LAMPIRAN

AdminLoginController

```
<?php

namespace App\Http\Controllers\admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

class AdminLoginController extends Controller
{
public function index() {
return view('admin.login');
}

public function authenticate(Request $request){

$validator = Validator::make($request->all(),[
'email' => 'required|email',
'password' => 'required'

]);
```

```
if ($validator->passes()) {

if(Auth::guard('admin')->attempt(['email' => $request->email,'password'=>
$request->password],$request->get('remember'))){

$admin = Auth::guard('admin')->user();

if($admin->role == 2){
return redirect()->route('admin.dashboard');
}else{

Auth::guard('admin')->logout();
return redirect()->route('admin.login')
->with('error', 'Anda Tidak Berwenang Mengakses Panel Admin.');
}

}else{
return redirect()->route('admin.login')->with('error', 'Salah satu Email/Kata Sandi Salah');
}

} else {
return redirect()->route('admin.login')
->withErrors($validator)
->withInput($request->only('email'));
```

```
}
}

}
```

BrandController

```
<?php

namespace App\Http\Controllers\admin;

use App\Exports\ExportBrand;
use App\Http\Controllers\Controller;
use App\Imports\BrandImport;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use App\Models\Brand;
use PDF;
use Carbon\Carbon;
use Maatwebsite\Excel\Facades\Excel;

class BrandController extends Controller
{

    public function index(Request $request) {
        $brands = Brand::oldest('id');
```

```
    if ($request->get('keyword')) {
        $brands = $brands->where('name','like','%' .$request->keyword.'%');
    }

    $brands = $brands->paginate(10);

    return view('admin.brands.list', compact('brands'));

}

public function create() {
    return view('admin.brands.create');
}

public function store(Request $request) {
    $validator = Validator::make($request->all(),[
        'name' => 'required',
        'slug' => 'required|unique:brands',
    ]);

    if ($validator->passes()) {
        $brand = new Brand();
        $brand->name = $request->name;
        $brand->slug = $request->slug;
```

```
        $brand->status = $request->status;
        $brand->save();

        $request->session()->flash('success','Brands Berhasil di Buat.');

        return response()->json([
            'status' => true,
            'message' => 'Brand Berhasil di Tambahkan'
        ]);

} else {
        return response()->json([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function edit($id, Request $request) {
    $brand = Brand::find($id);

    if(empty($brand)) {
        $request->session()->flash('error', 'Catatan tidak di Temukan,');
        return redirect()->route('brands.index');
    }
```

```
    $data['brand'] = $brand;
    return view('admin.brands.edit',$data);
}

public function update($id, Request $request) {

    $brand = Brand::find($id);

    if(empty($brand)) {
        $request->session()->flash('error', 'Catatan tidak di Temukan,');
        return response()->json([
            'status' => false,
            'notFound' => true
        ]);
    }

    $validator = Validator::make($request->all(),[
        'name' => 'required',
        'slug' => 'required|unique:brands,slug,'.$brand->id.',id',
    ]);

    if ($validator->passes()) {
        $brand->name = $request->name;
        $brand->slug = $request->slug;
```

```
        $brand->status = $request->status;
        $brand->save();

        $request->session()->flash('success','Brands di Updated.');

        return response()->json([
            'status' => true,
            'message' => 'Brand Berhasil Updated'
        ]);

} else {
        return response()->json([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function destroy($id, Request $request){
    $brand = Brand::find($id);

     if (empty($brand)) {
            $request->session()->flash('error','Catatan tidak di Temukan');
            return response([
                'status' => false,
```

```
            'notFound' =>true
        ]);
    }

    $brand->delete();

    $request->session()->flash('success','Brand Berhasil di Hapus.');

    return response([
        'status' => true,
        'message' => 'Brand Berhasil di Hapus.'
    ]);
}

public function export_excel(){
  return Excel::download(new ExportBrand, "brand.xlsx");
}

public function import_excel()
{
   try {
      Excel::import(new BrandImport, request()->file('file'));
      session()->flash('success', 'File Excel berhasil di Impor.');
   } catch (\Exception $e) {
      session()->flash('error', 'Kesalahan mengimpor file Excel: ' . $e-
```

```
>getMessage());
        }

        return back();
    }

    public function export_pdf()
    {
        // Retrieve order details and customer information
        $brands = Brand::all();
        $data['brands'] = $brands;
        $now = Carbon::now()->format('Y-m-d');
        $data['now'] = $now;

        // Generate the PDF
        $pdf = PDF::loadView('admin.report.brand', compact('data'));

        // Set options if needed (e.g., page size, orientation)
        $pdf->setPaper('A4', 'potrait');

        // Download the PDF with a specific filename
        return $pdf->stream('Brand.pdf');
    }
}
```

CategoryController

```
<?php

namespace App\Http\Controllers\admin;

use App\Exports\ExportCategory;
use App\Http\Controllers\Controller;
use App\Imports\CategoryImport;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use App\Models\Category;
use Illuminate\Support\Facades\File;
use App\Models\TempImage;
use Carbon\Carbon;
use Image;
use Maatwebsite\Excel\Facades\Excel;
use PDF;

class CategoryController extends Controller
{
    public function index(Request $request) {
        $categories = Category::oldest();
        if(!empty($request->get('keyword'))){
            $categories = $categories->where('name','like','%'.$request->get('keyword').'%');
```

```
    }
    $categories = $categories->paginate(10);

    return view('admin.category.list',compact('categories'));
}

public function create() {
    return view('admin.category.create');
}

public function store(Request $request){
    $validator = Validator::make($request->all(),[
        'name' => 'required',
        'slug' => 'required|unique:categories',
    ]);

    if ($validator->passes()){

        $category = new Category();
        $category->name = $request->name;
        $category->slug = $request->slug;
        $category->status = $request->status;
        $category->showHome = $request->showHome;
        $category->save();
```

```
// Save Image Here
if (!empty($request->image_id)) {
    $tempImage = TempImage::find($request->image_id);
    $extArray = explode('.',$tempImage->name);
    $ext = last($extArray);

    $newImageName = $category->id.'.'.$ext;
    $sPath = public_path().'/temp/'.$tempImage->name;
    $dPath = public_path().'/uploads/category/'.$newImageName;
    File::copy($sPath,$dPath);

    // Generate Image Thumbnail
    $dPath = public_path().'/uploads/category/thumb/'.$newImageName;
    $img = Image::make($sPath);
    $img->resize(450, 600);
    $img->save($dPath);

    $category->image = $newImageName;
    $category->save();
}

$request->session()->flash('success', 'Kategori Berhasil di Tambahkan');

return response()->json([
    'status' => true,
```

```
            'message' => 'Kategori Berhasil di Tambahkan'
        ]);

    }else {
        return response()->json([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function edit($categoryId, Request $request){
    $category = Category::find($categoryId);
    if(empty($category)) {
        return redirect()->route('categories.index');
    }
    return view('admin.category.edit',compact('category'));
}

public function update($categoryId, Request $request){

    $category = Category::find($categoryId);

    if(empty($category)) {
        $request->session()->flash('error', 'Kategori Kosong');
```

```
    return response()->json([
     'status' => false,
     'notFound' => true,
     'message' => 'Kategori Kosong'
    ]);
}

$validator = Validator::make($request->all(),[
    'name' => 'required',
    'slug' => 'required|unique:categories,slug,'.$category->id.',id',
]);

if ($validator->passes()){

    $category->name = $request->name;
    $category->slug = $request->slug;
    $category->status = $request->status;
    $category->showHome = $request->showHome;
    $category->save();

    $oldImage = $category->image;

    // Save Image Here
    if (!empty($request->image_id)) {
        $tempImage = TempImage::find($request->image_id);
```

```
    $extArray = explode('.',$tempImage->name);
    $ext = last($extArray);

    $newImageName = $category->id.'-'.time().'.'.$ext;
    $sPath = public_path().'/temp/'.$tempImage->name;
    $dPath = public_path().'/uploads/category/'.$newImageName;
    File::copy($sPath,$dPath);

    // Generate Image Thumbnail
    $dPath = public_path().'/uploads/category/thumb/'.$newImageName;
    $img = Image::make($sPath);
    $img->fit(450, 600, function ($constraint) {
        $constraint->upsize();
    });
    $img->save($dPath);
    $category->image = $newImageName;
    $category->save();

    // Delete Old Image Here
    File::delete(public_path().'/uploads/category/thumb/'.$oldImage);
    File::delete(public_path().'/uploads/category/'.$oldImage);
}

$request->session()->flash('success', 'Kategori Berhasil di Update');
```

```
            return response()->json([
                'status' => true,
                'message' => 'Kategori Berhasil di Update'
            ]);

        }else {
            return response()->json([
                'status' => false,
                'errors' => $validator->errors()
            ]);
        }
    }

    public function destroy($categoryId, Request $request){
        $category = Category::find($categoryId);

        if(empty($category)) {
            $request->session()->flash('error','Kategori Kosong');
            return response()->json([
                'status' => true,
                'message' => 'Kategori Kosong'
            ]);
            //return redirect()->route('categories.index');
        }
        File::delete(public_path().'/uploads/category/thumb/'.$category->image);
```

```
        File::delete(public_path().'/uploads/category/'.$category->image);

        $category->delete();

        $request->session()->flash('success','Kategori Berhasil di Hapus');

        return response()->json([
            'status' => true,
            'message' => 'Kategori Berhasil di Hapus'
        ]);

    }

    public function export_excel(){
      return Excel::download(new ExportCategory, "category.xlsx");
    }

    public function import_excel()
    {
        try {
            Excel::import(new CategoryImport, request()->file('file'));
            session()->flash('success', 'File Excel berhasil di Impor.');
        } catch (\Exception $e) {
            session()->flash('error', 'Kesalahan mengimpor file Excel: ' . $e->getMessage());
```

```
        }

        return back();
    }

    public function export_pdf()
    {
        // Retrieve order details and customer information
        $categories = Category::all();
        $data['categories'] = $categories;
        $now = Carbon::now()->format('Y-m-d');
        $data['now'] = $now;

        // Generate the PDF
        $pdf = PDF::loadView('admin.report.category', compact('data'));

        // Set options if needed (e.g., page size, orientation)
        $pdf->setPaper('A4', 'potrait');

        // Download the PDF with a specific filename
        return $pdf->stream('category.pdf');
    }
}
```

HomeController

```
<?php

namespace App\Http\Controllers\admin;

use App\Charts\MonthlyUsersChart;
use App\Charts\PendapatanBulanan;
use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Order;
use App\Models\Product;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;

   class HomeController extends Controller
   {
       public function index(PendapatanBulanan $chart) {
       $paymentStatus = [1, 3, 4];

       // Calculate total earnings for this month
       $currentMonth = Carbon::now()->startOfMonth()->format('Y-m-d');
       $currentMonthEarnings = Order::where('payment_status', 2)
           ->where('updated_at', '>=', $currentMonth)
```

```
            ->sum('grand_total');


        // Calculate total earnings
        $totalEarnings = Order::where('payment_status', 2)->sum('grand_total');


        //Last Month
        $lastMonthStartDate = Carbon::now()->subMonth()->startOfMonth()->format('Y-m-d');
        $lastMonthEndDate = Carbon::now()->subMonth()->endOfMonth()->format('Y-m-d');


        $LastMonthEarnings = Order::where('payment_status', 2)
                        ->whereDate('updated_at','>=', $lastMonthStartDate)
                        ->whereDate('updated_at','<=', $lastMonthEndDate)
                        ->sum('grand_total');


        $categoryCount = Category::count();
        $userCount = User::where('role', 1)->count();
        $productCount = Product::count();
        $orderCount = Order::count();
        $orderUnpaidCount = Order::whereIn('payment_status', $paymentStatus)->count();
        $orderPaidCount = Order::where('payment_status', 2)->count();
        $latestUser = User::orderBy('id', 'DESC')->where('status',1)->where('role',1)->take(8)->get();
        $latestProducts = Product::orderBy('id', 'DESC')->where('status',1)->take(4)->get();
```

```
        $orders = Order::latest('orders.created_at')->select('orders.*','users.name','users.email');

        $orders = $orders->leftJoin('users', 'users.id','orders.user_id')->take(6)->get();


        $data = [
            'categoryCount' => $categoryCount,
            'userCount' => $userCount,
            'productCount' => $productCount,
            'orderCount' => $orderCount,
            'orderUnpaidCount' => $orderUnpaidCount,
            'orderPaidCount' => $orderPaidCount,
            'latestUser' => $latestUser,
            'latestProducts' => $latestProducts,
            'orders' => $orders,
            'currentMonthEarnings' => $currentMonthEarnings,
            'totalEarnings' => $totalEarnings,
            'LastMonthEarnings' => $LastMonthEarnings,
            'chart' => $chart->build(),
        ];


        return view('admin.dashboard', compact('data'));
    }


    public function logout(){
        Auth::guard('admin')->logout();
```

```
        return redirect()->route('admin.login');
    }
}
```

OrderController

```
<?php

namespace App\Http\Controllers;

use App\Models\Brand;
use App\Models\Category;
use App\Models\Product;
use App\Models\ProductRating;
use App\Models\subCategory;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ShopController extends Controller
{
    public function index(Request $request, $categorySlug = null,
$subCategorySlug = null) {
    $categorySelected='';
    $subCategorySelected='';
    $brandsArray = [];
```

```
    if (!empty($request->get('brand'))) {
        $brandsArray = explode(',', $request->get('brand'));
    }

    $categories = Category::orderBy('name', 'ASC')->with('sub_category')-
>where('status',1)->get();
    $brands = Brand::orderBy('name', 'ASC')->where('status',1)->get();
    $products = Product::where('status',1);

    //Apply filters here
    if(!empty($categorySlug)){
        $category = Category::where('slug', $categorySlug)->first();
        $products = $products->where('category_id', $category->id);
        $categorySelected = $category->id;
    }

    if(!empty($subCategorySlug)){
        $subCategory = subCategory::where('slug', $subCategorySlug)->first();
        $products = $products->where('sub_category_id', $subCategory->id);
        $subCategorySelected = $subCategory->id;
    }

    if (!empty($request->get('brand'))) {
        $brandsArray = explode(',', $request->get('brand'));
        $products = $products->whereIn('brand_id',$brandsArray);
```

```
    }


    if ($request->get('price_max') != '' && $request->get('price_min') != '') {
        if ($request->get('price_max') == 1000000) {
            $products = $products->whereBetween('price',[intval($request->get('price_min')),1000000]);
        } else {
            $products = $products->whereBetween('price',[intval($request->get('price_min')),intval($request->get('price_max'))]);
        }

    }


    if(!empty($request->get('search'))) {
        $products = $products->where('title','like','%'.$request->get('search').'%');

    }


    //$products = Product::orderBy('id', 'DESC')->where('status',1)->get();
    if ($request->get('sort') != '') {
        if ($request->get('sort') == 'latest') {
            $products = $products->orderBy('id','DESC');
        } else if($request->get('sort') == 'price_asc') {
            $products = $products->orderBy('price','ASC');
        } else {
            $products = $products->orderBy('price','DESC');
```

```
        }
    } else {
        $products = $products->orderBy('id','DESC');
    }

    $products = $products->paginate(6);

    $data['categories'] = $categories;
    $data['brands'] = $brands;
    $data['products'] = $products;
    $data['categorySelected'] = $categorySelected;
    $data['subCategorySelected'] = $subCategorySelected;
    $data['brandsArray'] = $brandsArray;
    $data['priceMax'] = (intval($request->get('price_max')) == 0) ? 1000000 :
$request->get('price_max');
    $data['priceMin'] = intval($request->get('price_min'));
    $data['sort'] = $request->get('sort');

    return view('front.shop', $data);
    }

    public function product($slug) {
        //$slug;
        $product = Product::where('slug', $slug)->with('product_images')->first();
        if ($product == null) {
```

```
            abort(404);
        }

        $relatedProducts = [];
        // Fetch Related Products

        if ($product->related_products != '') {
            $productArray = explode(',',$product->related_products);

            $relatedProducts = Product::WhereIn('id', $productArray)->where('status',1)->get();
        }

        $data['product'] = $product;
        $data['relatedProducts'] = $relatedProducts;

        return view('front.product',$data);
    }

    public function saveRating($id, Request $request){
        $validator = Validator::make($request->all(),[
            'username' => 'required|min:5',
            'email' => 'required|email',
            'comment' => 'required|min:10',
            'rating' => 'required'
```

```
]);

if ($validator->fails()){
    return response()->json([
        'status' => false,
        'errors' => $validator->errors()
    ]);
}

$count = ProductRating::where('email', $request->email)->count();
if ($count > 0) {
   session()->flash('error','Anda sudah menilai produk ini.');
   return response()->json([
        'status' => true,
        'alreadyRated' => true,
    ]);
}

$productRating = new ProductRating;
$productRating->product_id = $id;
$productRating->username = $request->username;
$productRating->email = $request->email;
$productRating->comment = $request->comment;
$productRating->rating = $request->rating;
```

```
        $productRating->status = 0;
        $productRating->save();

        session()->flash('success','Terima kasih atas penilaian Anda');

        return response()->json([
            'status' => true,
            'message' => 'Terima kasih atas penilaian Anda'
        ]);

    }
}
```

PagesController

```
<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;
use App\Models\Page;
use Illuminate\Support\Facades\Validator;
use Illuminate\Http\Request;

class PagesController extends Controller
{
```

```
public function index(Request $request) {
    $pages = Page::latest();

    if($request->keyword != ''){
        $pages = $pages->where('name','like','%'.$request->keyword.'%');
    }

    $pages = $pages->paginate(10);

    return view('admin.pages.list',[
        'pages' => $pages
    ]);
}

public function create() {
    return view('admin.pages.create');

}

public function store(Request $request) {
    $validator = Validator::make($request->all(),[
        'name'=>'required',
        'slug'=>'required'
    ]);
```

```
    if ($validator->fails()) {
        return response()->json([
            'status'=>false,
            'errors' =>$validator->errors()
        ]);
    }

    $page = new Page;
    $page->name = $request->name;
    $page->slug = $request->slug;
    $page->content = $request->content;
    $page->save();

    $message = 'Page added successfuly';

    session()->flash('success',$message);

    return response()->json([
        'status'=>true,
        'message' => $message
    ]);

}

public function edit(Request $request, $id) {
```

```
    $page = Page::find($id);

    if ($page == null) {
        session()->flash('error','Page Kosong');
        return redirect()->route('pages.index');
    }

    return view('admin.pages.edit',[
        'page' => $page
    ]);
}

public function update (Request $request, $id) {

    $page = Page::find($id);

    if ($page == null) {
        session()->flash('error','Page Kosong');
        return response()->json([
            'status' => true,
        ]);
    }

    $validator = Validator::make($request->all(),[
```

```
    'name'=>'required',
    'slug'=>'required'
]);

if ($validator->fails()) {
    return response()->json([
        'status'=>false,
        'errors' =>$validator->errors()
    ]);
}


$page->name = $request->name;
$page->slug = $request->slug;
$page->content = $request->content;
$page->save();

$message = 'Page Berhasil di Updated';

session()->flash('success',$message);

return response()->json([
    'status'=>true,
    'message' => $message
]);
```

```
    }

    public function destroy($id) {
        $page = Page::find($id);

        if ($page == null) {
            session()->flash('error','Page Kosong');
            return response()->json([
                'status' => true,
            ]);
        }

        $page->delete();

        $message = 'Page Berhasil di Hapus';

        session()->flash('success',$message);

        return response()->json([
            'status'=>true,
            'message' => $message
        ]);
    }
}
```

ProductController

```
<?php

namespace App\Http\Controllers\admin;

use App\Exports\ExportProduct;
use App\Http\Controllers\Controller;
use App\Imports\ProductImport;
use App\Models\Brand;
use App\Models\Category;
use App\Models\Product;
use App\Models\ProductImage;
use App\Models\subCategory;
use App\Models\TempImage;
use PDF;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Validator;
use Image;
use Maatwebsite\Excel\Facades\Excel;

class ProductController extends Controller
{
```

```
public function index(Request $request){
    $products = Product::oldest('id')->with('product_images');

    if($request->get('keyword') !=""){
        $products = $products->where('title','like','%'.$request->keyword.'%');
    }

    $products = $products->paginate();
    $data['products'] = $products;
    return view('admin.product.list',$data);
}

public function create(){
    $data=[];
    $categories = Category::orderBy('name','ASC')->get();
    $brands = Brand::orderBy('name','ASC')->get();
    $data['categories'] = $categories;
    $data['brands'] = $brands;
    return view('admin.product.create', $data);
}

public function store(Request $request){

    // dd($request->image_array);
    // exit();
```

```
$rules = [
    'title' => 'required',
    'slug' => 'required|unique:products',
    'price' => 'required|numeric',
    'sku' => 'required|unique:products',
    'track_qty' => 'required|in:Yes,No',
    'category' => 'required|numeric',
    'is_featured' => 'required|in:Yes,No',
];

if(!empty($request->track_qty) && $request->track_qty == 'Yes'){
    $rules['qty'] = 'required|numeric';
}

$validator = Validator::make($request->all(),$rules);

if($validator->passes()){
    $product = new Product;
    $product->title = $request->title;
    $product->slug = $request->slug;
    $product->description = $request->description;
    $product->price = $request->price;
    $product->compare_price = $request->compare_price;
    $product->sku = $request->sku;
    $product->barcode = $request->barcode;
```

```
        $product->track_qty = $request->track_qty;

        $product->qty = $request->qty;

        $product->status = $request->status;

        $product->category_id = $request->category;

        $product->sub_category_id = $request->sub_category;

        $product->brand_id = $request->brand;

        $product->is_featured = $request->is_featured;

        $product->shipping_returns = $request->shipping_returns;

        $product->short_description = $request->short_description;

        $product->related_products = (!empty($request->related_products)) ?
implode(',',$request->related_products) : '';

        $product->save();


        // Buat Save Image Gallery

        if(!empty($request->image_array)){

           foreach ($request->image_array as $temp_image_id){


               $tempImageInfo = TempImage::find($temp_image_id);

               $extArray = explode('.',$tempImageInfo->name);

               $ext = last($extArray); // like jpg,png,gif etc


               $productImage = new ProductImage();

               $productImage->product_id = $product->id;

               $productImage->image = 'NULL';

               $productImage->save();
```

```
            $imageName = $product->id.'-'.$productImage->id.'-'.time().'.'.$ext;
            $productImage->image = $imageName;
            $productImage->save();

            // Generate Product Thumbnail

            // Large Image
            $sourcePath = public_path().'/temp/'.$tempImageInfo->name;
            $destPath = public_path().'/uploads/product/large/'.$imageName;
            $image = Image::make($sourcePath);
            $image->resize(1400, null, function($constraint){
                $constraint->aspectRatio();
            });
            $image->save($destPath);

            // Small Image
            $destPath = public_path().'/uploads/product/small/'.$imageName;
            $image = Image::make($sourcePath);
            $image->fit(300,300);
            $image->save($destPath);

        }
    }
```

```
            $request->session()->flash('success','Produk Berhasil di Tambahkan');

            return response()->json([
                'status' => true,
                'message' => 'Produk Berhasil di Tambahkan'
            ]);

        }else{
            return response()->json([
                'status' => false,
                'errors' => $validator->errors()
            ]);
        }
    }

    public function edit($id, Request $request){

        $product = Product::find($id);

        if(empty($product)){
            //$request->session()->flash('error', 'Product not Found');
            return redirect()->route('products.index')->with('error', 'Produk Kosong');
        }

        // Fetch Product Image
```

```
        $productImages = ProductImage::where('product_id',$product->id)->get();


        $subCategories = subCategory::where('category_id',$product->category_id)
->get();


        $relatedProducts = [];

        // Fetch Related Products


        if ($product->related_products != '') {

            $productArray = explode(',',$product->related_products);


            $relatedProducts = Product::WhereIn('id', $productArray)
->with('product_images')->get();

        }


        $data=[];

        $categories = Category::orderBy('name','ASC')->get();

        $brands = Brand::orderBy('name','ASC')->get();

        $data['categories'] = $categories;

        $data['brands'] = $brands;

        $data['product'] = $product;

        $data['subCategories'] = $subCategories;

        $data['productImages'] = $productImages;

        $data['relatedProducts'] = $relatedProducts;


        return view('admin.product.edit', $data);
```

```
}

public function update($id, Request $request){

    $product = Product::find($id);

    $rules = [
        'title' => 'required',
        'slug' => 'required|unique:products,slug,'.$product->id.',id',
        'price' => 'required|numeric',
        'sku' => 'required|unique:products,sku,'.$product->id.',id',
        'track_qty' => 'required|in:Yes,No',
        'category' => 'required|numeric',
        'is_featured' => 'required|in:Yes,No',
    ];

    if(!empty($request->track_qty) && $request->track_qty == 'Yes'){
        $rules['qty'] = 'required|numeric';
    }

    $validator = Validator::make($request->all(),$rules);

    if($validator->passes()){

        $product->title = $request->title;
```

```
            $product->slug = $request->slug;
            $product->description = $request->description;
            $product->price = $request->price;
            $product->compare_price = $request->compare_price;
            $product->sku = $request->sku;
            $product->barcode = $request->barcode;
            $product->track_qty = $request->track_qty;
            $product->qty = $request->qty;
            $product->status = $request->status;
            $product->category_id = $request->category;
            $product->sub_category_id = $request->sub_category;
            $product->brand_id = $request->brand;
            $product->is_featured = $request->is_featured;
            $product->shipping_returns = $request->shipping_returns;
            $product->short_description = $request->short_description;
        $product->related_products = (!empty($request->related_products)) ?
implode(',',$request->related_products) : '';
            $product->save();


            $request->session()->flash('success','Produk Berhasil di Update');


            return response()->json([
              'status' => true,
              'message' => 'Produk Berhasil di Update'
            ]);
```

```
    }else{
        return response()->json([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function destroy($id, Request $request){
    $product = Product::find($id);

    if(empty($product)){
        $request->session()->flash('error','Produk Kosong');
        return response()->json([
            'status' => false,
            'notFound' => true
        ]);
    }

    $productImage = ProductImage::where('product_id', $id)->get();

    $productImages = ProductImage::where('product_id', $id)->get();

    if (!empty($productImages)) {
```

```
            foreach ($productImages as $productImage) {
                File::delete(public_path('uploads/product/large/' . $productImage->image));
                File::delete(public_path('uploads/product/small/' . $productImage->image));
            }

            ProductImage::where('product_id', $id)->delete();
        }

        $product->delete();

        $request->session()->flash('success','Product Berhasil di Hapus');

        return response()->json([
            'status' => true,
            'message' => 'Produk Berhasil di Hapus'
        ]);

    }

    public function getProducts(Request $request){

        $tempProduct = [];
        if ($request->term != "") {
            $products = Product::where('title','like','%'.$request->term.'%')->get();
```

```
        if ($products != null) {
          foreach ($products as $product) {
           $tempProduct[] = array('id' => $product->id, 'text' => $product->title);
          }
        }
    }

    return response()->json([
        'tags' => $tempProduct,
        'status' => true
    ]);
}

public function export_excel(){
   return Excel::download(new ExportProduct, "product.csv");
}

public function import_excel()
{
    try {
        Excel::import(new ProductImport, request()->file('file'));
        session()->flash('success', 'File Excel berhasil di Impor.');
    } catch (\Exception $e) {
        session()->flash('error', 'Kesalahan mengimpor file Excel: ' . $e-
```

```
>getMessage());
        }

        return back();
    }

    public function export_pdf()
    {
        // Retrieve order details and customer information
        $products = Product::all();
        foreach ($products as $product) {
        $product->description = strip_tags($product->description);
    }
        $data['products'] = $products;
        $now = Carbon::now()->format('Y-m-d');
        $data['now'] = $now;

        // Generate the PDF
        $pdf = PDF::loadView('admin.report.product', compact('data'));

        // Set options if needed (e.g., page size, orientation)
        $pdf->setPaper('A4', 'potrait');

        // Download the PDF with a specific filename
        return $pdf->stream('Product.pdf');
```

```
    }
}
```

ProductImageController

```
<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;
use App\Models\ProductImage;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;
use Image;

class ProductImageController extends Controller
{
    public function update(Request $request){

        $image = $request->image;
        $ext = $image->getClientOriginalExtension();
        $sourcePath = $image->getPathName();

        $productImage = new ProductImage();
        $productImage->product_id = $request->product_id;
        $productImage->image = 'NULL';
```

```
$productImage->save();

$imageName = $request->product_id.'-'.$productImage->id.'-'.time().'.'.$ext;
$productImage->image = $imageName;
$productImage->save();

// Large Image
$destPath = public_path().'/uploads/product/large/'.$imageName;
$image = Image::make($sourcePath);
$image->resize(1400, null, function($constraint){
   $constraint->aspectRatio();
   });
$image->save($destPath);

// Small Image
$destPath = public_path().'/uploads/product/small/'.$imageName;
$image = Image::make($sourcePath);
$image->fit(300,300);
$image->save($destPath);

return response()->json([
   'status' => true,
   'image_id' => $productImage->id,
   'imagePath' => asset('uploads/product/small/'.$productImage->image),
   'message' => 'Image Saved Successfully.'
```

```
    ]);
}

public function destroy(Request $request){
    $productImage = ProductImage::find($request->id);

    if(empty($productImage)){
        return response()->json([
        'status' => false,
        'message' => 'Gambar Kosong.'
    ]);
    }

    // Delete image from folder

    File::delete(public_path('uploads/product/large/'.$productImage->image));
    File::delete(public_path('uploads/product/small/'.$productImage->image));

    $productImage->delete();

    return response()->json([
        'status' => true,
        'message' => 'Gambar Berhasil di Hapus.'
    ]);
}
```

```
}
```

SettingController

```
<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;

class SettingController extends Controller
{
    public function ShowChangePasswordForm() {
        return view('admin.change-password');
    }

    public function processChangePassword(Request $request) {
        $validator = Validator::make($request->all(), [
            'old_password' => 'required',
            'new_password' => 'required|min:5',
            'confirm_password' => 'required|same:new_password',
```

```
]);

$id = Auth::guard('admin')->user()->id;

$admin = User::where('id',$id)->first();

if($validator->passes()) {

    if(!Hash::check($request->old_password, $admin->password)){
        session()->flash('error','Password lama Anda salah, coba lagi.');
        return response()->json([
            'status' => true,
        ]);
    }

    User::where('id',$id)->update([
        'password' => Hash::make($request->new_password)
    ]);

    session()->flash('success','Anda telah berhasil mengubah Password Anda');
    return response()->json([
        'status' => true,
    ]);
```

```
        } else {
            return response()->json([
                'status' => false,
                'errors' => $validator->errors()
            ]);
        }
    }
}
```

SubCategoryController

```
<?php

namespace App\Http\Controllers\admin;

use App\Exports\ExportSubCategory;
use App\Http\Controllers\Controller;
use App\Imports\SubCategoryImport;
use App\Models\Category;
use App\Models\subCategory;
use PDF;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Maatwebsite\Excel\Facades\Excel;
```

```
class SubCategoryController extends Controller

{


    public function index(Request $request){

        $subCategories = subCategory::select('sub_categories.*','categories.name as
categoryName')

                        ->oldest('sub_categories.id')

                        ->leftJoin('categories','categories.id','sub_categories.category_id');



        if(!empty($request->get('keyword'))){

            $subCategories = $subCategories-
>where('sub_categories.name','like','%'.$request->get('keyword').'%');

            $subCategories = $subCategories-
>orWhere('categories.name','like','%'.$request->get('keyword').'%');

        }

        $subCategories = $subCategories->paginate(10);



        return view('admin.sub_category.list',compact('subCategories'));

    }



    public function create(){

        $categories = Category::orderBy('name', 'ASC')->get();

        $data['categories'] = $categories;

        return view('admin.sub_category.create',$data);

    }
```

```
public function store(Request $request){
    $validator = Validator::make($request->all(),[
        'name' => 'required',
        'slug' => 'required|unique:sub_categories',
        'category' => 'required',
        'status' => 'required',
    ]);

    if ($validator->passes()){

        $subCategory = new subCategory();
        $subCategory->name = $request->name;
        $subCategory->slug = $request->slug;
        $subCategory->status = $request->status;
        $subCategory->showHome = $request->showHome;
        $subCategory->category_id = $request->category;
        $subCategory->save();

        $request->session()->flash('success','Sub Kategori Berhasil Dibuat.');

        return response([
            'status' => true,
            'message' => 'Sub Kategori Berhasil Dibuat.'
        ]);
```

```
    }else {
        return response([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function edit($id, Request $request){

    $subCategory = subCategory::find($id);
    if (empty($subCategory)) {
        $request->session()->flash('error','Catatan tidak di temukan');
        return redirect()->route('sub-categories.index');
    }

    $categories = Category::orderBy('name', 'ASC')->get();
    $data['categories'] = $categories;
    $data['subCategory'] = $subCategory;
    return view('admin.sub_category.edit',$data);
}

public function update($id, Request $request){

    $subCategory = subCategory::find($id);
```

```
 if (empty($subCategory)) {
        $request->session()->flash('error','Catatan tidak di temukan');
        return response([
            'status' => false,
            'notFound' =>true
        ]);
    }

$validator = Validator::make($request->all(),[
    'name' => 'required',
    'slug' => 'required|unique:sub_categories,slug,'.$subCategory->id.',id',
    'category' => 'required',
    'status' => 'required',
]);

if ($validator->passes()){

    $subCategory->name = $request->name;
    $subCategory->slug = $request->slug;
    $subCategory->status = $request->status;
    $subCategory->showHome = $request->showHome;
    $subCategory->category_id = $request->category;
    $subCategory->save();
```

```
        $request->session()->flash('success','Sub Kategori Berhasil Di Update.');

        return response([
            'status' => true,
            'message' => 'Sub Kategori Berhasil Di Update.'
        ]);

    }else {
        return response([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}

public function destroy($id, Request $request){
    $subCategory = subCategory::find($id);

     if (empty($subCategory)) {
            $request->session()->flash('error','Catatan tidak di Temukan');
            return response([
                'status' => false,
                'notFound' =>true
            ]);
        }
```

```
        $subCategory->delete();

        $request->session()->flash('success','Sub Kategori Berhasil Dihapus.');

        return response([
            'status' => true,
            'message' => 'Sub Kategori Berhasil Dihapus.'
        ]);
    }

    public function export_excel(){
      return Excel::download(new ExportSubCategory, "sub category.xlsx");
    }

    public function import_excel()
    {
        try {
            Excel::import(new SubCategoryImport, request()->file('file'));
            session()->flash('success', 'File Excel berhasil di Impor.');
        } catch (\Exception $e) {
            session()->flash('error', 'Kesalahan mengimpor file Excel: ' . $e->getMessage());
        }
```

```
        return back();
    }

    public function export_pdf()
    {
        // Retrieve order details and customer information
        $subCategories = subCategory::select('sub_categories.*','categories.name as categoryName')
                    ->leftJoin('categories','categories.id','sub_categories.category_id')
                    ->get();
        $data['subCategories'] = $subCategories;
        $now = Carbon::now()->format('Y-m-d');
        $data['now'] = $now;

        // Generate the PDF
        $pdf = PDF::loadView('admin.report.subCategory', compact('data'));

        // Set options if needed (e.g., page size, orientation)
        $pdf->setPaper('A4', 'potrait');

        // Download the PDF with a specific filename
        return $pdf->stream('Sub Category.pdf');
    }
}
```

UserController

```
<?php

namespace App\Http\Controllers\admin;

use App\Exports\ExportUser;
use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Hash;
use Illuminate\Http\Request;
use Maatwebsite\Excel\Facades\Excel;

class UserController extends Controller
{
    public function index(Request $request) {
        // $users = User::latest(); //mengurutkan data ke yg paling baru
        $users = User::oldest();

        if(!empty($request->get('keyword'))) {
            $users = $users->where('name','like','%'.$request->get('keyword').'%');
            $users = $users->Where('email','like','%'.$request->get('keyword').'%');
            $users = $users->orWhere('role','like','%'.$request->get('keyword').'%');

        }
```

```
    $users = $users->paginate(10);

    return view('admin.users.list',[
        'users' => $users
    ]);

}

public function create(Request $request) {
    return view('admin.users.create',[
    ]);
}

public function store(Request $request) {
    $validator = Validator::make($request->all(),[
        'name' => 'required',
        'password' => 'required|min:5',
        'email' => 'required|email|unique:users',
        'phone' => 'required',
    ]);

    if ($validator->passes()) {

        $user = new User;
```

```
        $user->name = $request->name;
        $user->email = $request->email;
        $user->phone = $request->phone;
        $user->status = $request->status;
        $user->password = Hash::make($request->password);
        $user->save();

        $message = 'User Berhasil di Tambahkan.';

        session()->flash('success', $message);

        return response()->json([
            'status' => true,
            'message' => $message
        ]);

    } else {
        return response()->json([
            'status' => false,
            'errors' => $validator->errors()
        ]);
    }
}
public function edit (Request $request, $id) {
    $user = User::find($id);
```

```
    if  ($user == null) {
        $message = 'User tidak di Temukan';
        session()->flash('error',$message);
        return redirect()->route('users.index');
    }

    return view ('admin.users.edit',[
        'user' =>$user
    ]);
}
public function update(Request $request, $id) {

    $user = User::find($id);

    if  ($user == null) {
        $message = 'User tidak di Temukan';
        session()->flash('error',$message);

        return response()->json([
            'status' => true,
            'message' => $message
        ]);

    }
```

```
$validator = Validator::make($request->all(),[
    'name' => 'required',
    'email' => 'required|email|unique:users,email,'.$id.',id',
    'phone' => 'required',
]);

if ($validator->passes()) {

    $user->name = $request->name;
    $user->email = $request->email;
    $user->phone = $request->phone;
    $user->status = $request->status;

    if ($request->password != '') {
        $user->password = Hash::make($request->password);
    }


    $user->save();

    $message = 'User Berhasil di Tambahkan.';

    session()->flash('success', $message);
```

```
            return response()->json([
                'status' => true,
                'message' => $message
            ]);

        } else {
            return response()->json([
                'status' => false,
                'errors' => $validator->errors()
            ]);
        }
    }

    public function destroy($id) {

        $user = User::find($id);

        if  ($user == null) {
            $message = 'User tidak di Temukan';
            session()->flash('error',$message);

            return response()->json([
                'status' => true,
                'message' => $message
            ]);
```

```
        }

        $user->delete();

        $message = 'User Berhasil di Hapus';
            session()->flash('success',$message);

            return response()->json([
                'status' => true,
                'message' => $message
            ]);
        }

    public function export_excel(){
      return Excel::download(new ExportUser, "User.xlsx");
    }

}
```

FrontController

```
<?php

namespace App\Http\Controllers;

use App\Mail\ContactEmail;
```

```
use Illuminate\Http\Request;
use App\Models\Product;
use App\Models\Wishlist;
use App\Models\Page;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Validator;

class FrontController extends Controller
{
   public function index() {

      $products = Product::where('is_featured', 'Yes')
                  ->orderBy('id','ASC')
                  ->where('status',1)
                  ->take(8)
                  ->get();

      $data['featuredProducts'] = $products;

      $latestProducts = Product::orderBy('id', 'DESC')
                  ->where('status',1)
                  ->take(8)
                  ->get();
```

```
    $data['latestProducts'] = $latestProducts;
    return view ('front.home', $data);
}

public function addToWishlist(Request $request) {

if (Auth::check() == false) {

    session(['url.intended' => url ()->previous()]);

    return response()->json([
        'status' => false
    ]);
}

$product = Product::where('id',$request->id)->first();

if ($product == null) {
    return response()->json([
        'status' => false, // Change to false because the product was not found
        'message' => 'Produk Tidak Ditemukan'
    ]);
}
```

```
    $wishlist = Wishlist::updateOrCreate(
        [
            'user_id' => Auth::user()->id,
            'product_id' => $request->id,
        ],
        [
            'user_id' => Auth::user()->id,
            'product_id' => $request->id,
        ]
    );


    if ($wishlist->wasRecentlyCreated) {
        // Wishlist item was created, product added successfully
        return response()->json([
            'status' => true,
            'message' => '<div class="alert alert-success"><strong>"'.$product->title.'"</strong> Berhasil ditambahkan ke Wishlist Anda</div>'
        ]);
    } else {
        // Wishlist item already exists, product is already in the wishlist
        return response()->json([
            'status' => false, // Change to false because the product is already in the wishlist
            'message' => '<div class="alert alert-info"><strong>"'.$product->title.'"</strong> Sudah ada di Wishlist</div>'
        ]);
```

```
    }
}

    public function page($slug) {
        $page = Page::where('slug',$slug)->first();
        if($page == null) {
            abort(404);
        }
        return view('front.page',[
            'page'=>$page
        ]);
        //dd($page);

    }

    public function sendContactEmail(Request $request) {
       $validator = Validator::make($request->all(),[
            'name' =>'required',
            'email' => 'required|email',
            'subject' => 'required|min:10'
        ]);

        if ($validator->passes()) {

            // Kirim Email
```

```
            $mailData = [
                'name' => $request->name,
                'email' => $request->email,
                'subject' => $request->subject,
                'message' => $request->message,
                'mail_subject' => 'Anda telah menerima email kontak'
            ];

            $admin = User::where('id',1)->first();

            Mail::to($admin->email)->send(new ContactEmail($mailData));

            session()->flash('success','Terima kasih telah menghubungi kami, kami akan segera menghubungi Anda.');

            return response()->json([
                'status' => true,
            ]);

        } else {
            return response()->json([
                'status' => false,
                'errors' => $validator->errors()
            ]);
```

```
        }
    }
}
```

ShopController

```
<?php

namespace App\Http\Controllers;

use App\Models\Brand;
use App\Models\Category;
use App\Models\Product;
use App\Models\ProductRating;
use App\Models\subCategory;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ShopController extends Controller
{
    public function index(Request $request, $categorySlug = null,
$subCategorySlug = null) {
    $categorySelected='';
    $subCategorySelected='';
    $brandsArray = [];
```

```
    if (!empty($request->get('brand'))) {
        $brandsArray = explode(',', $request->get('brand'));
    }

    $categories = Category::orderBy('name', 'ASC')->with('sub_category')-
>where('status',1)->get();
    $brands = Brand::orderBy('name', 'ASC')->where('status',1)->get();
    $products = Product::where('status',1);

    //Apply filters here
    if(!empty($categorySlug)){
        $category = Category::where('slug', $categorySlug)->first();
        $products = $products->where('category_id', $category->id);
        $categorySelected = $category->id;
    }

    if(!empty($subCategorySlug)){
        $subCategory = subCategory::where('slug', $subCategorySlug)->first();
        $products = $products->where('sub_category_id', $subCategory->id);
        $subCategorySelected = $subCategory->id;
    }

    if (!empty($request->get('brand'))) {
        $brandsArray = explode(',', $request->get('brand'));
        $products = $products->whereIn('brand_id',$brandsArray);
```

```
    }

    if ($request->get('price_max') != '' && $request->get('price_min') != '') {
        if ($request->get('price_max') == 1000000) {
            $products = $products->whereBetween('price',[intval($request->get('price_min')),1000000]);
        } else {
            $products = $products->whereBetween('price',[intval($request->get('price_min')),intval($request->get('price_max'))]);
        }

    }

    if(!empty($request->get('search'))) {
        $products = $products->where('title','like','%'.$request->get('search').'%');

    }

    //$products = Product::orderBy('id', 'DESC')->where('status',1)->get();
    if ($request->get('sort') != '') {
        if ($request->get('sort') == 'latest') {
            $products = $products->orderBy('id','DESC');
        } else if($request->get('sort') == 'price_asc') {
            $products = $products->orderBy('price','ASC');
        } else {
            $products = $products->orderBy('price','DESC');
```

```
        }
    } else {
        $products = $products->orderBy('id','DESC');
    }

    $products = $products->paginate(6);

    $data['categories'] = $categories;
    $data['brands'] = $brands;
    $data['products'] = $products;
    $data['categorySelected'] = $categorySelected;
    $data['subCategorySelected'] = $subCategorySelected;
    $data['brandsArray'] = $brandsArray;
    $data['priceMax'] = (intval($request->get('price_max')) == 0) ? 1000000 :
$request->get('price_max');
    $data['priceMin'] = intval($request->get('price_min'));
    $data['sort'] = $request->get('sort');

    return view('front.shop', $data);
    }

    public function product($slug) {
        //$slug;
        $product = Product::where('slug', $slug)->with('product_images')->first();
        if ($product == null) {
```

```
            abort(404);
        }

        $relatedProducts = [];
        // Fetch Related Products

        if ($product->related_products != '') {
            $productArray = explode(',',$product->related_products);

            $relatedProducts = Product::WhereIn('id', $productArray)->where('status',1)->get();
        }

        $data['product'] = $product;
        $data['relatedProducts'] = $relatedProducts;

        return view('front.product',$data);
    }

    public function saveRating($id, Request $request){
        $validator = Validator::make($request->all(),[
            'username' => 'required|min:5',
            'email' => 'required|email',
            'comment' => 'required|min:10',
            'rating' => 'required'
```

```
]);

if ($validator->fails()){
    return response()->json([
        'status' => false,
        'errors' => $validator->errors()
    ]);
}

$count = ProductRating::where('email', $request->email)->count();
if ($count > 0) {
   session()->flash('error','Anda sudah menilai produk ini.');
   return response()->json([
        'status' => true,
        'alreadyRated' => true,
    ]);
}

$productRating = new ProductRating;
$productRating->product_id = $id;
$productRating->username = $request->username;
$productRating->email = $request->email;
$productRating->comment = $request->comment;
$productRating->rating = $request->rating;
```

```
        $productRating->status = 0;
        $productRating->save();

        session()->flash('success','Terima kasih atas penilaian Anda');

        return response()->json([
            'status' => true,
            'message' => 'Terima kasih atas penilaian Anda'
        ]);

    }
}
```