Kacey Morris

Ana Sanchez

CST:256 Activity 6

Professor Hughes

Part 1: Security Research and Middleware Integration

1. Security Service Middleware Service

Using the built in Laravel auth middleware that checks if a user is authenticated.

Text

Description automatically generated

The Authenticate middleware inherits from another Authenticate middleware located in the Laravel vendor directory. It takes the request of the current user and if they are currently authenticated it will then redirect them back to ‘login’.

Built in logger that will log when a user passes or fails the authenticate middleware.

Also added a “isAdmin” and “isSuspended” middleware for more security for certain admin pages and to not allow suspended users to access any of the page.

Text

Description automatically generated

Text

Description automatically generated

2. Logging Service

A screen shot of a computer

Description automatically generated with low confidence

Logging Service interface

Text

Description automatically generated

Logging Service Implementation Class

3. Logging Service Provider

Text

Description automatically generated

Logging Service Provider that registers MyLogger3, we are binding a singleton

Text

Description automatically generated

Added our LoggingService provider to the list of providers under the confid/app.php file.

4. Team Project Integration

Our clc project has the built in auth middleware as well as the isAdmin and isSuspended middleware implemented into it. We also injected the Logging Service into every middleware and controller in our project. An example of this in the LoginController.

Text

Description automatically generated

This means that when a user visits any controller in the app, we will have a log of that action as well as every authentication and every attempt to visit an admin page. While we are only showing the implementation of the Logging Service injection for the LoginController, below is a screenshot of the log for a user visiting multiple pages and entering multiple pages.

Text

Description automatically generated

As you can see the user logs and then edits their profile, joins an affinity group, searches for a job, applies for a job, and then logs out.

Text

Description automatically generated

This is a user accessing a admin page because they are authenticated as an admin with the isAdmin middleware

Text

Description automatically generated

This a user who is not authenticated as a admin and so they fail the isAdmin middleware check.

5. Security Research

1. **Research how authentication is supported using the built-in features of the Laravel framework. Outline how authentication is supported and provided a detailed, 100- to 200-word write-up for what would be required to integrate the security framework into your Login Module.**

The built in Laravel authentication features allow for a quick startup of a project. This includes the building of login and registration views, middleware to check if a user is authorized to view select pages, and a backend database integration. The backbone of the Laravel authentication consists of “guards” and “providers”. Guards are a method of authenticating a user through sessions which maintains states using session storage and cookies. This ensures that a user is are authenticated for each request. Providers define how users are going to be retrieved from the database. The built-in authentication model comes with Eloquent, a database query builder that makes it hassle free to access Users and other models from your database. All the components for the authentication are taken care for a user including the controllers such as LoginController and RegistrationController and all required routes. These controllers inherit authentication classes which allows the user to override different authentication actions such as redirect routes, logging in, logging out, and other actions. To activate the built-in authentication features for Laravel a developer just needs to run “php artisan make:auth”. This will create everything needed even the database tables.

1. **Research how authorization is supported using the built-in features of the Laravel framework. Outline how authorization is supported and provided a detailed, 100- to 200-word write-up for what would be required to integrate the security framework into your Administration Module to support the Admin Role.**

Other than including bult-in authentication services, Laravel also provides support for a built-in way to authorize users for select resources and pages. An example of needing to create authorization in an application is restricting certain users from accessing administration resources or restricting access to resources and change to delete models from the database. We needed to create authorization for our JobPostingSite for restricting our admin pages and restricting users that were suspended. Authorization is handled through two different methods: gates and policies. According to the Laravel documentation, gates and policies can be thought as similar to routes and controllers. Gates are a simple, closure like, method to authorization and policies focus on grouping logic around a particular model or resource. It is not required to stick to only one of the two methods of authorization and depending on what your goals are, it might be beneficial to mix up which method is used. For our JobPostingSite, we created two policies: IsAdmin and isSuspended. We made these policies by running the command “php artisan make:policy isAdmin –model User). These policies needed to be Registered in the App\Providers\AuthServiceProvider policies property.

1. **Create a logical drawing for how a Login Process would work using the Laravel security framework.**

