

Assignment Name: Laravel Concepts Assessment

Student Name: MD Tufik Hasan

Module: 15

Subject: Web Development with PHP & Laravel

Date of Submission: 29/05/2023

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:

```
Route::controller( AssignmentController::class )->group( function () {  
    Route::get( '/register', 'getRegisterForm' );  
    Route::post( '/register', 'register' );  
} );
```

Controller method:

```
function getRegisterForm() {  
    return view( 'register_form' );  
}  
// Task 1: Request Validation  
function register( Request $request ) {  
    $request->validate( [  
        'name'      => 'required|string|min:2',  
        'email'     => 'required|email',  
        'password'  => 'required|string|min:8',  
    ] );  
    if ( $request->filled( ['name', 'email', 'password'] ) ) {  
        return view( 'welcome' );  
    }  
    return redirect()->back();  
}
```

Register form:

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/register'. The page has a dark blue background. In the center, there is a white rounded rectangle containing the title 'Register Form' and the subtitle 'Create your account'. Below the subtitle are three input fields labeled 'Name', 'Email', and 'Password'. At the bottom of the white rectangle is a blue button with the text 'Sign up'.

Task 2: Request Redirect

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

Route:

```
Route::controller( AssignmentController::class )->group( function () {  
    Route::get( '/home', 'redirectDashboard' );  
    Route::get( '/dashboard', 'dashboard' );  
} );
```

Controller method:

```
// Task 2: Request Redirect  
function redirectDashboard() {  
    return redirect( '/dashboard', 302 );  
}
```

```
function dashboard() {  
    return 'Welcome to Dashboard';  
}
```

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.

middleware log info (**LogInformation**):

```
public function handle( Request $request, Closure $next ): Response{
    Log::info( 'Request: ' . $request->method() . ' ' . $request->fullUrl());
    return $next( $request );
}
```

Set (**LogInformation**) middleware as a global middleware

```
protected $middlewareGroups = [
    'web' => [
        \App\Http\Middleware\LogInformation::class,
        \App\Http\Middleware\EncryptCookies::class,
        \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
        \Illuminate\Session\Middleware\StartSession::class,
        \Illuminate\View\Middleware\ShareErrorsFromSession::class,
        // \App\Http\Middleware\VerifyCsrfToken::class,
        \Illuminate\Routing\Middleware\SubstituteBindings::class,
    ],
    'api' => [
        //
        \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
        \Illuminate\Routing\Middleware\ThrottleRequests::class . ':api',
        \Illuminate\Routing\Middleware\SubstituteBindings::class,
    ],
];
```

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.

Create a middleware (**AuthenticateUsers**) & write authentication method:

```
public function handle( Request $request, Closure $next ): Response{
    $email = $request->email;
    $password = $request->password;
    if ( 'tufikhasan05@gmail.com' != $email && '12345' != $password ) {
        return response()->json( ['error' => 'You are not an authenticated user'], 401 );
    }
    return $next( $request );
}
```

Authenticated Routes:

```
Route::middleware( AuthenticateUsers::class )->group( function () {
    // http://127.0.0.1:8000/profile?email=tufikhasan05@gmail.com&password=12345
}
```

```

Route::get( '/profile', function () {
    return "Welcome to profile page";
} );
// http://127.0.0.1:8000/settings?email=tufikhasan05@gmail.com&password=12345
Route::get( '/settings', function () {
    return "Welcome to setting page";
} );
} );

```

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.

Resource - ProductController all method:

```

public function index() {
    return Product::all();
}

public function create() {
    return Product::all();
}

public function store( Request $request ) {
    $product = new Product;
    $product->title = $request->title;
    $product->price = $request->price;

    $product->save();

    return $product;
}

public function show( string $id ) {
    return Product::findOrFail( $id );
}

public function edit( string $id ) {
    return Product::findOrFail( $id );
}

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

```

```

        $product→update( [
            'title' ⇒ $request→title,
            'price' ⇒ $request→price,
        ] );

        return response()→json( ['message' ⇒ 'Successfully updated Product',
$product], 200 );
    }
    public function destroy( string $id ) {
        $product = Product::findOrFail( $id );
        $product→delete();
        return response()→json( ['message' ⇒ 'Successfully deleted Product'], 202
    );
    }
}

```

Api Route::resource:

```
Route::resource( 'product', ProductController::class );
```

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.

ContactController __invoke function:

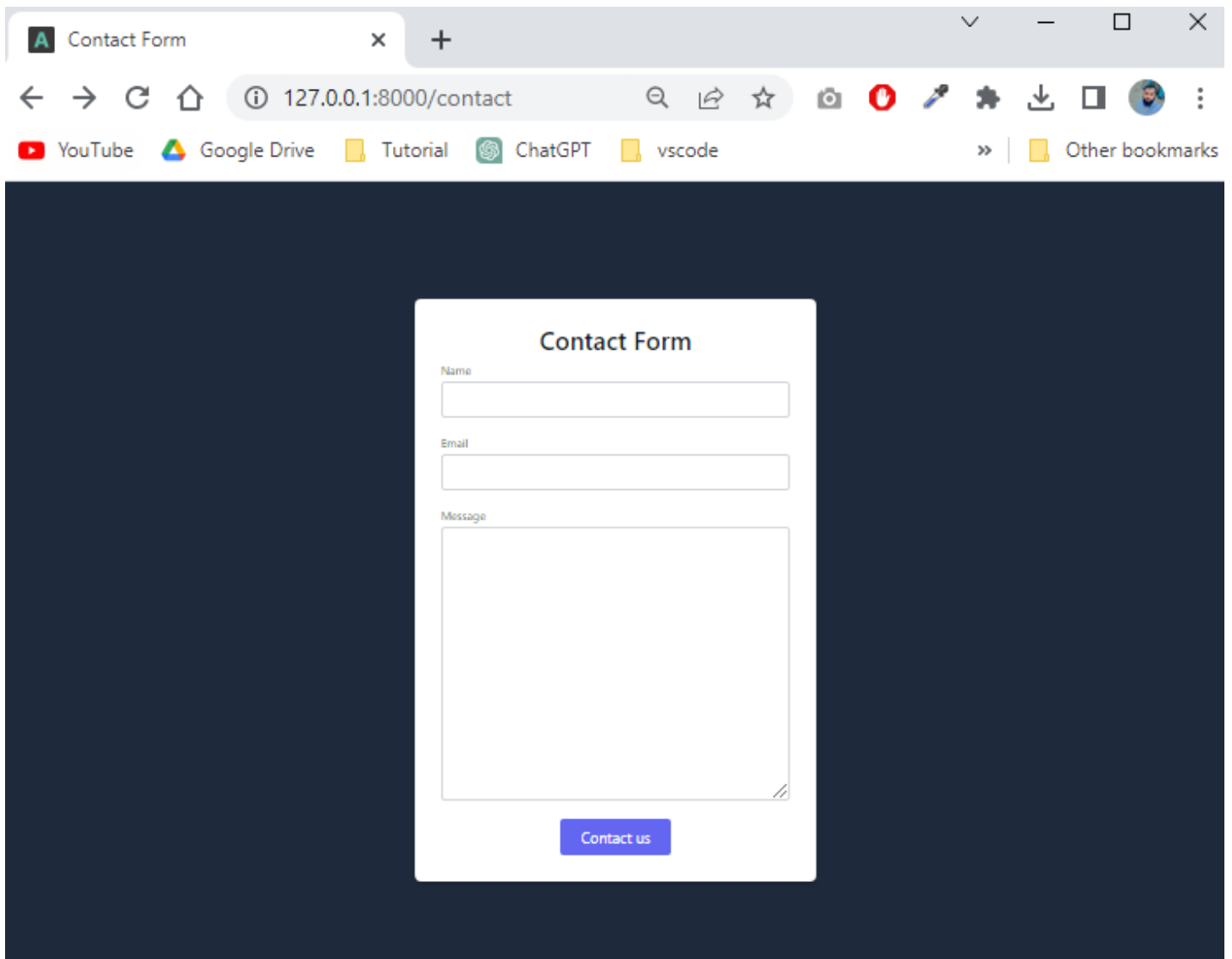
```

public function __invoke( Request $request ) {
    // Validating the request data
    $request→validate( [
        'name'      ⇒ 'required|min:2',
        'email'     ⇒ 'required|email',
        'message'   ⇒ 'required|min:20',
    ] );

    if ( $request→filled( ['name', 'email', 'message'] ) ) {
        // Return a JSON response
        return response()→json( [
            'name'      ⇒ $request→input( 'name' ),
            'email'     ⇒ $request→input( 'email' ),
            'message'   ⇒ $request→input( 'message' ),
            'Confirmation' ⇒ 'Message sent successfully! We will be in touch
very soon.',
        ] );
    }
    return redirect()→back();
}

```

Contact form:



The screenshot shows a web browser window with the title 'Contact Form'. The address bar displays '127.0.0.1:8000/contact'. The browser's bookmark bar includes links to YouTube, Google Drive, Tutorial, ChatGPT, and vscode. The contact form itself is centered on a dark blue background. It has a title 'Contact Form' and three input fields: 'Name', 'Email', and 'Message'. The 'Message' field is a larger text area. A blue button labeled 'Contact us' is positioned at the bottom of the form.

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.

Resource -PostController all method:

```
public function index() {  
    return Post::all();  
}  
  
public function create() {  
    return Post::all();  
}  
  
public function store( Request $request ) {  
    $posts = new Post;
```

```

        $posts→title = $request→title;
        $posts→description = $request→description;

        $posts→save();

        return $posts;
    }
    public function show( string $id ) {
        return Post::findOrFail( $id );
    }

    public function edit( string $id ) {
        return Post::findOrFail( $id );
    }

    public function update( Request $request, string $id ) {
        $posts = Post::findOrFail( $id );
        $posts→update( [
            'title'      ⇒ $request→title,
            'description' ⇒ $request→description,
        ] );

        return response()→json( ['message' ⇒ 'Successfully updated Post', $posts], 200
);
    }
    public function destroy( string $id ) {
        $posts = Post::findOrFail( $id );
        $posts→delete();
        return response()→json( ['message' ⇒ 'Successfully deleted Post'], 202 );
    }
}

```

Api Route::resource:

```
Route::resource( 'posts', 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:

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

View template:

Welcome

✕

+

⌵


—

□

✕

⬅ ➡ ↺ 🏠 ⓘ 127.0.0.1:8000 📄 ☆ 📷 🔴 🖋 ⚙ ⬇ 🖼 👤 ⋮

📺 YouTube 📁 Google Drive 📁 Tutorial 🤖 ChatGPT 📁 vscode 📁 HTML,CSS & Shapes » | 📁 Other bookmarks



Home

Register

Dashboard

Profile

Settings

Welcome to Laravel!