```
php artisan make:model post -m
```

## Post Model

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model {
    protected $fillable = [
        'user_id', 'category_id', 'post_date', 'post_title', 'post_slug',
'post_details', 'featured_image', 'youtube_video_url', 'publication_status',
'is_featured', 'view_count', 'meta_title', 'meta_keywords', 'meta_description',
    ];

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

    public function user() {
        return $this->belongsTo(User::class);
    }

    public function tags() {
        return $this->belongsToMany(Tag::class);
    }

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

## Post Migration

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up() {
        Schema::create('posts', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id')->unsigned()->index();
            $table->integer('category_id')->unsigned()->index();
            $table->date('post_date');
            $table->string('post_title');
            $table->string('post_slug');
            $table->text('post_details');
            $table->string('featured_image')->nullable();
            $table->string('youtube_video_url')->nullable();
            $table->tinyInteger('publication_status')->default(0);
            $table->tinyInteger('is_featured')->default(0);
            $table->integer('view_count')->default(0);
            $table->string('meta_title')->nullable();;
            $table->string('meta_keywords')->nullable();
            $table->text('meta_description')->nullable();
            $table->foreign('category_id')->references('id')->on('categories')-
>onDelete('cascade');
            $table->foreign('user_id')->references('id')->on('users')-
>onDelete('cascade');
            $table->timestamps();
        });
    }

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

```
php artisan make:model tag -m
```

```
Tag Model
```

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Tag extends Model {
    protected $fillable = [
        'user_id', 'tag_name', 'tag_slug', 'publication_status', 'meta_title',
'meta_keywords', 'meta_description',
    ];

    public function user() {
        return $this->belongsTo(User::class);
    }

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

## Tag Migration

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTagsTable extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up() {
        Schema::create('tags', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id')->unsigned()->index();
            $table->string('tag_name');
            $table->string('tag_slug', 190)->unique();
            $table->string('meta_title')->nullable();;
            $table->string('meta_keywords')->nullable();
            $table->text('meta_description')->nullable();
            $table->tinyInteger('publication_status')->default(0);
            $table->timestamps();
            $table->foreign('user_id')->references('id')->on('users')-
>onDelete('cascade');
        });
    }

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

```
php artisan make:migration create_post_tag_table
```

## Post-Tag Migration

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostTagTable extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up() {
        Schema::create('post_tag', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('post_id')->unsigned();
            $table->foreign('post_id')->references('id')->on('posts')-
>onDelete('cascade');
            $table->integer('tag_id')->unsigned();
            $table->foreign('tag_id')->references('id')->on('tags')-
>onDelete('cascade');
        });
    }

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

```
php artisan make:controller TagController
```

## Tag Controller

```
<?php

namespace App\Http\Controllers;

use App\Tag;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Response;
use Illuminate\Support\Facades\Validator;

class TagController extends Controller {

    public function __construct() {
        $this->middleware('auth');
    }

    public function index() {
        return view('admin.tag.index');
    }

    public function get() {
        $tags = Tag::all();

        return datatables()->of($tags)
            ->editColumn('created_at', '{{ date("d F Y", strtotime($created_at)) }}')
            ->editColumn('updated_at', '{{ date("d F Y", strtotime($updated_at)) }}')
            ->addColumn('username', function ($tags) {
                return '<a class="user-view-button" role="button" tabindex="0" data-
id="' . $tags->user->id . '">' . $tags->user->name . '</a>';})
            ->addColumn('publication_status', function ($tags) {
                if ($tags->publication_status == 1) {
                    return '<a href="' . route('admin.unpublishedTagsRoute', $tags-
>id) . '" class="btn btn-success btn-xs btn-flat btn-block" data-toggle="tooltip"
data-original-title="Click to Unpublished"><i class="icon fa fa-arrow-
down"></i>Published</a>';
                }
                return '<a href="' . route('admin.publishedTagsRoute', $tags->id) . '"
class="btn btn-warning btn-xs btn-flat btn-block" data-toggle="tooltip" data-original-
title="Click to Published"><i class="icon fa fa-arrow-up"></i> Unpublished</a>';})
            ->addColumn('action', function ($tags) {
                return '<button class="btn btn-info btn-xs view-button" data-id="' .
$tags->id . '" data-toggle="tooltip" data-original-title="View"><i class="fa fa-
eye"></i></button> <button class="btn btn-primary btn-xs edit-button" data-id="' .
$tags->id . '" data-toggle="tooltip" data-original-title="Edit"><i class="fa fa-
edit"></i></button> <button class="btn btn-danger btn-xs delete-button" data-id="' .
$tags->id . '"data-toggle="tooltip" data-original-title="Delete"><i class="fa fa-
trash"></i></button>';})
            ->rawColumns(['username', 'publication_status', 'action'])
            ->setRowId('id')
            ->make(true);
```

```php
    }

    public function store(Request $request) {
        $validator = $validator = Validator::make($request->all(), [
            'tag_name' => 'required|max:250',
            'tag_slug' => 'required|alpha_dash|min:5|max:150|unique:tags',
            'publication_status' => 'required',
            'meta_title' => 'required|max:250',
            'meta_keywords' => 'required|max:250',
            'meta_description' => 'required|max:400',
        ], [
            'tag_name.required' => 'Tag name is required.',
        ]);

        if ($validator->passes()) {
            $tag = Tag::create([
                'user_id' => Auth::user()->id,
                'tag_name' => $request->input('tag_name'),
                'tag_slug' => $request->input('tag_slug'),
                'publication_status' => $request->input('publication_status'),
                'meta_title' => $request->input('meta_title'),
                'meta_keywords' => $request->input('meta_keywords'),
                'meta_description' => $request->input('meta_description'),
            ]);
            $tag_id = $tag->id;

            if (!empty($tag_id)) {
                $request->session()->flash('message', 'Tag add successfully.');
            } else {
                $request->session()->flash('exception', 'Operation failed !');
            }

            return Response::json(['success' => '1']);
        }
        return Response::json(['errors' => $validator->errors()]);
    }

    public function show($id) {
        $tag = Tag::where('id', $id)
            ->first();
        return json_encode($tag);
    }

    public function update(Request $request, $id) {
        $tag = Tag::find($id);

        if ($tag->tag_slug == $request->tag_slug) {
            $tag_slug = "required|alpha_dash|min:5|max:150";
        } else {
            $tag_slug = "required|alpha_dash|min:5|max:150|unique:tags";
        }

        $validator = $validator = Validator::make($request->all(), [
            'tag_name' => 'required|max:250',
            'tag_slug' => $tag_slug,
            'publication_status' => 'required',
            'meta_title' => 'required|max:250',
            'meta_keywords' => 'required|max:250',
            'meta_description' => 'required|max:400',
        ], [
            'tag_name.required' => 'Tag name is required.',
        ]);

        if ($validator->passes()) {
            $tag->tag_name = $request->get('tag_name');
            $tag->tag_slug = $request->get('tag_slug');
            $tag->publication_status = $request->get('publication_status');
            $tag->meta_title = $request->get('meta_title');
            $tag->meta_keywords = $request->get('meta_keywords');
            $tag->meta_description = $request->get('meta_description');
            $affected_row = $tag->save();

            if (!empty($affected_row)) {
                $request->session()->flash('message', 'Tag update successfully.');
            } else {
                $request->session()->flash('exception', 'Operation failed !');
            }
            return Response::json(['success' => '1']);
        }
        return Response::json(['errors' => $validator->errors()]);
    }

    public function published($id) {
        $affected_row = Tag::where('id', $id)
            ->update(['publication_status' => 1]);

        if (!empty($affected_row)) {
            return redirect()->back()->with('message', 'Published successfully.');
        }
        return redirect()->back()->with('exception', 'Operation failed !');
    }

    public function unpublished($id) {
        $affected_row = Tag::where('id', $id)
            ->update(['publication_status' => 0]);

        if (!empty($affected_row)) {
            return redirect()->back()->with('message', 'Unpublished successfully.');
        }
        return redirect()->back()->with('exception', 'Operation failed !');
    }
}
```

```
php artisan make:controller PostController
```

## Post Controller

```
<?php

namespace App\Http\Controllers;

use App\Category;
use App\Post;
use App\Tag;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Str;
use Image;
use Purifier;

class PostController extends Controller {

    public function __construct() {
        $this->middleware('auth');
    }

    public function index() {
        return view('admin.post.index');
    }

    public function get() {
        $posts = Post::all();

        return datatables()->of($posts)
            ->editColumn('created_at', '{{ date("d F Y", strtotime($created_at)) }}')
            ->editColumn('updated_at', '{{ date("d F Y", strtotime($updated_at)) }}')
            ->editColumn('post_title', function($posts){
                return Str::limit($posts->post_title, 30);
            })
            ->addColumn('username', function ($posts) {
                return '<a class="user-view-button" role="button" tabindex="0" data-
id="' . $posts->user->id . '">' . $posts->user->name . '</a>';})
            ->addColumn('publication_status', function ($posts) {
                if ($posts->publication_status == 1) {
                    return '<a href="' . route('admin.unpublishedPostsRoute', $posts-
>id) . '" class="btn btn-success btn-xs btn-flat btn-block" data-toggle="tooltip"
data-original-title="Click to Unpublished"><i class="icon fa fa-arrow-
down"></i>Published</a>';
                }
                return '<a href="' . route('admin.publishedPostsRoute', $posts->id) .
'" class="btn btn-warning btn-xs btn-flat btn-block" data-toggle="tooltip" data-
original-title="Click to Published"><i class="icon fa fa-arrow-up"></i>
Unpublished</a>';})
            ->addColumn('action', function ($posts) {
                $is_delete = '';
                if(!is_null(is_demo_admin())){
                    $is_delete = 'disabled';
                }
                return '<button class="btn btn-info btn-xs view-button" data-id="' .
$posts->id . '" data-toggle="tooltip" data-original-title="View"><i class="fa fa-
eye"></i></button> <a href="' . route('admin.posts.edit', $posts->id) . '" class="btn
btn-primary btn-xs" data-toggle="tooltip" data-original-title="Edit"><i class="fa fa-
edit"></i></a> <button class="btn btn-danger btn-xs delete-button" data-id="' .
$posts->id . '"data-toggle="tooltip" data-original-title="Delete"' . $is_delete .'><i
class="fa fa-trash"></i></button>';})
            ->addColumn('featured_image', function ($posts) {
                return '<img src="' . get_featured_image_thumbnail_url($posts-
>featured_image) . '" width="60" class="img img-thumbnail img-responsive">';})
            ->rawColumns(['username', 'publication_status', 'action',
'featured_image'])
            ->setRowId('id')
            ->make(true);
    }

    public function create() {
        $categories = Category::where('publication_status', 1)->get(['id',
'category_name']);
        $tags = Tag::where('publication_status', 1)->get(['id', 'tag_name']);
        return view('admin.post.create', compact('categories', 'tags'));
    }

    public function store(Request $request) {
        $url = '/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/';
        $post = request()->validate([
            'category_id' => 'required',
            'post_date' => 'required|date',
            'post_title' => 'required|string|max:255',
            'post_slug' => 'required|alpha_dash|min:5|max:150|unique:posts',
            'post_details' => 'required|string',
            'featured_image' =>
'required|image|mimes:jpeg,png,jpg,gif,svg|max:50240|dimensions:max_width=4032,max_hei
ght=2268',
            'youtube_video_url' => 'nullable|max:255|regex:' . $url,
            'youtube_video_url' => 'nullable|max:255',
            'publication_status' => 'required',
            'is_featured' => 'required',
            'meta_title' => 'required|max:250',
            'meta_keywords' => 'required|max:250',
            'meta_description' => 'required|max:400',
        ], [
            'featured_image.dimensions' => 'Max dimensions 2268x4032',
            'category_id.required' => 'The category field is required.',
        ]);

        $url = $request->input('youtube_video_url');
        $youtube_video_url = str_replace('youtu.be', 'youtube.com/embed', $url);

        $post = Post::create([
            'user_id' => Auth::user()->id,
            'post_title' => $request->input('post_title'),
            'post_slug' => $request->input('post_slug'),
            'category_id' => $request->input('category_id'),
            'post_date' => $request->input('post_date'),
            'publication_status' => $request->input('publication_status'),
            'is_featured' => $request->input('is_featured'),
            'youtube_video_url' => $youtube_video_url,
            'post_details' => Purifier::clean($request->input('post_details')),
            'meta_title' => $request->input('meta_title'),
            'meta_keywords' => $request->input('meta_keywords'),
            'meta_description' => $request->input('meta_description'),
        ]);

        if (isset($request->post_tags)) {
            $post->tags()->sync($request->post_tags, false);
        } else {
            $post->tags()->sync(array());
        }

        if ($request->hasFile('featured_image')) {
            $image = $request->file('featured_image');
            $file_name = $this->featured_image($post->id, $image);
            $file_name = $this->featured_image_thumbnail($post->id, $image);
            Post::find($post->id)->update(['featured_image' => $file_name]);
        }

        if (!empty($post->id)) {
            return redirect()->back()->with('message', 'Post add successfully.');
        } else {
            return redirect()->back()->with('exception', 'Operation failed !');
        }
    }


    public function update(Request $request, $id) {
        $post = Post::find($id);

        if ($post->post_slug == $request->post_slug) {
            $post_slug = "required|alpha_dash|min:5|max:150";
        } else {
            $post_slug = "required|alpha_dash|min:5|max:150|unique:posts";
        }

        $url = '/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/';
        request()->validate([
            'category_id' => 'required',
            'post_date' => 'required|date',
            'post_title' => 'required|string|max:255',
            'post_slug' => $post_slug,
            'post_details' => 'required|string',
```

```php
            'featured_image' =>
'nullable|image|mimes:jpeg,png,jpg,gif,svg|max:10240|dimensions:max_width=10000,max_he
ight=60000',
            //'youtube_video_url' => 'nullable|max:255|regex:' . $url,
            'youtube_video_url' => 'nullable|max:255',
            'publication_status' => 'required',
            'is_featured' => 'required',
            'meta_title' => 'required|max:250',
            'meta_keywords' => 'required|max:250',
            'meta_description' => 'required|max:400',
        ], [
            'category_id.required' => 'The category field is required.',
        ]);

        $post->post_title = $request->input('post_title');
        $post->post_slug = $request->input('post_slug');
        $post->category_id = $request->input('category_id');
        $post->post_date = $request->input('post_date');
        $post->publication_status = $request->input('publication_status');
        $post->is_featured = $request->input('is_featured');
        $post->youtube_video_url = $request->input('youtube_video_url');
        $post->post_details = Purifier::clean($request->input('post_details'));
        $post->meta_title = $request->input('meta_title');
        $post->meta_keywords = $request->input('meta_keywords');
        $post->meta_description = $request->input('meta_description');

        if ($request->hasFile('featured_image')) {
            $image = $request->file('featured_image');
            $file_name = $this->featured_image($id, $image);
            $this->featured_image_thumbnail($id, $image);
            $post->featured_image = $file_name;
        }

        $affected_row = $post->save();

        if (isset($request->post_tags)) {
            $post->tags()->sync($request->post_tags);
        } else {
            $post->tags()->sync(array());
        }

        if (!empty($affected_row)) {
            return redirect()->back()->with('message', 'Post update successfully.');
        } else {
            return redirect()->back()->with('exception', 'Operation failed !');
        }
    }
}
```