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.

Solution:

Web.php

Route::post('/form',[formController::class,'store'])->name('hello');

formController

namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

 function store(Request $request)

   {

*//dd($request->input());*

      $validate = $request->validate([

         'name' => 'required|string|min:2',

         'email' => 'required|email',

         'password' => 'required|string|min:8'

      ]);

      $user = new User();

      $user->name = $validate['name'];

      $user->email = $validate['email'];

      $user->password = bcrypt($validate['password']);

      $user->save();

      return redirect()->route('welcome');

   }

Task 2: Request Redirect

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

Solution:

Route::get('/home',function(){

    return Redirect::to('/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.

Solution:

Kernel.php

A screen shot of a computer program

Description automatically generated with medium confidence

logfile.php

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Log;

use Symfony\Component\HttpFoundation\Response;

class logFile

{

*/\*\**

*\* Handle an incoming request.*

*\**

*\* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next*

*\*/*

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

    {

*//dd($request);*

*//(Log::info($request->method()));*

       Log::info('Request Input Value: '. json\_encode($request->input()));

       Log::info('Request server: ' . json\_encode($request->server()));

       Log::info('Request Method: ' . $request->method());

       Log::info('Request header: ' . json\_encode($request->header()));

       Log::info('Request cookies: ' .json\_encode($request->cookie()));

       Log::info('Request URL: ' . $request->fullUrl());

        return $next($request);

    }

}

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.

Solution:

Task 5: Controller

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

index(): Display a list of all products.

create(): Display the form to create a new product.

store(): Store a newly created product.

edit($id): Display the form to edit an existing product.

update($id): Update the specified product.

destroy($id): Delete the specified product.

Solution:

<?php

namespace App\Http\Controllers;

use App\Models\Product;

use Illuminate\Http\Request;

class ProductController extends Controller

{

*/\*\**

*\* Display a listing of the resource.*

*\*/*

    public function index()

    {

        $products=Product::all();

        return response()->json($products);

    }

*/\*\**

*\* Show the form for creating a new resource.*

*\*/*

    public function create()

    {

        return ('View Form Page');

    }

*/\*\**

*\* Store a newly created resource in storage.*

*\*/*

    public function store(Request $request)

    {

        $validatedData = $request->validate([

            'name' => 'required',

            'price' => 'required|numeric',

        ]);

        Product::create($validatedData);

        return ("Successfully Stored the data");

    }

*/\*\**

*\* Display the specified resource.*

*\*/*

    public function show(string $id)

    {

*//*

    }

*/\*\**

*\* Show the form for editing the specified resource.*

*\*/*

    public function edit(string $id)

    {

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

        return response()->json($product);

    }

*/\*\**

*\* Update the specified resource in storage.*

*\*/*

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

    {

        $validatedData = $request->validate([

            'name' => 'required',

            'price' => 'required|numeric',

        ]);

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

        $product->update($validatedData);

        return response()->json($product);;

    }

*/\*\**

*\* Remove the specified resource from storage.*

*\*/*

    public function destroy(string $id)

    {

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

        $product->delete();

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

    }

}

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.

Solution:

Route::get("/contact",ContactController::class);

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ContactController extends Controller

{

*/\*\**

*\* Handle the incoming request.*

*\*/*

    public function \_\_invoke(Request $request)

    {

        $validatedData = $request->validate([

            'name' => 'required',

            'email' => 'required|email',

            'message' => 'required',

        ]);

        return response("Invocable Controller")->json($request->input());

    }

}

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.

Solution:

Route::resource('posts',postController::class);

<?php

namespace App\Http\Controllers;

use App\Models\post;

use Illuminate\Http\Request;

class postController extends Controller

{

*/\*\**

*\* Display a listing of the resource.*

*\*/*

    public function index()

    {

        $posts = post::all();

        return view($posts);

    }

*/\*\**

*\* Show the form for creating a new resource.*

*\*/*

    public function create()

    {

        return ('Post created');

    }

*/\*\**

*\* Store a newly created resource in storage.*

*\*/*

    public function store(Request $request)

    {

        $validatedData = $request->validate([

            'title' => 'required',

            'content' => 'required',

        ]);

        post::create($validatedData);

        return response()->with('success', 'Post created successfully');

    }

*/\*\**

*\* Display the specified resource.*

*\*/*

    public function show(string $id)

    {

        $post = Post::findOrFail($id);

        return view($post);

    }

*/\*\**

*\* Show the form for editing the specified resource.*

*\*/*

    public function edit(string $id)

    {

        $post = Post::findOrFail($id);

        return view($post);

    }

*/\*\**

*\* Update the specified resource in storage.*

*\*/*

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

    {

        $validatedData = $request->validate([

            'title' => 'required',

            'content' => 'required',

        ]);

        post::create($validatedData);

        return response()->with('success', 'Post updated successfully');

    }

*/\*\**

*\* Remove the specified resource from storage.*

*\*/*

    public function destroy(string $id)

    {

        $post = Post::findOrFail($id);

        $post->delete();

        return response()->with('success', 'Post deleted successfully');

    }

}

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!".

Solution:

@extends('layout.layout')

@section('content')

<h1>"Welcome to Laravel!".</h1>

@endsection