

Module-16 Laravel Migration Concepts

Assignment-16:

Task 1: Create a new migration file to add a new table named 'categories' to the database. The table should have the following columns:

Step 1: Create a new migration file

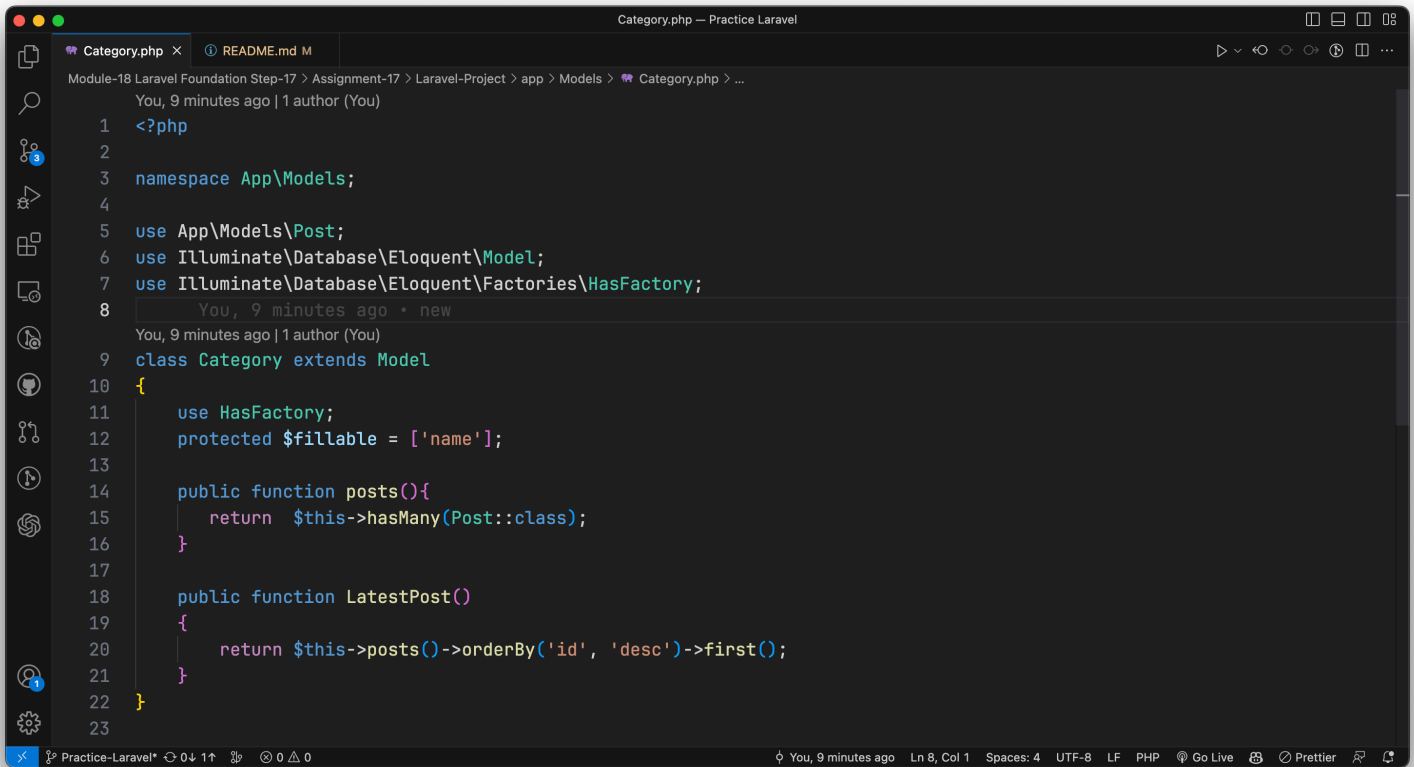
1. Open Terminal.
2. Run the following command:

```
php artisan make:migration create_Categories_table
```

```
*/  
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();  
    });  
}
```

Task 2: Create a new model named "Category" associated with the 'categories' table. Define the necessary properties and relationships.

Step 1: Defining Relationship:



```
Category.php — Practice Laravel
Module-18 Laravel Foundation Step-17 > Assignment-17 > Laravel-Project > app > Models > Category.php > ...
You, 9 minutes ago | 1 author (You)
1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Post;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8
9 class Category extends Model
10 {
11     use HasFactory;
12     protected $fillable = ['name'];
13
14     public function posts(){
15         return $this->hasMany(Post::class);
16     }
17
18     public function latestPost()
19     {
20         return $this->posts()->orderBy('id', 'desc')->first();
21     }
22 }
23
You, 9 minutes ago Ln 8, Col 1 Spaces: 4 UTF-8 LF PHP Go Live Prettier
```

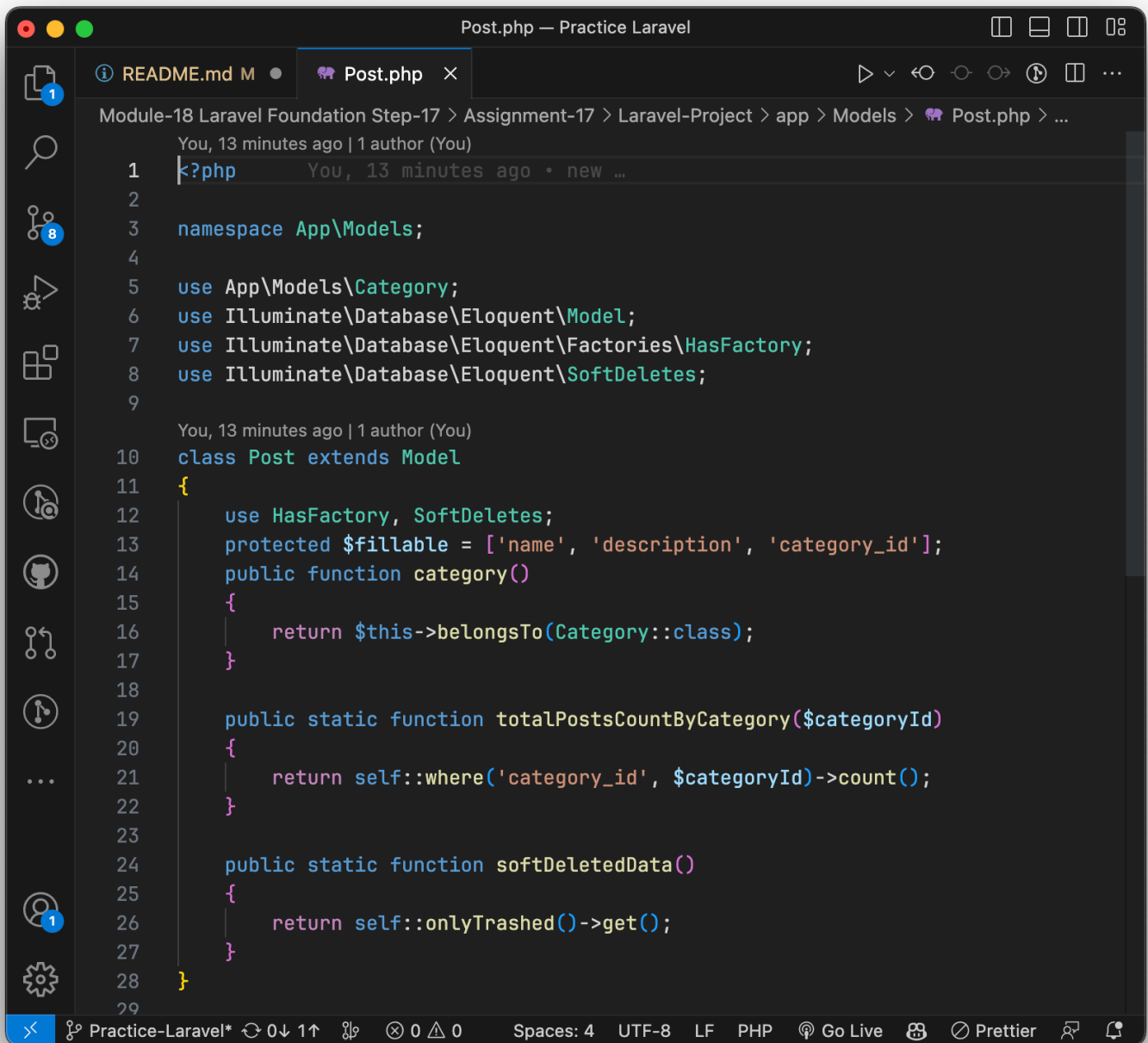
Task 3: Write a migration file to add a foreign key constraint to the "posts" table. The foreign key should reference the "categories" table on the "category_id" column.

Step 1: Posts table Migration file:

```
public function up(): void
{
    Schema::create('posts', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->text('description')->nullable();
        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
        $table->foreignId('category_id')->constrained('categories')
            ->cascadeOnDelete()
            ->cascadeOnUpdate();
    });
}
```

Task 4: Create a relationship between the "Post" and "Category" models. A post belongs to a category, and a category can have multiple posts.

Step 1: Post Model



The screenshot shows a code editor window titled "Post.php — Practice Laravel". The file path in the breadcrumb is "Module-18 Laravel Foundation Step-17 > Assignment-17 > Laravel-Project > app > Models > Post.php". The code defines a `Post` model that extends `Model`. It includes the following code:

```
1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Category;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8 use Illuminate\Database\Eloquent\SoftDeletes;
9
10 class Post extends Model
11 {
12     use HasFactory, SoftDeletes;
13     protected $fillable = ['name', 'description', 'category_id'];
14     public function category()
15     {
16         return $this->belongsTo(Category::class);
17     }
18
19     public static function totalPostsCountByCategory($categoryId)
20     {
21         return self::where('category_id', $categoryId)->count();
22     }
23
24     public static function softDeletedData()
25     {
26         return self::onlyTrashed()->get();
27     }
28 }
```

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.

Step 1: Controller method:

```
public function postData()
{
    $postdata = Post::with('category')->get();
    return $postdata;
}
```

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.

Step 1: Route

```
Route::controller(AdminController::class)->group(function () {  
  
    Route::get('/categoryPost/{category_id}', 'categoryPost');  
  
});
```

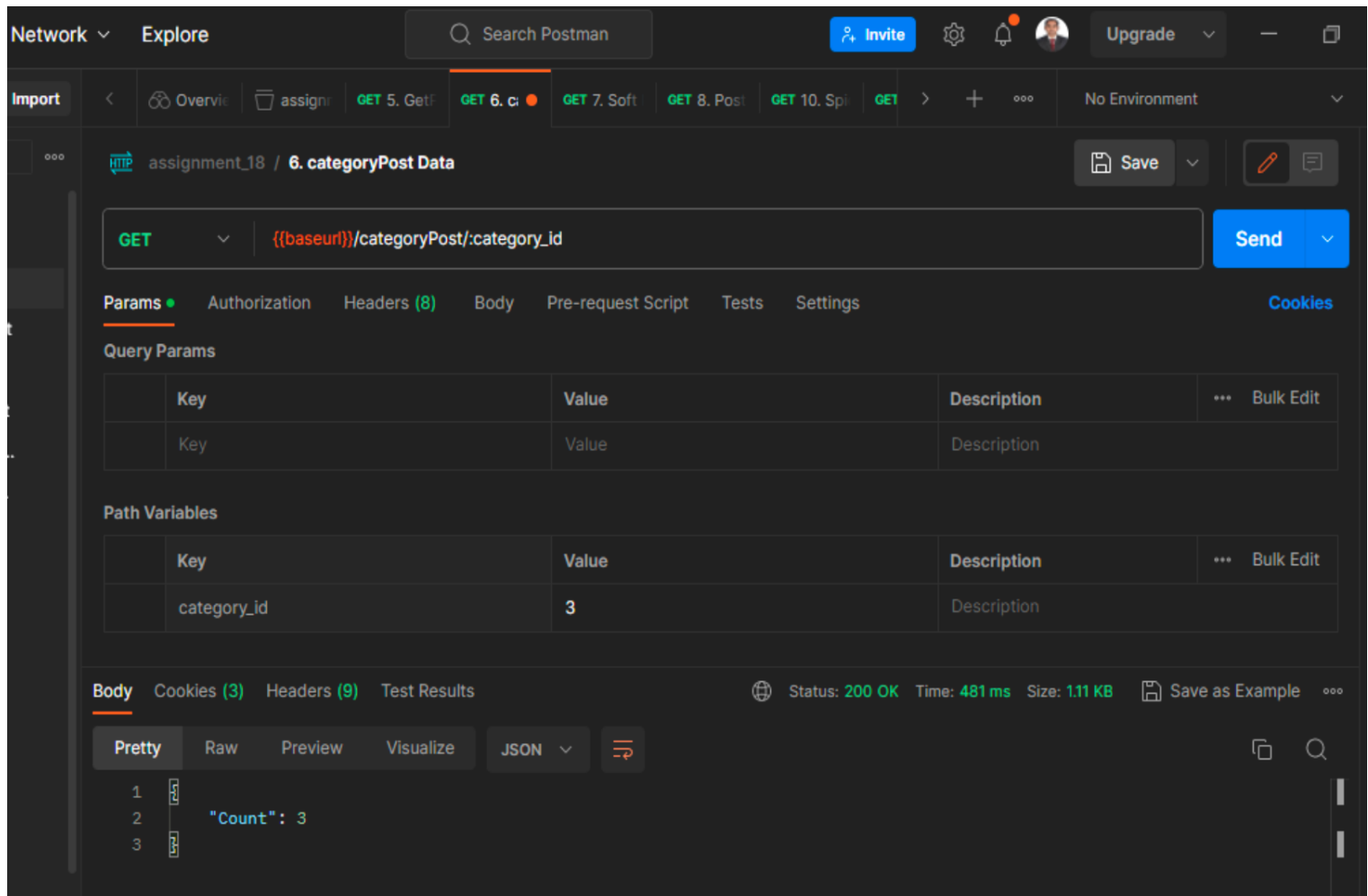
Step 2: Implemented Method in Post Model

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

Step 3: Controller method for testing In Postman

```
public function categoryPost($category_id)  
{  
    $postCount = Post::totalPostsCountByCategory($category_id);  
    return response()->json(['Count' => $postCount]);  
}
```

Step 4: Postman Output



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.

Step 1: Route

```
Route::controller(AdminController::class)->group(function () {
    Route::get('/softData', 'softData');
});
```

Step 2: Controller method

```
//Task 7
public function softDelete($id)
{
    $softDelete = Post::findOrFail($id)->delete();
    if ($softDelete) {
        return 'successfully soft deleted';
    } else {
        return 'failed to softdelete';
    }
}
```

Step 3: Postman Output:

```
//Task 7
public function softDelete($id)
{
    $softDelete = Post::findOrFail($id)->delete();
    if ($softDelete) {
        return 'successfully soft deleted';
    } else {
        return 'failed to softdelete';
    }
}
```

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.

Step 1: Route

```
Route::controller(AdminController::class)->group(function () {
    Route::get('/softData', 'softData');
});
```

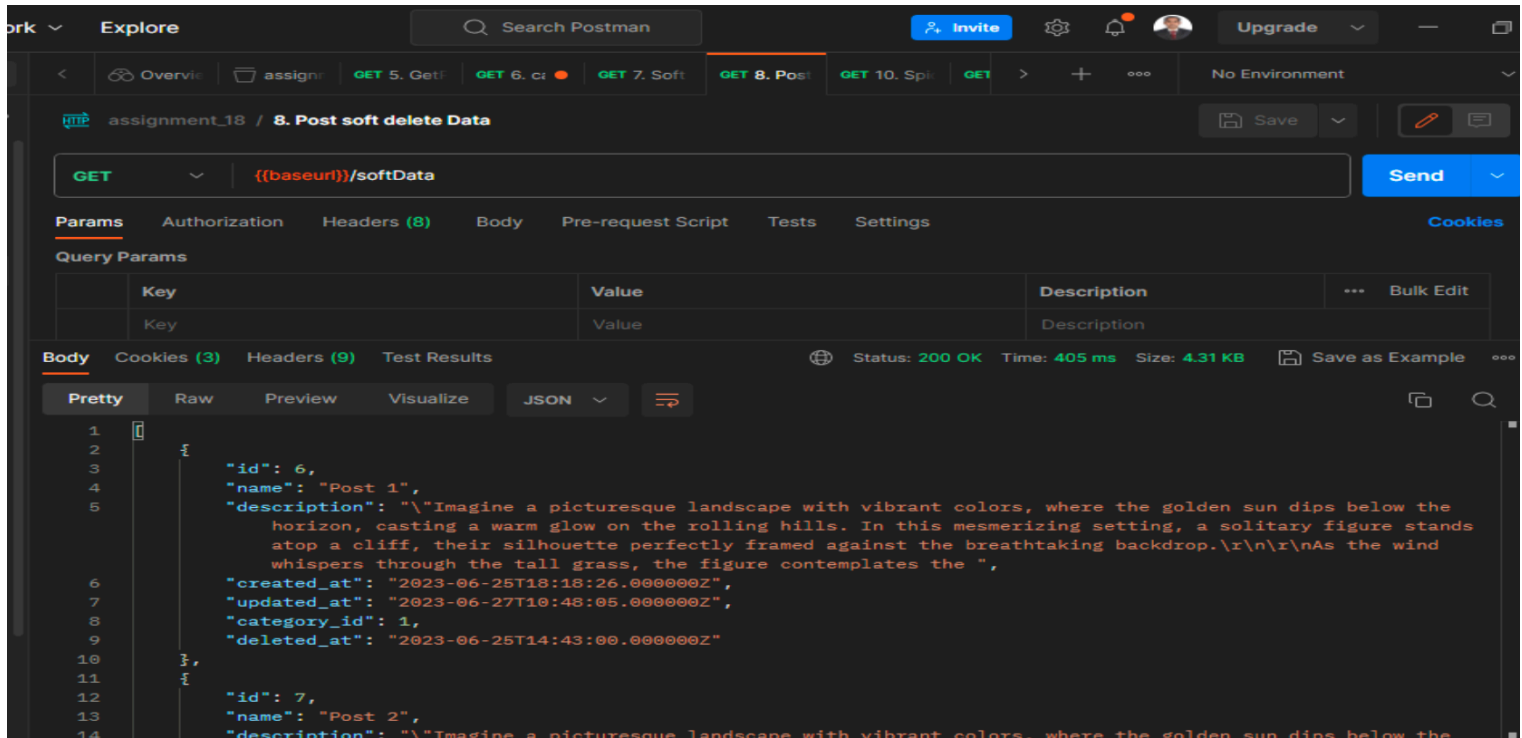
Step 2: Implemented Method in Post Model

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

Step 3: Controller method for testing In Postman

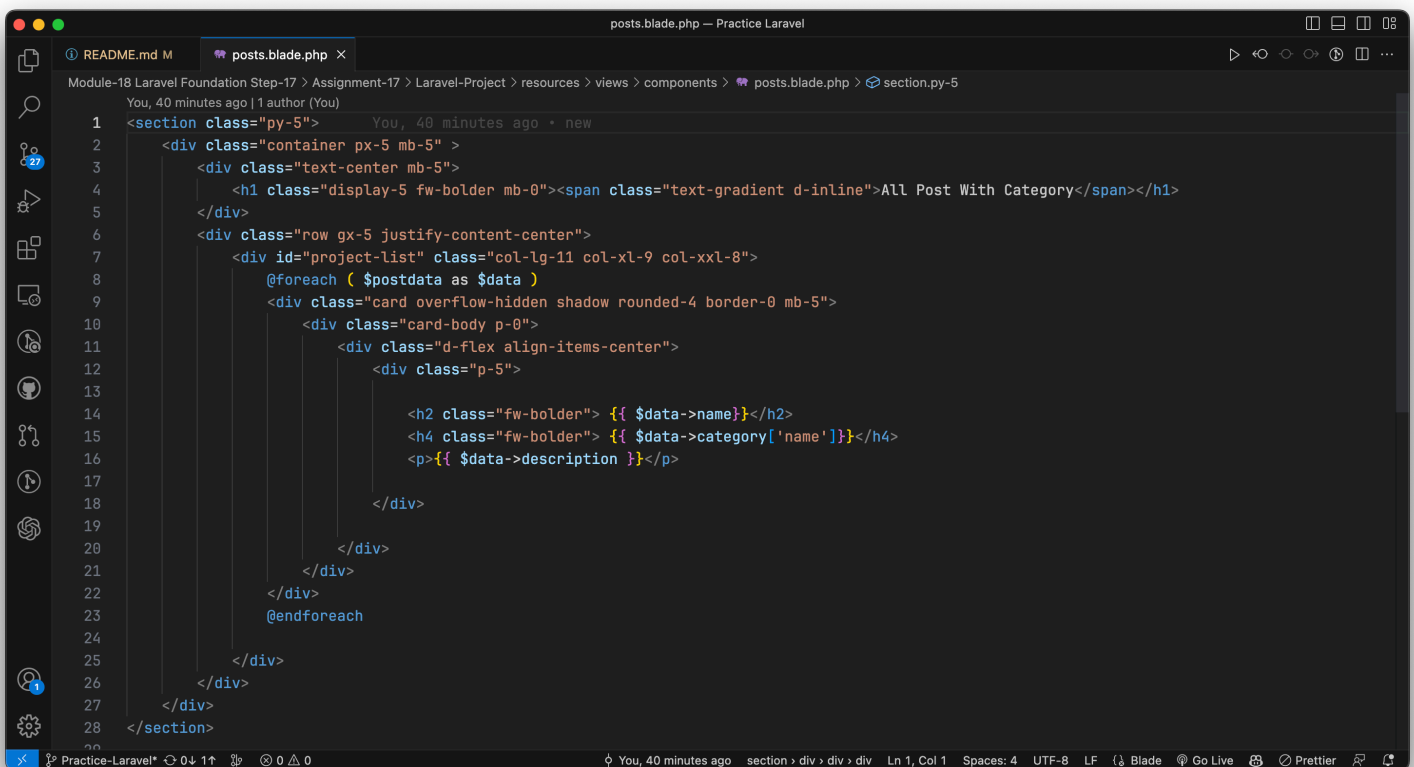
```
public function softData()
{
    $softData = Post::softDeletedData();
    return $softData;
}
```

Step 4: Postman output



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.

Step 1: Blade File



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.

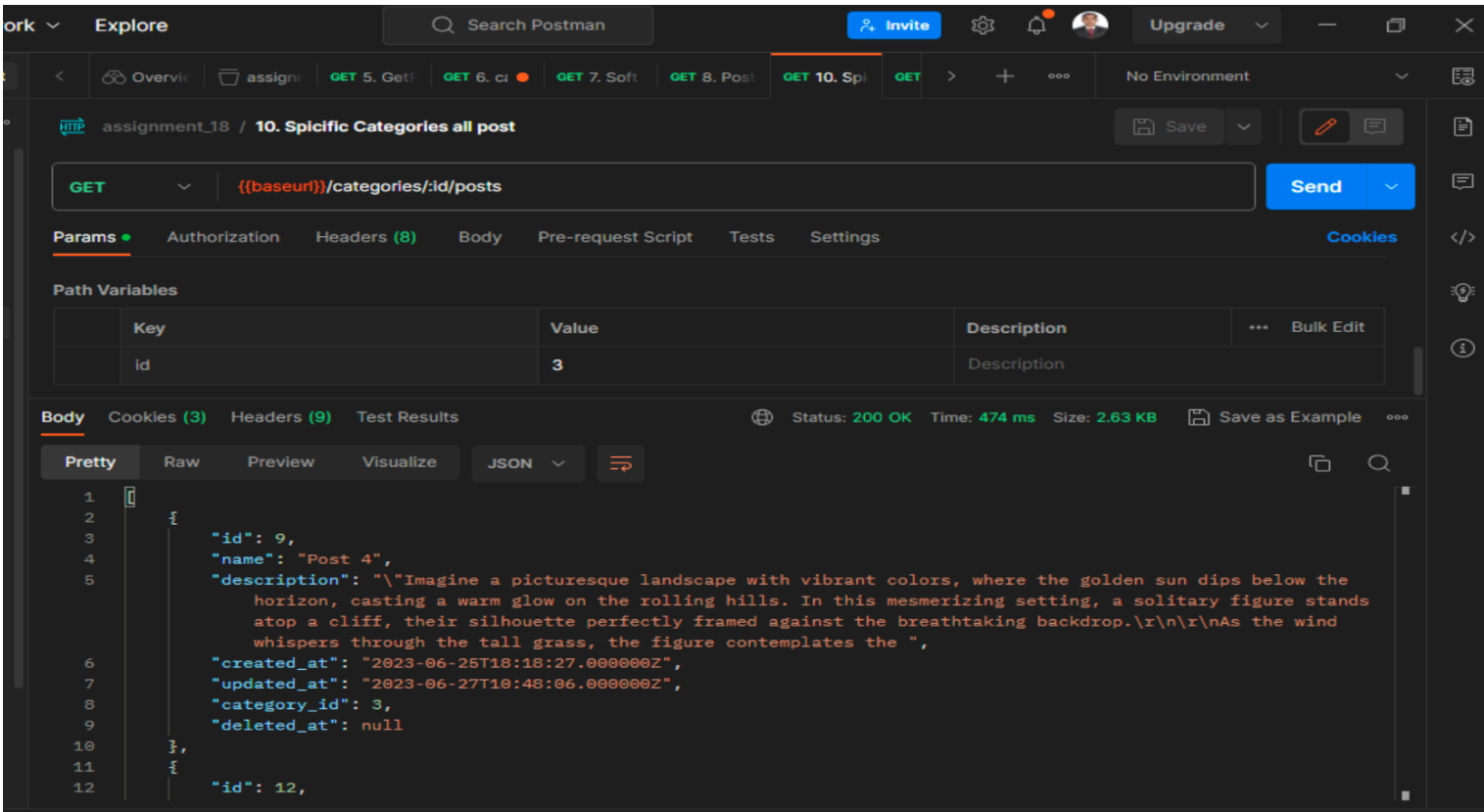
Step 1: Route

```
Route::controller(AdminController::class)->group(function () {  
  
    Route::get('/categories/{id}/posts', 'specificCatPost');  
  
});
```

Step 2: Controller method:

```
//Task 10  
public function specificCatPost($id)  
{  
    try {  
        $category = Category::findOrFail($id);  
        return $category->posts;  
    } catch (\Exception $e) {  
        return 'Category not found.';  
    }  
}
```

Step 3: Postman output:



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.

Step 1: Route

```
Route::controller(AdminController::class)->group(function () {  
  
    Route::get('/categories/{id}/latestpost', 'latestPost');  
  
});
```

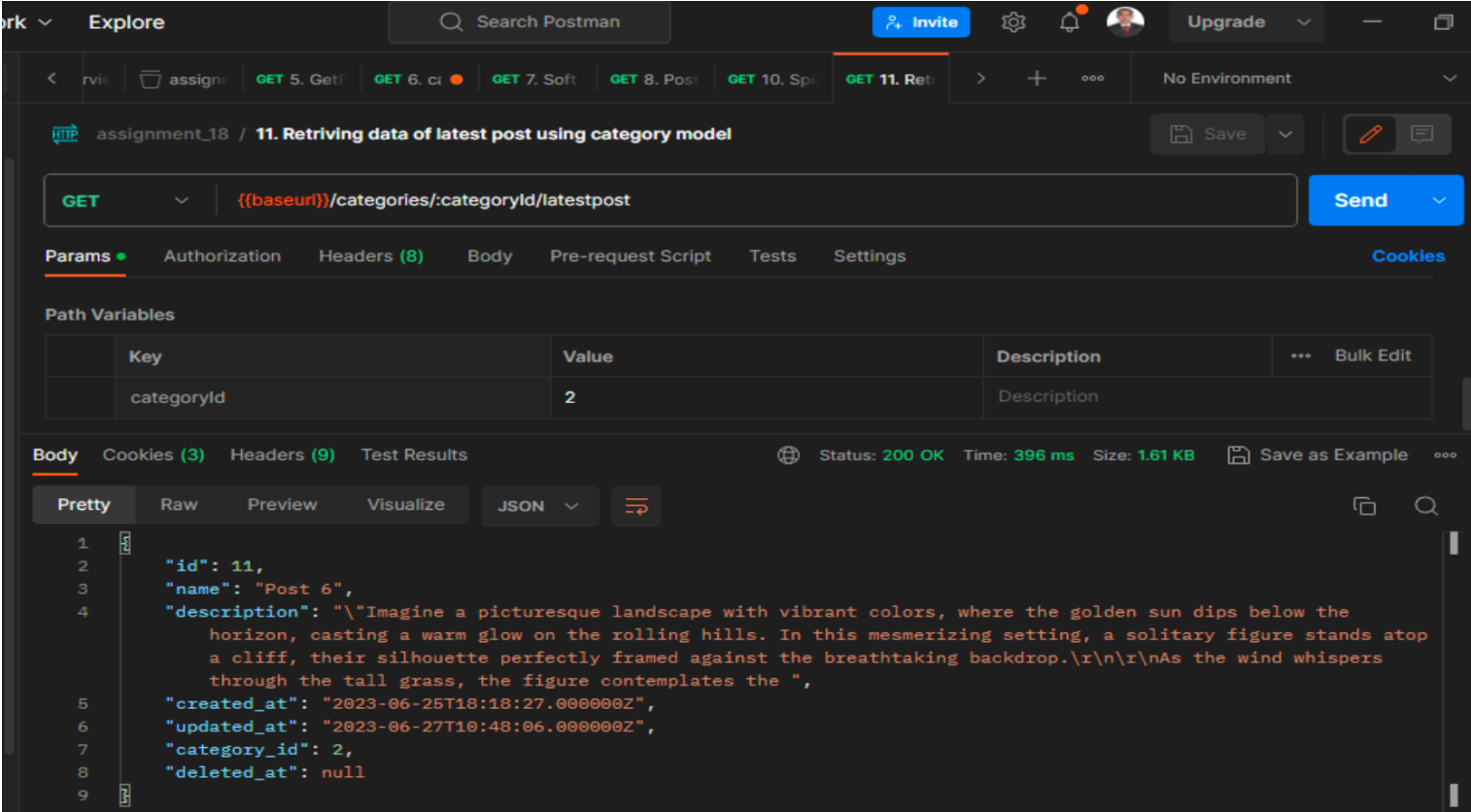
Step 2: Implemented Method in Category Model

```
public function LatestPost()  
{  
    return $this->posts()->orderBy('id', 'desc')->first();  
}
```

Step 3: Controller method for testing In Postman

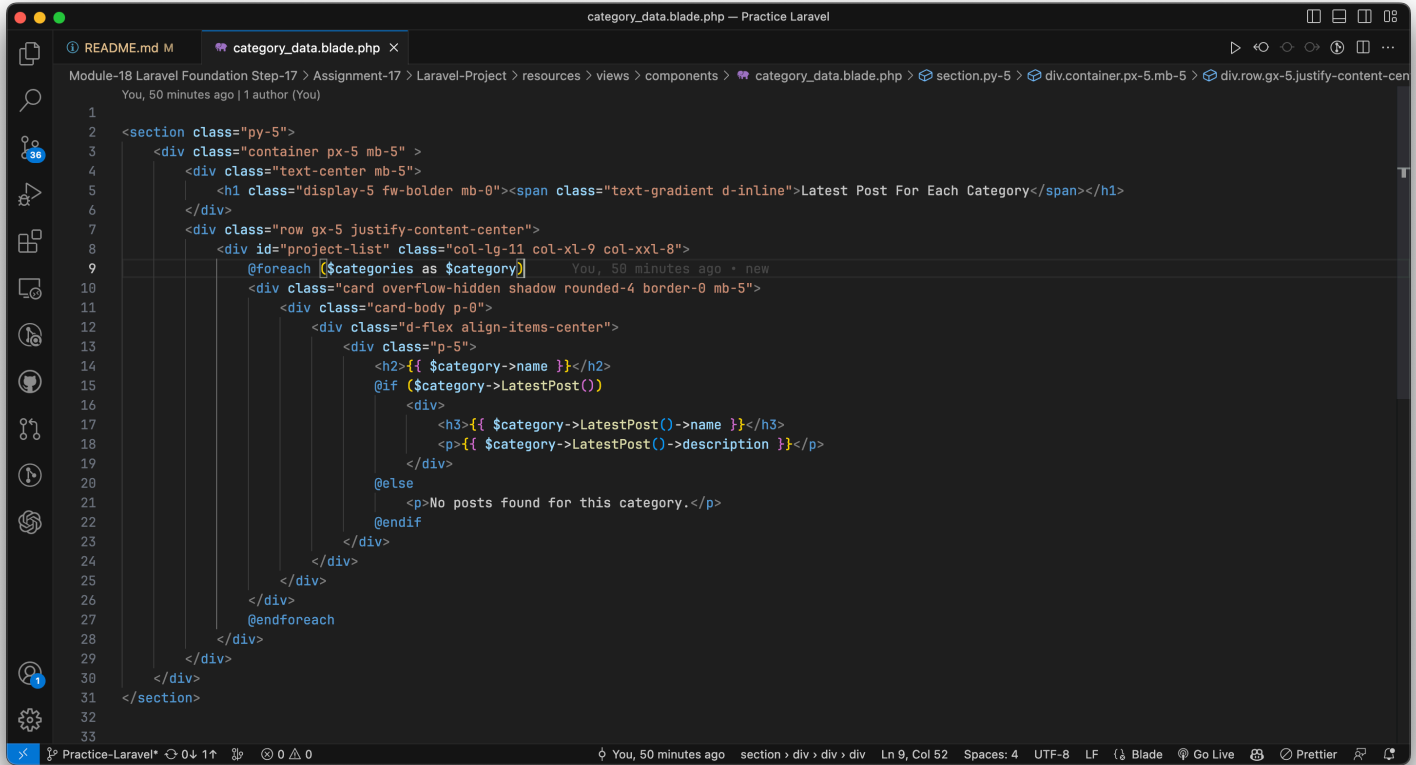
```
public function latestPost($id)  
{  
    $category = Category::findOrFail($id)->LatestPost();  
    return $category;  
}
```

Step 4: Postman output:



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.

Step 1: Blade



```
1 <section class="py-5">
2
3   <div class="container px-5 mb-5">
4     <div class="text-center mb-5">
5       <h1 class="display-5 fw-bolder mb-0"><span class="text-gradient d-inline">Latest Post For Each Category</span></h1>
6     </div>
7     <div class="row gx-5 justify-content-center">
8       <div id="project-list" class="col-lg-11 col-xl-9 col-xxl-8">
9         @foreach ($categories as $category)
10          <div class="card overflow-hidden shadow rounded-4 border-0 mb-5">
11            <div class="card-body p-0">
12              <div class="d-flex align-items-center">
13                <div class="p-5">
14                  <h2>{{ $category->name }}</h2>
15                  @if ($category->LatestPost())
16                    <div>
17                      <h3>{{ $category->LatestPost()->name }}</h3>
18                      <p>{{ $category->LatestPost()->description }}</p>
19                    </div>
20                  @else
21                    <p>No posts found for this category.</p>
22                  @endif
23                </div>
24              </div>
25            </div>
26          </div>
27        @endforeach
28      </div>
29    </div>
30  </div>
31 </section>
32
33
```

Author: Marajul Islam