Assignment Module 18

Name- Tahsin Shahriar

Task 1: Create a new migration file to add a new table named "categories" to the database.

you would find these in the `database/migrations` directory of the Laravel project.

Task 2: Create a new model named `Category` associated with the `categories` table.

The Category model is associated with the `categories` table. It has a relationship with the Post model, where a Category can have multiple Posts.

```
namespace App\Models;

use App\Models\Post;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

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

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

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

Task 3: Write a migration file to add a foreign key constraint to the "posts" table.

you would find these in the `database/migrations` directory of the Laravel project.

```
 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.

The `Post` model has a `belongsTo` relationship with the `Category` model.

```
namespace App\Models;

use App\Models\Category;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model
{
  use HasFactory, SoftDeletes;
  protected $fillable = ['name', 'description', 'category_id'];

  public function category()
  {
    return $this->belongsTo(Category::class);
  }
}
```

Task 5: Write a query using Eloquent ORM to retrieve all posts along with their associated categories.

The `postData` method in the `ActionController` retrieves all `posts` along with their associated categories.

```
public function postData()
{
  $postdata = Post::with('category')->get();
  return view('pages.home', compact('postdata'));
}
```

Task 6: Implement a method in the "Post" model to get the total number of posts belonging to a specific category.

The `categoryPost` method in the `ActionController` gets the total number of posts belonging to a specific category.

```
public function categoryPost($id)
{
  $postCount = Post::where('category_id', $id)->count();
  return $postCount;
}
```

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.

The `softDelete` method in the `ActionController` deletes a post by its ID.

```
PHP 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 `softData` method in the `ActionController` gets all posts that have been soft deleted.

```
public function softData()
{
  $softData = Post::onlyTrashed()->get();
  return $softData;
}
```

Task 9: Write a Blade template to display all posts and their associated categories.

you would find these in the "resources/views" directory.

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 `specificCatPost` method in the `ActionController` retrieves all posts belonging to a specific category.

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

Task 11: Implement a method in the “Category” model to get the latest post associated with the category.

The `LatestPost` method in the `Category` model gets the latest post associated with the category.

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

Task 12: Write a Blade template to display the latest post for each category.

you would find these in the “resources/views” directory.