UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

ASSIGNMENT: Blog Management System

January 2021 Trimester

ID	Student Name	Email
1703648	Tan Ying Yao	yyaoutar@1utar.my
1701231	Lim Thien Chung	thienchung@1utar.my
1704368	Wong Yin Khye	stevenwong0121@1utar.my

A. Title and description of your web application as well as a statement or table explaining or describing the authorization logic.

Blog Management System

This web application is a blog management system that allows for creation of blog posts and categories. There are three roles in this system which are Admin, Users and Writers. For non-registered users, they are only able to see the 'Home' page which contains blog posts, popular categories and archives. After registering or logging in, the user is prompted to a dashboard that allows for CRUD function.

All three roles are able to access the backend dashboard, however each role has their own specific functions available:

i) Users can only search and view blog posts without any permission to edit or delete them

ii) Writers can search, view and even add new blog posts along with editing or deleting their own blogs

iii) Admins have full functions of users and writers and the ability to create new categories and publish or draft blog posts.

Gates is the main authorization of this application. Each time a user is trying to access CRUD functions, gates will be called to verify the current role of the user using 'isAdmin', 'isUser' or 'isWriter'. The gates will check the database for the role of the current user. Each role has their own function as stated above.

Admin Login Credentials:

E-mail: bruh@gmail.com

Password: 123

Database Seeding:

The project contains dummy data that can be seeded if needed by running:

php artisan database:seed

B. Code Listings

i. Code listing of all database migrations

2021_03_26_164545_create_blogs_table.php

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateBlogsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('blogs', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('user_id');
            $table->foreign('user_id')->references('id')->on('users');
            $table->string('title');
            $table->text('description');
            $table->text('body');
            $table->string('image')->nullable();
            $table->boolean('status')->default(false);
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
```

```
    public function down()
    {
        Schema::dropIfExists('blogs');
    }
}
```

2021_03_26_164854_add_role_to_user.php

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class AddRoleToUser extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->enum('role',  ['user','writer', 'admin'])->default('user');
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('user', function (Blueprint $table) {
            $table->dropColumn('role');
        });
```

```
    }
}
```

2021_03_27_072911_add_subscription_column_in_users.php

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class AddSubscriptionColumnInUsers extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->boolean('subscription')->default(false);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('subscription');
        });
    }
}
```

2021_03_27_125032_create_categories_table.php

```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->boolean('status')->default(false);
            $table->timestamps();
        });
    }

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

2021_03_27_125220_add_new_column_category_id_in_blogs.php

```php
<?php
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class AddNewColumnCategoryIdInBlogs extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('blogs', function (Blueprint $table) {
            $table->unsignedBigInteger('category_id');
            $table->foreign('category_id')->references('id')->on('categories');
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('blogs', function (Blueprint $table) {
            $table->dropForeign(['category_id']);
            $table->dropColumn(['category_id']);
        });
    }
}
```

ii. Code listing of your route file

api.php

```
<?php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
```

web.php

```
<?php
use Illuminate\Support\Facades\Route;
Route::get('/', 'FrontendController@index')->name('root');
Route::get('/blog/{blog}/view', 'FrontendController@show')->name('blog.view');
Auth::routes();
Route::group(['middleware' => ['auth']], function() {
    Route::get('/home', 'HomeController@index')->name('home');
    Route::resource('/blog', 'BlogController');
    Route::resource('/category', 'CategoryController');
});
```

iii. Code listing of all controller classes

HomeController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }
}
```

FrontendController.php

```
<?php
namespace App\Http\Controllers;
use App\Blog;
use App\Service\BlogService;
```

```
use App\Service\CategoryService;
use Illuminate\Contracts\View\Factory;
use Illuminate\Http\Request;
use Illuminate\View\View;

class FrontendController extends Controller
{

    /**
     * @var BlogService
     */
    private $service;
    private $view;
    private $redirectUrl;
    private $title;
    /**
     * @var CategoryService
     */
    private $categoryService;

    /**
     * BlogController constructor.
     * @param BlogService $service
     * @param CategoryService $categoryService
     */
    public function __construct(BlogService $service, CategoryService $categoryService)
    {
        $this->title = "Blog";
        $this->view = "frontend.blog.";
        $this->redirectUrl = "/";
        $this->service = $service;
        $this->categoryService = $categoryService;
    }
```

```
    /**
     * @param Request $request
     * @return Factory|View
     */
    public function index(Request $request)
    {
        $title = $this->title;
        $emptyBlog = new Blog();
        $blogs = $this->service->getAll(['title', 'body', 'description', 'user.name'],
$request)->orderBy('created_at','desc')->paginate(10);
        $archives = $this->service->getAllActive()->toArray();
        return view($this->view . "index", compact('blogs', 'title', 'emptyBlog', 'archives'));
    }

    /**
     * @param Blog $blog
     * @return Factory|View
     */
    public function show(Blog $blog)
    {
        $title = $blog->title;
        $archives = $this->service->getAllActive()->toArray();
        return view($this->view . "show", compact('title', 'blog', 'archives'));
    }
}
```

CategoryController.php

```
<?php
namespace App\Http\Controllers;
use App\Category;
use App\Http\Requests\CategoryRequestValidation;
use App\Service\CategoryService;
use App\Service\MessageService;
use Exception;
use Illuminate\Contracts\View\Factory;
```

```
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Routing\Redirector;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Gate;
use Illuminate\View\View;
class CategoryController extends Controller
{
    /**
     * @var CategoryService
     */
    private $service;
    private $view;
    private $redirectUrl;
    private $title;

    /**
     * CategoryController constructor.
     * @param CategoryService $service
     */
    public function __construct(CategoryService $service)
    {
        $this->title = "Category";
        $this->view = "backend.category.";
        $this->redirectUrl = "/category";
        $this->service = $service;
    }

    /**
     * @param Request $request
     * @return Factory|View
     */
    public function index(Request $request)
```

```
    {
        $emptyCategory = new Category();
        if(!Gate::allows('viewAny', $emptyCategory)) {
            return abort(403);
        }
        $title = $this->title;
        $categories = $this->service->getAll(['title'],
$request->keywords)->orderBy('created_at','desc')->paginate(10);
        return view($this->view . "index", compact('categories', 'title', 'emptyCategory'));
    }

    /**
     * @return Factory|View
     */
    public function create()
    {
        $title = "Add " . $this->title;
        return view($this->view . "create", compact('title'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param Request $request
     * @return Response
     */
    public function store(CategoryRequestValidation $request)
    {
        $data = [];
        try {
            DB::beginTransaction();
            $data['status'] = $request->status ? $request->status : false;
            $data['title'] = $request->title;
            $this->service->store($data);
```

```
        DB::commit();
        return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::SUCCESS]);
    } catch (Exception $exception) {
        DB::rollBack();
        return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::ADD_FAIL]);
    }
}

/**
 * Display the specified resource.
 *
 * @param Category $category
 * @return Response
 */
public function show(Category $category)
{
    $title = $category->title;
    return view($this->view . "show", compact('title', 'category'));
}

/**
 * @param Category $category
 * @return Factory|View
 */
public function edit(Category $category)
{
    $title = "Edit " . $this->title;
    return view($this->view . "edit", compact('category', 'title'));
}

/**
 * @param CategoryRequestValidation $request
 * @param Category $category
 * @return RedirectResponse|Redirector
```

```
     */
    public function update(CategoryRequestValidation $request, Category $category)
    {
        try {
            DB::beginTransaction();
            $category->update($request->all());
            DB::commit();
            return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::UPDATE]);
        } catch (Exception $exception) {
            DB::rollBack();
            return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::UPDATE_FAIL]);
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Category $category
     * @return Response
     */
    public function destroy(Category $category)
    {
        try {
            DB::beginTransaction();
            $category->delete();
            DB::commit();
            return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::DELETE]);
        } catch (Exception $exception) {
            DB::rollBack();
            return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::DELETE_FAIL]);
        }
    }
}
```

BlogController.php

```
<?php
namespace App\Http\Controllers;
use App\Blog;
use App\Events\BlogAdded;
use App\Events\NotifySubscriber;
use App\Http\Requests\BlogRequestValidation;
use App\Service\BlogService;
use App\Service\CategoryService;
use App\Service\MessageService;
use Exception;
use Illuminate\Contracts\View\Factory;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Routing\Redirector;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
use Illuminate\View\View;
class BlogController extends Controller
{
    /**
     * @var BlogService
     */
    private $service;
    private $view;
    private $redirectUrl;
    private $title;
    /**
     * @var CategoryService
```

```
 */
private $categoryService;
/**
 * BlogController constructor.
 * @param BlogService $service
 * @param CategoryService $categoryService
 */
public function __construct(BlogService $service, CategoryService $categoryService)
{
    $this->title = "Blog";
    $this->view = "backend.blog.";
    $this->redirectUrl = "/blog";
    $this->service = $service;
    $this->categoryService = $categoryService;
}
/**
 * @param Request $request
 * @return Factory|View
 */
public function index(Request $request)
{
    $title = $this->title;
    $emptyBlog = new Blog();
    $blogs = $this->service->getAll(['title', 'body', 'description', 'user.name'], $request->keywords);
    $user = Auth::user();
    if ($user->role != "admin") {
        $blogs = $blogs->where('user_id', $user->id);
    }
    $blogs = $blogs->orderBy('created_at','desc')->paginate(10);
    return view($this->view . "index", compact('blogs', 'title', 'emptyBlog'));
}
/**
 * @return Factory|View
 */
```

```
    public function create()
    {
        $title = "Add " . $this->title;
        $category = $this->categoryService->getActiveCategoryArray("status", "=", true);
        return view($this->view . "create", compact('title', 'category'));
    }
    /**
     * Store a newly created resource in storage.
     *
     * @param Request $request
     * @return Response
     */
    public function store(BlogRequestValidation $request)
    {
        $data = [];
        try {
            DB::beginTransaction();
            if ($request->hasFile('image')) {
                $name = $this->saveImage($request->file('image'));
                $data['image'] = $name;
            }
            $data['user_id'] = Auth::user()->id;
            $data['status'] = $request->status ? $request->status : false;
            $data['title'] = $request->title;
            $data['category_id'] = $request->category_id;
            $data['body'] = $request->body;
            $data['description'] = $request->description;
            $blog = $this->service->store($data);

            // notify all subscribed user through email
            event(new NotifySubscriber($blog, Auth::user()));
            $eventData["message"] = "New blog with title '" . $data['title'] . "' has been added by " .
Auth::user()->name . ".";
            $eventData["url"] = "/blog/" . $blog->id . "/view";
```

```
        // send push notification
        event(new BlogAdded($eventData));
        DB::commit();
        return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::SUCCESS]);
    } catch (Exception $exception) {
        DB::rollBack();
        return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::ADD_FAIL]);
    }
}
/**
 * @param $image
 * @return string
 */
private function saveImage($image)
{
    $extension = $image->getClientOriginalExtension();
    $name = time() . Str::random(5) . '.' . $extension;
    Storage::disk('public')->put("blog/" . $name, File::get($image));
    return $name;
}

/**
 * Display the specified resource.
 *
 * @param Blog $blog
 * @return Response
 */
public function show(Blog $blog)
{
    $title = $blog->title;
    $category = $this->categoryService->getActiveCategoryArray("status", "=", true);
    return view($this->view . "show", compact('title', 'blog', 'category'));
}
```

```
/**
 * @param Blog $blog
 * @return Factory|View
 */
public function edit(Blog $blog)
{
    $title = "Edit " . $this->title;
    $category = $this->categoryService->getActiveCategoryArray("status", "=", true);
    return view($this->view . "edit", compact('blog', 'title', 'category'));
}
/**
 * @param BlogRequestValidation $request
 * @param Blog $blog
 * @return RedirectResponse|Redirector
 */
public function update(BlogRequestValidation $request, Blog $blog)
{
    try {
        DB::beginTransaction();
        if ($request->hasFile('image')) {
            Storage::disk('public')->delete("blog/" . $blog->image);
            $name = $this->saveImage($request->file('image'));
            $blog->image = $name;
        }
        $blog->update($request->except('image'));
        DB::commit();
        return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::UPDATE]);
    } catch (Exception $exception) {
        DB::rollBack();
        return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::UPDATE_FAIL]);
    }
}
```

```
    /**
     * Remove the specified resource from storage.
     *
     * @param Blog $blog
     * @return Response
     */
    public function destroy(Blog $blog)
    {
        try {
            DB::beginTransaction();
            Storage::disk('public')->delete("blog/" . $blog->image);
            $blog->delete();
            DB::commit();
            return redirect($this->redirectUrl)->withErrors(['alert-success' => MessageService::DELETE]);
        } catch (Exception $exception) {
            DB::rollBack();
            return redirect($this->redirectUrl)->withErrors(['alert-danger' => MessageService::DELETE_FAIL]);
        }
    }
}
```

Auth/ConfirmPasswordController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Auth\ConfirmsPasswords;

class ConfirmPasswordController extends Controller
{
    use ConfirmsPasswords;
    /**
     * Where to redirect users when the intended url fails.
```

```
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }
}
```

Auth/LoginController.php

```
<?php
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
class LoginController extends Controller
{
    use AuthenticatesUsers;
    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;

    /**
     * Create a new controller instance.
     *
     * @return void
```

```
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Auth/RegisterController.php

```
<?php
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use App\User;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
class RegisterController extends Controller
{
    use RegistersUsers;

    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
```

```
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:3', 'confirmed'],
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'role' => 'writer',
            'password' => Hash::make($data['password']),
        ]);
    }
}
```

iv. Code listing of all Eloquent model classes.

Blog.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
class Blog extends Model
{
    protected $fillable = ['title', 'description', 'body', 'status', 'image', 'user_id', 'category_id'];
    /**
     * @return BelongsTo
     */
    public function user()
    {
        return $this->belongsTo(User::class, "user_id", "id");
    }

    /**
     * @return BelongsTo
     */
    public function category()
    {
        return $this->belongsTo(Category::class,"category_id", "id");
    }
}
```

Category.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Category extends Model
{
    protected $fillable = ["title","status"];
}
```

User.php

```
<?php
namespace App;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
class User extends Authenticatable
{
    use Notifiable;
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password','subscription'
    ];
    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
```

```
    ];
}
```

References:

https://laravel.com/docs/8.x/installation