***Controller/auth***

**LoginController.php**

<?php

namespace App\Http\Controllers\auth;

use App\Http\Controllers\Controller;

use Exception;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class LoginController extends Controller

{

    public function loginView()

    {

        return view('auth.login');

    }

    public function login(Request $request)

    {

        try {

            $credentials = $request->validate([

                'email' => 'required|email',

                'password' => 'required|min:8'

            ]);

            if (Auth::attempt($credentials)){

                return redirect()->route('home');

            }

            throw new Exception('Login Failed');

        } catch(Exception $e){

            return back()->with('error', $e->getMessage());

        }

    }

    public function logout(Request $request){

        Auth::logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect()->route('auth.login')->with('success', 'Log out Successfull');

    }

}

**RegisterController.php**

<?php

namespace App\Http\Controllers\auth;

use App\Http\Controllers\Controller;

use App\Models\User;

use Exception;

use Illuminate\Http\Request;

class RegisterController extends Controller

{

    public function registerView()

    {

        return view('auth.register');

    }

    public function register(Request $request){

        try {

            $request->validate([

                'name' => 'required|max:255',

                'email' => 'required|email',

                'phone' => 'required|numeric',

                'password' => 'required|min:8'

            ]);

            $data = [

                'name' => $request->name,

                'email' => $request->email,

                'phone' => $request->phone,

                'password' => bcrypt($request->password)

            ];

            User::create($data);

            return redirect()->route('auth.login')->with('success', 'Registration Succesful')->with('email', $request->email);

        } catch (Exception $e) {

            return back()->with('error', $e->getMessage());

        }

    }

}

***Controller/admin***

**AdminController.php**

<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class AdminController extends Controller

{

    public function index(){

        return view('admin.home');

    }

}

**ManageAdminController.php**

<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;

use App\Models\User;

use Exception;

use Illuminate\Http\Request;

class ManageAdminController extends Controller

{

    public function index(){

        $admins = User::where('role', 'ADMIN')->get();

        return view('admin.manageadmin.index', ['admins' => $admins]);

    }

    public function create(){

        return view('admin.manageadmin.create');

    }

    public function add\_admin(Request $request){

        try {

            $request->validate([

                'name' => 'required|max:255',

                'email' => 'required|email',

                'phone' => 'required|numeric',

                'password' => 'required|min:8'

            ]);

            $data = [

                'name' => $request->name,

                'email' => $request->email,

                'phone' => $request->phone,

                'password' => bcrypt($request->password),

                'role' => 'ADMIN'            ];

            User::create($data);

            return redirect()->route('admin.manageadmin.index')->with('success', 'Admin Register');

        } catch (Exception $e) {

            return back()->with('error', $e->getMessage());

        }

    }

}

**ManageUserController.php**

<?php

namespace App\Http\Controllers\admin;

use App\Http\Controllers\Controller;

use App\Models\User;

use Illuminate\Http\Request;

class ManageUserController extends Controller

{

    public function index(){

        $users = User::where('role', 'USER')->get();

        return view('admin.manageuser.index', ['users' => $users]);

    }

    public function delete\_user(Request $request){

        $id = $request->id;

        $user = User::findOrFail($id);

        $user->delete();

        return redirect()->back()->with('success', 'delete success');

    }

}

***Controller/user***

**ManageArticleController.php**

<?php

namespace App\Http\Controllers\user;

use App\Http\Controllers\Controller;

use App\Models\Article;

use App\Models\Category;

use Illuminate\Http\Request;

use function PHPUnit\Framework\fileExists;

class ManageArticleController extends Controller

{

    public function index(){

        $articles = auth()->user()->articles;

        return view('user.article.index', ['articles' => $articles]);

    }

    public function create(){

        $categories = Category::all();

        return view('user.article.create', ['categories' => $categories]);

    }

    public function store(Request $request){

        $request->validate([

            'title' => 'required|string',

            'description' => 'required',

            'category\_id' => 'required|exists:categories,id',

            'image' => 'required|image|mimes:jpg,png,jpeg'

        ]);

        if(!$request->hasFile('image')){

            return redirect()->back()->with('error', 'Image is required.');

        }

        $image = $request->file('image');

        $imageName = time().'.'.$image->getClientOriginalExtension();

        $image->move(public\_path('posts'), $imageName);

        Article::create([

            'title' => $request->title,

            'description' => $request->description,

            'category\_id' => $request->category\_id,

            'image' => $imageName,

            'user\_id' => auth()->user()->id

        ]);

        return redirect()->route('user.article.index')->with('success', 'Article Created!');

    }

    public function delete\_article(Request $request){

        $id = $request->id;

        $article = Article::findOrFail($id);

        $image\_path = public\_path('posts').'/'.$article->image;

        if(file\_exists($image\_path)){

            unlink($image\_path);

        }

        $article->delete();

        return redirect()->route('user.article.index')->with('success', 'Article deleted!');

    }

}

**ManageProfiController.php**

<?php

namespace App\Http\Controllers\user;

use App\Http\Controllers\Controller;

use App\Models\User;

use Illuminate\Http\Request;

class ManageProfilController extends Controller

{

    public function index(){

        $user = auth()->user();

        return view('user.profile', ['user' => $user]);

    }

    public function update(Request $request){

        $request->validate([

            'name' => 'required',

            'email' => 'required|email',

            'phone' => 'required|numeric'

        ]);

        $check = User::where('email', $request->email)->first();

        if ($check && $check->id != auth()->user()->id){

            return redirect()->back()->with('error', 'Email already exists');

        }

        $user = auth()->user();

        $user->update([

            'name' => $request->name,

            'email' => $request->email,

            'phone' => $request->phone

        ]);

    return redirect()->route('user.home')->with('success', 'Profile Updated');

    }

}

**UserController.php**

<?php

namespace App\Http\Controllers\user;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class UserController extends Controller

{

    public function index(){

        return view('user.home');

    }

}

**GuestController.php**

<?php

namespace App\Http\Controllers;

use App\Models\Article;

use App\Models\Category;

use Illuminate\Http\Request;

class GuestController extends Controller

{

    public function article\_index($id = 1){

        $category = Category::findOrFail($id);

        return view('guest.article.index', ['category' => $category]);

    }

    public function article\_detail($id){

        $article = Article::findOrFail($id);

        return view('guest.article.detail', ['article' => $article]);

    }

}

***Middleware (php artisan make:middleware nama\_middleware)***

**isAdmin.php**

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class isAdmin

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        if(auth()->user()->role == 'ADMIN'){

            return $next($request);

        }

        return redirect()->route('home')->with('error', 'Admin Only');

    }

}

**isLogin.php**

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class isLogin

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        if(auth()->check()){

            return $next($request);

        }

        return redirect()->route('auth.login')->with('error', 'login first to access this page.');

    }

}

**isNotLogin.php**

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class isNotLogin

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        if(auth()->check()){

            // udh login

            return redirect()->route('home')->with('error', 'Unauthorized.');

        }

        return $next($request);

    }

}

**isUser.php**

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class isUser

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        if(auth()->user()->role == 'USER'){

            return $next($request);

        }

        return redirect()->route('home')->with('error', 'User Only');

    }

}

**Localization.php**

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class Localization

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        //cek user ada ga pencet bahasa

        $locale = session('locale');

        $langs = ['en', 'id'];

        if(!in\_array($locale, $langs)){

            $locale = 'en';

        }

        app()->setLocale($locale);

        return $next($request);

    }

}

**Models**

**Article.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Article extends Model

{

    protected $fillable = ['user\_id', 'category\_id', 'title', 'description', 'image'];

    use HasFactory;

    public function user()

    {

        return $this->belongsTo(User::class, 'user\_id', 'id');

    }

    public function category()

    {

        return $this->belongsTo(Category::class);

    }

}

**Category.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Category extends Model

{

    use HasFactory;

    public function articles()

    {

        return $this->hasMany(Article::class);

    }

}

**User.php**

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

    use HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

        'phone',

        'role'

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* Get the attributes that should be cast.

     \*

     \* @return array<string, string>

     \*/

    protected function casts(): array

    {

        return [

            'email\_verified\_at' => 'datetime',

            'password' => 'hashed',

        ];

    }

    public function articles(){

        return $this->hasMany(Article::class);

    }

}

***Migrations***

**Create\_users**

Schema::create('users', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->string('email')->unique();

            $table->string('phone')->nullable();

            $table->string('role')->default('USER');

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('password');

            $table->rememberToken();

            $table->timestamps();

        });

**Create\_categories**  
Schema::create('categories', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->timestamps();

        });

**Create\_articles**

Schema::create('articles', function (Blueprint $table) {

            $table->id();

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

            $table->foreignId('category\_id')->constrained('categories')->onDelete('cascade');

            $table->string('title', 255);

            $table->longText('description');

            $table->string('image', 255);

            $table->timestamps();

        });

**Seeder**

**UserSeeder.php**

User::create([

            'name' => 'Admin Adam',

            'email' => 'admin@admin.com',

            'phone' => '081234567890',

            'password' => bcrypt('admin123'),

            'role' => 'ADMIN'

        ]);

        User::create([

            'name' => 'User Widoko',

            'email' => 'user@user.com',

            'phone' => '08987654321',

            'password' => bcrypt('admin123')

        ]);

        for ($i = 0; $i<10; $i++){

            User::create([

                'name' => fake()->name(),

                'email' => fake()->email(),

                'phone' => fake()->phoneNumber(),

                'password' => bcrypt('user123'),

            ]);

        }

**ArticleSeeder.php**

        $users = User::all();

        $categories = Category::all();

        foreach($categories as $item){

            for($i = 1 ; $i<=3; $i++){

                Article::create([

                    'user\_id' => $users->random()->id,

                    'category\_id' => $item->id,

                    'title' => $item->name." Article".$i,

                    'image' => 'default'.rand(1, 3).'.png',

                    'description' => fake()->paragraph(5),

                ]);

            }

        }

**DatabaseSeeder.php**

public function run(): void

    {

       $this->call([

            UserSeeder::class,

            CategorySeeder::class,

            ArticleSeeder::class,

       ]);

    }

**MasterTemplatenya**

**View/layouts (php artisan make:view nama\_folder.nama\_blade)**



**Alert.blade.php**



**Navbar.blade.php**

**Navbar Guest**

A screen shot of a computer

Description automatically generated

**Navbar Admin**

A screen shot of a computer

Description automatically generated

**Navbar User**

A computer screen shot of text

Description automatically generated

**Template tiap page**

A screenshot of a computer program

Description automatically generated

**Auth Blade**

**Login.blade.php**

A screenshot of a computer program

Description automatically generated

**Register.blade.php**

A screen shot of a computer screen

Description automatically generated

**Admin View**

**ManageAdmin**

**Index.blade.php**

A screen shot of a computer screen

Description automatically generated

A close-up of a website

Description automatically generated

**Create.blade.php**

A screen shot of a computer

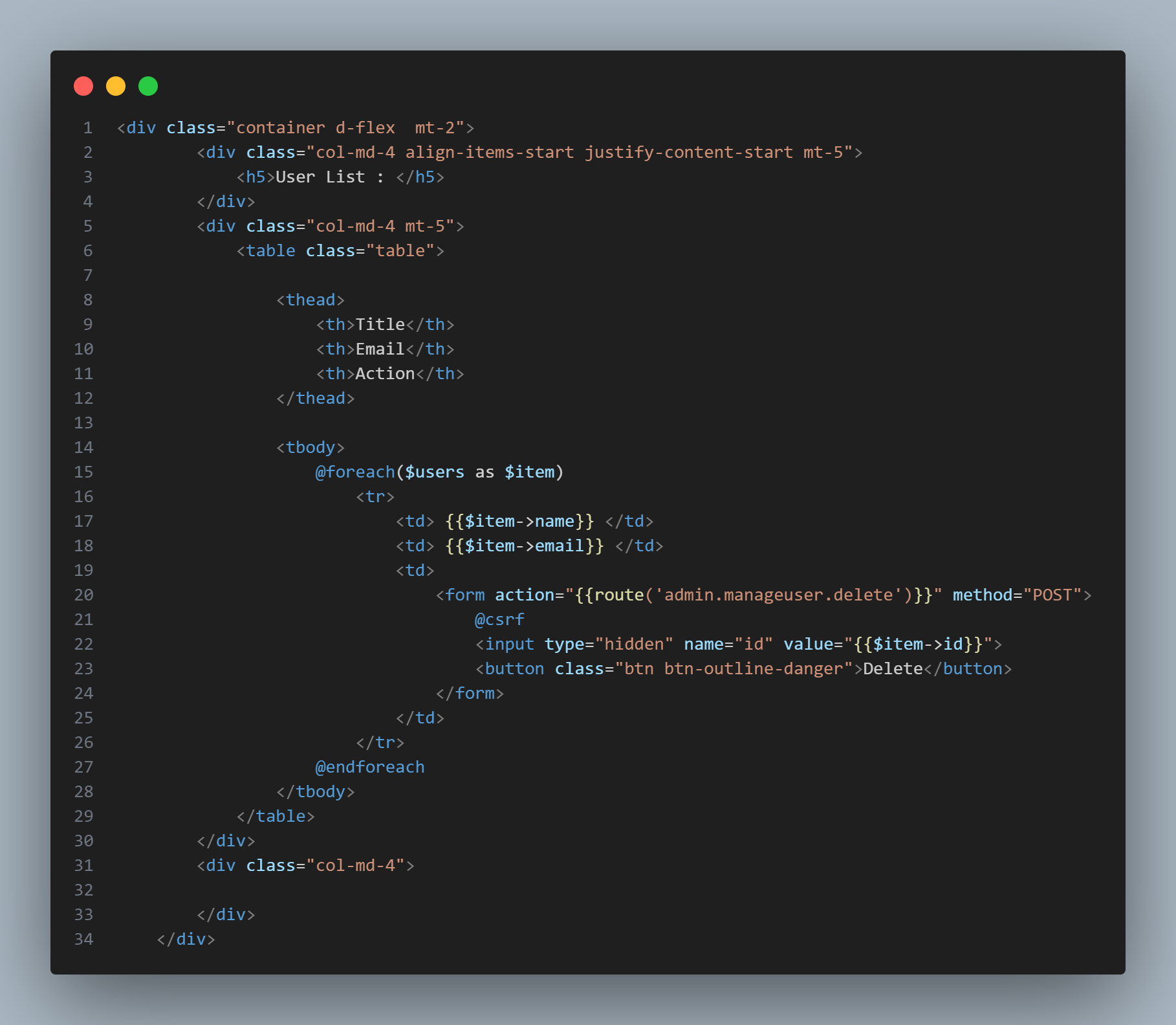
Description automatically generated

A screenshot of a computer

Description automatically generated

**ManageUser**

**Index.blade.php**



A screenshot of a computer

Description automatically generated

**Admin /** [**Home.blade.php**](http://Home.blade.php)



**Profile.blade.php**

A screenshot of a computer

Description automatically generated

**User/article/index.blade.php**



A screenshot of a computer

Description automatically generated

**User/article/create.blade.php**

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated

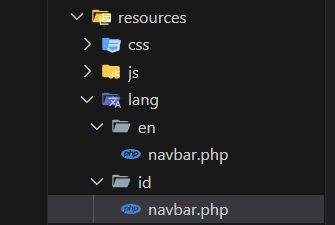
**Routing**

****

**A screen shot of a computer code

Description automatically generated**

**Localization**



Bikin folder lang

A screenshot of a computer

Description automatically generated

Cara makenya kayak ini:



KALO ENV ERROR MAKE INI:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=latianuacwoy

DB\_USERNAME=root

DB\_PASSWORD=

**DB\_COLLATION = utf8mb4\_unicode\_ci**