wemwgjøvwni ivngvwbi ivwng

Follow on: https://www.youtube.com/watch?v=L6xaLfuqGfU&list=PLjsp2uDzfb31NqnycsDO41j18bYeKOGp-&index=4&ab\_channel=WebJourney  
jviv‡fj ïiæ:

1. composer create-project --prefer-dist laravel/laravel blog

Zvici

1. php artisan serve

KgvÛ w`‡q †`Le mvfv©i ivb K‡i wKbv Ges[**http://127.0.0.1:8000/**](http://127.0.0.1:8000/) GLv‡b wM‡q †`Le mvBU wVKg‡Zv BÝUªj n‡q‡Q wKbv | wVK \_vK‡j mvBU †`Lv‡e |  
Gevi mivmwi P‡j hv‡ev ‡KvW GwWU‡i Ges cª‡Rw± I‡cb Kie |

Gici .env dvBj I‡cb K‡i WvUv‡e‡mi bvg BDRvi bvg I cvmIqvW© ewm‡q †mf w`‡ev |

1. DB\_CONNECTION=mysql
2. DB\_HOST=127.0.0.1
3. DB\_PORT=3306
4. DB\_DATABASE=laravel
5. DB\_USERNAME=root
6. DB\_PASSWORD= 123

jviv‡fj †RUw÷ªg A‡\_bwU‡Kkb wm‡÷g Pvjy Kivi Rb¨ wb‡Pi avc¸‡jv Abymib Kiæb | wb‡Pi KgvÛ ivb Kiæb

1. composer **require** laravel/jetstream

Gevi †RUw÷ªg Gi mv‡\_ jvBfIq¨vi BÝUªj Kivi Rb¨ wb‡Pi KgvÛ ivb Kiæb |

1. php artisan jetstream:install livewire
2. php artisan jetstream:install livewire --teams
4. #Or, Install Jetstream With Inertia
6. php artisan jetstream:install inertia
7. php artisan jetstream:install inertia --teams

Gevi wb‡Pi KgvÛ¸‡jv ivb Kiæb

1. npm install
2. npm run dev

Gevi gvB‡MÖkb dvBj GwWU Ki‡Z n‡e gvB‡MÖkb KgvÛ ivb Kivi Av‡M wb‡Pi g‡Zv K‡i | GLv‡b Avgiv BDRvi Gi Z\_¨ ivLvi Rb¨ K‡qKwU wdì AwZwi³ wb‡q wb‡qwQ | †hgb, BDRvi‡bg, ‡dvb b¤^i, †mvm¨vj wgwWqv, ev‡qv, BDRvi UvBc, BDRvi ÷¨vUvm BZ¨vw`|

1. **public** **function** up()
2. {
3. Schema::create('users', **function** (Blueprint $table) {
4. $table->id();
5. $table->string('name');
6. $table->string('username')->unique();
7. $table->string('email')->unique();
8. $table->timestamp('email\_verified\_at')->nullable();
9. $table->string('password');
10. $table->string('phone\_no')->nullable();;
11. $table->string('social\_media')->nullable();;
12. $table->string('bio')->nullable();;
13. $table->integer('user\_type')->**default**('0');
14. $table->tinyInteger('status')->**default**('1');
15. $table->rememberToken();
16. $table->foreignId('current\_team\_id')->nullable();
17. $table->string('profile\_photo\_path', 2048)->nullable();
18. $table->timestamps();
19. });
20. }

Gevi gvB‡MÖkb KgvÛ ivb K‡i †Uwej ˆZwi m¤úbœ Kie

1. php artisan migrate

Gevi Avgiv BDRvi †iwR‡óªkb †cB‡R wM‡q †fwj‡Wkb GW Kie BDRvi‡bg Gi Rb¨|

1. App\Actions\Fortify\CreateNewUser
2. **public** **function** create(**array** $input)
3. {
4. Validator::make($input, [
5. 'name' => ['required', 'string', 'max:255'],
6. **'username' => ['required', 'string','min:3', 'max:100', 'unique:users'],**
7. 'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
8. 'password' => $this->passwordRules(),
9. 'terms' => Jetstream::hasTermsAndPrivacyPolicyFeature() ? ['required', 'accepted'] : '',
10. ])->validate();
12. **return** DB::transaction(**function** () **use** ($input) {
13. **return** tap(User::create([
14. 'name' => $input['name'],
15. **'username' => $input['username'],**
16. 'email' => $input['email'],
17. 'password' => Hash::make($input['password']),
18. ]), **function** (User $user) {
19. $this->createTeam($user);
20. });
21. });
22. }

Gevi bZzb BDRvi‡bg wdì ˆZwi Kie ‡eøW dvB‡j bZzb wmGmGm †hvM K‡i wb‡Pi g‡Zv K‡i |

1. Go to - resources/views/auth/register.blade.php
2. <div **class**="mt-4">
3. <x-jet-label **for**="username" value="{{ \_\_('Username') }}" />
4. <x-jet-input id="username" **class**="block mt-1 w-full" type="text" name="username" :value="old('username')" required />
5. </div>

Gevi BDRvi g‡W‡j wM‡q wd‡jej wd‡ì bZzb wdì¸‡jv †hvM Kie wb‡Pi g‡Zv K‡i |

App\Models\User

1. **protected** $fillable = [
2. 'name', 'username', 'phone\_no', 'social\_media', 'bio', 'user\_type', 'status', 'email', 'password',
3. ];

Gevi Avgiv †nvg †c‡R wM‡q bZzb BDRvi ‡iwR‡óªkb Kie Ges jwMb K‡i †cÖvdvBj Qwe Avc‡jvW Kie | Qwe Avc‡jvW Kivi Av‡M †cÖvdvBj wcK AckbwU Pvjy Ki‡Z n‡e Ges †÷v‡iR KgvÛ Pvjv‡Z n‡e | Avmyb †`wL wb‡Pi KvR¸‡jv :

wb‡Pi dvBjwU I‡cb K‡i K‡g›U Kiv wdPvi AckbwU AvbK‡g›U K‡i Pvjy Kwi |

config/jetstream.php

1. 'features' => [
2. // Features::termsAndPrivacyPolicy(),
3. **Features::profilePhotos(),**
4. // Features::api(),
5. //Features::teams(['invitations' => true]),
6. Features::accountDeletion(),
7. ],

Gevi .env dvB‡j wM‡q Gc BDAviGj Gi mv‡\_ 8000 w`‡ev Ges wb‡Pi †÷v‡iR KgvÛ Pvwj‡q †cÖvdvBj wcK Avc‡jvW Kwi

1. php artisan storage:link
2. APP\_URL=http://localhost  // Replace to Like Below Line
3. APP\_URL=http://localhost:8000

Gevi Avgiv BDRvi‡bg w`‡q jwMb Ki‡Z n‡j jwMb dvBj GwWU Kiv jvM‡e wb‡Pi g‡Zv K‡i |

Go to - resources/views/auth/login.blade.php

1. <div>
2. <x-jet-label **for**="email" value="{{ \_\_('Email / Username / Phone No') }}" />
3. <x-jet-input id="email" **class**="block mt-1 w-full" type="text" name="email" :value="old('email')" required autofocus />
4. </div>

Gevi Avgiv hv‡ev app/provider/JetstreamServiceProvider.php GLv‡b Avgiv wb‡Pi jvj Kvjvi Kiv †KvWUzKz emv‡ev RvqMvg‡Zv |

1. <?php
2. namespace App\Providers;
3. **use** App\Models\User;
4. **use** Illuminate\Http\Request;
5. **use** Laravel\Fortify\Fortify;
6. **use** Laravel\Jetstream\Jetstream;
7. **use** Illuminate\Support\Facades\Hash;
8. **use** App\Actions\Jetstream\CreateTeam;
9. **use** App\Actions\Jetstream\DeleteTeam;
10. **use** App\Actions\Jetstream\DeleteUser;
11. **use** Illuminate\Support\ServiceProvider;
12. **use** App\Actions\Jetstream\AddTeamMember;
13. **use** App\Actions\Jetstream\UpdateTeamName;
14. **use** App\Actions\Jetstream\InviteTeamMember;
15. **use** App\Actions\Jetstream\RemoveTeamMember;
17. **class** JetstreamServiceProvider **extends** ServiceProvider
18. {
19. /\*\*
20. \* Register any application services.
21. \*
22. \* @return void
23. \*/
24. **public** **function** register()
25. {
26. //
27. }
29. /\*\*
30. \* Bootstrap any application services.
31. \*
32. \* @return void
33. \*/
34. **public** **function** boot()
35. {
36. $this->configurePermissions();
38. Jetstream::createTeamsUsing(CreateTeam::**class**);
39. Jetstream::updateTeamNamesUsing(UpdateTeamName::**class**);
40. Jetstream::addTeamMembersUsing(AddTeamMember::**class**);
41. Jetstream::inviteTeamMembersUsing(InviteTeamMember::**class**);
42. Jetstream::removeTeamMembersUsing(RemoveTeamMember::**class**);
43. Jetstream::deleteTeamsUsing(DeleteTeam::**class**);
44. Jetstream::deleteUsersUsing(DeleteUser::**class**);
46. **Fortify::authenticateUsing(function (Request $request) {**
47. **$user = User::where('email', $request->email)**
48. **->orWhere('username', $request->email)**
49. **->orWhere('phone\_no', $request->email)**
50. **->first();**
52. **if ($user && Hash::check($request->password, $user->password)) {**
53. **if ($user->status == 1) {  // it will return if status == 1**
54. **return $user;**
55. **} else {**
56. **//Session::flash('error', 'Your account is blocked !');**
57. **$request->session()->flash('block-warning', 'Sorry ! Your account is blocked !');**
58. **return false;**
59. **}**
60. **}**
61. **});**
62. }
64. /\*\*
65. \* Configure the roles and permissions that are available within the application.
66. \*
67. \* @return void
68. \*/
69. **protected** **function** configurePermissions()
70. {
71. Jetstream::defaultApiTokenPermissions(['read']);
73. Jetstream::role('admin', 'Administrator', [
74. 'create',
75. 'read',
76. 'update',
77. 'delete',
78. ])->description('Administrator users can perform any action.');
80. Jetstream::role('editor', 'Editor', [
81. 'read',
82. 'create',
83. 'update',
84. ])->description('Editor users have the ability to read, create, and update.');
85. }
86. }

Gevi Avgiv jwMb †c‡R wM‡q eøM BDRvi hLb jwMb Ki‡Z PvB‡e ZLb ‡m jwMb Ki‡Z cvi‡e bv | Zv‡K GKUv †g‡mR †`Lv‡Z n‡e †hb †m ey‡S †h †m eøK n‡q Av‡Q | Gfv‡e wjL‡Z n‡e login.blade.php †c‡R

1. @**if** (session('block-warning'))
2. <div **class**="mb-4 text-lg text-center font-bold text-red-600">
3. {{ session('block-warning') }}
4. </div>
5. @**else**
6. <x-jet-validation-errors **class**="mb-4" />
7. @**endif**

Gevi Avgiv mwVKfv‡e jwMb Ki‡Z cvi‡ev B‡gBj/BDRvi‡bg/‡dvb b¤^i e¨envi K‡i | hw` BDRvi eøK \_v‡K Zvn‡j †m GKUv Gii †g‡mR †`L‡Z cvi‡K †h †m eøK Av‡Q |

jwMb †kl Gevi Avgiv †cÖvdvBj GwWU Kivq KvR Kie |

cÖ\_gZ BDRvi †cÖvdvBj Gi ivDU †`Iqv Av‡Q wb‡Pi †jv‡Kk‡b| GLv‡b †h‡Z Av‡M GB KgvÛwU Pvjv‡Z n‡e

1. php artisan vendor:publish --tag=jetstream-routes

Gevi ivDU †dvìv‡i wM‡q †RUw÷ªg dvBjwU I‡cb Ki‡jB wb‡Pi g‡Zv GKUv ivDU †`Lv hv‡e |

1. Route::get('/user/profile', [UserProfileController::**class**, 'show'])
2. ->name('profile.show');

Go to - vendor/laravel/jetstream/src/Http/Controllers/Livewire/UserProfileController.php

1. **public** **function** show(Request $request)
2. {
3. **return** view('profile.show', [
4. 'request' => $request,
5. 'user' => $request->user(),
6. ]);
7. }

Now Go to “resources/views/profile” and open show.blade.php file and Go to “update-profile-information-form.blade.php”

Gevi wb‡Pi wjLv wdì¸‡jv GW Kwi

1. <!-- Name -->
2. <div **class**="col-span-6 sm:col-span-4">
3. <x-jet-label **for**="name" value="{{ \_\_('Name') }}" />
4. <x-jet-input id="name" type="text" **class**="mt-1 block w-full" wire:model.defer="state.name" autocomplete="name" />
5. <x-jet-input-error **for**="name" **class**="mt-2" />
6. </div>
8. <!-- Username -->
9. <div **class**="col-span-6 sm:col-span-4">
10. <x-jet-label **for**="username" value="{{ \_\_('Username') }}" />
11. <x-jet-input id="username" type="text" **class**="mt-1 block w-full" wire:model.defer="state.username" />
12. <x-jet-input-error **for**="username" **class**="mt-2" />
13. </div>
15. <!-- Email -->
16. <div **class**="col-span-6 sm:col-span-4">
17. <x-jet-label **for**="email" value="{{ \_\_('Email') }}" />
18. <x-jet-input id="email" type="email" **class**="mt-1 block w-full" wire:model.defer="state.email" />
19. <x-jet-input-error **for**="email" **class**="mt-2" />
20. </div>
22. <!-- Phone Number -->
23. <div **class**="col-span-6 sm:col-span-4">
24. <x-jet-label **for**="phone\_no" value="{{ \_\_('Phone Number') }}" />
25. <x-jet-input id="phone\_no" type="number" **class**="mt-1 block w-full" wire:model.defer="state.phone\_no" />
26. <x-jet-input-error **for**="phone\_no" **class**="mt-2" />
27. </div>
29. <!-- Social Media Link -->
30. <div **class**="col-span-6 sm:col-span-4">
31. <x-jet-label **for**="social\_media" value="{{ \_\_('Social Media Profile Link') }}" />
32. <x-jet-input id="social\_media" type="text" **class**="mt-1 block w-full" wire:model.defer="state.social\_media" />
33. <x-jet-input-error **for**="social\_media" **class**="mt-2" />
34. </div>
36. <!-- User Short Biography -->
37. <div **class**="col-span-6 sm:col-span-4">
38. <x-jet-label **for**="bio" value="{{ \_\_('User Short Bio') }}" />
39. <x-jet-textarea id="bio" **class**="mt-1 block w-full" wire:model.defer="state.bio" />
40. <x-jet-input-error **for**="bio" **class**="mt-2" />
41. </div>

Gevi Avgiv wdì¸‡jvi Z\_¨ Avc‡WU Ki‡Z K‡›Uªvjv‡i P‡j hv‡ev GB †jv‡Kk‡b : "app/action/fortify/UpdateUserProfileInformation.php” Open it and edit like this..

1. **public** **function** update($user, **array** $input)
2. {
3. Validator::make($input, [
4. 'name' => ['required', 'string', 'max:255'],
5. **'username' => ['required', 'min:3', 'max:255', Rule::unique('users')->ignore($user->id)],**
6. **'email' => ['required', 'email', 'max:255', Rule::unique('users')->ignore($user->id)],**
7. **'phone\_no' => ['nullable', 'string', 'max:255'],**
8. **'social\_media' => ['nullable', 'string', 'max:255'],**
9. **'bio' => ['nullable', 'string', 'max:255'],**
10. 'photo' => ['nullable', 'mimes:jpg,jpeg,png', 'max:1024'],
11. ])->validateWithBag('updateProfileInformation');
13. **if** (isset($input['photo'])) {
14. $user->updateProfilePhoto($input['photo']);
15. }
17. **if** ($input['email'] !== $user->email **&& $input['username'] !== $user->username** &&
18. $user instanceof MustVerifyEmail) {
19. $this->updateVerifiedUser($user, $input);
20. } **else** {
21. $user->forceFill([
22. **'name' => $input['name'],**
23. **'username' => $input['username'],**
24. **'email' => $input['email'],**
25. **'phone\_no' => $input['phone\_no'],**
26. **'social\_media' => $input['social\_media'],**
27. **'bio' => $input['bio'],**
28. ])->save();
29. }
30. }
32. /\*\*
33. \* Update the given verified user's profile information.
34. \*
35. \* @param  mixed  $user
36. \* @param  array  $input
37. \* @return void
38. \*/
39. **protected** **function** updateVerifiedUser($user, **array** $input)
40. {
41. $user->forceFill([
42. 'name' => $input['name'],
43. **'username' => $input['username'],**
44. **'email' => $input['email'],**
45. **'phone\_no' => $input['phone\_no'],**
46. **'social\_media' => $input['social\_media'],**
47. **'bio' => $input['bio'],**
48. 'email\_verified\_at' => null,
49. ])->save();
51. $user->sendEmailVerificationNotification();
52. }

GLb †cÖvdvBj †c‡R ev‡qv Gi wdìUv‡K †U· Gwiqv Ki‡Z n‡e wb‡Pi g‡Zv K‡i

1. <x-jet-textarea id="bio" **class**="mt-1 block w-full" wire:model.defer="state.bio" />

‡RUw÷ª‡g ‡U·Gwiqv Kg‡cv‡b›U wWdìfv‡e ˆZwi Kiv \_v‡K bv GUv‡K ˆZwi K‡i †iwRóªvi Ki‡Z n‡e | GB dvBjwU ˆZwi Ki‡Z n‡e resource/views/vendor/jetstream/component Gi wfZ‡i Gfv‡e textarea.blade.php

1. @props(['disabled' => false])
3. <textarea rows='5' {{ $disabled ? 'disabled' : '' }}  {!! $attributes->merge(['class' => 'border-gray-300 focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50 rounded-md shadow-sm']) !!}></textarea>

Gevi GUv‡K †mf w`‡q P‡j †h‡Z n‡e Kg‡cv‡b›U †iwRóªvi †c‡R | vendor\laravel\jetstream\src

JetstreamServiceProvider.php

dvBjwU I‡cb K‡i wb‡Pi jvj †KvWUzKz GW Kie

1. **protected** **function** configureComponents()
2. {
3. $this->callAfterResolving(BladeCompiler::**class**, **function** () {
4. $this->registerComponent('action-message');
5. $this->registerComponent('action-section');
6. $this->registerComponent('application-logo');
7. $this->registerComponent('application-mark');
8. $this->registerComponent('authentication-card');
9. $this->registerComponent('authentication-card-logo');
10. $this->registerComponent('banner');
11. $this->registerComponent('button');
12. $this->registerComponent('confirmation-modal');
13. $this->registerComponent('confirms-password');
14. $this->registerComponent('danger-button');
15. $this->registerComponent('dialog-modal');
16. $this->registerComponent('dropdown');
17. $this->registerComponent('dropdown-link');
18. $this->registerComponent('form-section');
19. $this->registerComponent('input');
20. **$this->registerComponent('textarea');**
21. $this->registerComponent('checkbox');
22. $this->registerComponent('input-error');
23. $this->registerComponent('label');
24. $this->registerComponent('modal');
25. $this->registerComponent('nav-link');
26. $this->registerComponent('responsive-nav-link');
27. $this->registerComponent('responsive-switchable-team');
28. $this->registerComponent('secondary-button');
29. $this->registerComponent('section-border');
30. $this->registerComponent('section-title');
31. $this->registerComponent('switchable-team');
32. $this->registerComponent('validation-errors');
33. $this->registerComponent('welcome');
34. });
35. }

Gevi †cRwU †jvW w`‡j †`Lv hv‡e GKUv †U·U Gwiqv e· Avm‡Q ev‡qv wjLvi Rb¨ | Gevi cÖ‡qvRbxq Z\_¨ wj‡L †mf w`‡Z n‡e

Gevi Avgiv BDRvi UvBc wb‡q KvR Kie | Avgv‡`i \_vK‡e 5 cÖKv‡ii BDRvi ‡hgb: GWwgwb‡÷ªUi, GwWUi, A\_i, Kw›UªweDUi, mve¯ŒvBfvi | Zvn‡j cÖ‡Z¨‡Ki Rb¨ GKUv K‡i wgWjIq¨vi ˆZwi K‡i wb‡Z n‡e wb‡Pi KgvÛ w`‡q |  
**php artisan make:middleware IsAdministrator**

Gfv‡e 4 Uv wgWjIq¨vi evbv‡ev Avi jv÷ Gi mve¯ŒvBfvi wWdë wn‡m‡e \_vK‡e |

Gevi Avgiv wgWjIq¨vi‡K ivDU wgWjIq¨vi GW Ki‡Z n‡e me¸‡jv‡K | wb‡Pi †jv‡Kkb wM‡q evwK KvRUv wb‡Pi g‡Zv Kwi

**app/Http/Kernel.php**

1. **protected** $routeMiddleware = [
2. 'auth' => \App\Http\Middleware\Authenticate::**class**,
3. 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::**class**,
4. 'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::**class**,
5. 'can' => \Illuminate\Auth\Middleware\Authorize::**class**,
6. 'guest' => \App\Http\Middleware\RedirectIfAuthenticated::**class**,
7. 'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::**class**,
8. 'signed' => \Illuminate\Routing\Middleware\ValidateSignature::**class**,
9. 'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::**class**,
10. 'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::**class**,
12. 'is\_admin' => \App\Http\Middleware\Admin\IsAdmin::**class**,
13. 'is\_author' => \App\Http\Middleware\Admin\IsAuthor::**class**,
14. 'is\_editor' => \App\Http\Middleware\Admin\IsEditor::**class**,
15. 'is\_contributor' => \App\Http\Middleware\Admin\IsContributor::**class**,
16. ];

Gevi Avgiv cÖ‡Z¨KwU wgWjIq¨v‡i Bd KwÛkb emv‡ev wb‡Pi g‡Zv K‡i|

1. <?php
2. namespace App\Http\Middleware\Admin;
4. **use** Closure;
5. **use** Illuminate\Http\Request;
7. **class** IsAdmin
8. {
9. **public** **function** handle(Request $request, Closure $next)
10. {
11. if (auth()->user()->user\_type == 1) {
12. return $next($request);
13. }
15. return redirect('home')->with('error', "You don't have admin access.");
16. }
17. }

Gevi P‡j hv‡ev **routes/web.php** wM‡q cÖ‡Z¨K BDRvi Gi Rb¨ Avjv`v Avjv`v ivDU MÖæc ‰Zwi Ki‡Z n‡e |

1. Route::group(['prefix' => 'admin', 'as' => 'admin.', 'middleware' => ['auth:sanctum', 'verified', 'is\_admin']], **function** () {
2. Route::get('/dashboard', [DashboardController::**class**, 'show'])->name('dashboard'); // admin.dashboard
4. Route::get('/post', **function** () {
5. **return** 'Got it';
6. });
7. });


11. // General Users / Subcribers
12. Route::group(['prefix' => 'user', 'as' => 'user.', 'middleware' => ['auth:sanctum', 'verified']], **function** () {
13. Route::get('/dashboard', [DashboardController::**class**, 'show'])->name('dashboard'); // admin.dashboard
15. Route::get('/post', **function** () {
16. **return** 'Got it';
17. });
18. });

wfD †c‡Ri Rb¨ GWwgb G‡·m †bIqv Dcvq: 2 wU c×wZ †`Iqv n‡jv

cÖ\_‡g Avgiv K‡›Uªvjvi †c‡R wM‡q Gfv‡e wjLe

1. **class** DashboardController **extends** Controller
2. {
3. **public** **function** show(){
5. $isAdmin = Auth()->user()->user\_type == 1;
6. $isEditor = Auth()->user()->user\_type == 2;
7. $isAuthor = Auth()->user()->user\_type == 3;
8. $isContributor = Auth()->user()->user\_type == 4;
9. **return** view('dashboard', compact('isAdmin', 'isEditor', 'isAuthor', 'isContributor'));
10. }
11. }

Mehtod 2 Go App/Provider/AppServiceProvier.php

1. Gate::define('isAdmin', **function**($user) {
2. **return** $user->user\_type == '1';
3. });

Access in Blade Page like this

1. {{ Auth::user()->name }}
3. @**if**(Auth::check() && Auth::user()->user\_type == "1")
4. {{ \_\_('Admin Dashboard') }}
5. @**else**
6. {{ \_\_('User Dashboard') }}
7. @**endif**
9. @**if**($isAdmin == 1)
10. <h1>Admin Iqbal</h1>
11. @**elseif** ($isEditor == 2)
12. <h1>Editor Iqbal</h1>
13. @**elseif** ($isAuthor == 3)
14. <h1>Author Iqbal</h1>
15. @**elseif** ($isContributor == 4)
16. <h1>Contributor Iqbal</h1>
17. @**else**
18. <h1>User Iqbal</h1>
19. @**endif**
21. @can('isAdmin')
22. <div **class**="btn btn-success btn-lg">
23. You have Admin Access
24. </div>
25. e
26. <div **class**="btn btn-info btn-lg">
27. You have User Access
28. </div>
29. can

jwMb ev †iwRóªvi Kivi ci wWdì ‡Wk‡evW© BDAviGj cwieZ©b Ki‡Z n‡j Ap/Provider/AuthRouthProvider.php

**public** **const** HOME = '/user/dashboard';

    public const ADMIN = '/admin/dashboard';

    public const EDITOR = '/editor/dashboard';

    public const AUTHOR = '/author/dashboard';

    public const CONTRIBUTOR = '/contributor/dashboard';

    public const HOME = '/user/dashboard';

Gevi Config/fortify.php wM‡q Gfv‡e wj‡L w`‡Z n‡e Zvn‡j jwMb Kivi ci BDRvi Abyhvqx wb‡Ri †c‡R P‡j hv‡e

    //'home' => RouteServiceProvider::HOME,

    'home' => function () {

        $user\_type = Auth::user()->user\_type;

           if ($user\_type == 1) {

                return RouteServiceProvider::ADMIN;

            }

            elseif ($user\_type == 2) {

                return RouteServiceProvider::EDITOR;

            }

           elseif ($user\_type == 3) {

                return RouteServiceProvider::AUTHOR;

            }

           elseif ($user\_type == 4) {

                return RouteServiceProvider::CONTRIBUTOR;

            }

            else {

                return RouteServiceProvider::HOME;

            }

    },