Task 1: Request Validation

Implement request validation for a registration form that contains the following fields: name, email, and password. Validate the following rules:

name: required, string, minimum length 2.

email: required, valid email format.

password: required, string, minimum length 8.

Route:

```
//Task 1 Route
Route::post('/registrationForm',[TasksController::class,'validation'])
;
```

Validation Through `StorepostsRequest:`

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Contracts\Validation\Validator;
use Illuminate\Http\Exceptions\HttpResponseException;


class StorepostsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string,
\Illuminate\Contracts\Validation\ValidationRule|array|string>
     */
```

```php
    public function rules(): array
    {
        return [
            'name' => 'required|string|min:2',
            'email' => 'required|email',
            'password' => 'required|string|min:8',
        ];
    }

    /**
     * Get the error messages for the defined validation rules.
     *
     * @return array<string, string>
     */
    public function messages(): array
    {
        return [
            'name.required' => 'The name field is required.',
            'name.min' => 'The name must be at least 2 characters.',
            'email.required' => 'The email field is required.',
            'email.email' => 'Please enter a valid email address.',
            'password.required' => 'The password field is required.',
            'password.min' => 'The password must be at least 8
characters.',
        ];
    }


    protected function failedValidation(Validator $validator): void
    {
        throw new HttpResponseException(
            response()->json([
                'errors' => $validator->errors(),
            ], 422)
        );
    }
}
```

Controller Function:

```
    public function validation(StorepostsRequest $request)
    {
        // If the request passes validation, you can access the
validated data
        $validatedData = $request->validated();

        return response()->json([
            'data' => $validatedData,
            'message' => 'Validation successful',
        ]);
    }
```

Task 2: Request Redirect

Create a route /home that redirects to /dashboard using a 302 redirect.

```
//Task 2 route
Route::get('/dashboard', function () {
    return "I'm from dashboard";
});

Route::get('/home', function () {
    return redirect('/dashboard', 302);
});
```

Task 3: Global Middleware

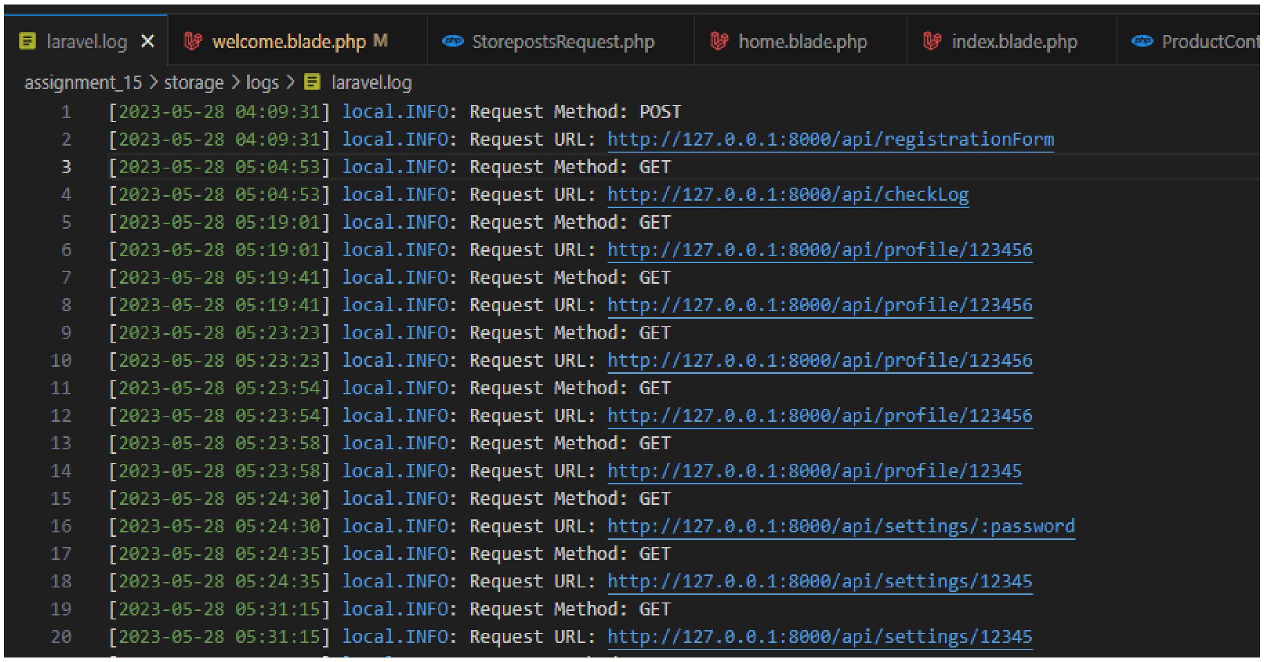
Create a global middleware that logs the request method and URL for every incoming request. Log the information to the Laravel log file.

GlobalMiddleware:

```
<?php
```

```php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Symfony\Component\HttpFoundation\Response;

class GlobalMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        Log::info('Request Method: ' . $request->method());
        Log::info('Request URL: ' . $request->fullUrl());

        return $next($request);
    }
}
```

Output of all logs for this Global Middleware:



```
[2023-05-28 04:09:31] local.INFO: Request Method: POST
[2023-05-28 04:09:31] local.INFO: Request URL: http://127.0.0.1:8000/api/registrationForm
[2023-05-28 05:04:53] local.INFO: Request Method: GET
[2023-05-28 05:04:53] local.INFO: Request URL: http://127.0.0.1:8000/api/checkLog
[2023-05-28 05:19:01] local.INFO: Request Method: GET
[2023-05-28 05:19:01] local.INFO: Request URL: http://127.0.0.1:8000/api/profile/123456
[2023-05-28 05:19:41] local.INFO: Request Method: GET
[2023-05-28 05:19:41] local.INFO: Request URL: http://127.0.0.1:8000/api/profile/123456
[2023-05-28 05:23:23] local.INFO: Request Method: GET
[2023-05-28 05:23:23] local.INFO: Request URL: http://127.0.0.1:8000/api/profile/123456
[2023-05-28 05:23:54] local.INFO: Request Method: GET
[2023-05-28 05:23:54] local.INFO: Request URL: http://127.0.0.1:8000/api/profile/123456
[2023-05-28 05:23:58] local.INFO: Request Method: GET
[2023-05-28 05:23:58] local.INFO: Request URL: http://127.0.0.1:8000/api/profile/12345
[2023-05-28 05:24:30] local.INFO: Request Method: GET
[2023-05-28 05:24:30] local.INFO: Request URL: http://127.0.0.1:8000/api/settings/:password
[2023-05-28 05:24:35] local.INFO: Request Method: GET
[2023-05-28 05:24:35] local.INFO: Request URL: http://127.0.0.1:8000/api/settings/12345
[2023-05-28 05:31:15] local.INFO: Request Method: GET
[2023-05-28 05:31:15] local.INFO: Request URL: http://127.0.0.1:8000/api/settings/12345
```

Task 4: Route Middleware

Create a route group for authenticated users only. This group should include routes for /profile and /settings. Apply a middleware called AuthMiddleware to the route group to ensure only authenticated users can access these routes.

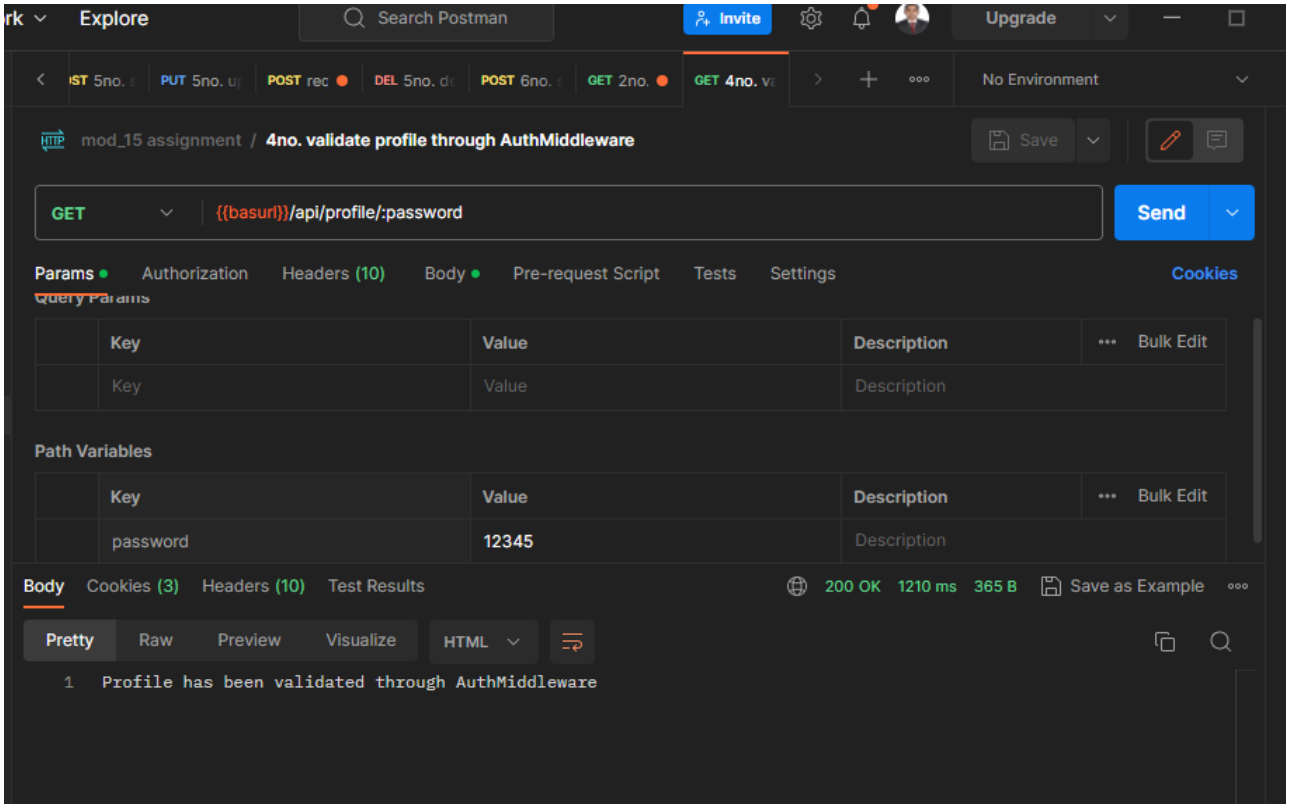
Route:

```
//Task 4
// middilware to a group route
Route::middleware(['AuthMiddleware'])->group(function () {
    Route::get('/profile/{password}', [TasksController::class, 'profile']);
    Route::get('/settings/{password}', [TasksController::class, 'settings']);
});
```

Middleware for this route Group:

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class AuthMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        $password = $request->password;
        if ($password == '12345') {
            return $next($request);
        } else {

            return response()->json('unauthorized', 401);
        }
    }
}
```

Response of this Route group in Postman:



```
{{basurl}}/api/profile/:password
```

Path Variables

Key	Value	Description
password	12345	

```
Profile has been validated through AuthMiddleware
```

Task 5: Controller

Create a controller called ProductController that handles CRUD operations for a resource called Product. Implement the following methods:

ProductController:

```php
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;
use App\Http\Resources\ProductResource;
use App\Http\Requests\StoreProductRequest;
use App\Http\Requests\UpdateProductRequest;

class ProductController extends Controller
{
    public function index()
    {
        //return ProductResource::collection(Product::all());
        $products = Product::all();
        return view('page.index', compact('products'));
    }

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

    public function store(StoreProductRequest $request)
    {
        $product = Product::create($request->validated());
        // return ProductResource::make($product);

        return redirect()->route('page.index')->with('success', 'Product created
successfully');
    }

    public function edit($id)
    {
        $product = Product::findOrFail($id);

        return view('page.edit', compact('product'));
    }

    public function update(UpdateProductRequest $request, $id)
    {
        $product = Product::findOrFail($id);

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

        return redirect()->route('page.index')->with('success', 'Product updated
successfully');
    }
   public function destroy($id)
    {
        $product = Product::findOrFail($id);
        $product->delete();

        return redirect()->route('page.index')->with('success', 'Product deleted
successfully');
    }
```

```
}
```

Routes for this Controller:

```
Route::get('/homepage', [ProductController::class, 'index'])->name('page.index');
Route::get('/create', [ProductController::class, 'create'])->name('page.create');
Route::post('/store', [ProductController::class, 'store'])->name('page.store');
Route::get('/page/{id}/edit', [ProductController::class, 'edit'])-
>name('page.edit');
Route::put('/page/{id}', [ProductController::class, 'update'])-
>name('page.update');
Route::delete('/page/{id}', [ProductController::class, 'destroy'])-
>name('page.destroy');
```

Database Migration for this controller:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description');
            $table->decimal('price', 8, 2);
            $table->integer('quantity');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('products');
    }
};
```

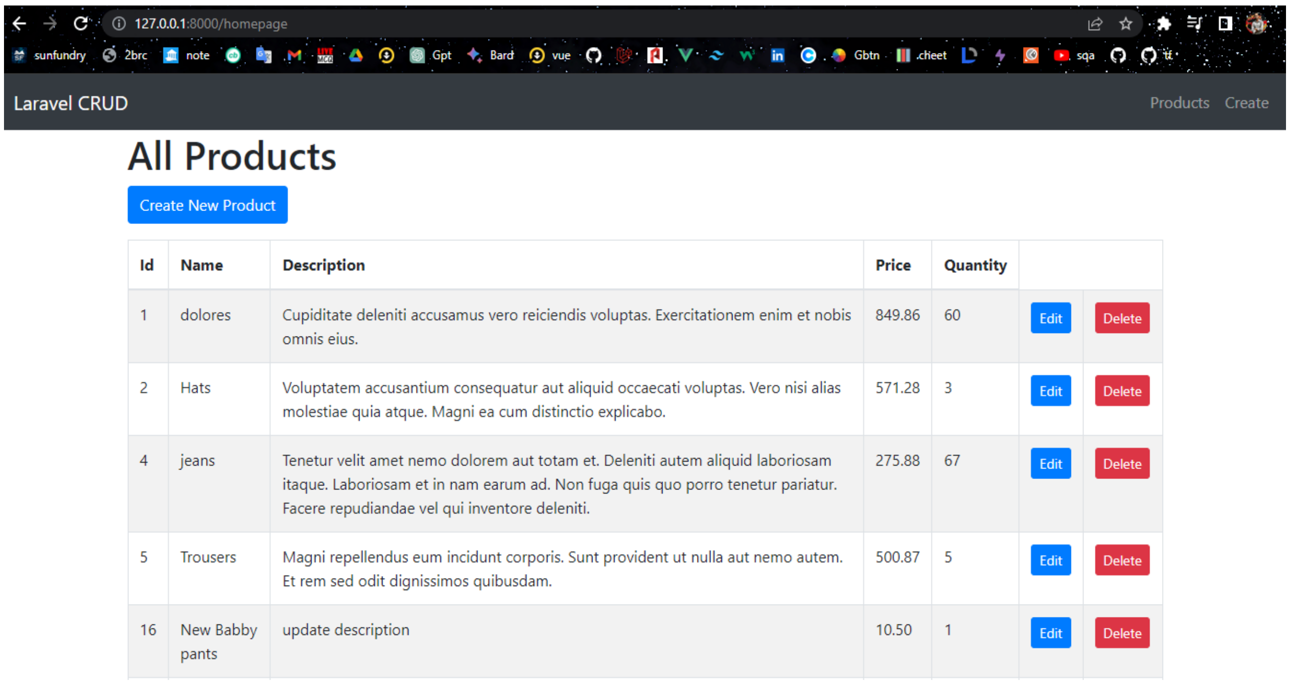
I have seed the data using Factory and seeder:

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Product>
 */
class ProductFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'name' => fake()->word,
            'description' => fake()->paragraph,
            'price' => fake()->randomFloat(2, 1, 1000),
            'quantity' => fake()->numberBetween(1, 100),
        ];
    }
}
```

Seeder:

```php
<?php

namespace Database\Seeders;

use App\Models\Product;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class ProductSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Product::factory(15)->create();
    }
}
```

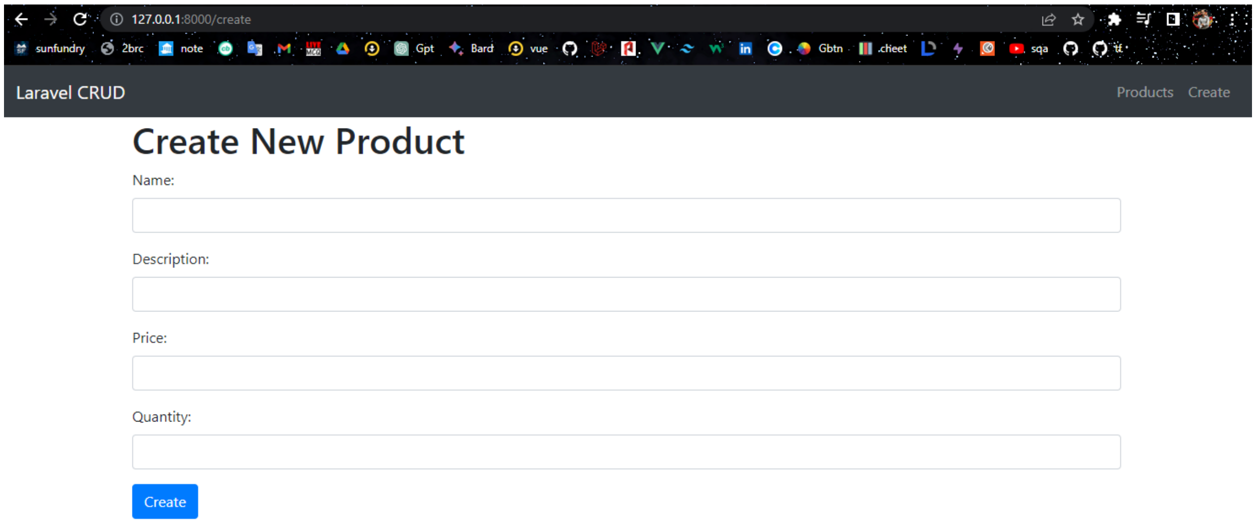
Showing Data of My Product CRUD:



All Products

Id	Name	Description	Price	Quantity		
1	dolores	Cupiditate deleniti accusamus vero reiciendis voluptas. Exercitationem enim et nobis omnis eius.	849.86	60	Edit	Delete
2	Hats	Voluptatem accusantium consequatur aut aliquid occaecati voluptas. Vero nisi alias molestiae quia atque. Magni ea cum distinctio explicabo.	571.28	3	Edit	Delete
4	jeans	Tenetur velit amet nemo dolorem aut totam et. Deleniti autem aliquid laboriosam itaque. Laboriosam et in nam earum ad. Non fuga quis quo porro tenetur pariatur. Facere repudiandae vel qui inventore deleniti.	275.88	67	Edit	Delete
5	Trousers	Magni repellendus eum incidunt corporis. Sunt provident ut nulla aut nemo autem. Et rem sed odit dignissimos quibusdam.	500.87	5	Edit	Delete
16	New Babby pants	update description	10.50	1	Edit	Delete

Create Page:


127.0.0.1:8000/create

Laravel CRUD — Products Create

Create New Product

Name:

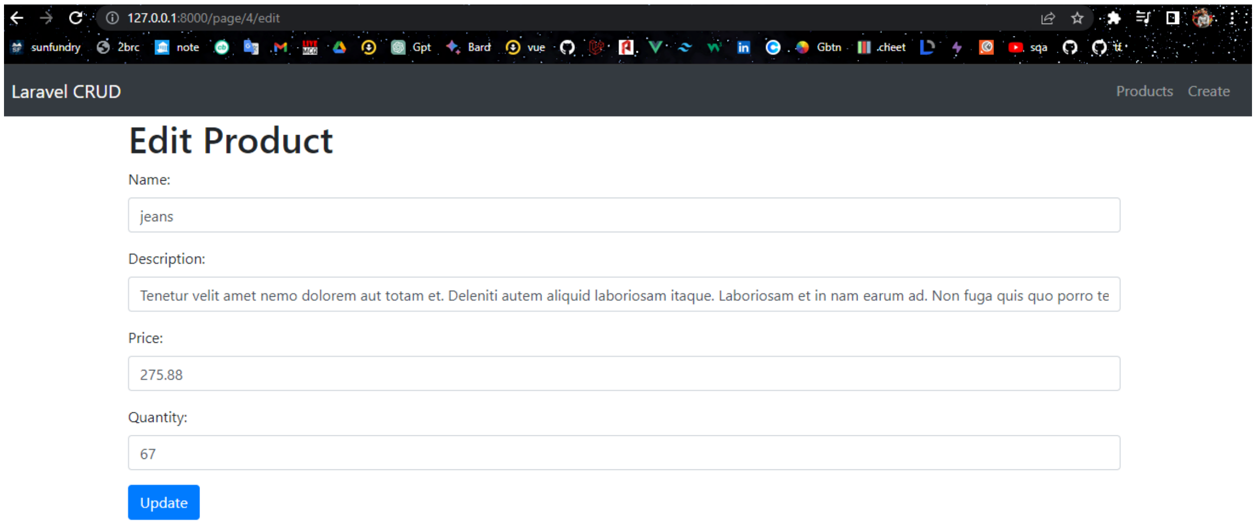
Description:

Price:

Quantity:

Create

Update Page:


127.0.0.1:8000/page/4/edit

Laravel CRUD — Products Create

Edit Product

Name: jeans

Description: Tenetur velit amet nemo dolorem aut totam et. Deleniti autem aliquid laboriosam itaque. Laboriosam et in nam earum ad. Non fuga quis quo porro te

Price: 275.88

Quantity: 67

Update

Task 6: Single Action Controller

Create a single action controller called ContactController that handles a contact form submission. Implement the __invoke() method to process the form submission and send an email to a predefined address with the submitted data

Route:

```php
//Task 6
Route::post('/contact', ContactController::class);
```

ContactController:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ContactController extends Controller
{
    /**
     * Handle the incoming request.
     */
    public function __invoke(Request $request)
    {
        // Validating the request data
        $validator = Validator::make($request->all(), [
            'name' => 'required|min:2',
            'email' => 'required|email',
            'message' => 'required|min:20',
        ], [
            'name.min' => 'The name must be at least 2 characters.',
            'message.min' => 'The message must be at least 20 characters.',
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' => $validator->errors()], 422);
        }

        // Get the form data
        $name = $request->input('name');
        $email = $request->input('email');
        $message = $request->input('message');

        // Return a JSON response
        return response()->json([
            'name' => $name,
            'email' => $email,
            'message' => $message,
            'Confirmation' => 'Message sent successfully! We will be in touch
very soon.',
        ]);
    }
}
```

Task 7: Resource Controller

Create a resource controller called PostController that handles CRUD operations for a resource called Post. Ensure that the controller provides the necessary methods for the resourceful routing conventions in Laravel.

PostController:

```
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;
use App\Http\Resources\PostResource;
use App\Http\Requests\StorepostsRequest;
use App\Http\Requests\UpdatepostsRequest;


class PostController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
```

```php
    {
        //return StorepostsRequest::collection(Product::all());
        $products = Product::all();
        return view('page.index', compact('products'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return view('page.create');
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StorepostsRequest $request)
    {
        $product = Product::create($request->validated());
        // return ProductResource::make($product);

        return redirect()->route('page.index')->with('success', 'Product created
successfully');
    }

    /**
     * Display the specified resource.
     */
    public function show($id)
    {
        $product = Product::findOrFail($id);

        return view('page.index', compact('product'));
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit($id)
    {
        $product = Product::findOrFail($id);

        return view('page.edit', compact('product'));
    }
    /**
     * Update the specified resource in storage.
     */
    public function update(UpdatepostsRequest $request, $id)
    {
        $product = Product::findOrFail($id);

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

        return redirect()->route('page.index')->with('success', 'Product updated
successfully');
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy($id)
    {
        $product = Product::findOrFail($id);
        $product->delete();

        return redirect()->route('page.index')->with('success', 'Product deleted
successfully');
    }
}
```

Route for this:

```
//Task 7
Route::apiResource('/product',PostController::class);
```

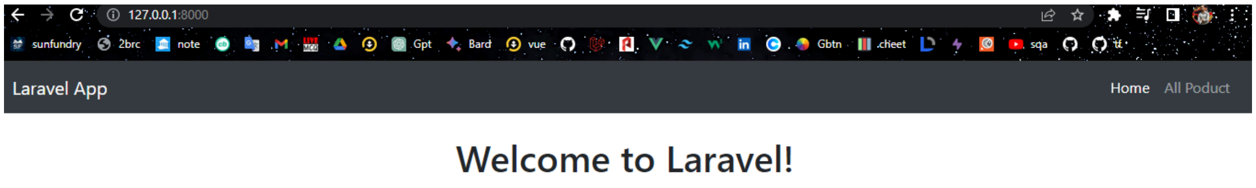
Task 8: Blade Template Engine

Create a Blade view called welcome.blade.php that includes a navigation bar and a section displaying the text "Welcome to Laravel!".

Route:

```
//Task 8
Route::get('/', function () {
    return view('welcome');
});
```

Welcome page:



Welcome to Laravel!