How to create Laravel In admin Panel Steps

Important Link How to create Project admin :

https://www.youtube.com/watch?v=ovLX74sAKqw&list=PLLUtELdNs2ZaHaFmydqjcQ-YyeQ19Cd6u&index=2

https://laravel.com/docs/7.x/authentication

https://www.youtube.com/watch?v=saZRMJLYmmY&list=PLLUtELdNs2ZaHaFmydqjcQ-YyeQ19Cd6u&index=1

https://docs.google.com/document/d/1MLN7LuLU7sRx3WC6xXk-HYECniBKqjN52xXvvl69M6M/edit

****

Steps Create Laravel Project with commands in 2020 Laravel 6.0 :

****

Lecture 1 :

Welcome to Laravel 6 Advance E-com Series. In this series, we will work on Laravel Advance features like Multi Auth, Guards, Vue scaffolding, Migrations, Seeding etc. etc.

Laravel is one of the most popular PHP framework to develop websites. And recently, Laravel 6 version has been launched that we will use for our Advance E-com website.

In this video, we will install Laravel 6 and then will install Authentication for creating default login/register.

1) Install Laravel 6.0 :-
Follow our another tutorial to install Laravel 6.0 :

https://youtu.be/tsDWyuwpOB4 (for Windows)
https://youtu.be/VxqNkYfiSik (for Mac)

2) Install Authentication / Scaffolding :-
After installing the Laravel 6.0 successfully, now we require to install Authentication / Scaffolding that helps to make default login/register panel in our new Laravel website.
Scaffolding refers to quickly set up skeleton for the website that covers the Authentication of the website.

Follow below video to add Authentication :-
https://youtu.be/GWI5-VLCiJA

Now you can check in video, Authentication has been installed and default Login/Register panel is ready in Laravel 6 E-commerce website.

****

Lecture 2 :

In Part 2, we will install one of the best Admin template for our Advance E-commerce Series.

In the 2nd Part of Advance E-com series, we will download one of the best free responsive admin template i.e. AdminLTE 3 and merge it into our Laravel 6 E-com website.

1) Download Template :-
Open website https://adminlte.io to download the latest version of AdminLTE 3.

2) Merge Template :-
Now we will start merging AdminLTE template files into our Laravel website.

We will copy CSS, JS, Images and Plugins from AdminLTE template to our Laravel file structure.

2.1) Copy CSS Files :-
We will create admin_css folder under /public/css and copy all CSS files from dist/css/ folder from AdminLTE into it.

2.2) Copy JS Files :-
We will create admin_js folder under /public/js and copy all JS files from dist/js/ AdminLTE into it.

2.3) Copy Images :-
We will create admin_images folder under /public/images and copy all Images files from dist/img/ AdminLTE into it.

2.4) Plugins Folder :-
We will copy plugins folder from AdminLTE folder and paste into public folder in our Laravel website.

3) Create admin_layout Folder and admin_layout.blade.php File :-
We will create admin_layout Folder under resources/views/layouts/ path in which we will create admin_layout.blade.php file.

We will copy content from index.html file from AdminLTE folder and paste it into this admin_layout.blade.php file. Before making further changes in this file, we will also create below files under admin_layout Folder :-

admin_header.blade.php file
admin_footer.blade.php file
admin_sidebar.blade.php file

We will add content into above three files and will also update admin_layout.blade.php file as shown in video.

In admin_header.blade.php file, add Navbar related code and we will include admin_header.blade.php file in admin_layout.blade.php file like below :-
@include('layouts.admin_layout.admin_header')

In admin_footer.blade.php file, add footer related code and we will include admin_footer.blade.php file in admin_layout.blade.php file like below :-.
@include('layouts.admin_layout.admin_footer')

In admin_sidebar.blade.php file, add sidebar related code and and we will include admin_sidebar.blade.php file in admin_layout.blade.php file like below :-
@include('layouts.admin_layout.admin_sidebar')

For middle content, add admin_dashboard.blade.php file under admin folder that we will create under /resources/views/ path and we will include in admin_layout.blade.php file like below :-
@yield('content')

In Part 3, we will create admin login page and will also update layout files, header, footer, sidebar and dashboard pages.

****

Lecture 3 :

In Part 3, we will update layout files; header, footer and sidebar to correct css/js/images and plugins paths. We will also create route, function for admin dashboard page and set its design as well.

1) Update admin_layout.blade.php file :-

First of all, we will update admin_layout.blade.php file to add Laravel asset/url to css/js/images and plugins paths.

2) Update admin_header.blade.php file :-
Now we will update admin_header.blade.php file to add Laravel asset to images paths.

3) Update admin_sidebar.blade.php file :-
Now we will update admin_sidebar.blade.php file to add Laravel asset to images paths.

4) Update admin_dashboard.blade.php file :-
Now we will update admin_dashboard.blade.php file to add Laravel asset to images paths.

5) Create AdminController.php file :-
Now create Admin folder under /app/Http/Controllers/ and then create AdminController.php file under Admin folder by running below artisan command :-
php artisan make:controller Admin/AdminController

We will keep all Admin Controllers separate from Front Controllers that will help us to do clear coding.

6) Create Route :-
We will create separate group in web.php file for admin routes so that we can keep them separately with namespace Admin and prefix admin.

Learn more about Routing/Controllers including namespaces/route groups in Laravel 6 Basics playlist :-
https://www.youtube.com/playlist?list...

We have added dashboard route without any authentication for now but in upcoming videos, we will add Guard Auth for admin routes.

7) Create function :-
We will create dashboard function in AdminController and will return to admin_dashboard.blade.php file.

You can see the view of dashboard page by opening below link :-
http://127.0.0.1:8000/admin/dashboard

In next video, we will create admin login page as well. After that we will start working on Multi Auth; Guard for admin and default auth for users.

****

Lecture 4 :

In Part-4, we will create admin login page. We will copy design/content of admin login from AdminLTE template and merge into our Laravel 6 E-com Admin Panel.

1) Create Route :-
First of all, create GET/POST route for admin login in Admin route group like below :-
Route::match(['get','post'],'/','AdminController@login');

2) Create Function :-
Now create login function in AdminController that will return to admin_login.blade.php file that we will create in next step.

3) Create admin_login.blade.php file :-
Now we will create admin_login.blade.php file in /resources/views/admin/ folder in which we will add content from login.html page from AdminLTE/pages/examples/ folder. We will not add admin design to it as admin design layout with header, footer and sidebar is for the internal pages of admin.

We will also correct paths in admin_login.blade.php file.

You can see in video; our admin login page is ready.

In next video, we will start working on Multi Auth and will use Guards for admin and default auth for users.

****

Lecture 5 :

In Part-5, we will start working on Multi Authentication for our Advance E-com website in Laravel 6. We will use Guards for Admins and default Laravel Auth for Users.

We will follow below steps to set Guards for Admin Panel / Admins.

1) Create Migration File
First of all, we will create migration file with name create_admins_table for creating admins table with below columns :-
id, name, type, mobile, email, email_verified_at, password, image, status

So, we will run below artisan command to create migration file for admins :-
php artisan make:migration create_admins_table

Open create_admins_table migration file and add all required columns mentioned earlier.

Now we will run below artisan command to create admins table with required columns :-
php artisan migrate

2) Create Admin model :-
Now we will create Admin model with below artisan command :-
php artisan make:model Admin

We will update content of Admin model file to set protected guard variable for admin and set other variables as shown in video.

We will also extends Admin class to Authenticatable and add its namespace as well.

3) Update auth.php file :-
We will update auth.php file located at config\auth.php to set guards for admin to assign session in driver and admins in provider as shown in video.

We will also set providers for admins to assign eloquent in driver and Admin class in model.

4) Create Admin Middleware :-
Now we will create Admin Middleware file by running below command :-
php artisan make:middleware Admin

5) Update kernel.php file :-
Now we will update kernel.php file located at app\http\ folder to register Admin middleware as global as shown in video.

6) Update Admin Middleware
Add Auth:guard check in Admin Middleware to protect the admin routes. This check will be false for now as we have not registered the admin guard yet.

7) Update web.php file :-
Add admin middleware group and move admin dashboard route under it to protect it from unauthorised access.

Now no one can access admin dashboard without login into the admin panel. We have used Guards to protect the admin routes including dashboard route.

In next video, we will work on admin login and logout functionality. We will register admin guard every time when admin logged in and destroy it every time when admin logged out from the admin panel.

****

Lecture 6 :

In Part-6 of Advance E-com series, though we will continue working on Multi-Authentication and will use guards for admin login. But in this video, we will work on Seeding for our Laravel 6 E-com website. Seeding helps us to insert data into table from file and will also help us for future projects to automatically create table with migration and data with seeding.

We will create Seeding for admins table to automatically insert admin data from file and it will help us in our next project as well.

1) Writing Seeder / Create AdminsTableSeeder file :-
First of all, we will generate seeder and create AdminsTableSeeder file where we will add record for admins table.

Run below artisan command to generate Seeder and create AdminsTableSeeder file :-
php artisan make:seeder AdminsTableSeeder

Above command will create AdminsTableSeeder.php file at \database\seeds\

Now open AdminsTableSeeder file and add record for admin.

We will generate hash password for admin by using Hash::make function as shown in video.

2) Update Admin model :-
Now we need to update Admin model to add all admins table columns in fillable array as shown in video.

3) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add AdminsTableSeeder class as shown in video.

4) Running Seeders / Run below command :-
Once you have written your seeder, you may need to regenerate Composer's autoloader using the dump-autoload command:
composer dump-autoload

5) Run below command :-
Now run last command that will finally insert admin record into admins table that we can use for admin login.
php artisan db:seed

You can see in video; we able to generate record for admins table with hash password.

In next video, we will work on admin login and logout functionality. We will register admin guard every time when admin logged in and destroy it every time when admin logged out from the admin panel.

****

Lecture 7 :

In Part-7, we will continue working on Multi Authentication for our Advance E-com website in Laravel 6.

We will work on admin login and logout functionality with Guards. We will register admin guard every time when admin logged in and destroy it every time when admin logged out from the admin panel.

1) Update admin_login.blade.php file :-
First of all, we will update admin_login.blade.php file to update Login form. We will set its action, CSRF token, username (email) and password.

2) Update login function :-
Now we will update login function at AdminController. We will add the condition for posted data and use guard for admin login as shown in video.

You can try login with username admin@admin.com and password 123456 that we have set in last video with Seeding.

Once logged in, we will redirect the user to Dashboard page and if the username or password is incorrect then we can redirect back the user and flash error message in admin login page.

3) Update admin_header.blade.php file :-
Remove all unwanted code and add "Logout" link for the user at top right side of the header as shown in video.

4) Create Route :-
Now we will create GET route for Admin Logout in web.php file like below :-

```
// Admin Logout
Route::get('logout','AdminController@logout');
```

5) Create logout function :-
Now we will create logout function in AdminController in which we will unset admin guard as shown in video.

6) Update admin_login.blade.php file :-
Update admin_login.blade.php file to show the error message if admin user or password is incorrect.

We can get the "error message alert bootstrap" script from below website :-
https://getbootstrap.com/docs/4.0/com...

Now check in video, we able to login in admin panel with Guard and logout as well. If username or password is incorrect then we are displaying error message in admin login page.

****

Lecture 8 :

In Part-8 of Advance E-com series, we will add Laravel validations for admin login form.

We will follow below Laravel link for integrating Laravel validations for admin login :-
https://laravel.com/docs/6.x/validation

1) Update login function :-
Update login function located at AdminController and write validator code for email and password as shown in video.

Take help from https://laravel.com/docs/6.x/validati... for writing validation logic.

2) Update admin_login.blade.php file :-
Now update admin_login.blade.php file to display the error if validation fails for email and password. You can check this at Laravel website under "Displaying The Validation Errors" section.
https://laravel.com/docs/6.x/validati...

3) Update Header Statement :-
Now add below statement at the top of AdminController to include Validator class for Laravel validation :-
use Validator;

Check now validation is working fine in admin login form. Default error message will come for email and password but we can write custom error message for email and password.

4) Update login function :-
For custom error messages, we will write validation logic in one array and custom error messages in another and then validate them as shown in video.

We will learn more about Laravel validations in future videos when we work on other forms.

In next video, we will work on settings page for change password functionality.

****

Lecture 9 :

In Part-9 of Advance E-com series, we will create settings page for our admin panel. In settings page, we will create update password form from where admin can update password of admin panel.

1) Create Route :-
First of all, create GET route for settings page in web.php file like below :-
Route::get('settings', 'AdminController@settings');

We will also keep this route in admin middleware Route::group to protect it from unauthorized access.

2) Create settings function :-
Now we will create settings function in AdminController that we will return to setting blade file.

3) Create admin_settings.blade.php :-
Now we will create admin_settings.blade.php file similar to admin_dashboard.blade.php file that we have created earlier on.

Now copy the form design content from AdminLTE template from path /pages/forms/general.html

We will add email, current password, new password and confirm password fields.

4) Update settings function :-
Now we will update settings function in AdminController to get the current admin details like email and password from Auth guard admin and return to settings page.

5) Include Admin Model :-
Now we will include Admin model at top of the AdminController like below :-
use App\Admin;

6) Update admin_settings.blade.php :-
Now we will update settings page and will show email in update password form that we have got from settings function at AdminController from admin guard.

7) Update admin_header.blade.php file :-
Now we will also update admin header to show settings link along with admin name and type who logged in.

In next video, we will upgrade our Laravel 6 project to Laravel 7 and after that will continue working on update password functionality.

****

Lecture 10 :

In Part-10 of Advance E-com series, we are going to upgrade our Laravel 6 E-commerce project to Laravel 7 by taking few simple steps :-

1) Update composer.json file :-

First of all, update composer.json file to upgrade PHP version from 7.2 to 7.2.5 and laravel/framework to 7.0.* from 6.0 along with many other dependencies as shown in video.

```
"require": {
        "php": "^7.2.5",
        "fideloper/proxy": "^4.2",
        "fruitcake/laravel-cors": "^1.0",
        "guzzlehttp/guzzle": "^6.3",
        "laravel/framework": "^7.0",
        "laravel/tinker": "^2.0"
    },
    "require-dev": {
        "facade/ignition": "^2.0",
        "fzaninotto/faker": "^1.9.1",
        "mockery/mockery": "^1.3.1",
        "nunomaduro/collision": "^4.1",
        "phpunit/phpunit": "^8.5"
    },
```

2) Update Handler.php file :-
Now we will update Handler.php file located at /app/Exceptions/ folder and will replace Exception instance with Throwable everywhere in file as shown in video.

3) Upgrade Laravel/ui Package :-
Now run below composer command to upgrade Laravel/ui Package for Auth :-
composer require laravel/ui "^2.0"

4) Run "composer update" command :-
Now run "composer update" command to finally upgrade our Laravel 6 E-com project to Laravel 7

5) Verify Laravel version :-
Now you can verify the laravel version by running below artisan command :-
php artisan --version

In my case, it's coming like Laravel Framework 7.0.3, check yours.

6) Run Laravel Project :-
Now run your Laravel E-com project by running below artisan command :-
php artisan serve

Check everything if working fine after updating your project to Laravel 7.0

In next video, we will be back to update password functionality. We will check if current password filled by admin in update password form is correct or not. We will check in Ajax and display message with current password field. Lastly, we will work on update password functionality.

****

Lecture 11 :

n part 11 of Advance E-com series, we will check if current password filled by admin in update password form is correct or not. We will check in Ajax and display message with current password field in Update Password form that we have created in Settings page of admin.

1) Update admin_settings.blade.php file :-
First of all, we will update "Update Password" form by adding action, name, id to form.

2) Create admin_script.js file :-
First of all, we will create admin_script.js file at /public/js/admin_js/ folder. Then we will add Jquery/Ajax in this file to check if current password entered by the admin is correct or not.

3) Update admin_layout.blade.php file :-
Now we will include admin_script.js file in admin_layout.blade.php file.

4) Create Route :-
Now we will create POST route in web.php file for checking current password that we have passed via Ajax URL :-
Route::POST('check-current-pwd','AdminController@chkCurrentPassword');

5) Create chkCurrentPassword function :-
Now we will create chkCurrentPassword function at AdminController in which we will check if current password entered by admin is correct or not. We will return true or false to Ajax to display the message at update password form.

6) Update admin_settings.blade.php file :-
Now in admin_settings file, in Password form, below Current password field, we will add one span tag with id to display message that we have returned from Ajax.

Now we will update our Jquery function. We will check true or false value in Ajax return to display the success or error message in chkPwd ID that we have just created.

7) Update VerifyCsrfToken.php
We will add '/admin/check-current-pwd' route in VerifyCsrfToken.php to exclude it from CSRF verification.

Now see in video, true comes if we give correct password 123456 and false comes if we give incorrect password. And we have added condition in Ajax to display error or success message based on it.

See now in video, message comes based on our current password. So if we give correct password then it shows Current Password is Correct, otherwise Incorrect.

In next video, we will finally update the password and then login with the updated password.

****

Lecture 12 :

In Part-12 of Advance E-com Series, we will continue working on update admin password functionality and in this video we are going to finally update the admin password. Also we will perform some laravel validation checks in update password form.

1) Update admin_header.blade.php file :-
First of all, update admin header with Admin name who logged in and Settings page link.

2) Create Route :-
Now we will create POST route in web.php file for updating current admin password :-
Route::POST('update-current-pwd','AdminController@updateCurrentPassword');

3) Create updateCurrentPassword function :-
Now we will create updateCurrentPassword function to update the current password and set the new password entered by the user but first we will check if current password entered is correct or not. If not correct we will send back the admin to update password form with error message. And if correct then we will compare new password with confirm password, if correct then we will update new password and return success message otherwise will return error message.

4) Update admin_settings.blade.php file :-
Update admin settings page with success and error message div's.

Check in video : Admin can able to update current admin panel password.

****

Lecture 13 :

In Part-13 of Advance E-com Series, we will start working on updating and validating admin details like admin name, image and mobile.

1) Create Route :-
Create GET/POST route for updating admin details in web.php file like below :-
Route::match(['get','post'],'update-admin-details', 'AdminController@updateAdminDetails');

2) Create updateAdminDetails function :-

Create updateAdminDetails function in AdminController and return to update_admin_details.blade.php file.

3) Create update_admin_details.blade.php file :-
Now create update_admin_details.blade.php file at resources/views/admin/ folder.

We will create update admin details form with admin name, email, image and mobile with email as read only.

4) Update updateAdminDetails function :-
Now update updateAdminDetails function to get admin name and mobile and update in admins table.

We will also validate name and mobile and return to update admin details form in case name and mobile is not valid.

5) Update update_admin_details.blade.php file :-
We will add alert div at update_admin_details.blade.php file that we will display in case if name or mobile is not valid.

In next video, we will validate and upload admin image as well.

****

Lecture 14 :

In Part-14 of Advance E-com Series, we will continue working on updating admin details and this time we will validate and upload admin image.

For uploading images, we will install intervention package that will help us to resize images as well.

1) Install Intervention Package :-
Simply run below composer command to install Intervention Package :-
composer require intervention/image

2) Update update_admin_details.blade.php file :-
Add enctype="multipart/form-data" in update admin details form to accept files and we will also add condition to show admin image and add another hidden field for current admin image.

3) Update updateAdminDetails function :-
Now we will update updateAdminDetails function to add validation for image and will add upload image script and finally save the image name in admins table as well.

We will create admin_photos folder under admin_images folder where we will store all admin images.

4) Update admin_sidebar.blade.php file :-
Now we will update admin sidebar with admin image who logged in. If no admin image exists then we will show some dummy image.

****

Lecture 15 :

In Part-15 of Advance E-com series, we will highlight current open section at admin sidebar. Like if we open dashboard then we will highlight dashboard at admin sidebar, if we open admin password page then we are going to highlight "Update Admin Password" at admin sidebar and so on.

We will set and compare session values to highlight current open section.

1) Update dashboard function :-
Update dashboard function with page session having dashboard value.

2) Update settings function :-
Update settings function with page session having settings value.
Session::put('page','settings');

3) Update updateAdminDetails function :-
Update updateAdminDetails function with page session having update-admin-details value.
Session::put('page','update-admin-details');

Future pages also we will add in session so that we can also highlight them on left sidebar.

4) Update admin_sidebar.blade.php file :-
We will add conditions at left sidebar to highlight sections if it matches with the page session value.

****

Lecture 16 :

In Part-16 of Advance E-com series, we will start working on sections that we will add as Men, Women and Kids for our E-commerce website.

We will first work on Sections module in which we will add Men, Women and Kids as Sections. Every section will have its own categories that we will work later on.

In this video, we will create sections table with Migration and will add records into it with Seeding.

We will also create controller and model for sections.

1) Create sections table :-
First of all, we will create sections table with migration. Create migration file with name create_sections_table for creating sections table with below columns :-
id, name, status

So, we will run below artisan command to create migration file for sections :-
php artisan make:migration create_sections_table

Open create_sections_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create sections table with required columns :-
php artisan migrate

2) Create Section model :-
Create Section model by running below command :-
php artisan make:model Section

3) Create SectionController :-
Create SectionController by running below command :-
php artisan make:controller Admin/SectionController

Now, We will create Seeding for sections table to insert sections like Men, Women and Kids from file.

4) Writing Seeder / Create SectionsTableSeeder file :-
First of all, we will generate seeder and create SectionsTableSeeder file where we will add records for sections table.

Run below artisan command to generate Seeder and create SectionsTableSeeder file :-
php artisan make:seeder SectionsTableSeeder

Above command will create SectionsTableSeeder.php file at \database\seeds\

Now open SectionsTableSeeder file and add record for section.

5) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add SectionsTableSeeder class as shown in video.

6) Run below command :-
Now run below command that will finally insert records into sections table.
php artisan db:seed

In next video, we will display sections in admin panel and will also work on enable/disable any section from admin panel.

****

Lecture 17 :

In Part-17 of Advance E-com series, we will display sections in admin panel.

1) Create SectionController :-
In last video, we have created SectionController under app/Http/Controllers/ but we need to remove from their and will add under app/Http/Controllers/Admin folder. We will add all admin related controllers under Admin folder and front related controllers under Front folder.

Create SectionController by running below command :-
php artisan make:controller Admin/SectionController

2) Create Route :-
Create GET route in web.php file in admin middleware group prefixed with admin and having namespace Admin for displaying sections in admin panel :-
Route::get('sections','SectionController@sections);

3) Create sections function :-
Now create sections function in SectionController to write query to display all the sections in admin panel and return to sections blade file that we will create under /resources/views/admin/sections/ folder.

4) Include Section model :-
Include Section model at top of SectionController.
use App\Section;

5) Create sections.blade.php file :-
Now create sections.blade.php file under /resources/views/admin/sections/ folder in which we will add content from LTE admin template data.html file located at folder /pages/tables/data.html and will display sections within foreach loop.

6) Update admin_layout.blade.php file :-
Now include paths for CSS and JS datatable files required for displaying sections in admin panel.

Now check in video; Men, Women and kids sections are displayed in datatable.

****

Lecture 18 :

In Part-18 of Advance E-com series, we will work on active/inactive status for Sections. We will use Jquery and Ajax to active/inactive status of section from sections table. Active status will have 1 value and Inactive status will have 0 value in sections table.

1) Update sections.blade.php file :-
Add id, class and section_id attributes for Active and Inactive status for sections at sections.blade.php file that are required to update the status with jquery and ajax.

2) Update admin_script.js file :-
Add updateSectionStatus jquery function in admin_script.js file in which we will pass status and section_id that we will return to ajax via admin/update-section-status route.

3) Create Route :-
Now we will create below Post route in admin middleware group in web.php file for updating status that we pass via ajax in last step.
Route::post('update-section-status','SectionController@updateSectionStatus');

4) Update VerifyCsrfToken.php :-
Add route "admin/update-section-status" in VerifyCsrfToken.php file so that CSRF token mismatch error won't come.

5) Create updateSectionStatus :-
Now we will create updateSectionStatus in SectionController to update the status of section in sections table and return back the updated status to ajax via json.

6) Update admin_script.js file :-
Update admin_script.js file again to get the status and section id in ajax response and update status in sections.blade.php file.

7) Update admin_sidebar.blade.php file :-
Update Admin sidebar to add sections tab. We will add Catalogues as heading or you can add heading of your choice. We will keep sections, categories, products and coupons under catalogues only.

****

Lecture 19 :

In Part-19 of Advance E-com Series in Laravel 7, we will start working on Categories module. We will create categories table with migration and will also add one category in it with Seeding though later on we will add category from admin panel.

We will also create controller and model for categories.

1) Create categories table :-

First of all, we will create categories table with migration. Create migration file with name create_categories_table for creating categories table with below columns :-
id, parent_id, section_id, category_name, category_image, category_discount, description, url, meta_title, meta_description, meta_keywords and status

So, we will run below artisan command to create migration file for categories :-
php artisan make:migration create_categories_table

Open create_categories_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create categories table with required columns :-
php artisan migrate

Now categories table has been created with all the required columns.

2) Create Category model :-
Create Category model by running below command :-
php artisan make:model Category

3) Create CategoryController :-
Create CategoryController in Admin folder at /app/Http/Controllers/Admin/ by running below command :-
php artisan make:controller Admin/CategoryController

Now, We will create Seeding for categories table to insert one test category like T-Shirts from file.

4) Writing Seeder / Create CategoryTableSeeder file :-
First of all, we will generate seeder and create CategoryTableSeeder file from where we will add one category for categories table.

Run below artisan command to generate Seeder and create CategoryTableSeeder file :-
php artisan make:seeder CategoryTableSeeder

Above command will create CategoryTableSeeder.php file at \database\seeds\

Now open CategoryTableSeeder file and add record for category.

5) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add CategoryTableSeeder class as shown in video.

6) Run below command :-
Now run below command that will finally insert category into categories table.
php artisan db:seed

In next video, we will work on View Categories page in admin panel.

****

Lecture 20 :

In Part-20 of Advance E-com Series in Laravel 7, we will continue working on Categories module. In this video, we will work on View Categories page in admin panel and will also add Active/Inactive status for categories.

1) Create Route :-
Create GET route in web.php file in admin middleware group prefixed with admin and having namespace Admin for displaying categories in admin panel :-

```
// Categories
Route::get('categories','CategoryController@categories');
```

2) Create categories function :-
Now create categories function in CategoryController to write query to display all the categories in admin panel and return to categories blade file that we will create under /resources/views/admin/categories/ folder.

3) Include Category model :-
Include Category model at top of CategoryController.

```
use App\Category;
```

4) Create categories.blade.php file :-
Now create categories.blade.php file under /resources/views/admin/categories/ folder in which we will add content from LTE admin template data.html file located at folder /pages/tables/data.html and will display categories within foreach loop.

Now we can see in video; we can able to see T-Shirts category in "View Categories" page along with URL and status in admin panel.

Now we will add functionality to toggle Active/Inactive status for categories like we have done for sections.

Add id, class and category_id attributes for Active and Inactive status for categories at categories.blade.php file that are required to update the status with jquery and ajax.

5) Update admin_script.js file :-
Add updateCategoryStatus jquery function in admin_script.js file in which we will pass status and category_id that we will return to ajax via admin/update-category-status route.

6) Create Route :-
Now we will create below Post route in admin middleware group in web.php file for updating status that we pass via ajax in last step.

Route::post('update-category-status','CategoryController@updateCategoryStatus');

7) Update VerifyCsrfToken.php :-
Add route "admin/update-category-status" in VerifyCsrfToken.php file so that CSRF token mismatch error won't come.

8) Create updateCategoryStatus :-
Now we will create updateCategoryStatus in CategoryController to update the status of category in categories table and return back the updated status to ajax via json.

9) Update admin_script.js file :-
Update admin_script.js file again to get the status and category id in ajax response and update status in categorys.blade.php file.

In next video, we will work on "Add/Edit Category" functionality in admin panel.

****

Lecture 21 :

In Part-21 of Advance E-com Series in Laravel 7, we will continue working on Categories module. We will start "Add/Edit Category" functionality and in this video, we will add Add/Edit Category page and its form.

1) Update categories.blade.php file :-
First of all, we will show "Add Category" link at top right side of the categories page in admin panel.

2) Create Route :-
Create GET/POST route for Add/Edit Category in web.php file under admin group with id parameter as optional (that is required in case of edit category) like below :-
Route::match(['get','post'],'add-edit-category/{id?}','CategoryController@addEditCategory');

3) Create addEditCategory function :-
Create addEditCategory function in CategoryController with parameter $id as optional. We will add condition to execute "Add Category" functionality in case $id is empty otherwise "Edit Category" functionality if $id is coming.
We will also get all the categories and return to add_edit_category.blade.php file that we will create in next step.

4) Create add_edit_category.blade.php file :-
Now we will create add_edit_category.blade.php file at path /resources/views/admin/categories/ and will add admin design to it.

We will create Add/Edit Category form with the help of General/Advance forms given in AdminLTE template. We will copy form design from AdminLTE template from general.html and advance.html files located at path /AdminLTE-3.0.2/pages/forms/ of AdminLTE folder as shown in video.

5) Update admin_layout.blade.php file :-
Add select2 CSS/JS files for advance html form that is having better select box script as shown in video.

Now check in video; our "Add Category Form" design is ready.

In next video, we will work on submitting Add Category form and also we are going to select Categories according to Sections with the help of Ajax.

****

Lecture 22 :

Url :
https://www.youtube.com/watch?v=tD64C9bK3Rw&list=PLLUtELdNs2ZaHaFmydqjcQ-YyeQ19Cd6u&index=22

In Part-22 of Advance E-com Series in Laravel 7, we will continue working on Categories module. In this video, we will work on "Add Category" functionality. We will add category details into categories table and will also add Laravel validations.

1) Update add_edit_category.blade.php file :-
First of all, we will update add_edit_category.blade.php file to make sure to add form action, name and id's for form and for all fields.

2) Update addEditCategory function :-
Now we will update addEditCategory function at CategoryController to add query for adding category details in categories table and return the user to categories page with success message.

We will also add Laravel validations to make sure correct category data added.

3) Include Header Statements :-
Include Session and Image class at top of CategoryController :-
use Session;
use Image;

4) Update add_edit_category.blade.php file :-
Now show error message above form at add_edit_category.blade.php file :-

5) Update categories.blade.php file :-

We will show success message in categories page if category successfully added.

Now you can see in video we can able to add category successfully.

****

Lecture 23 :

In Part-23 of Advance E-com Series in Laravel 7, we will continue working on "Add Category" functionality in admin panel. In this video, we will work on categories level select box in add category form that we will display all the categories with Ajax. This "Category Level" select box will help us to add sub categories as well.

1) Create append_categories_level.blade.php file :-
First of all, we will create append_categories_level.blade.php file under resources/views/admin/categories/ folder and add categories level select box from add_edit_category.blade.php file.

2) Update add_edit_category.blade.php file :-
We will add div with id appendCategoriesLevel and include newly created append_categories_level.blade.php file into it as shown in video.

3) Update admin_script.js file :-
Now we will create jquery function that will come into role when admin select section and we will pass section id with ajax to fetch categories to show in category level select box.

4) Create Route :-
Now we will create post route in web.php file to append categories level :-
Route::post('append-categories-level','CategoryController@appendCategoriesLevel');

5) Create appendCategoriesLevel function :-
Now we will create appendCategoriesLevel function in CategoryController to write the query to get the root categories of particular section and return into category level select box.

6) Update addEditCategory function :-
We want to allow the admin to add alphabets with spaces in "Category Name" field. Last time we have added alpha option that Laravel provides but it will not allow spaces so we will search on net for better solution.

Search for the keyword like 'laravel validation for alphabetic characters and spaces' and open below stackoverflow link to find the solution.

https://stackoverflow.com/questions/3...

We will add regular expression for allowing alphabets with spaces as shown in video.

7) Update VerifyCsrfToken.php file :-
Update VerifyCsrfToken.php file to add url /admin/append-categories-level to prevent CSRF token.

8) Update Category.php Model :-
Add subcategories function with hasMany relation to fetch all sub categories of parent categories.

9) Update appendCategoriesLevel function :-
Now update query in appendCategoriesLevel function with subcategories relation to get all sub categories of categories.

10) Update append_categories_level.blade.php file :-
Finally, update append_categories_level.blade.php file to add foreach loop for sub categories as shown in video.

Now Categories and Sub Categories depends upon Sections. If you select Men section then all categories and sub categories of men will appear and if you select Women section then all categories and sub categories of women will appear and so on..

In next video, we will update categories page to show parent categories and sections as well.

****

Lecture 24 :

In Part-24 of Advance E-com Series in Laravel 7, we will continue working on categories module in admin panel. In this video, we will update categories page to show parent categories and sections as well.

First of all, we will take steps to show sections in categories page.

1) Update Category model :-
Create section function in Category model to make belongsTo relation between categories and section in which every category belongs to section.

2) Update categories function :-
Now update categories function to update query to add section relation with it so that we can get section of every category.

3) Update categories.blade.php file :-
Now update categories.blade.php file to add section into it.

Now, we will take steps to show parent categories in categories page.

4) Update Category Model :-

Create parentcategory function in Category model to make belongsTo relation between categories and parent categories with parent_id in categories table.

5) Update categories function :-
Now update categories function to update query to add parentcategory relation with it so that we can get parent category of every category if exists.

6) Update categories.blade.php file :-
Now update categories.blade.php file to add parent category into it.

Now parent category and section appears in categories page in admin panel.

In next video, we will work on "Edit Category" functionality in admin panel.

****

Lecture 25 :

In Part-25 of Advance E-com Series in Laravel 7, we will start working on "Edit Category" functionality in admin panel. We will first show all the category details in edit category form when we try to edit any category details and finally update the category.

1) Update categories.blade.php file :-
First of all, update categories page in admin panel with Actions column that is having Edit and Delete links for category. We will add Edit link for now for editing the category.

2) Update addEditCategory function :-
Now, we will update addEditCategory function in CategoryController. We will fetch category data from query that we want to edit and return that category data to add_edit_category.blade.php file.

3) Update add_edit_category.blade.php file :-
Now we will update add_edit_category.blade.php file and show category data in form that we want to edit. We will also change action of form if category id coming from $categorydata array is not empty. We will allow the admin to change the section, category level and other category details along with category image.

4) Update addEditCategory function :-
Update addEditCategory function again in CategoryController to get the category level that we have selected at the time of "Add Category". We now need to select that category level in edit category form as well so we will return category level.

To be continued in next video.. please stay tune :)

****

Lecture 26 :

In Part-26 of Advance E-com Series in Laravel 7, we will continue working on "Edit Category" functionality in admin panel. In this video, we are going to display the category level in Edit Category form so that we can update level of category as well along with other category details.

1) Update append_categories_level.blade.php :-
Now we will update append_categories_level.blade.php file to select category level.

2) Update addEditCategory function :-
Finally, update addEditCategory function to update the query to update category details.

Now, you can check in video; we able to edit category as well.

In next video, we will work on delete category functionality along with delete category image.

****

Lecture 27 :

In Part-27, we will work on delete category functionality along with delete category image.

First of all, we will show Category Image with Delete link and functionality.

1) Update add_edit_category.blade.php file :-
Show Category Image if already added along with Delete link so add condition for it at add_edit_category.blade.php file.

2) Create Route :-
Now we will create GET route to delete category image with parameter category id in web.php file like below :-
Route::get('delete-category-image/{id}','CategoryController@deleteCategoryImage');

3) Create deleteCategoryImage function :-
Now create deleteCategoryImage function in CategoryController in which we will write query to delete category image from categories table and category_images folder.

Now we will delete category as well.

4) Update categories.blade.php file :-
Now we will update categories.blade.php file to add delete category link with every category listing.

5) Create Route :-
Now we will create GET route with parameter category id to delete category in web.php file like below :-

Route::get('delete-category/{id}','CategoryController@deleteCategory');

6) Create deleteCategory function :-
Now we will create deleteCategory function in CategoryController to write the query to delete the category with category id that we will get as parameter. After deleting the category, we will return to categories page with success message.

So, we able to delete the category image as well as category.

In next video, we will work on SweetAlert to alert the user before deleting the category as well as category image.

****

Lecture 28 :

In Part-28, we will add SweetAlert 2 Confirmation jQuery Library that will ask the admin to confirm deletion of category and category images.

We will first add simple javascript confirm alert and then SweetAlert.

1) Simple JavaScript Alert

2) SweetAlert 2 Javascript Library

1) Simple JavaScript Alert

Add confirmDelete class and name Category in delete cateogory link and then create confirmDelete click event in javascript to add simple confirm alert.

Update admin_layout.blade.php file to add confirmDelete click event in jQuery.

Now update admin_script.js file to add confirmDelete function in which we will add confirm script. If admin confirm then we will pass return true otherwise return false.

2) SweetAlert Javascript Library

Now remove earlier simple script to add SweetAlert jQuery script.

Follow below link to install SweetAlert 2
https://sweetalert2.github.io

Run below command to install SweetAlert 2 :-
npm install sweetalert2

Also, update admin_layout.blade.php file with below JS script

Check if SweetAlert script is working by adding sample script given in sweetalert2 website in admin_script.js file.

If script working then we will continue further.

Add record and recordid attributes in delete category link.

In record, we will pass "category" and in recordid, we will pass category id.

And disable href and keep class "confirmDelete" that we have added earlier.

Now update admin_script.js file and add Jquery script there with SweetAlert.

Now check in video, our SweetAlert is working fine while deleting categories.

Now, we will add SweetAlert for deleting category image. So update "Delete Image" link at add_edit_category.blade.php with record as "category-image", recordid as "category id".

And no need to do jquery function again to add SweetAlert. Same function will work for deleting category images as well.

****

Lecture 29 :

In Part-29 of Advance E-com Series in Laravel 7, we will start working on Products module. We will create products table with migration and will also add one product in it with Seeding though later on we will add product from admin panel.

We will also create controller and model for products.

1) Create products table :-
First of all, we will create products table with migration. Create migration file with name create_products_table for creating products table with below columns :-
id, category_id, section_id, product_name, product_code, product_color, main_image, description, wash_care, fabric, pattern, sleeve, fit, occasion, product_price, product_discount, product_weight, product_video, meta_title, meta_description, meta_keywords, is_featured and status

So, we will run below artisan command to create migration file for products :-
php artisan make:migration create_products_table

Open create_products_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create products table with required columns :-
php artisan migrate

Now products table has been created with all the required columns.

2) Create Product model :-
Create Product model by running below command :-
php artisan make:model Product

3) Create ProductsController :-
Create ProductsController in Admin folder at /app/Http/Controllers/Admin/ by running below command :-
php artisan make:controller Admin/ProductsController

Now, We will create Seeding for products table to insert one test product from file.

4) Writing Seeder / Create ProductsTableSeeder file :-
First of all, we will generate seeder and create ProductsTableSeeder file from where we will add one product for products table.

Run below artisan command to generate Seeder and create ProductsTableSeeder file :-
php artisan make:seeder ProductsTableSeeder

Above command will create ProductsTableSeeder.php file at \database\seeds\

Now open ProductsTableSeeder file and add record for product.

5) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add ProductsTableSeeder class as shown in video.

6) Run below command :-
Now run below command that will finally insert product into products table.
php artisan db:seed

In next video, we will work on View Product page in admin panel.

****

Lecture 30 :

In Part-30 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will work on View Products page in admin panel and will also add Active/Inactive status for products and will add the delete functionality as well.

1) Create Route :-
Create GET route in web.php file in admin middleware group prefixed with admin and having namespace Admin for displaying products in admin panel :-

// Products
Route::get('products','ProductsController@products');

2) Create products function :-
Now create products function in ProductsController to write query to display all the products in admin panel and return to products blade file that we will create under /resources/views/admin/products/ folder.

3) Include Product model :-
Include Product model at top of ProductsController.
use App\Product;

4) Create products.blade.php file :-
Now create products.blade.php file under /resources/views/admin/products/ folder to display products within foreach loop. We can copy design structure from categories.blade.php file that we have copied from LTE Admin template explained in earlier videos.

Now we can see in video; we can able to see Blue and Red Casual T-Shirts products in "View Products" page along with their product name/code/color and status in admin panel.

Now we will add functionality to toggle Active/Inactive status for products like we have done for categories.

Add id, class and product_id attributes for Active and Inactive status for products at products.blade.php file that are required to update the status with jquery and ajax.

5) Update admin_script.js file :-
Add updateProductStatus jquery function in admin_script.js file in which we will pass status and product_id that we will return to ajax via admin/update-product-status route.

6) Create Route :-
Now we will create below Post route in admin middleware group in web.php file for updating status that we pass via ajax in last step.
Route::post('update-product-status','ProductsController@updateProductStatus');

7) Update VerifyCsrfToken.php :-
Add route "admin/update-product-status" in VerifyCsrfToken.php file so that CSRF token mismatch error won't come.

8) Create updateProductStatus :-
Now we will create updateProductStatus in ProductsController to update the status of product in products table and return back the updated status to ajax via json.

9) Update admin_script.js file :-

Update admin_script.js file again to get the status and product id in ajax response and update status in products.blade.php file.

Delete functionality will automatically work after creating delete-product route and update id's at delete product link and make identical function like delete category.

In next video, we will display category and section for products and then will work on Add/Edit Product functionality.

****

Lecture 31 :

In Part-31 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will display category and section for products in view products page that we have created in last video. We will create Eloquent Relationships between product and category/section. Also, we will do sub query to get only required data to make our query fast.

1) Update Product model :-
First of all, update Product model to make Eloquent Relationships to get category and section of product.

2) Update products function :-
Now update products function to attach category and section relations in products query to get the category and section of the product.

3) Update products.blade.php file :-
Finally, we will show category and section of product in products.blade.php file in foreach loop.

4) Update products function :-
We will update products function once again to add sub query to select only required data from category and section model that we require. It will make our query fast also and we will do this in practise.

Now you can see in video; we can able to display category and section of the product.

In next video, we will work on Add Product functionality in admin panel.

****

Lecture 32 :

In Part-32 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will work on Add Product page in admin panel. We will make common Add/Edit Product form/route like we have done for categories in earlier videos.

1) Update products.blade.php file :-
First of all, we will show "Add Product" link at top right side of the products page in admin panel.

2) Create Route :-
Create GET/POST route for Add/Edit Product in web.php file under admin group with id parameter as optional (that is required in case of edit product) like below :-
Route::match(['get','post'],'add-edit-product/{id?}','ProductsController@addEditProduct');

3) Create addEditProduct function :-
Create addEditProduct function in ProductsController with parameter $id as optional. We will add condition to execute "Add Product" functionality in case $id is empty otherwise "Edit Product" functionality if $id is coming.
We will also get all the products and return to add_edit_product.blade.php file that we will create in next step.

4) Create add_edit_product.blade.php file :-
Now we will create add_edit_product.blade.php file at path /resources/views/admin/products/ and will add admin design to it similar to add_edit_categories.blade.php file.

We will create Add Product form with all columns that we have created in products table along with "Select Category" drop down in which we will show all categories and sub categories under sections.

5) Update addEditProduct function :-
We will update addEditProduct function to create filters arrays so that we can return filter options to display in "Add Product" form.

****

Lecture 33 :

Overview of the E-commerce Website in Laravel 6 / 7 that we are making.

Overview of the Admin Panel - Basic Structure
1 to 15 videos belongs to admin panel with Basic Structure that is required for all

Overview of the E-com related Admin Panel - Up till Part 32
After basic admin panel, we started E-com related admin panel that includes Sections, Categories and products structure so far.

We have completed 32 videos and currently working on products section...

Relation between Sections, Categories and Products tables..

Debug your code to identify the issue and resolve it..

Please watch below playlist for Advance E-com Series in Laravel 6 / Laravel 7 :-
https://www.youtube.com/playlist?list...

****

In Part-33 of Advance E-com Series in Laravel 7, we will continue working on Add Product form. In this video, we will add filters select boxes and category level drop down in which we will show all sections, their categories and sub categories under them. Like under Men Section; Tshirts, Shirts and Denims Categories will come and under Tshirts Category; Casual Tshirts Category will come. And user will able to select one of the category for the product.

1) Update add_edit_product.blade.php file :-
We will update add_edit_product.blade.php file to add filters select boxes.

2) Update Section model :-
Now, we will create categories function so that we can get categories and sub categories of section. We will add condition to pick all root and enabled categories.

3) Update addEditProduct function :-
We will update addEditProduct function to get sections with categories relation so that we can display sections, their categories and their sub categories in "Add Product" form.

4) Update add_edit_product.blade.php file :-
Now we will update add_edit_product.blade.php file once again to show sections with their categories and sub categories in "Select Category" drop down.

Now our "Add Product" form is ready. In next video, we will insert product in products table and will show it in view products page.

****

****

Lecture 34 :

In Part-34 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will insert product details in products table and will display it in view products page. And will also work on Laravel validations for adding product.

1) Update addEditProduct function :-

First of all, we will update addEditProduct function to get posted data and debug all product data is coming fine. After that, add laravel validations for mandatory data like category_id, product_name, product_code, product_price and product_color.

2) Update add_edit_product.blade.php file :-
Now we will update add_edit_product.blade.php file to show validation error messages and show earlier filled data as well so that admin/client not required to fill all product information again.

3) Update addEditProduct function :-
Update addEditProduct function once again to write query to save the products data in products table. We will also get section_id from category_id and will save it as well. We will return to add products page again with success message after saving the product.

In next video, we will add product main image and will resize into small, medium and large sizes. And we will also add product video as well.

****

Overview of the E-commerce Website in Laravel 6 / 7 that we are making up till Part 34. Debugging the code | Difference between JSON Object and Arrays | Which one is better to use? Please watch below playlist for Advance E-com Series in Laravel 6 / Laravel 7 :- https://www.youtube.com/playlist?list...
SHOW LESS

****

****

Lecture 35 :

In Part-35 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will add product main image after resize into small, medium and large sizes with the help of Intervention package that we have installed earlier in the beginning of the series. And we will also add product video as well in this video.

1) Create product_images folder :-
First of all, create product_images folder under /public/images/ path and then add small, medium and large folders under product_images folder that will look like below :-
/public/images/product_images/small/
/public/images/product_images/medium/
/public/images/product_images/large/

2) Update addEditProduct function :-

Update addEditProduct function in ProductsController to write code to add product main image by resizing it to small, medium and large sizes that we will upload in small, medium and large folders created in last step.

Intervention package helps us to resize images into 3 parts and add in folders.

3) Include Header Statement :-
Now include below statement at the top of ProductsController so that we can add/update and resize images with Intervention package that we have installed in Part-14 :- https://youtu.be/qdrmd7HVlgk
use image;

Now you can see in video, we able to add product images into their respective folders after resizing them.

Now we will add product videos.

4) Create product_videos folder :-
Create videos folder under /public/ folder and then create product_videos folder under /public/videos/ folder where we will upload all product videos.

5) Update addEditProduct function :-
Update addEditProduct function in ProductsController to write code to add product video.

Now we can able to add video of the product as well.

In next video, we will display product image in view products page.

****

Lecture 36 :

In Part-36 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we are going to show product images in view products page in admin panel. If product image of the product not added then we will show dummy image of the product. And we will generate dummy image online.

Update products.blade.php file :-
Update view products page in admin panel to show product main image if it exists otherwise show dummy product image that we will generate from below website :-
https://dummyimage.com

We will create dummy image of width 100px and height 115px and will add text "No Image" in it.

Now we will add one more condition that if product image not there in products folder then again it will show dummy image.

In next video, we will start working on edit functionality for the product.

****

Overview of the E-commerce Website in Laravel 6 / 7 that we are making up till Part 34.

One to one interaction with channel members and subscribers and will try to answer most of the queries.

Please watch below playlist for Advance E-com Series in Laravel 6 / Laravel 7 :-
https://www.youtube.com/playlist?list...

****

****

Lecture 37 :

In Part-37 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will start working on edit functionality for the product.

1) Update addEditProduct function :-
First of all, we will update addEditProduct function to get the product details from product id and return the product details to edit product form to display there.

2) Update add_edit_product.blade.php file :-
Update "Select Category" drop down and other fields to show the existing product details that we can update.

3) Update addEditProduct function :-
Update addEditProduct function once again to write query to update the product details from the edit product form on the updation of the form.

Now we able to update product details, image and video as well.

In next video, we will show product image and video in edit product form with delete link and will work on its functionality.

****

Lecture 38 :

In Part-38 of Advance E-com Series in Laravel 7, we will continue working on Products module. In this video, we will show product image and video in edit product form with delete link and will work on its functionality.

1) Update add_edit_product.blade.php file :-
Update add_edit_product.blade.php file to show product image in edit product form if user already added it while adding the product. We will show "Delete Image" link as well if product image is there.

Now we will show download link for the video if video added by the admin while adding the product. We will show "Delete Video" link as well if product image is there.

2) Create Routes :-
Now we will create GET routes for deleting product image and video in web.php file like below :-
Route::get('delete-product-image/{id}','ProductsController@deleteProductImage');
Route::get('delete-product-video/{id}','ProductsController@deleteProductVideo');

3) Create deleteProductImage function :-
Now create deleteProductImage function to delete product images from small, medium and large folders and from products table.

4) Create deleteProductVideo function :-
Now create deleteProductVideo function to delete product video from videos folders and from products table.

We able to show and delete both Product Main Image and Video.

In next video, we will start working on Products Attributes.

****

Lecture 39 :

In Part-39 of Advance E-com Series in Laravel 7, we will start working on Products attributes module. We will create products_attributes table with migration and will also add one product attribute in it with Seeding though later on we will add product attribute from admin panel.

We will also create model for products attributes.

1) Create products_attributes table :-

First of all, we will create products_attributes table with migration. Create migration file with name create_products_attributes_table for creating products_attributes table with below columns :-

id, product_id, sku, color, size, price, stock and status

So, we will run below artisan command to create migration file for products_attributes :-

```
php artisan make:migration create_products_attributes_table
```

Open create_products_attributes_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create products_attributes table with required columns :-

```
php artisan migrate
```

Now products_attributes table has been created with all the required columns.

2) Create ProductsAttribute model :-

Create ProductsAttribute model by running below command :-

```
php artisan make:model ProductsAttribute
```

Now, We will create Seeding for products_attributes table to insert one test product attribute from file.

4) Writing Seeder / Create ProductsAttributesTableSeeder file :-

First of all, we will generate seeder and create ProductsAttributesTableSeeder file from where we will add one product attribute for products_attributes table.

Run below artisan command to generate Seeder and create ProductsAttributesTableSeeder file :-

php artisan make:seeder ProductsAttributesTableSeeder

Above command will create ProductsAttributesTableSeeder.php file at \database\seeds\

Now open ProductsAttributesTableSeeder file and add record for product attribute.

5) Update DatabaseSeeder.php file :-

Now update DatabaseSeeder.php file located at database/seeds/ to add ProductsAttributesTableSeeder class as shown in video.

6) Run below command :-

Now run below command that will finally insert product into products_attributes table.

php artisan db:seed

In next video, we will work on View Product Attributes page in admin panel.

****

Lecture 40 :

In Part-40 of Advance E-com Series in Laravel 7, we will continue working on Products attributes module. From this video, we will start working on Add/View Product Attributes page in admin panel. For Products Attributes, we will add Product Size, SKU, Price and Stock. Like if Product is Red Casual T-Shirt then its product attributes can be like below :-

Size	SKU	Price	Stock
Small	RC01-S	1000	10
Medium	RC01-M	1100	10
Large	RC01-L	1200	10

There is a common page in which we will show Add Products Attribute form and at bottom we will show the products attributes that we have added so far for the product. In this video, we will create Add Products Attribute Page and going to show some Product Details like Product Code, Name, Color along with Product Image. Also, we are going to find and integrate Multiple fields Add/Remove script for product attributes.

1) Update products.blade.php file :-
First of all, we will update products.blade.php file to add product attributes link in Actions column. Also we are going to update all action links to fontawesome icons that will look good and attractive.

2) Create Route :-
Now we will create GET/POST Route for add-attributes in web.php file like below :-

```
Route::match(['get','post'],'add-attributes/{id}','ProductsController@addAttributes');
```

3) Create addAttributes function :-
Now we will create addAttributes function in ProductsController and return product data that we will get from product id and return to add_attributes.blade.php file.

4) Create add_attributes.blade.php file :-
Now we will create add_attributes.blade.php file under /resources/views/admin/products/ folder and will show product details like Product Name, Product Code, Product Color and Product main image.

5) Search Google for Multiple Add/Remove fields script :-
Now we will search in Google for "add remove fields dynamically in jquery" so that we can add multiple attributes like multiple sizes, stock, price and there sku's. It will save our time otherwise we can also build of our own as well.

We can take help from below website :-
https://www.codexworld.com/add-remove...

6) Update admin_script.js
Now we will update admin_script.js to add the jquery script from the website that we have found in last step.

7) Update add_attributes.blade file
Now we will update add_attributes.blade file to add all attributes fields Size, SKU, Price and Stock with Add/Remove link.

We need to update both add_attributes.blade.php file and admin_script.js vice versa so please watch in video..

8) Update addAttributes function :-
Update addAttributes function once again to get the posted attributes and check/debug in function if we are getting complete data.

In next video, we will continue and save products attributes in products_attributes table after adding some conditions.

****

Lecture 41 :

Review Class : (Lecture 1-40) ⇒

Welcome to the Live Session of Advance E-commerce Series in Laravel 7.

In this live session, we will review whatever we have done so far that includes basic admin panel, e-com related admin panel related to Sections, Categories, Products and Products Attributes.

We will also take overview of the upcoming module including Product Alternate Images.

We will also learn about the relation between sections, categories, products, products_attributes and products_images tables.

1) Section - hasMany - Categories
One section can have many categories so we use hasMany relation here

2) Category - belongsTo - Section
One category can belong to one section so we use belongsTo relation here

3) Category - hasMany - Products
One category can have many products so we use hasMany relation here

4) Product - belongsTo - Category
One product can belong to one category so we use belongsTo relation here

5) Product - hasMany - Attributes
One product can have many attributes so we use hasMany relation here

6) Attribute - belongsTo - Product
One attribute can belong to one product so we use belongsTo relation here

7) Product - hasMany - Images
One product can have many images so we use hasMany relation here

8) Image - belongsTo - Product
One image can belong to one product so we use belongsTo relation here

Basic Admin Panel

Part 1 :- Install Laravel 6 or 7
Part 2 to 4 :- Merge AdminLTE template into Laravel :- Admin login and dashboard
Part 5 :- Multi Auth Settings | Guard for Admins | Default Auth for Users
Part 6 :- Database Seeding | Create admins table with Migration and insert data with Seeding
Part 7: - Admin Panel Login with Guard
Part 8 :- Laravel Validations
Part 9 :- Change Admin Password
Part 10 :- Upgrade Laravel 6 to 7
Part 11 & 12 :- Continue to Change Admin Password
Part 13 & 14 :- Update Admin Details | Install Intervention Package for upload Admin Photo
Part 15 :- Update Admin Sidebar

E-com related Admin Panel

Sections Module
Part 16 :- Create sections table with Migration and add data with Seeding
Part 17 & 18 :- Display Sections with Active/Inactive Status

Categories Module
Part 19 :- Create categories table with Migration and add data with Seeding
Part 20 to 28 :- CRUD Operation for Categories - Add/Edit/View/Delete

Products Module
Part 29 :- Create products table with Migration and add data with Seeding

Part 30 to 38 :- CRUD Operation for Products - Add/Edit/View/Delete

Products Attributes Module (In Progress)
Part 39 :- Create products_attributes table with Migration and add data with Seeding
Part 40 :- Create Add Product Attributes Form

Please watch below playlist for Advance E-com Series in Laravel 6 / Laravel 7 :-
https://www.youtube.com/playlist?list...

Please watch below playlist for Basic E-com Series in Laravel 5.6 / 5.7 / 5.8 / 6 :-
https://www.youtube.com/playlist?list...

Also, join Stack Developers Group to get time to time update, get your queries resolved and help others to join live with me :-
https://facebook.com/groups/stackdeve...
****

Lecture 41 :

In Part-41 of Advance E-com Series in Laravel 7, we will continue working on Products attributes module. In this video, we are going to add products attributes in products_attributes table after validating them.

Update addAttributes function :-

Update addAttributes function to save all attributes in products_attributes table after adding two validations :-

i) All SKU must be unique so if SKU already exists then we will return with error message.

ii) If same size already added for Product then we will return with error message.

After passing the validations, we are going to save the attributes in products_attributes table and return with success message.

Now admin can able to add product attributes after validating them.

In next video, we are going to display product attributes added by the admin.

****

Lecture 42 :

In Part-42 of Advance E-com Series in Laravel 7, we will continue working on Products attributes module. In this video, we are going to display product attributes added in last video.

1) Create attributes function :-
First of all, create attributes function in Product model to create hasMany relation between product and their attributes. One Product can have many attributes so hasMany relation will work here.

2) Update addAttributes function :-
Update addAttributes function once again to update query to get product data along with their attributes by attaching attributes function with it.

3) Update add_attributes.blade.php file :-
Update add_attributes.blade.php file to show added product attributes that we return from the function in last step.

Now admin can able to add and view product attributes in the same page.

In next video, we are going to work on update/delete of product attributes.

****

Lecture 43 :

In Part-43 of Advance E-com Series in Laravel 7, we will continue working on Products attributes module. In this video, we are going to work on the update of the product attributes stock and price. We will provide option to update stock and price only that can vary from time to time but SKU and Size will remain fix but admin can delete it in case added wrong by mistake.

1) Update add_attributes.blade.php file :-

We will update add_attributes.blade.php file to add form and with price and stock as input fields in array so that we can update multiple attributes together.

2) Create Route :-
Now we will create POST route to edit attributes and will pass product id as parameter in web.php file like below :-
Route::match(['get','post'],'edit-attributes/{id}','ProductsController@editAttributes');

3) Create editAttributes function :-
Now create editAttributes function in ProductsController to get the attribute data with product id to update attributes and return to product attributes page with success message.

Now we can able to update the product attributes price and stock.

In next video, we will work on disable and delete functionality for the product attribute.

****

Lecture 44 :

In Part-44 of Advance E-com Series in Laravel 7, we will continue working on Products attributes module. In this video, we will work on disable and delete functionality for the product attribute.

1) Update add_attributes.blade.php file :-

First of all, update add_attributes.blade.php file to show Active/Inactive and Delete Links.

First we will work on Active/Inactive status for Product Attributes.

2) Update admin_script.js file :-

Now add jquery function for update attribute status at admin_script.js file.

3) Create Route :-

Now we will create POST route to update attribute status in web.php file like below :-

```
Route::post('update-attribute-status','ProductsController@updateAttributeStatus');
```

4) Update VerifyCsrfToken.php

Update VerifyCsrfToken.php to add "/admin/update-attribute-status" route to disable CSRF token check.

5) Create updateAttributeStatus function :-

Now create updateAttributeStatus function in ProductsController to write query to update Active/Inactive status for the attribute.

We now able to update active/inactive status for the product attribute.

For delete attribute; take below steps :-

6) Create Route :-

Now create GET route for delete attribute in web.php file like below :-

```
Route::get('delete-attribute/{id?}','ProductsController@deleteAttribute');
```

7) Create deleteAttribute function :-

Now create deleteAttribute function for writing query to delete the attribute and return with success message.

****

Lecture 45 :

In Part-45 of Advance E-com Series in Laravel 7, we will start working on Products images module. We will create products_images table with migration and will also add one product image in it with Seeding though later on we will add product image from admin panel. We will also create model for products images.

1) Create products_images table :-
First of all, we will create products_images table with migration. Create migration file with name create_products_images_table for creating products_images table with below columns :-
id, product_id, image and status

So, we will run below artisan command to create migration file for products_images :-
php artisan make:migration create_products_images_table

Open create_products_images_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create products_images table with required columns :-
php artisan migrate

Now products_images table has been created with all the required columns.

2) Create ProductsImage model :-
Create ProductsImage model by running below command :-
php artisan make:model ProductsImage

Now, We will create Seeding for products_images table to insert one test product image from seeder command.

3) Writing Seeder / Create ProductsImagesTableSeeder file :-
First of all, we will generate seeder and create ProductsImagesTableSeeder file from where we will add one product image for products_images table.

Run below artisan command to generate Seeder and create ProductsImagesTableSeeder file :-
php artisan make:seeder ProductsImagesTableSeeder

Above command will create ProductsImagesTableSeeder.php file at \database\seeds\

Now open ProductsImagesTableSeeder file and add record for product image.

4) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add ProductsImagesTableSeeder class as shown in video.

5) Run below command :-
Now run below command that will finally insert product into products_images table.
php artisan db:seed

In next video, we will work on Add/View Product Images page in admin panel.

****

Lecture 46 :

In Part-46 of Advance E-com Series in Laravel 7, we will continue working on Products images module. In this video, we will create Add/View Product Images page in admin panel similar to products attributes page.

First of all, we will update few things in our advance ecom series for products module :-

Update addEditProduct function :-
Update addEditProduct function in ProductsController and remove all conditions.

Update database.php file :-
Just update strict to false in mysql array in database.php file located at config folder.

Update products table :-
And update products table to set "No" value for "is_featured" column.

ALTER TABLE `products` CHANGE `is_featured` `is_featured` ENUM('No','Yes') CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'No';

Now we will create Add/View Product Images page in admin panel.

1) Update products.blade.php file :-
Create "Add Images" link icon in Actions column at products.blade.php file

2) Create Route :-
Now create GET/POST route for add images with parameter product id in web.php file like below :-

```
// Images
Route::match(['get','post'],'add-images/{id}','ProductsController@addImages');
```

3) Create addImages function :-

Now create addImages function in ProductsController with parameter $id and return product details to add_images.blade.php file that we will create in next step.

Also create images relation in Product model and attach with query. One Product can have many images so we will add hasMany relation.

4) Create add_images.blade.php file :-
Now create add_images.blade.php file similar to add_attributes.blade.php file in which we will replace attributes fields with image field and replace added attributes with images.

In next video, we will add multiple images and will show them. Also, we will work on delete/disable functionality.

****

Lecture 47 :

In Part-47 of Advance E-com Series in Laravel 7, we will continue working on Products images module. In this video, we will add multiple images and will show them. Also, we will work on delete/disable functionality.

1) Update addImages function :-
We will update addImages function to upload single or multiple alternate product images after resize.

We will save product images in products_images table and upload into small/medium and large folders located at /public/images/product_images/

Now check in video; we can able to upload multiple images of the product.

Now we will work on active/inactive functionality for the product image.

2) Update add_images.blade.php file :-
Update add_images.blade.php file to show Active/Inactive and Delete Links with their respective classes and id's.

3) Update admin_script.js file :-
Now update admin_script.js file to add jquery click function with updateImageStatus class for updating image status.

4) Create Route :-
Now create POST route for updating image status in web.php file like below :-
Route::post('update-image-status','ProductsController@updateImageStatus');

5) Create updateImageStatus function :-

Now create updateImageStatus function to update status for the image from Active to Inactive or Inactive to Active.

Now check in video; we can able to update image status of the product.

Now we will work on delete functionality for the product image.

6) Create Route :-
Now create GET route for deleting product image in web.php file like below :-
Route::get('delete-image/{id?}','ProductsController@deleteImage');

7) Create deleteImage function :-
Create deleteImage function to delete the product image from products_images table as well as small/medium and large folders.

****

Lecture 48 :

In Part-48 of Advance E-com series, we will start working on brands that we can add dynamically from admin panel. Every product we will add under some brand that we will manage from admin panel.

In this video, we will create brands table with Migration and will add records into it with Seeding.

We will also create controller and model for brands.

1) Create brands table :-
First of all, we will create brands table with migration. Create migration file with name create_brands_table for creating brands table with below columns :-
id, name, status, created_at, updated_at

So, we will run below artisan command to create migration file for brands :-
php artisan make:migration create_brands_table

Open create_brands_table migration file and add all required columns mentioned earlier.

Now, we will run below artisan command to create brands table with required columns :-
php artisan migrate

2) Create Brand model :-
Create Brand model by running below command :-
php artisan make:model Brand

3) Create BrandController :-

Create BrandController by running below command :-
php artisan make:controller Admin/BrandController

Now, We will create Seeding for brands table to insert dummy brands like Arrow, Gap, Lee, Monte Carlo and Peter England from file. You can add your own choice of brands as per your requirement.

4) Writing Seeder / Create BrandsTableSeeder file :-
First of all, we will generate seeder and create BrandsTableSeeder file where we will add records for brands table.

Run below artisan command to generate Seeder and create BrandsTableSeeder file :-
php artisan make:seeder BrandsTableSeeder

Above command will create BrandsTableSeeder.php file at \database\seeds\

Now open BrandsTableSeeder file and add query for adding brands.

5) Update DatabaseSeeder.php file :-
Now update DatabaseSeeder.php file located at database/seeds/ to add BrandsTableSeeder class as shown in video.

6) Run below command :-
Now run below command that will finally insert records into brands table.
php artisan db:seed

In next video, we will display brands in admin panel and will also work on enable/disable any brand from admin panel.

****

Lecture 49 :

Watch: https://youtu.be/70nnbY8whiw (Better video made due to issues in live session)

Welcome to the Live Session of Advance E-commerce Series in Laravel 7.

In this live session, we will review whatever we have done so far that includes basic admin panel, e-com related admin panel related to Sections, Categories, Products, Attributes and Brands.

And, we will learn how to debug the code to identify the issue to resolve in Laravel, jQuery and Ajax. In jQuery and Ajax, we will check the issue in console to find and resolve.

Basic Admin Panel

Part 1 :- Install Laravel 6 or 7

Part 2 to 4 :- Merge AdminLTE template into Laravel :- Admin login and dashboard
Part 5 :- Multi Auth Settings | Guard for Admins | Default Auth for Users
Part 6 :- Database Seeding | Create admins table with Migration and insert data with Seeding
Part 7: - Admin Panel Login with Guard
Part 8 :- Laravel Validations
Part 9 :- Change Admin Password
Part 10 :- Upgrade Laravel 6 to 7
Part 11 & 12 :- Continue to Change Admin Password
Part 13 & 14 :- Update Admin Details | Install Intervention Package for upload Admin Photo
Part 15 :- Update Admin Sidebar

E-com related Admin Panel

Sections Module
Part 16 :- Create sections table with Migration and add data with Seeding
Part 17 & 18 :- Display Sections with Active/Inactive Status

Categories Module
Part 19 :- Create categories table with Migration and add data with Seeding
Part 20 to 28 :- CRUD Operation for Categories - Add/Edit/View/Delete

Products Module
Part 29 :- Create products table with Migration and add data with Seeding
Part 30 to 38 :- CRUD Operation for Products - Add/Edit/View/Delete

Products Attributes Module
Part 39 :- Create products_attributes table with Migration and add data with Seeding
Part 40 :- Create Add Product Attributes Form
Part 41 :- Add/Validate Product Attributes
Part 42 :- Display Product Attributes
Part 43 :- Update Product Attributes : Stock / Price
Part 44 :- Disable/Delete Product Attributes

Products Images Module
Part 45 :- Create products_images table with Migration and add dummy images with Seeding
Part 46 :- Add/View Product Images Form/Page
Part 47 :- Add Multiple Product Images | Delete | Disable

Brands Module
Part 48 :- Create brands table with Migration and add dummy brands with Seeding
(Upcoming Videos)
Part 49 :- Add/Edit/Delete Active/Inactive Brands (CRUD)
Part 50 :- Add Brands under Products | Final touch to Admin Panel

Please support the channel by doing following :-

Free Steps :-

1) Subscribe Stack Developers Youtube Channel (if not subscribed yet) :-
https://www.youtube.com/stackdevelopers

2) Join Stack Developers Facebook Group :- https://www.facebook.com/groups/stack...

3) Like Stack Developers Facebook Page :- https://www.facebook.com/stacdevelopers

4) Like every video of Advance E-com Series seen/made so far (in case following advance series)
https://www.youtube.com/playlist?list...

Like every video of Basic E-com Series seen/made so far (in case following basic series)
https://www.youtube.com/playlist?list...

Or

Simply become channel member :-
https://www.youtube.com/channel/UCExO...

Friends who supports the channel always get special attention :)

****

Lecture 50 :

Debug Laravel Code to identify/resolve issues | Resolve jQuery/Ajax errors | Learn Inspect Element

Url :

https://www.youtube.com/watch?v=70nnbY8whiw&list=PLLUtELdNs2ZaHaFmydqjcQ-YyeQ19Cd6u&index=54

****

In this video, we will learn how to debug the code to identify the issue to resolve in Laravel, jQuery and Ajax.

jQuery and Ajax issues, we will identify in inspect element to resolve in Firefox, Chrome and Safari.

(Due to issues in Live Session, better video made to learn debugging in Laravel to resolve jQuery/Ajax Issues)

****