I. Create login with username

1. Open command and run :

|  |
| --- |
| php artisan make:auth |

2. Change file migration

\database\migrations\2014\_10\_12\_000000\_create\_users\_table.php

|  |
| --- |
| **public function** up() {  Schema::*create*(**'users'**, **function** (Blueprint $table) {  $table->increments(**'id'**);  $table->string(**'username'**)->unique();  $table->string(**'password'**, 60);  $table->string(**'email'**)->unique();  $table->tinyInteger(**'level'**);  $table->rememberToken();  *//$table->timestamps();* $table->timestamp(**'created\_at'**)->default(DB::*raw*(**'CURRENT\_TIMESTAMP'**));  $table->timestamp(**'updated\_at'**)->default(DB::*raw*(**'CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP'**));  }); } |

3. run command

|  |
| --- |
| php artisan migrate |

4. change validate and add protected username variable of \app\Http\Controllers\Auth\AuthController.php file :

|  |
| --- |
| **<?php  namespace** App\Http\Controllers\Auth;  **use** App\Http\Requests\LoginRequest; **use** App\User; **use** Validator; **use** App\Http\Controllers\Controller; **use** Illuminate\Foundation\Auth\ThrottlesLogins; **use** Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;   **class** AuthController **extends** Controller {  */\*  |--------------------------------------------------------------------------  | Registration & Login Controller  |--------------------------------------------------------------------------  |  | This controller handles the registration of new users, as well as the  | authentication of existing users. By default, this controller uses  | a simple trait to add these behaviors. Why don't you explore it?  |  \*/* **use** AuthenticatesAndRegistersUsers, ThrottlesLogins;   */\*\*  \* Where to redirect users after login / registration.  \*  \** ***@var*** *string  \*/* **protected $redirectTo** = **'/'**;   **protected $username** = **'username'**;    **protected $redirectAfterLogout** = **'/login'**;   */\*\*  \* Create a new authentication controller instance.  \*  \** ***@return*** *void  \*/* **public function** \_\_construct()  {  $this->middleware($this->guestMiddleware(), [**'except'** => **'logout'**]);  }   */\*\*  \* 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|max:255',* **'username'** => **'required|unique:users'**,  **'email'** => **'required|email|max:255|unique:users'**,  **'password'** => **'required|min:6|confirmed'**,  ]);  }   */\*\*  \* Create a new user instance after a valid registration.  \*  \** ***@param*** *array $data  \** ***@return*** *User  \*/* **protected function** create(**array** $data)  {  **return** User::*create*([  *//'name' => $data['name'],* **'username'** => $data[**'username'**],  **'email'** => $data[**'email'**],  **'password'** => bcrypt($data[**'password'**]),  ]);  } } |

5. Change login view \resources\views\auth\login.blade.php

|  |
| --- |
| **@extends('layouts.app')  @section('content')** <**div class="container"**>  <**div class="row"**>  <**div class="col-md-8 col-md-offset-2"**>  <**div class="panel panel-default"**>  <**div class="panel-heading"**>Login</**div**>  <**div class="panel-body"**>  <**form class="form-horizontal" role="form" method="POST" action="**{{ url(**'/login'**) }}**"**>  {{ csrf\_field() }}   <**div class="form-group**{{ $errors->has(**'username'**) ? **' has-error'** : **''** }}**"**>  <**label for="email" class="col-md-4 control-label"**>Username</**label**>   <**div class="col-md-6"**>  <**input id="username" type="text" class="form-control" name="username" value="**{{ old(**'username'**) }}**"**>   **@if (**$errors->has(**'username'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'username'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group**{{ $errors->has(**'password'**) ? **' has-error'** : **''** }}**"**>  <**label for="password" class="col-md-4 control-label"**>Password</**label**>   <**div class="col-md-6"**>  <**input id="password" type="password" class="form-control" name="password"**>   **@if (**$errors->has(**'password'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'password'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group"**>  <**div class="col-md-6 col-md-offset-4"**>  <**div class="checkbox"**>  <**label**>  <**input type="checkbox" name="remember"**> Remember Me  </**label**>  </**div**>  </**div**>  </**div**>   <**div class="form-group"**>  <**div class="col-md-6 col-md-offset-4"**>  <**button type="submit" class="btn btn-primary"**>  <**i class="fa fa-btn fa-sign-in"**></**i**> Login  </**button**>   <**a class="btn btn-link" href="**{{ url(**'/password/reset'**) }}**"**>Forgot Your Password?</**a**>  </**div**>  </**div**>  </**form**>  </**div**>  </**div**>  </**div**>  </**div**> </**div**> **@endsection** |

6. Change register view \resources\views\auth\register.blade.php

|  |
| --- |
| **@extends('layouts.app')  @section('content')** <**div class="container"**>  <**div class="row"**>  <**div class="col-md-8 col-md-offset-2"**>  <**div class="panel panel-default"**>  <**div class="panel-heading"**>Register</**div**>  <**div class="panel-body"**>  <**form class="form-horizontal" role="form" method="POST" action="**{{ url(**'/register'**) }}**"**>  {{ csrf\_field() }}   <**div class="form-group**{{ $errors->has(**'username'**) ? **' has-error'** : **''** }}**"**>  <**label for="name" class="col-md-4 control-label"**>Username</**label**>   <**div class="col-md-6"**>  <**input id="username" type="text" class="form-control" name="username" value="**{{ old(**'username'**) }}**"**>   **@if (**$errors->has(**'username'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'username'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group**{{ $errors->has(**'email'**) ? **' has-error'** : **''** }}**"**>  <**label for="email" class="col-md-4 control-label"**>E-Mail Address</**label**>   <**div class="col-md-6"**>  <**input id="email" type="email" class="form-control" name="email" value="**{{ old(**'email'**) }}**"**>   **@if (**$errors->has(**'email'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'email'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group**{{ $errors->has(**'password'**) ? **' has-error'** : **''** }}**"**>  <**label for="password" class="col-md-4 control-label"**>Password</**label**>   <**div class="col-md-6"**>  <**input id="password" type="password" class="form-control" name="password"**>   **@if (**$errors->has(**'password'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'password'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group**{{ $errors->has(**'password\_confirmation'**) ? **' has-error'** : **''** }}**"**>  <**label for="password-confirm" class="col-md-4 control-label"**>Confirm Password</**label**>   <**div class="col-md-6"**>  <**input id="password-confirm" type="password" class="form-control" name="password\_confirmation"**>   **@if (**$errors->has(**'password\_confirmation'**)**)** <**span class="help-block"**>  <**strong**>{{ $errors->first(**'password\_confirmation'**) }}</**strong**>  </**span**>  **@endif** </**div**>  </**div**>   <**div class="form-group"**>  <**div class="col-md-6 col-md-offset-4"**>  <**button type="submit" class="btn btn-primary"**>  <**i class="fa fa-btn fa-user"**></**i**> Register  </**button**>  </**div**>  </**div**>  </**form**>  </**div**>  </**div**>  </**div**>  </**div**> </**div**> **@endsection** |

7. check file \vendor\laravel\framework\src\Illuminate\Foundation\Auth\AuthenticatesUsers.php

View function loginUsername()

|  |
| --- |
| **<?php  namespace** Illuminate\Foundation\Auth;  **use** Illuminate\Http\Request; **use** Illuminate\Support\Facades\Auth; **use** Illuminate\Support\Facades\Lang;  **trait** AuthenticatesUsers {  **use** RedirectsUsers;   */\*\*  \* Show the application login form.  \*  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** getLogin()  {  **return** $this->showLoginForm();  }   */\*\*  \* Show the application login form.  \*  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** showLoginForm()  {  $view = property\_exists($this, **'loginView'**)  ? $this->**loginView** : **'auth.authenticate'**;   **if** (view()->exists($view)) {  **return** view($view);  }   **return** view(**'auth.login'**);  }   */\*\*  \* Handle a login request to the application.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** postLogin(Request $request)  {  **return** $this->login($request);  }   */\*\*  \* Handle a login request to the application.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** login(Request $request)  {  $this->validateLogin($request);   *// If the class is using the ThrottlesLogins trait, we can automatically throttle  // the login attempts for this application. We'll key this by the username and  // the IP address of the client making these requests into this application.* $throttles = $this->isUsingThrottlesLoginsTrait();   **if** ($throttles && $lockedOut = $this->hasTooManyLoginAttempts($request)) {  $this->fireLockoutEvent($request);   **return** $this->sendLockoutResponse($request);  }   $credentials = $this->getCredentials($request);   **if** (Auth::*guard*($this->getGuard())->attempt($credentials, $request->has(**'remember'**))) {  **return** $this->handleUserWasAuthenticated($request, $throttles);  }   *// If the login attempt was unsuccessful we will increment the number of attempts  // to login and redirect the user back to the login form. Of course, when this  // user surpasses their maximum number of attempts they will get locked out.* **if** ($throttles && ! $lockedOut) {  $this->incrementLoginAttempts($request);  }   **return** $this->sendFailedLoginResponse($request);  }   */\*\*  \* Validate the user login request.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@return*** *void  \*/* **protected function** validateLogin(Request $request)  {  $this->validate($request, [  $this->loginUsername() => **'required'**, **'password'** => **'required'**,  ]);  }   */\*\*  \* Send the response after the user was authenticated.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@param*** *bool $throttles  \** ***@return*** *\Illuminate\Http\Response  \*/* **protected function** handleUserWasAuthenticated(Request $request, $throttles)  {  **if** ($throttles) {  $this->clearLoginAttempts($request);  }   **if** (*method\_exists*($this, **'authenticated'**)) {  **return** $this->authenticated($request, Auth::*guard*($this->getGuard())->user());  }   **return** redirect()->intended($this->redirectPath());  }   */\*\*  \* Get the failed login response instance.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@return*** *\Illuminate\Http\Response  \*/* **protected function** sendFailedLoginResponse(Request $request)  {  **return** redirect()->back()  ->withInput($request->only($this->loginUsername(), **'remember'**))  ->withErrors([  $this->loginUsername() => $this->getFailedLoginMessage(),  ]);  }   */\*\*  \* Get the failed login message.  \*  \** ***@return*** *string  \*/* **protected function** getFailedLoginMessage()  {  **return** Lang::*has*(**'auth.failed'**)  ? Lang::*get*(**'auth.failed'**)  : **'These credentials do not match our records.'**;  }   */\*\*  \* Get the needed authorization credentials from the request.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@return*** *array  \*/* **protected function** getCredentials(Request $request)  {  **return** $request->only($this->loginUsername(), **'password'**);  }   */\*\*  \* Log the user out of the application.  \*  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** getLogout()  {  **return** $this->logout();  }   */\*\*  \* Log the user out of the application.  \*  \** ***@return*** *\Illuminate\Http\Response  \*/* **public function** logout()  {  Auth::*guard*($this->getGuard())->logout();   **return** redirect(property\_exists($this, **'redirectAfterLogout'**) ? $this->**redirectAfterLogout** : **'/'**);  }   */\*\*  \* Get the guest middleware for the application.  \*/* **public function** guestMiddleware()  {  $guard = $this->getGuard();   **return** $guard ? **'guest:'**.$guard : **'guest'**;  }   */\*\*  \* Get the login username to be used by the controller.  \*  \** ***@return*** *string  \*/* **public function** loginUsername()  {  **return** property\_exists($this, **'username'**) ? $this->**username** : **'email'**;  }   */\*\*  \* Determine if the class is using the ThrottlesLogins trait.  \*  \** ***@return*** *bool  \*/* **protected function** isUsingThrottlesLoginsTrait()  {  **return** *in\_array*(  ThrottlesLogins::**class**, class\_uses\_recursive(**static**::**class**)  );  }   */\*\*  \* Get the guard to be used during authentication.  \*  \** ***@return*** *string|null  \*/* **protected function** getGuard()  {  **return** property\_exists($this, **'guard'**) ? $this->**guard** : **null**;  } } |

8. In controller, you want login first to access, add code :

|  |
| --- |
| **public function** \_\_construct() {  $this->middleware(**'auth'**); } |

Or in routes.php file, you can add **'middleware'** => **'auth'**:

|  |
| --- |
| Route::*group*([**'prefix'** => **'admin'**, **'middleware'** => **'auth'**], **function** (){Route::*group*([**'prefix'** => **'cate'**], **function** (){  Route::*get*(**'list'**, [**'as'** => **'admin.cate.list'**, **'uses'** => **'CateController@getList'**]);  }); |

Tip : show username, and logout

|  |
| --- |
| <**ul class="nav navbar-nav navbar-right"**>  *<!-- Authentication Links -->* **@if (**Auth::*guest*()**)** <**li**><**a href="**{{ url(**'/login'**) }}**"**>Login</**a**></**li**>  <**li**><**a href="**{{ url(**'/register'**) }}**"**>Register</**a**></**li**>  **@else** <**li class="dropdown"**>  <**a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-expanded="false"**>  {{ Auth::*user*()->**username** }} <**span class="caret"**></**span**>  </**a**>   <**ul class="dropdown-menu" role="menu"**>  <**li**><**a href="**{{ url(**'/logout'**) }}**"**><**i class="fa fa-btn fa-sign-out"**></**i**>Logout</**a**></**li**>  </**ul**>  </**li**>  **@endif** </**ul**> |

II. Check login level for admin

1. Create middleware for check level

|  |
| --- |
| php artisan make:middleware CheckLevel |

2. Declare middleware in \app\Http\Kernel.php file :

|  |
| --- |
| **'level'** => \App\Http\Middleware\CheckLevel::**class**, |

|  |
| --- |
| **<?php  namespace** App\Http;  **use** Illuminate\Foundation\Http\Kernel **as** HttpKernel;  **class** Kernel **extends** HttpKernel {  */\*\*  \* The application's global HTTP middleware stack.  \*  \* These middleware are run during every request to your application.  \*  \** ***@var*** *array  \*/* **protected $middleware** = [  \Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode::**class**,  ];   */\*\*  \* The application's route middleware groups.  \*  \** ***@var*** *array  \*/* **protected $middlewareGroups** = [  **'web'** => [  \App\Http\Middleware\EncryptCookies::**class**,  \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::**class**,  \Illuminate\Session\Middleware\StartSession::**class**,  \Illuminate\View\Middleware\ShareErrorsFromSession::**class**,  \App\Http\Middleware\VerifyCsrfToken::**class**,  ],   **'api'** => [  **'throttle:60,1'**,  ],  ];   */\*\*  \* The application's route middleware.  \*  \* These middleware may be assigned to groups or used individually.  \*  \** ***@var*** *array  \*/* **protected $routeMiddleware** = [  **'auth'** => \App\Http\Middleware\Authenticate::**class**,  **'auth.basic'** => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::**class**,  **'can'** => \Illuminate\Foundation\Http\Middleware\Authorize::**class**,  **'guest'** => \App\Http\Middleware\RedirectIfAuthenticated::**class**,  **'throttle'** => \Illuminate\Routing\Middleware\ThrottleRequests::**class**,  **'level'** => \App\Http\Middleware\CheckLevel::**class**,  ]; } |

3. Check level in middleware

|  |
| --- |
| **<?php  namespace** App\Http\Middleware;  **use** Closure; **use** Auth;  **class** CheckLevel {  */\*\*  \* Handle an incoming request.  \*  \** ***@param*** *\Illuminate\Http\Request $request  \** ***@param*** *\Closure $next  \** ***@return*** *mixed  \*/* **public function** handle($request, Closure $next)  {  **if** (Auth::*user*()->**level** != 1) {  **return** redirect()->guest(**'login'**);  }   **return** $next($request);  } } |

4. Call middle ware in routes.php for check admin :

|  |
| --- |
| Route::*group*([**'prefix'** => **'admin'**, **'middleware'** => [**'auth'**,**'level'**]], **function** (){ *//Route::group(['prefix' => 'admin'], function (){* Route::*group*([**'prefix'** => **'cate'**], **function** (){  Route::*get*(**'list'**, [**'as'** => **'admin.cate.list'**, **'uses'** => **'CateController@getList'**]);  Route::*get*(**'add'**, [**'as'** => **'admin.cate.getAdd'**, **'uses'** => **'CateController@getAdd'**]);  Route::*post*(**'add'**, [**'as'** => **'admin.cate.postAdd'**, **'uses'** => **'CateController@postAdd'**]);  Route::*get*(**'delete/{id}'**, [**'as'** => **'admin.cate.getDelete'**, **'uses'** => **'CateController@getDelete'**]);  Route::*get*(**'edit/{id}'**, [**'as'** => **'admin.cate.getEdit'**, **'uses'** => **'CateController@getEdit'**]);  Route::*post*(**'edit/{id}'**, [**'as'** => **'admin.cate.postEdit'**, **'uses'** => **'CateController@postEdit'**]);  }); }); |