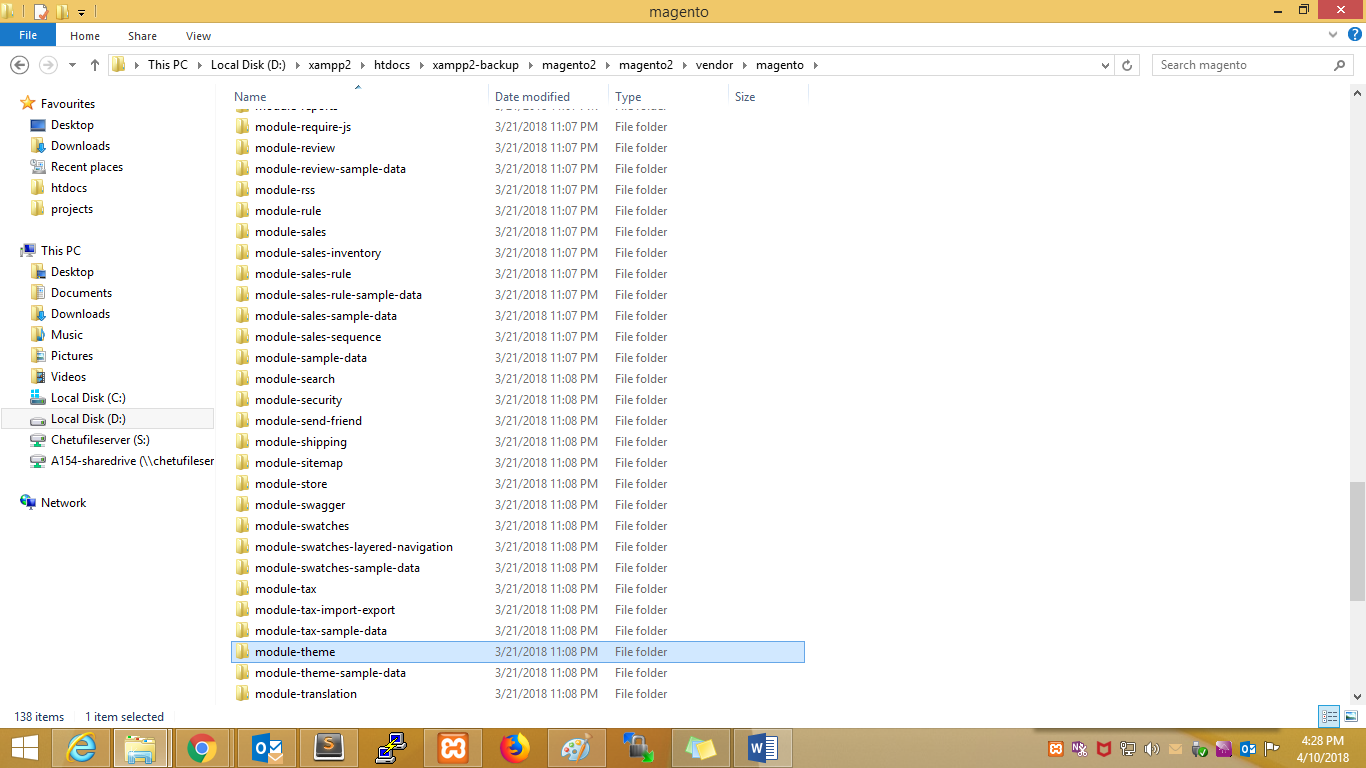
**env.php**

Database connection settings, backend name, session save settings and cache settings are declared in app/etc/env.php

**config.php**

config.php has list of active and inactive modules

vendor folder contains base theme and module:-

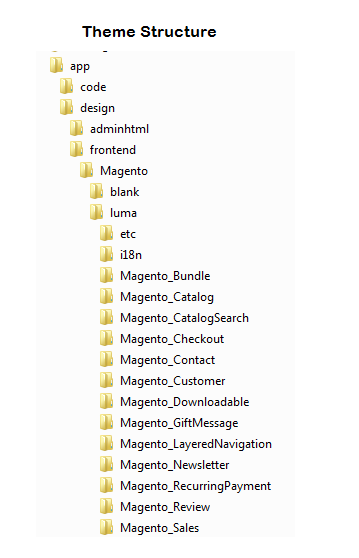


**Theme Folder Structure**

Magento 2 comes with very drastic changes on the frontend. Significant changes has been apply on theme structure for performance improvements.

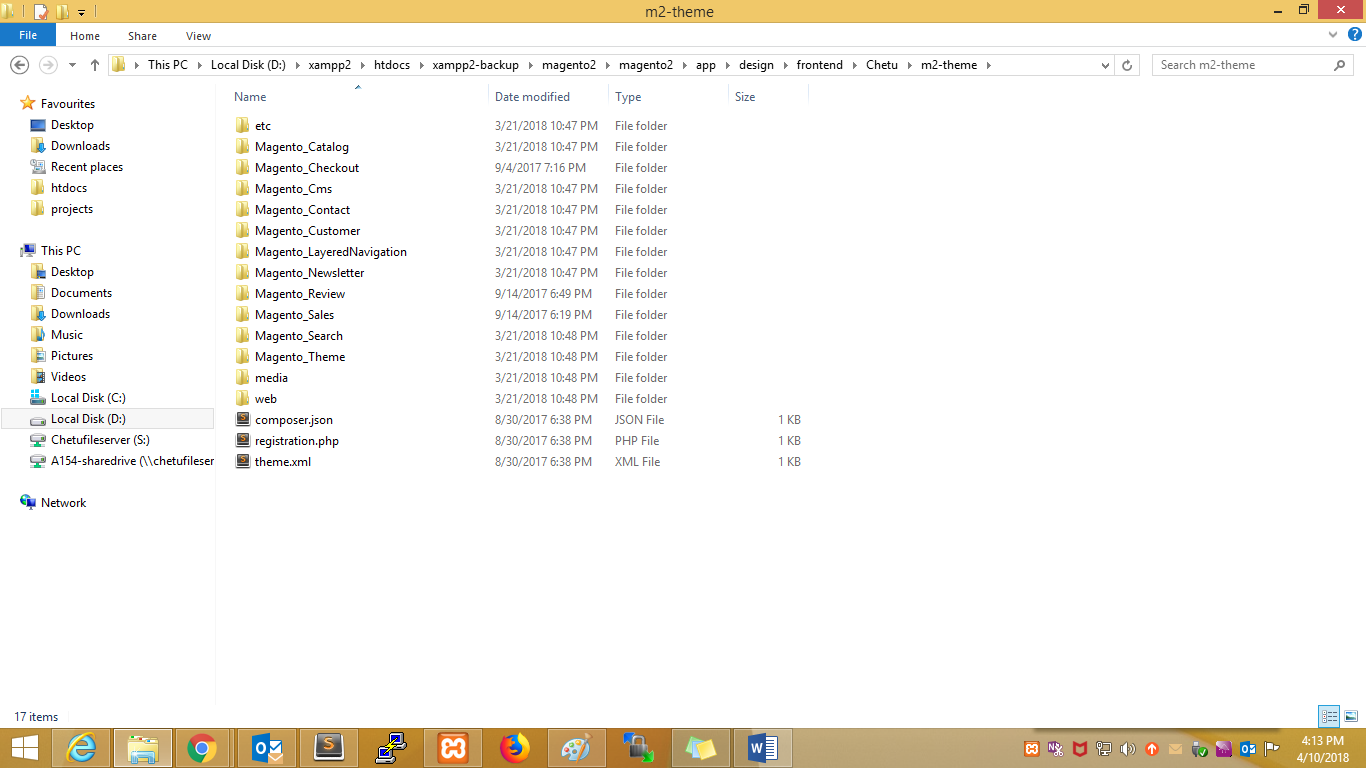
As like earlier version magento 1.x base package ‘app/design/frontend/base/’ will no more exists. Each core module in Magento 2 has new folder introduced which is called ‘view’. This ‘view’ folder contains all module specific files like email template, js, layout xmls, phtml for frontend and backend.

Themes are grouped by ‘vendor’ which we known as ‘package’ in magento 1.x. Theme structure in Magento 2 is like ‘app/design/frontend/vendor\_name/theme\_name’.



Custom theme:-

Chetu vendor :- m2-theme



Theme.xml contains:-

<theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Config/etc/theme.xsd">

<title>m2-theme</title>

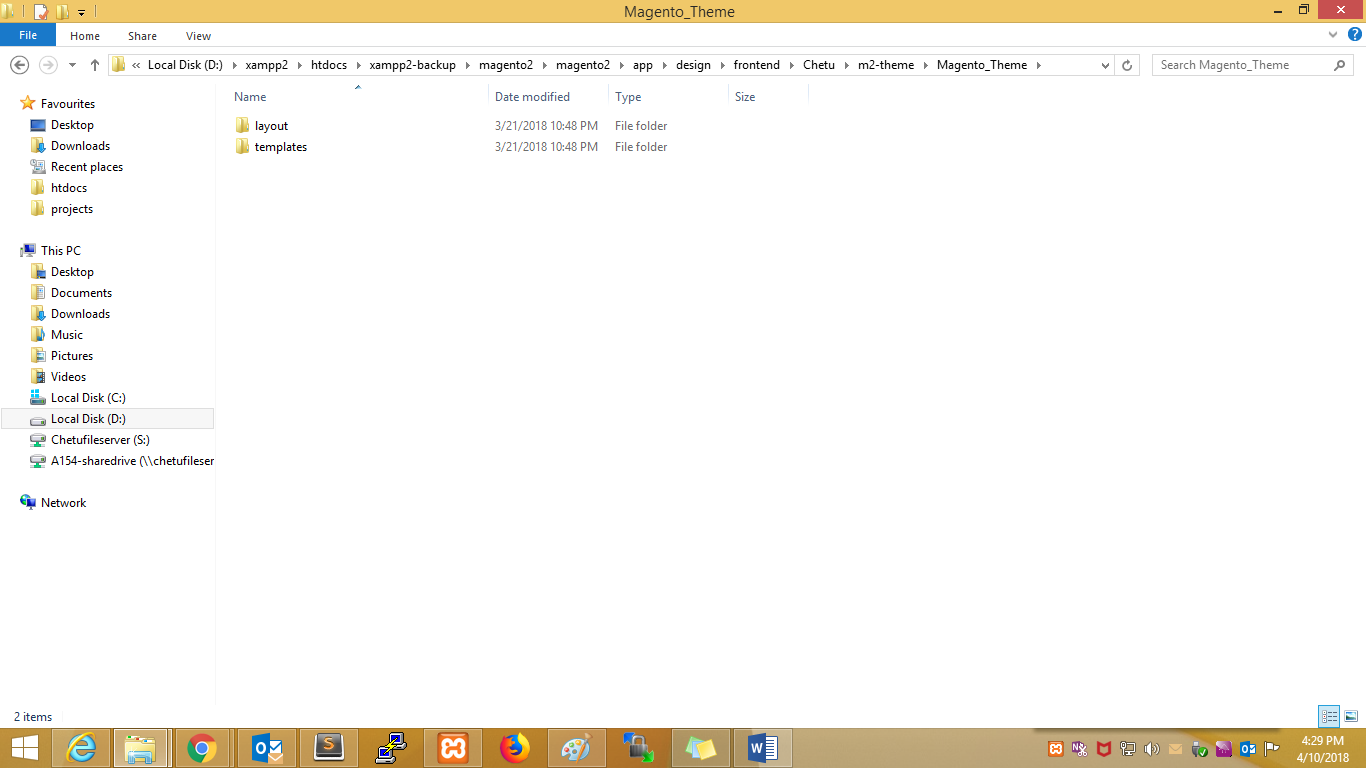
<parent>Magento/blank</parent>

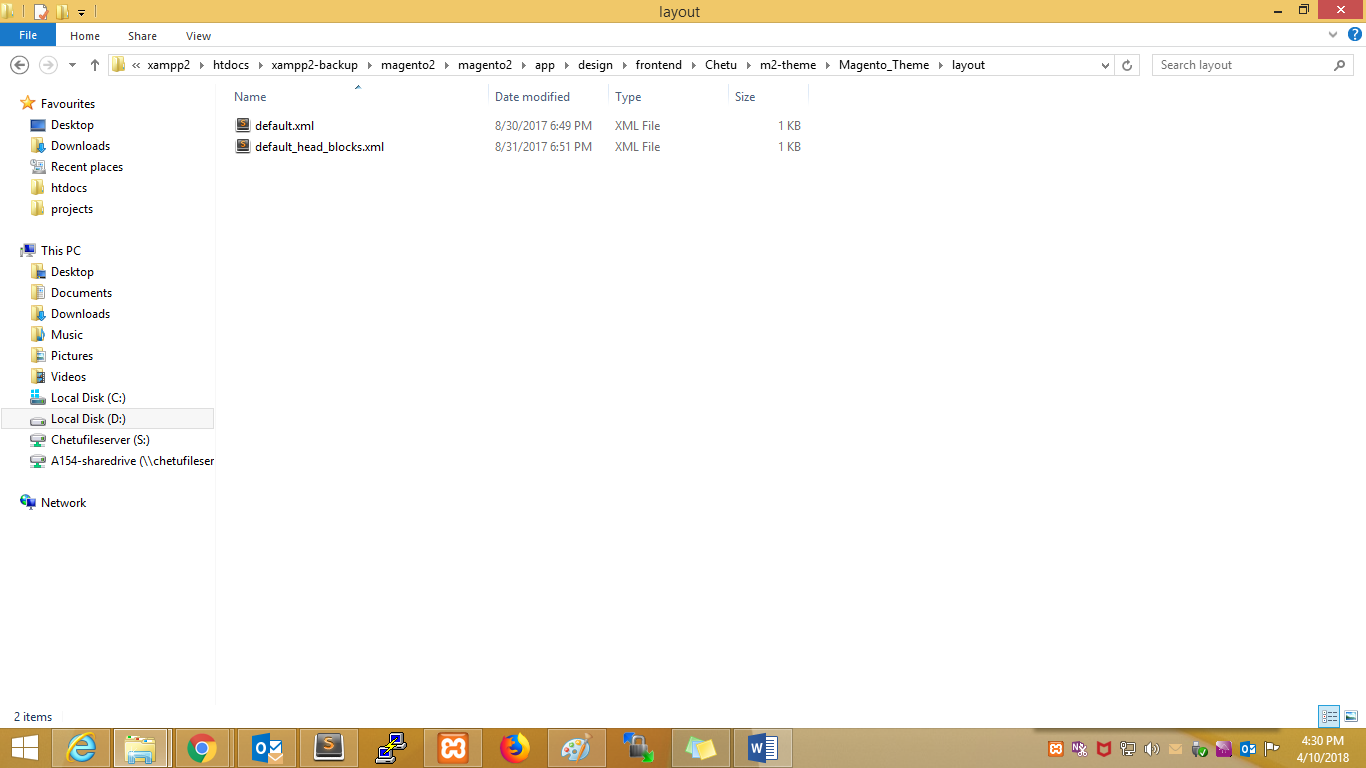
<media>

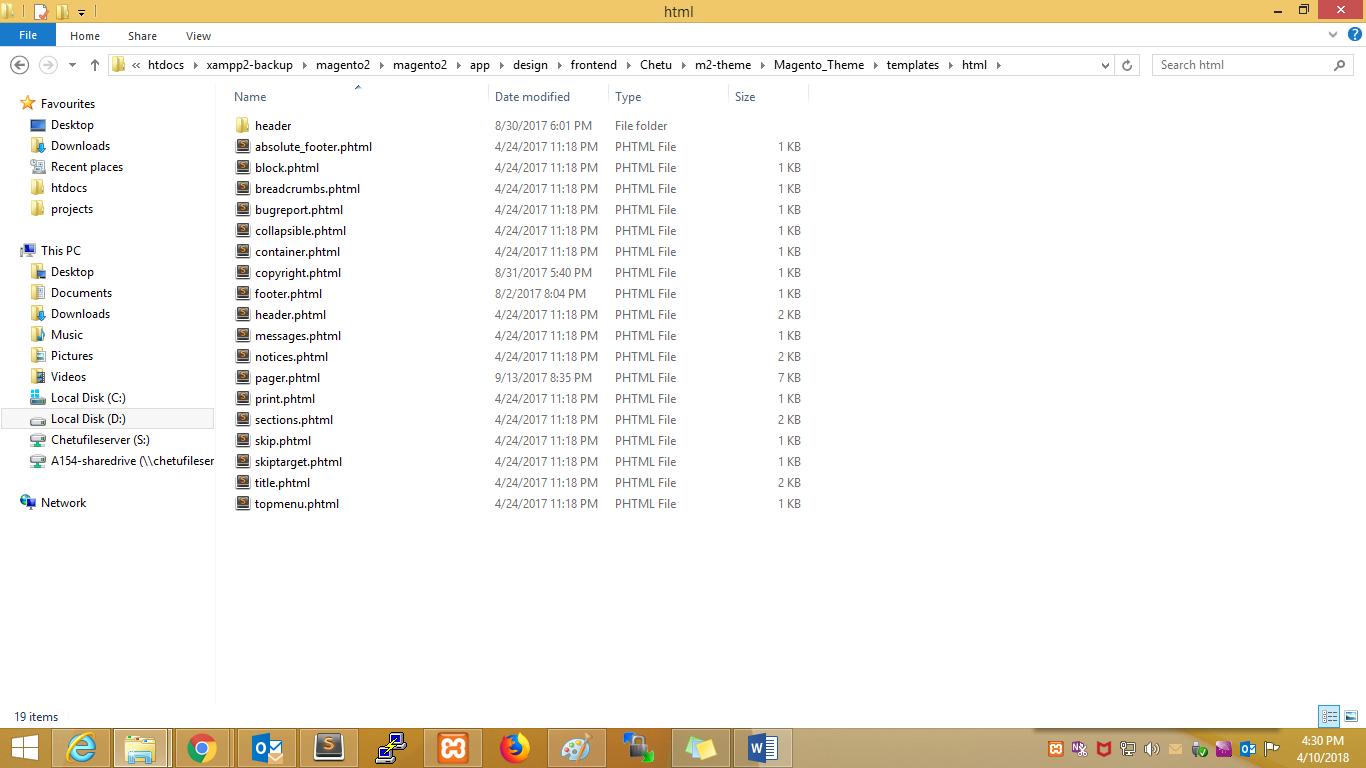
<preview\_image>media/theme\_img.jpg</preview\_image>

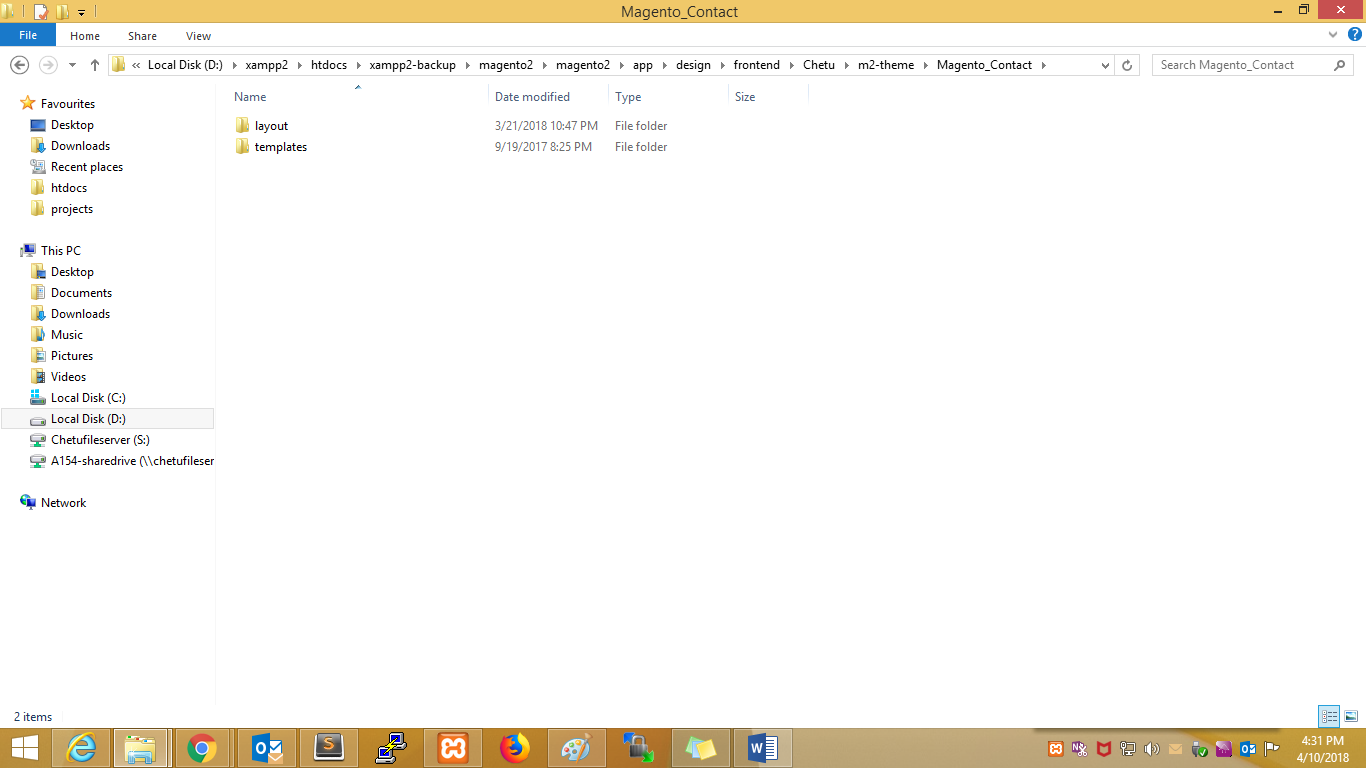
</media>

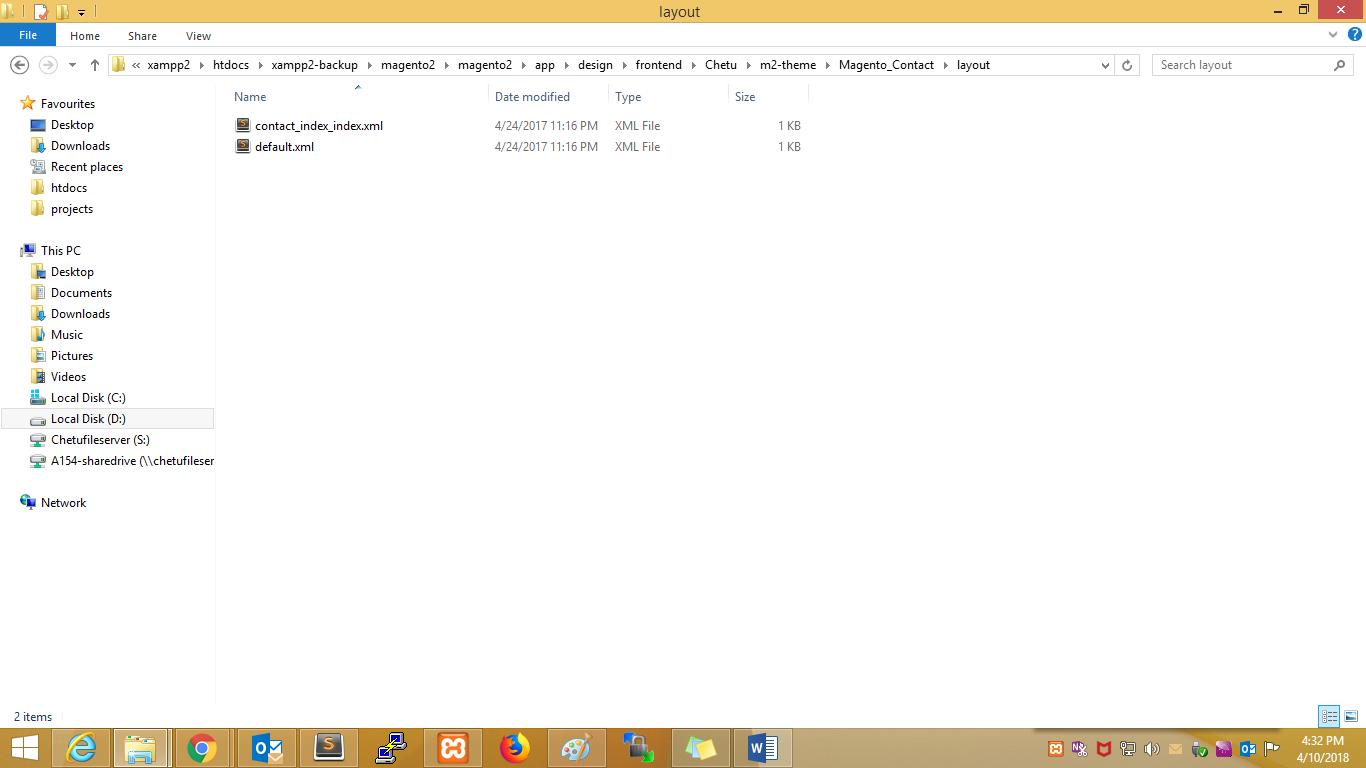
</theme>

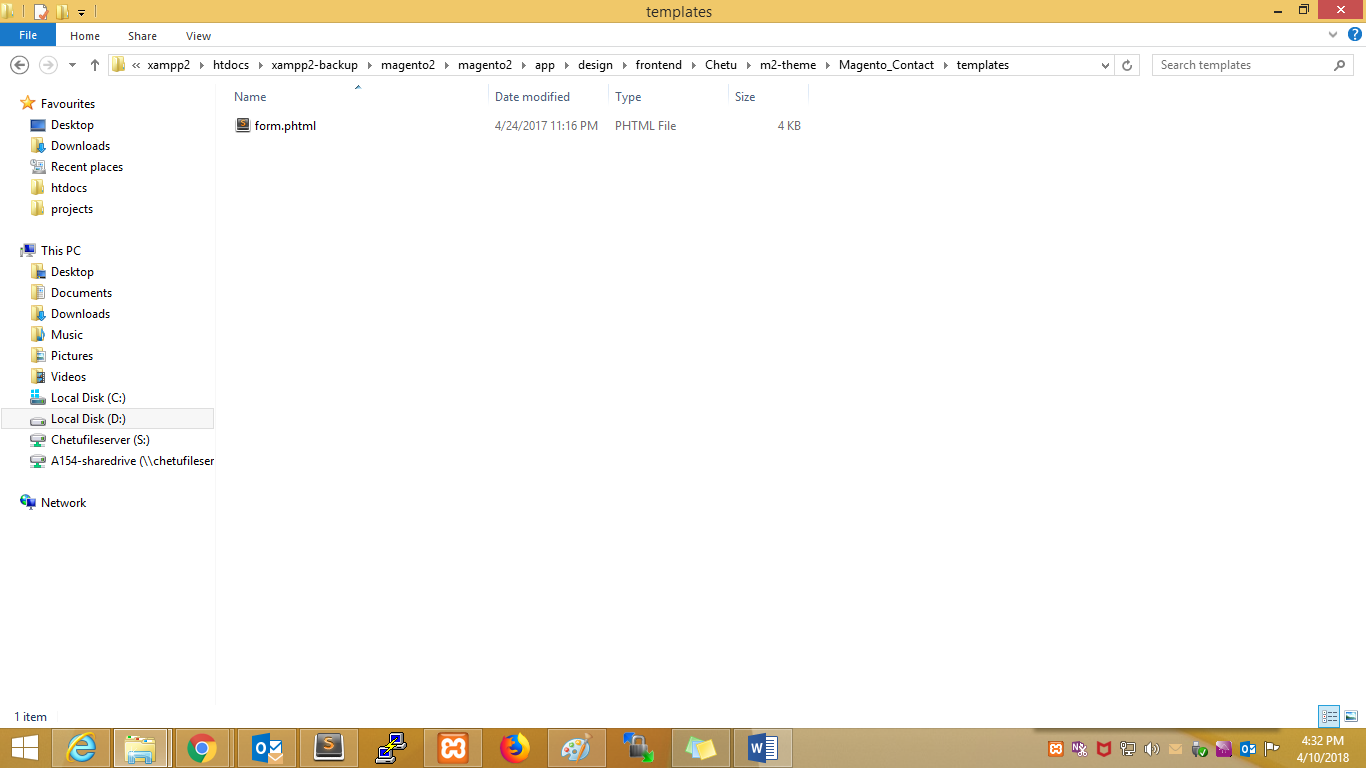




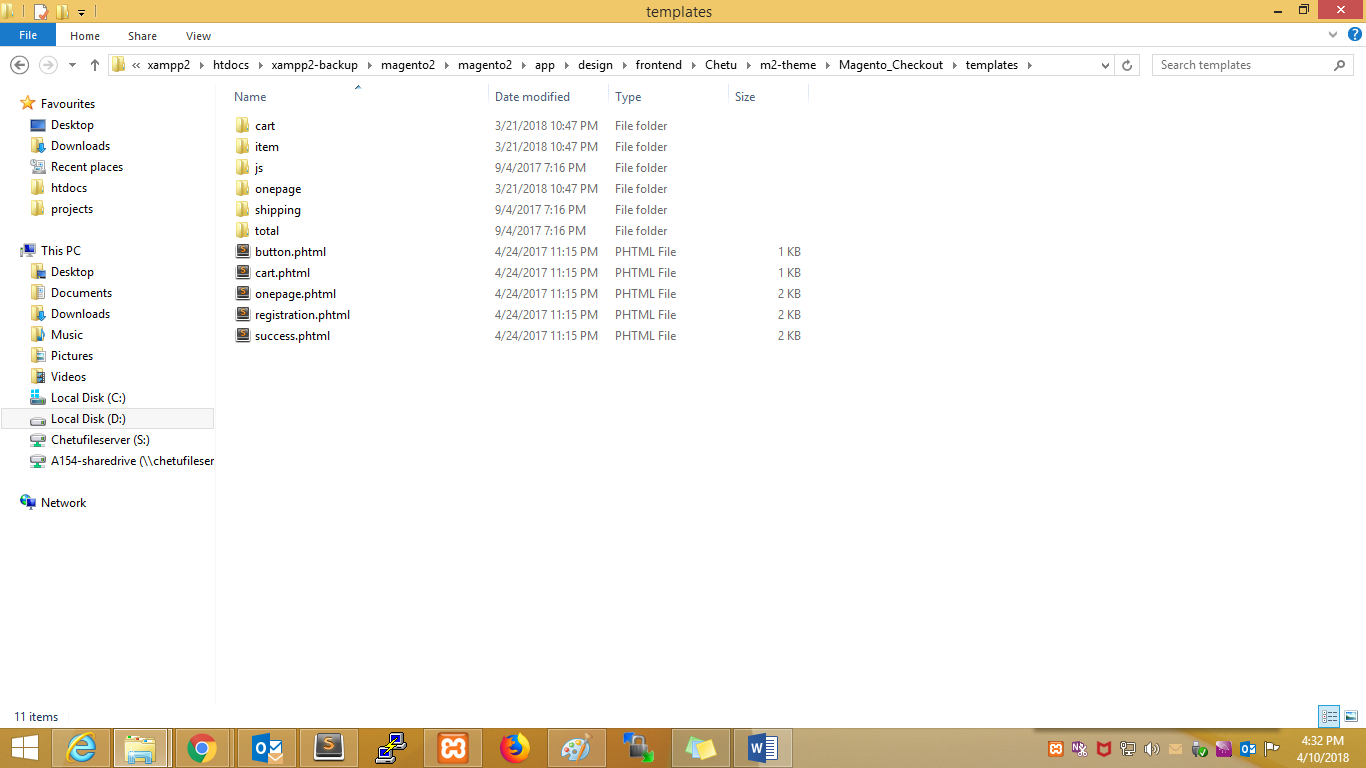
Overided module template files inside theme:-Magento\_Contact (in vendor it is module-cpntact)







Magento\_Checkout :-



**Core module**

Magento core components are exists in your\_magento\_web\_root/vendor/magento directory.

**Module Folder Structure**

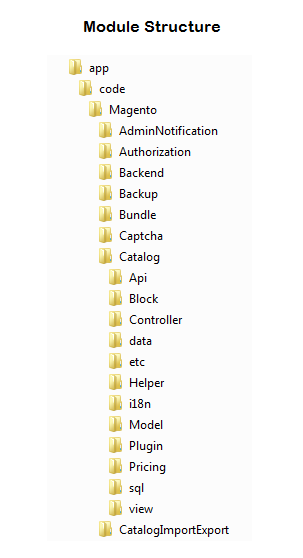
In Magento 2 app/etc/modules folder is removed and module’s declaration is moved to app/code/Vendor\_name/Module\_name/etc/module.xml file. Each module has its own registration.php file in its root. This file is for Magento to identify it as a module.

app/code/core, /local, /community folders are removed and module structure becomes like  
[root]/app/code/[Namespace]/[Module]

* Namespace is a name of a module’s vendor
* Module is a name assigned to a module by its vendor

Next to Controller, Model, Helper, Block, etc Magento 2 has introduced new folders which are:

* Api
* i18n – contains module specific translation
* view – contains module specific layout xmls, phtml

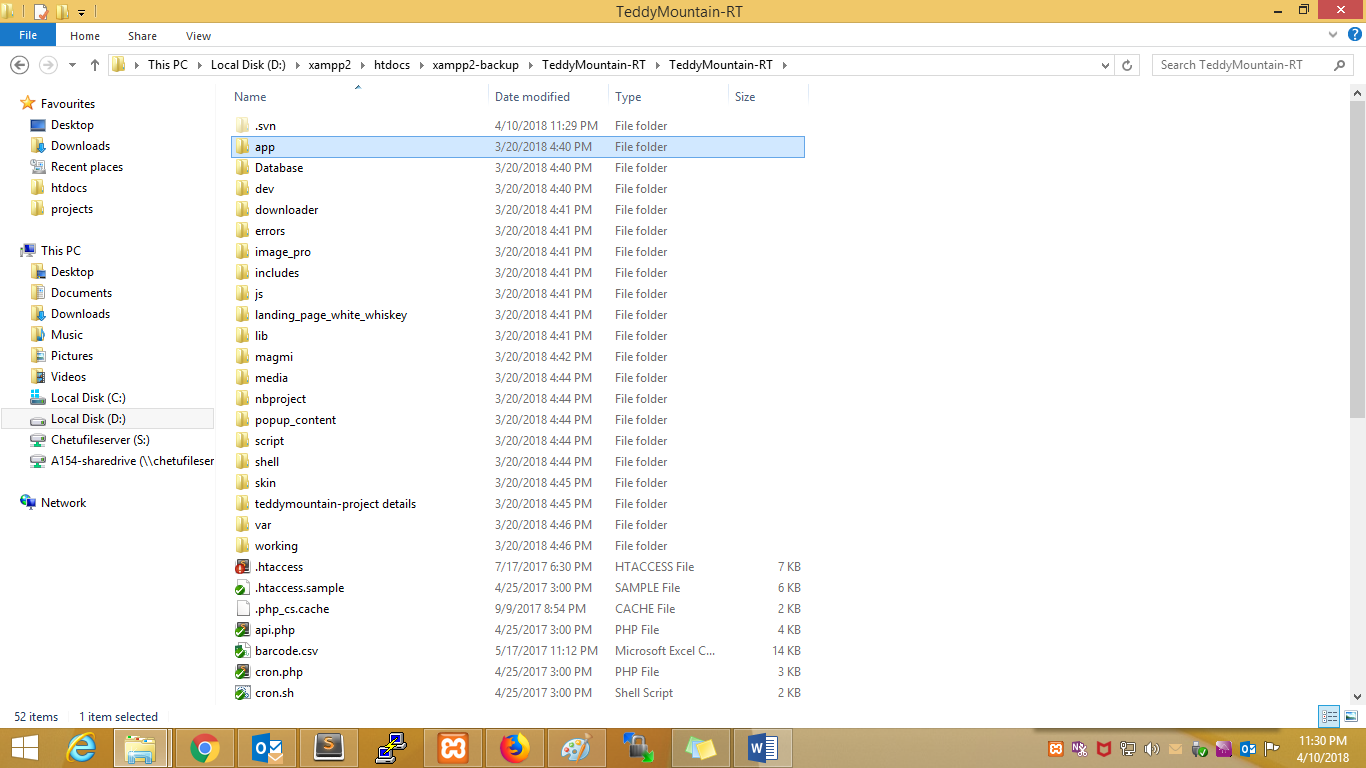


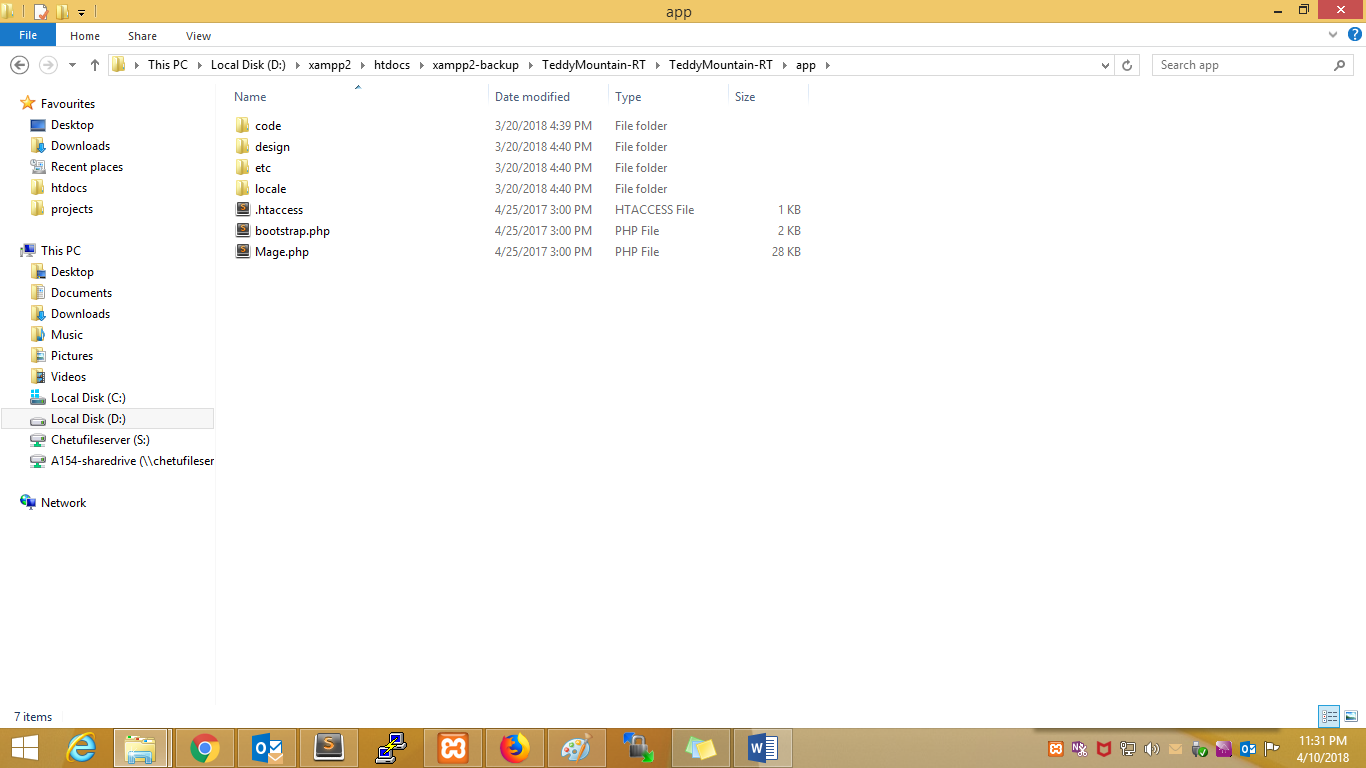
Module example:-



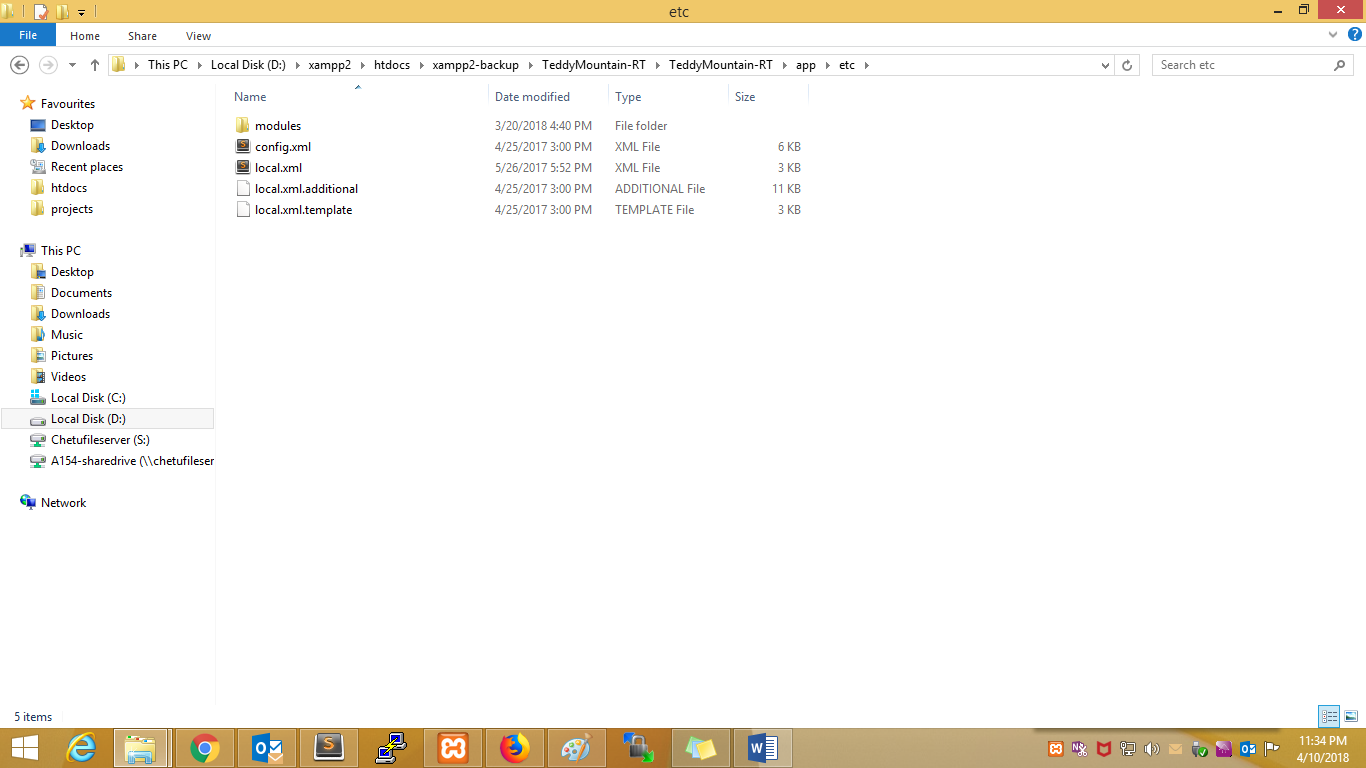
Magento1:-

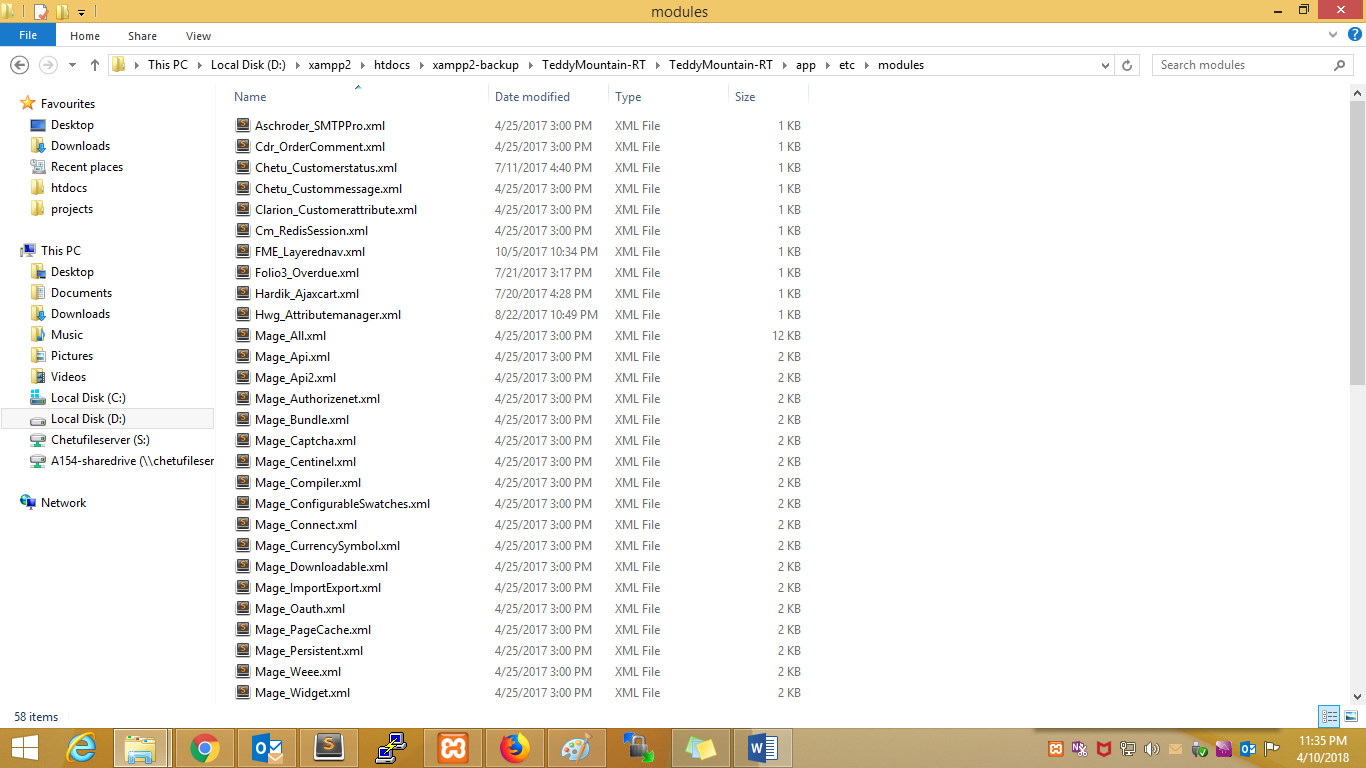
Structure:-



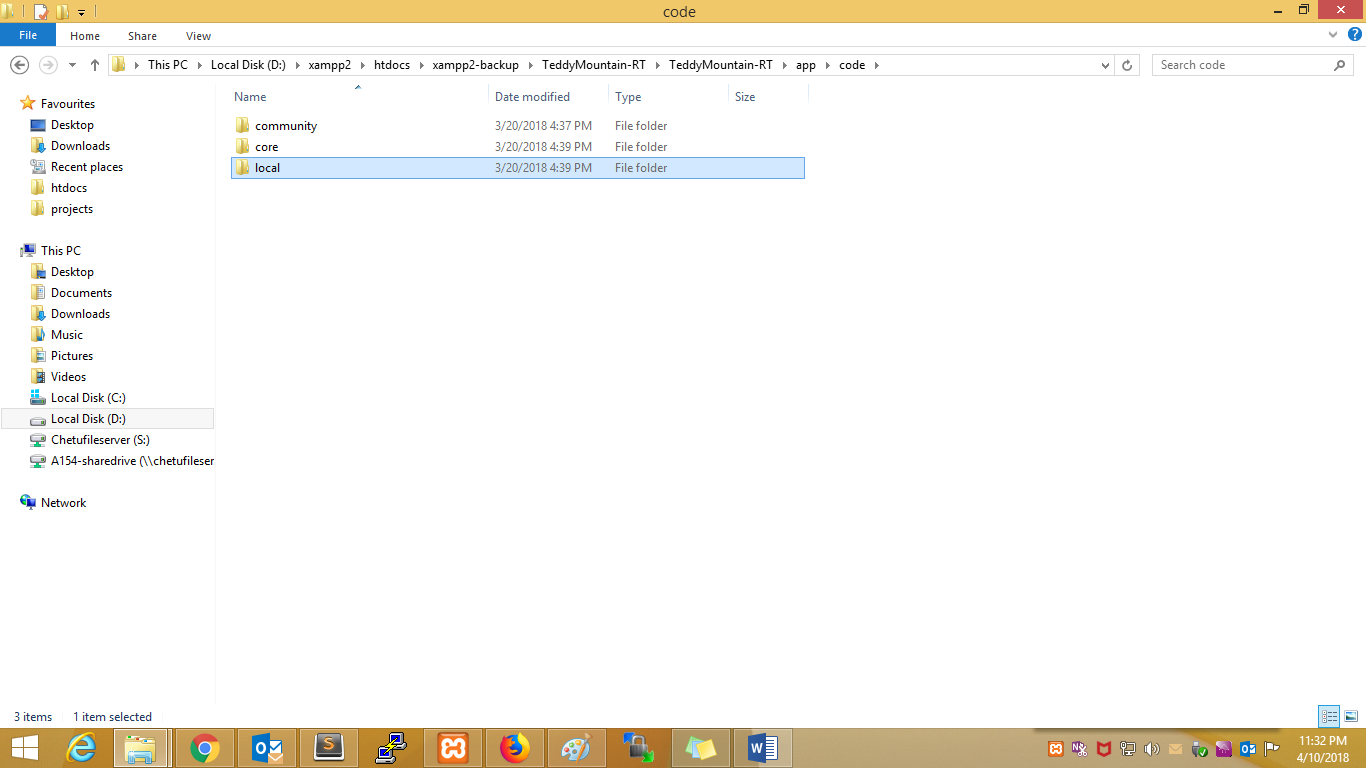
Inside app:-  


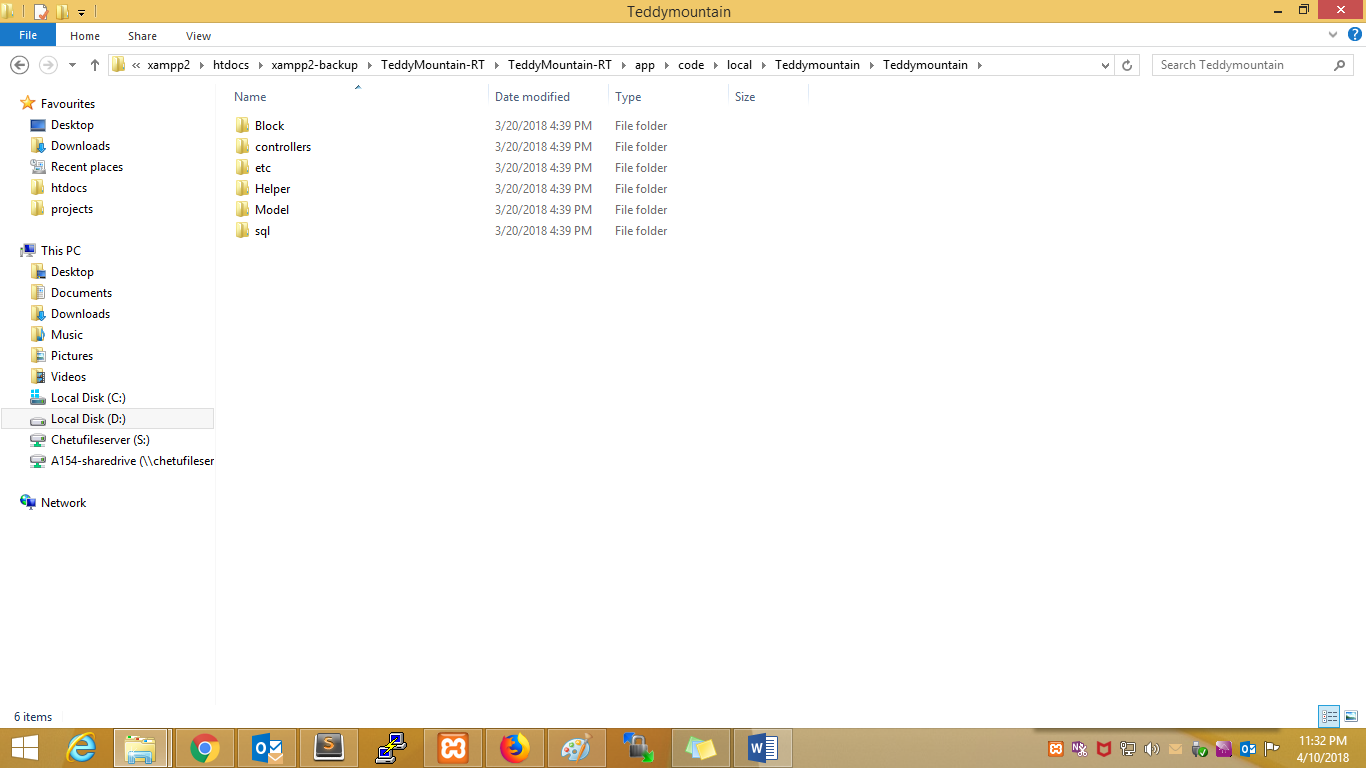
Inside etc:-

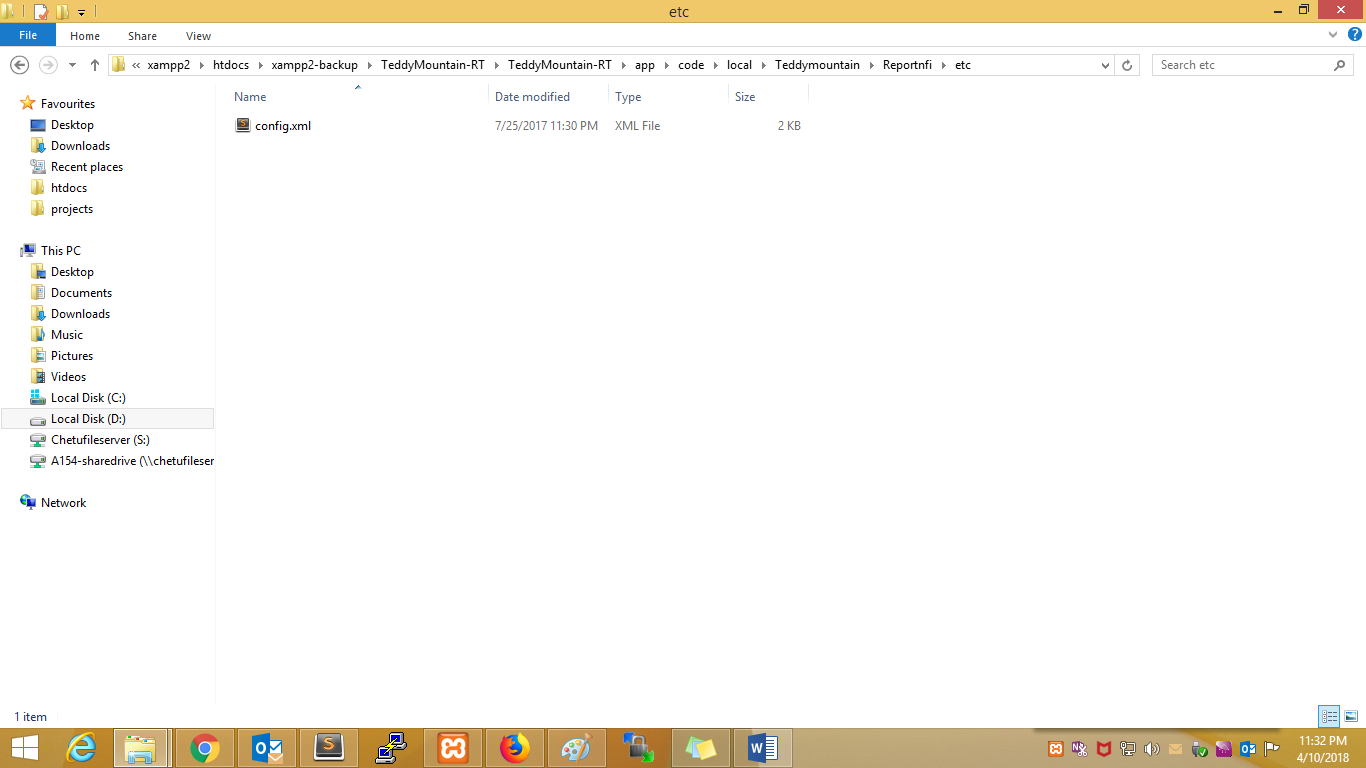


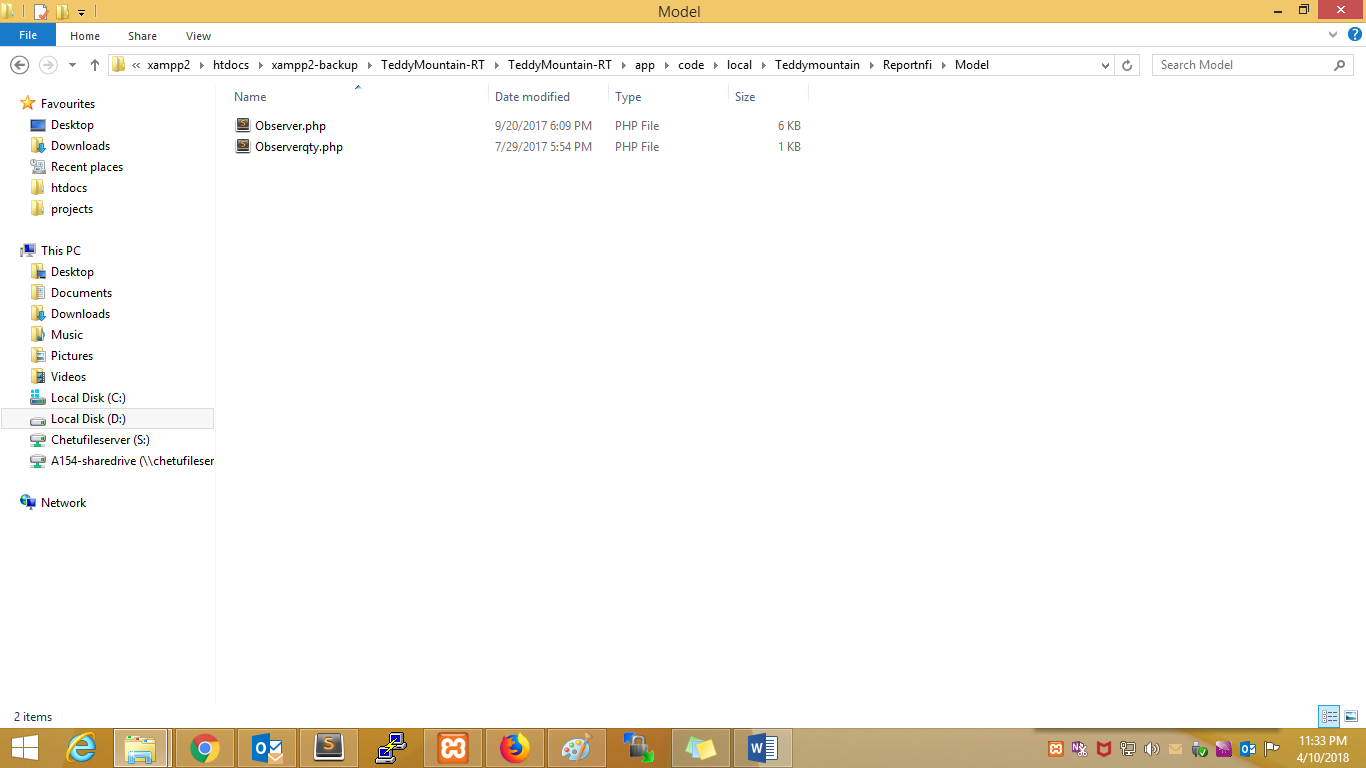


Inside code:-

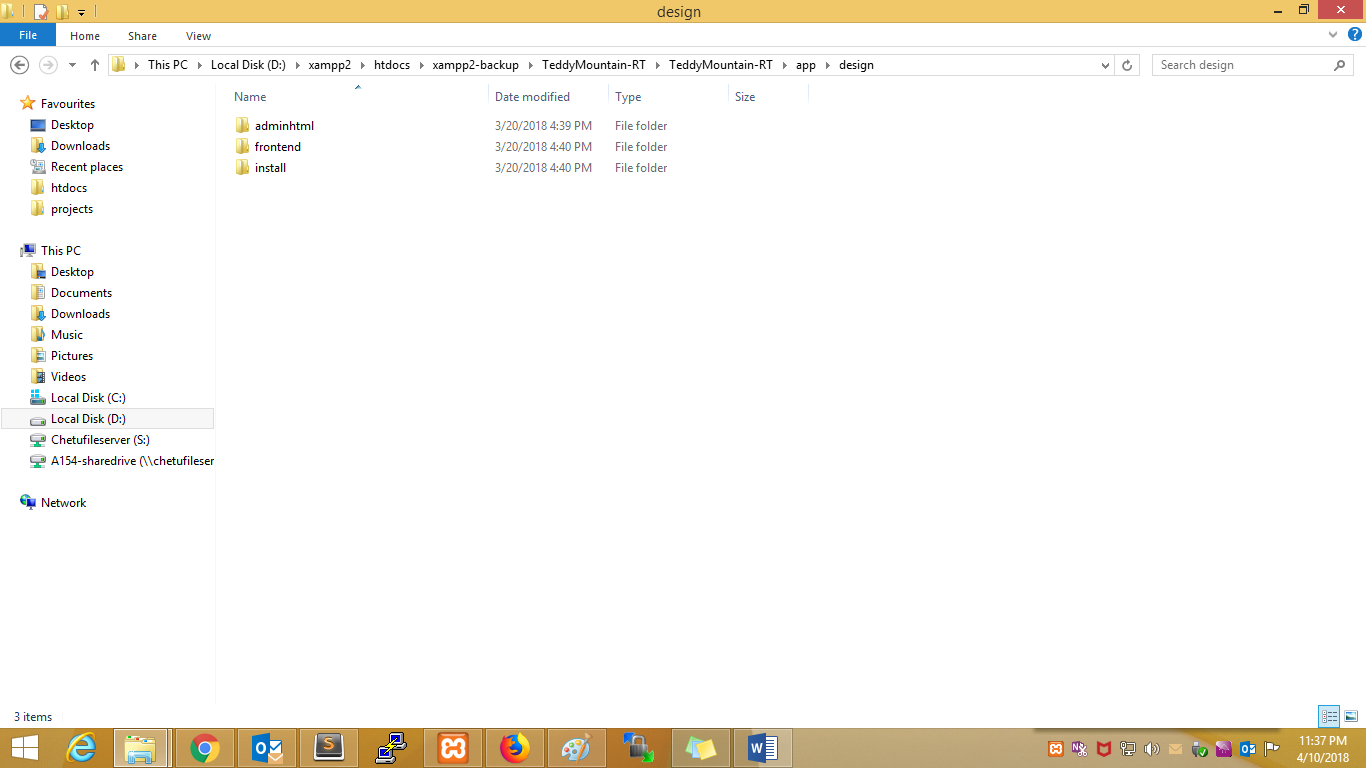




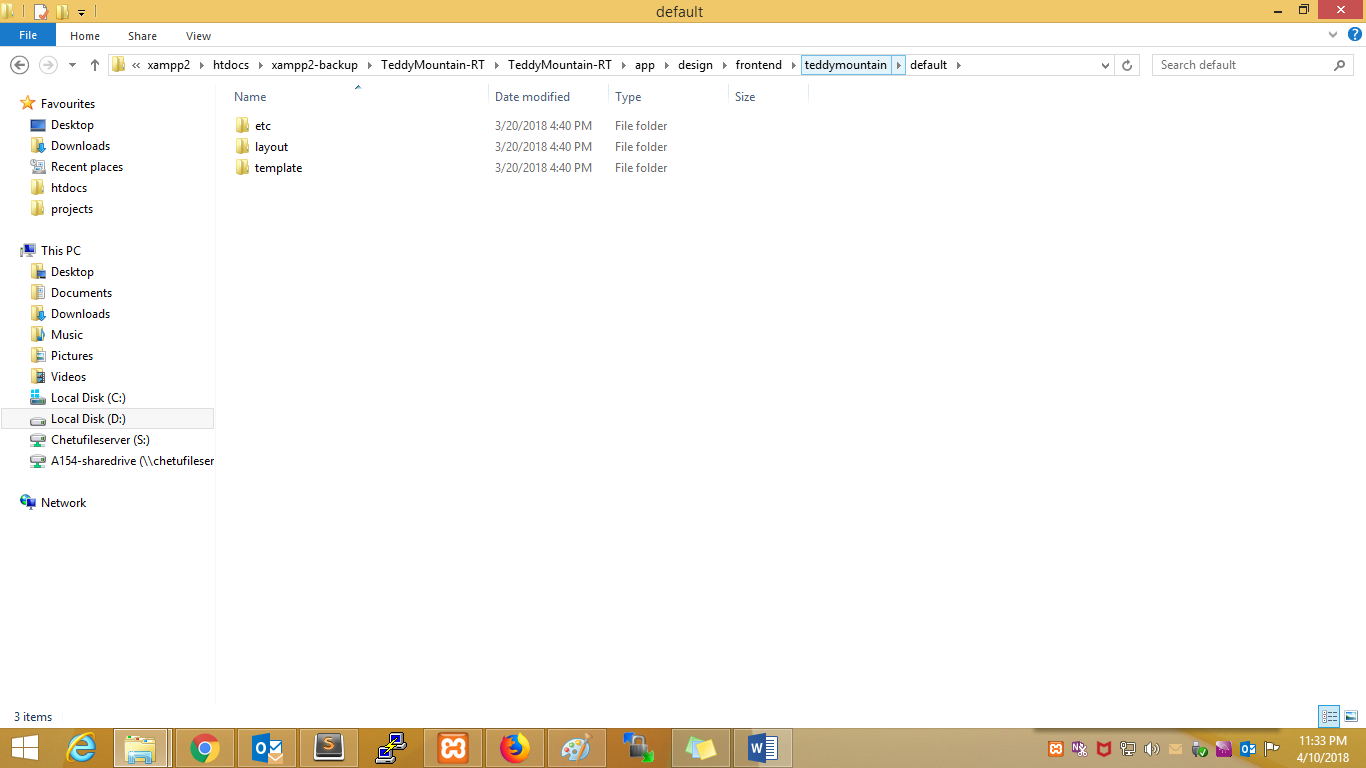




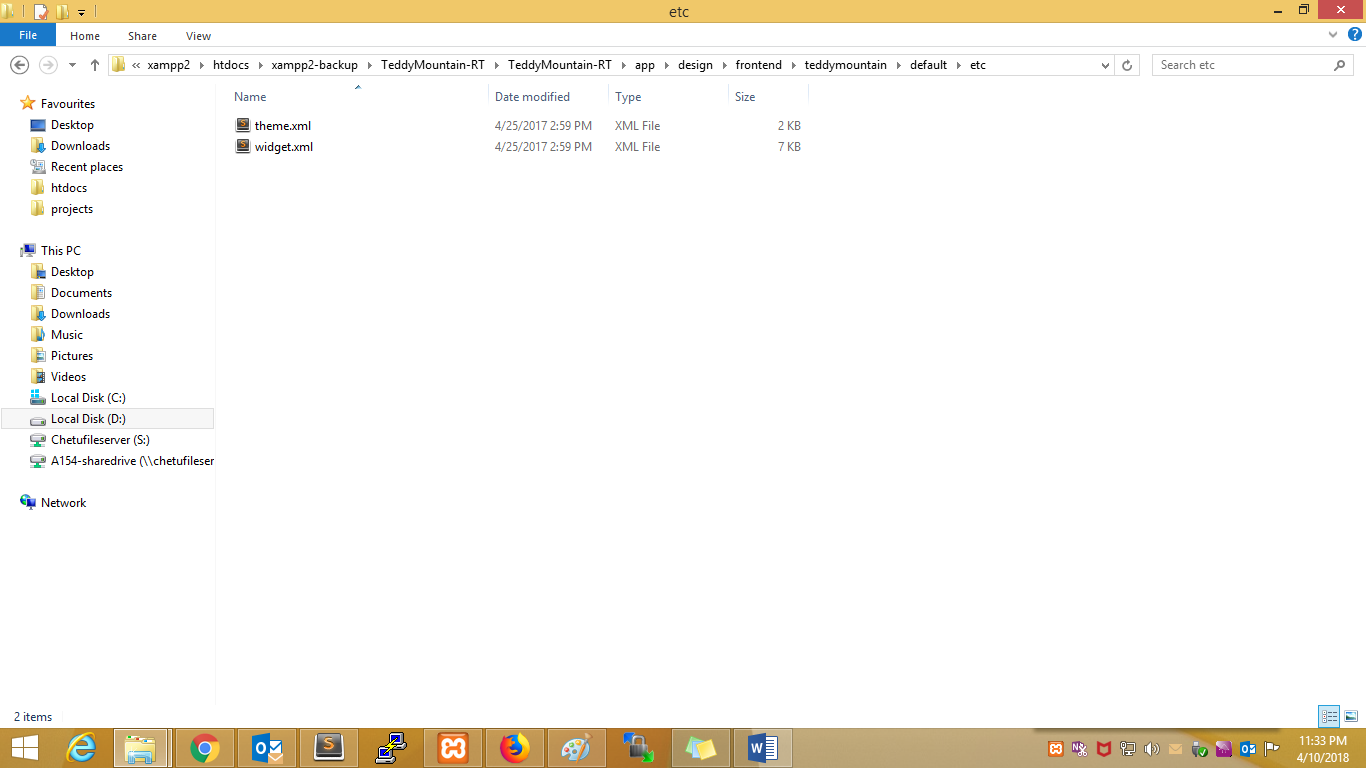
Inside design:-



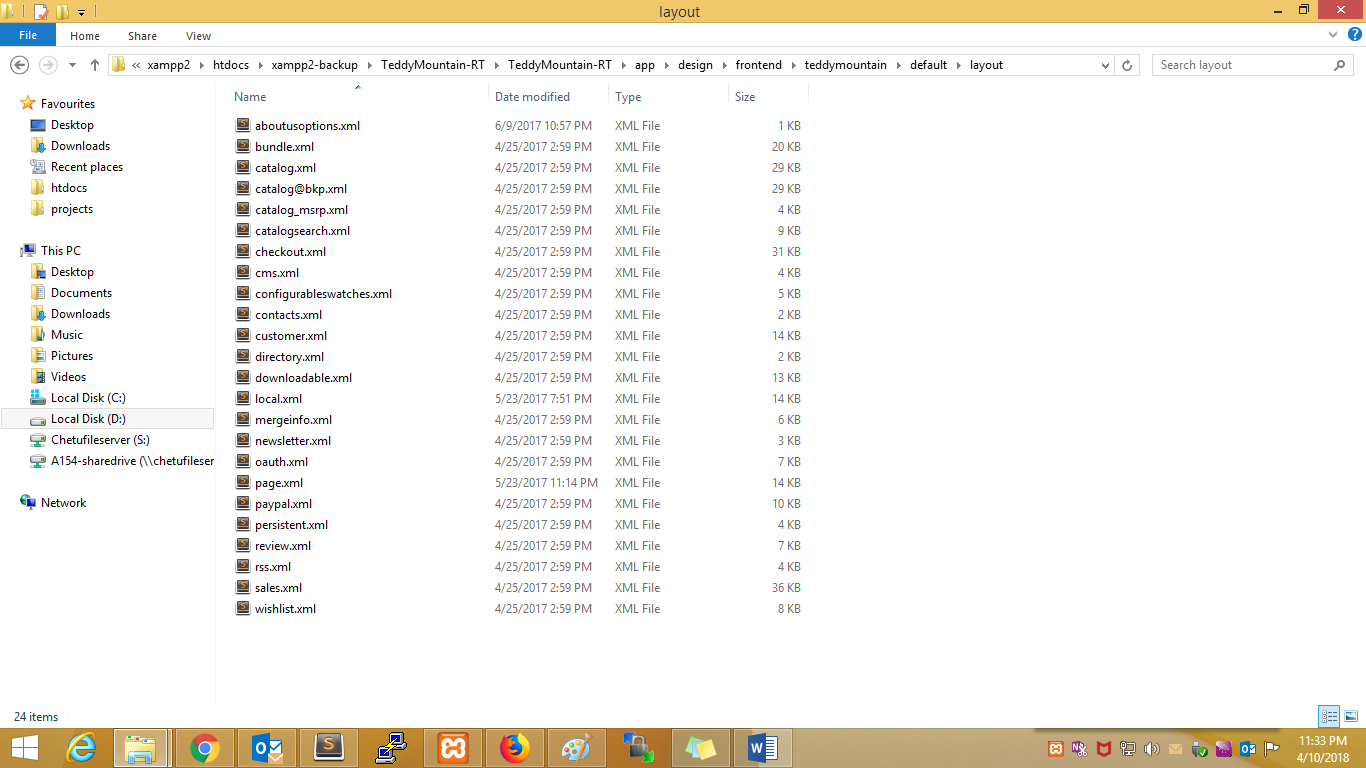
Theme:-



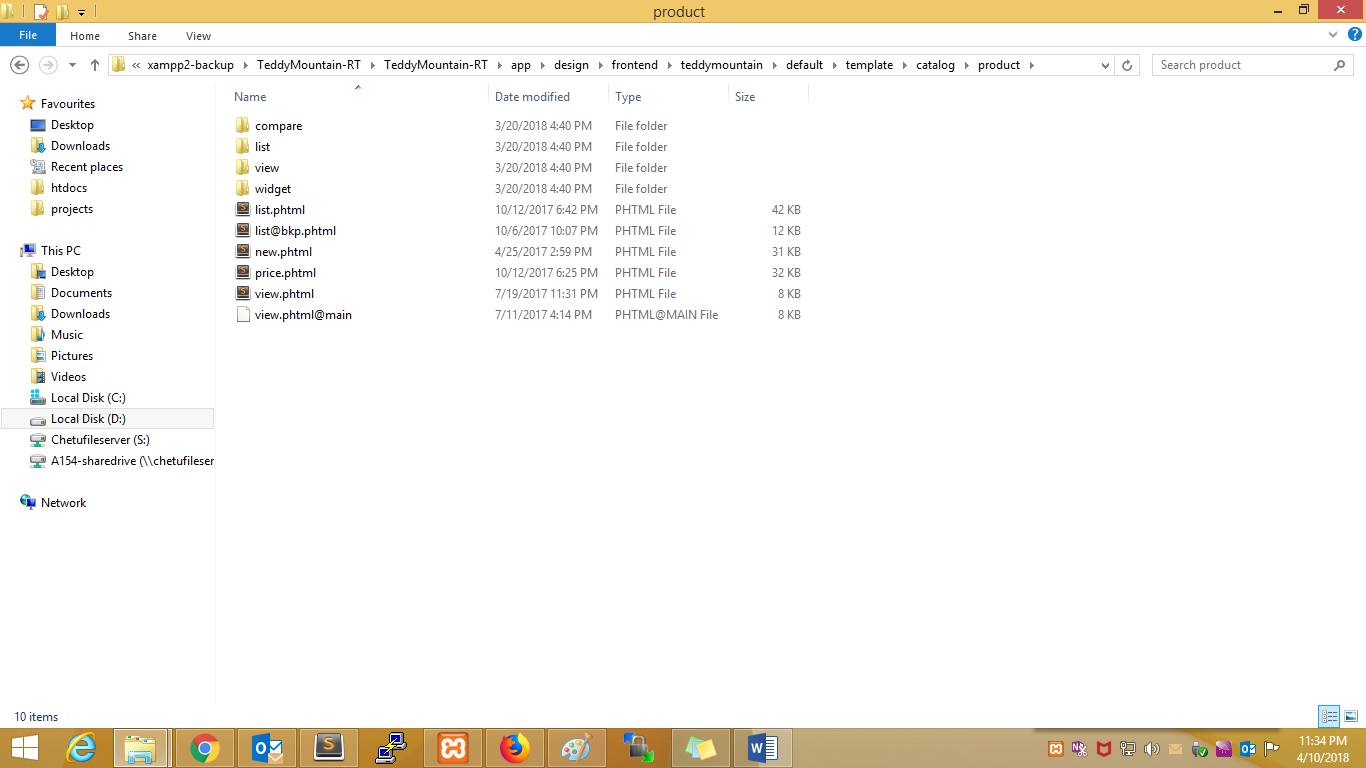
etc:-



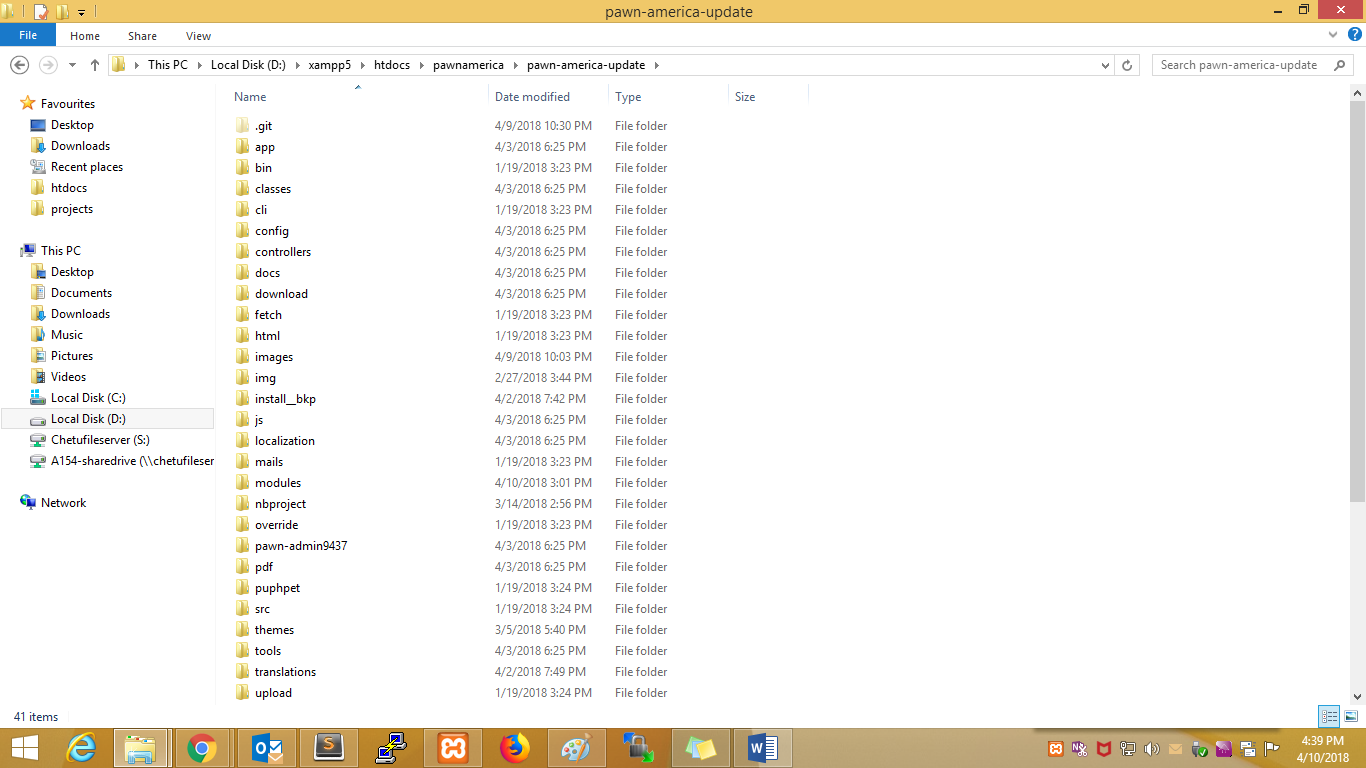
Layout:-



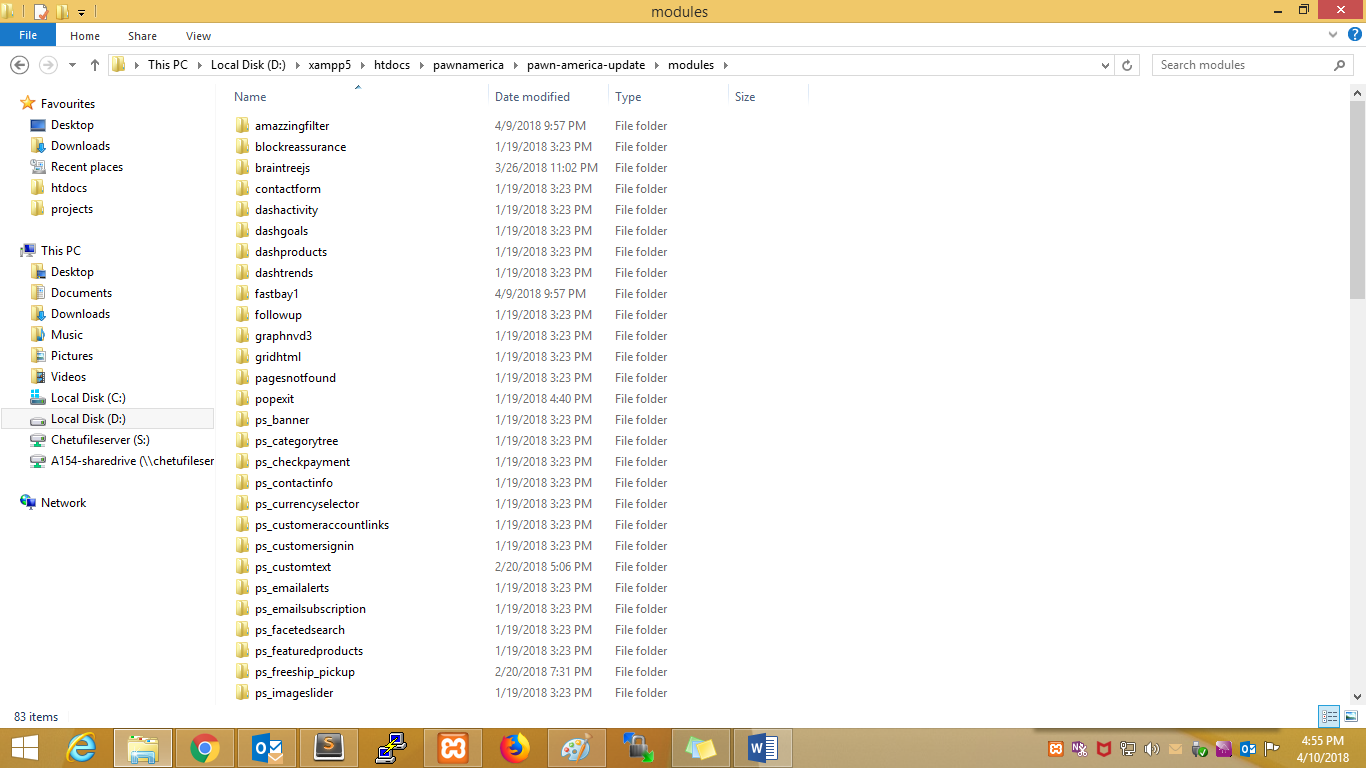
Template:-



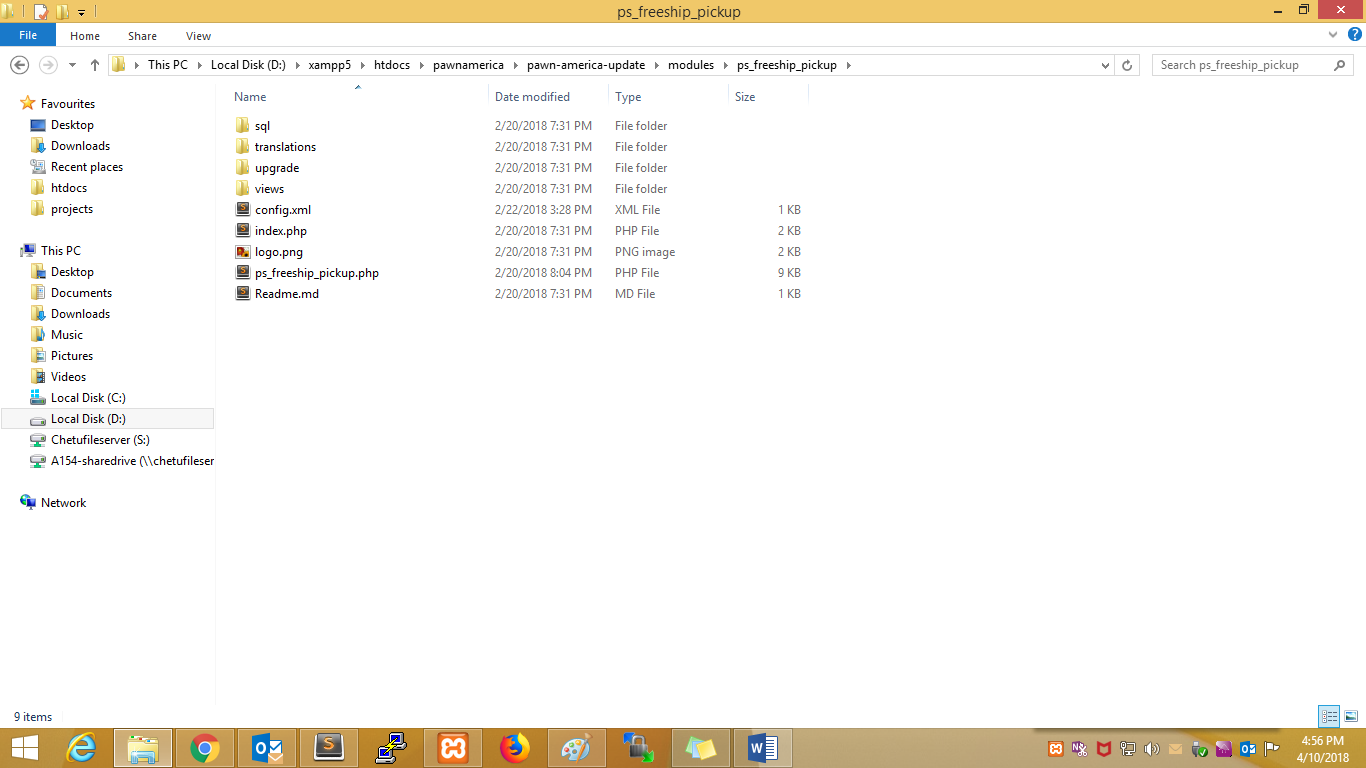
Prestashop:--1.7.0.4:--- root structure:-



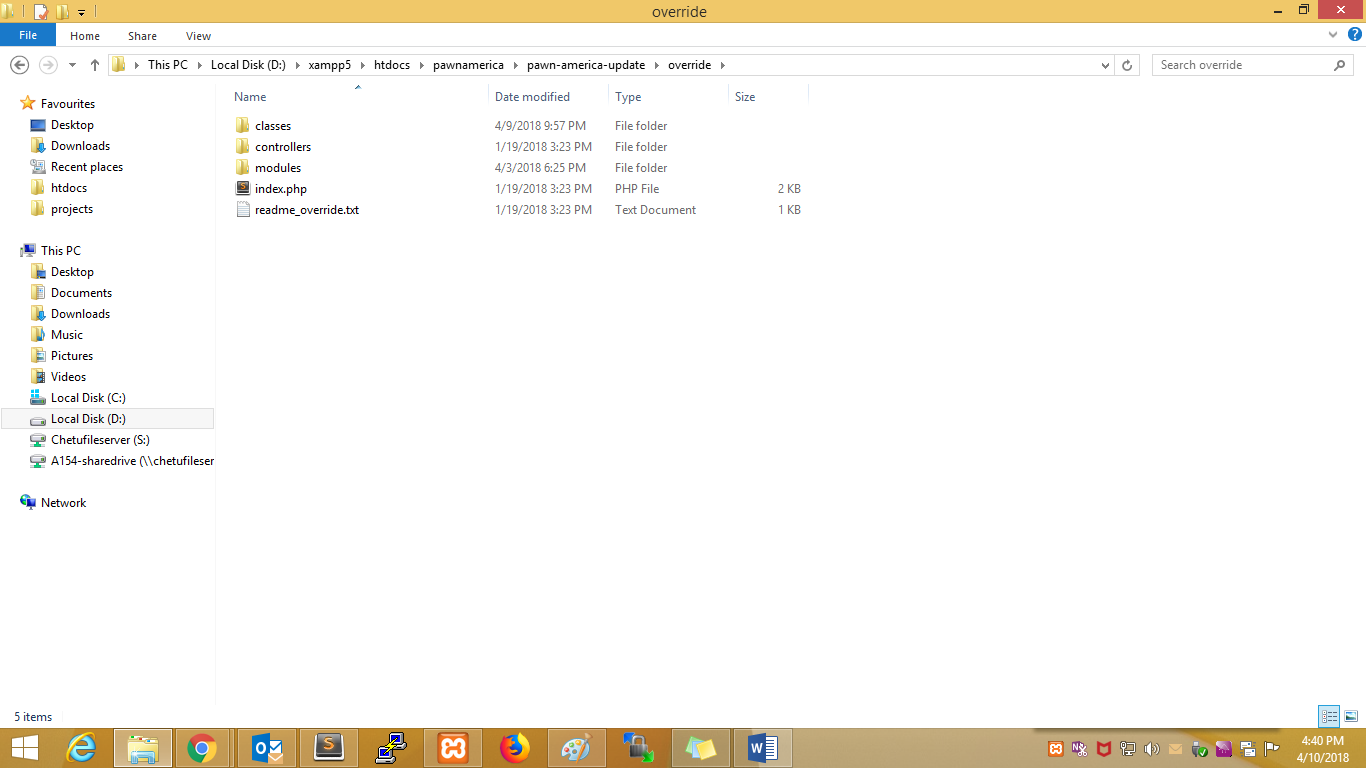
Module folder:-

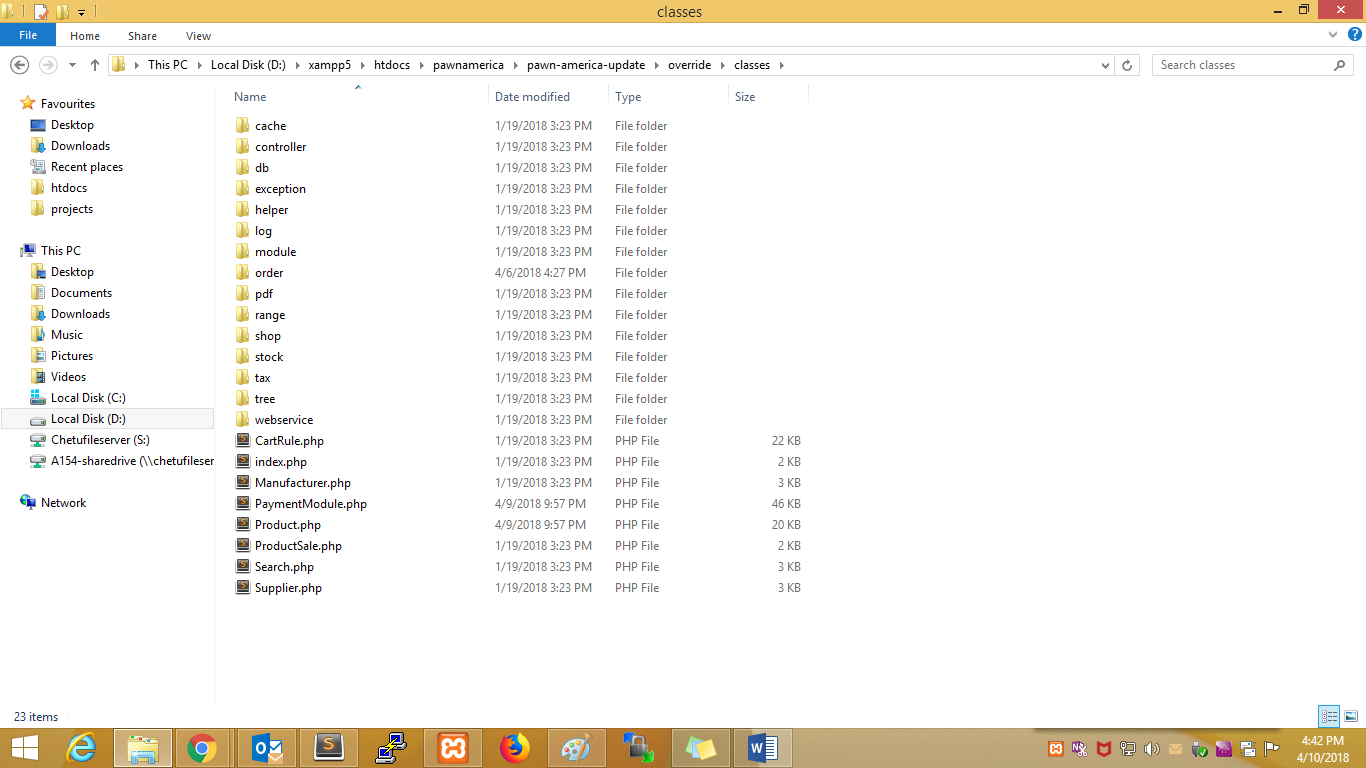


Module structure:-



All classes controller and modules files are overrided in override folder:-





/\*\*

\* overriding Class Product

\*/

class Product extends ProductCore {

/\*\*

\* Author: Chetu Development Team

\* Func Name: getPricesDrop

\* Created On: Jan 12, 2017

\* Created For: Overrided to show discounted product at home page.

\*/

public static function getPricesDrop($id\_lang, $page\_number = 0, $nb\_products = 10, $count = false, $order\_by = null, $order\_way = null, $beginning = false, $ending = false, Context $context = null) {}

}

abstract class PaymentModule extends PaymentModuleCore

{

public function validateOrder($id\_cart, $id\_order\_state, $amount\_paid, $payment\_method = 'Unknown',

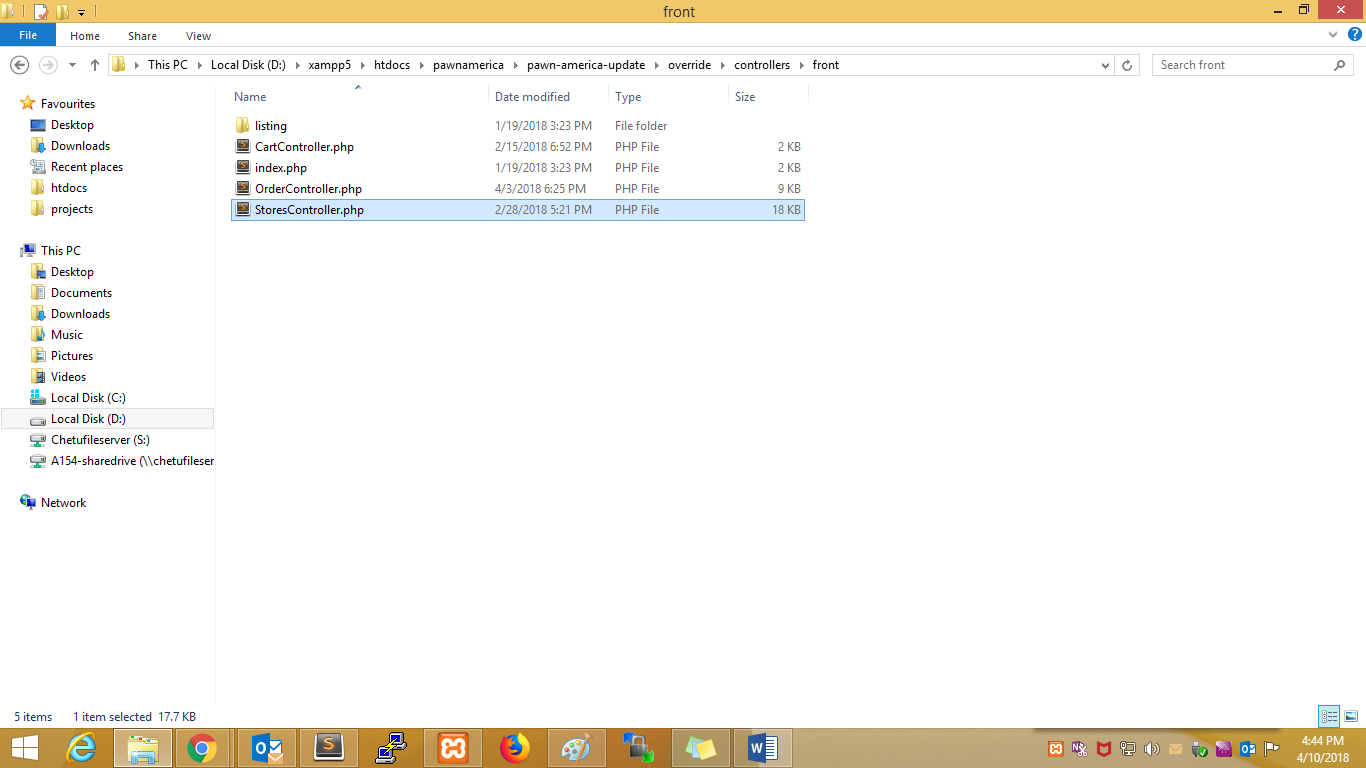
$message = null, $extra\_vars = array(), $currency\_special = null, $dont\_touch\_amount = false,

$secure\_key = false, Shop $shop = null)

{}

}

Controller override:-



/\*\*

\* OrderController

\* overriding OrderController php file

\*/

use PrestaShop\PrestaShop\Core\Foundation\Templating\RenderableProxy;

use PrestaShop\PrestaShop\Adapter\Product\PriceFormatter;

class OrderController extends OrderControllerCore {

public $ssl = true;

public $php\_self = 'order';

public $page\_name = 'checkout';

protected $checkoutProcess;

/\*\*

\* Author: Chetu Team

\* Func Name: init

\* Created On: Feb 14, 2018

\* Created For: init

\*/

public function init() {

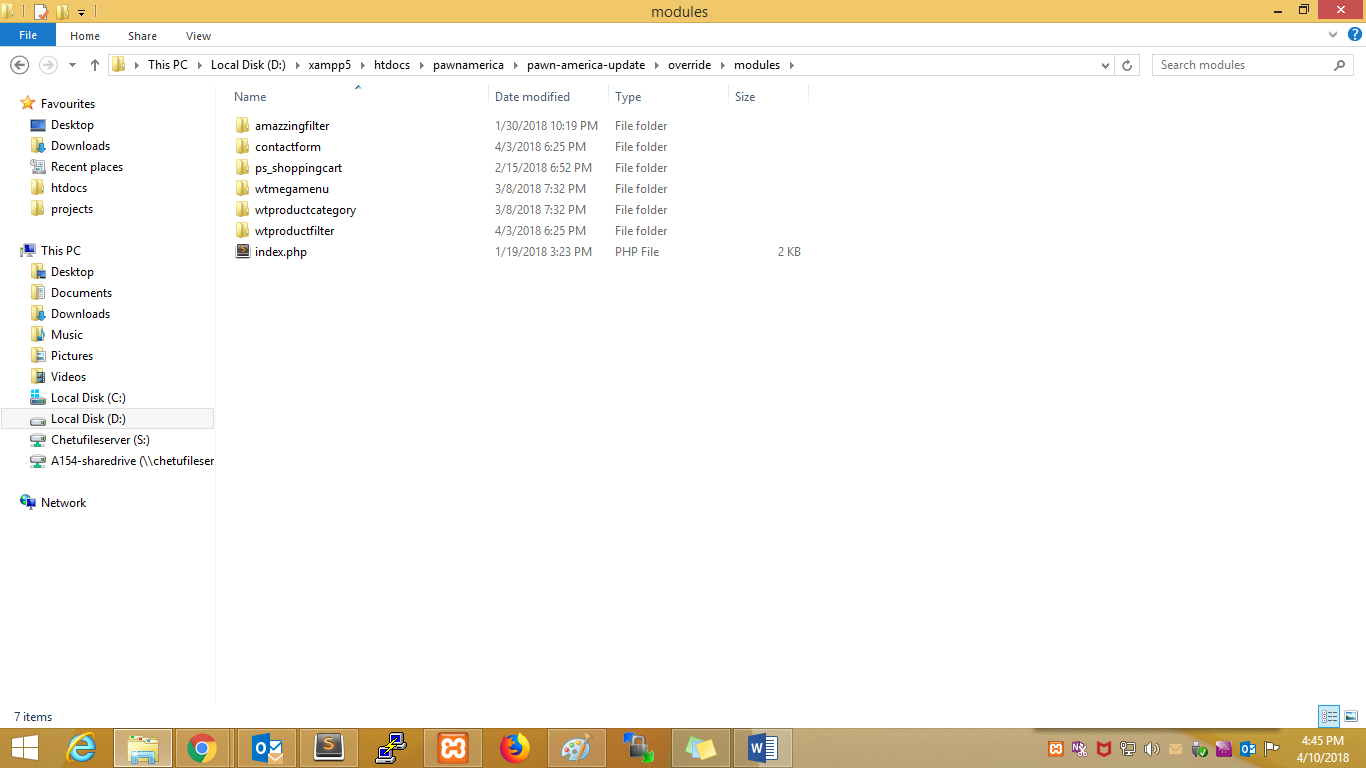
parent::init();

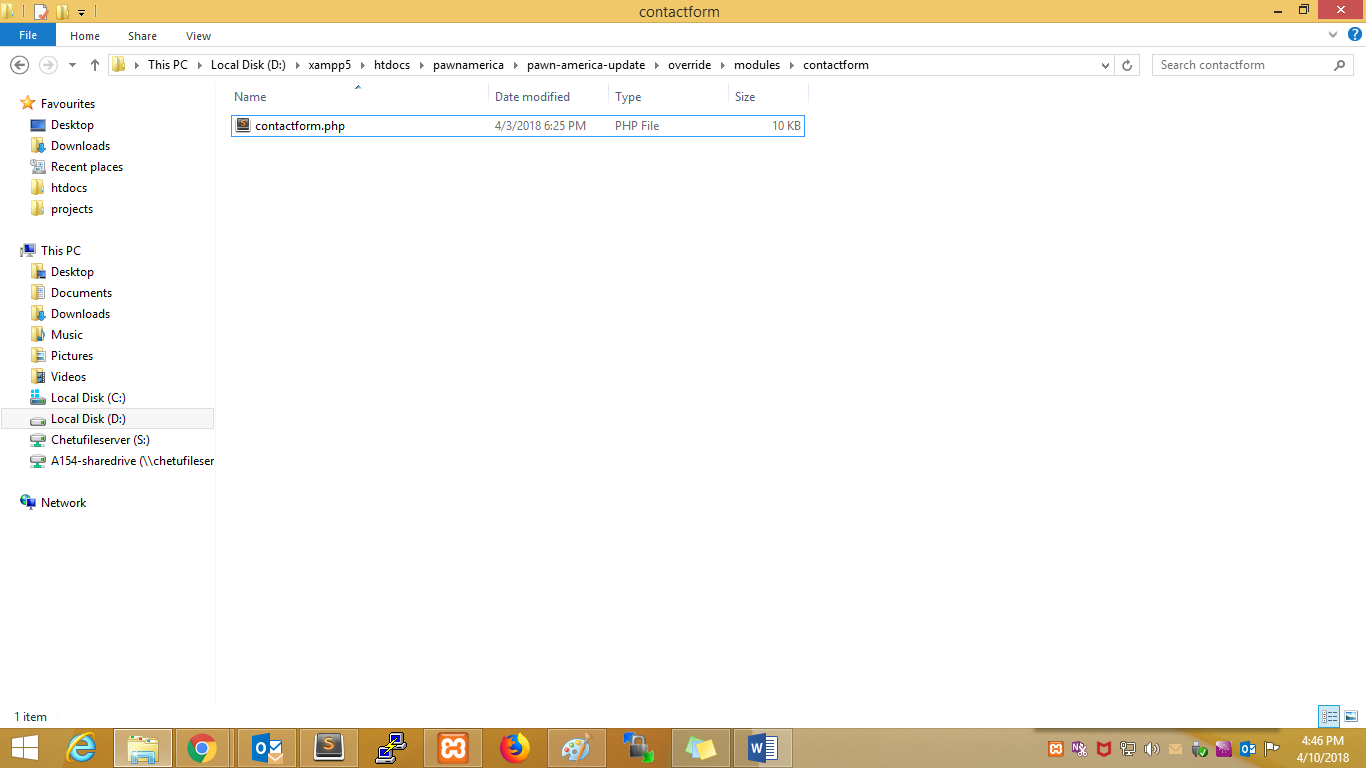
$this->cartChecksum = new CartChecksum(new AddressChecksum());

}

}

Module override:--





/\*\*

\* Contactform

\* overriding Contactform php file

\*/

class ContactformOverride extends Contactform

{

/\*\*

\* Author: Chetu Team

\* Func Name: sendMessage

\* Created On: Mar 15, 2017

\* Created For: Adding validation for captcha

\*/

public function sendMessage()

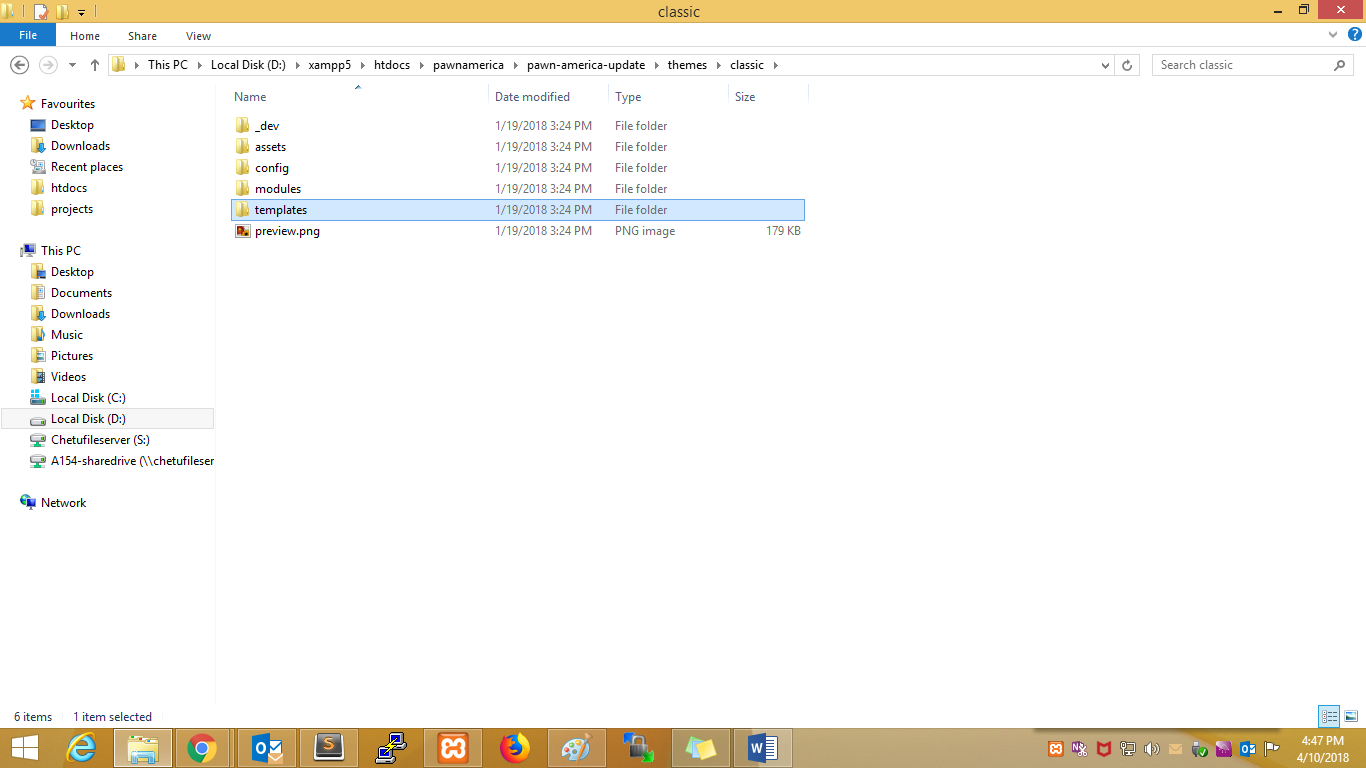
{

}

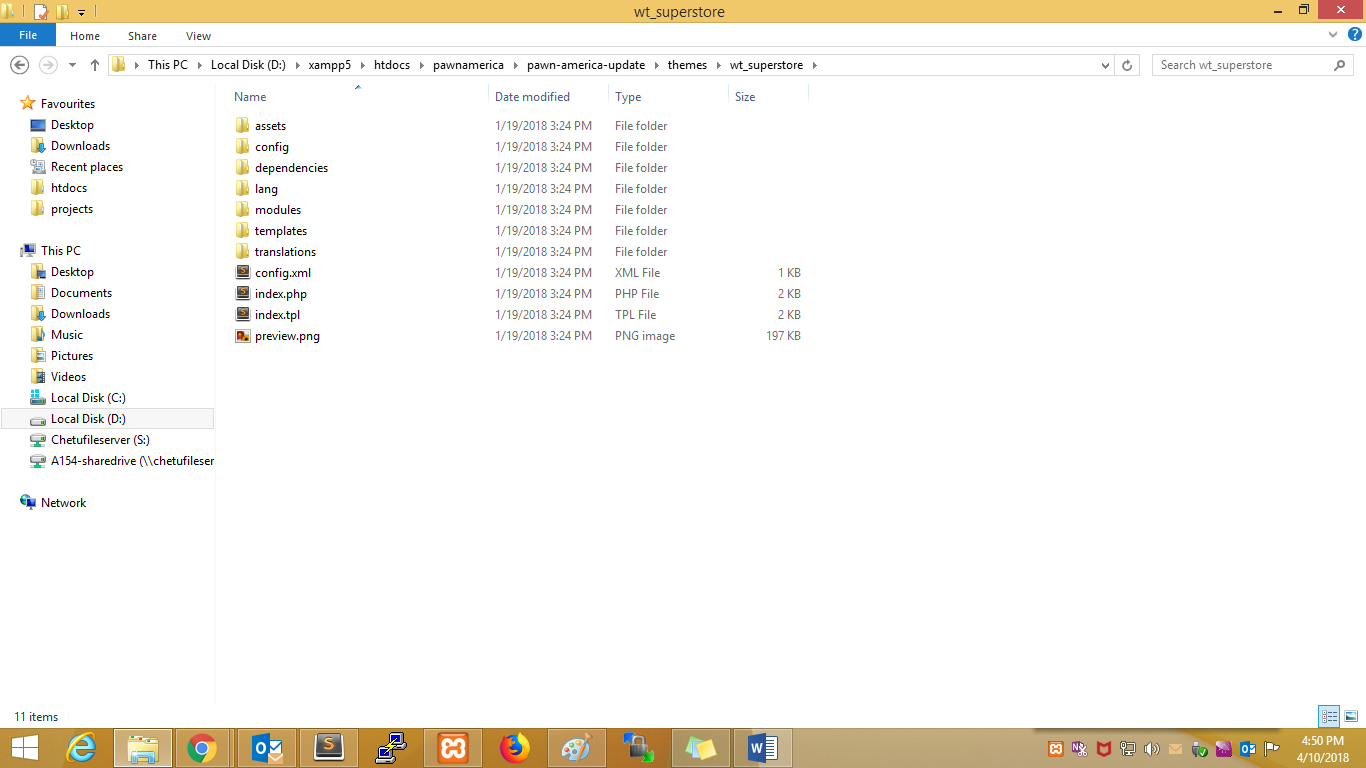
}

New child theme creation :-

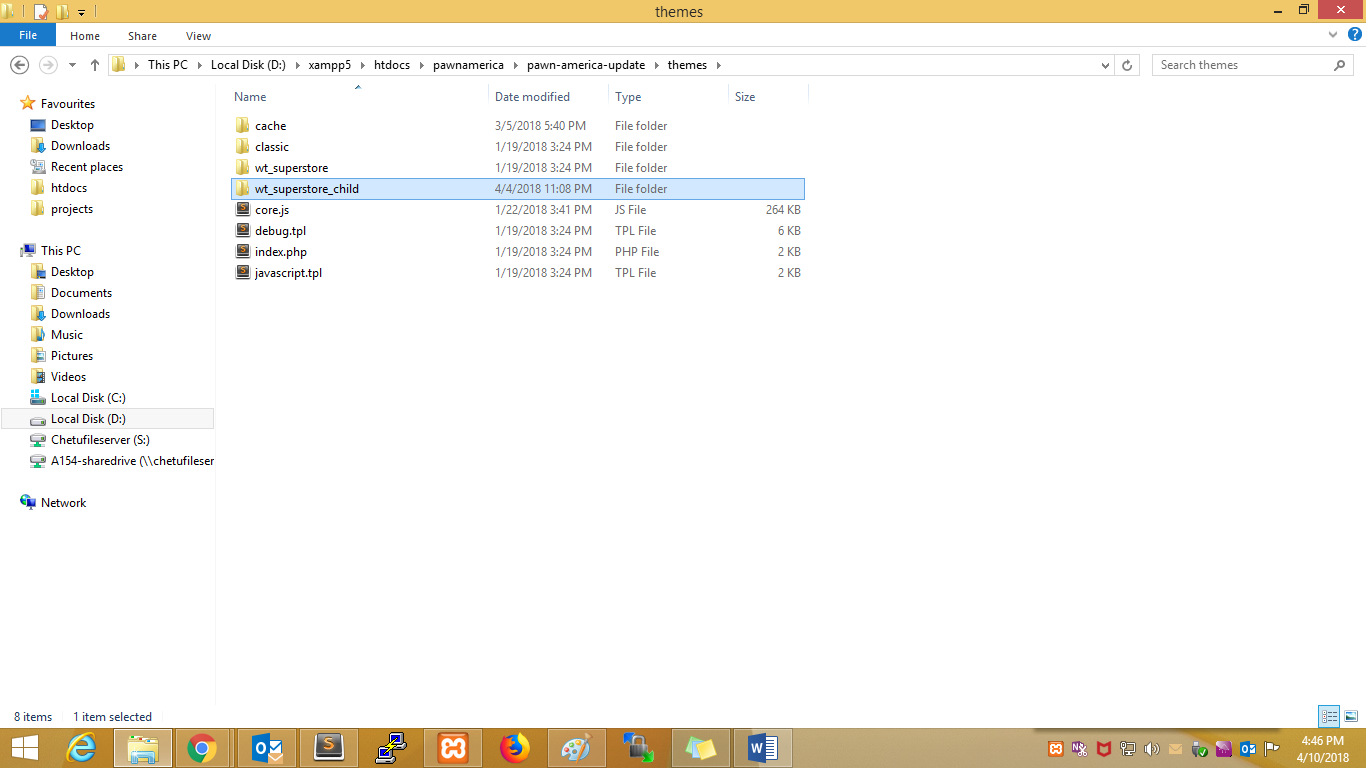
Base theme:-

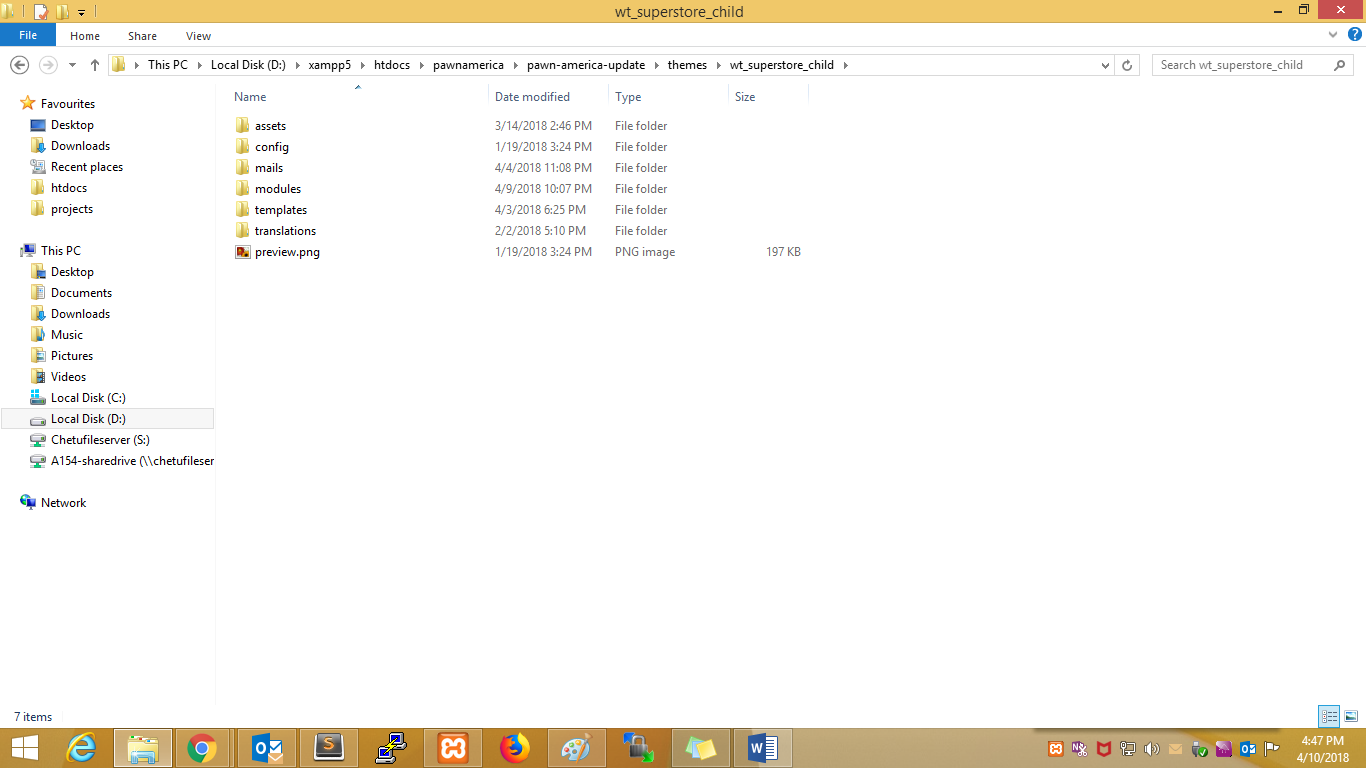


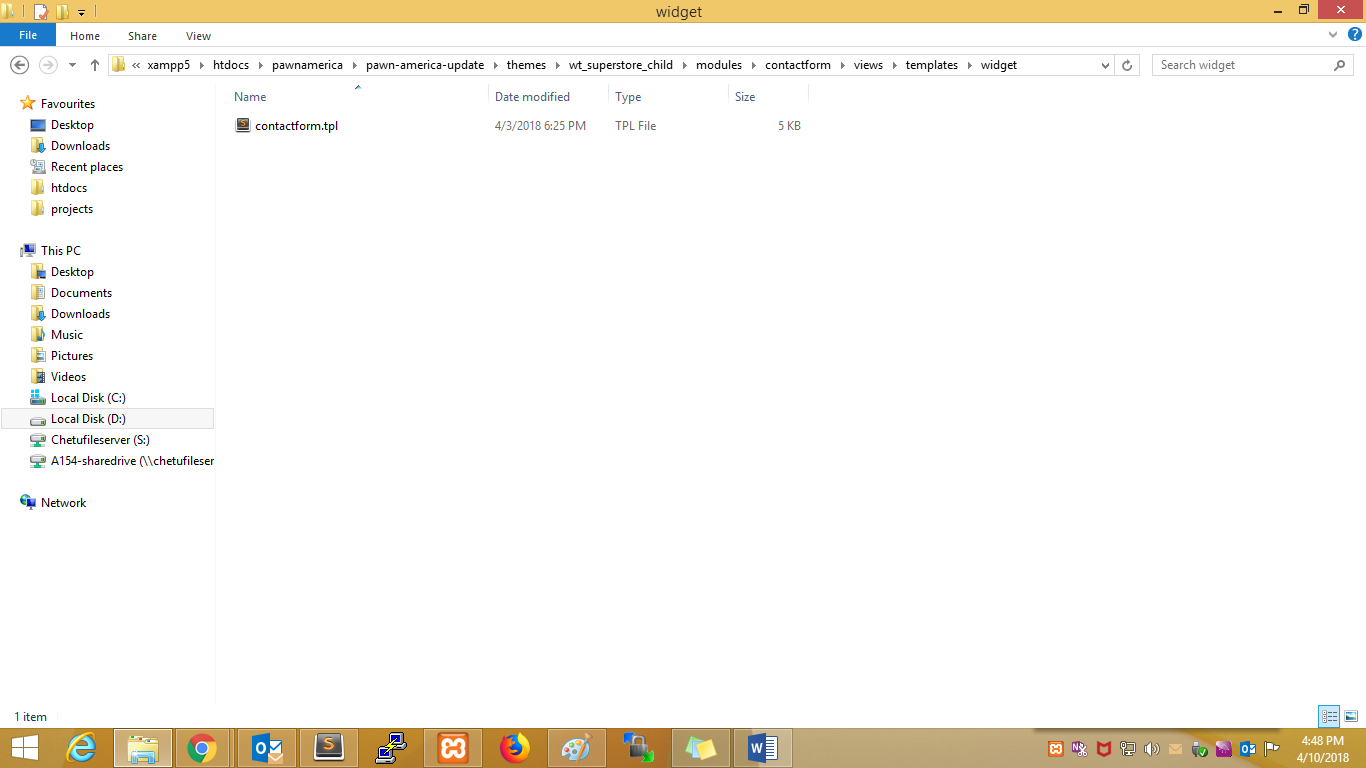
New theme:-

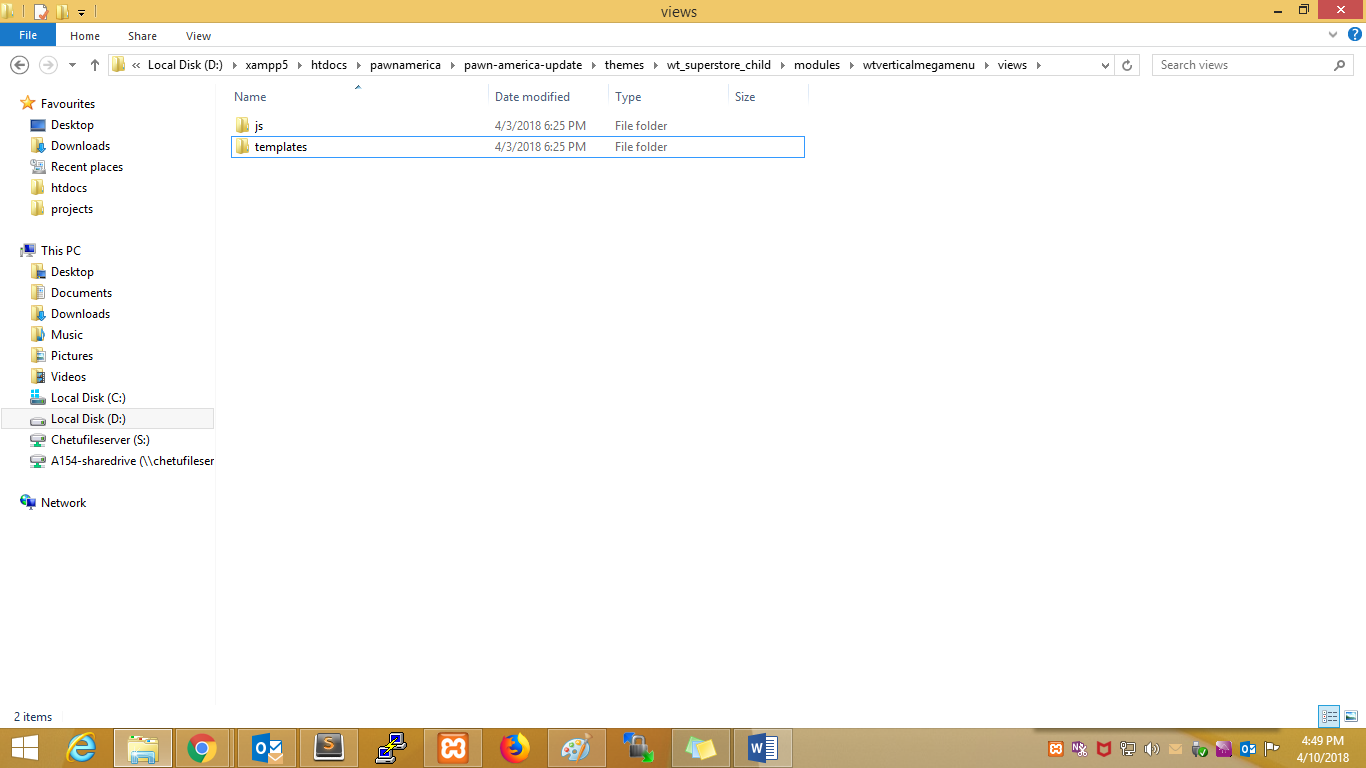


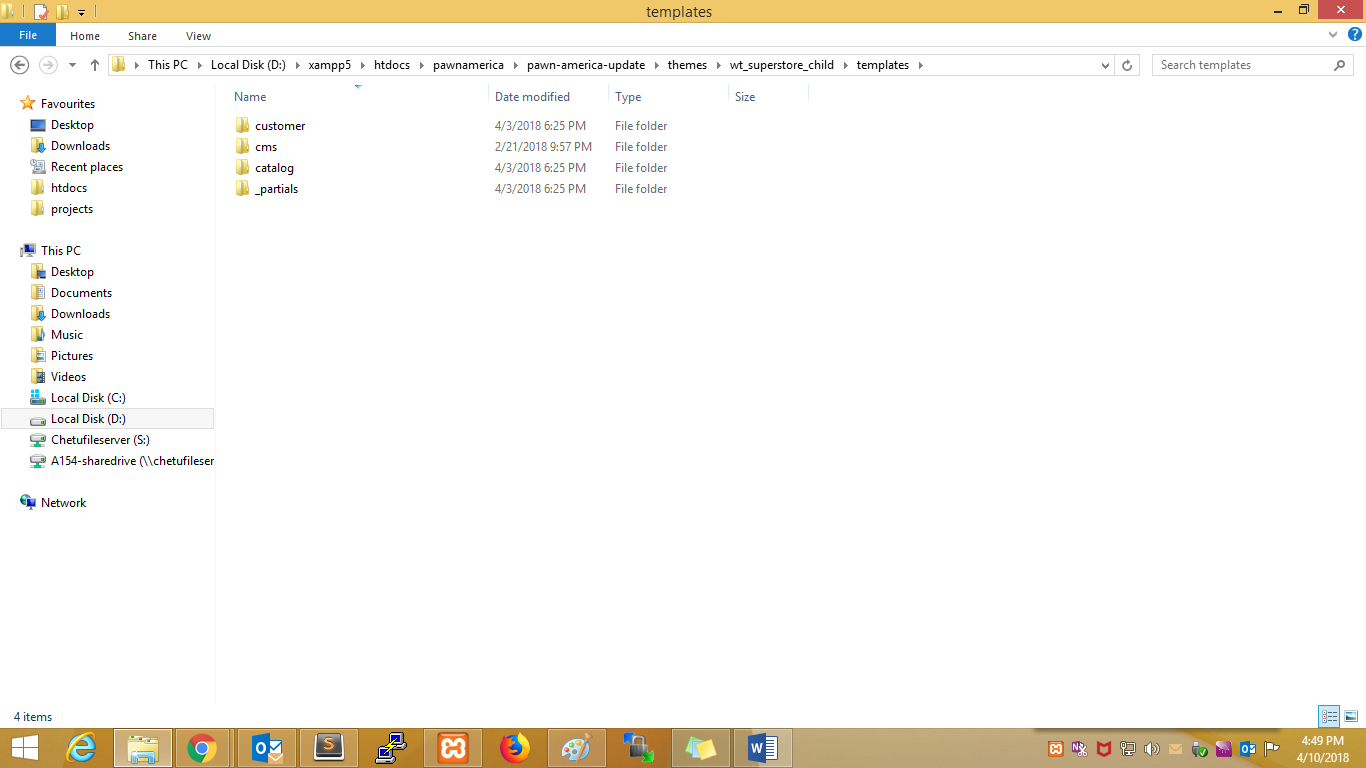
Child theme:--













Wordpress:-

**Q1.         What are hooks and define types of hooks in WordPress?**

**Ans.**Hooks are provided by WordPress to allow your plugin to ‘hook into’ the rest of WordPress; that is, to call functions in your plugin at specific times, and thereby set your plugin in motion. There are two kinds of hooks:

Actions Hooks: Actions hooks are the hooks that the WordPress core launches at specific points during execution, or when specific events occur. Your plugin can specify that one or more of its PHP functions which are executed at these points, using the Action API.

Filters Hooks: Filters hooks are the hooks that WordPress launches to modify text of various types before adding it to the database or sending it to the browser screen. Your plugin can specify that one or more of its PHP functions which are executed to modify specific types of text at these times, using the Filter API.

**Actions Functions**:  
has\_action()  
add\_action()  
do\_action()  
do\_action\_ref\_array()  
did\_action()  
remove\_action()  
remove\_all\_actions()

**Filter Functions**:  
has\_filter()  
add\_filter()  
apply\_filters()  
apply\_filters\_ref\_array()  
current\_filter()  
merge\_filters()  
remove\_filter()  
remove\_all\_filters()

**Q2.         What are the template tags in WordPress?**

**Ans.**A template tag is code that instructs WordPress to “do” or “get” something. Like in header.php  we will use the tag bloginfo(‘name’) to get “Site Title” from wp-options table which is set in Setting > General at WordPress dashboard.

The the\_title() template tag is used to display the post title.

wp\_list\_cats() is  for display categories.

get\_header() for getting header.

get\_sidebar() for display the sidebar on page.

get\_footer() for get the footer content on page.

**Q3.         how to write the short code in WordPress php file?**

**Ans.**Using do\_shortcode() function inside of php echo tag. A very simple solution is to use the do\_shortcode() function inside a PHP echo tag.

|  |  |
| --- | --- |
| 0  1 | <?php do\_shortcode("[shortcode]"); ?> |

Short code is used in WordPress post or page and text box widget and php file.

**Q4.         What are rules to follow in WordPress plugin development?**

**Ans.**

* Find a unique name
* Setup a prefix (related to your brand)
* Create the plugin’s folder
* Create sub-folders for PHP files, assets, and translations
* Create the main plugin file and fill in obligatory header information
* Create a readme.txt file
* Use proper constants and functions to detect paths to plugin files
* Create additional PHP files and include them inside the main one
* Create activation and deactivation functions
* Create an uninstall script

**Q5.         What are the steps you can take if your WordPress file is hacked?**

**Ans.**

* Install security plugin like WP security
* Re-install the latest version of WordPress
* Change password and user-ids for all your users
* Check your themes and plugins are up to date

**Q6.         In which cases you don’t see plugin menu?**

**Ans.**You can’t see your plugin menu when the blog is hosted on free [*wordpress.com*](http://wordpress.com/) as you cannot add plugin there.  Also, if you do not have an account of an administrator level on your WordPress dashboard, it is not possible to see plugin.

**Q7.         What is the difference between the wp\_title and the\_title tags?**

**Ans.**       wp\_title() function is for use outside “The Loop” to display the title of a Page.  the\_title()on the other hand is used within “The Loop“.

**Q8.         How to modify the parent theme behavior within the child theme?**

**Ans.**The child theme could optionally override other template files like author.php, category.php, functions.php, style.css etc. The WordPress framework first looks for a template file in the child theme directory and then will pick it up from the parent directory if not found.

For more details: [http://code.tutsplus.com/articles/how-to-modify-the-parent-theme-behavior-within-the-child-theme–wp-31006](http://code.tutsplus.com/articles/how-to-modify-the-parent-theme-behavior-within-the-child-theme--wp-31006)

**Q9.         How to Create Custom Post Type?**

**Ans.**Custom post types are new post types you can create. A custom post type can be added to WordPress via the [*register\_post\_type()*](http://codex.wordpress.org/Function_Reference/register_post_type) function. This function allows you to define a new post type by its labels, supported features, availability and other specifics.

Here’s a basic example of adding a custom post type:



**Q10.       How to check if a page exists by url?**

**Ans.**You can use get\_page\_by\_path() function. Refer [check page existance by url](http://fellowtuts.com/wordpress/check-if-a-page-exists-by-url/) detailed article.

**Q11.       What do next\_posts\_link() and previous\_posts\_link() do?**

**Ans.**Because post queries are usually sorted in reverse chronological order, next\_posts\_link()usually points to older entries (toward the end of the set) and previous\_posts\_link() usually points to newer entries (toward the beginning of the set).

**Q12.       How to create mailchimp or vertical response campaign for newsletter subscribers & link with WordPress ?**

**Ans.**First Create List & Campaign on mailchimp/ WordPress account . Then subscribe users from WordPress in mailchimp list by plugin or  manual hard code webform.

**Q13.       What options are there to implement a multi language site ?**

**Ans.**WordPress has no bi/multi-language feature built in by default. The other road would be to extend the site with Add-ons like plugins and themes that have multi-language features built-in.

**Q14.       How will you retrieve adjacent posts (next/previous) within the same category?**

**Ans.**We can retrieve previous post using [get\_adjacent\_post()](http://codex.wordpress.org/Function_Reference/get_adjacent_post) function and providing it ‘true‘ and taxonomy name in first and last parameters respectively. Setting third parameter to false will bring next adjacent post in front of you.

**Q15.       How to create Custom Taxonomy specific to a Custom Post type?**

**Ans.**First of all if you want to show taxonomy metabox only to custom post type then register the taxonomy to only that custom post type by passing the custom post type name as argument in register\_taxonomy(). By doing this the taxonomy metabox appears only to custom post type.

For more details: [http://WordPress.stackexchange.com/questions/57493/custom-taxonomy-specific-to-a-custom-post-type](http://wordpress.stackexchange.com/questions/57493/custom-taxonomy-specific-to-a-custom-post-type)

**Q16.       How can I stop WordPress from prompting me to enter FTP information when doing updates?**

**Ans.**If you edit your wp-config.php file you can preload these FTP settings as constants read by WordPress. Keep in mind, on a shared host, you should be mindful of possible security implications.

**Q17.       How To Activate Plugins via Code?**

**Ans.**Plugins are stored in an array in the ‘active\_plugins‘ option. The array contains the file path to each plugin that is active. To activate a plugin you need to determine what it’s path will be, then pass that path to activate\_plugin($plugin\_path).

Before you can call activate\_plugin() you need to include the plugin.php file from wp-admin/includes/. You should also check to make sure your plugin isn’t already active. The result looks something like this:



**Q18.       What’s the difference between site\_url() and home\_url()?**

**Ans.**The site\_url() will always be the location where you can reach the site by tacking on /wp-admin on the end, while home\_url() would not reliably be this location. The home\_url() would be where you have set your homepage by setting General > Settings “Site Address (URL)” field.

**Q19.       How to Link External jQuery/Javascript files with WordPress**

**Ans.**Add your own scripts via wp\_enqueue\_script() in your template’s functions.php, appropriately setting dependences on jQuery, and wp\_head() will add the scripts for you.

**Q20.       How to set global variable in functions.php**

**Ans.**Pure PHP question, really.

|  |  |
| --- | --- |
| 0  1  2  3 | global $urlSmall;    $urlSmall = $thumbSmall['0']; |

If you declare the variable with the ‘global‘ keyword when you initialize it, it will be available thereafter.

**Q21.       How to retrieve an image attachment’s alt text?**

**Ans.**The wp\_get\_attachment\_image() function which will return an HTML string containing these attributes:

* ‘src‘
* ‘class‘
* ‘alt‘
* ‘title‘.