Laravel

Outlines of Discussion

- Authentication in Laravel
- Authorization in Laravel
- Roles and Permissions
- Roles and Permissions Example

Authentication In Laravel

- Laravel provides a powerful, out-of-the-box system for **Authentication** (identifying users) and **Authorization** (restricting or allowing access to certain resources). Below is a comprehensive, step-by-step explanation:

Authentication vs. Authorization

1. Understanding Authentication vs. Authorization

Authentication:

- It verifies the user's identity (e.g., login).
- Determines *who* the user is.

Authorization:

- It decides what actions or resources a user can access based on their roles or permissions.
- Determines *what* the user can do.

Setting Up Authentication

2. Setting Up Authentication

Step 1: Install Laravel

- Make sure you have Laravel installed.
- Use the command:

bash Copy code

```
composer create-project laravel/laravel example-app
cd example-app
```

Cont. ...

Step 2: Add Authentication Scaffolding

Laravel offers multiple ways to implement authentication scaffolding. Choose one based on your project requirements:

1. **Laravel Breeze** (lightweight):

bash Copy code

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install && npm run dev
php artisan migrate
```

Cont. ...

2. **Laravel Jetstream** (advanced features like 2FA):

bash Copy code

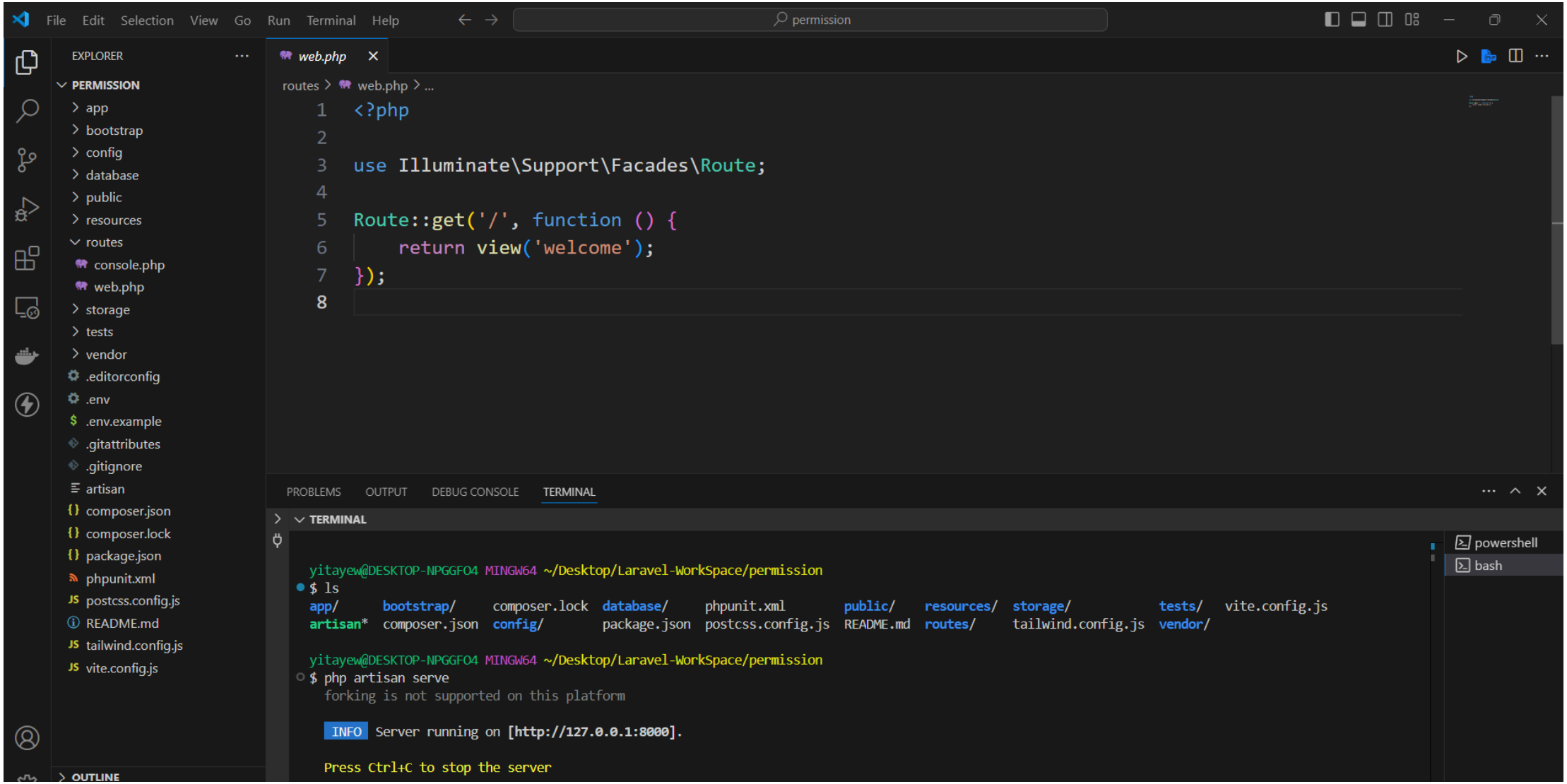
```
composer require laravel/jetstream
php artisan jetstream:install livewire
npm install && npm run dev
php artisan migrate
```

3. **Laravel Fortify** (backend-only):

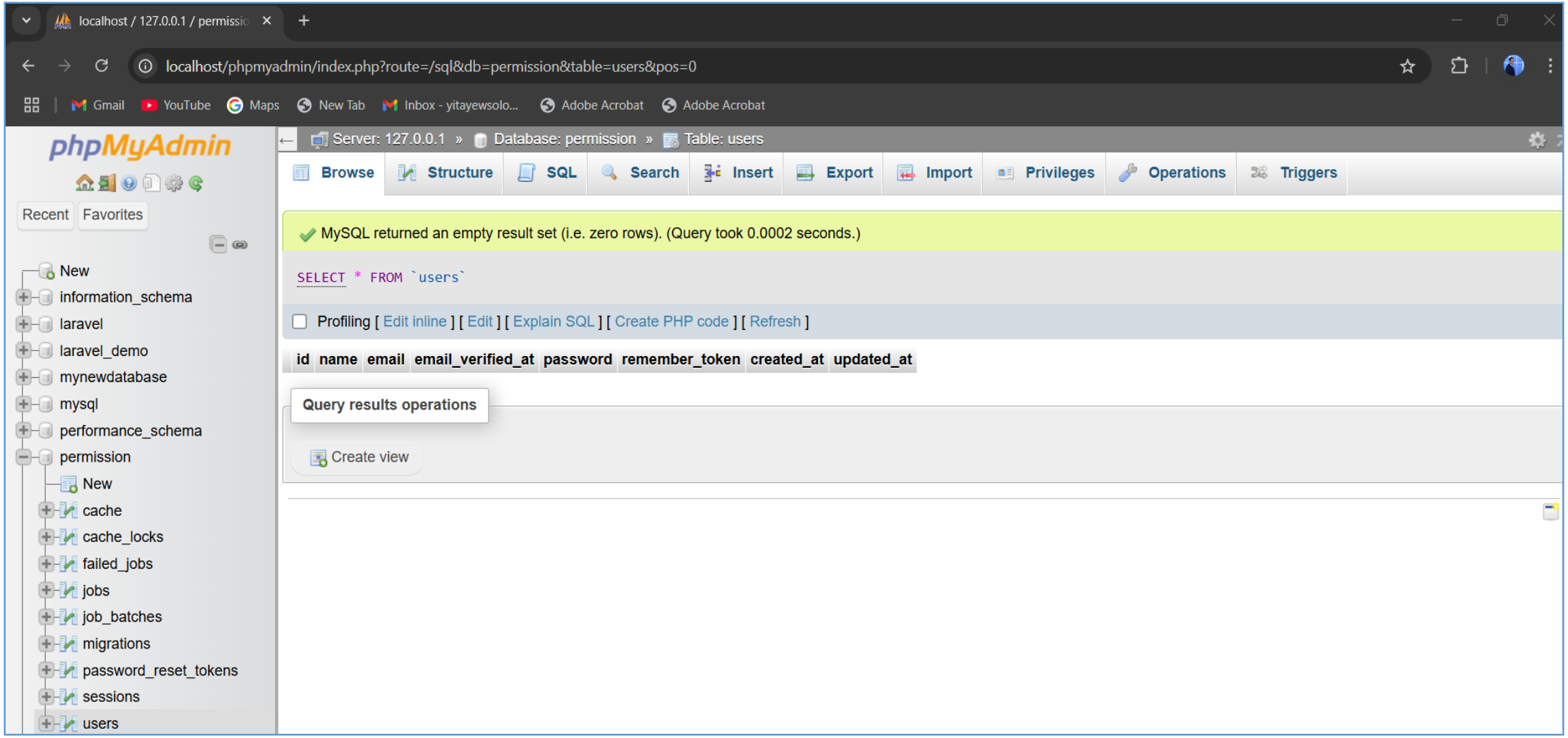
bash Copy code

```
composer require laravel/fortify
php artisan vendor:publish --provider="Laravel\Fortify\FortifyServiceProvider"
php artisan migrate
```

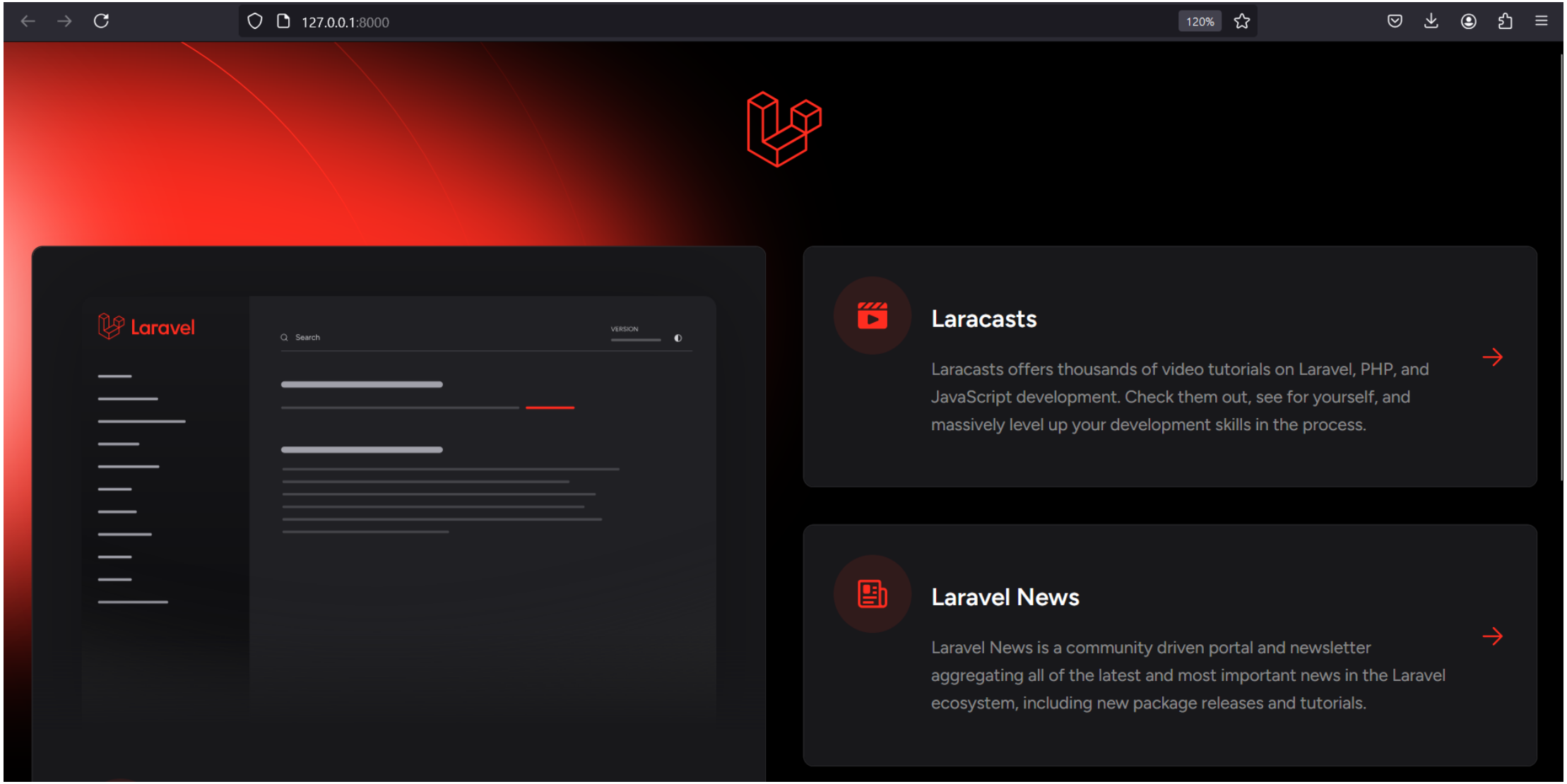
Authentication In laravel (Creating New Project)



Connecting to Database



Configuring Laravel Breeze



Laravel Breeze

- Laravel Breeze is a simple, lightweight starter kit for Laravel that provides a basic authentication system. It’s ideal for quickly scaffolding authentication functionality in your Laravel application, with minimal setup and clean implementation. Laravel Breeze comes with views built using Blade templates, along with an option for Inertia.js and Vue or React if you prefer modern frontend tools.

Features of Laravel Breeze

Features of Laravel Breeze

1. **Authentication:**
    - User registration
    - Login/logout
    - Password reset
    - Email verification
2. **Frontend:**
    - Blade templates for server-side rendering.
    - Optionally, Vue.js or React for client-side interactivity via Inertia.js.
3. **Minimal dependencies** to keep the implementation simple and lightweight.
4. **Tailwind CSS** integration for quick and b tiful styling.

Installing Breeze

Installing Laravel Breeze

Step 1: Install Laravel

Ensure you have a Laravel application installed:

bash Copy code

```
composer create-project laravel/laravel my-app
cd my-app
```

Step 2: Require Breeze Package

Install the Laravel Breeze package:

bash Copy code

```
composer require laravel/breeze --dev
```

Cont. ...

Step 3: Install Breeze

Run the Breeze installation command:

bash Copy code

```
php artisan breeze:install
```

By default, Breeze installs with Blade templates. If you want to use a frontend framework like Vue or React, you can specify it:

- For Vue:

bash Copy code

```
php artisan breeze:install vue
```

Cont. ...

Step 4: Install Dependencies

After installing Breeze, install the frontend dependencies:

bash Copy code

```
npm install
npm run dev
```

Step 5: Run Migrations

Breeze comes with the required migrations for users. Run the migrations to create the necessary database tables:

bash Copy code

```
php artisan migrate
```

Cont. ...

Using Laravel Breeze

1. Login and Registration:
    - Breeze provides routes, controllers, and views for authentication:
        - `/register`
        - `/login`
        - `/forgot-password`
        - `/reset-password`
        - `/email/verify`

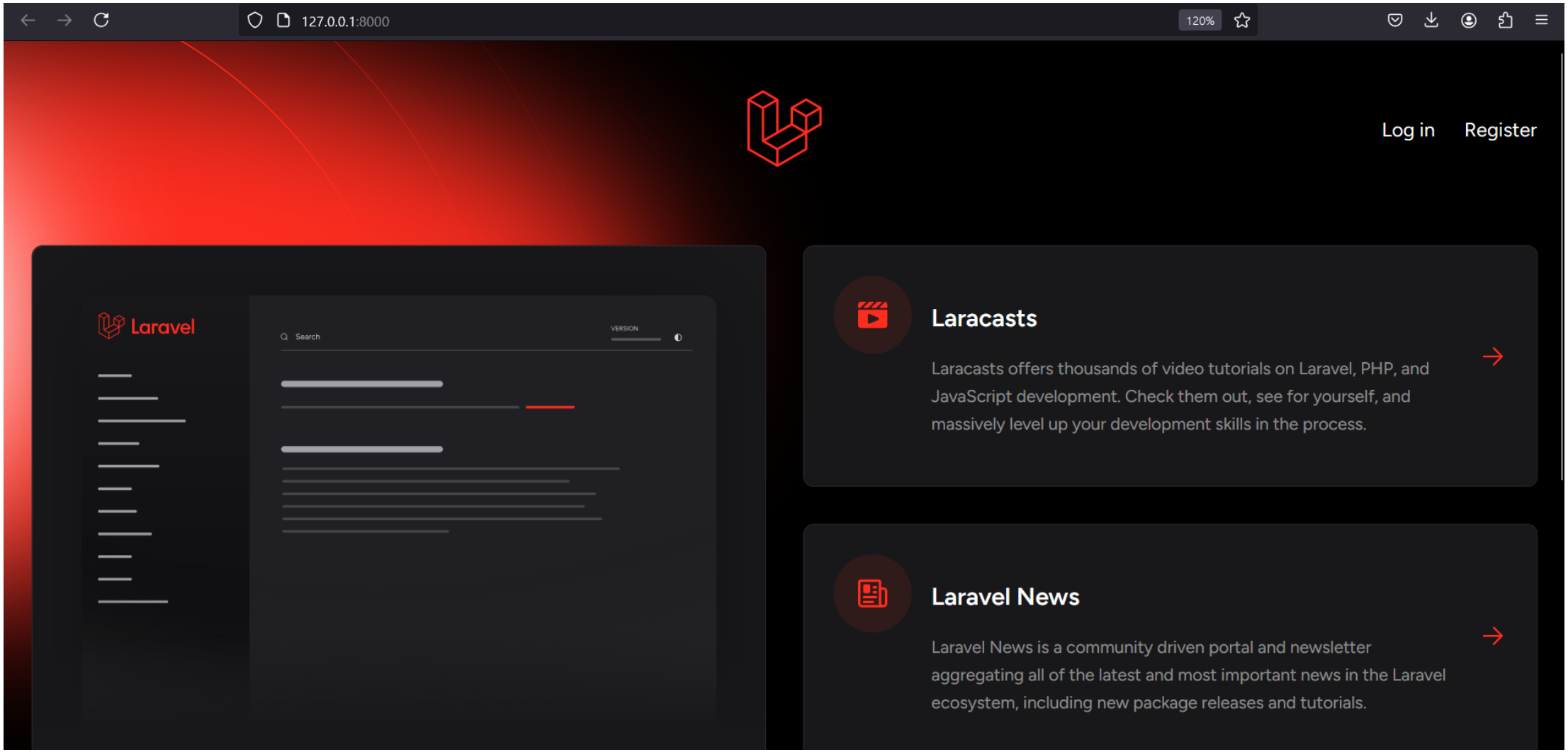
Cont. ...

2. **Route Protection:** Use the `auth` middleware to protect routes:

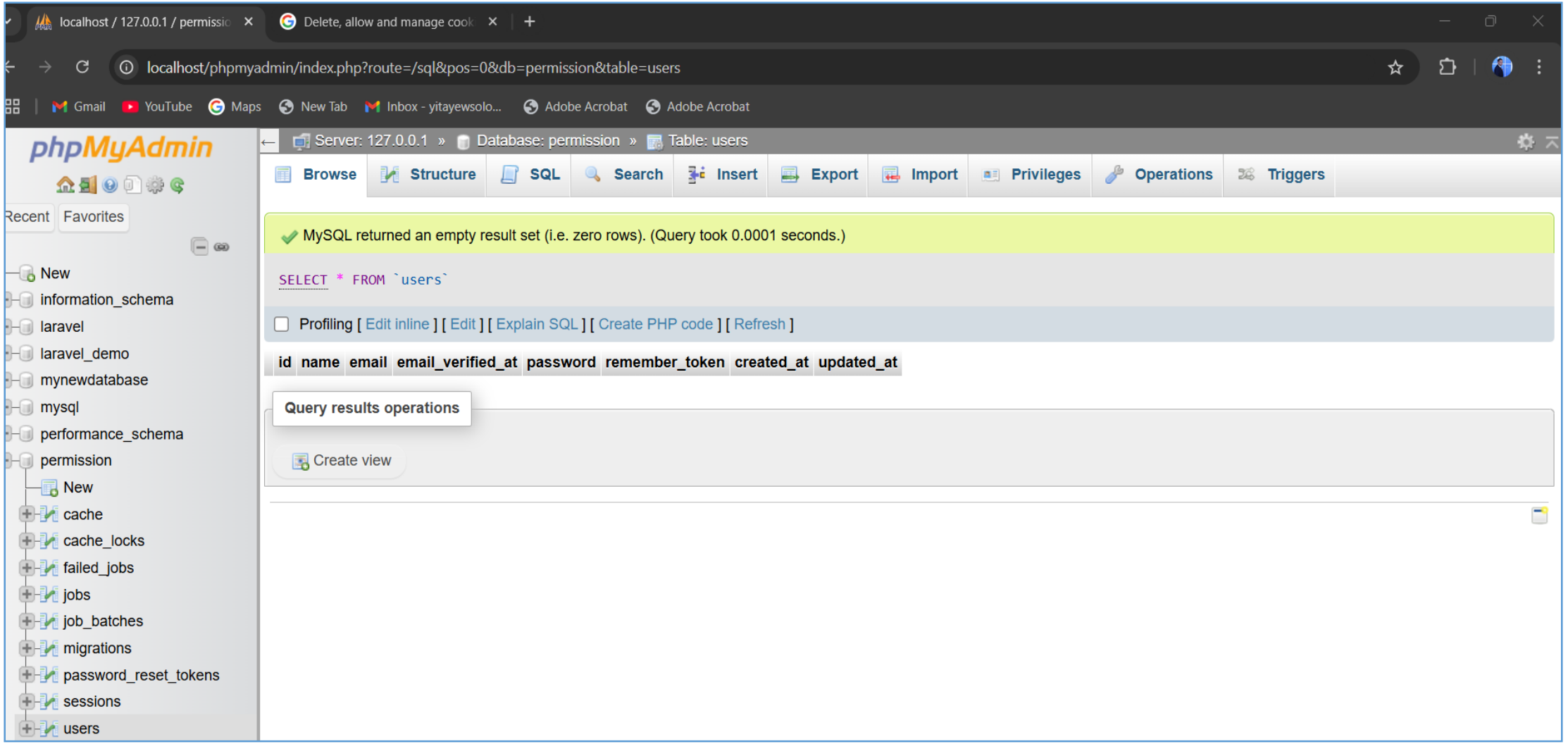
```php
Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth'])->name('dashboard');
```

3. **Customization:**
   - Edit the authentication views in `resources/views/auth`.
   - Modify the controller logic in `app/Http/Controllers/Auth`.
4. **Frontend Customization:**
   - Breeze uses Tailwind CSS for styling, which you can extend by editing the `tailwind.config.js` file.

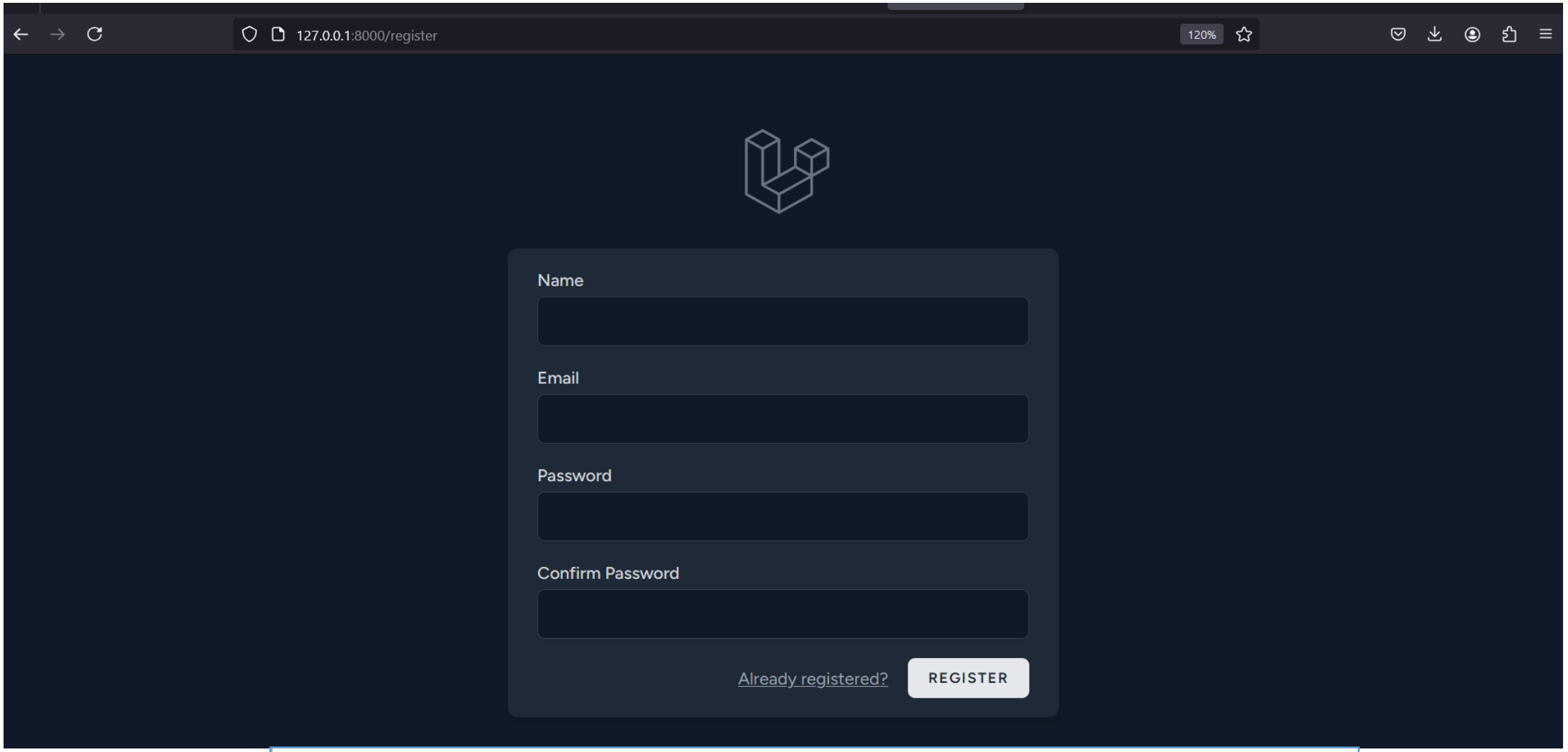
Output (After Breeze Installation)



Database Tables After Migration

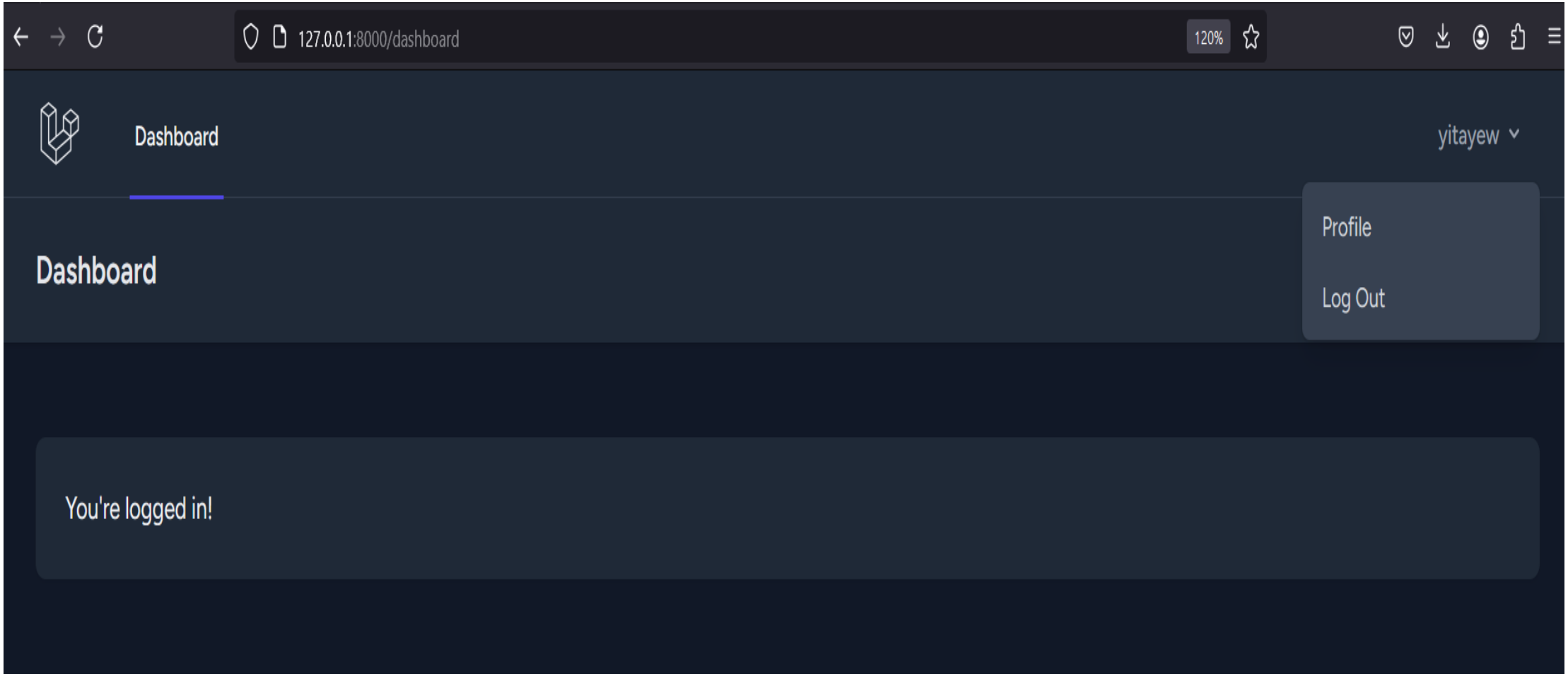


Registration Page (http://127.0.0.1:8000/register)

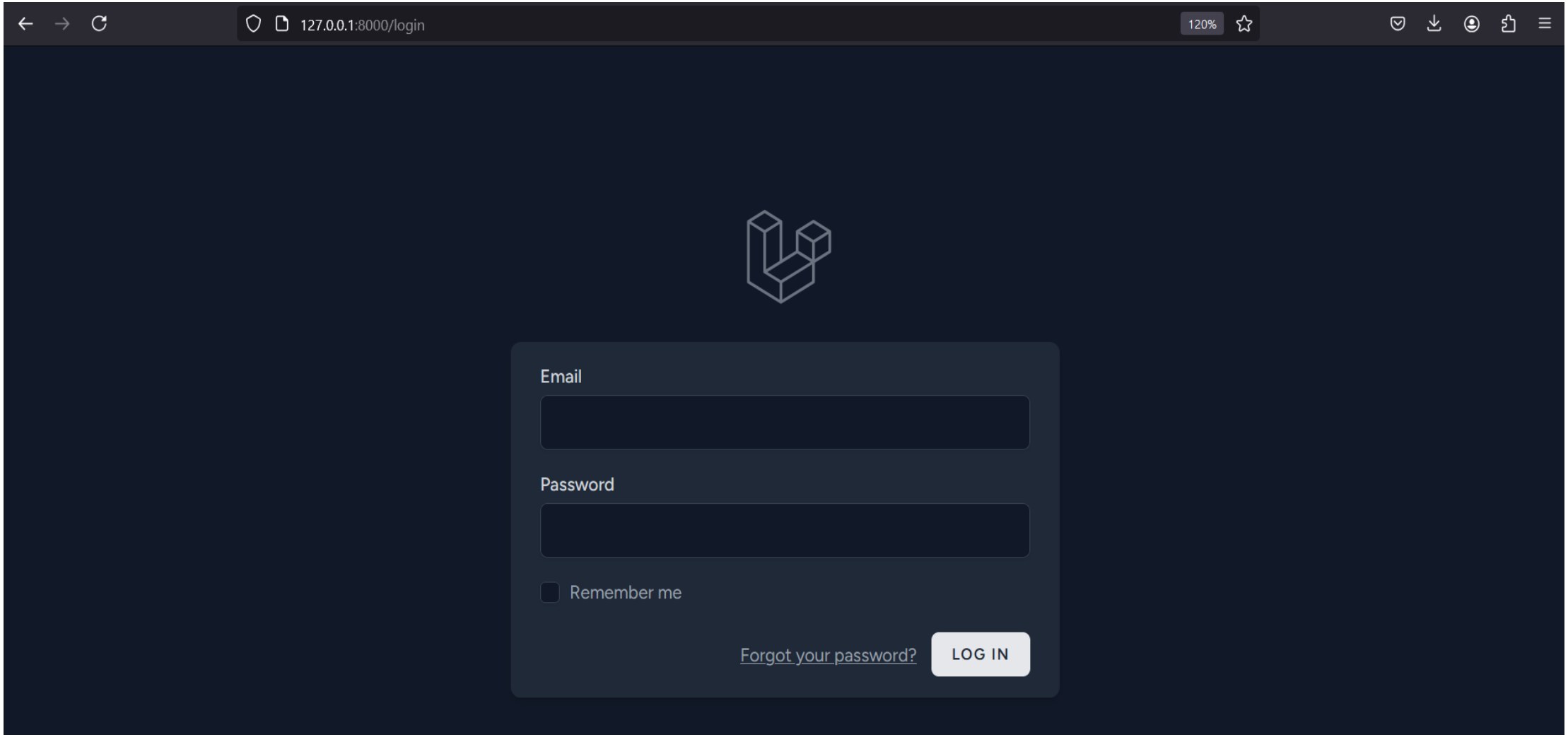


	id	name	email	email_verified_at	password	remember_token	cre
Edit Copy Delete	1	yitayew	yitayewsolomon3@gmail.com	NULL	$2y$12$CEs2jfiMRPR1HqJGQqYHPOJD5/lbPHt5m09h/UySorB...	NULL	202 09:

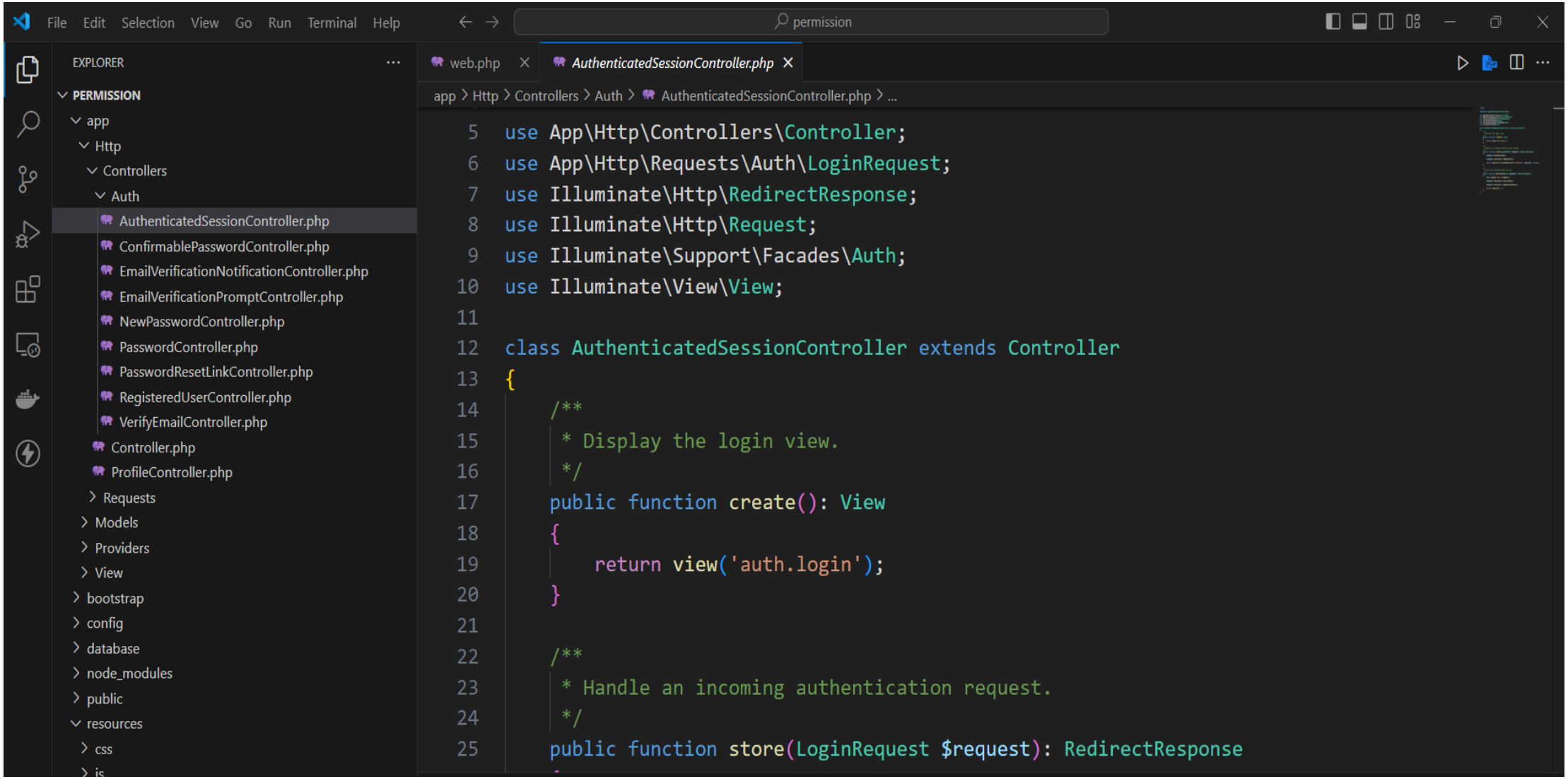
User Loge in (http://127.0.0.1:8000/dashboard)



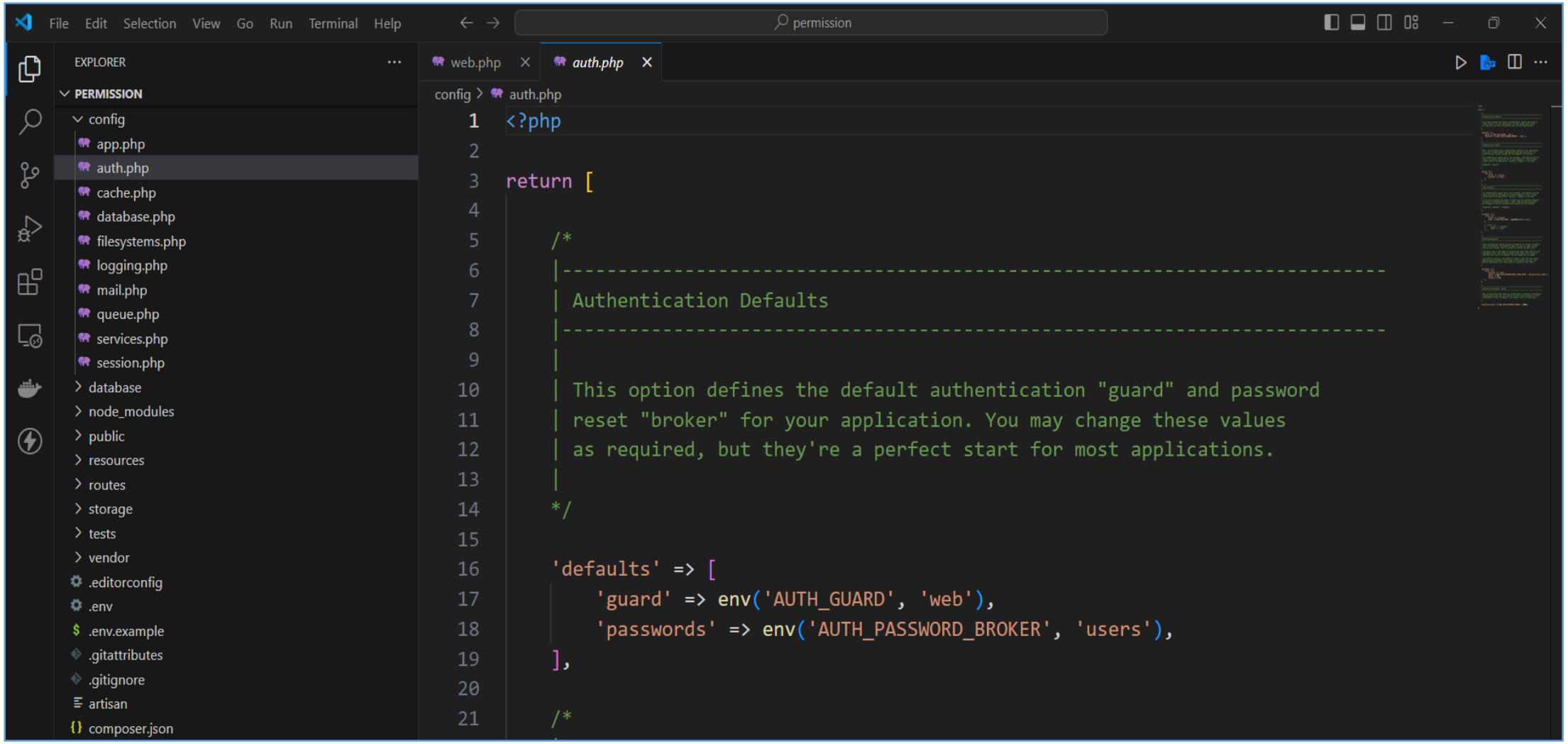
Login(http://127.0.0.1:8000/login)



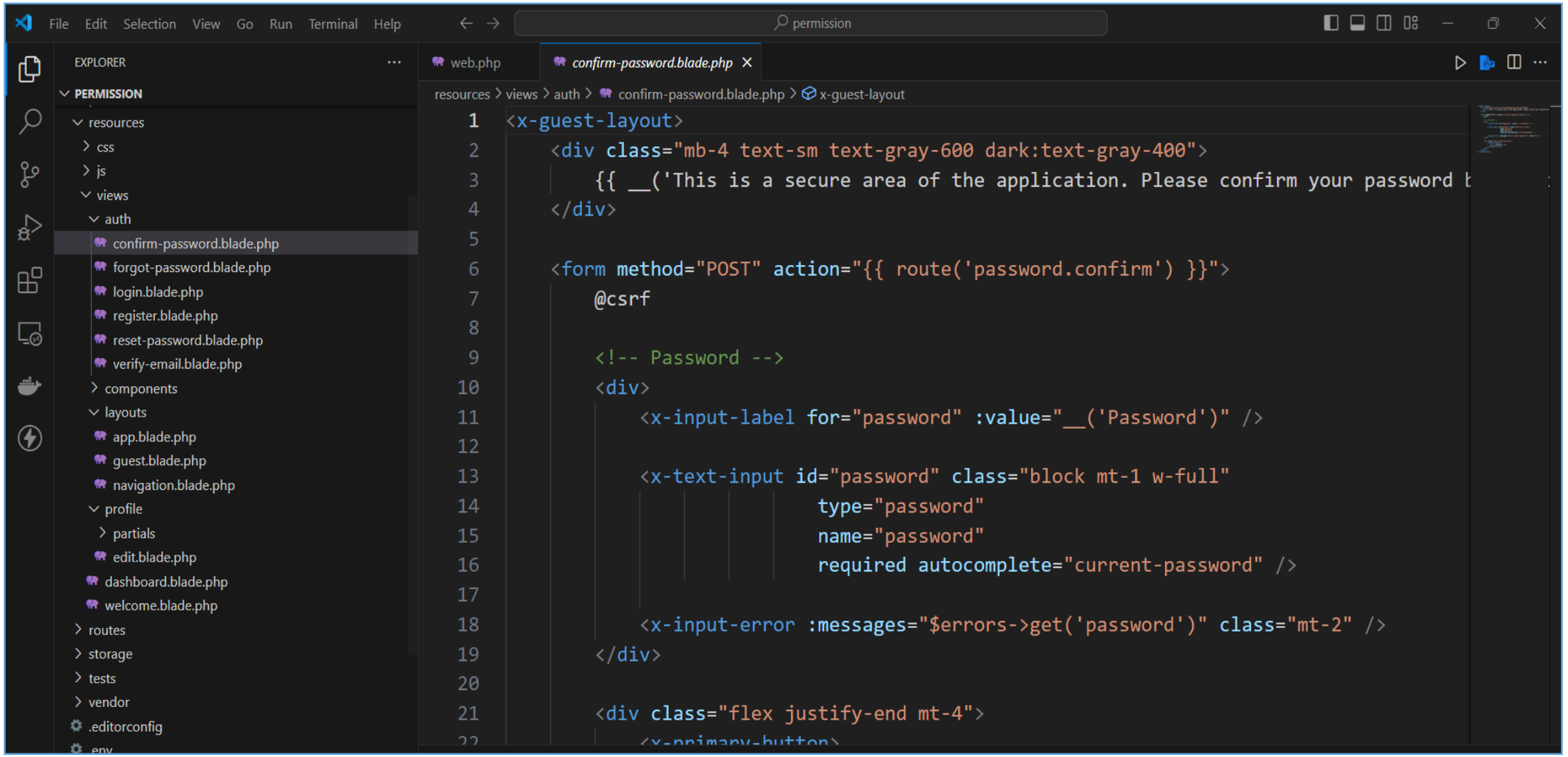
Project Structure (After Installing Breeze)



Config


File Edit Selection View Go Run Terminal Help
permission
EXPLORER
PERMISSION
config
app.php
auth.php
cache.php
database.php
filesystems.php
logging.php
mail.php
queue.php
services.php
session.php
database
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
web.php
auth.php
config > auth.php
<?php
return [
/*
| Authentication Defaults
| This option defines the default authentication "guard" and password
| reset "broker" for your application. You may change these values
| as required, but they're a perfect start for most applications.
*/
'defaults' => [
'guard' => env('AUTH_GUARD', 'web'),
'passwords' => env('AUTH_PASSWORD_BROKER', 'users'),
],
/*

Resource


File Edit Selection View Go Run Terminal Help
permission
EXPLORER
PERMISSION
resources
css
js
views
auth
confirm-password.blade.php
forgot-password.blade.php
login.blade.php
register.blade.php
reset-password.blade.php
verify-email.blade.php
components
layouts
app.blade.php
guest.blade.php
navigation.blade.php
profile
partials
edit.blade.php
dashboard.blade.php
welcome.blade.php
routes
storage
tests
vendor
.editorconfig
.env
web.php
confirm-password.blade.php
resources > views > auth > confirm-password.blade.php > x-guest-layout
<x-guest-layout>

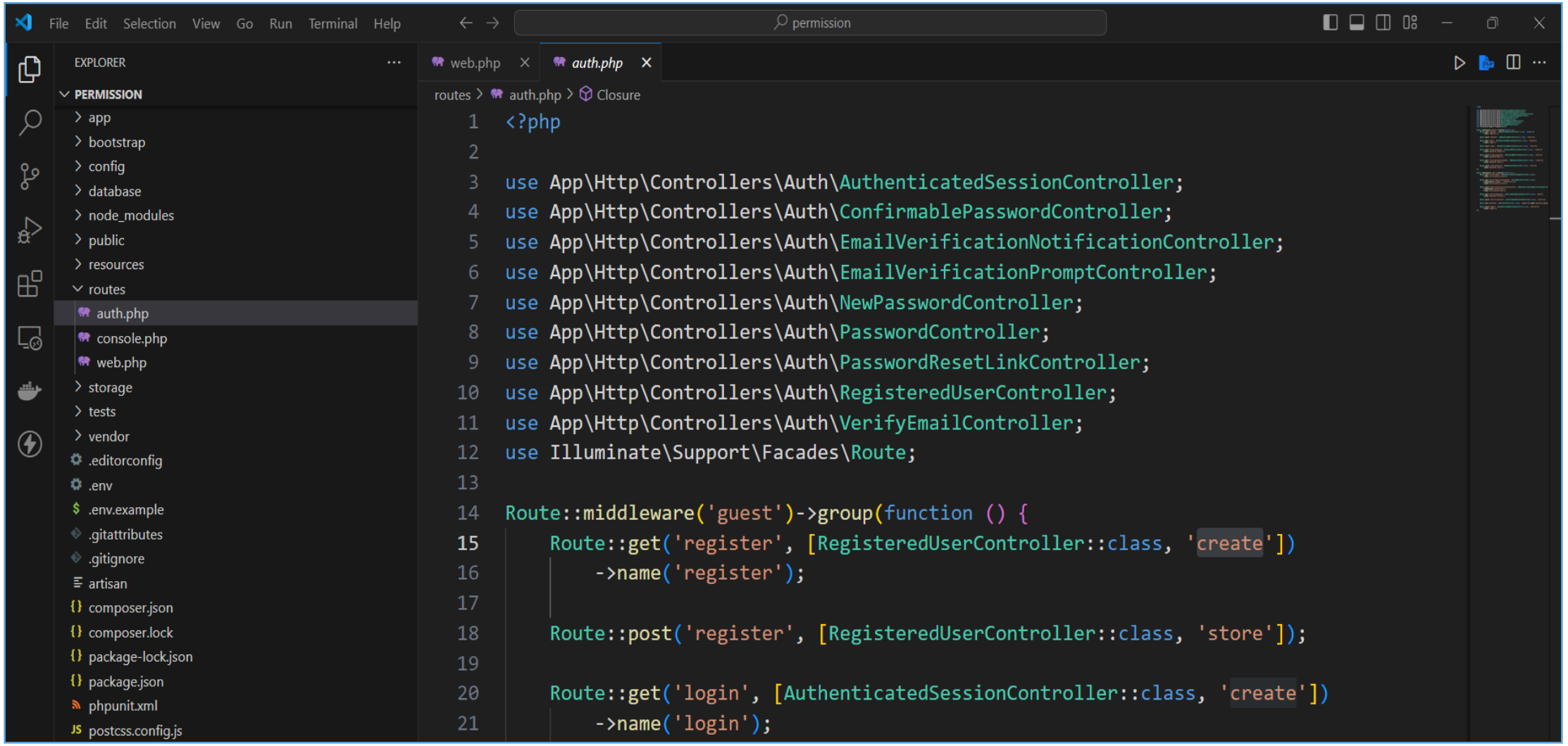
{{ __('This is a secure area of the application. Please confirm your password b

<form method="POST" action="{{ route('password.confirm') }}">
@csrf
<!-- Password -->

<x-input-label for="password" :value="__('Password')" />
<x-text-input id="password" class="block mt-1 w-full"
type="password"
name="password"
required autocomplete="current-password" />
<x-input-error :messages="$errors->get('password')" class="mt-2" />

<x-primary-button>

Route


File Edit Selection View Go Run Terminal Help
permission
EXPLORER
PERMISSION
app
bootstrap
config
database
node_modules
public
resources
routes
auth.php
console.php
web.php
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
web.php
auth.php
routes > auth.php > Closure
<?php
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;
Route::middleware('guest')->group(function () {
Route::get('register', [RegisteredUserController::class, 'create'])
->name('register');
Route::post('register', [RegisteredUserController::class, 'store']);
Route::get('login', [AuthenticatedSessionController::class, 'create'])
->name('login');

Authorization (Roles and Permissions)

- Roles and Permissions in Laravel allow you to manage **user access control** efficiently. Laravel doesn't include a built-in role and permission system, but popular packages like **Spatie Laravel** Permission make it straightforward to implement.

Key Concepts of Roles and Permissions

Key Concepts of Roles and Permissions

1. **Role:** A named group of permissions, e.g., *Admin*, *Editor*, *User*.
2. **Permission:** A specific action that a role or user can perform, e.g., *edit articles*, *delete users*, *view dashboard*.
3. **Role-Based Access Control (RBAC):** Assigning roles to users to determine what actions they can perform.
4. **Direct Permissions:** Assign permissions directly to users or associate them with roles.

Laravel 11 ACL - Roles and Permissions Example

Step for Laravel 11 ACL - Roles and Permissions Example

- **Step 1:** Install Laravel 11
- **Step 2:** Install spatie/laravel-permission Package
- **Step 3:** Create Product Migration
- **Step 4:** Create Models
- **Step 5:** Add Middleware
- **Step 6:** Create Authentication
- **Step 7:** Create Routes
- **Step 8:** Add Controllers
- **Step 9:** Add Blade Files
- **Step 10:** Create Seeder For Permissions and AdminUser
- **Run Laravel App**

Step 1 (Installing Laravel)

Step 1: Install Laravel 11

First of all, we need to get a fresh Laravel 11 version application using the command below because we are starting from scratch. So, open your terminal or command prompt and run the command below:

```
composer create-project laravel/laravel example-app
```

Step two (Install Spatie)

Step 2: Install spatie/laravel-permission Package

Now, we require to install the Spatie package for ACL; that way, we can use its method. Also, we will install the form collection package. So, open your terminal and run the below command.

```
composer require spatie/laravel-permission
```

We can also customize changes on the Spatie package, so if you also want to make changes, you can fire the command below and get the config file in config/permission.php and migration files.

```
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
```

Now you can see the permission.php file and one migration. So you can run the migration using the following command:

```
php artisan migrate
```

Step 3 (Creating Migrations)

Step 3: Create Product Migration

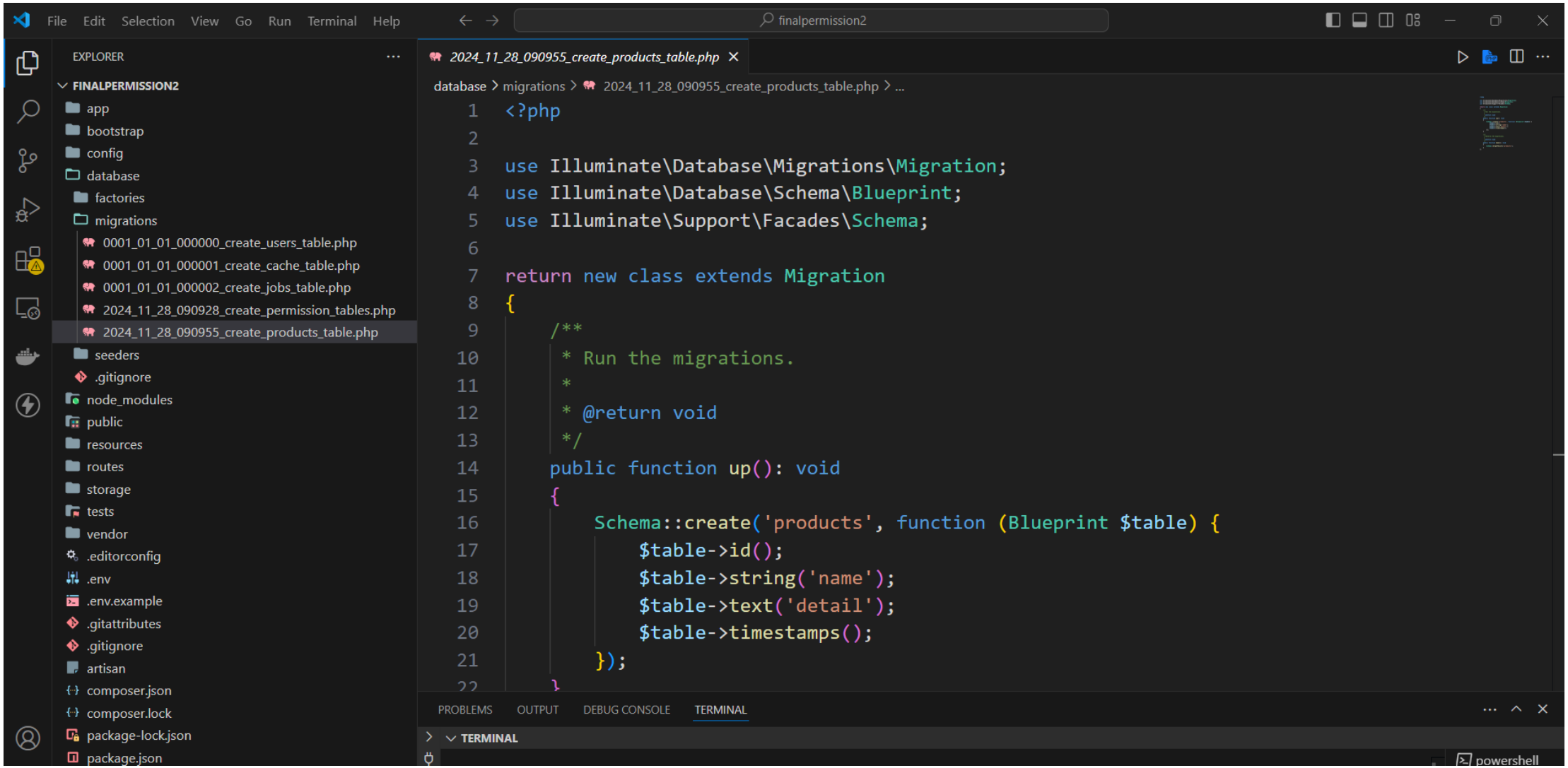
In this step, we have to create three migrations for the products table using the below command:

```
php artisan make:migration create_products_table
```

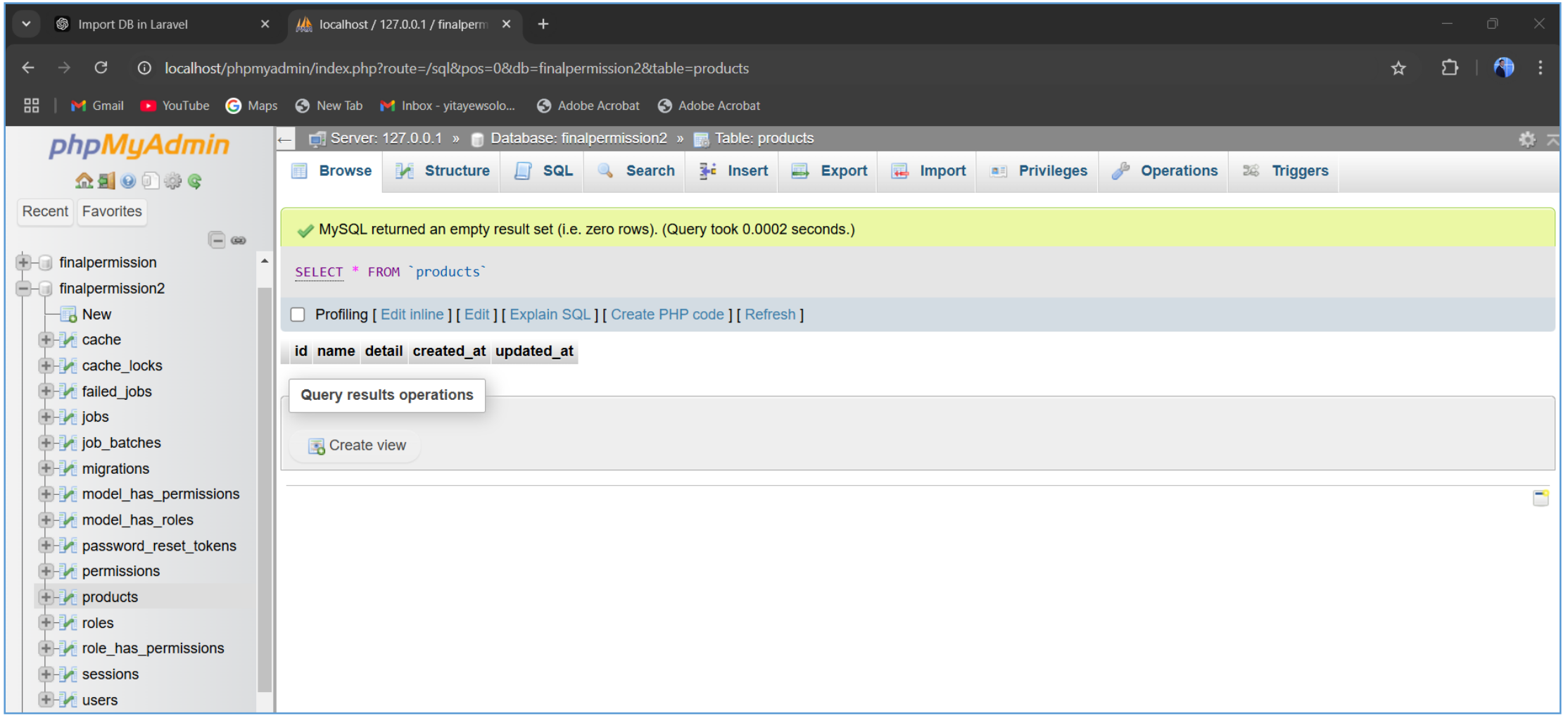
Now, run migration:

```
php artisan migrate
```

Cont. ...


File Edit Selection View Go Run Terminal Help
finalpermission2
EXPLORER
FINALPERMISSION2
app
bootstrap
config
database
factories
migrations
0001_01_01_000000_create_users_table.php
0001_01_01_000001_create_cache_table.php
0001_01_01_000002_create_jobs_table.php
2024_11_28_090928_create_permission_tables.php
2024_11_28_090955_create_products_table.php
seeders
.gitignore
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
2024_11_28_090955_create_products_table.php
database > migrations > 2024_11_28_090955_create_products_table.php > ...
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
/**
* Run the migrations.
*
* @return void
*/
public function up(): void
{
Schema::create('products', function (Blueprint $table) {
$table->id();
$table->string('name');
$table->text('detail');
$table->timestamps();
});
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
powershell

Cont. ...

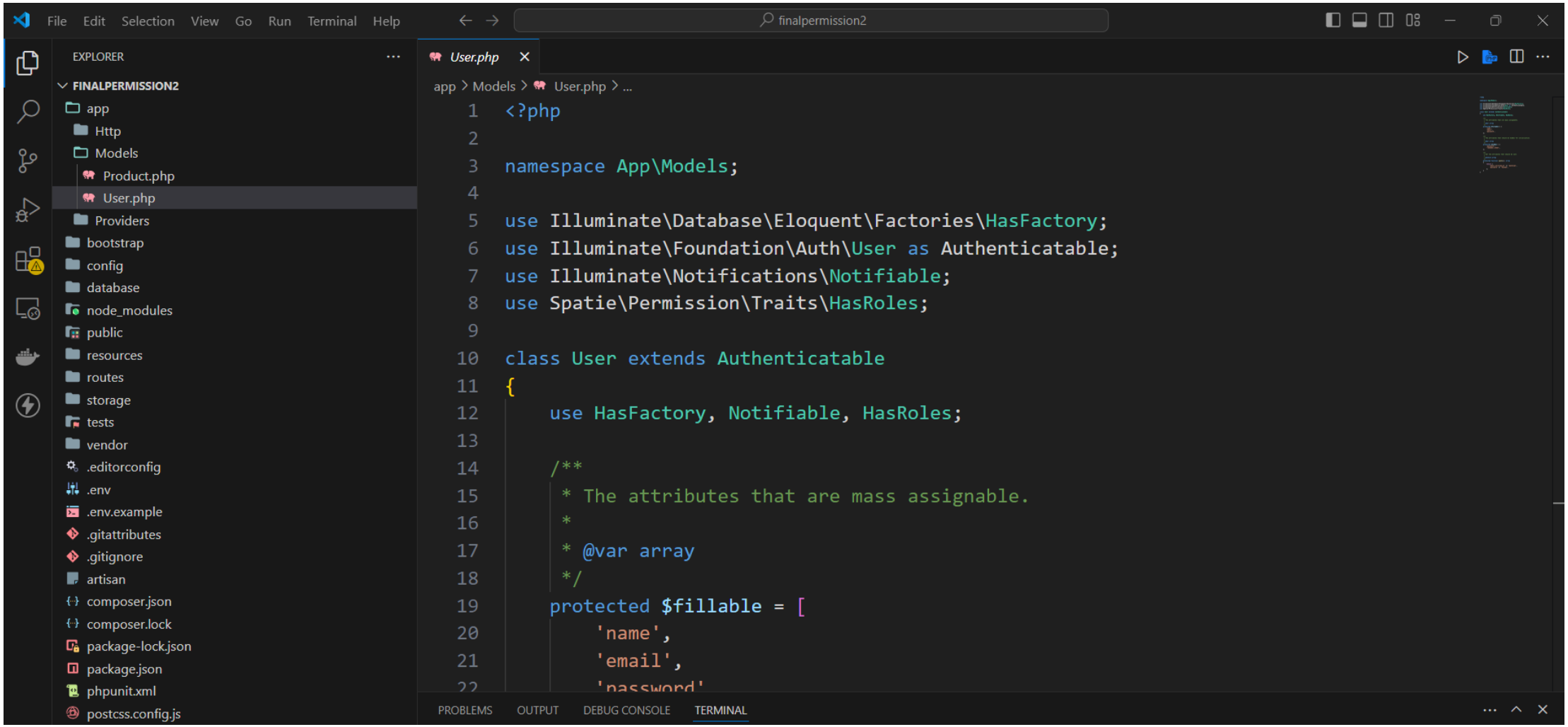

Import DB in Laravel
localhost / 127.0.0.1 / finalperm
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=finalpermission2&table=products
Gmail
YouTube
Maps
New Tab
Inbox - yitayewsolo...
Adobe Acrobat
Adobe Acrobat
phpMyAdmin
Recent
Favorites
finalpermission
finalpermission2
New
cache
cache_locks
failed_jobs
jobs
job_batches
migrations
model_has_permissions
model_has_roles
password_reset_tokens
permissions
products
roles
role_has_permissions
sessions
users
Server: 127.0.0.1 » Database: finalpermission2 » Table: products
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
SELECT * FROM `products`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
id
name
detail
created_at
updated_at
Query results operations
Create view

Step 4 (Creating Models)

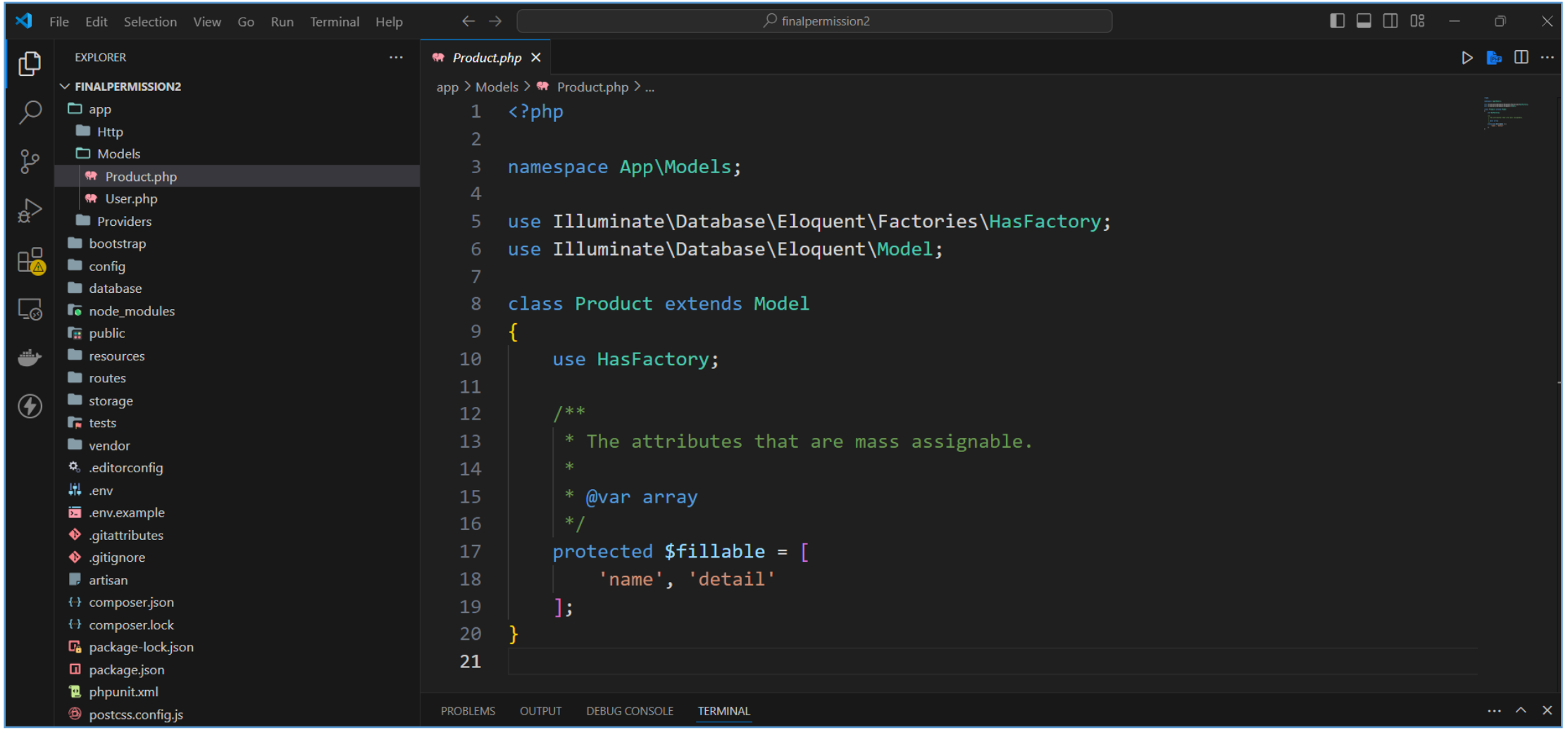
Step 4: Create Models

In this step, we have to create a model for the User and Product tables. If you get a fresh project, you will have a User model, so just replace the code, and for the other, you should create one.

Model (User.php)



Model (Product.php)



Step 5 (Adding Middleware)

Step 5: Add Middleware

Spatie package provides its in-built middleware, which we can use simply and is displayed as below:

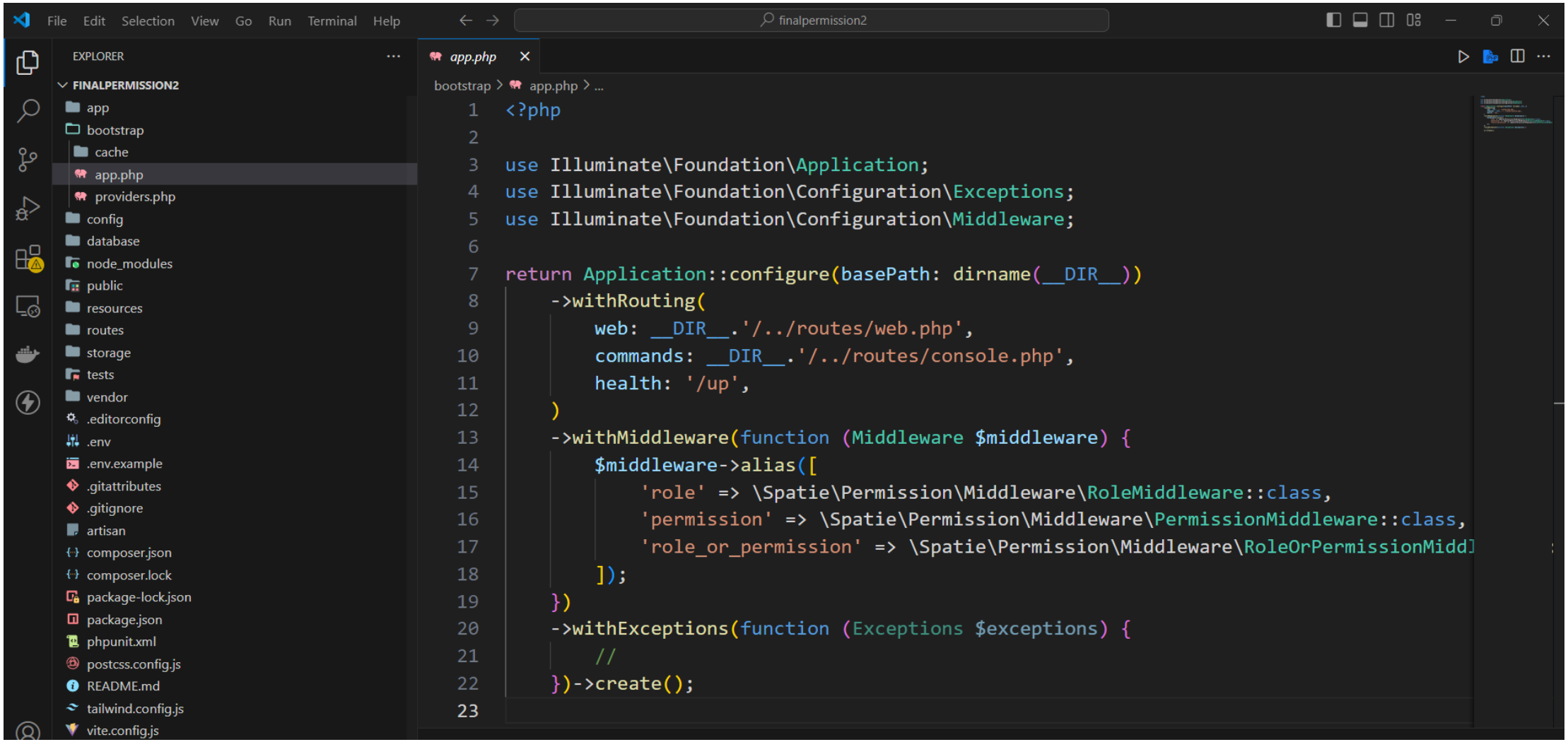
`role`

`permission`

So, we have to add middleware in the app.php file this way:

`bootstrap/app.php`

Cont. ...


File Edit Selection View Go Run Terminal Help
finalpermission2
EXPLORER
FINALPERMISSION2
app
bootstrap
cache
app.php
providers.php
config
database
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
README.md
tailwind.config.js
vite.config.js
app.php
bootstrap > app.php > ...
<?php
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
return Application::configure(basePath: dirname(__DIR__))
->withRouting(
web: __DIR__.'/../routes/web.php',
commands: __DIR__.'/../routes/console.php',
health: '/up',
)
->withMiddleware(function (Middleware $middleware) {
$middleware->alias([
'role' => \Spatie\Permission\Middleware\RoleMiddleware::class,
'permission' => \Spatie\Permission\Middleware\PermissionMiddleware::class,
'role_or_permission' => \Spatie\Permission\Middleware\RoleOrPermissionMiddl
]);
})
->withExceptions(function (Exceptions $exceptions) {
//
})->create();

Step 6 (Creating Authentication Using laravel/ui)

Step 6: Create Authentication

You have to follow a few steps to make auth in your Laravel 11 application.

First, you need to install the laravel/ui package as shown below:

```
composer require laravel/ui
```

Here, we need to generate auth scaffolding in Laravel 11 using Laravel UI command. So, let's generate it by the below command:

```
php artisan ui bootstrap --auth
```

Now you need to run the npm command; otherwise, you cannot see the better layout of the login and register page.

Install NPM:

```
npm install
```

Run NPM:

```
npm run build
```

npm run dev

- ***npm run dev*** is a command used in Laravel projects (and other JavaScript projects) to compile frontend assets, such as CSS and JavaScript, in development mode.
- It leverages the Vite build tool in Laravel (replacing Webpack Mix in earlier versions) to bundle and serve assets efficiently.

How to run npm

Full Syntax of `npm run dev`

1. **Run in Project Root**: You must run the command in the root directory of your Laravel project, where the `package.json` file is located.

bash Copy code

```
npm run dev
```

Cont. ...

ChatGPT

Share

2. **What Happens Under the Hood:**

- `npm run dev` executes the `dev` script defined in the `package.json` file.
- In a Laravel project using Vite, the `dev` script typically looks like this:

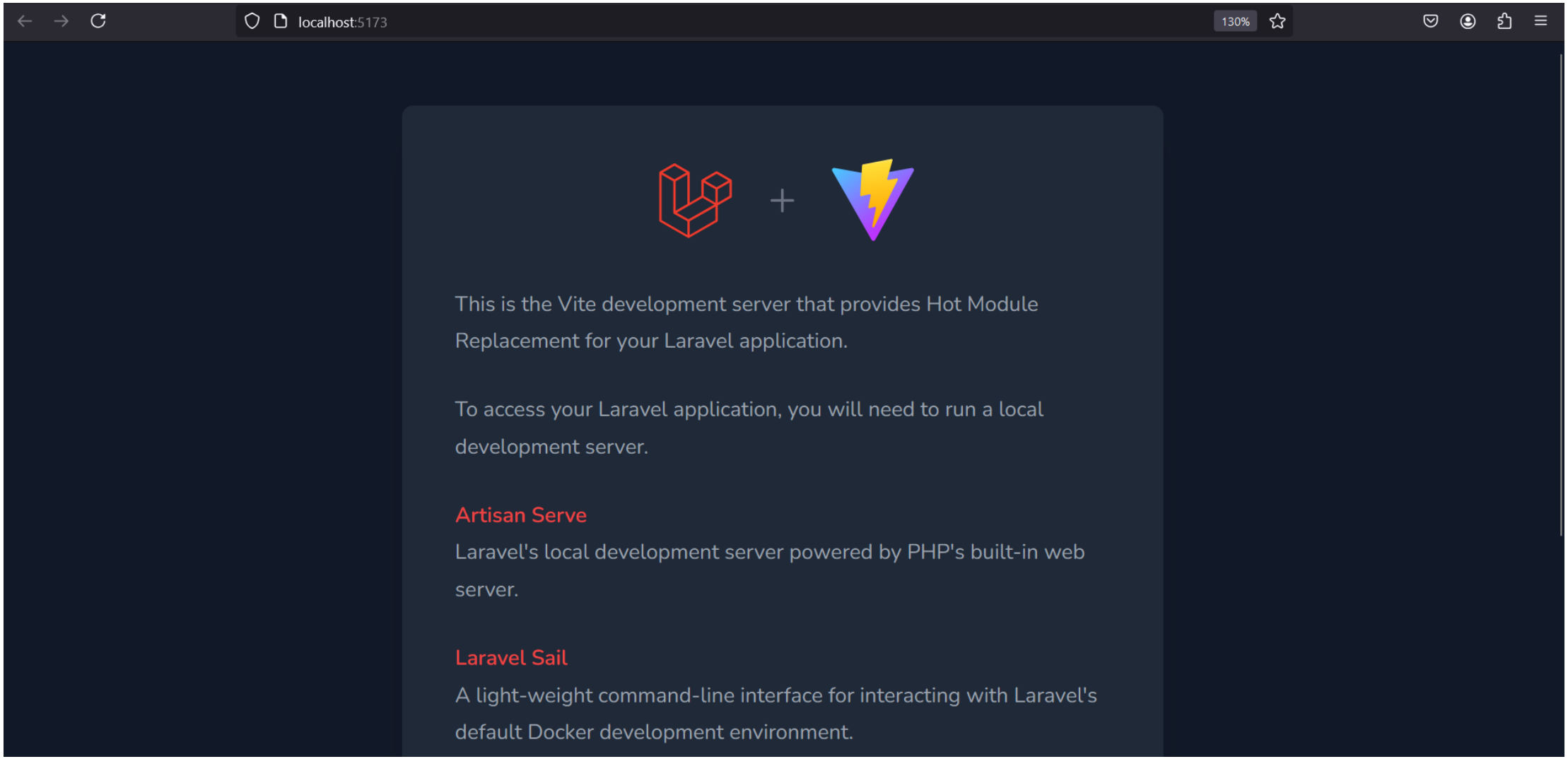
json

Copy code

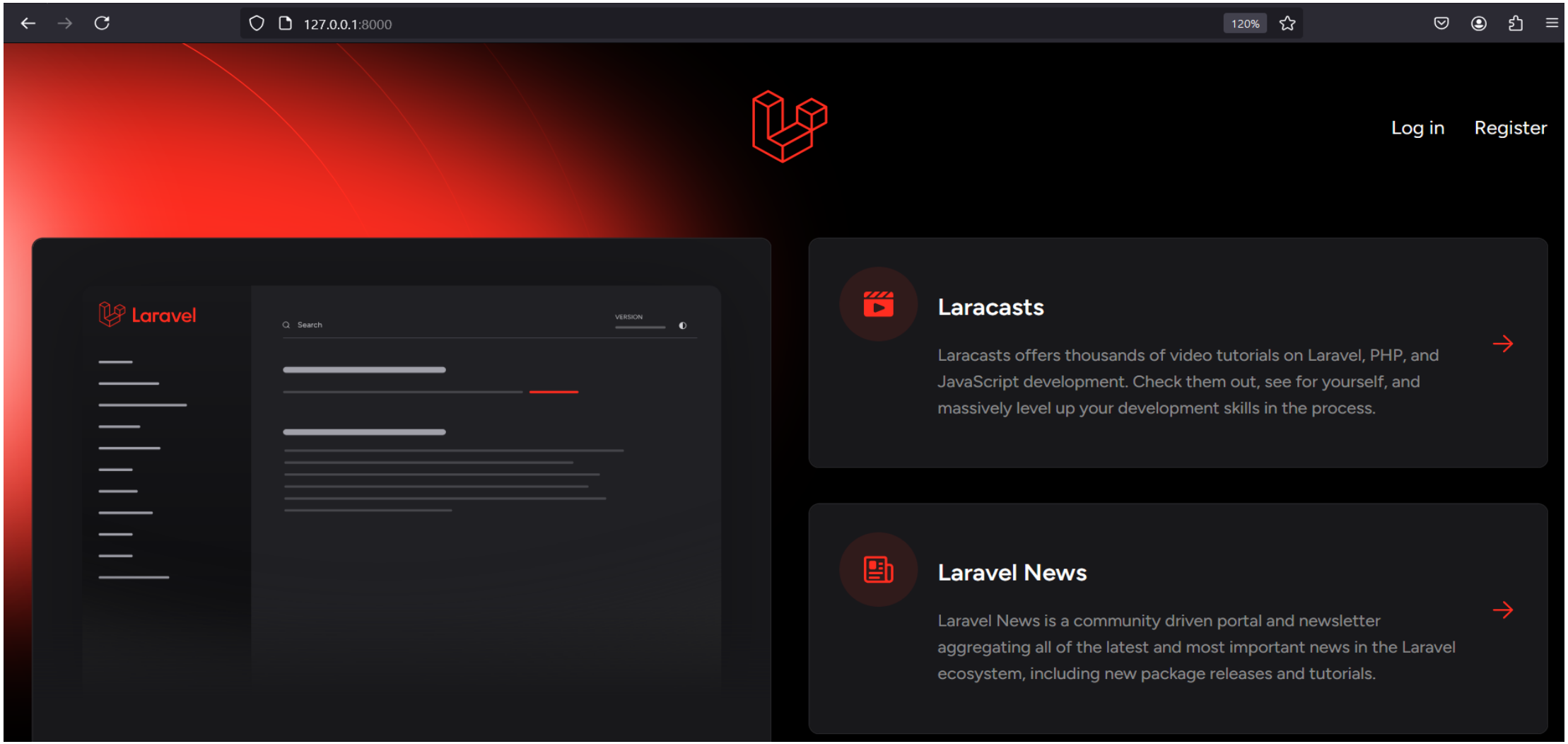
```
"scripts": {
    "dev": "vite",
    "build": "vite build"
}
```

- The `vite` command starts the development server for asset compilation.

Output (http://localhost:5173/)



Output (http://127.0.0.1:8000/)



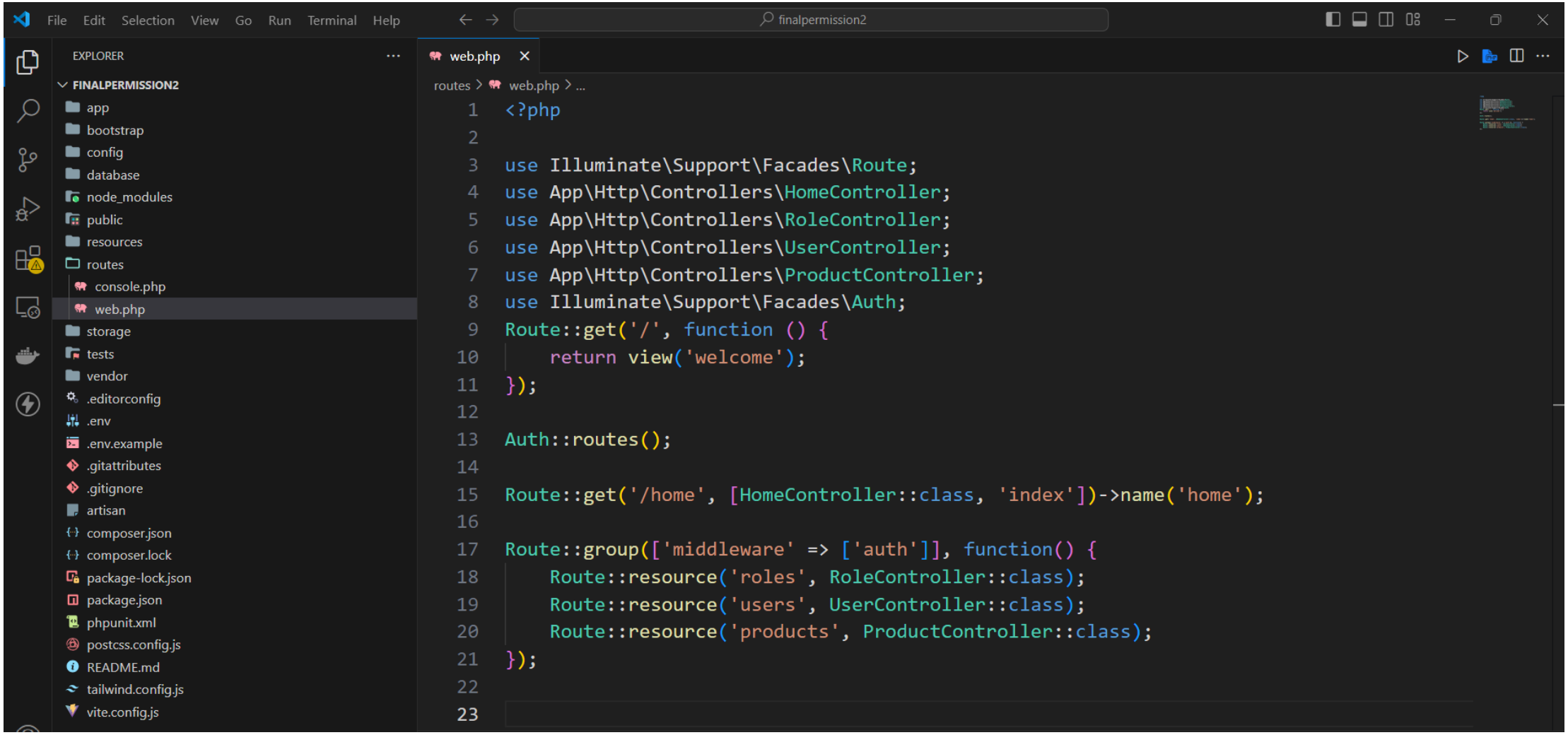
Step 7 (Creating Routes)

Step 7: Create Routes

We need to add a number of routes for the users module, products module, and roles module. In these routes, I also use middleware with permissions for the roles and products routes, so add routes this way:

routes/web.php

Cont. ...


File Edit Selection View Go Run Terminal Help
finalpermission2
EXPLORER
FINALPERMISSION2
app
bootstrap
config
database
node_modules
public
resources
routes
console.php
web.php
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
README.md
tailwind.config.js
vite.config.js
web.php
routes > web.php > ...
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\HomeController;
5 use App\Http\Controllers\RoleController;
6 use App\Http\Controllers\UserController;
7 use App\Http\Controllers\ProductController;
8 use Illuminate\Support\Facades\Auth;
9 Route::get('/', function () {
10 return view('welcome');
11 });
12
13 Auth::routes();
14
15 Route::get('/home', [HomeController::class, 'index'])->name('home');
16
17 Route::group(['middleware' => ['auth']], function() {
18 Route::resource('roles', RoleController::class);
19 Route::resource('users', UserController::class);
20 Route::resource('products', ProductController::class);
21 });
22
23

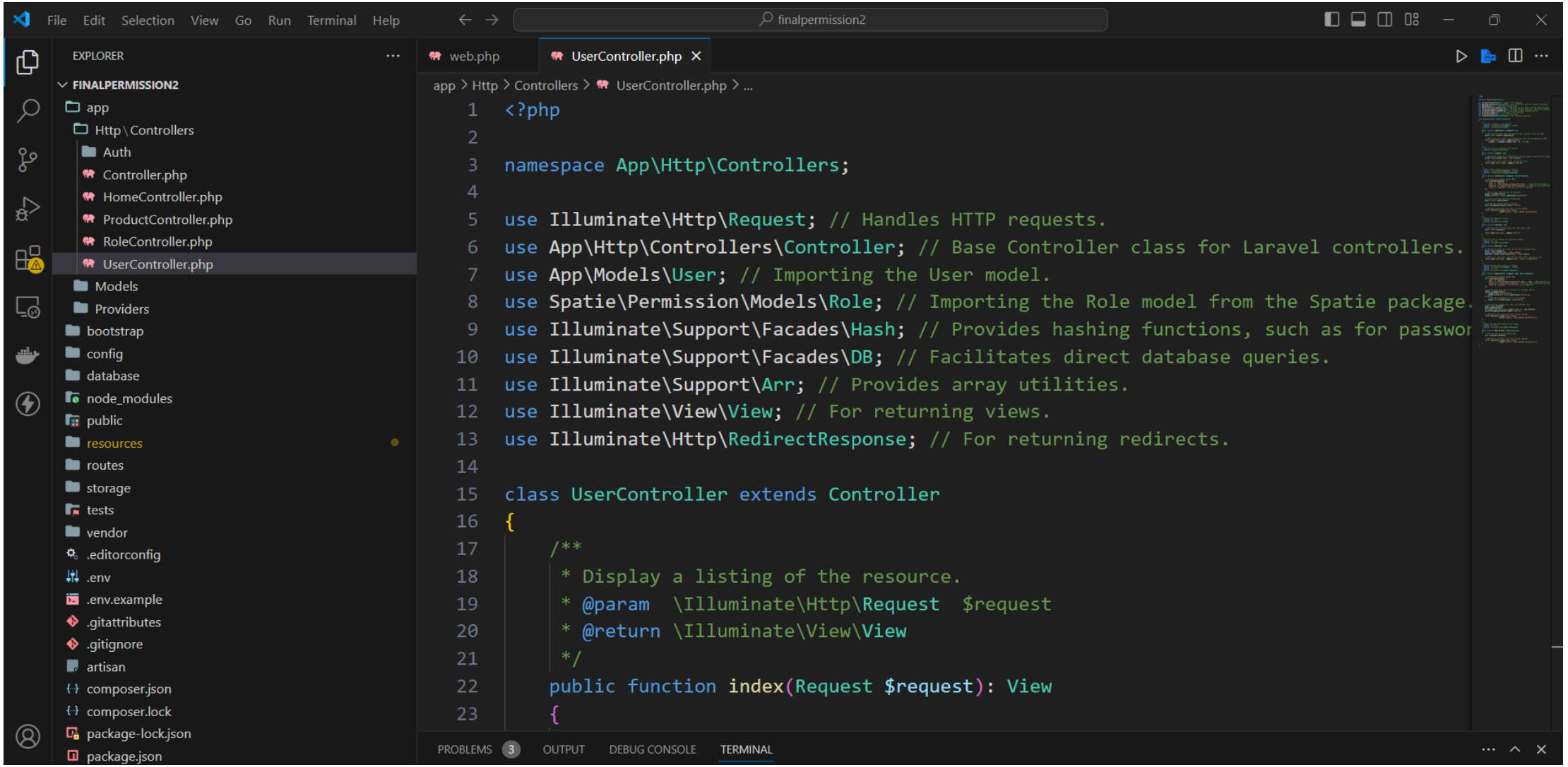
Step 8 (Add Controllers)

Step 8: Add Controllers

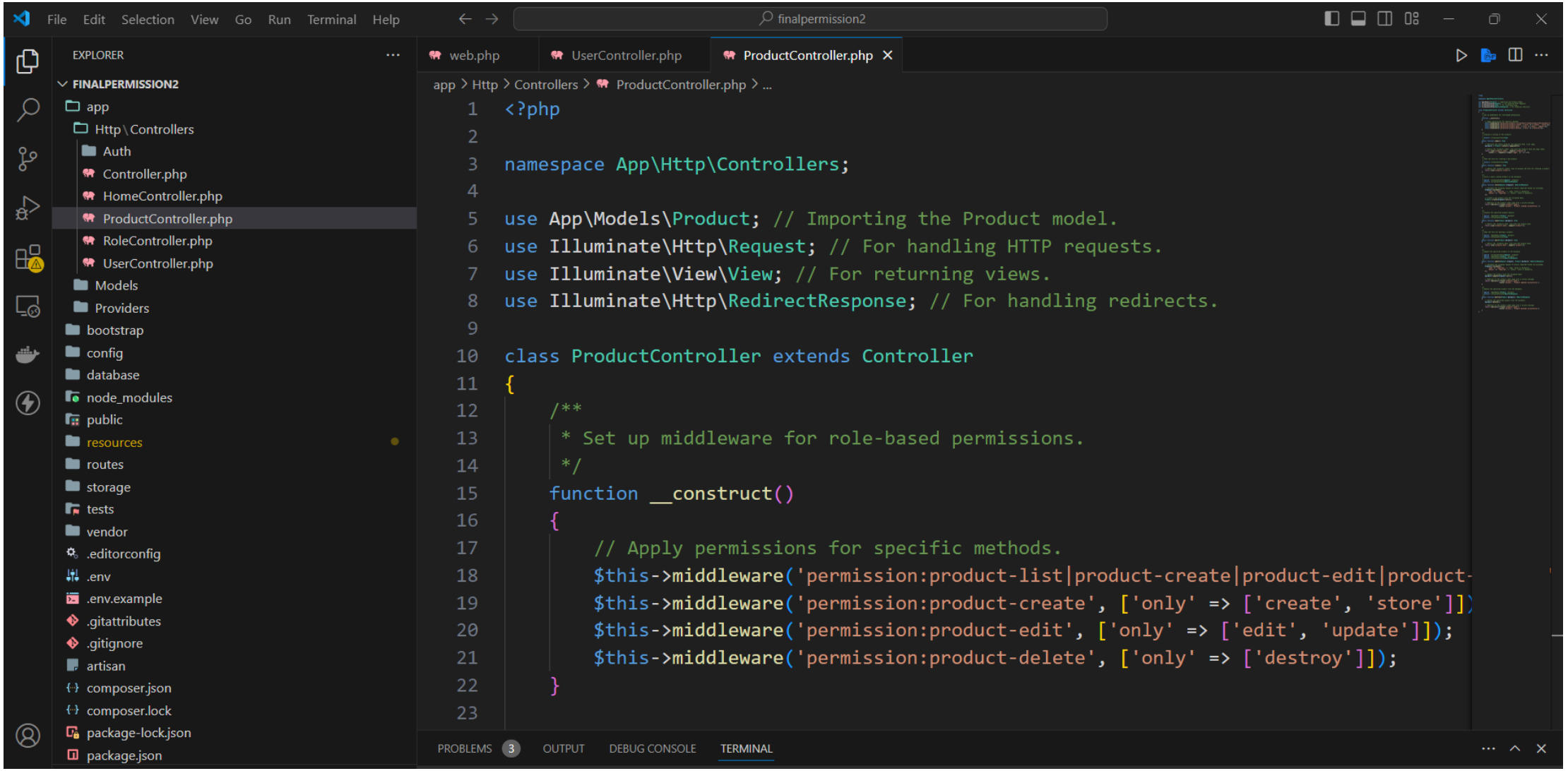
In this step, we have added three controllers for the users module, products module, and roles module. So, you can create three controllers as below:

app/Http/Controllers/UserController.php

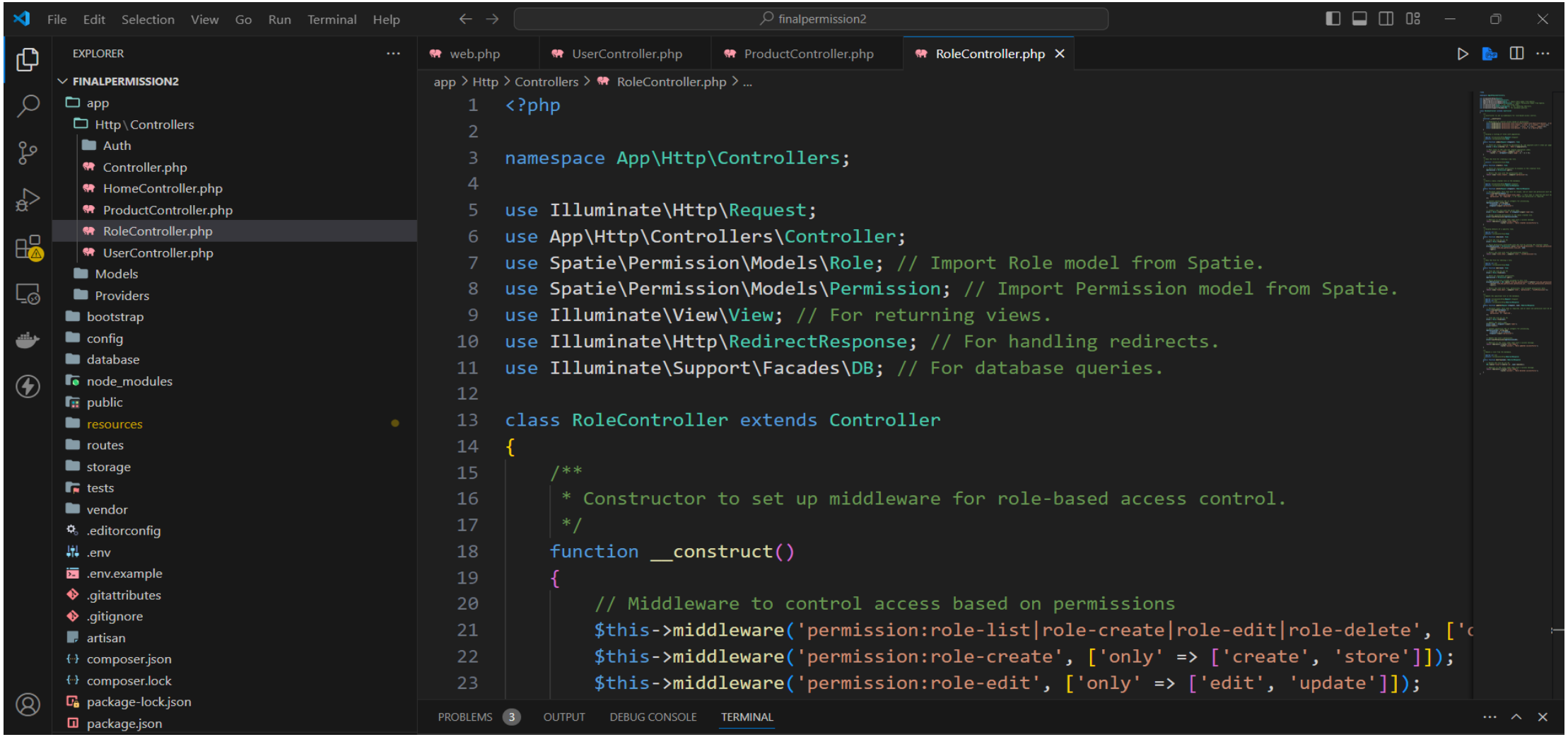
app/Http/Controllers/UserController.php



app/Http/Controllers/ProductController.php



app/Http/Controllers/RoleController.php



Step 9 (Adding Blade Files)

Step 9: Add Blade Files

In this step, we need to create the following files as listed below:

Theme Layout

app.blade.php

Users Module

index.blade.php create.blade.php edit.blade.php show.blade.php

Roles Module

index.blade.php create.blade.php edit.blade.php show.blade.php

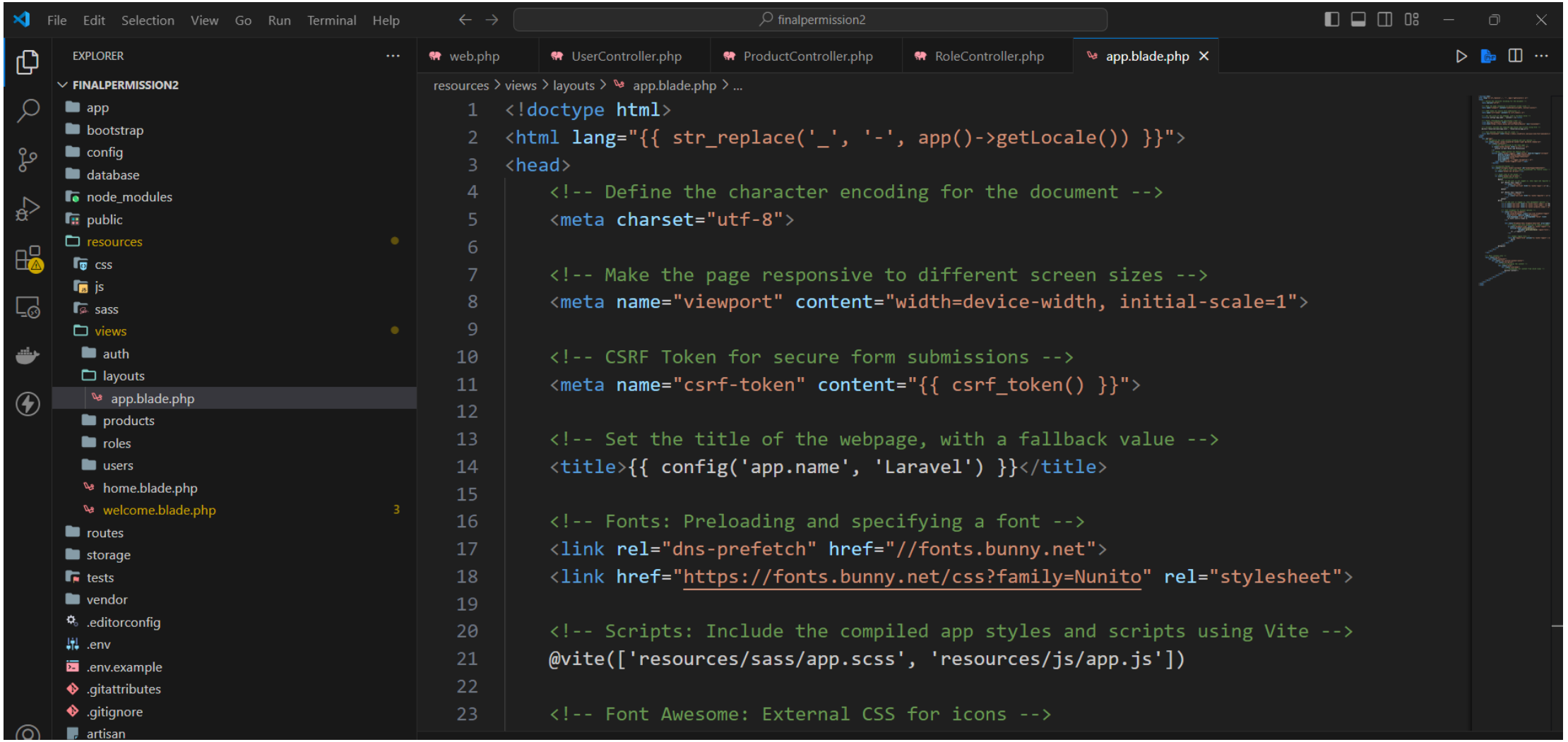
Product Module

index.blade.php create.blade.php edit.blade.php show.blade.php

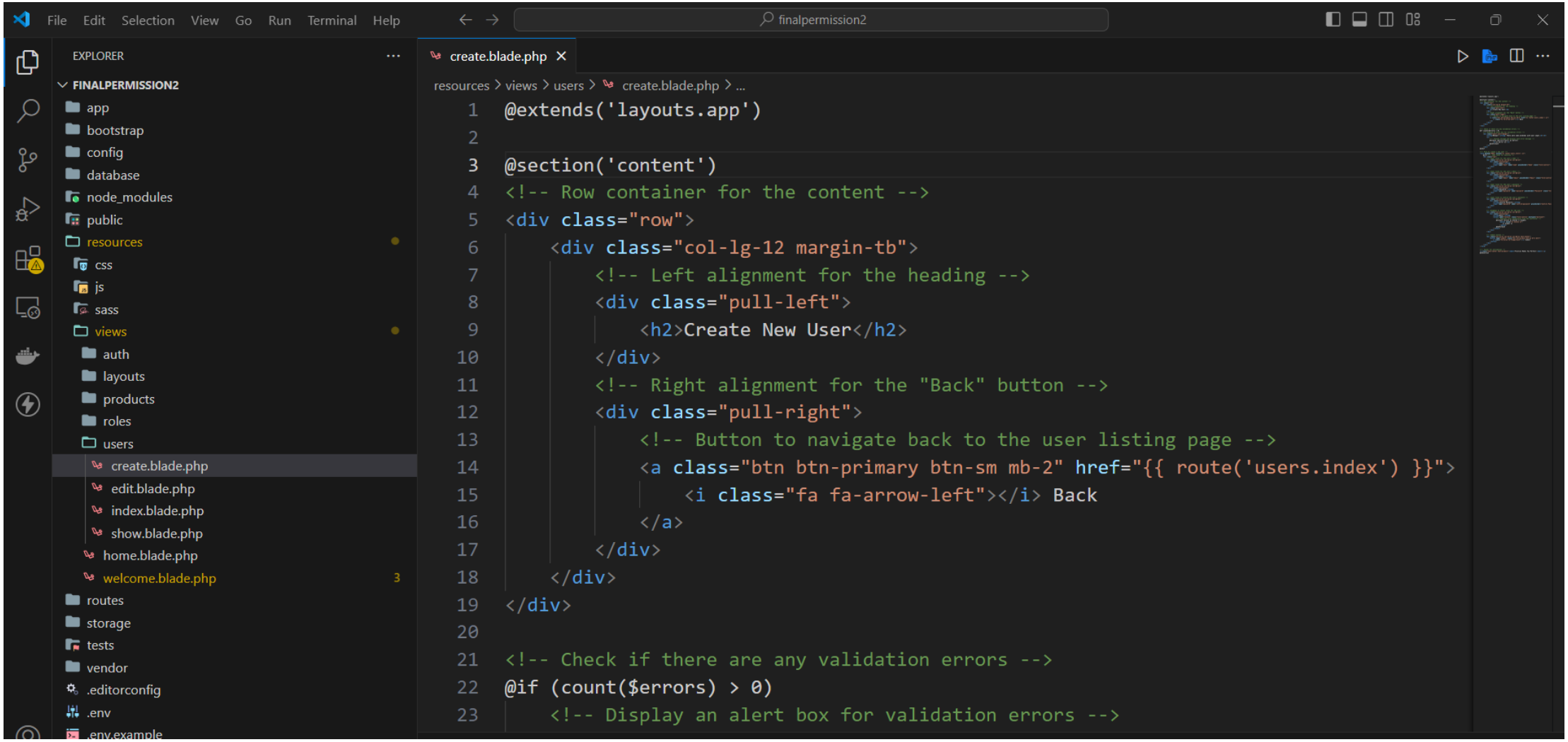
So, let's create following files:

`resources/views/layouts/app.blade.php`

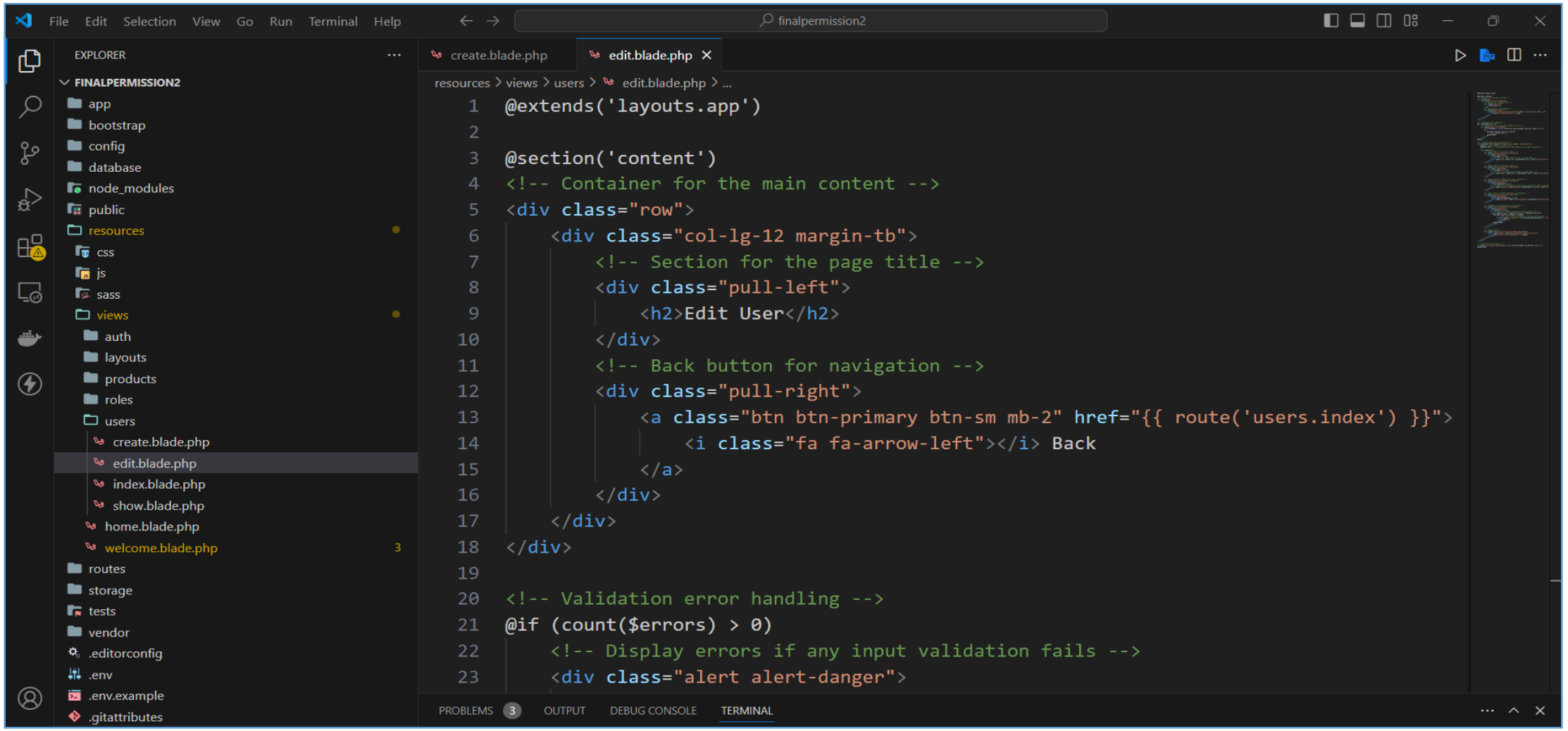
resources/views/layouts/app.blade.php



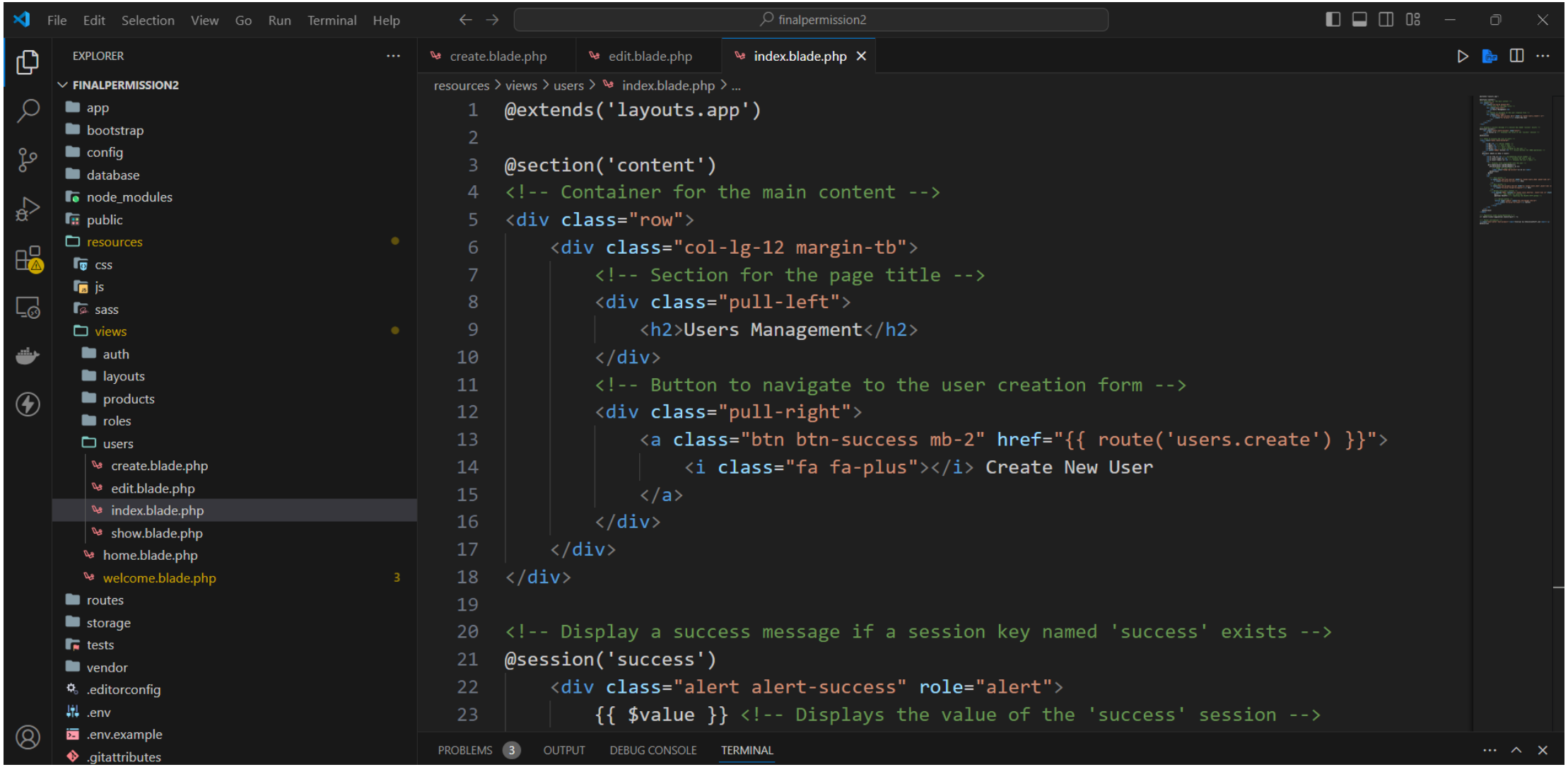
resources/views/users/index.blade.php



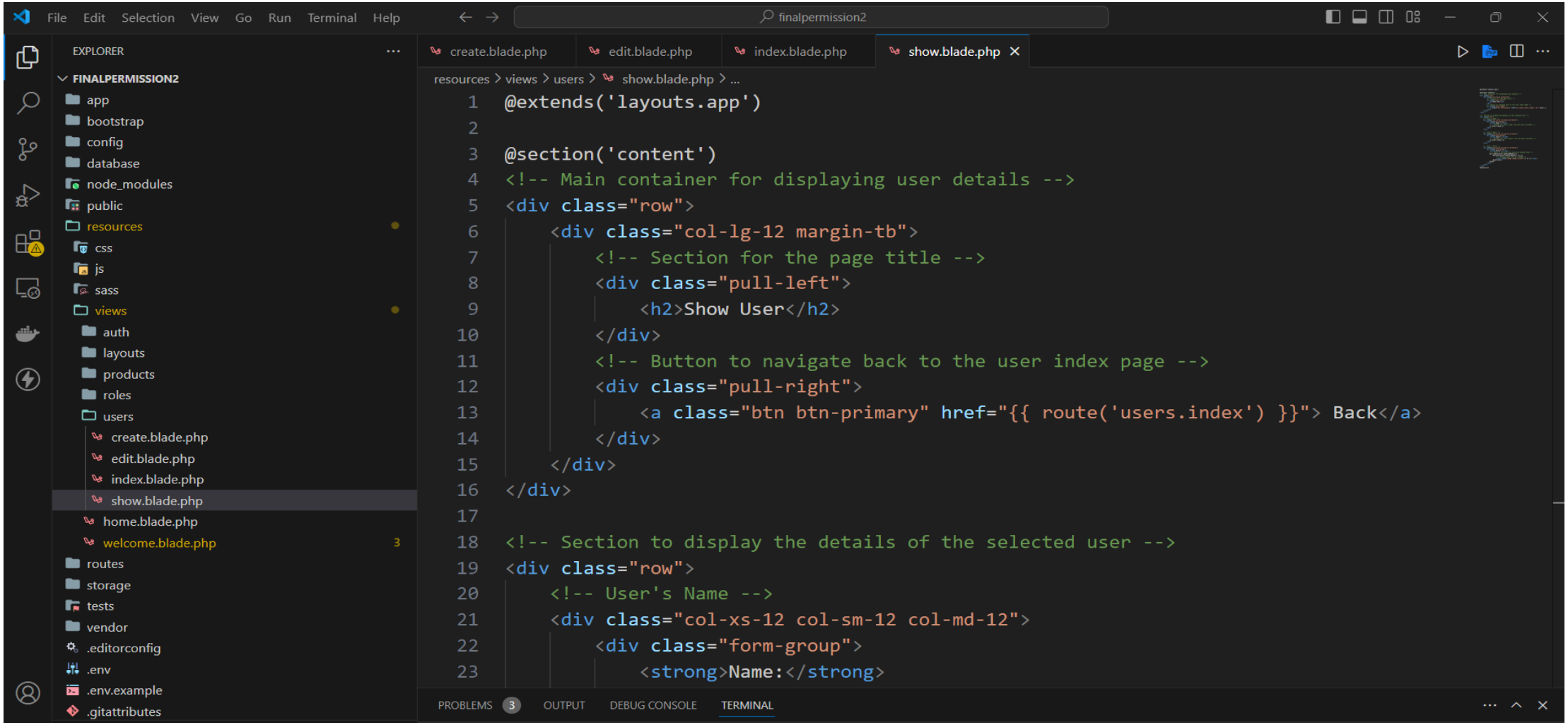
resources/views/users/edit.blade.php



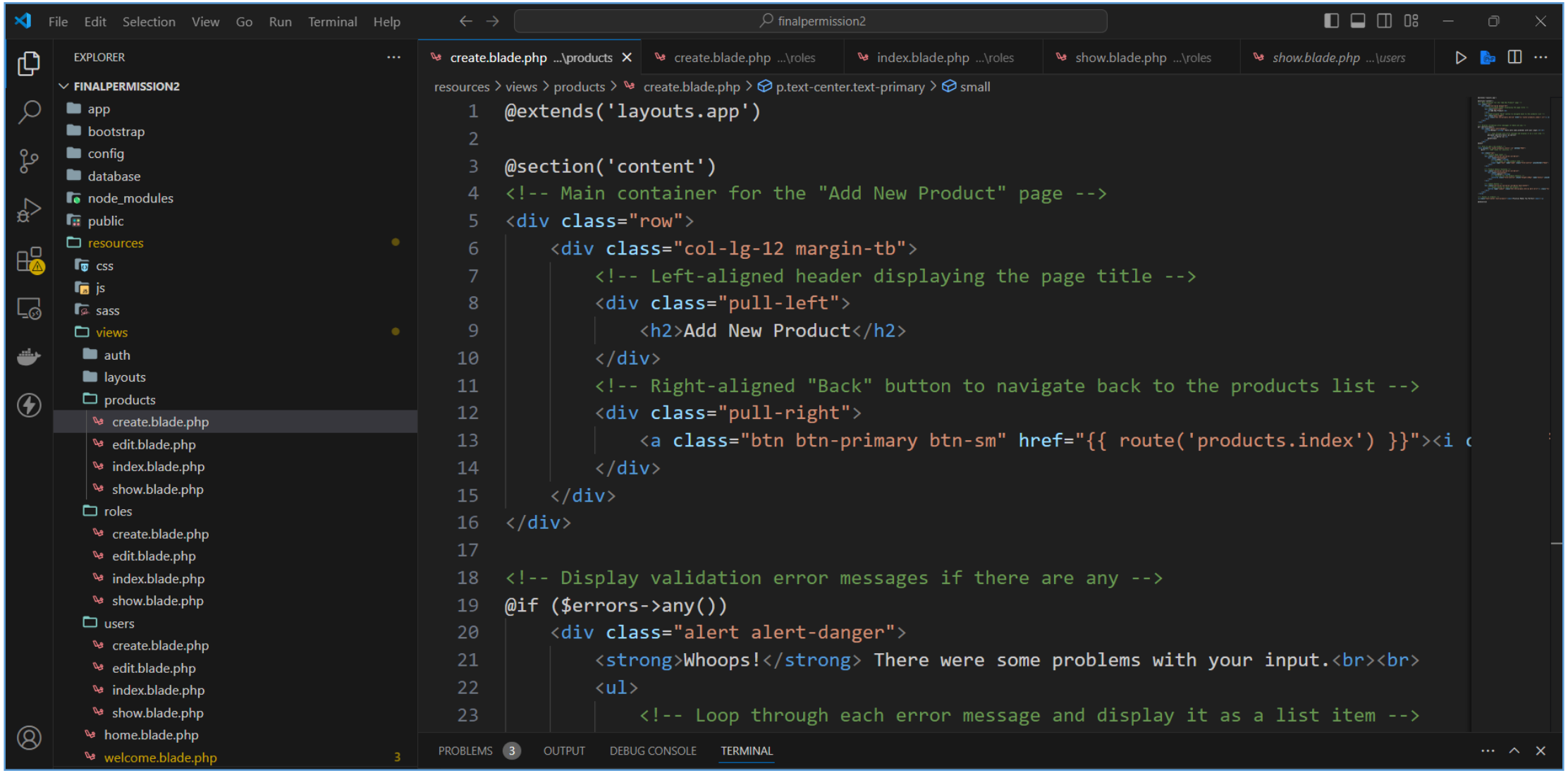
resources/views/users/index.blade.php



resources/views/users/show.blade.php



For Product and User Roles



Step 10 (Creating Seeder)

Step 10: Create Seeder For Permissions and AdminUser

In this step, we will create a seeder for permissions. Right now, we have fixed permissions, so we'll create them using a seeder as listed below. However, if you want, you can add more permissions as you wish.

1.role-list

2.role-create

3.role-edit

4.role-delete

5.product-list

6.product-create

7.product-edit

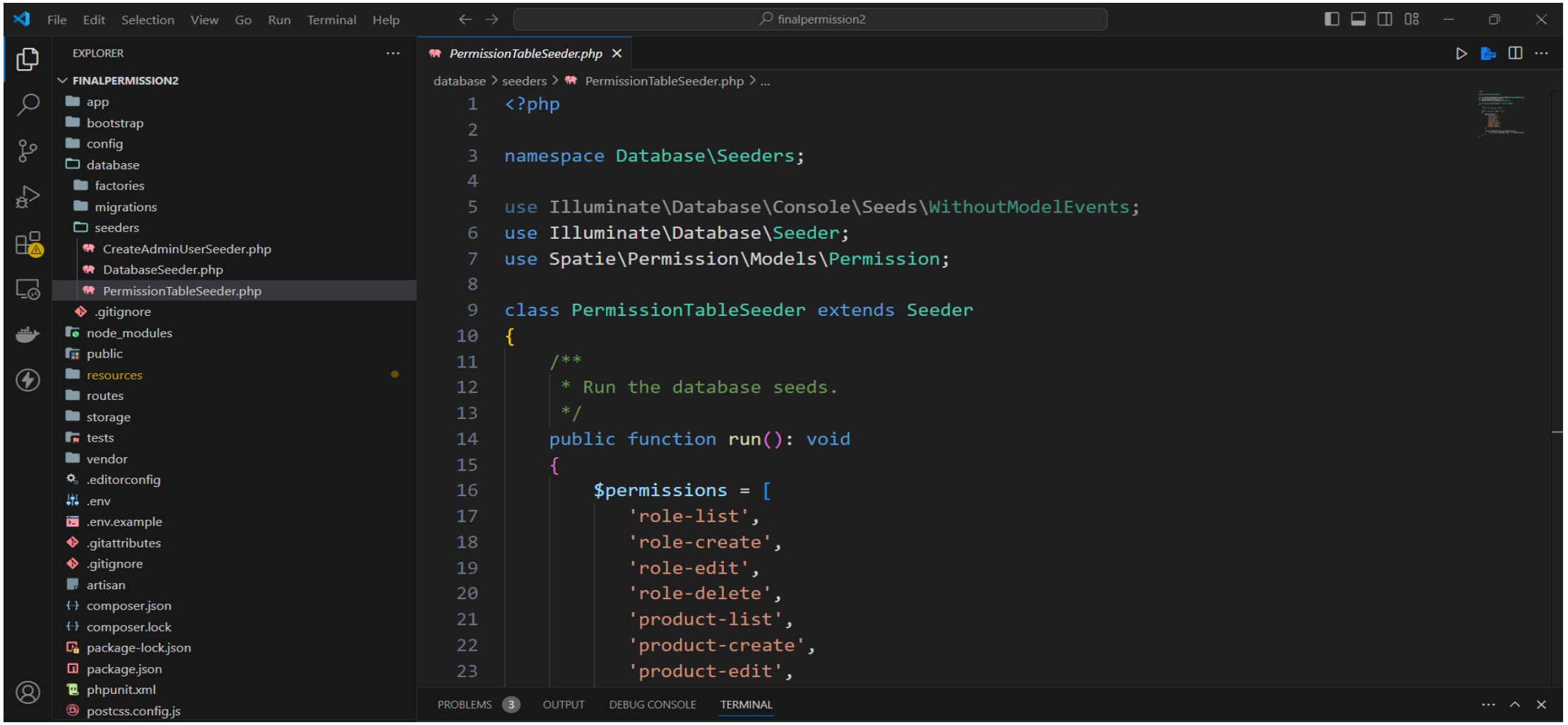
8.product-delete

So, first create a seeder using the below command:

```
php artisan make:seeder PermissionTableSeeder
```

And put bellow code in PermissionTableSeeder seeder this way:

database/seeders/PermissionTableSeeder.php



Cont. ...

After this, we have to run the below command to execute the PermissionTableSeeder seeder:

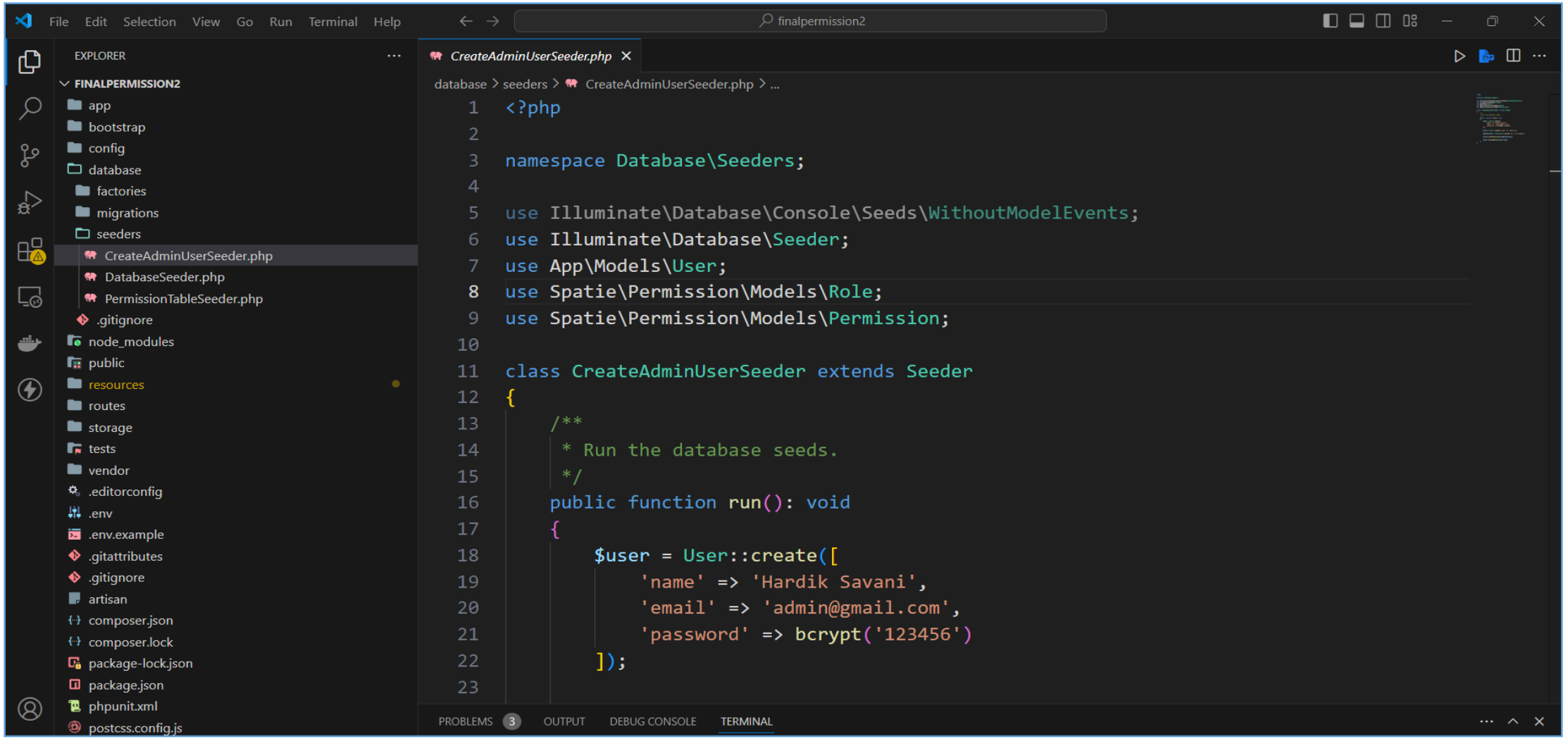
```
php artisan db:seed --class=PermissionTableSeeder
```

Now let's create a new seeder for creating an admin user.

```
php artisan make:seeder CreateAdminUserSeeder
```

database/seeders/CreateAdminUserSeeder.php

database/seeders/CreateAdminUserSeeder.php



Cont. ...

Then, run seeder using the following command:

```
php artisan db:seed --class=CreateAdminUserSeeder
```

Run Laravel App:

All the required steps have been done, now you have to type the given below command and hit enter to run the Laravel app:

```
php artisan serve
```

Now, Go to your web browser, type the given URL and view the app output:

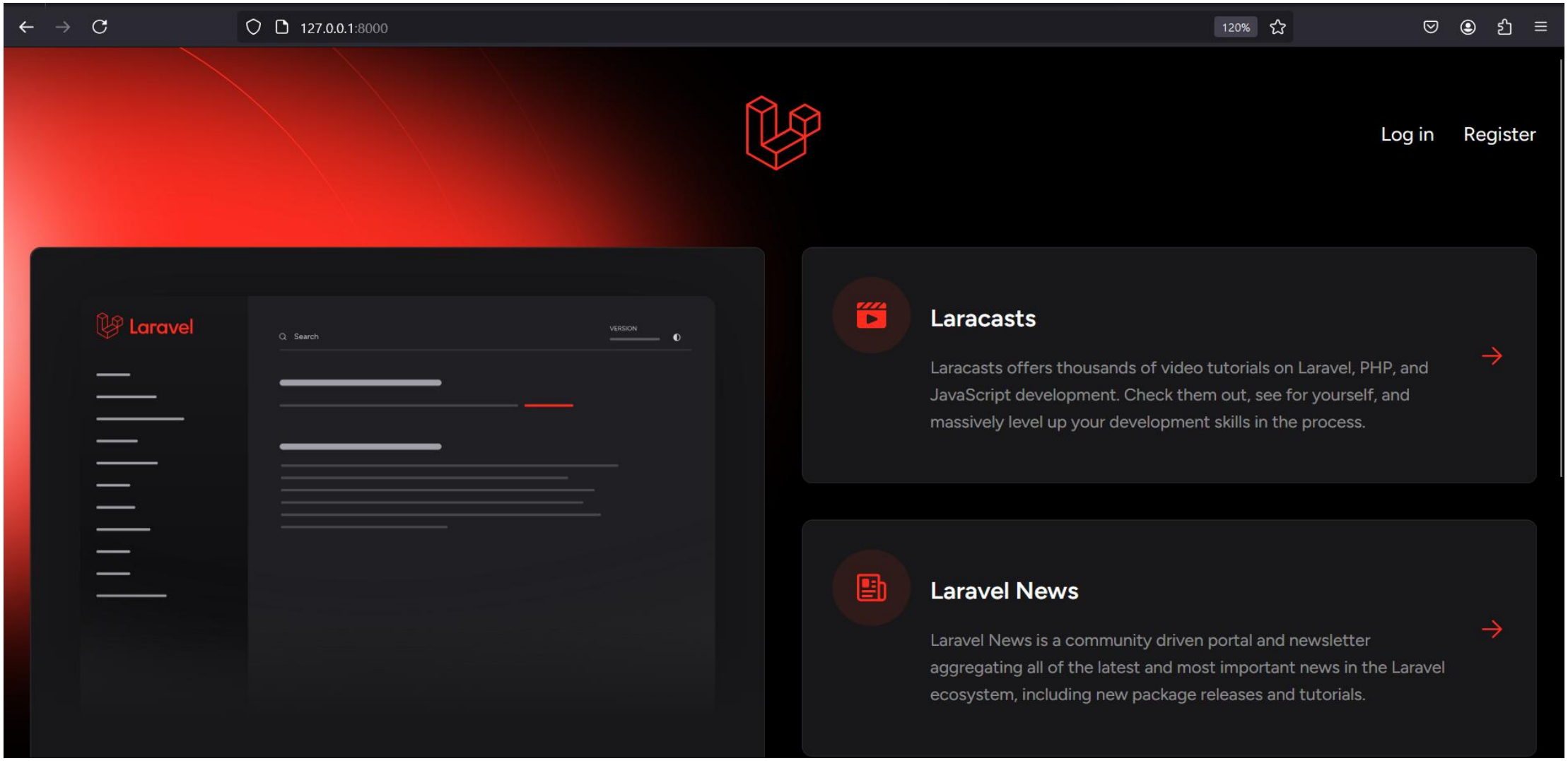
```
http://localhost:8000
```

Now you can login with following credential:

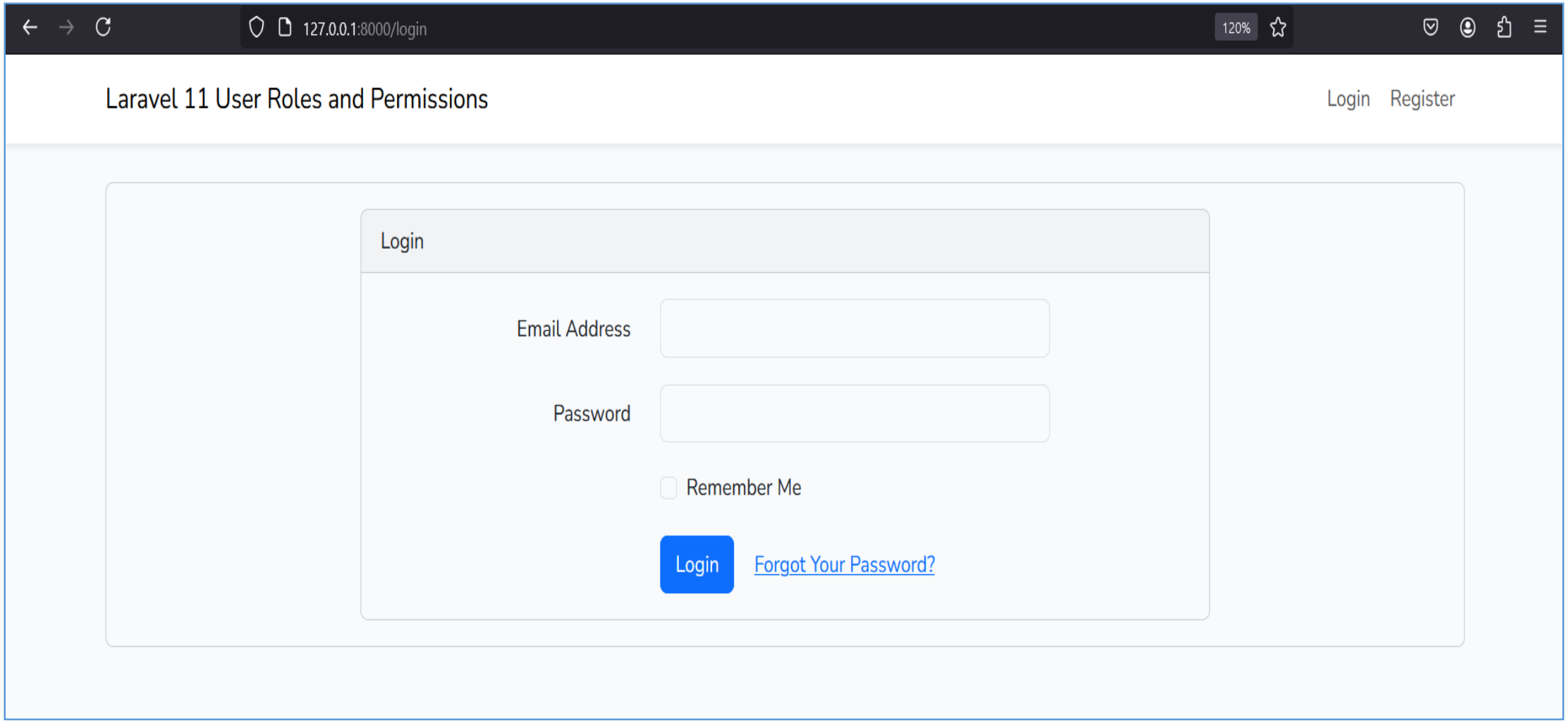
```
Email: admin@gmail.com
Password: 123456
```

Now you can run and check.

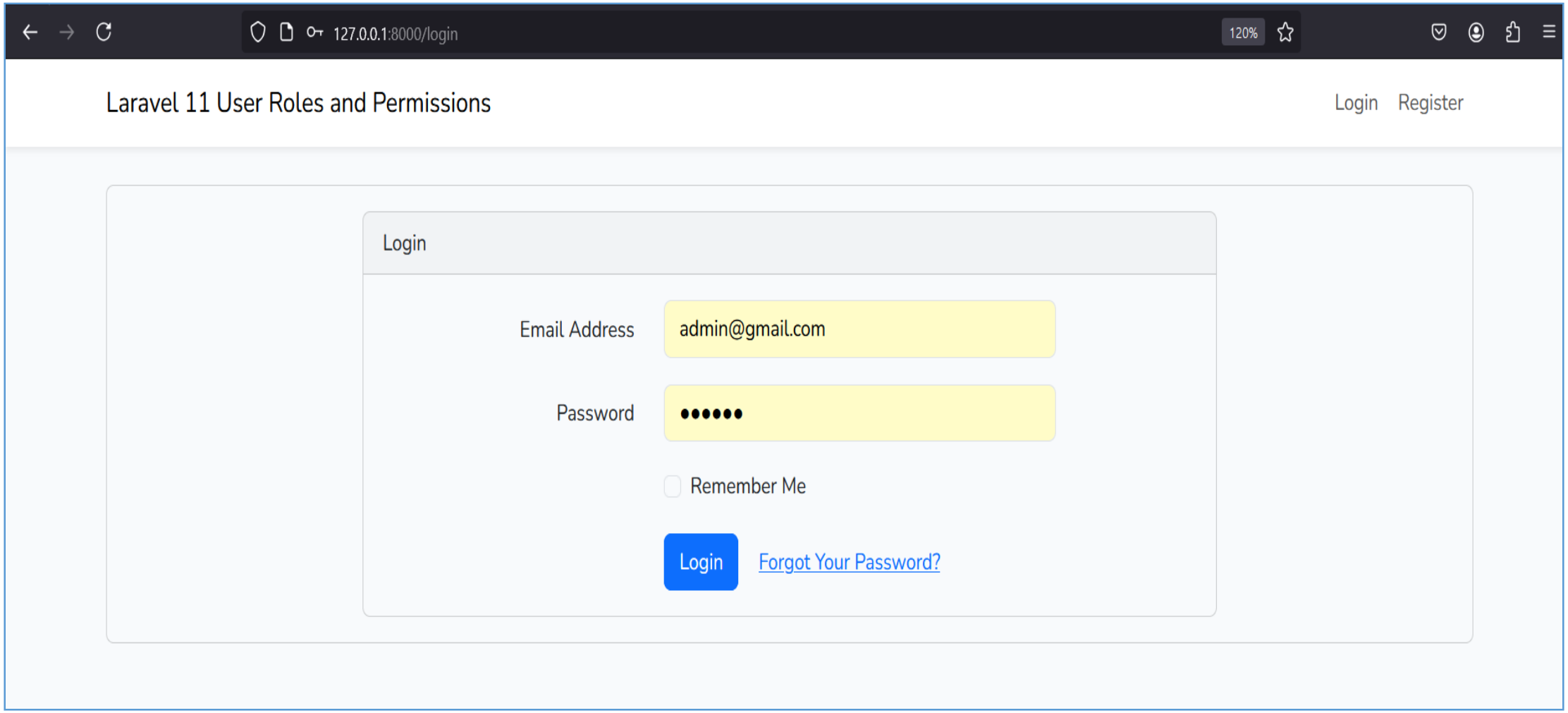
Run The Application (http://127.0.0.1:8000/)



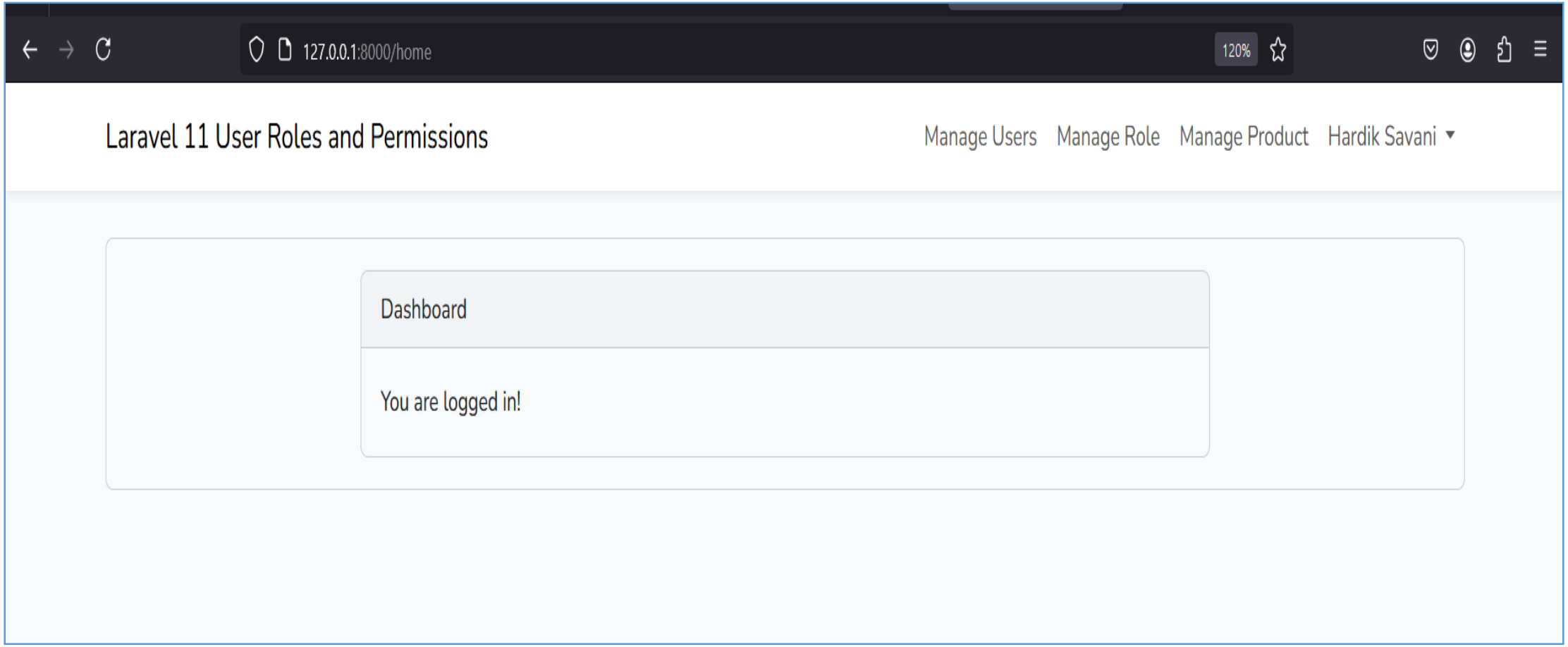
http://127.0.0.1:8000/login



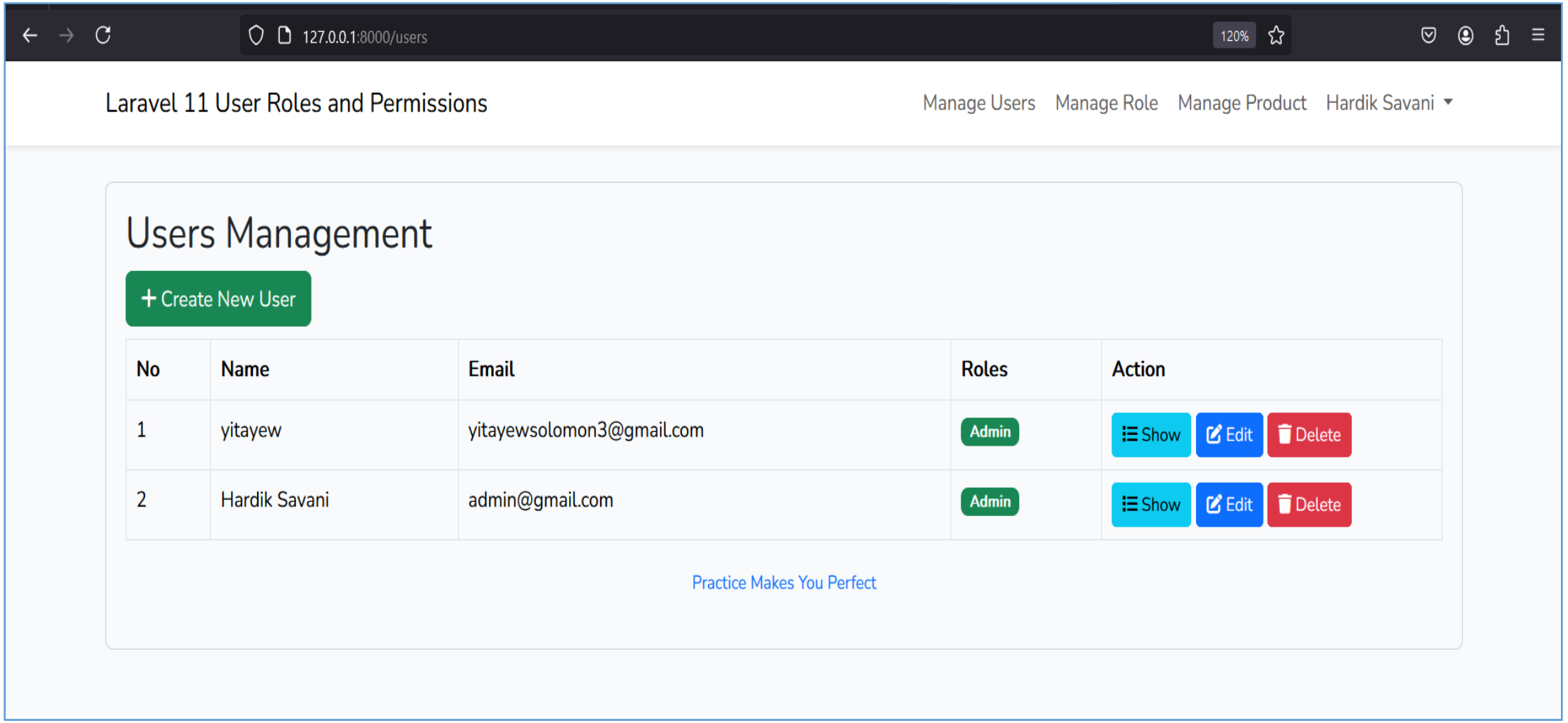
http://127.0.0.1:8000/login



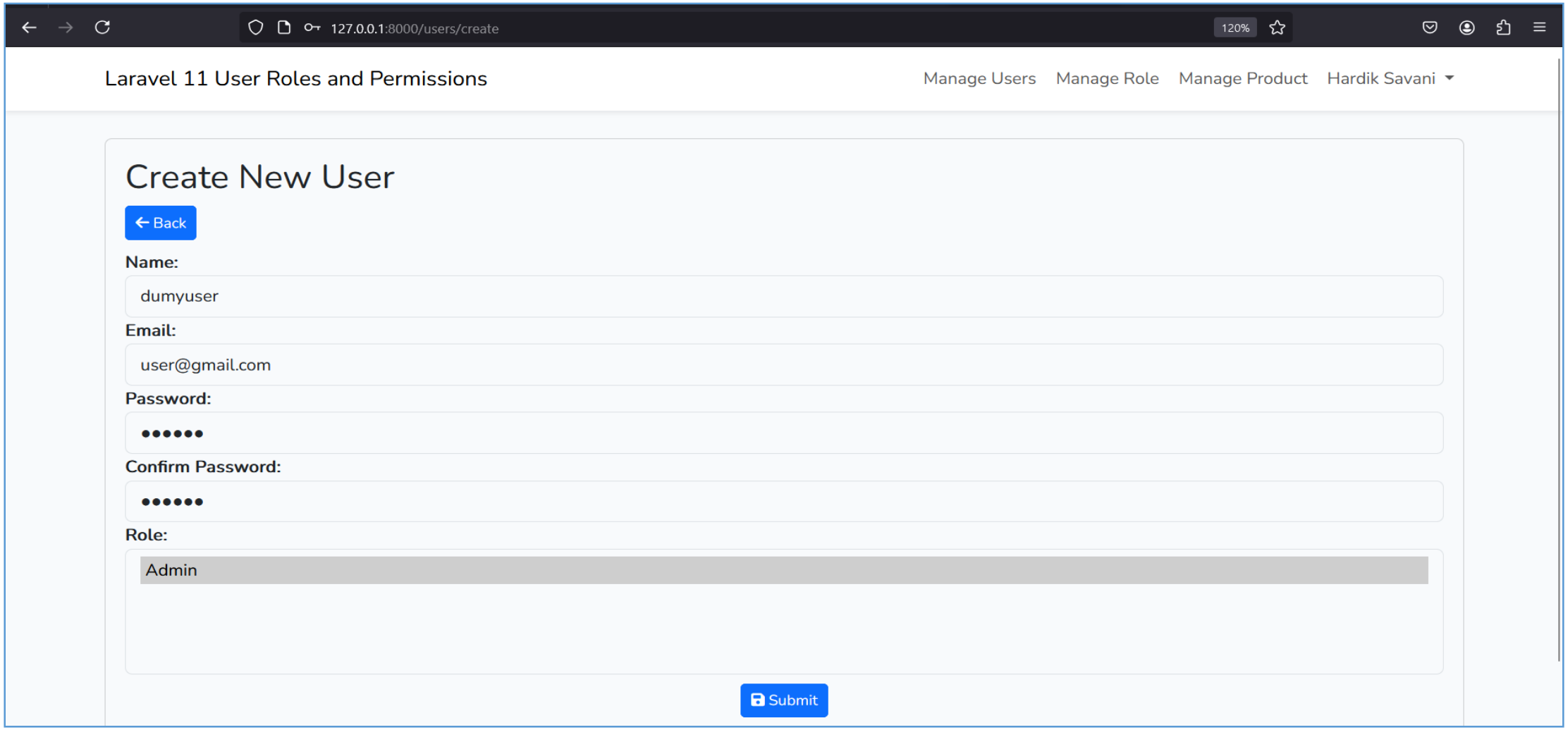
http://127.0.0.1:8000/home



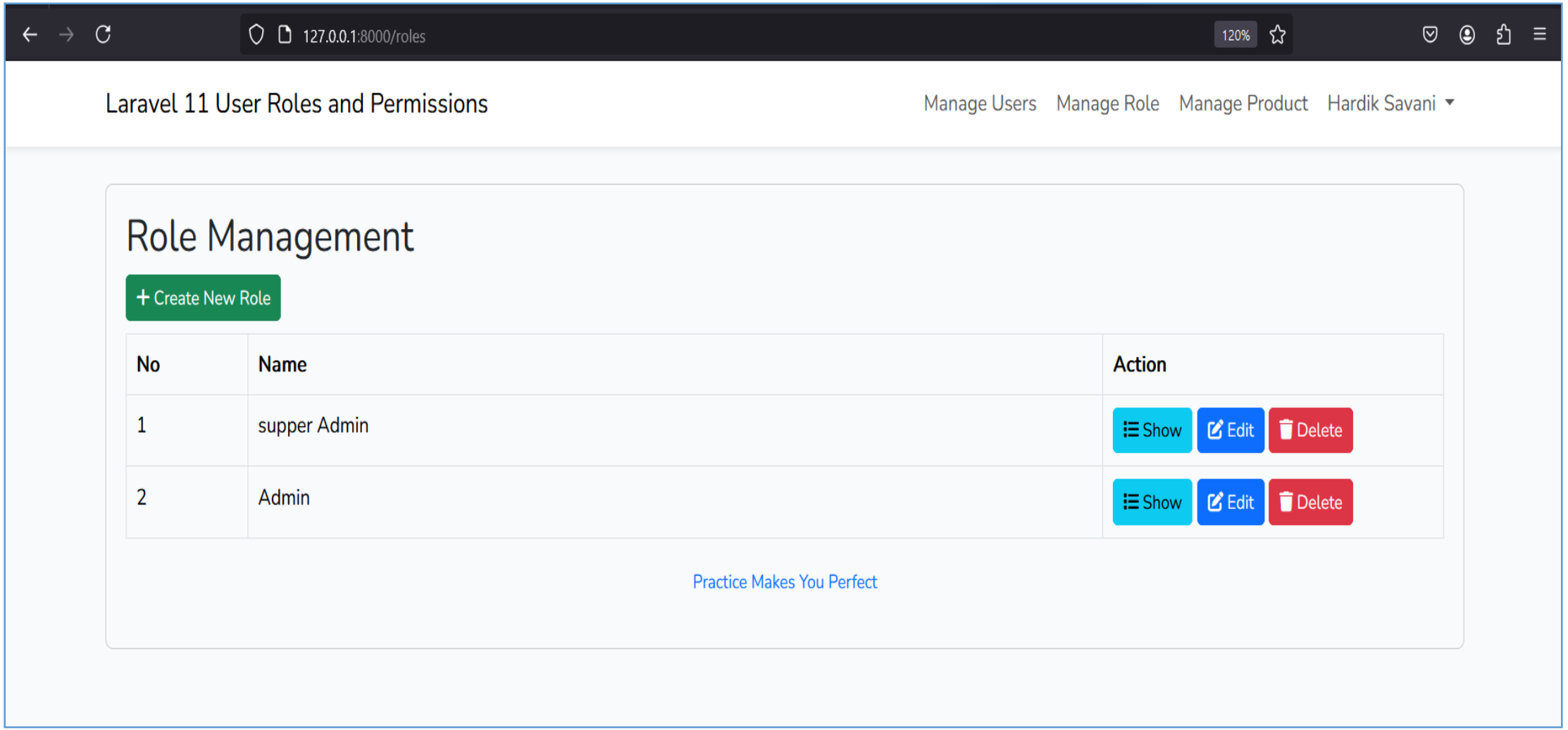
http://127.0.0.1:8000/users



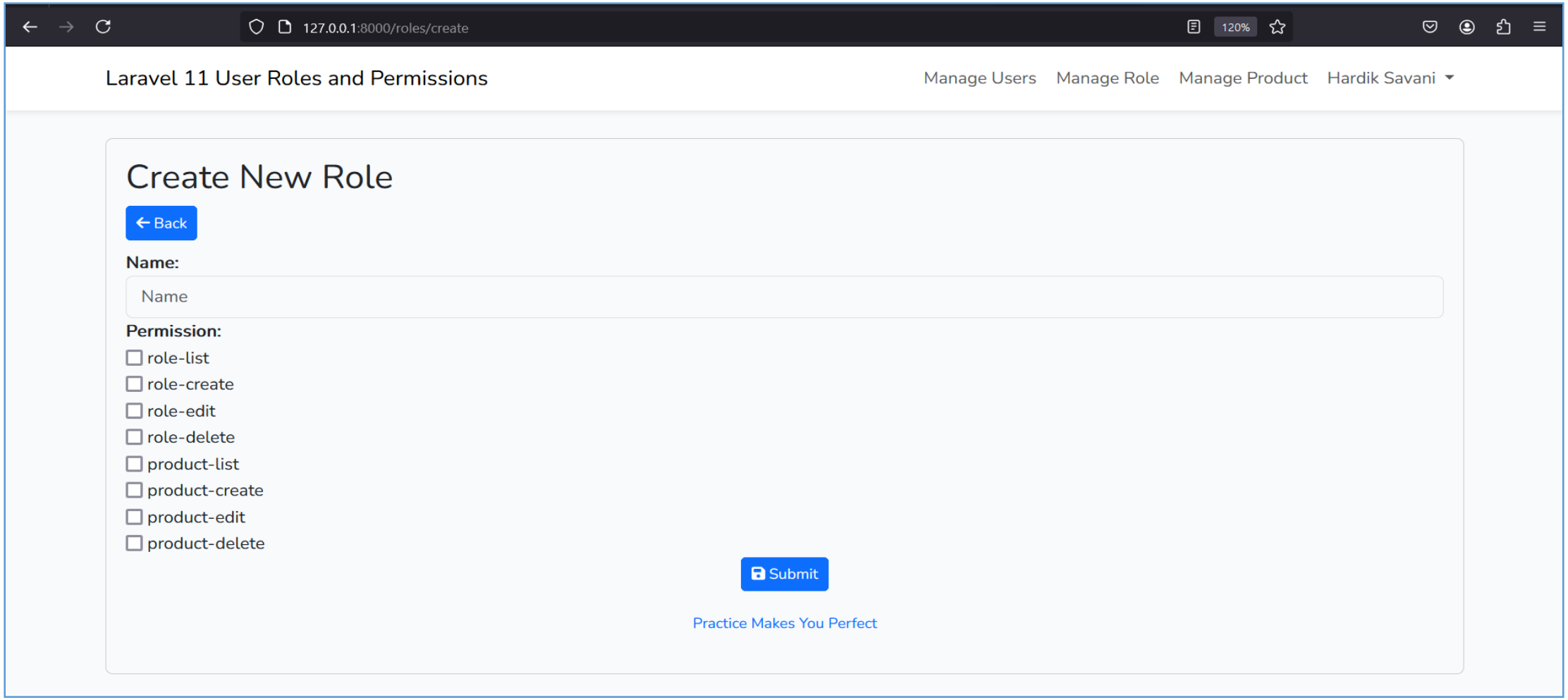
http://127.0.0.1:8000/users/create



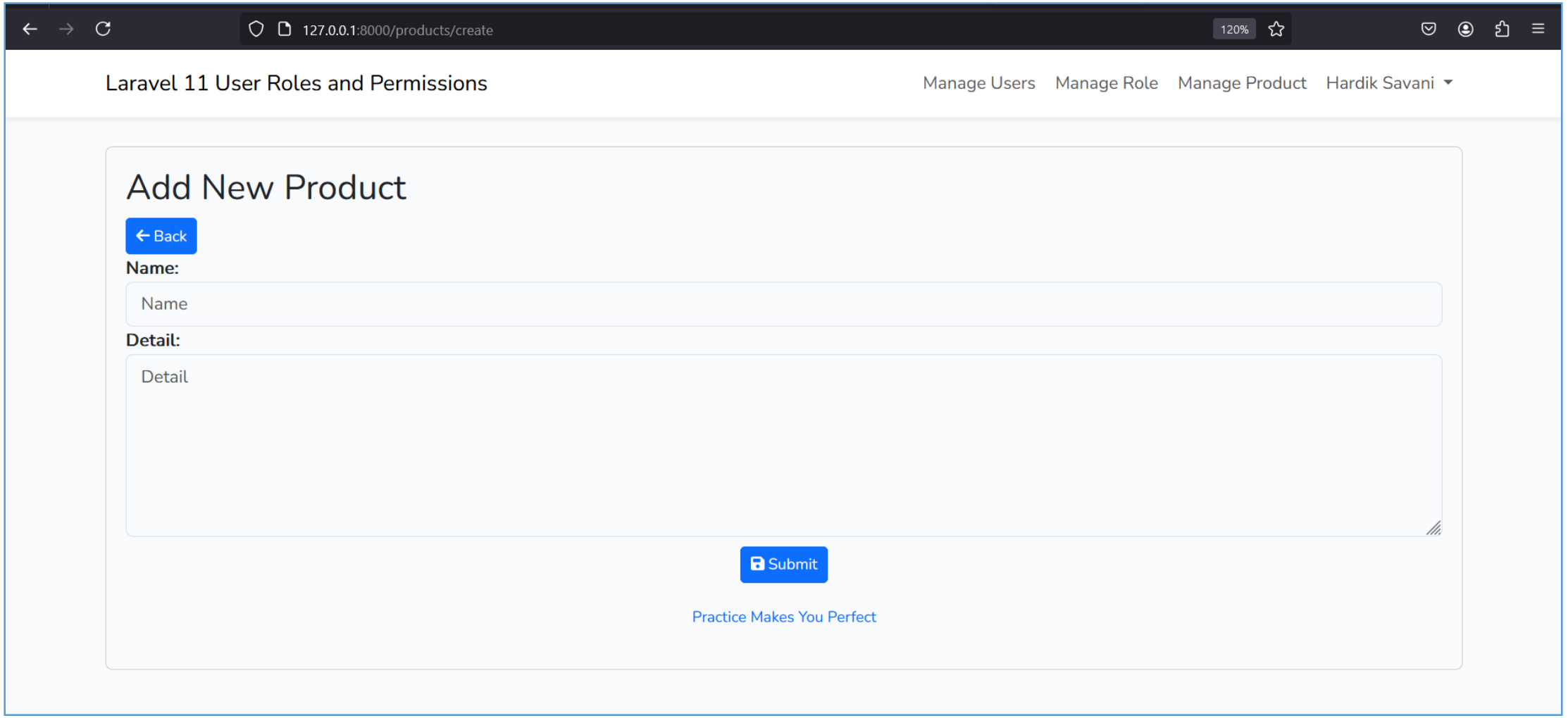
http://127.0.0.1:8000/roles



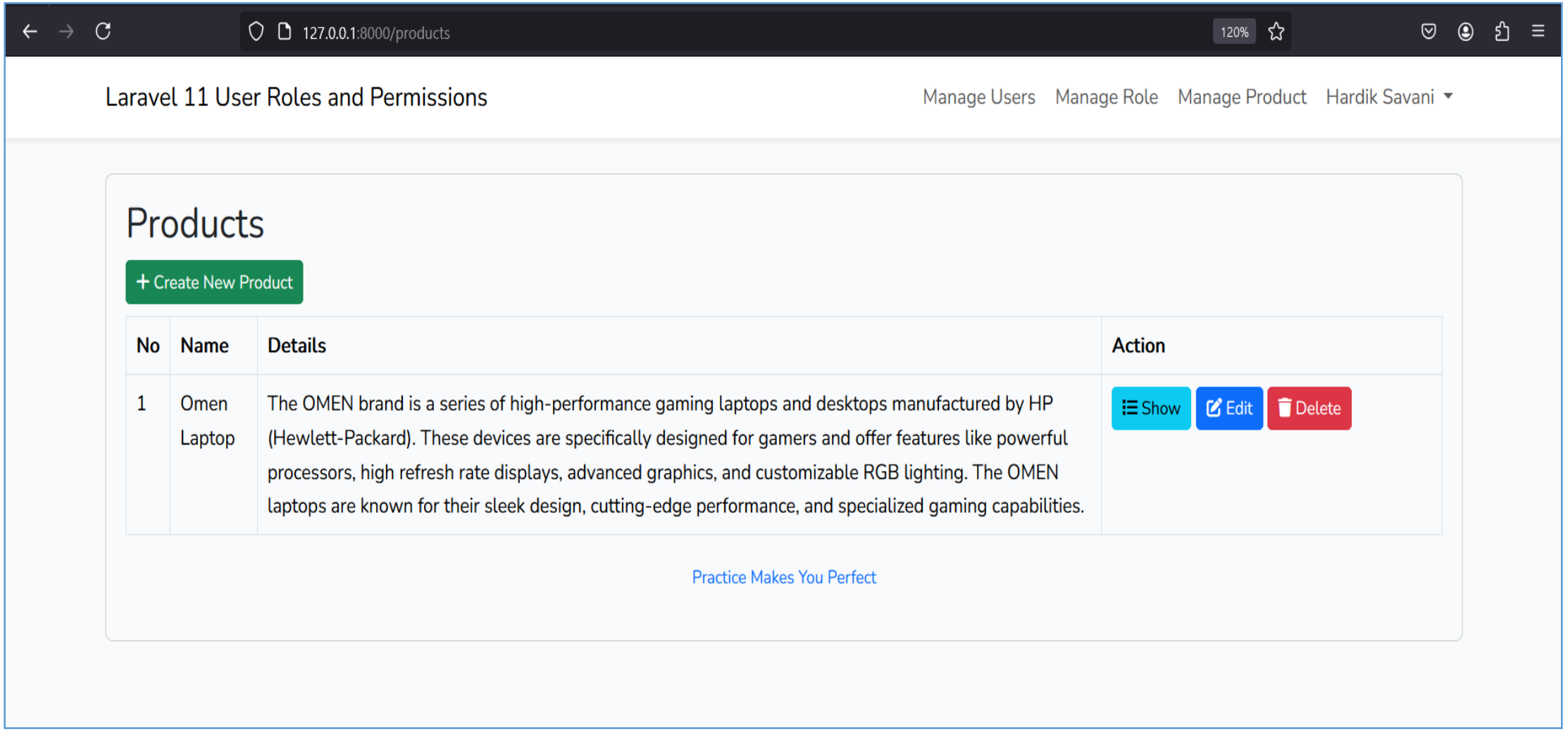
http://127.0.0.1:8000/roles/create



http://127.0.0.1:8000/products/create



http://127.0.0.1:8000/products



Database Tables

	id	name	email	email_verified_at	password	remember_token
Edit Copy Delete	1	Hardik Savani	admin@gmail.com	NULL	$2y$12$h/bMZixuwWlw6AGnrvPip.XKtIaq2FoHG54KC92hxM7...	NULL
Edit Copy Delete	2	yitayew	yitayewsolomon3@gmail.com	NULL	$2y$12$G6Qb8anJ0Zf1uH5565drnumrhcDH8HXJ6HH5oGPKkpA...	NULL
Edit Copy Delete	3	dumyuser	user@gmail.com	NULL	$2y$12$HXUJ9auzkbaKWzvdm/ft7ONAsegDN3cD1UfHD2B.HrH...	NULL

	id	name	guard_name	created_at	updated_at
Edit Copy Delete	1	Admin	web	2024-11-28 09:31:00	2024-11-28 09:31:00
Edit Copy Delete	2	supper Admin	web	2024-11-28 13:25:45	2024-11-28 13:25:45

Check all *With selected:* Edit Copy Delete Export

	id	name	detail	created_at	updated_at
Edit Copy Delete	1	Omen Laptop	The OMEN brand is a series of high-performance gam...	2024-11-28 13:27:43	2024-11-28 13:27:43

Check all *With selected:* Edit Copy Delete Export

Thank you!

Appreciate your action.