Ostad

Assignment Module 18

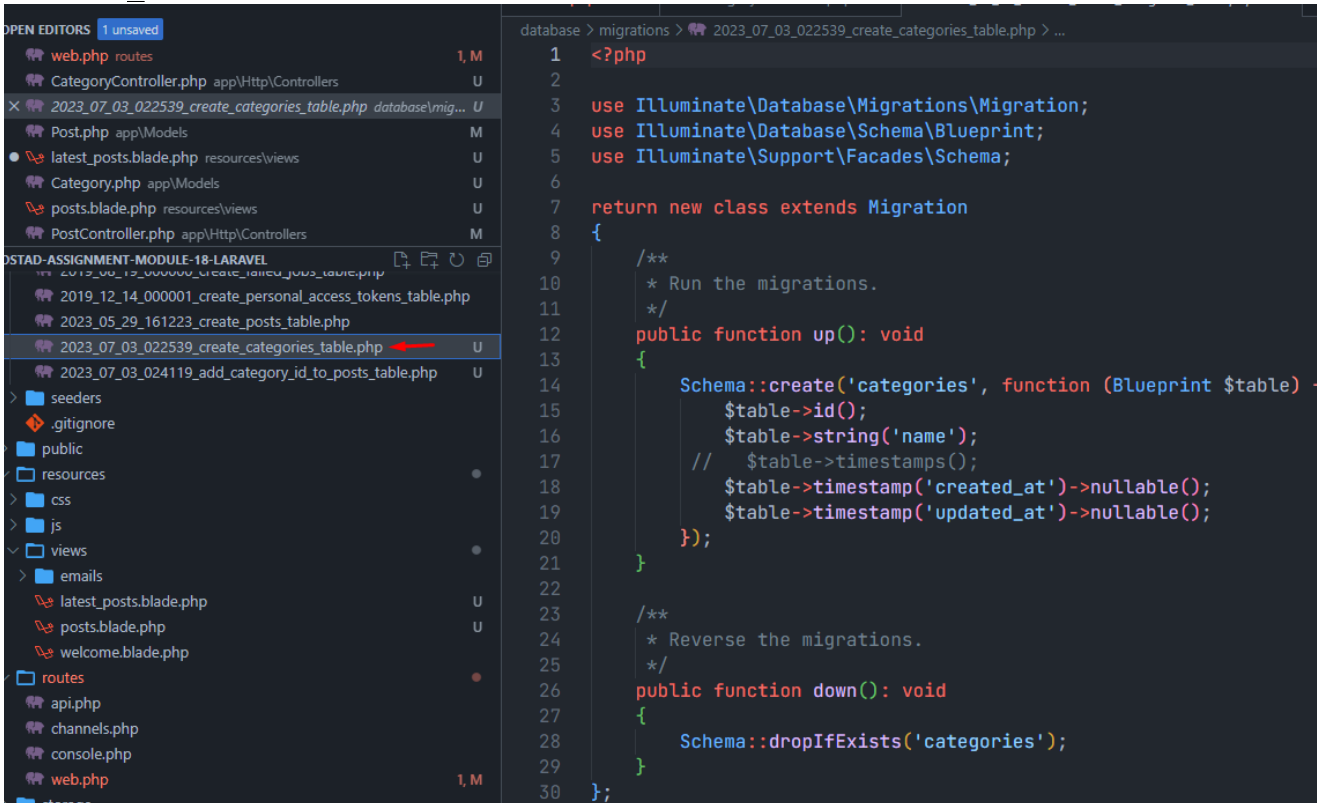
Full Name:	**Sakir Ahamed Bissas**
Phone:	**01995542277**
Date of Submission:	**03 July 2023**

https://github.com/sakirgit/Ostad-Assignment-module-18-laravel

Task: 1

Create a new migration file to add a new table named "categories" to the database. The table should have the following columns:

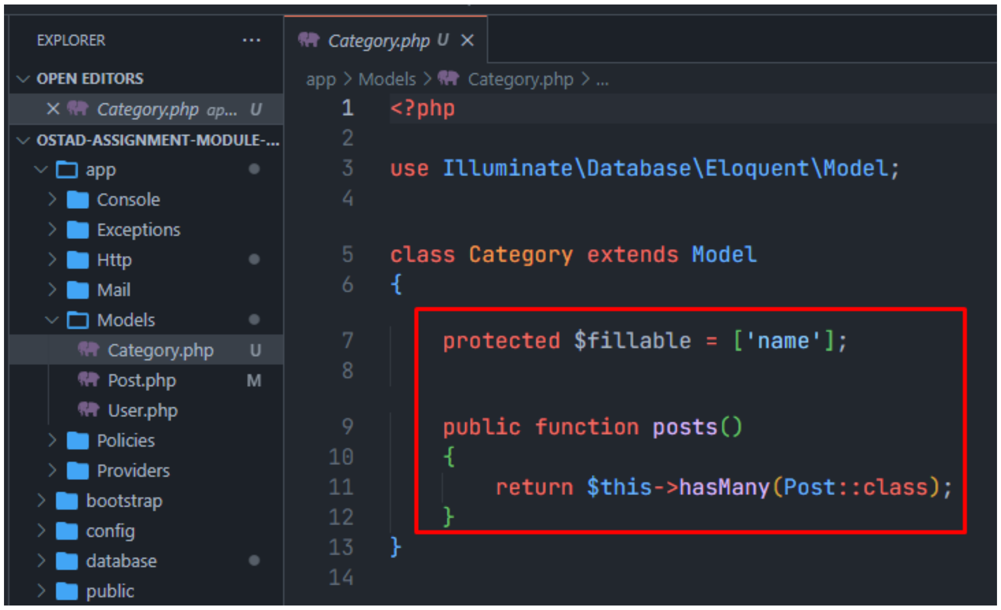
```
id (primary key, auto-increment)
name (string)
created_at (timestamp)
updated_at (timestamp)
```


OPEN EDITORS 1 unsaved
web.php routes 1, M
CategoryController.php app\Http\Controllers U
2023_07_03_022539_create_categories_table.php database\mig... U
Post.php app\Models M
latest_posts.blade.php resources\views U
Category.php app\Models U
posts.blade.php resources\views U
PostController.php app\Http\Controllers M
OSTAD-ASSIGNMENT-MODULE-18-LARAVEL
2019_12_14_000001_create_personal_access_tokens_table.php
2023_05_29_161223_create_posts_table.php
2023_07_03_022539_create_categories_table.php U
2023_07_03_024119_add_category_id_to_posts_table.php U
seeders
.gitignore
public
resources
css
js
views
emails
latest_posts.blade.php U
posts.blade.php U
welcome.blade.php
routes
api.php
channels.php
console.php
web.php 1, M
database > migrations > 2023_07_03_022539_create_categories_table.php > ...

```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.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table)
            $table->id();
            $table->string('name');
        //   $table->timestamps();
            $table->timestamp('created_at')->nullable();
            $table->timestamp('updated_at')->nullable();
        });
    }

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

Task: 2

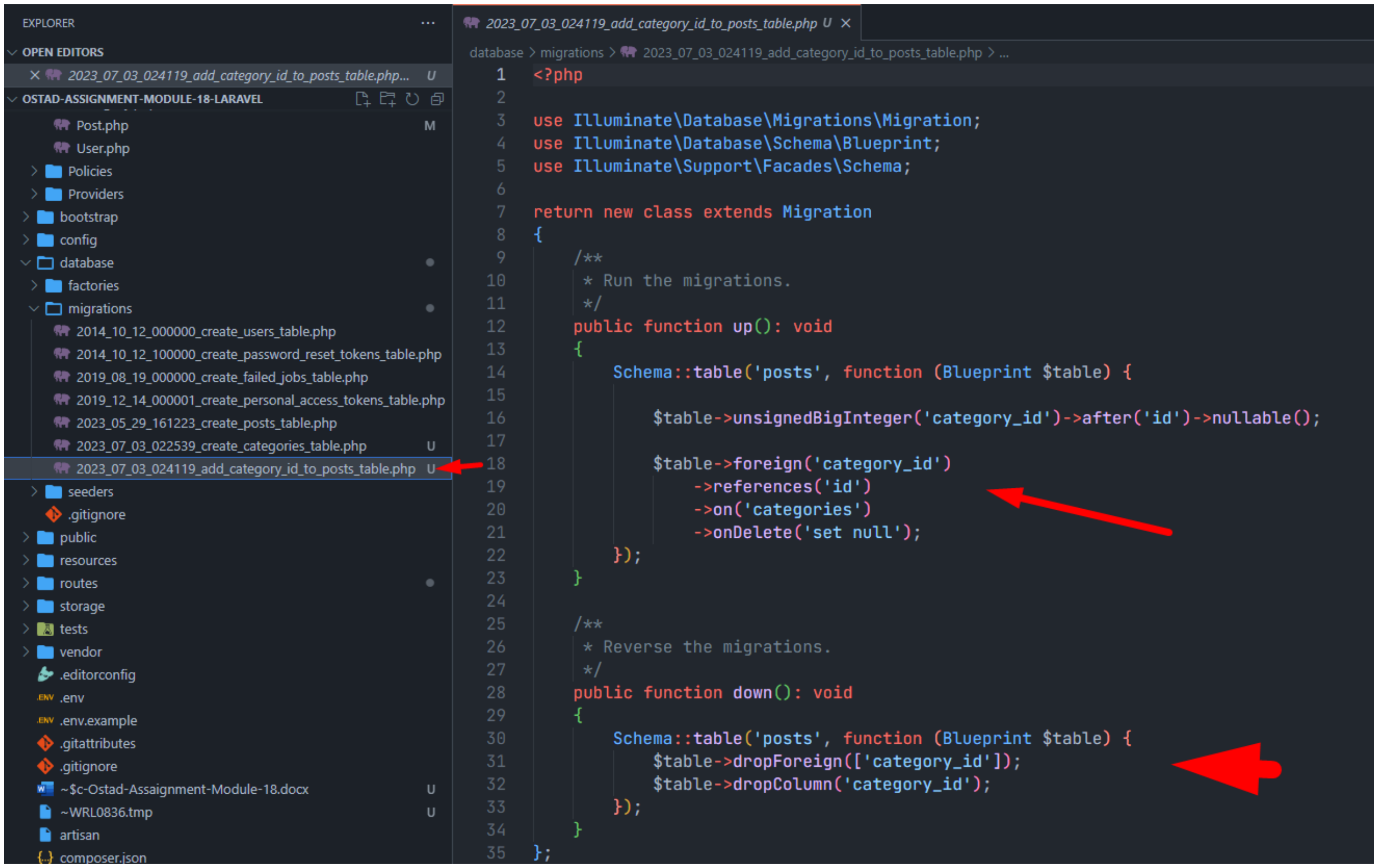
Create a new model named "Category" associated with the "categories" table. Define the necessary properties and relationships.



```php
<?php

use Illuminate\Database\Eloquent\Model;


class Category extends Model
{
    protected $fillable = ['name'];


    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}
```

Task 3:

Write a migration file to add a foreign key constraint to the "posts" table. The foreign key should reference the "categories" table on the "category_id" column.

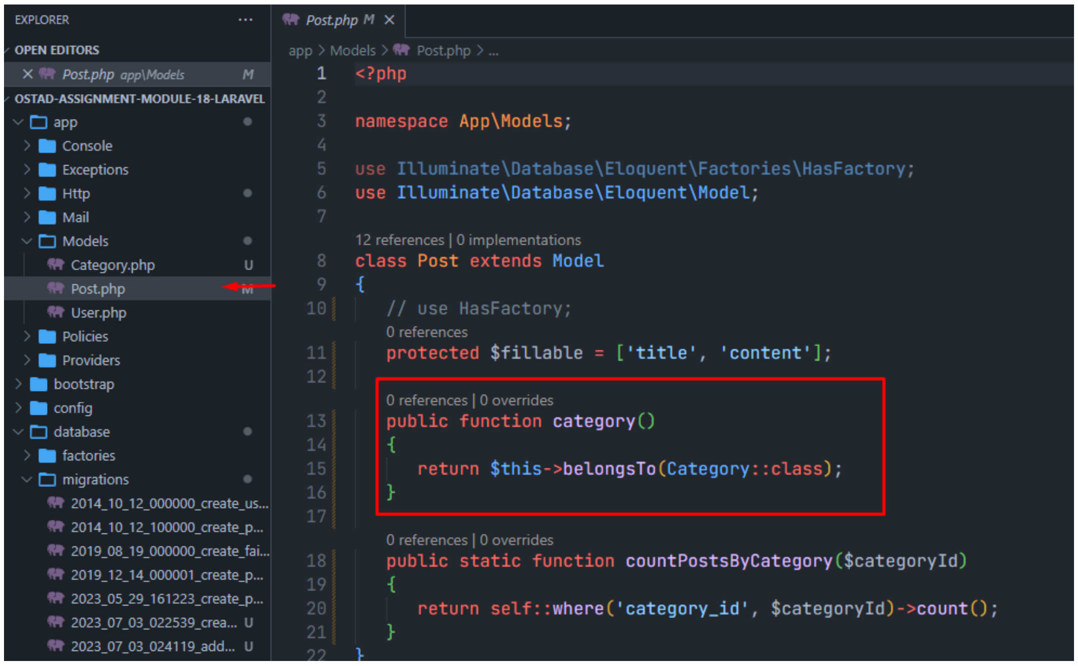


EXPLORER
OPEN EDITORS
2023_07_03_024119_add_category_id_to_posts_table.php U
OSTAD-ASSIGNMENT-MODULE-18-LARAVEL
Post.php M
User.php
Policies
Providers
bootstrap
config
database
factories
migrations
2014_10_12_000000_create_users_table.php
2014_10_12_100000_create_password_reset_tokens_table.php
2019_08_19_000000_create_failed_jobs_table.php
2019_12_14_000001_create_personal_access_tokens_table.php
2023_05_29_161223_create_posts_table.php
2023_07_03_022539_create_categories_table.php U
2023_07_03_024119_add_category_id_to_posts_table.php U
seeders
.gitignore
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
~$c-Ostad-Assaignment-Module-18.docx U
~WRL0836.tmp U
artisan
composer.json
database > migrations > 2023_07_03_024119_add_category_id_to_posts_table.php > ...

```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.
     */
    public function up(): void
    {
        Schema::table('posts', function (Blueprint $table) {

            $table->unsignedBigInteger('category_id')->after('id')->nullable();

            $table->foreign('category_id')
                ->references('id')
                ->on('categories')
                ->onDelete('set null');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('posts', function (Blueprint $table) {
            $table->dropForeign(['category_id']);
            $table->dropColumn('category_id');
        });
    }
};
```

Task 4:

Create a relationship between the "Post" and "Category" models. A post belongs to a category, and a category can have multiple posts.



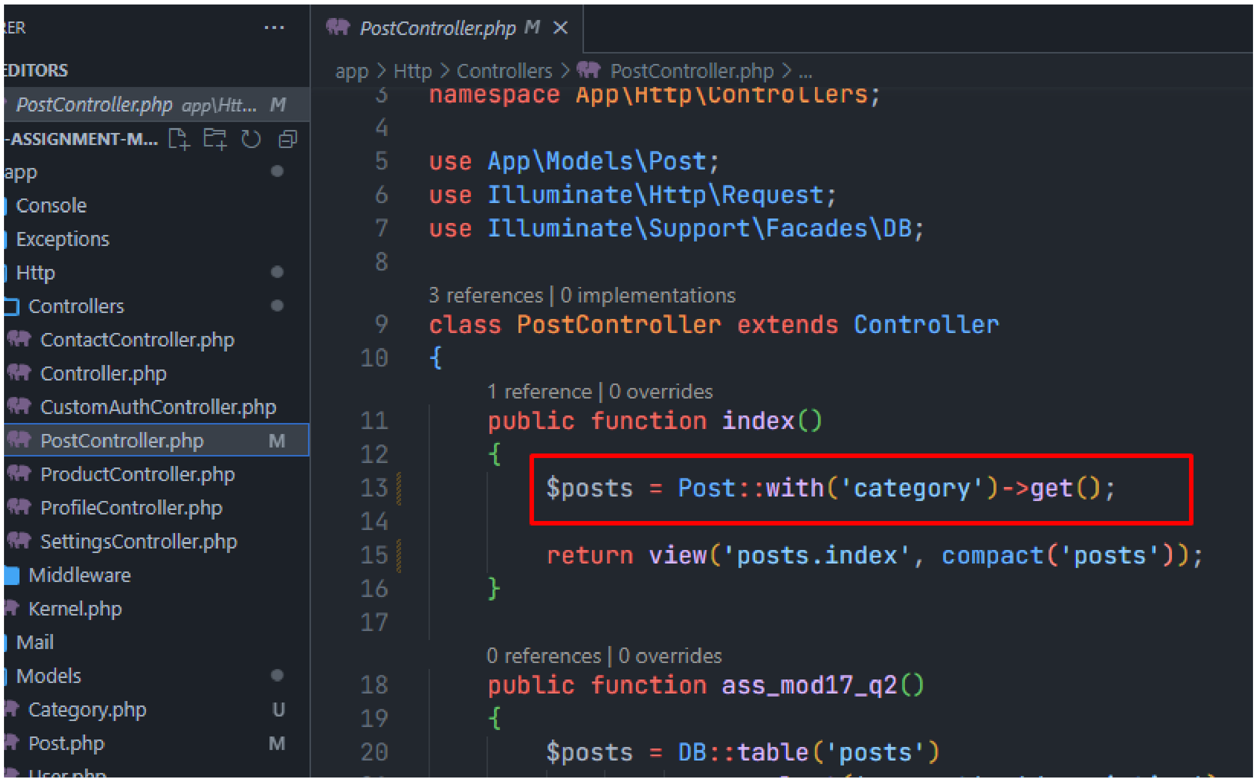
EXPLORER
OPEN EDITORS
Post.php app\Models M
OSTAD-ASSIGNMENT-MODULE-18-LARAVEL
app
Console
Exceptions
Http
Mail
Models
Category.php U
Post.php M
User.php
Policies
Providers
bootstrap
config
database
factories
migrations
2014_10_12_000000_create_us...
2014_10_12_100000_create_p...
2019_08_19_000000_create_fai...
2019_12_14_000001_create_p...
2023_05_29_161223_create_p...
2023_07_03_022539_crea... U
2023_07_03_024119_add... U
app > Models > Post.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

12 references | 0 implementations
class Post extends Model
{
    // use HasFactory;
    0 references
    protected $fillable = ['title', 'content'];

    0 references | 0 overrides
    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    0 references | 0 overrides
    public static function countPostsByCategory($categoryId)
    {
        return self::where('category_id', $categoryId)->count();
    }
}
```

Task 5:

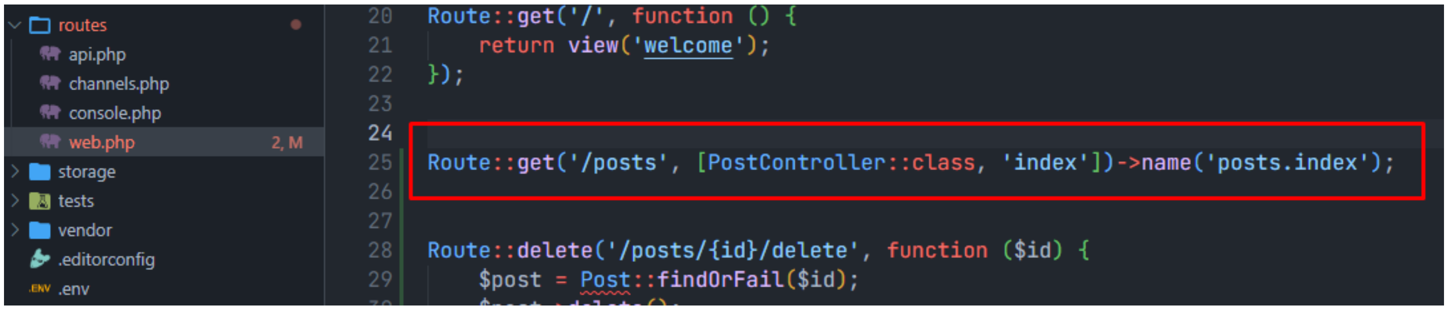
Write a query using Eloquent ORM to retrieve all posts along with their associated categories. Make sure to eager load the categories to optimize the query.

Controller:



```
namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

3 references | 0 implementations
class PostController extends Controller
{
    1 reference | 0 overrides
    public function index()
    {
        $posts = Post::with('category')->get();

        return view('posts.index', compact('posts'));
    }

    0 references | 0 overrides
    public function ass_mod17_q2()
    {
        $posts = DB::table('posts')
```

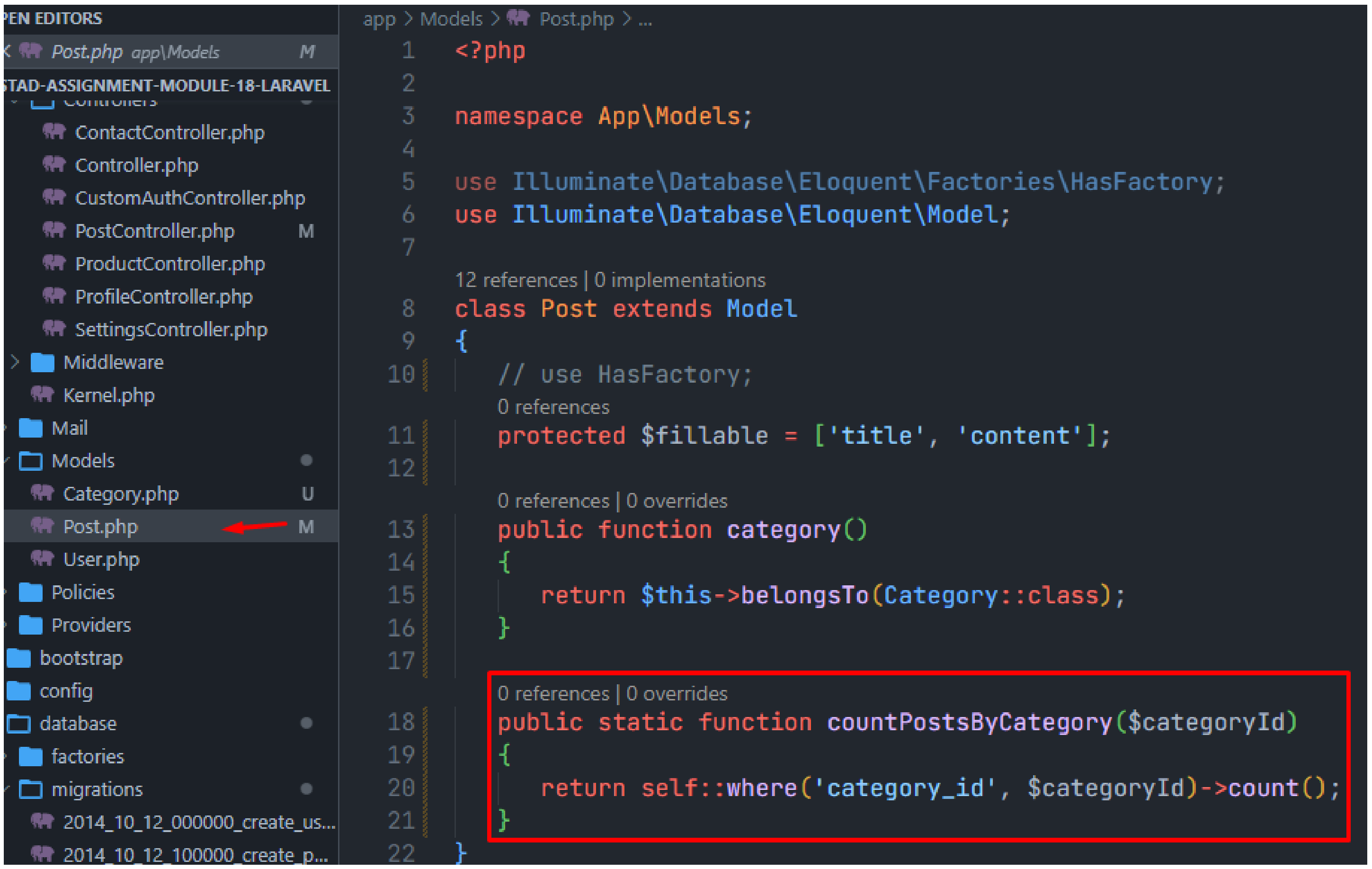
Route:



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


Route::get('/posts', [PostController::class, 'index'])->name('posts.index');


Route::delete('/posts/{id}/delete', function ($id) {
    $post = Post::findOrFail($id);
```

Task 6:

Implement a method in the "Post" model to get the total number of posts belonging to a specific category. The method should accept the category ID as a parameter and return the count.

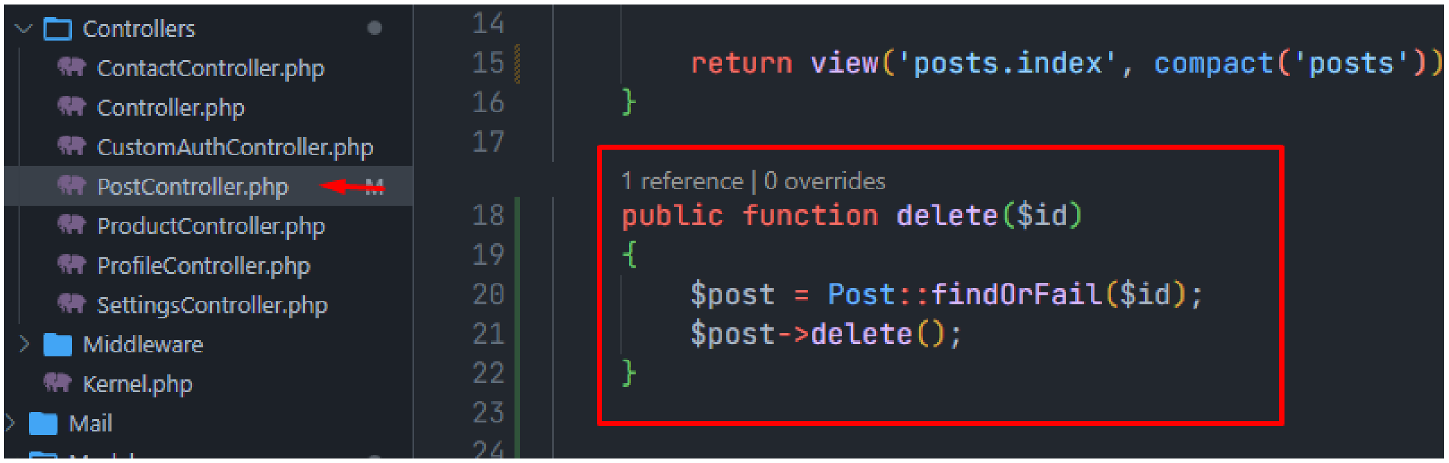


```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

12 references | 0 implementations
class Post extends Model
{
    // use HasFactory;
    0 references
    protected $fillable = ['title', 'content'];

    0 references | 0 overrides
    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    0 references | 0 overrides
    public static function countPostsByCategory($categoryId)
    {
        return self::where('category_id', $categoryId)->count();
    }
}
```

Task 7:

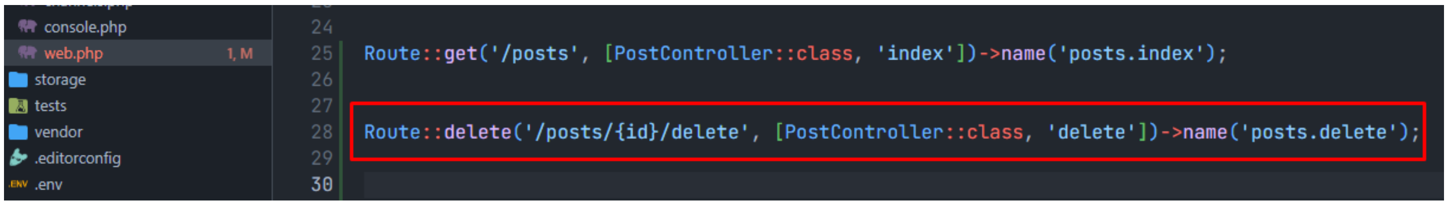
Create a new route in the web.php file to handle the following URL pattern: "/posts/{id}/delete". Implement the corresponding controller method to delete a post by its ID. Soft delete should be used.

Controller:



```php
        return view('posts.index', compact('posts'))
    }

    1 reference | 0 overrides
    public function delete($id)
    {
        $post = Post::findOrFail($id);
        $post->delete();
    }
```

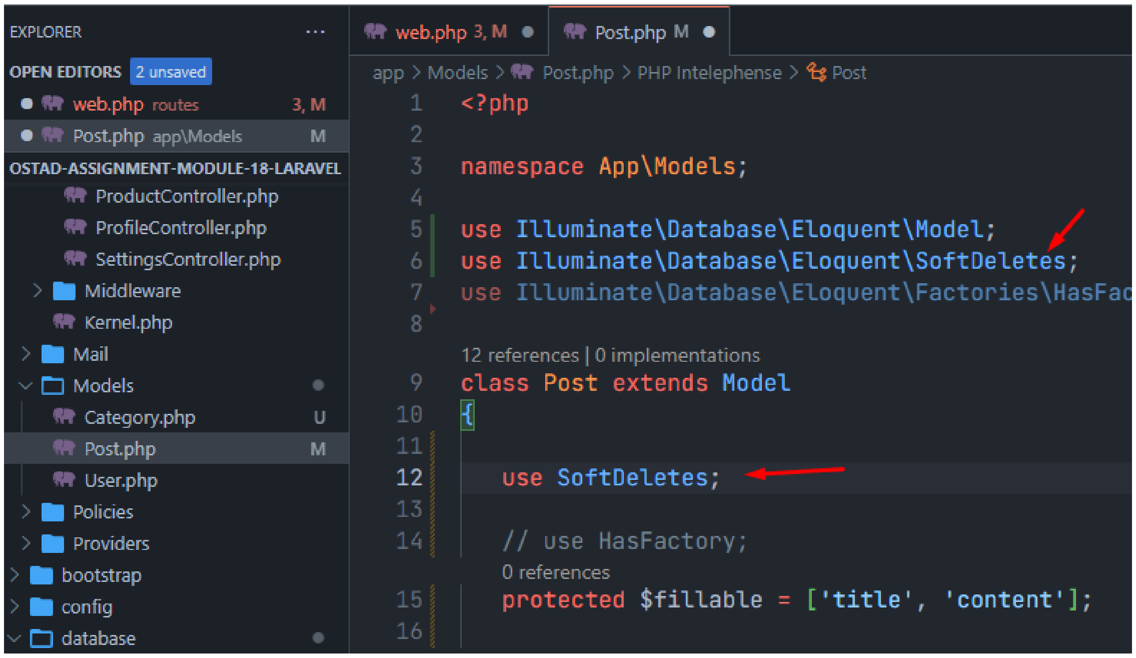
Route:



console.php
web.php 1, M
storage
tests
vendor
.editorconfig
.env

```
Route::get('/posts', [PostController::class, 'index'])->name('posts.index');


Route::delete('/posts/{id}/delete', [PostController::class, 'delete'])->name('posts.delete');

```

Model:



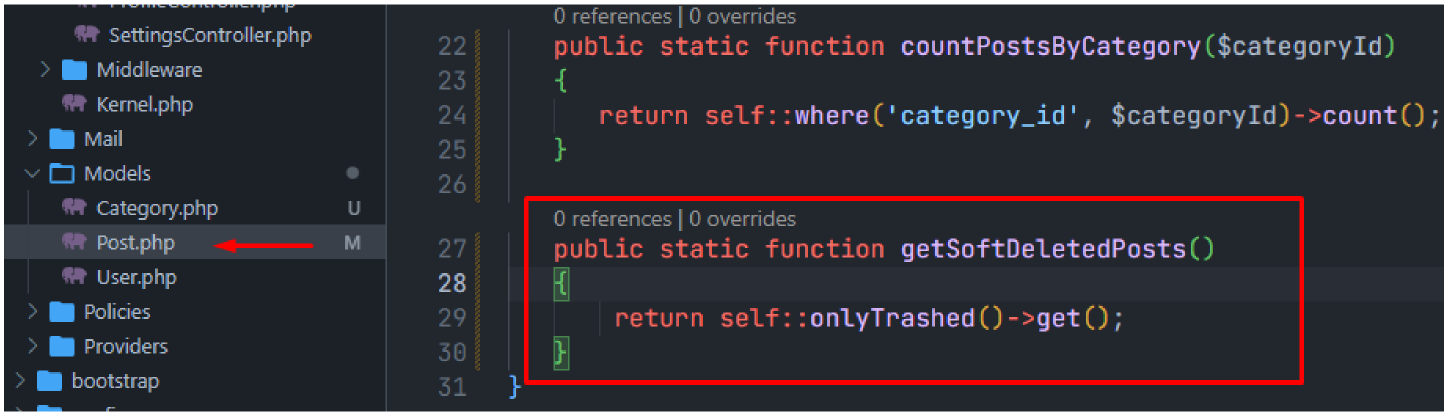
EXPLORER
OPEN EDITORS 2 unsaved
web.php routes 3, M
Post.php app\Models M
OSTAD-ASSIGNMENT-MODULE-18-LARAVEL
ProductController.php
ProfileController.php
SettingsController.php
Middleware
Kernel.php
Mail
Models
Category.php U
Post.php M
User.php
Policies
Providers
bootstrap
config
database

web.php 3, M · Post.php M
app > Models > Post.php > PHP Intelephense > Post

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Database\Eloquent\Factories\HasFac

12 references | 0 implementations
class Post extends Model
{

    use SoftDeletes;

    // use HasFactory;
    0 references
    protected $fillable = ['title', 'content'];

```

<u>**Task 8:**</u>

Implement a method in the "Post" model to get all posts that have been soft deleted. The method should return a collection of soft deleted posts.

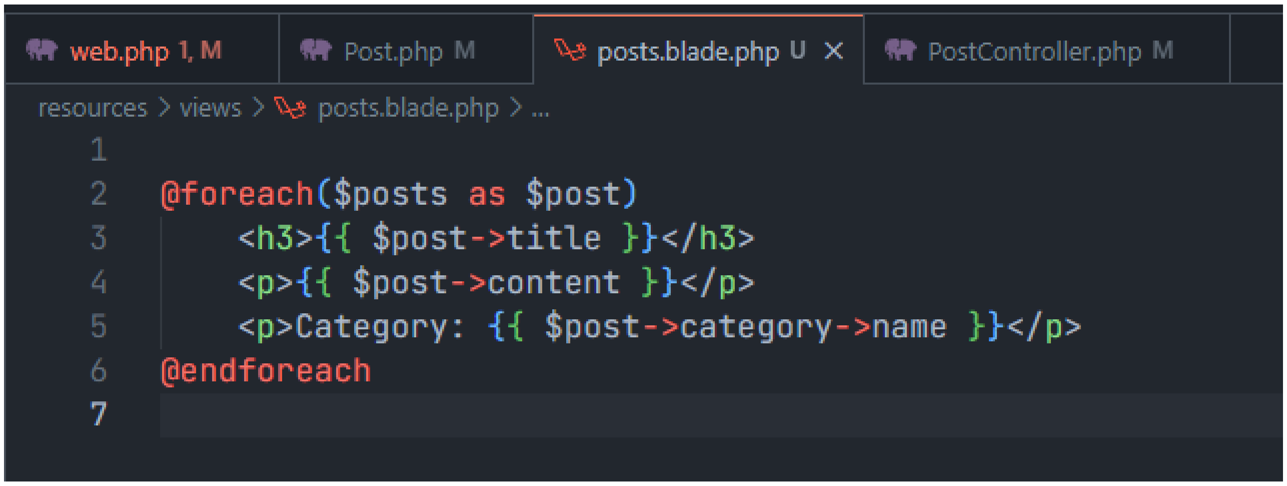


SettingsController.php
Middleware
Kernel.php
Mail
Models
Category.php U
Post.php M
User.php
Policies
Providers
bootstrap

```
    0 references | 0 overrides
    public static function countPostsByCategory($categoryId)
    {
        return self::where('category_id', $categoryId)->count();
    }

    0 references | 0 overrides
    public static function getSoftDeletedPosts()
    {
        return self::onlyTrashed()->get();
    }
}
```

Task 9:

Write a Blade template to display all posts and their associated categories. Use a loop to iterate over the posts and display their details.



web.php 1, M | Post.php M | posts.blade.php U × | PostController.php M

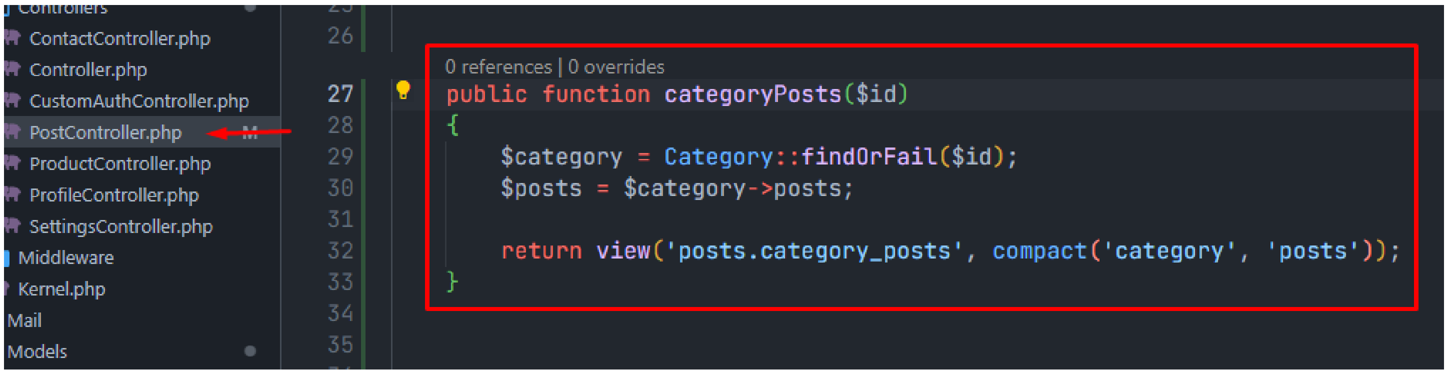
resources > views > posts.blade.php > ...

```

@foreach($posts as $post)
    <h3>{{ $post->title }}</h3>
    <p>{{ $post->content }}</p>
    <p>Category: {{ $post->category->name }}</p>
@endforeach

```

Task 10:

Create a new route in the web.php file to handle the following URL pattern: "/categories/{id}/posts". Implement the corresponding controller method to retrieve all posts belonging to a specific category. The category ID should be passed as a parameter to the method.

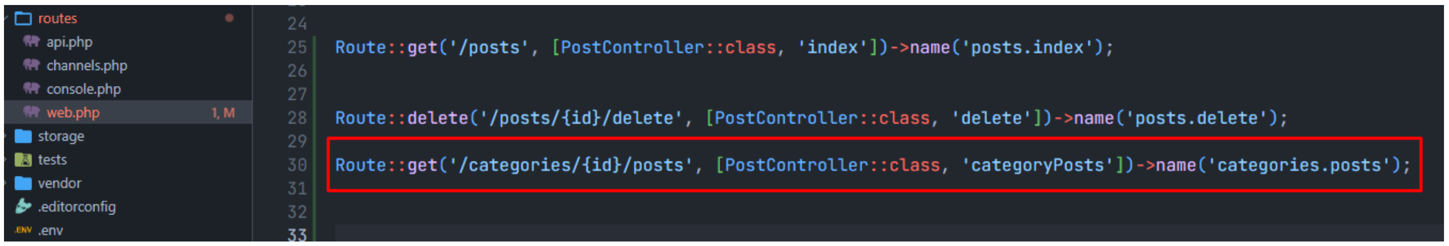
Controller:



Controllers: ContactController.php, Controller.php, CustomAuthController.php, PostController.php, ProductController.php, ProfileController.php, SettingsController.php, Middleware, Kernel.php, Mail, Models

```
0 references | 0 overrides
public function categoryPosts($id)
{
    $category = Category::findOrFail($id);
    $posts = $category->posts;

    return view('posts.category_posts', compact('category', 'posts'));
}
```

Route:

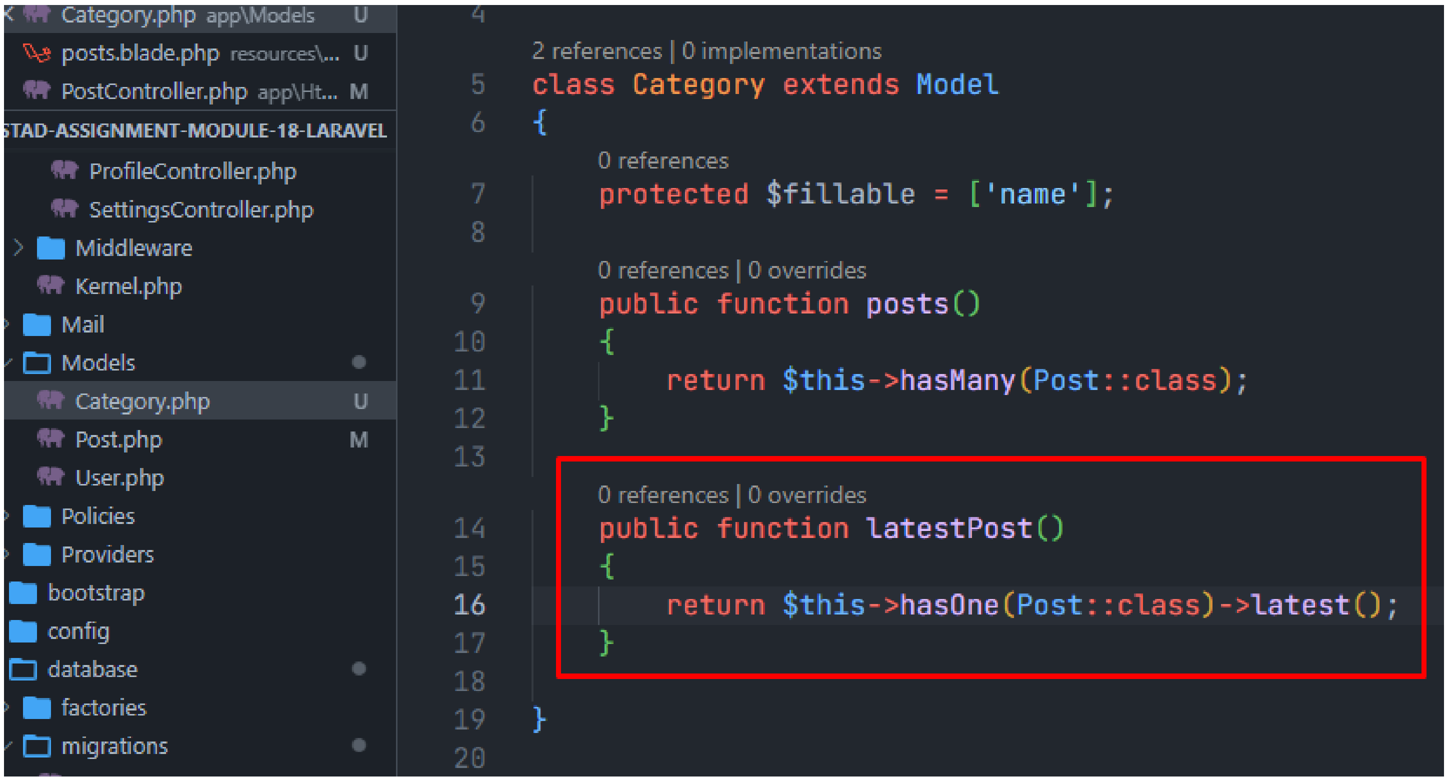


routes: api.php, channels.php, console.php, web.php 1, M, storage, tests, vendor, .editorconfig, .env

```
Route::get('/posts', [PostController::class, 'index'])->name('posts.index');


Route::delete('/posts/{id}/delete', [PostController::class, 'delete'])->name('posts.delete');

Route::get('/categories/{id}/posts', [PostController::class, 'categoryPosts'])->name('categories.posts');
```

Task 11:

Implement a method in the "Category" model to get the latest post associated with the category. The method should return the post object.

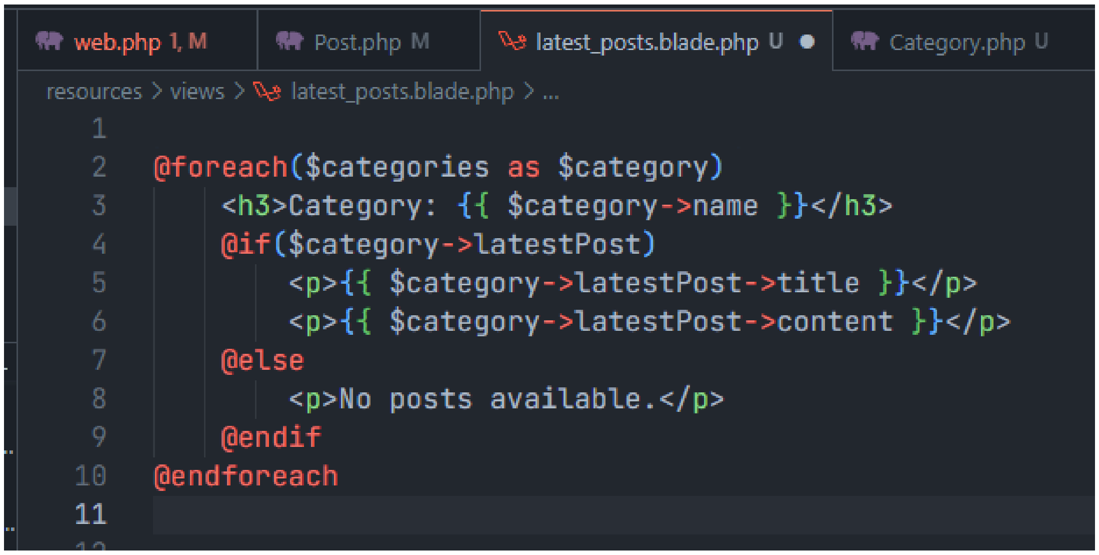


```php
2 references | 0 implementations
class Category extends Model
{
    0 references
    protected $fillable = ['name'];

    0 references | 0 overrides
    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    0 references | 0 overrides
    public function latestPost()
    {
        return $this->hasOne(Post::class)->latest();
    }
}
```

Task 12:

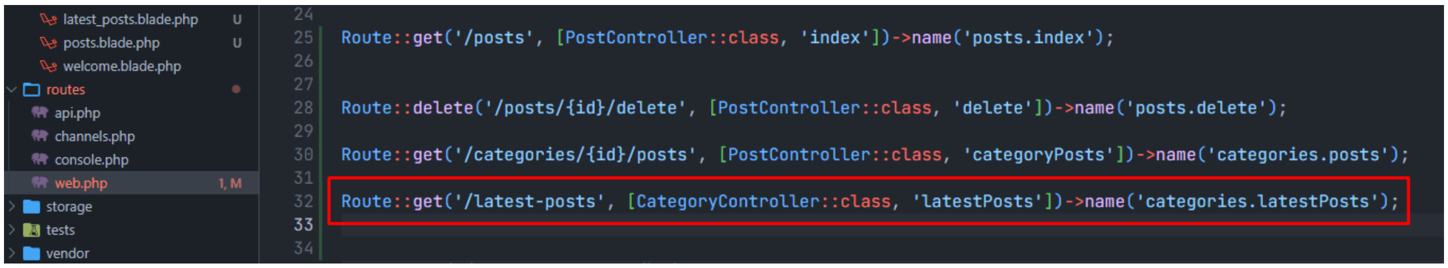
Write a Blade template to display the latest post for each category. Use a loop to iterate over the categories and display the post details.

View:



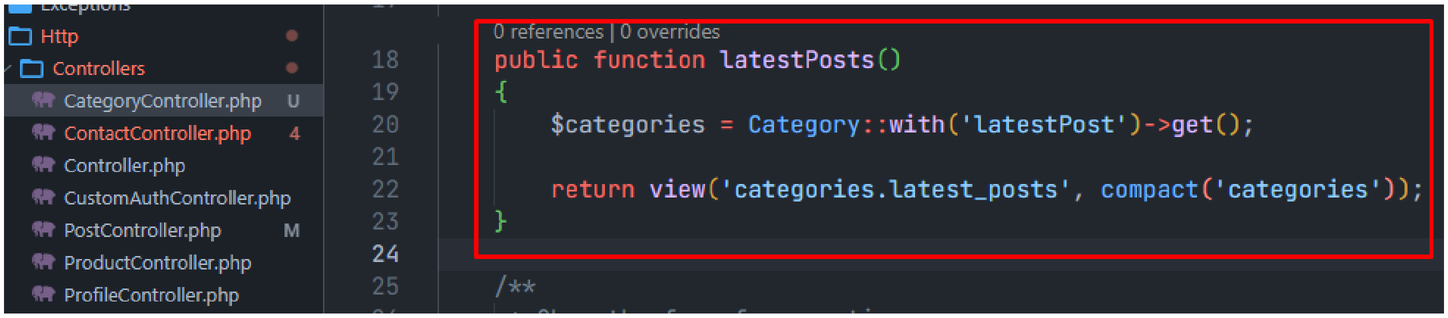
```blade

@foreach($categories as $category)
    <h3>Category: {{ $category->name }}</h3>
    @if($category->latestPost)
        <p>{{ $category->latestPost->title }}</p>
        <p>{{ $category->latestPost->content }}</p>
    @else
        <p>No posts available.</p>
    @endif
@endforeach
```

Route:



```
Route::get('/posts', [PostController::class, 'index'])->name('posts.index');


Route::delete('/posts/{id}/delete', [PostController::class, 'delete'])->name('posts.delete');

Route::get('/categories/{id}/posts', [PostController::class, 'categoryPosts'])->name('categories.posts');

Route::get('/latest-posts', [CategoryController::class, 'latestPosts'])->name('categories.latestPosts');
```

Controller:



```
public function latestPosts()
{
    $categories = Category::with('latestPost')->get();

    return view('categories.latest_posts', compact('categories'));
}

/**
```