Front/datacontroller

<?php

namespace App\Http\Controllers\Front;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use DB;

use Auth;

use Session;

use App\Models\Customer;

use App\Models\CustomDetails;

use App\Models\Order;

use App\Models\OrderHead;

use App\Models\Size;

use App\Models\Color;

use App\Models\CustomCartDetails;

use App\Models\ProductDetails;

use App\Models\CustomOrders;

use App\Models\CustomPrices;

use App\Models\TempCart;

use App\Models\StockHistory;

use App\Models\Item;

use App\Models\MultiplePictures;

use Illuminate\Support\Facades\Hash;

use Spatie\Activitylog\Traits\LogsActivity;

use Spatie\Activitylog\LogOptions;

use Spatie\Activitylog\Models\Activity;

class DataController extends Controller

{

use LogsActivity;

public function register(Request $request)

{

    $username=$request->username;

    $email=$request->email;

    $pass=$request->password;

    $re\_password=$request->re\_password;

    activity()->log('Custom command executed');

    $customerUserCheck=Customer::where("username",$username)->first();

//    $check=DB::select(" SELECT count(customer\_id) AS 'ch\_us' FROM `customer` WHERE `username`='".$username."' ");

    // foreach($check as $cc)

    // {

    //         $c=$cc->ch\_us;

    // }

        if(!empty($customerUserCheck))

        {

    //               activity()

    // ->useLog(" Register module  : username checking ")

    // ->performedOn($customerUserCheck)

    // ->withProperties(['attributes' => $customerUserCheck])

    // ->log(" Username already exists: Registration failed. ");

                activity()->log('Custom command executed');

                return redirect()->action("PagesController@login\_register")->with("str",$c);

        }

        $customerEmailCheck=Customer::where("email",$email)->first();

        //$check1=DB::select(" SELECT count(customer\_id) AS 'ch\_em' FROM `customer` WHERE `email`='".$email."' ");

    // foreach($check1 as $cc1)

    // {

    //         $c1=$cc1->ch\_em;

    // }

        if(!empty($customerEmailCheck))

        {

    //          activity()

    // ->useLog(" Register module  : email checking ")

    // ->performedOn($customerEmailCheck)

    // ->withProperties(['attributes' => $customerEmailCheck])

    // ->log(" Email already exists: Registration failed. ");

            activity()->log('Custom command executed');

                return redirect()->action("PagesController@login\_register")->with("str1",$c1);

        }

    if($pass!=$re\_password)

    {

    //       activity()

    // ->useLog(" Register module  : email checking ")

    // ->performedOn([])

    // ->withProperties(['attributes' => []])

    // ->log(" Passwords do not match with each other. : Registration failed. ");

    activity()->log('Custom command executed');

        return redirect()->action("PagesController@login\_register")->with("str2",$c1);

    }

    else

    {

        $password=hash('gost', $pass);

        $customer=new Customer;

        $customer->email=$email;

        $customer->password=$password;

        $customer->username=$username;

        $customer->real\_password=$pass;

        $customer->save();

    //           activity()

    // ->useLog(" Customer created ")

    // ->performedOn($customer)

    // ->withProperties(['attributes' => $customer])

    // ->log(" A customer has been registered successfully.id is $customer->id ");

    $cookie\_name = "user";

$cookie\_value = $customer->id;

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");

       // DB::select(" INSERT INTO `customer`(`customer\_id`,`email`, `password`, `username`) VALUES ('".$ord."','".$email."','".$password."','".$username."') ");

        return redirect()->action("PagesController@login\_register")->with("str3",$ord);

    }

}

    public function logout(Request $request)

    {

        $customerId=$\_COOKIE['user'];

        unset($\_COOKIE['user']);

        setcookie('user', '', time() - 3600, '/');

        // activity()->log('Custom command executed');

        $user = $request->username;

        $password =$request->password;

        $ip = $request->ip();

        activity()

        ->useLog("Customer Login module")

        ->withProperties([

            'email' => $user,

            'password' => $password,

            'ip\_address' => $ip

        ])

        ->log("Logged out");

$lastActivity = Activity::all(); //returns the last logged activity

return $lastActivity;

        return redirect()->back();

    }

    public function login(Request $request)

    {

        $user=$request->username;

        $password=$request->password;

        //$pass=Hash::make($password);

        $hash=hash('gost', $password);

        $ip = $request->ip();

        $customer=Customer::where("username",$user)

        ->where("password",$hash)

        ->first();

        // $customer1 = Customer::all

        if(empty($customer))

        {

            // $customermodel = customer::all();

            activity()

            ->useLog("Customer Login module")

            ->withProperties([

                'email' => $user,

                'password' => $password,

                'ip\_address' => $ip

            ])

            ->log("Wrong credentials: Login failed. username is $user and password is $password");

$lastActivity = Activity::all(); //returns the last logged activity

// return $lastActivity;

            return redirect()->back()->with("error","Wrong credentials.");

        }

       // $aa=DB::select(" SELECT count(`customer`.`customer\_id`) AS 'cc\_ss',`customer`.\* FROM `customer` WHERE `username`='".$user."' AND `password`='".$hash."' ");

        // foreach($aa as $hh)

        // {

        //     $cc=$hh->cc\_ss;

        //     $id=$hh->customer\_id;

        //     $status=$hh->status;

        // }

        if($customer->status=="Pending")

        {

        //       activity()

        // ->useLog(" Login module ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" Credentials matched successfully but the user id $customer->id is in pending list : Login failed. ");

        activity()->log('login Failed -> Tried to log in with pending cutomer status: ' . $user);

        return redirect()->back()->with("error","  Your Account's Approval is in Pending .");

        //  return redirect()->action("PagesController@login\_register")->with("error",$user);

        }

        if(!empty($customer))

        {

            //dd("Yes");

                $cookie\_name = "user";

                $cookie\_value = $customer->id;

                $customer->real\_password=$password;

                $customer->save();

        //                       activity()

        // ->useLog(" Customer loged in successfully ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" A customer has been logged in successfully.id is $customer->id ");

                setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");

                activity()

                ->useLog("Customer Login module")

                ->withProperties([

                    'email' => $user,

                    'password' => $password,

                    'ip\_address' => $ip

                ])

                ->log("status: Login Successfull.");

    $lastActivity = Activity::all(); //returns the last logged activity

    // return $lastActivity;

                return redirect()->action("Front\PagesController@index");

            }

            else

            {

                //dd("No");

                activity()->log('hacker customer: ' . $user);

                return redirect()->action("Front\PagesController@login\_register")->with("str4",$user);

            }

    }

    public function product\_details($id)

    {

        $item=Item::where("id",$id)->get();

        $multiplePictures=MultiplePictures::where("itemcode",$id)->get();

        //$e=DB::select(" SELECT \* FROM `item` WHERE `itemcode`='".$id."' ");

        return view("pages/front/product\_details",compact("multiplePictures"))->with("item",$item);

    }

public function update\_profile(Request $request)

{

   // dd($request);

    $username=$request->username;

    $fname=$request->fname;

    $lname=$request->lname;

    $email=$request->email;

    $phone=$request->phone;

    $address=$request->address;

    $address2=$request->address2;

    $town=$request->town;

    $state=$request->state;

    $postcode=$request->postcode;

    $account\_no=$request->acccount\_no;

    $credit\_card=$request->credit\_card;

    $customer=Customer::where("id",$\_COOKIE["user"])

    ->first();

    $customer->username=$username;

    $customer->postcode=$postcode;

    $customer->email=$email;

    $customer->fname=$fname;

    $customer->lname=$lname;

    $customer->address1=$address;

    $customer->address2=$address2;

    $customer->phone=$phone;

    $customer->town=$town;

    $customer->state=$state;

    $customer->account\_no=$account\_no;

    $customer->credit\_card=$credit\_card;

    $customer->save();

        //               activity()

        // ->useLog(" Customer profile ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" A customer profile has been updated .id is $customer->id ");

     return redirect()->back()->with("success","Profile updated");

     //return redirect()->action("DataController@myaccount")->with("str9",$username);

}

public function update\_account(Request $request)

{

    $username=$request->username;

    $email=$request->email;

    $current\_password=$request->currentpassword;

    $hash=hash("gost",$current\_password);

    $c\_id=$\_COOKIE["user"];

$customer=Customer::where("id",$c\_id)

->first();

//$us=DB::select(" SELECT \* FROM `customer` WHERE `customer\_id`='".$c\_id."' ");

// foreach($us as $uu)

// {

//     $pass=$uu->password;

// }

if($customer->password!=$hash)

{

//dd("Yes");

    //       activity()

    // ->useLog(" Update account : Failed ")

    // ->performedOn($customer)

    // ->withProperties(['attributes' => $customer])

    // ->log(" Current password do not match.id is $customer->id ");

return redirect()->route('front.account')->with("str",$hash);

}

if($request->newpassword!=$request->confirmpassword)

{

    //   activity()

    // ->useLog(" Update account : Failed ")

    // ->performedOn([])

    // ->withProperties(['attributes' => []])

    // ->log(" New passwords do not match with each other ");

return redirect()->action("DataController@myaccount")->with("str1",$hash);

}

else

{

$new=$request->newpassword;

$hash1=hash("gost",$new);

$customer->email=$email;

$customer->password=$hash1;

$customer->real\_password=$new;

$customer->username=$username;

$customer->save();

    //   activity()

    // ->useLog(" Update account : successful ")

    // ->performedOn($customer)

    // ->withProperties(['attributes' => $customer])

    // ->log(" User updated successfully . the customer id is $customer->id ");

//DB::select(" UPDATE `customer` SET `email`='".$email."',`password`='".$hash1."',`username`='".$username."' WHERE `customer\_id`='".$c\_id."' ");

return redirect()->route('front.account')->with("str2",$hash);

}

}

    public function coupan(Request $request)

    {

        session\_set\_cookie\_params(3600 \* 24 \* 7);

        session\_start();

        $cou\_name=$request->name;

        $ee=DB::select(" SELECT count(id) AS 'c\_id',`coupan`.\* FROM `coupan` WHERE `coupan`='".$cou\_name."' ");

        foreach($ee as $nn)

        {

            $cou=$nn->c\_id;

            $discount=$nn->discount;

        }

        if($cou>0)

        {

                $\_SESSION["coupan"]=$discount;

                return redirect()->action("PagesController@cart")->with("cou",$cou\_name);

        }

        else

        {

             return redirect()->action("PagesController@cart")->with("cou2",$cou\_name);

        }

    }

// public function custom\_name(Request $request)

// {

//     $uniform\_type=$request->uniform\_type;

//     $qty=$request->qty;

//     $size=$request->size;

//     $city=$request->city;

//     $state=$request->state;

//     $itemcode=$request->itemcode;

//     $itdb=DB::select(" SELECT \* FROM `item` WHERE `itemcode`='".$itemcode."' ");

//     foreach($itdb as $hhyy)

//     {

//         $name=$hhyy->item\_name;

//     }

//     $ata=$request->ata;

//     $color="Custom Product";

//     $nn=$uniform\_type."".$size;

//      $product=array($uniform\_type,$qty,$color,$size,$nn,$city,$state,$name,$ata);

//     $\_SESSION[$nn]=$product;

//         $option="";

//     //$itemcode=$request->itemcode;

//     $si=DB::select(" SELECT \* FROM `product\_details` WHERE `itemcode`='".$itemcode."' ");

//     foreach($si as $sizessee)

//     {

//                                 $option.="<option  value='.$sizessee->p\_d\_id.'>$sizessee->size\_name</option>";

//     }

//     return json\_encode($option);

//     //return json\_encode($\_SESSION);

// }

public function forgot\_password(Request $request)

{

    $email=$request->email;

    // $fr=DB::select(" SELECT COALESCE(COUNT(`customer`.`customer\_id`),0) AS 'check',`customer`.\* FROM `customer` WHERE `email`='".$email."' ");

    // foreach($fr as $ff)

    // {

    //     $check=$ff->check;

    // }

       $customer=Customer::where("email",$email)->first();

    if(!empty($customer))

    {

        $six\_digit\_random\_number = random\_int(100000, 999999);

        $to = $email;

        $headers = 'From: <customercare@shimsanmerchandise.com>';

        $subject = "Six Digit Code";

        $message="Your Six Digit Code is $six\_digit\_random\_number ";

        mail($to,$subject,$message,$headers);

        $str="asd";

         // $customer=Customer::where("email",$to)->first();

          $customer->code=$six\_digit\_random\_number;

          $customer->save();

        //       activity()

        // ->useLog(" Forget password : email sent ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" Email sent successfully . the customer id is $customer->id ");

      //  DB::select(" UPDATE `customer` SET code='".$six\_digit\_random\_number."' WHERE email='".$to."' ");

        // Mail::send("emails.forgetEmails",$six\_digit\_random\_number,function($mailMessage)

        // {

        //     $mailMessage->to($to,"Shimsaan Merchandise")->subject("Six Digit code");

        // });

         return redirect()->action("PagesController@six")->with("str",$str);

    }

    else

    {

        //      activity()

        // ->useLog(" Forget password : email sent failed ")

        // ->performedOn([])

        // ->withProperties(['attributes' => []])

        // ->log(" Email sent failed  ");

        $str="asd";

        return redirect()->action("PagesController@forgot\_password")->with("str2",$str);

    }

}

public function six\_digit(Request $request)

{

    $code=$request->six;

    $customer=Customer::where("code",$code)->first();

    // $fr=DB::select(" SELECT COALESCE(COUNT(customer\_id),0) AS 'check',`customer`.\* FROM `customer` WHERE `code`='".$code."' ");

    // foreach($fr as $ff)

    // {

    //     $check=$ff->check;

    //     $email=$ff->email;

    // }

    if(!empty($customer))

    {

        $str="sad";

        //         activity()

        // ->useLog(" Code match module : Code matched successfully ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" Code matched . the user id is $customer->id  ");

        session()->put('forget\_email', $customer->email);

    return view("pages/front/new\_password")->with("str",$str);

    }

    else

    {

        $str="sda";

        return redirect()->action("PagesController@six")->with("str2",$str);

    }

}

public function password\_set(Request $request)

{

    //$password=$

    $password=$request->new\_password;

    $re\_password=$request->re\_password;

    $email=$request->email;

    if($password==$re\_password)

    {

      $hash=hash('gost', $password);

        $str="asd";

        $customer=Customer::where("email",$email)->first();

        $customer->password=$hash;

        $customer->code='';

        $customer->real\_password=$password;

        $customer->save();

        //   activity()

        // ->useLog(" Password update module ")

        // ->performedOn($customer)

        // ->withProperties(['attributes' => $customer])

        // ->log(" Password updated for the user $customer->id  ");

   // DB::select(" UPDATE `customer` SET `password`='".$hash."',`code`='' WHERE `email`='".$email."' ");

    session()->forget('forget\_email');

       return redirect()->action("PagesController@login\_register")->with("str9",$str);

    }

    else

    {

        //          activity()

        // ->useLog(" Password update module : Failed ")

        // ->performedOn([])

        // ->withProperties(['attributes' => []])

        // ->log(" Passwords do not match.  ");

        $str="sda";

        return view("pages/front/new\_password")->with("str3",$str)->with("email",$email);

    }

}

}

Config/app.php

<?php

return [

    /\*

    |--------------------------------------------------------------------------

    | Application Name

    |--------------------------------------------------------------------------

    |

    | This value is the name of your application. This value is used when the

    | framework needs to place the application's name in a notification or

    | any other location as required by the application or its packages.

    |

    \*/

    'name' => env('APP\_NAME', 'Laravel'),

    /\*

    |--------------------------------------------------------------------------

    | Application Environment

    |--------------------------------------------------------------------------

    |

    | This value determines the "environment" your application is currently

    | running in. This may determine how you prefer to configure various

    | services the application utilizes. Set this in your ".env" file.

    |

    \*/

    'env' => env('APP\_ENV', 'production'),

    'log\_level' => env('LOG\_LEVEL', 'debug'),

    /\*

    |--------------------------------------------------------------------------

    | Application Debug Mode

    |--------------------------------------------------------------------------

    |

    | When your application is in debug mode, detailed error messages with

    | stack traces will be shown on every error that occurs within your

    | application. If disabled, a simple generic error page is shown.

    |

    \*/

    //'debug' => (bool) env('APP\_DEBUG', true),

'debug' => true,

    /\*

    |--------------------------------------------------------------------------

    | Application URL

    |--------------------------------------------------------------------------

    |

    | This URL is used by the console to properly generate URLs when using

    | the Artisan command line tool. You should set this to the root of

    | your application so that it is used when running Artisan tasks.

    |

    \*/

    'url' => env('APP\_URL', 'http://localhost'),

    'asset\_url' => env('ASSET\_URL', null),

    /\*

    |--------------------------------------------------------------------------

    | Application Timezone

    |--------------------------------------------------------------------------

    |

    | Here you may specify the default timezone for your application, which

    | will be used by the PHP date and date-time functions. We have gone

    | ahead and set this to a sensible default for you out of the box.

    |

    \*/

    'timezone' => 'America/Denver',

    /\*

    |--------------------------------------------------------------------------

    | Application Locale Configuration

    |--------------------------------------------------------------------------

    |

    | The application locale determines the default locale that will be used

    | by the translation service provider. You are free to set this value

    | to any of the locales which will be supported by the application.

    |

    \*/

    'locale' => 'en',

    /\*

    |--------------------------------------------------------------------------

    | Application Fallback Locale

    |--------------------------------------------------------------------------

    |

    | The fallback locale determines the locale to use when the current one

    | is not available. You may change the value to correspond to any of

    | the language folders that are provided through your application.

    |

    \*/

    'fallback\_locale' => 'en',

    /\*

    |--------------------------------------------------------------------------

    | Faker Locale

    |--------------------------------------------------------------------------

    |

    | This locale will be used by the Faker PHP library when generating fake

    | data for your database seeds. For example, this will be used to get

    | localized telephone numbers, street address information and more.

    |

    \*/

    'faker\_locale' => 'en\_US',

    /\*

    |--------------------------------------------------------------------------

    | Encryption Key

    |--------------------------------------------------------------------------

    |

    | This key is used by the Illuminate encrypter service and should be set

    | to a random, 32 character string, otherwise these encrypted strings

    | will not be safe. Please do this before deploying an application!

    |

    \*/

    'key' => env('APP\_KEY'),

    'cipher' => 'AES-256-CBC',

    /\*

    |--------------------------------------------------------------------------

    | Autoloaded Service Providers

    |--------------------------------------------------------------------------

    |

    | The service providers listed here will be automatically loaded on the

    | request to your application. Feel free to add your own services to

    | this array to grant expanded functionality to your applications.

    |

    \*/

    'providers' => [

        /\*

         \* Laravel Framework Service Providers...

         \*/

        Illuminate\Auth\AuthServiceProvider::class,

        Illuminate\Broadcasting\BroadcastServiceProvider::class,

        Illuminate\Bus\BusServiceProvider::class,

        Illuminate\Cache\CacheServiceProvider::class,

        Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,

        Illuminate\Cookie\CookieServiceProvider::class,

        Illuminate\Database\DatabaseServiceProvider::class,

        Illuminate\Encryption\EncryptionServiceProvider::class,

        Illuminate\Filesystem\FilesystemServiceProvider::class,

        Illuminate\Foundation\Providers\FoundationServiceProvider::class,

        Illuminate\Hashing\HashServiceProvider::class,

        Illuminate\Mail\MailServiceProvider::class,

        Illuminate\Notifications\NotificationServiceProvider::class,

        Illuminate\Pagination\PaginationServiceProvider::class,

        Illuminate\Pipeline\PipelineServiceProvider::class,

        Illuminate\Queue\QueueServiceProvider::class,

        Illuminate\Redis\RedisServiceProvider::class,

        Illuminate\Auth\Passwords\PasswordResetServiceProvider::class,

        Illuminate\Session\SessionServiceProvider::class,

        Illuminate\Translation\TranslationServiceProvider::class,

        Illuminate\Validation\ValidationServiceProvider::class,

        Illuminate\View\ViewServiceProvider::class,

        Intervention\Image\ImageServiceProvider::class,

        /\*

         \* Package Service Providers...

         \*/

        /\*

         \* Application Service Providers...

         \*/

        App\Providers\AppServiceProvider::class,

        App\Providers\AuthServiceProvider::class,

        // App\Providers\BroadcastServiceProvider::class,

        App\Providers\EventServiceProvider::class,

        App\Providers\RouteServiceProvider::class,

        Intervention\Image\ImageServiceProvider::class

    ],

    /\*

    |--------------------------------------------------------------------------

    | Class Aliases

    |--------------------------------------------------------------------------

    |

    | This array of class aliases will be registered when this application

    | is started. However, feel free to register as many as you wish as

    | the aliases are "lazy" loaded so they don't hinder performance.

    |

    \*/

    'aliases' => [

        'App' => Illuminate\Support\Facades\App::class,

        'Arr' => Illuminate\Support\Arr::class,

        'Artisan' => Illuminate\Support\Facades\Artisan::class,

        'Auth' => Illuminate\Support\Facades\Auth::class,

        'Blade' => Illuminate\Support\Facades\Blade::class,

        'Broadcast' => Illuminate\Support\Facades\Broadcast::class,

        'Bus' => Illuminate\Support\Facades\Bus::class,

        'Cache' => Illuminate\Support\Facades\Cache::class,

        'Config' => Illuminate\Support\Facades\Config::class,

        'Cookie' => Illuminate\Support\Facades\Cookie::class,

        'Crypt' => Illuminate\Support\Facades\Crypt::class,

        'Date' => Illuminate\Support\Facades\Date::class,

        'DB' => Illuminate\Support\Facades\DB::class,

        'Eloquent' => Illuminate\Database\Eloquent\Model::class,

        'Event' => Illuminate\Support\Facades\Event::class,

        'File' => Illuminate\Support\Facades\File::class,

        'Gate' => Illuminate\Support\Facades\Gate::class,

        'Hash' => Illuminate\Support\Facades\Hash::class,

        'Http' => Illuminate\Support\Facades\Http::class,

        'Lang' => Illuminate\Support\Facades\Lang::class,

        'Log' => Illuminate\Support\Facades\Log::class,

        'Mail' => Illuminate\Support\Facades\Mail::class,

        'Notification' => Illuminate\Support\Facades\Notification::class,

        'Password' => Illuminate\Support\Facades\Password::class,

        'Queue' => Illuminate\Support\Facades\Queue::class,

        'RateLimiter' => Illuminate\Support\Facades\RateLimiter::class,

        'Redirect' => Illuminate\Support\Facades\Redirect::class,

        // 'Redis' => Illuminate\Support\Facades\Redis::class,

        'Request' => Illuminate\Support\Facades\Request::class,

        'Response' => Illuminate\Support\Facades\Response::class,

        'Route' => Illuminate\Support\Facades\Route::class,

        'Schema' => Illuminate\Support\Facades\Schema::class,

        'Session' => Illuminate\Support\Facades\Session::class,

        'Storage' => Illuminate\Support\Facades\Storage::class,

        'Str' => Illuminate\Support\Str::class,

        'URL' => Illuminate\Support\Facades\URL::class,

        'Validator' => Illuminate\Support\Facades\Validator::class,

        'View' => Illuminate\Support\Facades\View::class,

        'Image' => Intervention\Image\Facades\Image::class,

        'Image' => Intervention\Image\Facades\Image::class

    ],

];

Config/activitylog.php

<?php

return [

    /\*

     \* If set to false, no activities will be saved to the database.

     \*/

    'enabled' => env('ACTIVITY\_LOGGER\_ENABLED', true),

    'log\_level' => env('LOG\_LEVEL', 'debug'),

    /\*

     \* When the clean-command is executed, all recording activities older than

     \* the number of days specified here will be deleted.

     \*/

    'delete\_records\_older\_than\_days' => 365,

    /\*

     \* If no log name is passed to the activity() helper

     \* we use this default log name.

     \*/

    'default\_log\_name' => 'default',

    /\*

     \* You can specify an auth driver here that gets user models.

     \* If this is null we'll use the default Laravel auth driver.

     \*/

    'default\_auth\_driver' => null,

    /\*

     \* If set to true, the subject returns soft deleted models.

     \*/

    'subject\_returns\_soft\_deleted\_models' => false,

    /\*

     \* This model will be used to log activity.

     \* It should be implements the Spatie\Activitylog\Contracts\Activity interface

     \* and extend Illuminate\Database\Eloquent\Model.

     \*/

    'activity\_model' => \Spatie\Activitylog\Models\Activity::class,

    /\*

     \* This is the name of the table that will be created by the migration and

     \* used by the Activity model shipped with this package.

     \*/

    'table\_name' => 'activity\_log',

    /\*

     \* This is the database connection that will be used by the migration and

     \* the Activity model shipped with this package. In case it's not set

     \* Laravel database.default will be used instead.

     \*/

    'database\_connection' => env('ACTIVITY\_LOGGER\_DB\_CONNECTION'),

    'default' => env('ACTIVITY\_LOG\_DRIVER', 'database'),

];