<?php

namespace App\Http\Controllers\Api;

use App\Models\User;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\Validator;

use App\Notifications\EmailVerificationNotification;

class AuthController extends Controller

{

   /\*\*

     \* Create a new AuthController instance.

     \*

     \* @return void

     \*/

    public function \_\_construct()

    {

        $this->middleware('auth:api', ['except' => ['login', 'register']]);

    }

    /\*\*

     \* Get a JWT via given credentials.

     \*

     \* @return \Illuminate\Http\JsonResponse

     \*/

//Login

    public function login(Request $request)

    {

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

            'email'    => 'required|string|email',

            'password' => 'required|string|min:6',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        if (! $token = auth()->attempt($validator->validated())) {

            return response()->json(['error' => 'Unauthorized'], 401);

        }

        return $this->createNewToken($token);

    }

    /\*\*

     \* Register a User.

     \*

     \* @return \Illuminate\Http\JsonResponse

     \*/

//Register

    public function register(Request $request)

    {

        //dd($request);

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

            'type'       => 'sometimes|required|integer|in:0,1',   // allow customer or seller only

            'first\_name' => 'required|string|between:2,100',

            'last\_name'  => 'nullable|string|between:2,100',

            'email'      => 'required|max:100|unique:users',

            'phone'      => 'required|string|min:6|unique:users',

            'password'   => 'required|string|confirmed|min:6',

        ]);

        //dd($validator->validated());

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $user = User::create(array\_merge(

            $validator->validated(),

            ['password' => bcrypt($request->password)]

        ));

        $user->refresh();

        $user->notify(new EmailVerificationNotification());

        return response()->json([

            'message' => 'User successfully registered',

            'user' => $user

        ], 201);

    }

    /\*\*

     \* Log the user out (Invalidate the token).

     \*

     \* @return \Illuminate\Http\JsonResponse

     \*/

//LogOut

    public function logout()

    {

        auth()->logout();

        return response()->json(['message' => 'User successfully signed out']);

    }

       protected function createNewToken($token)

    {

        return response()->json([

            'access\_token' => $token,

            'token\_type' => 'bearer',

            'expires\_in' => auth()->factory()->getTTL(),

            'user' => auth()->user()

        ]);

    }

//Update User Account

    public function update(Request $request){

        $user = Auth::user();

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

            'first\_name' => 'sometimes|string|max:255',

            'last\_name' => 'sometimes|string|max:255',

            'email' => 'sometimes|string|email|max:255|unique:users,email,'.$user->id,

            'email' => 'sometimes|string|max:20|unique:users,phone,'.$user->id,

            'current\_password' => 'sometimes|string|min:6',

            'password' => 'nullable|string|min:6|confirmed',

        ]);

        if ($validator->fails()) {

            return response()->json(['error' => $validator->errors()], 422);

        }

        if ($request->filled('first\_name')) {

            $user->update(['first\_name' => $request->input('first\_name')]);

        }

        if ($request->filled('last\_name')) {

            $user->update(['last\_name' => $request->input('last\_name')]);

        }

        if ($request->filled('email')) {

            $user->update(['email' => $request->input('email')]);

        }

        if ($request->filled('phone')) {

            $user->update(['phone' => $request->input('phone')]);

        }

        // Verify user's current password

        if (!Hash::check($request->input('current\_password'), $user->password)) {

            return response()->json(['error' => 'Current password is incorrect.'], 422);

        }

        if ($request->filled('password')) {

            $user->update(['password' => bcrypt($request->input('password'))]);

        }

        return response()->json(['message' => 'User account updated successfully.'], 200);

    }

}

CheckOut

<?php

namespace App\Http\Controllers;

use App\Models\Cart;

use App\Models\User;

use App\Models\Order;

use App\Mail\OrderPlaced;

use Illuminate\Http\Request;

use App\Mail\AdminOrderPlaced;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Mail;

use App\Http\Resources\CheckOutResource;

use Illuminate\Support\Facades\Validator;

class CheckOutController extends Controller

{

    public function checkout(Request $request)

    {

        $cart = Cart::where('user\_id', $request->user()->id)->get();

        if ($cart->isEmpty()) {

            return response()->json(['message' => 'Cart is empty'], 400);

        }

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

            'first\_name' => 'required|string|between:2,100',

            'last\_name'  => 'nullable|string|between:2,100',

            'phone'      => 'required|string|min:6|unique:users',

            'address' => ['required', 'string'],

            'company' => ['nullable', 'string'],

            'total' => ['nullable', 'string'],

            'subtotal' => ['nullable', 'string'],

            'shipingCost' => ['nullable', 'string'],

            'payment' => ['required', 'string', 'in:0,1,2'],

            'notes' => ['nullable', 'string'],

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $order = new Order();

        $order->user\_id = Auth::id();

        $order->product\_id = $cart->first()->product\_id;

        $order->fill($validator->validated());

        $order->save();

    // Send email to the user.

    Mail::to($request->user()->email)->send(new OrderPlaced($order,$cart));

    // Send email to the admin.

    $user = User::find($order->user\_id);

        if ($user->type !== 2) {

            $admins = User::where('type', 2)->get();

            foreach ($admins as $admin) {

                Mail::to($admin->email)->send(new AdminOrderPlaced($order,$cart));

            }

        }

    Cart::where('user\_id', $request->user()->id)->delete();

        return [(new CheckOutResource($order))];

    }

}

<?php

namespace App\Http\Controllers;

use App\Models\Product;

use App\Traits\MediaTrait;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Http\Resources\ProductResource;

use App\Models\User;

use Illuminate\Support\Facades\Validator;

class ProductController extends Controller

{

    use MediaTrait;

//Add Product

    public function store(Request $request)

    {

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

            'title'    => 'required',

            'newPrice' => 'required',

            'oldPrice'    => 'nullable',

            'offer'    => 'sometimes',

            'category'    => 'required',

            'color'    => 'required',

            'size'    => 'required',

            'abstract'    => 'required',

            'featuer'    => 'required',

            'pin\_code'    => 'sometimes|max:3|unique:products',

            'description'    => 'required',

            'videos'    => 'required',

            'is\_special'=> 'required',

            'is\_complete'=> 'required',

            'images'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_1'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_2'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_3'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_4'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_5'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

            'image\_6'    => 'sometimes|file|image|mimes:jpg,gif,png,webp',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $product = new Product();

        $product->user\_id= Auth::id();

        $product->fill($validator->validated());

        $product->save();

        $this->handleRequestMediaFiles($product, $request);

        $product = Product::with('media')

            ->withCount('reviews')

            ->findOrFail($product->id);

        return new ProductResource($product);

    }

//Get One Product

    public function show($id)

    {

        // get the product with media images and reviews count

        $product = Product::with('media', 'reviews', 'reviews.user')

            ->withCount('reviews')

            ->rating()

            ->findOrFail($id);

        return new ProductResource($product);

    }

//Show All Products

    public function index(Request $request)

    {

        $products = Product::with('media')

            ->withCount('reviews')

            ->rating()

//New Arrival Products

            ->when($request->query('new\_arrival'), function ($query) {

                $query

                    ->where('created\_at', '>=', now()->subMonth())

                    ->orderBy('id', 'DESC');

            })

//Top Rated Products

            ->when($request->query('top\_rated'), function ($query) {

                $query

                    ->having('rating', '>', 0)

                    ->orderBy('rating', 'DESC');

            })

//Special Products

            ->when($request->query('is\_special'), function ($query) {

                $query

                    ->where('is\_special', '=', 1);

            })->get();

        return ProductResource::collection($products);

    }

//Get Seller Products

    public function sellerProducts(Request $request, $id)

    {

        if ($id != auth()->user()->id) {

            return response()->json(['error' => 'Unauthorized'], 401);

        }

        $products = Product::with('media')

            ->withCount('reviews')

            ->rating()

            ->where('user\_id', $id)

            ->when($request->query('new\_arrival'), function ($query) {

                $query

                    ->where('created\_at', '>=', now()->subMonth())

                    ->orderBy('id', 'DESC');

            })

            ->when($request->query('top\_rated'), function ($query) {

                $query

                    ->having('rating', '>', 0)

                    ->orderBy('rating', 'DESC');

            })

            ->when($request->query('is\_special'), function ($query) {

                $query

                    ->where('is\_special', '=', 1);

            })

            ->get();

        return ProductResource::collection($products);

    }

//Related Products

    public function relatedProducts(Request $request, $id)

    {

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

        $products = Product::with('media')

            ->withCount('reviews')

            ->rating()

            ->whereNot('id', $product->id)

            ->where('category', $product->category)

            ->when($request->query('new\_arrival'), function ($query) {

                $query

                    ->where('created\_at', '>=', now()->subMonth())

                    ->orderBy('id', 'DESC');

            })

            ->when($request->query('top\_rated'), function ($query) {

                $query

                    ->having('rating', '>', 0)

                    ->orderBy('rating', 'DESC');

            })

            ->when($request->query('is\_special'), function ($query) {

                $query

                    ->where('is\_special', '=', 1);

            })

            ->get();

        return ProductResource::collection($products);

    }

//Delete Product

public function delete($id){

    $product= Product::find($id);

    $product->delete($id);

    return response()->json('deleted');

}

//Update Product

public function update(Request $request)

{

        $product = Product::find($request->id);

        if (!$product) {

            return response()->json(['error' => 'Product not found'], 404);

        }

        $product->user\_id= Auth::id();

        $product->update($request->all());

        $product->save();

        $this->handleRequestMediaFiles($product, $request);

        return["updated Successfully"];

}

}

Review

<?php

namespace App\Http\Controllers;

use App\Models\Review;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Validator;

class ReviewController extends Controller

{

//Add Review

    public function store(Request $request)

    {

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

            'product\_id'    => ['required', 'integer', 'exists:products,id'],

            'star'    => ['required', 'integer', 'min:1', 'max:5'],

            'review' => ['required', 'string'],

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $review = new Review();

        $review->fill($validator->validated());

        $review->user\_id = Auth::id();

        $review->save();

        return response($review, 200, ["added successfully"]);

    }

//Delete Review

    public function delete($id){

        $product= Auth::user()->reviews()->findOrFail($id);

        $product->delete($id);

        return response()->json('Review Deleted');

    }

}

Customer Middleware

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

class IsCustomer

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next

     \* @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse

     \*/

    public function handle(Request $request, Closure $next)

    {

        if ($request->user() && ( $request->user()->type === 0)) {

            return $next($request);

        }

        return $request->expectsJson()

            ? response([

                'message' => 'Access Denied.',

            ], 403)

            : abort(403, 'Access Denied.');

    }

}

Seller Middleware

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

class IsSeller

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next

     \* @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse

     \*/

    public function handle(Request $request, Closure $next, ...$guards)

    {

        if ($request->user() && ( $request->user()->type === 1)) {

            return $next($request);

        }

        return $request->expectsJson()

            ? response([

                'message' => 'Access Denied.',

            ], 403)

            : abort(403, 'Access Denied.');

    }

}

CheckOut Resource

<?php

namespace App\Http\Resources;

use App\Models\User;

use App\Http\Resources\UserResource;

use Illuminate\Support\Facades\Auth;

use App\Http\Resources\ProductResource;

use App\Models\Product;

use Illuminate\Http\Resources\Json\JsonResource;

class CheckOutResource extends JsonResource

{

    private $paymentNames = [

        0 => 'Paypal',

        1 => 'Cash',

        2 => 'Bank',

    ];

    public function toArray($request)

    {

        $paymentName = $this->paymentNames[$this->payment];

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

        return [

            'message' => 'Order Place Successfully',

            'id' =>  $this->id,

            'total' => $this->total,

            'address' => $this->address,

            'payment' => [

                'code' => $this->payment,

                'name' => $paymentName,

            ],

            'company' => $this->company,

            'first\_name' => $this->first\_name,

            'last\_name' => $this->last\_name,

            'phone' => $this->phone,

            'subtotal' => $this->subtotal,

            'shipingCost' => $this->shipingCost,

            'user' => [

                'id' =>  $user->id,

                'email' => $user->email,

            ],

        ];

    }

}

Product Resource

<?php

namespace App\Http\Resources;

use App\Http\Resources\ReviewResource;

use Illuminate\Http\Resources\Json\JsonResource;

class ProductResource extends JsonResource

{

    /\*\*

     \* The "data" wrapper that should be applied.

     \*

     \* @var string|null

     \*/

    public static $wrap = null;

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable

     \*/

    public function toArray($request)

    {

        return [

            'id' =>  $this->id,

            'title' => $this->title,

            'newPrice' => $this->newPrice,

            'oldPrice' => $this->oldPrice,

            'offer' => $this->offer,

            'reviews\_count' => $this->reviews\_count,

            'rating' => $this->rating,

            'is\_complete' => $this->is\_complete,

            'is\_special' => $this->is\_special,

            'details' => [

                'abstract' => $this->abstract,

                'pin\_code' => $this->pin\_code,

                'description' => $this->description,

                'videos' => [$this->videos],

                'video\_html' => $this->when($this->videos, function () {

                    $query\_str = parse\_url($this->videos, PHP\_URL\_QUERY);

                    if (! $query\_str) {

                        return null;

                    }

                    parse\_str($query\_str, $params);

                    if (! isset($params['v'])) {

                        return null;

                    }

                    return '<iframe width="560" height="315" src="https://www.youtube.com/embed/' . $params['v'] . '" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>';

                }, null),

                'colors' => $this->when($this->color, function () {

                    return collect(explode(',', $this->color))

                        ->filter(fn ($item) => trim($item))

                        ->map(fn ($item) => ucfirst(trim($item)))

                        ->values();

                }, null),

                'featuers' => $this->when($this->featuer, function () {

                    return collect(explode(',', $this->featuer))

                        ->filter(fn ($item) => trim($item))

                        ->map(fn ($item) => ucfirst(trim($item)))

                        ->values();

                }, null),

                'category' => $this->when($this->category, function () {

                    return collect(explode(',', $this->category))

                        ->filter(fn ($item) => trim($item))

                        ->map(fn ($item) => ucfirst(trim($item)))

                        ->values();

                }, null),

                'sizes' => $this->when($this->size, function () {

                    return collect(explode(',', $this->size))

                        ->filter(fn ($item) => trim($item))

                        ->map(fn ($item) => strtoupper(trim($item)))

                        ->values();

                }, null),

                'images' => $this->whenLoaded('media', function () {

                    $images = [];

                    if ($this->hasMedia('image')) {

                        $images[] =[

                            $this?->getFirstMediaUrl('image')

                        ];

                        for ($i = 2; $i <= 6; $i++) {

                            $media = $this?->getFirstMedia('images', ['form\_key' => 'image\_' . $i]);

                            if (! $media) {

                                continue;

                            }

                            $images[] = $media?->getUrl();

                        }

                    }

                    return $images;

                }),

                'reviews' => ReviewResource::collection($this->whenLoaded('reviews')),

            ],

        ];

    }

}

Review Resource

<?php

namespace App\Http\Resources;

use App\Http\Resources\UserResource;

use Illuminate\Http\Resources\Json\JsonResource;

class ReviewResource extends JsonResource

{

    /\*\*

     \* The "data" wrapper that should be applied.

     \*

     \* @var string|null

     \*/

    public static $wrap = null;

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable

     \*/

    public function toArray($request)

    {

        return [

            'id' =>  $this->id,

            'star' => (int) $this->star,

            'review' => $this->review,

            'user' => new UserResource($this->whenLoaded('user')),

            'product' => $this->whenLoaded('product'),

        ];

    }

}

Order Model

<?php

namespace App\Models;

use App\Models\User;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use App\Models\Cart;

class Order extends Model

{

    protected $fillable = [

        'total',

        'address',

        'company',

        'payment',

        'user\_id',

        'first\_name',

        'last\_name',

        'phone',

        'subtotal',

        'shipingCost',

        'paymentNames',

    ];

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    public function product()

    {

        return $this->belongsTo(Product::class);

    }

    public function cart()

{

    return $this->belongsTo(Cart::class);

}

}

Product Model

<?php

namespace App\Models;

use App\Models\Review;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Spatie\Image\Manipulations;

use Spatie\MediaLibrary\HasMedia;

use Spatie\MediaLibrary\InteractsWithMedia;

use Spatie\MediaLibrary\MediaCollections\File;

use Spatie\MediaLibrary\MediaCollections\Models\Media;

class Product extends Model implements HasMedia

{

    use InteractsWithMedia;

    protected $fillable = [

        'title','newPrice','oldPrice','offer','category','color','size'  , 'abstract','featuer','pin\_code','description','videos','is\_complete','is\_special','user\_id'

    ];

    public function reviews()

    {

        return $this->hasMany(Review::class);

    }

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    public function scopeRating($query)

    {

        $query->addSelect(\DB::raw('(

            SELECT sum(reviews.star) / count(\*) FROM reviews where reviews.product\_id = products.id LIMIT 1

        ) AS rating'));

    }

    /\*\*

     \* Returns a list of the allowed files to be uploaded.

     \*

     \* @return array

     \*/

    public static function formRequestFileKeys(): array

    {

        return [

            'image' => [

                'image\_1',

            ],

            'images' => [

                'image\_2',

                'image\_3',

                'image\_4',

                'image\_5',

                'image\_6',

            ],

        ];

    }

    public function registerMediaCollections(): void

    {

        $this

            ->addMediaCollection('images')

            ->acceptsFile(function (File $file) {

                return in\_array($file->mimeType, config('media-library.allowed\_image\_types'));

            })

            ->useFallbackUrl(asset('assets/img/no-image.jpg'));

        $this

            ->addMediaCollection('image')

            ->singleFile()

            ->acceptsFile(function (File $file) {

                return in\_array($file->mimeType, config('media-library.allowed\_image\_types'));

            })

            ->useFallbackUrl(asset('assets/img/no-image.jpg'));

    }

    public function registerMediaConversions(Media $media = null): void

    {

        foreach (['50', '200'] as $size) {

            $this->addMediaConversion($size)

                ->performOnCollections('images', 'image')

                ->format(Manipulations::FORMAT\_JPG)

                ->quality(90)

                ->fit(Manipulations::FIT\_CROP, $size, $size)

                ->optimize()

                ->{$size == 200 ? 'nonQueued' : 'queued'}();

        }

        $this->addMediaConversion('large')

            ->performOnCollections('images', 'image')

            ->format(Manipulations::FORMAT\_JPG)

            ->quality(90)

            // ->width(1200)

            ->fit(Manipulations::FIT\_CROP, 1200, ceil(1200 / 16 \* 9))

            ->optimize();

        $this->addMediaConversion('medium')

            ->performOnCollections('images', 'image')

            ->format(Manipulations::FORMAT\_JPG)

            ->quality(90)

            ->width(640)

            // ->fit(Manipulations::FIT\_CROP, 640, ceil(640/16\*9))

            ->optimize();

        $this->addMediaConversion('small')

            ->performOnCollections('images', 'image')

            ->format(Manipulations::FORMAT\_JPG)

            ->quality(90)

            ->fit(Manipulations::FIT\_CROP, 360, ceil(360 / 4 \* 3))

            // ->width(360)

            ->optimize();

    }

    public function cart()

    {

        return $this->hasOne(Cart::class);

    }

    public function wishlist()

    {

        return $this->hasOne(WishList::class);

    }

    public function order()

    {

        return $this->hasOne(Order::class);

    }

    public function orderItem()

    {

        return $this->hasOne(OrderItem::class);

    }

}

Review Model

<?php

namespace App\Models;

use App\Models\Product;

use App\Models\User;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Review extends Model

{

    use HasFactory;

    protected $fillable = [

        'review',

        'star',

        'product\_id',

        'user\_id',

    ];

    public function product()

    {

        return $this->belongsTo(Product::class);

    }

    public function user()

    {

        return $this->belongsTo(User::class);

    }

}

User Model

<?php

namespace App\Models;

use App\Models\Order;

use App\Models\Review;

use Ichtrojan\Otp\Models\Otp;

use Laravel\Sanctum\HasApiTokens;

use Tymon\JWTAuth\Contracts\JWTSubject;

use Illuminate\Notifications\Notifiable;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable implements JWTSubject

{

    use HasApiTokens;

    use HasFactory;

    use Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'type',

        'first\_name',

        'last\_name',

        'email',

        'phone',

        'password',

        'email\_verified\_at',

        'varifed',

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

        'oauth\_provider',

        'oauth\_id',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

        'type' => 'integer',

    ];

    protected $appends = [

        'type\_text'

    ];

    public function getJWTIdentifier()

    {

        return $this->getKey();

    }

    public function getJWTCustomClaims()

    {

        return [];

    }

    public function getTypeTextAttribute(): string

    {

        return match ($this->attributes['type']) {

            1       => 'Seller',

            2       => 'Admin',

            default => 'Customer',

        };

    }

    public function reviews()

    {

        return $this->hasMany(Review::class);

    }

    public function products()

    {

        return $this->hasMany(Product::class);

    }

    public function cart()

    {

        return $this->hasMany(Cart::class);

    }

    public function wishlist()

    {

        return $this->hasMany(WishList::class);

    }

    public function order()

    {

        return $this->hasMany(Order::class);

    }

}

Mail send to Admin after CheckOut

<?php

namespace App\Mail;

use App\Models\Order;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Queue\SerializesModels;

class AdminOrderPlaced extends Mailable

{

    use Queueable, SerializesModels;

    public $order;

    public $cart;

    /\*\*

     \* Create a new message instance.

     \*

     \* @param Order $order

     \*/

    public function \_\_construct(Order $order, $cart)

    {

        $this->order = $order;

        $this->cart = $cart;

    }

    /\*\*

     \* Get the message envelope.

     \*

     \* @return \Illuminate\Mail\Mailables\Envelope

     \*/

    public function build()

    {

        return $this->subject('Admin Order Placed')

                    ->view('Mails.admin\_order\_placed')

                    ->with([

                        'order' => $this->order,

                        'cart' => $this->cart,

                    ]);

    }

    /\*\*

     \* Get the attachments for the message.

     \*

     \* @return array

     \*/

    public function attachments()

    {

        return [];

    }

}

Email format

Dear Admin,<br>

A new order has been placed by {{ $order->first\_name }} {{ $order->last\_name }} with {{ $order->user->email }} with the following details:<br>

@foreach ($cart as $item)

    Product Name: {{ $item->product->title }}<br>

    Price: {{ $item->product->newPrice }}$<br>

    Quantity: {{ $item->quantity }}<br>

    Color: {{ $item->product->color }}<br>

    Size: {{ $item->product->size }}<br>

    Category: {{ $item->product->category }}<br>

@endforeach

Total price: {{ $order->total }}$ EGP <br>

Payment Method: {{ $order->payment == 0 ? 'Paypal' : ($order->payment == 1 ? 'Cash' : 'Bank') }}<br>Address: {{ $order->address }}<br>Thank you for your attention.

Best regards

Mail send to Customer after CheckOut

<?php

namespace App\Mail;

use App\Models\Order;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Queue\SerializesModels;

class OrderPlaced extends Mailable

{

    use Queueable, SerializesModels;

    public $order;

    public $total;

    public $cart;

    /\*\*

     \* Create a new message instance.

     \*

     \* @param Order $order

     \*/

    public function \_\_construct(Order $order, $cart)

    {

        $this->order = $order;

        $this->total = $order->total;

        $this->cart = $cart;

    }

    /\*\*

     \* Get the message envelope.

     \*

     \* @return \Illuminate\Mail\Mailables\Envelope

     \*/

    public function envelope()

    {

        return new Envelope(

            subject: 'Order Placed',

        );

    }

    /\*\*

     \* Build the message.

     \*

     \* @return $this

     \*/

    public function build()

    {

        return $this->view('Mails.OrderPlaced');

    }

}

Email format

Dear {{ $order->first\_name }},<br>

@foreach ($cart as $item)

    Product Name: {{ $item->product->title }}<br>

    Price: {{ $item->product->newPrice }}$<br>

@endforeach

Total price: {{ $order->total }}$ EGP <br>

Payment Method: {{ $order->payment == 0 ? 'Paypal' : ($order->payment == 1 ? 'Cash' : 'Bank') }}<br>Address: {{ $order->address }}<br>Address: {{ $order->address }}<br>

Phone: {{ $order->phone }}<br>

Shiping Cost: {{ $order->shipingCost }}$<br>

Thanks again for your order!

Routes

<?php

use App\Http\Controllers\Api\AuthController;

use App\Http\Controllers\Auth\EmailVerificationController;

use App\Http\Controllers\CartController;

use App\Http\Controllers\CheckOutController;

use App\Http\Controllers\ContactController;

use App\Http\Controllers\WishListController;

use App\Http\Controllers\ProductController;

use App\Http\Controllers\OrderToFinishController;

use Spatie\MediaLibrary\MediaCollections\Models\Media;

use App\Http\Controllers\ReviewController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {

    return $request->user();

});

Route::group([

    'middleware' => 'api',

    'prefix' => 'auth'

], function ($router) {

    Route::post('/login', [AuthController::class, 'login']);

    Route::post('/register', [AuthController::class, 'register']);

    Route::post('/logout', [AuthController::class, 'logout']);

    Route::post('/updateAccount', [AuthController::class, 'update']);

    //EmailVerification

    Route::post('/email\_verification', [EmailVerificationController::class, 'email\_verification']);

    Route::get('/sendEmailVerification', [EmailVerificationController::class, 'sendEmailVerification']);

});

#################################################################################

Route::middleware('auth')->post('/reviews', [ReviewController::class, 'store']);

Route::middleware('auth')->delete('/delete/{id}', [ReviewController::class, 'delete']);

#################################################################################

Route::group([

    'prefix' => 'seller',

         'middleware' => ['api', 'auth', 'is.seller'],

     ], function () {

        Route::post('/products', [ProductController::class, 'store']);

        Route::post('/products/{id}', [ProductController::class, 'update']);

        Route::delete('/products/{id}', [ProductController::class, 'delete']);

        Route::get('/{id}/products', [ProductController::class, 'sellerProducts']);

        Route::get('/{id}/products', [ProductController::class, 'sellerProducts']);

        Route::delete('/media/{id}', function (Request $request, int $id) {

        $media = Media::findOrFail($id);

        $media->delete();

    return [

        'message' => \_\_('The file has been deleted successfully.')

    ];

})->name('api.media.delete');

    });

#################################################################################

Route::group([

    'prefix' => 'customer',

         'middleware' => ['api', 'auth', 'is.customer'],

     ], function () {

        Route::post('/cart', [CartController::class, 'store']);

        Route::get('/cart', [CartController::class, 'index']);

        Route::delete('/cart/{id}', [CartController::class, 'delete']);

#################################################################################

        Route::post('/wishlist', [WishListController::class, 'store']);

        Route::get('/wishlist', [WishListController::class, 'index']);

        Route::delete('/wishlist/{id}', [WishListController::class, 'delete']);

        Route::post('/orders', [CheckOutController::class, 'checkout']);

    });

#################################################################################

Route::get('/products', [ProductController::class, 'index']);

Route::get('/products/{id}', [ProductController::class, 'show']);

Route::get('/products/{id}/relatedProducts', [ProductController::class, 'relatedProducts']);

//Mail Route

Route::post('/contact', [ContactController::class, 'submit']);

Route::post('/ordertofinish', [OrderToFinishController::class, 'send']);

Cart Controller

<?php

namespace App\Http\Controllers;

use App\Http\Resources\CartResource;

use App\Models\Cart;

use App\Models\Product;

use App\Models\Shop;

use App\Traits\MediaTrait;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Validator;

class CartController extends Controller

{

    public function store(Request $request)

    {

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

            'product\_id'    => 'required|exists:products,id',

            'quantity'    => 'required|integer|min:1',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        Auth::user()->cart()->updateOrCreate(

            [

                'product\_id' => $request->product\_id,

            ],

            $validator->validated()

        );

        $cartItems = Auth::user()->cart()

            ->with('product', 'product.media')

            ->get();

        return CartResource::collection($cartItems);

    }

    public function index()

    {

        $cartItems = Auth::user()->cart()

            ->with('product', 'product.media')

            ->get();

        return CartResource::collection($cartItems);

    }

    public function delete($id)

    {

        $cart = Auth::user()->cart()->findOrFail($id);

        $cart->delete($id);

        return response()->json('deleted');

    }

}

Cart Resource

<?php

namespace App\Http\Resources;

use App\Http\Resources\ProductResource;

use App\Http\Resources\UserResource;

use Illuminate\Http\Resources\Json\JsonResource;

class CartResource extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable

     \*/

    public function toArray($request)

    {

        return [

            'id' =>  $this->id,

            'quantity' => $this->quantity,

            'product' => new ProductResource($this->whenLoaded('product')),

            'user' => new UserResource($this->whenLoaded('user')),

        ];

    }

}

Cart Model

<?php

namespace App\Models;

use App\Models\Product;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Cart extends Model

{

    protected $fillable = [

        'title','newPrice','oldPrice','quantity','product\_id',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

        'quantity' => 'integer',

    ];

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    public function product()

    {

        return $this->belongsTo(Product::class);

    }

}

Wishlist Controller

<?php

namespace App\Http\Controllers;

use App\Http\Resources\WishListResource;

use App\Models\WishList;

use App\Models\Product;

use App\Models\Shop;

use App\Traits\MediaTrait;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Validator;

class WishListController extends Controller

{

    public function store(Request $request)

    {

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

            'product\_id'    => 'required|exists:products,id',

            //'quantity'    => 'required|integer|min:1',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        Auth::user()->wishlist()->updateOrCreate(

            [

                'product\_id' => $request->product\_id,

            ],

            $validator->validated()

        );

        $wishlistItems = Auth::user()->wishlist()

            ->with('product', 'product.media')

            ->get();

        return WishListResource::collection($wishlistItems);

    }

    public function index()

    {

        $wishlistItems = Auth::user()->wishlist()

            ->with('product', 'product.media')

            ->get();

        return WishListResource::collection($wishlistItems);

    }

    public function delete($id)

    {

        $wishlist = Auth::user()->wishlist()->findOrFail($id);

        $wishlist->delete($id);

        return response()->json('deleted');

    }

}

Wishlist Resource

<?php

namespace App\Http\Resources;

use App\Http\Resources\ProductResource;

use App\Http\Resources\UserResource;

use Illuminate\Http\Resources\Json\JsonResource;

class WishListResource extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable

     \*/

    public function toArray($request)

    {

        return [

            'id' =>  $this->id,

            //'quantity' => $this->quantity,

            'product' => new ProductResource($this->whenLoaded('product')),

            'user' => new UserResource($this->whenLoaded('user')),

        ];

    }

}

Wishlist Model

<?php

namespace App\Models;

use App\Models\Product;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class WishList extends Model

{

    protected $fillable = [

        'title','newPrice','oldPrice','product\_id',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    /\*

    protected $casts = [

        'quantity' => 'integer',

    ];

    \*/

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    public function product()

    {

        return $this->belongsTo(Product::class);

    }

}

Contact us Controller

<?php

namespace App\Http\Controllers;

use App\Mail\contact as MailContact;

use App\Models\Contact;

use Illuminate\Http\Request;

use Mail;

class ContactController extends Controller

{

    public function submit(Request $request)

     {

        //store data in database

        Contact::create($request->all());

        // Send mail to Application Admin

        Mail::to("homebazzar.technology@gmail.com")->send(new MailContact( $request->name, $request->phone, $request->email, $request->subject, $request->message));

        return response()->json(['success' => 'we have recieved your message and would like to thank you for writing to us.']);

      }

}

Contact us Request

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ContactRequest extends FormRequest

{

    /\*\*

     \* Determine if the user is authorized to make this request.

     \*

     \* @return bool

     \*/

    public function authorize()

    {

        return true;

    }

    /\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, mixed>

     \*/

    public function rules()

    {

        return [

            'name' => ['required'],

            'email' => ['required', 'email'],

            'message' => ['required'],

        ];

    }

}

Contact us mail

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Queue\SerializesModels;

class contact extends Mailable

{

    use Queueable, SerializesModels;

    public $name;

    public $phone;

    public $email;

    public $message;

    public $subject;

    /\*\*

     \* Create a new message instance.

     \*

     \* @return void

     \*/

    public function \_\_construct($name, $phone, $email, $message, $subject)

    {

       $this->name = $name;

       $this->phone = $phone;

       $this->email = $email;

       $this->message = $message;

       $this->subject = $subject;

    }

    /\*\*

     \* Get the message envelope.

     \*

     \* @return \Illuminate\Mail\Mailables\Envelope

     \*/

    public function envelope()

    {

        return new Envelope(

            subject: 'contactus',

        );

    }

    public function build()

    {

        return $this->markdown('Mails.contactus');

    }

    /\*\*

     \* Get the message content definition.

     \*

     \* @return \Illuminate\Mail\Mailables\Content

     \*/

    // public function content()

    // {

    //     return new Content(

    //         view: 'view.name',

    //     );

    // }

    /\*\*

     \* Get the attachments for the message.

     \*

     \* @return array

     \*/

    public function attachments()

    {

        return [];

    }

}

Contact us Mail Format

<h1> Hello </h1> <br><br>

You have got an email from: {{$name}}  <br><br>

User Details: <br><br>

Name:    {{$name}}        <br>

Email:   {{$email}}       <br>

Phone:   {{$phone}}       <br>

subject: {{$subject}}     <br>

message: {{$message}}     <br>

Thanks

Contact Model

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Contact extends Model

{

    use HasFactory;

    //protected $primaryKey = '$id';

    public $table = 'contacts';

    public $fillable = ['name', 'phone','email', 'subject', 'message'];

    //protected $timestamps = true;

    // public function user()

    // {

    //     return $this->belongsTo(User::class);

    // }

}

Order to finish Controller

<?php

namespace App\Http\Controllers;

use App\Mail\CustomerMail;

use App\Mail\ordertofinish as Mailordertofinish;

use App\Models\OrderToFinish;

use App\Models\Product;

use Illuminate\Http\Request;

use Mail;

class OrderToFinishController extends Controller

{

    public function send(Request $request)

     {

        $product =Product::find($request->product\_id);

       // $productPrice = Product::find($product->newprice);

       // dd($product);

       $offer = 10;

       if($product){

        OrderToFinish::create($request->all());

        $product->newprice = $product->oldPrice - ($offer / 100) \* $product->oldPrice;

        $product->save();

        //  Send mail to Application Admin

        Mail::to("homebazzar.technology@gmail.com")->send(new Mailordertofinish( $request->name, $request->product\_id, $request->email, $request->address, $request->phone\_number, $request->description));

        //  send mail from admin to customer

        Mail::to($request->email)->send(new CustomerMail( $request->name, $request->email, $request->address, $request->phone\_number, $request->description, $product->newPrice));

        return response()->json(['success' => 'we have recieved your message and would like to thank you for writing to us.']);

    }

    else{

        return response()->json(['Error' => 'Please send your order Again']);

    }

       }

}

Order to finish Mail

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Queue\SerializesModels;

class ordertofinish extends Mailable

{

    use Queueable, SerializesModels;

    public $name;

    public $product\_id;

    public $email;

    public $address;

    public $phone\_number;

    public $description;

    /\*\*

     \* Create a new message instance.

     \*

     \* @return void

     \*/

    public function \_\_construct($name, $product\_id, $email, $address, $phone\_number, $description)

    {

        $this->name = $name;

        $this->product\_id = $product\_id;

        $this->email = $email;

        $this->address = $address;

        $this->phone\_number = $phone\_number;

        $this->description = $description;

    }

    /\*\*

     \* Get the message envelope.

     \*

     \* @return \Illuminate\Mail\Mailables\Envelope

     \*/

    public function envelope()

    {

        return new Envelope(

            subject: 'ordertofinish',

        );

    }

    public function build()

    {

        return $this->markdown('Mails.ordertofinish');

    }

    /\*\*

     \* Get the message content definition.

     \*

     \* @return \Illuminate\Mail\Mailables\Content

     \*/

    // public function content()

    // {

    //     return new Content(

    //         view: 'view.name',

    //     );

    // }

    /\*\*

     \* Get the attachments for the message.

     \*

     \* @return array

     \*/

    public function attachments()

    {

        return [];

    }

}

Order to finish mail format

<h1> Order Details </h1> <br><br>

You have got an email from: {{$name}}  <br><br>

User Details: <br><br>

Name:           {{$name}}        <br>

Product id:     {{$product\_id}}        <br>

Email:          {{$email}}       <br>

Address:        {{$address}}       <br>

Phone:          {{$phone\_number}}     <br>

Description:    {{$description}}     <br>

Thanks.

Order to finish Request

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class OrderToFinishRequest extends FormRequest

{

    /\*\*

     \* Determine if the user is authorized to make this request.

     \*

     \* @return bool

     \*/

    public function authorize()

    {

        return true;

    }

    /\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, mixed>

     \*/

    public function rules()

    {

        return [

            'name' => ['required'],

            'product\_id' => ['required'],

            'email' => ['required', 'email'],

            'address' => ['required'],

            'phone\_number' => ['required'],

            'description' => ['required'],

        ];

    }

}

Order to finish model

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class OrderToFinish extends Model

{

    use HasFactory;

    public $table = 'ordertofinish';

    public $fillable = ['name', 'product\_id', 'email','address', 'phone\_number', 'description'];

    public function product()

    {

        return $this->belongsTo(Product::class);

    }

}

Customer Mail

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Queue\SerializesModels;

use App\Models\Product;

class CustomerMail extends Mailable

{

    use Queueable, SerializesModels;

    public $name;

    public $email;

    public $address;

    public $phone\_number;

    public $description;

    public $productprice;

    public $deliverytime = '10-15 business days';

    /\*\*

     \* Create a new message instance.

     \*

     \* @return void

     \*/

    public function \_\_construct($name, $email, $address, $phone\_number, $description,  $productprice)

    {

        $this->name = $name;

        $this->email = $email;

        $this->address = $address;

        $this->phone\_number = $phone\_number;

        $this->description = $description;

        $this->productprice = $productprice;

        //$this->deliveryTime = $deliveryTime;

    }

    /\*\*

     \* Get the message envelope.

     \*

     \* @return \Illuminate\Mail\Mailables\Envelope

     \*/

    public function envelope()

    {

        return new Envelope(

            subject: 'CustomerMail',

        );

    }

    /\*\*

     \* Get the message content definition.

     \*

     \* @return \Illuminate\Mail\Mailables\Content

     \*/

    public function build()

    {

        return $this->markdown('Mails.CustomerMail');

    }

    /\*\*

     \* Get the attachments for the message.

     \*

     \* @return array

     \*/

    public function attachments()

    {

        return [];

    }

}

Customer Mail format

<h1> Order Details </h1> <br><br>

Dear {{$name}},

Thank you for your recent purchase! We wanted to confirm the details of your order:<br><br>

Email: {{$email}}    <br>

Address: {{$address}} <br>

Phone: {{$phone\_number}} <br>

Description: {{$description}} <br>

Product Price: {{$productprice}} <br>

Delivery Time: {{$deliverytime}} <br><br>

If you have any questions or concerns, please do not hesitate to contact us.<br>

Thank you for your business!<br>

Best regards.<br>

Company Name: Home Bazzar.

Email Verification Controller

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Http\Requests\Auth\EmailVerificationRequest;

use App\Models\User;

use Illuminate\Http\Request;

use App\Notifications\EmailVerificationNotification;

use Otp;

class EmailVerificationController extends Controller

{

    private $otp;

    public function \_\_construct()

    {

           $this->otp = new Otp;

    }

    public function sendEmailVerification (Request $request)

    {

       $request->user()->notify(new EmailVerificationNotification());

       $success['The new code has been sent to your Gmail, please go and check it'] = true;

       return response()->json($success, 200);

    }

    public function email\_verification(EmailVerificationRequest $request)

    {

        $otp2 = $this->otp->validate($request->email, $request->otp);

       // dd($otp2);

        if(!$otp2->status){

            return response()->json(['error' => $otp2], 401);

        }

        $user = User::where('email', $request->email)->first();

         $user->update(['email\_verified\_at' =>now(), 'varifed'=> 1]);

        $success['email has been verified'] = true;

        return response()->json($success, 200);

    }

}

Email Verification Request

<?php

namespace App\Http\Requests\Auth;

use Illuminate\Foundation\Http\FormRequest;

class EmailVerificationRequest extends FormRequest

{

    /\*\*

     \* Determine if the user is authorized to make this request.

     \*

     \* @return bool

     \*/

    public function authorize()

    {

        return true;

    }

    /\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, mixed>

     \*/

    public function rules()

    {

        return [

            'email' => ['required', 'email', 'exists:users'],

            'otp' => ['required', 'max:6'],

        ];

    }

}

Email Verification Notification

<?php

namespace App\Notifications;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Notifications\Messages\MailMessage;

use Illuminate\Notifications\Notification;

use Otp;

class EmailVerificationNotification extends Notification

{

    use Queueable;

    public $message;

    public $subject;

    public $fromEmail;

    public $mailer;

    private $otp;

    /\*\*

     \* Create a new notification instance.

     \*

     \* @return void

     \*/

    public function \_\_construct()

    {

        $this->message = 'Use the below code for verification process';

        $this->subject = 'Verification Needed';

        $this->fromEmail = 'rowida001@gmail.com';

        $this->mailer = 'smtp';

        $this->otp = new Otp;

    }

    /\*\*

     \* Get the notification's delivery channels.

     \*

     \* @param  mixed  $notifiable

     \* @return array

     \*/

    public function via($notifiable)

    {

        return ['mail'];

    }

    /\*\*

     \* Get the mail representation of the notification.

     \*

     \* @param  mixed  $notifiable

     \* @return \Illuminate\Notifications\Messages\MailMessage

     \*/

    public function toMail($notifiable)

    {

        $otp = $this->otp->generate($notifiable->email,6, 60);

        return (new MailMessage)

        ->mailer('smtp')

        ->subject($this->subject)

        ->greeting('Hello'.$notifiable->first\_name)

        ->line($this->message)

        ->line('code: '. $otp->token);

    }

    /\*\*

     \* Get the array representation of the notification.

     \*

     \* @param  mixed  $notifiable

     \* @return array

     \*/

    public function toArray($notifiable)

    {

        return [

            //

        ];

    }

}