Task 1:

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

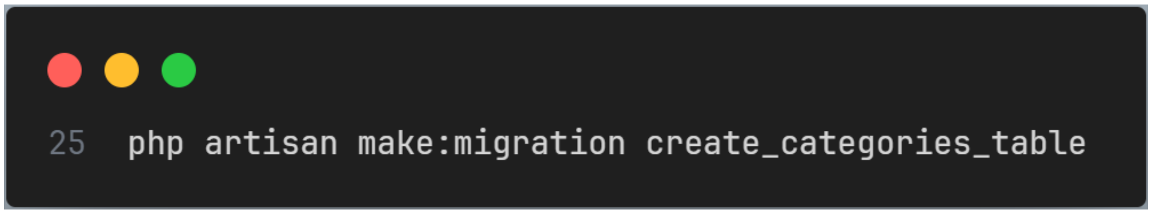
id (primary key, auto-increment)

name (string)

created_at (timestamp)

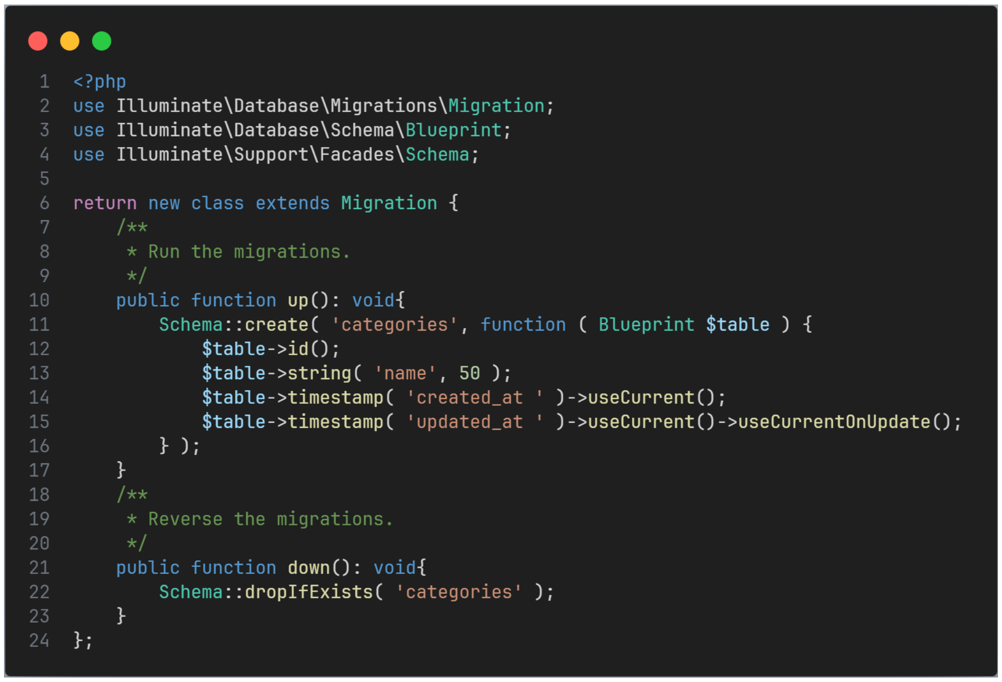
updated_at (timestamp)

Create migration command:



```
php artisan make:migration create_categories_table
```

Migration file:

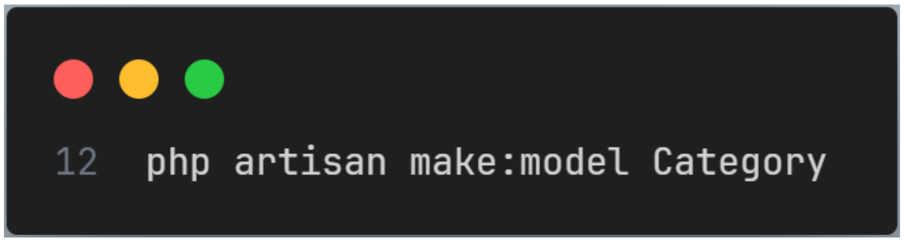


```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    /**
     * Run the migrations.
     */
    public function up(): void{
        Schema::create( 'categories', function ( Blueprint $table ) {
            $table->id();
            $table->string( 'name', 50 );
            $table->timestamp( 'created_at ' )->useCurrent();
            $table->timestamp( 'updated_at ' )->useCurrent()->useCurrentOnUpdate();
        } );
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void{
        Schema::dropIfExists( 'categories' );
    }
};
```

Task 2:

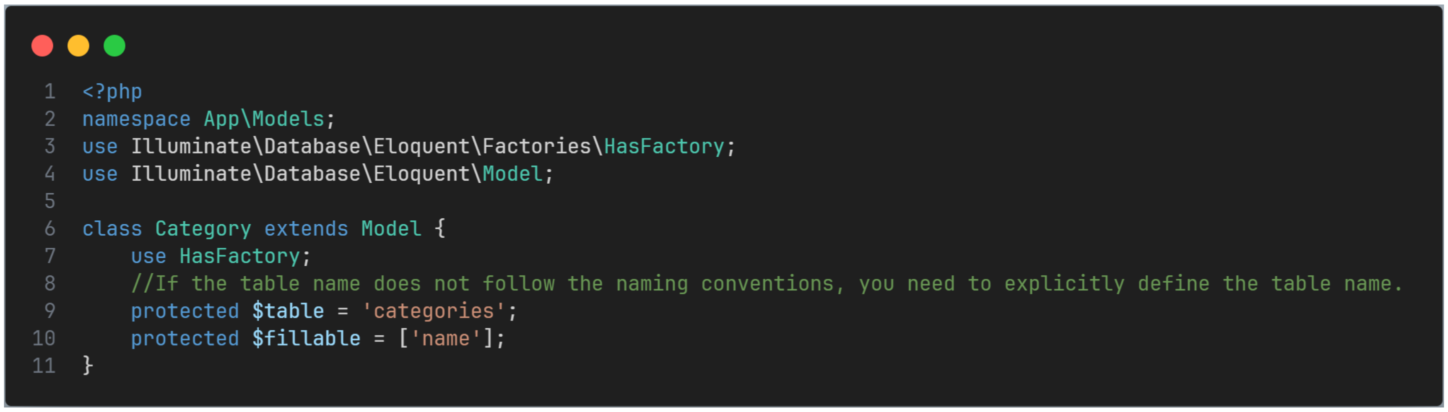
Create a new model named "Category" associated with the "categories" table. Define the necessary properties and relationships.

Create Model command:



```
php artisan make:model Category
```

Category Model:

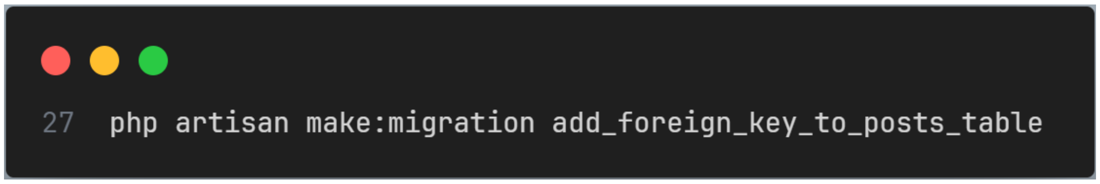


```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model {
    use HasFactory;
    //If the table name does not follow the naming conventions, you need to explicitly define the table name.
    protected $table = 'categories';
    protected $fillable = ['name'];
}
```

Task 3:

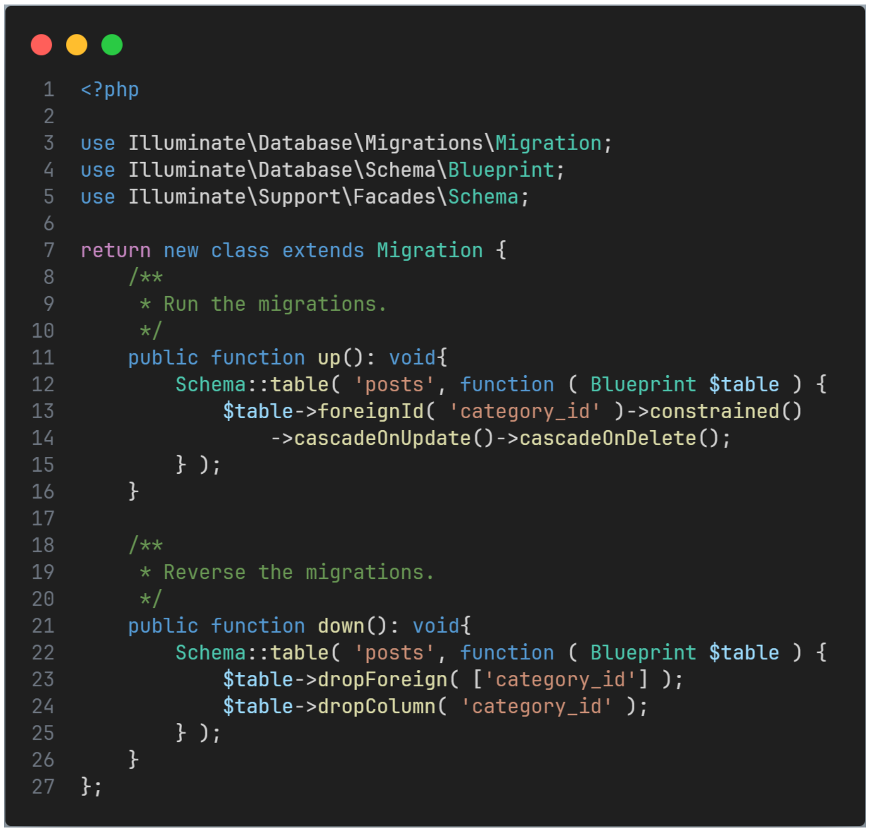
Write a migration file to add a foreign key constraint to the "posts" table. The foreign key should reference the "categories" table on the "category_id" column.

Create migration command add a foreign key constraint to the post table:



```
php artisan make:migration add_foreign_key_to_posts_table
```

Add foreign key migration file:

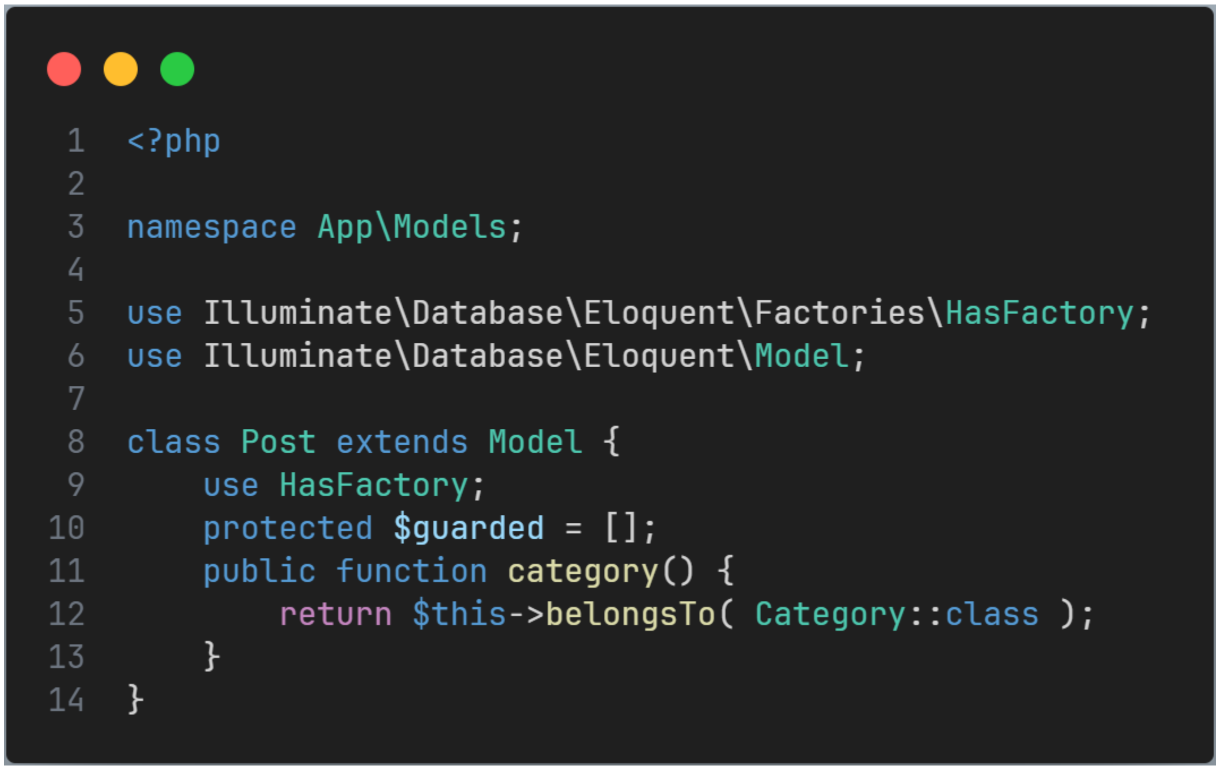


```php
<?php

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->foreignId( 'category_id' )->constrained()
                ->cascadeOnUpdate()->cascadeOnDelete();
        } );
    }

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

Task 4:

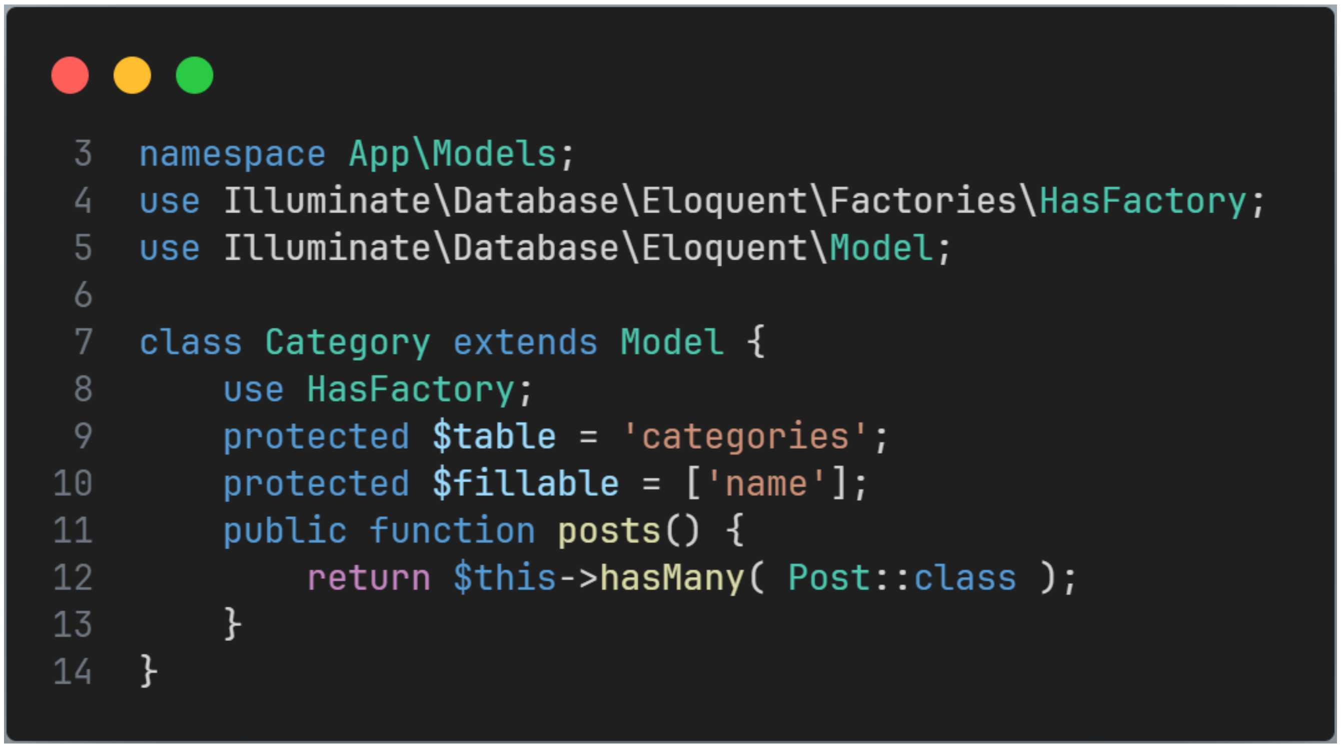
Create a relationship between the "Post" and "Category" models. A post belongs to a category, and a category can have multiple posts.

Post belongs to category:



```php
<?php

namespace App\Models;

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

class Post extends Model {
    use HasFactory;
    protected $guarded = [];
    public function category() {
        return $this->belongsTo( Category::class );
    }
}
```

Category can have multiple posts:

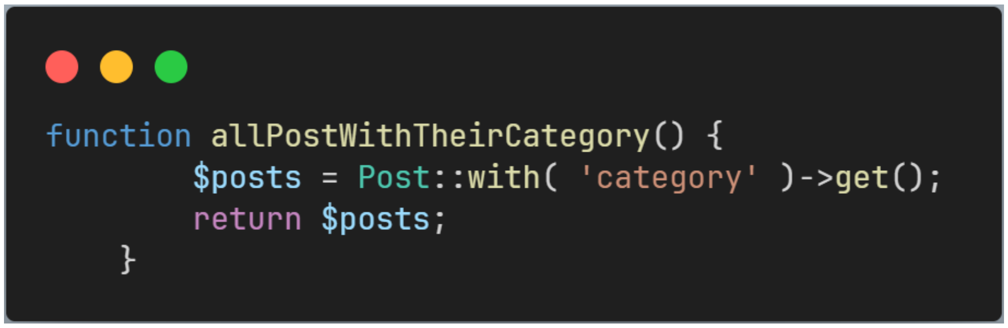


```php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model {
    use HasFactory;
    protected $table = 'categories';
    protected $fillable = ['name'];
    public function posts() {
        return $this->hasMany( Post::class );
    }
}
```

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 method:

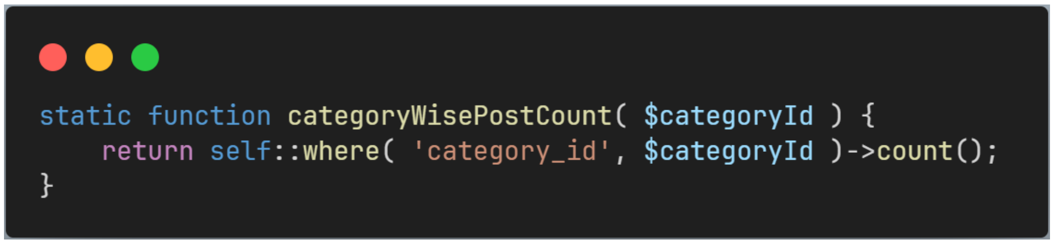


```php
function allPostWithTheirCategory() {
        $posts = Post::with( 'category' )->get();
        return $posts;
    }
```

Task 6:

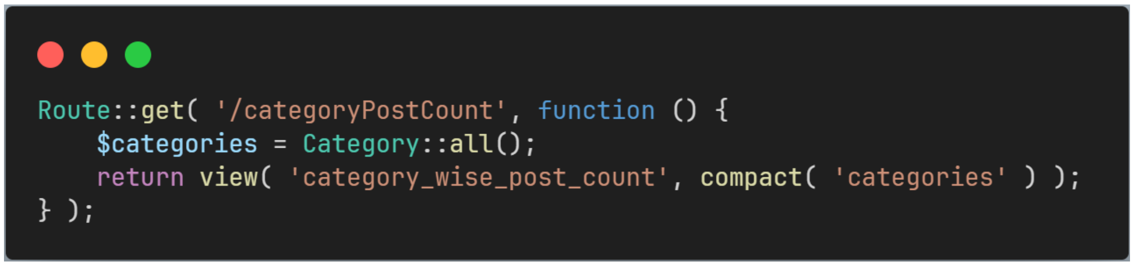
Implement a method in the "Post" model to get the total number of posts belonging to a specific category. The method should accept the category ID as a parameter and return the count.

Implement a method in the post model:



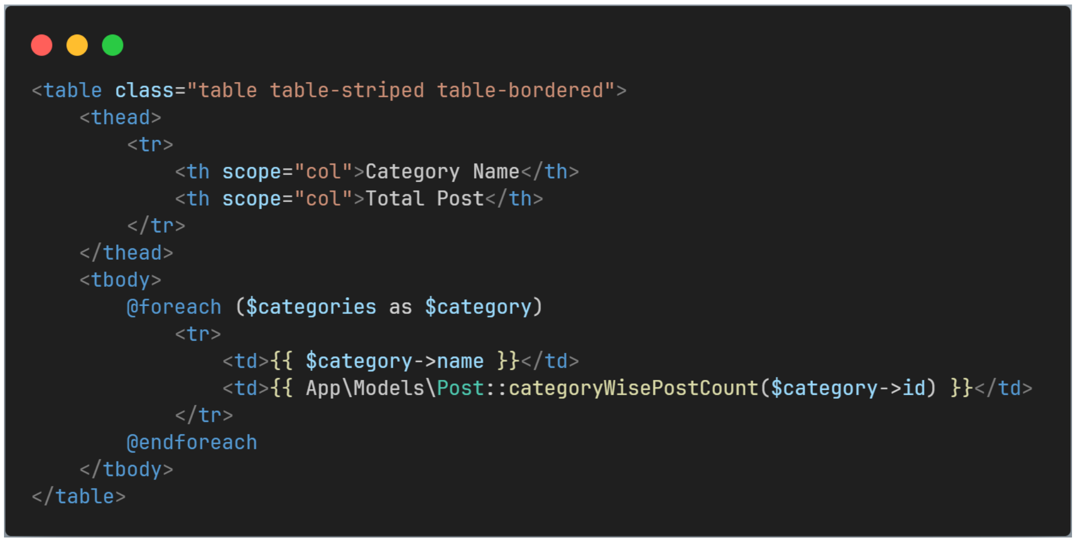
```php
static function categoryWisePostCount( $categoryId ) {
    return self::where( 'category_id', $categoryId )->count();
}
```

Route:



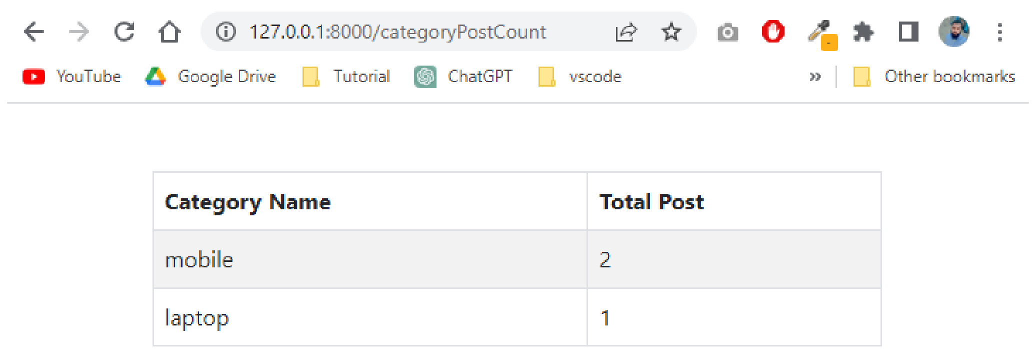
```php
Route::get( '/categoryPostCount', function () {
    $categories = Category::all();
    return view( 'category_wise_post_count', compact( 'categories' ) );
} );
```

blade:



```html
<table class="table table-striped table-bordered">
    <thead>
        <tr>
            <th scope="col">Category Name</th>
            <th scope="col">Total Post</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($categories as $category)
            <tr>
                <td>{{ $category->name }}</td>
                <td>{{ App\Models\Post::categoryWisePostCount($category->id) }}</td>
            </tr>
        @endforeach
    </tbody>
</table>
```

OutPut:



Category Name	Total Post
mobile	2
laptop	1

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.

Use the SoftDeletes trait Post model:

```
use HasFactory, SoftDeletes;
```

Create a migration file for add a `deleted_at` column posts table:

```
php artisan make:migration add_deleted_at_to_posts_table
```

Run migration:

```
php artisan migrate
```

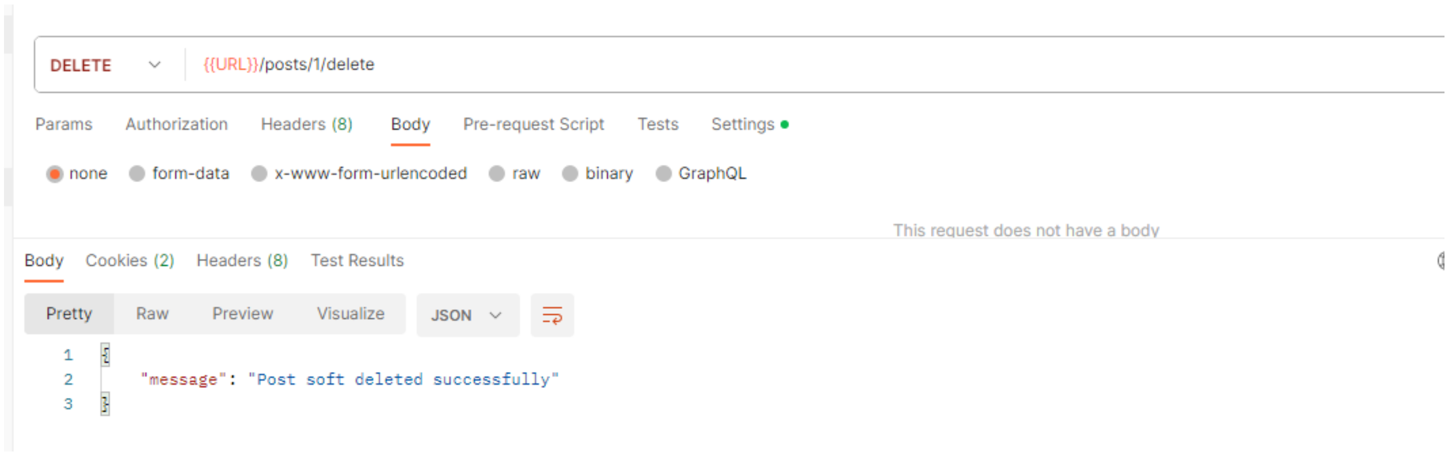
Route:

```
Route::delete( '/posts/{id}/delete', [AssignmentController::class, 'softDelete'] );
```

Controller:

```php
public function softDelete( Request $request ) {
    $post = Post::find( $request->id );
    if ( !$post ) {
        return response()->json( ['message' => 'Post not found'], 404 );
    }
    $post->delete();
    return response()->json( ['message' => 'Post soft deleted successfully'] );
}
```

Output:



DELETE {{URL}}/posts/1/delete

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies (2) Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "Post soft deleted successfully"
}
```

Task 8:

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.

Implement Method in Post Model:

```php
public static function softDeletedPosts(){
    return self::onlyTrashed()->get();
}
```

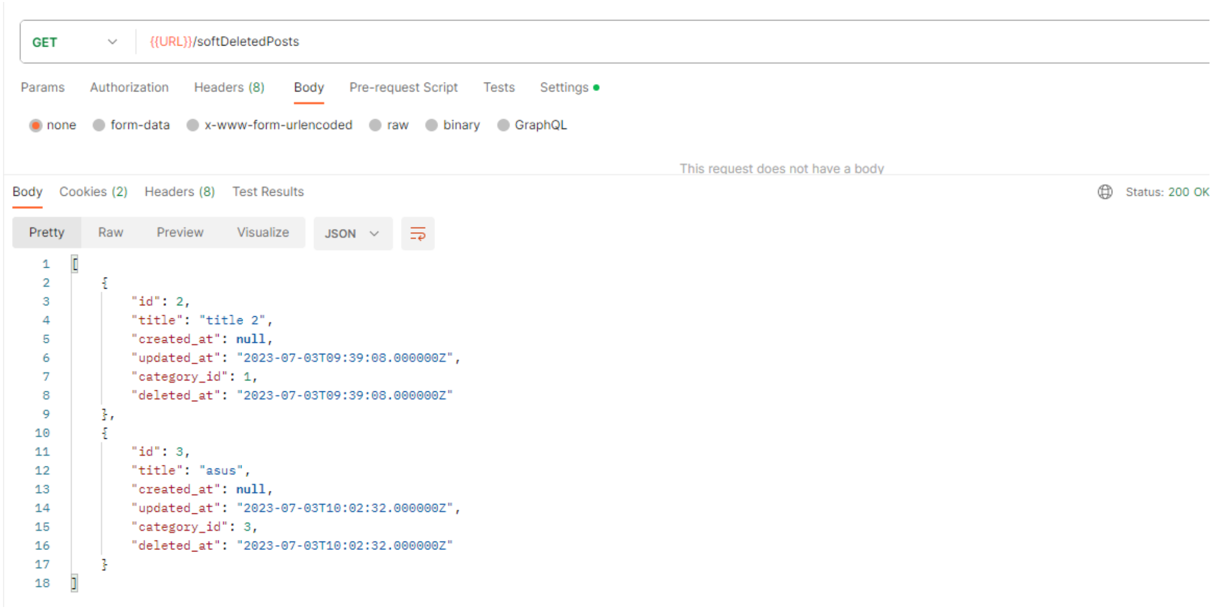
Route:

```php
Route::get( '/softDeletedPosts', [AssignmentController::class, 'softDeletedPosts'] );
```

Controller:

```php
public function softDeletedPosts() {
    $softDeletedPosts = Post::softDeletedPosts();
    return $softDeletedPosts;
}
```

Output:



GET {{URL}}/softDeletedPosts

Params Authorization Headers (8) Body Pre-request Script Tests Settings

● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

This request does not have a body

Body Cookies (2) Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```json
[
    {
        "id": 2,
        "title": "title 2",
        "created_at": null,
        "updated_at": "2023-07-03T09:39:08.000000Z",
        "category_id": 1,
        "deleted_at": "2023-07-03T09:39:08.000000Z"
    },
    {
        "id": 3,
        "title": "asus",
        "created_at": null,
        "updated_at": "2023-07-03T10:02:32.000000Z",
        "category_id": 3,
        "deleted_at": "2023-07-03T10:02:32.000000Z"
    }
]
```

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.

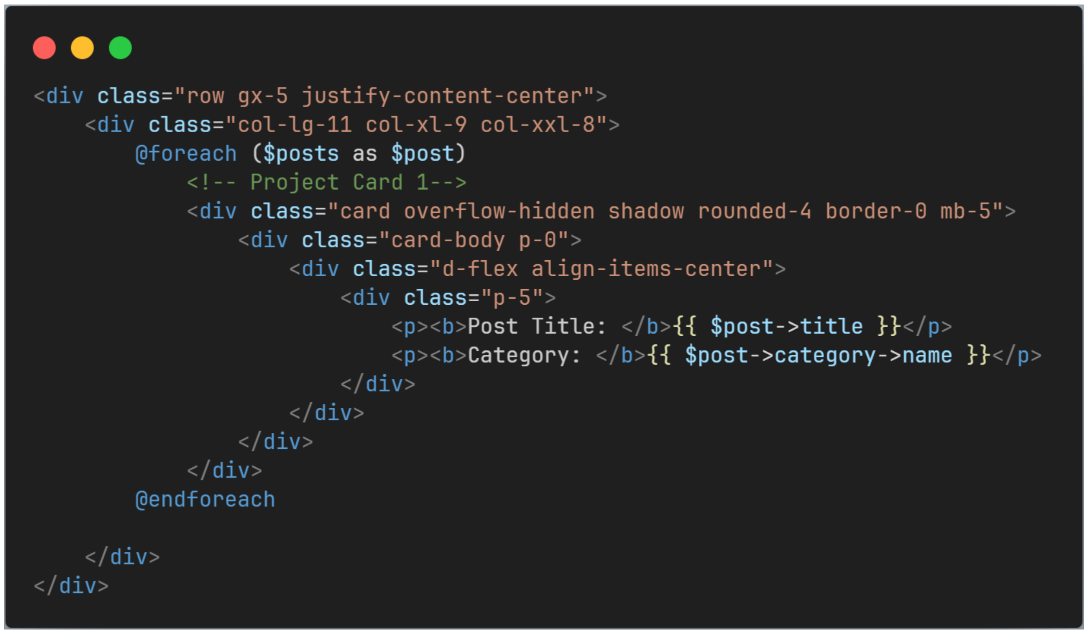
Route:

```php
Route::get( '/all_post_with_category', [AssignmentController::class, 'all_post_with_category'] );
```

Controller:

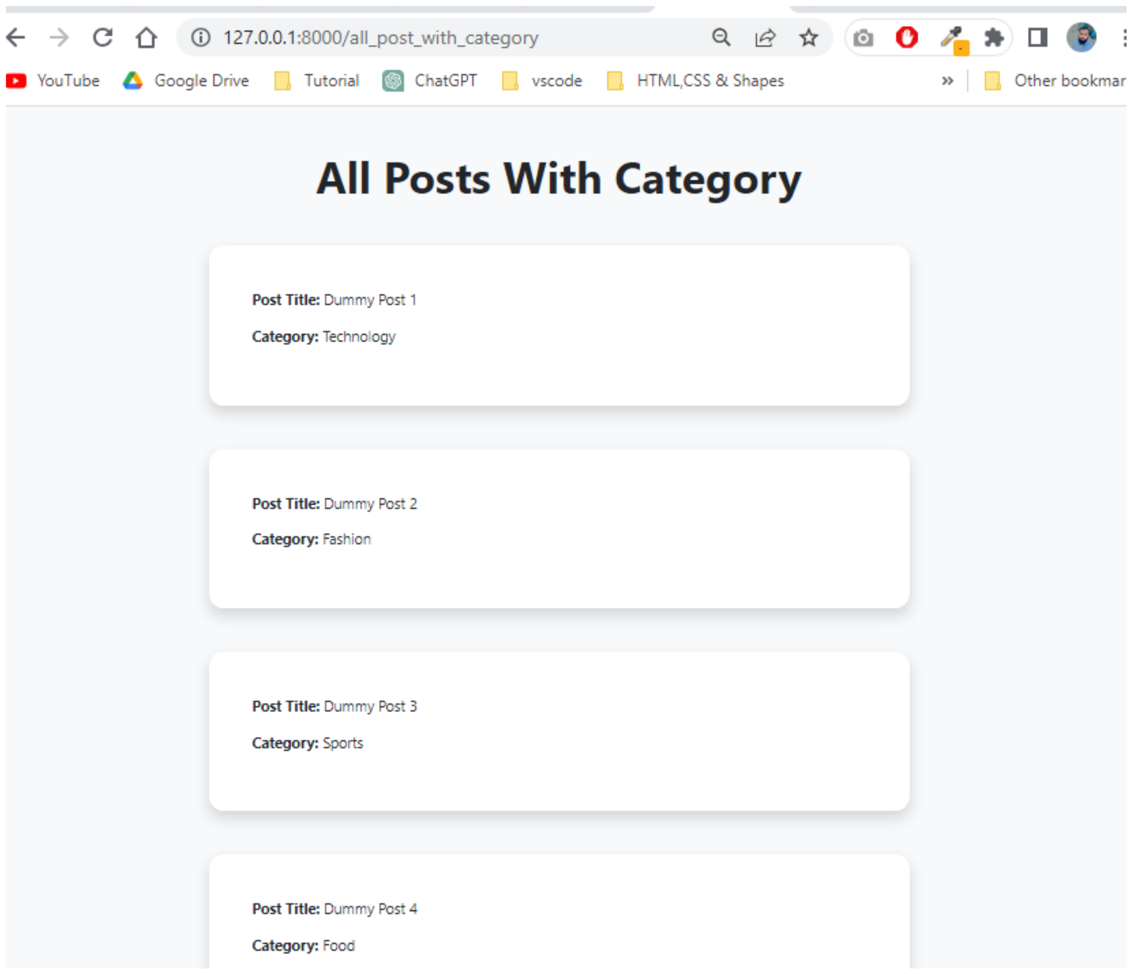
```php
public function all_post_with_category() {
    $posts = Post::with( 'category' )->get();
    return view( 'display_all_post_with_category', compact( 'posts' ) );
}
```

Blade:



```
<div class="row gx-5 justify-content-center">
    <div class="col-lg-11 col-xl-9 col-xxl-8">
        @foreach ($posts as $post)
            <!-- Project Card 1-->
            <div class="card overflow-hidden shadow rounded-4 border-0 mb-5">
                <div class="card-body p-0">
                    <div class="d-flex align-items-center">
                        <div class="p-5">
                            <p><b>Post Title: </b>{{ $post->title }}</p>
                            <p><b>Category: </b>{{ $post->category->name }}</p>
                        </div>
                    </div>
                </div>
            </div>
        @endforeach

    </div>
</div>
```

Output:



All Posts With Category

Post Title: Dummy Post 1

Category: Technology

Post Title: Dummy Post 2

Category: Fashion

Post Title: Dummy Post 3

Category: Sports

Post Title: Dummy Post 4

Category: Food

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.

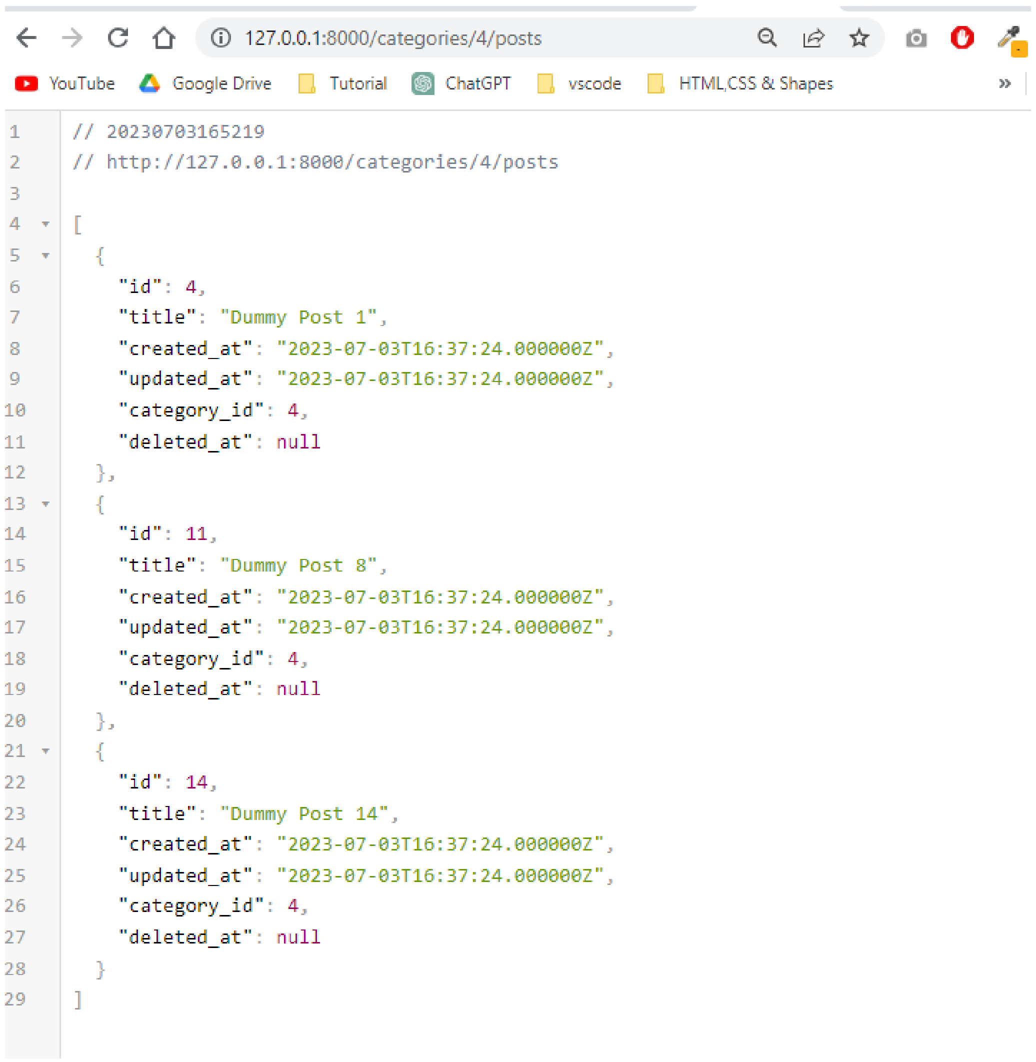
Route:

```php
Route::get( '/categories/{id}/posts', [AssignmentController::class, 'specificCatPosts'] );
```

Controller:

```php
public function specificCatPosts( Request $request ) {
    $posts = Post::where( 'category_id', $request->id )->get();
    return $posts;
}
```

Output:



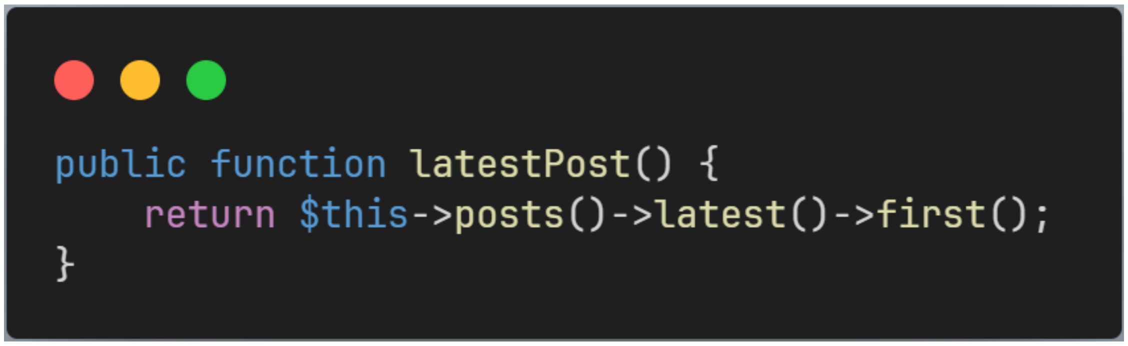
127.0.0.1:8000/categories/4/posts

```
// 20230703165219
// http://127.0.0.1:8000/categories/4/posts

[
  {
    "id": 4,
    "title": "Dummy Post 1",
    "created_at": "2023-07-03T16:37:24.000000Z",
    "updated_at": "2023-07-03T16:37:24.000000Z",
    "category_id": 4,
    "deleted_at": null
  },
  {
    "id": 11,
    "title": "Dummy Post 8",
    "created_at": "2023-07-03T16:37:24.000000Z",
    "updated_at": "2023-07-03T16:37:24.000000Z",
    "category_id": 4,
    "deleted_at": null
  },
  {
    "id": 14,
    "title": "Dummy Post 14",
    "created_at": "2023-07-03T16:37:24.000000Z",
    "updated_at": "2023-07-03T16:37:24.000000Z",
    "category_id": 4,
    "deleted_at": null
  }
]
```

Task 11:

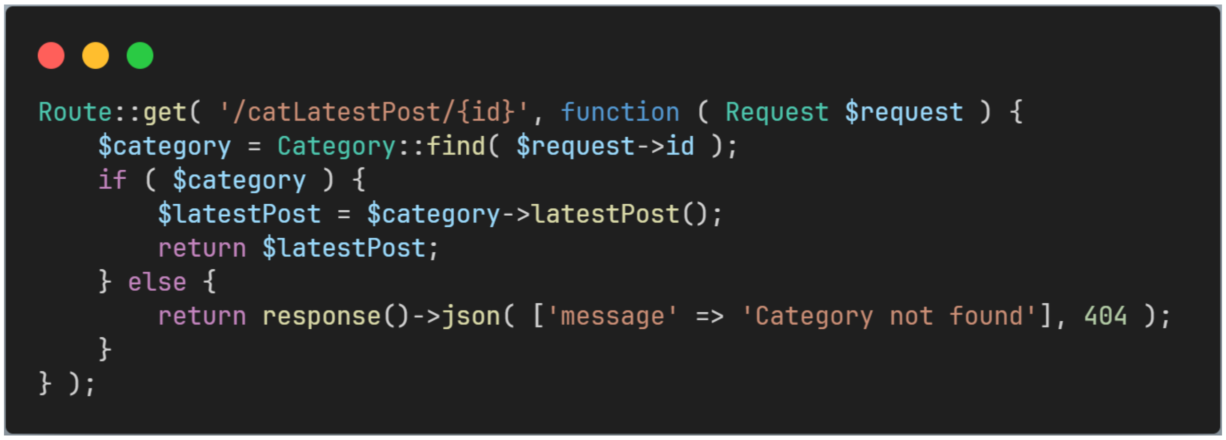
Implement a method in the "Category" model to get the latest post associated with the category. The method should return the post object.

Implement Method in Post Model:



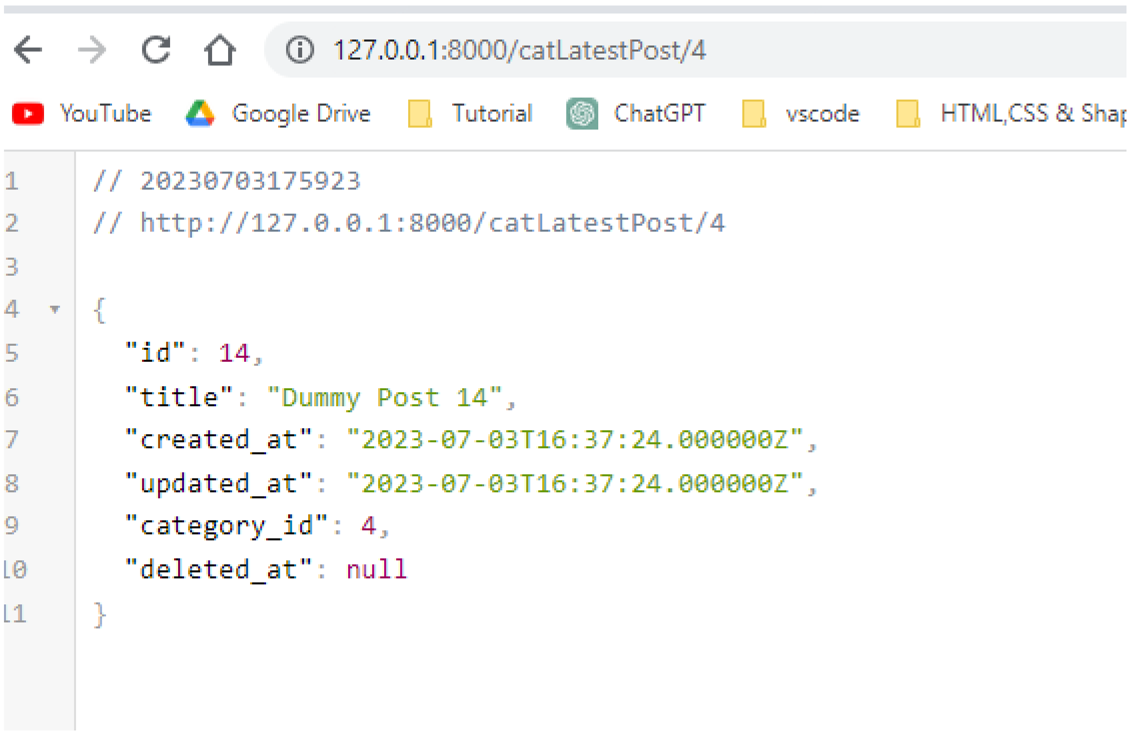
```php
public function latestPost() {
    return $this->posts()->latest()->first();
}
```

Route with method:



```php
Route::get( '/catLatestPost/{id}', function ( Request $request ) {
    $category = Category::find( $request->id );
    if ( $category ) {
        $latestPost = $category->latestPost();
        return $latestPost;
    } else {
        return response()->json( ['message' => 'Category not found'], 404 );
    }
} );
```

Output:

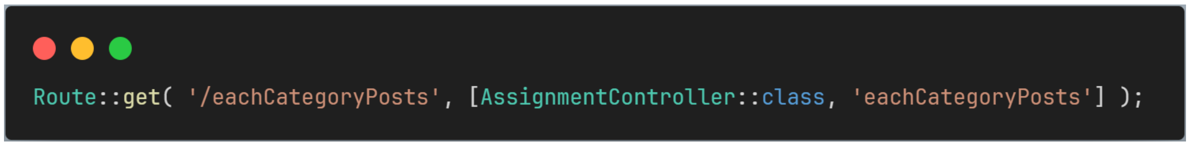


```
// 20230703175923
// http://127.0.0.1:8000/catLatestPost/4

{
  "id": 14,
  "title": "Dummy Post 14",
  "created_at": "2023-07-03T16:37:24.000000Z",
  "updated_at": "2023-07-03T16:37:24.000000Z",
  "category_id": 4,
  "deleted_at": null
}
```

Task 12:

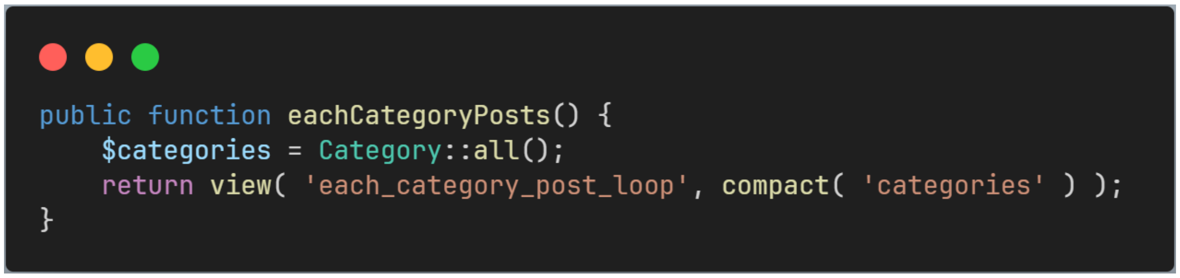
Write a Blade template to display the latest post for each category. Use a loop to iterate over the categories and display the post details.

Route:



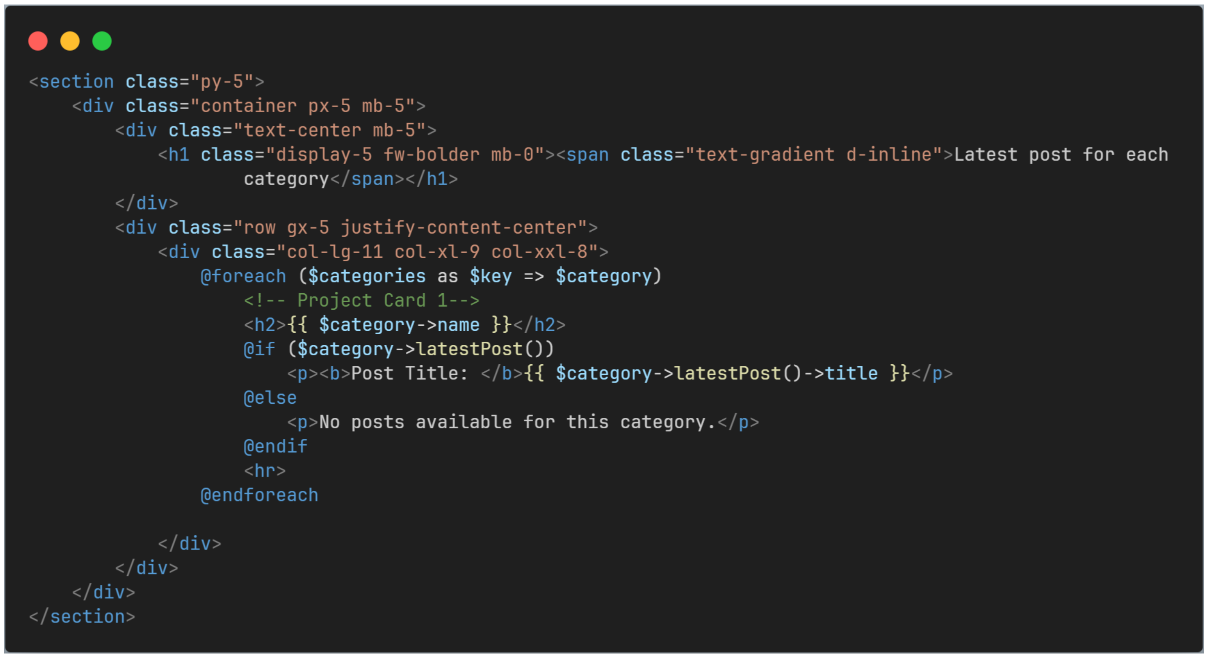
```php
Route::get( '/eachCategoryPosts', [AssignmentController::class, 'eachCategoryPosts'] );
```

Controller:



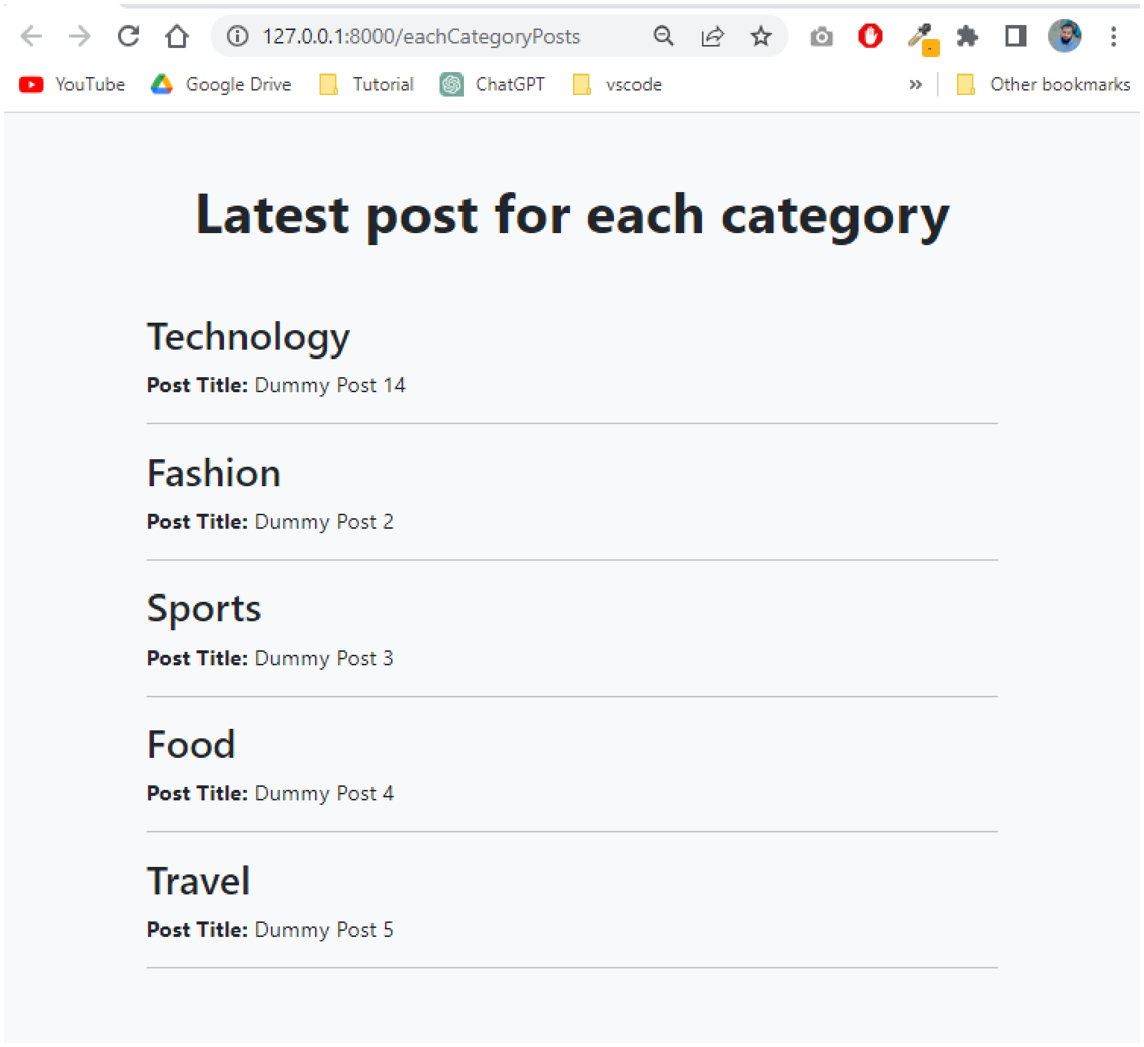
```php
public function eachCategoryPosts() {
    $categories = Category::all();
    return view( 'each_category_post_loop', compact( 'categories' ) );
}
```

Blade:



```blade
<section class="py-5">
    <div class="container px-5 mb-5">
        <div class="text-center mb-5">
            <h1 class="display-5 fw-bolder mb-0"><span class="text-gradient d-inline">Latest post for each
                    category</span></h1>
        </div>
        <div class="row gx-5 justify-content-center">
            <div class="col-lg-11 col-xl-9 col-xxl-8">
                @foreach ($categories as $key => $category)
                    <!-- Project Card 1-->
                    <h2>{{ $category->name }}</h2>
                    @if ($category->latestPost())
                        <p><b>Post Title: </b>{{ $category->latestPost()->title }}</p>
                    @else
                        <p>No posts available for this category.</p>
                    @endif
                    <hr>
                @endforeach

            </div>
        </div>
    </div>
</section>
```

Output:



Latest post for each category

Technology

Post Title: Dummy Post 14

Fashion

Post Title: Dummy Post 2

Sports

Post Title: Dummy Post 3

Food

Post Title: Dummy Post 4

Travel

Post Title: Dummy Post 5