Authentication and Authorization in Laravel

Before Proceeding to implementation, let us understand difference between these terminologies.

In **Authentication**, the Web application identifies users via credential they provide. If it finds that the credentials are valid, user will be authenticated and given access to the functions which are allowed. In case of incorrect credentials, the access will be denied.

In **Authorization,** the Web application checks if the authenticated user can access the resources/views that they are being accesses. In other words, authorization checks rights and permissions over requested resources.

There are two ways to implement Authentication,

- Laravel's built-in authentication
- Manual authentication

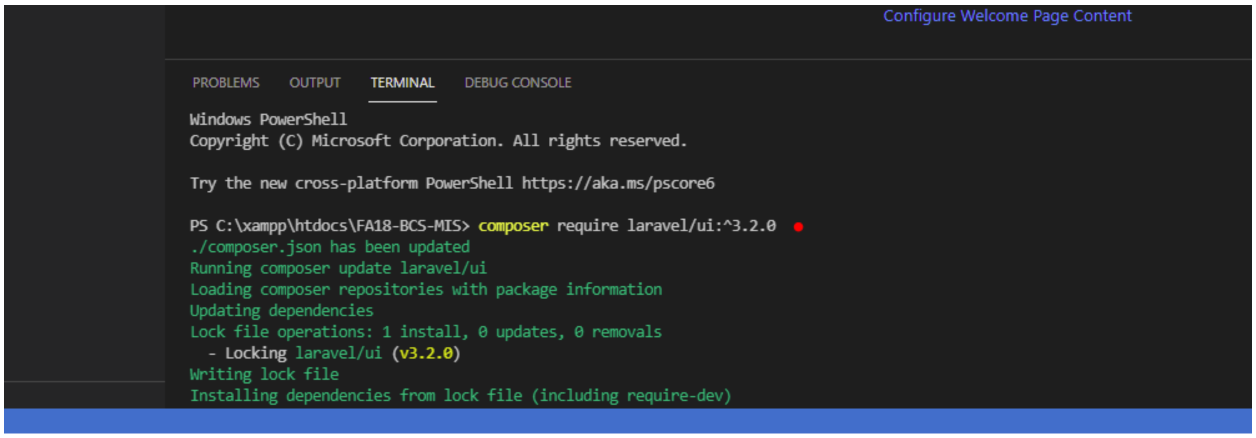
Let us proceed towards implementing Laravel's built-in Authentication

Authentication

First, install Laravel/ui package that provide a quick way to scaffold all the necessary routes and views you need for authentication using a few simple commands.

Note: Below commands will work on project that you build from scratch, commands will not work on downloaded or copy paste projects. For Successful implementation, you need to maintain Sequence of command as written below.

```
composer require laravel/ui:^3.2.0
```

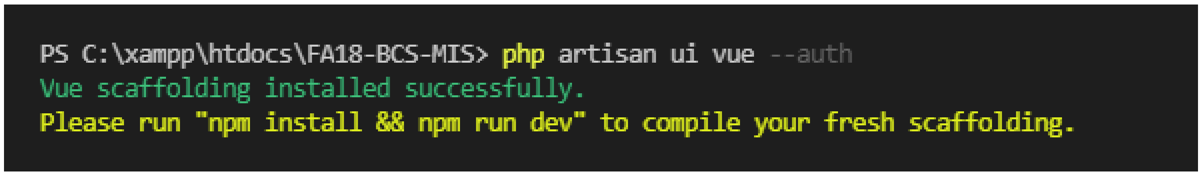


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\xampp\htdocs\FA18-BCS-MIS> composer require laravel/ui:^3.2.0
./composer.json has been updated
Running composer update laravel/ui
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking laravel/ui (v3.2.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
```

Once it's completed then run below command

php artisan ui vue –auth

or you can also use bootstrap ui using php artisan ui bootstrap --auth



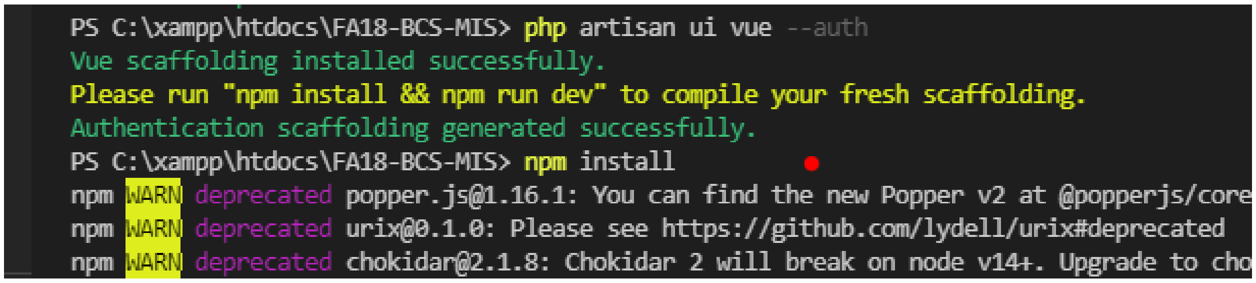
```
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan ui vue --auth
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
```

After successful completion of these commands, suggestion found like,

Please run "npm install && npm run dev" to compile your fresh scaffolding.

So download node.js from [Link]() and install it, then run command

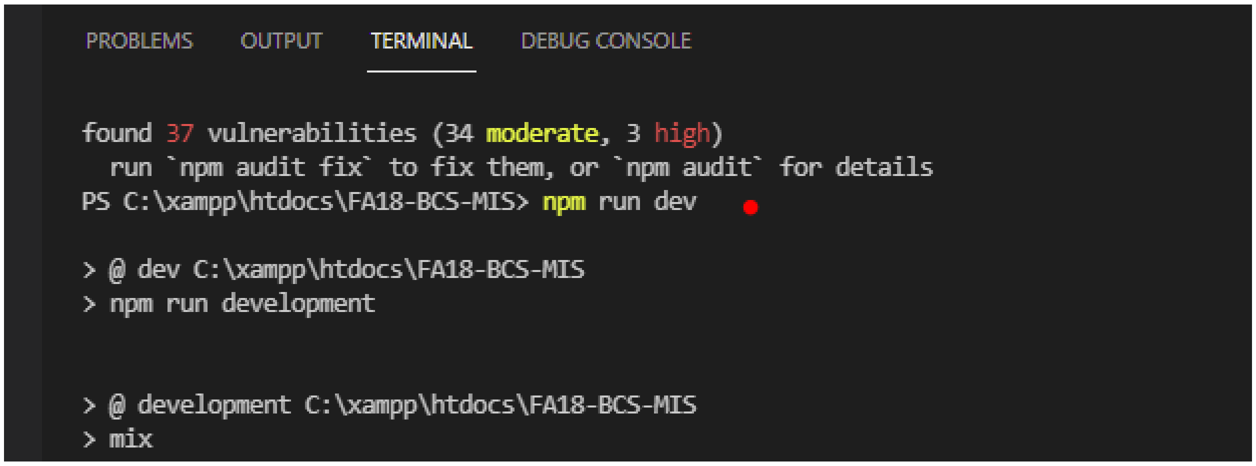
npm install



```
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan ui vue --auth
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
PS C:\xampp\htdocs\FA18-BCS-MIS> npm install
npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at @popperjs/core
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to cho
```

Then run this command

npm run dev

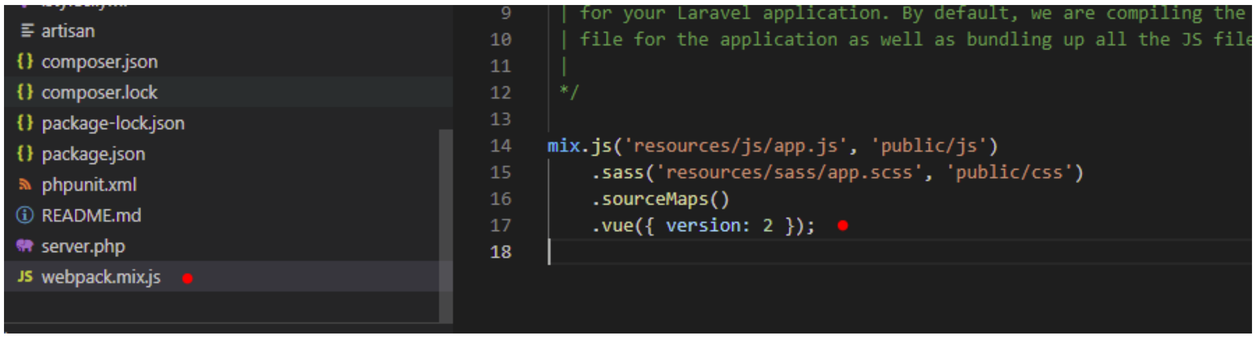


```
found 37 vulnerabilities (34 moderate, 3 high)
  run `npm audit fix` to fix them, or `npm audit` for details
PS C:\xampp\htdocs\FA18-BCS-MIS> npm run dev

> @ dev C:\xampp\htdocs\FA18-BCS-MIS
> npm run development


> @ development C:\xampp\htdocs\FA18-BCS-MIS
> mix
```

If npm run dev gives error, then make following changes in webpack.mix.js in root directory

Add this line in webpack.mix.js if you are getting error while running npm run dev

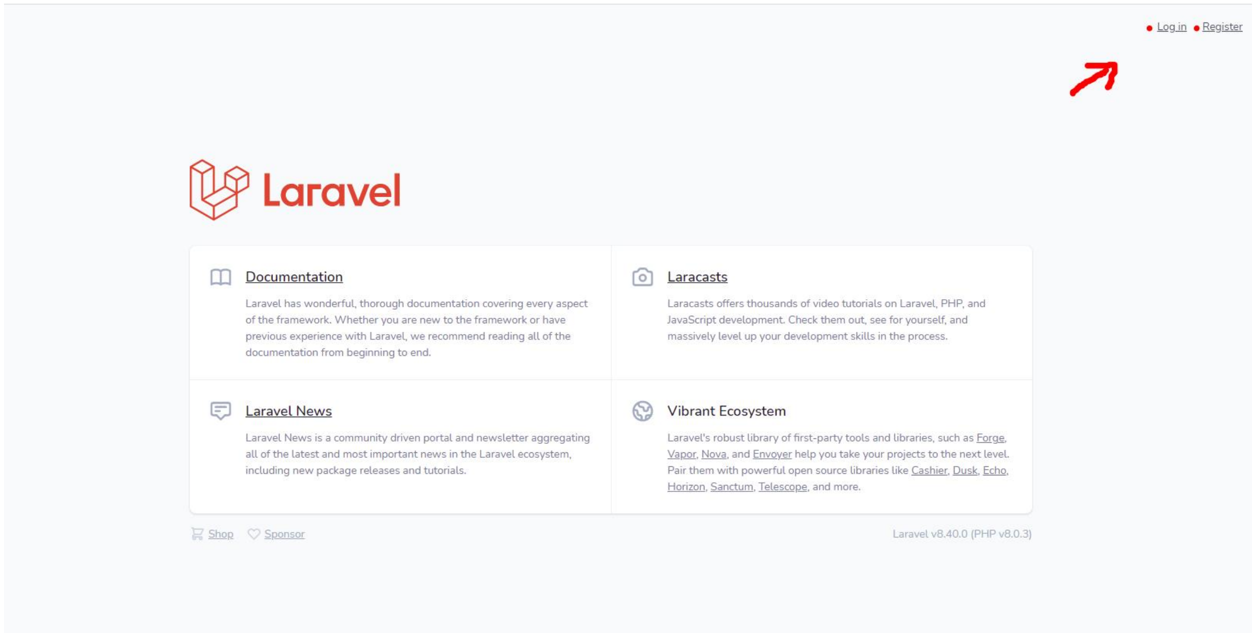
```
.vue({ version: 2 })
```



```
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
README.md
server.php
webpack.mix.js
```

```
 | for your Laravel application. By default, we are compiling the
 | file for the application as well as bundling up all the JS file
 |
 */

mix.js('resources/js/app.js', 'public/js')
    .sass('resources/sass/app.scss', 'public/css')
    .sourceMaps()
    .vue({ version: 2 });
```

Now when you run php artisan serve and open link [localhost]() you can view two buttons on welcome page



Log in Register

Laravel

Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

Shop Sponsor

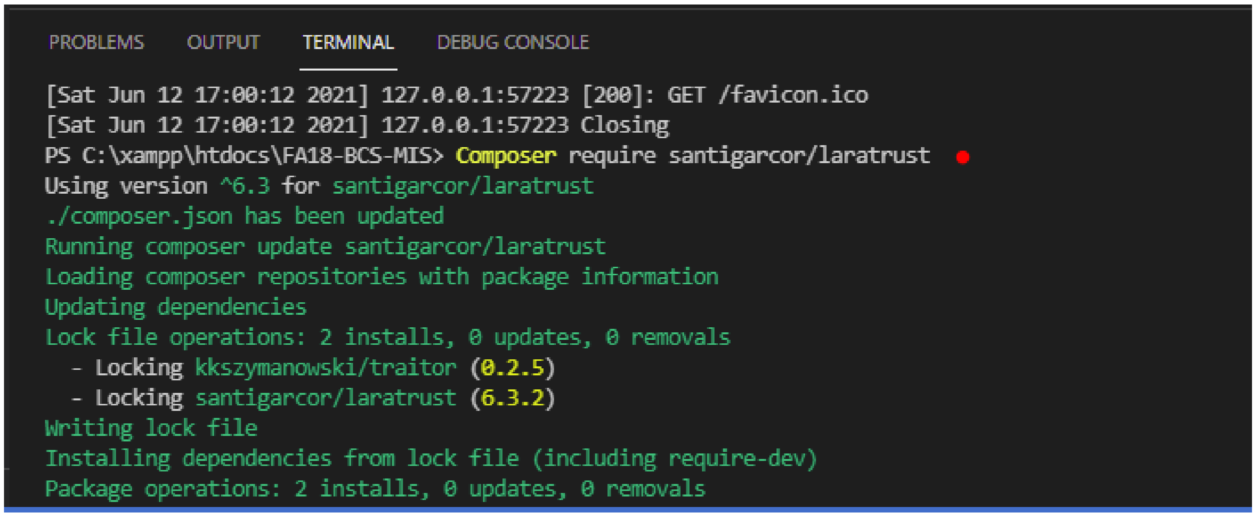
Laravel v8.40.0 (PHP v8.0.3)

Authorization

Here, if a user wants to access some resources, authorization will be performed to check if this user is eligible to access the resource.

We use [Laratrust]() for authorization.

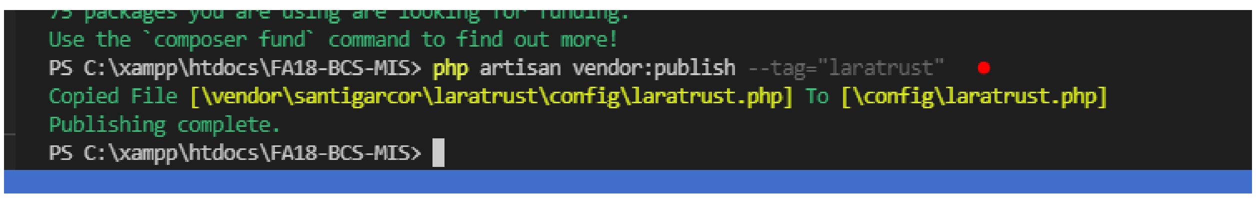
Composer require santigarcor/laratrust



```
[Sat Jun 12 17:00:12 2021] 127.0.0.1:57223 [200]: GET /favicon.ico
[Sat Jun 12 17:00:12 2021] 127.0.0.1:57223 Closing
PS C:\xampp\htdocs\FA18-BCS-MIS> Composer require santigarcor/laratrust
Using version ^6.3 for santigarcor/laratrust
./composer.json has been updated
Running composer update santigarcor/laratrust
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
  - Locking kkszymanowski/traitor (0.2.5)
  - Locking santigarcor/laratrust (6.3.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
```

Once it’s completed then run below command

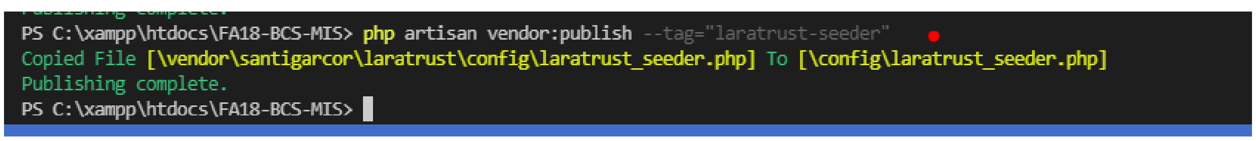
php artisan vendor:publish --tag="laratrust"



```
75 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan vendor:publish --tag="laratrust"
Copied File [\vendor\santigarcor\laratrust\config\laratrust.php] To [\config\laratrust.php]
Publishing complete.
PS C:\xampp\htdocs\FA18-BCS-MIS>
```

then run this command

php artisan vendor:publish --tag="laratrust-seeder"

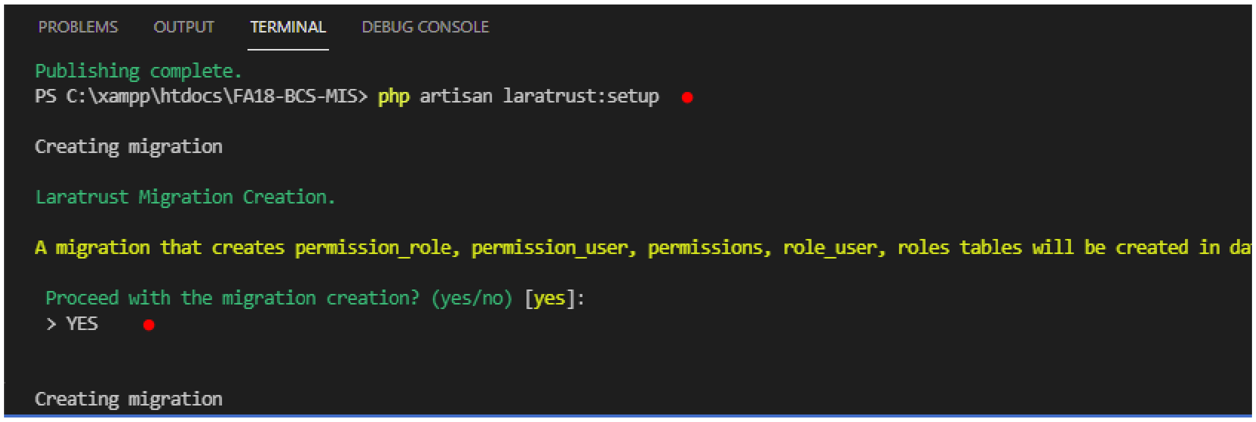


```
Publishing complete.
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan vendor:publish --tag="laratrust-seeder"
Copied File [\vendor\santigarcor\laratrust\config\laratrust_seeder.php] To [\config\laratrust_seeder.php]
Publishing complete.
PS C:\xampp\htdocs\FA18-BCS-MIS>
```

It will publish all configuration files. If this command does not work then run php artisan config: clear

Now run setup command. This will generate migrations of tables used for authorization, role, and permission model.

php artisan laratrust:setup

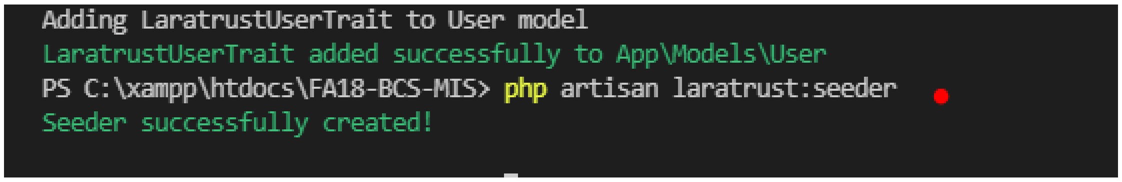


```
Publishing complete.
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan laratrust:setup

Creating migration

Laratrust Migration Creation.

A migration that creates permission_role, permission_user, permissions, role_user, roles tables will be created in da

 Proceed with the migration creation? (yes/no) [yes]:
 > YES


Creating migration
```

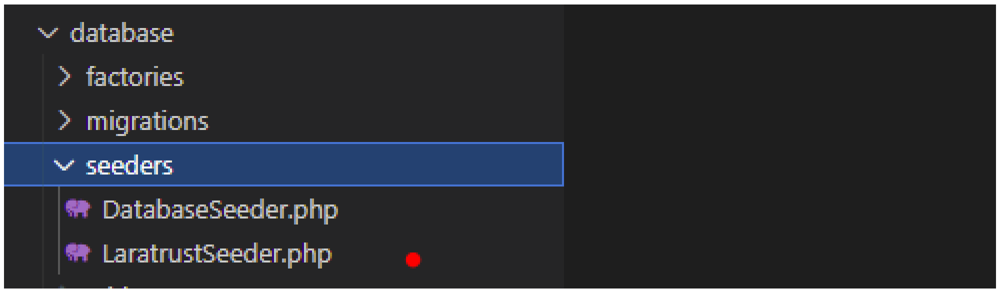
Once it has completed then you can view role and permission model, LaratrustSeeder and migrations in project.

sometimes command run successfully but we cannot see Laratrustseeder inside Seeds. To resolve this, we need run this command

php artisan laratrust:seeder

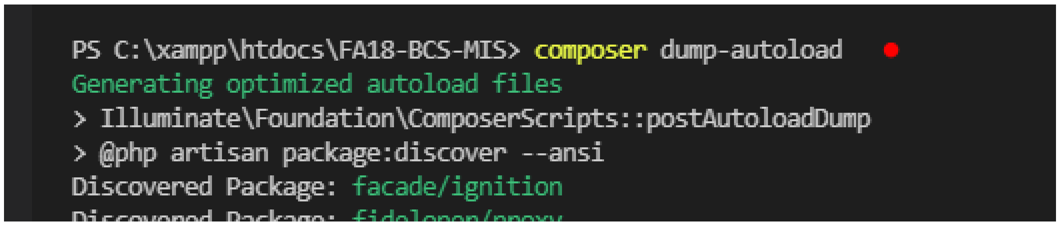


```
Adding LaratrustUserTrait to User model
LaratrustUserTrait added successfully to App\Models\User
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan laratrust:seeder
Seeder successfully created!
```



Now dump autoload with below command.

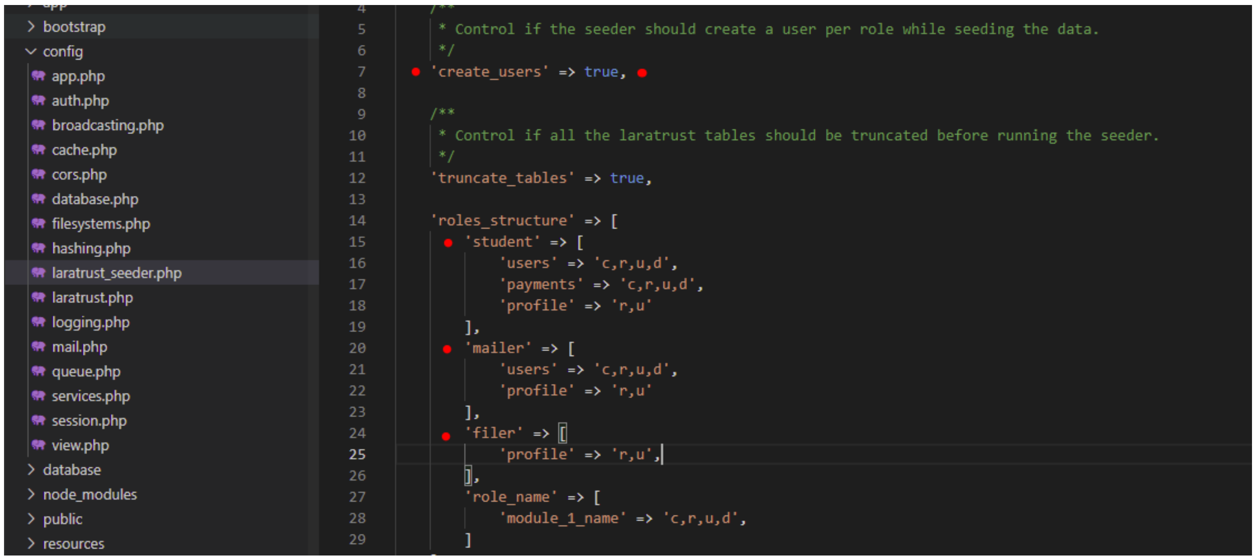
composer dump-autoload



```
PS C:\xampp\htdocs\FA18-BCS-MIS> composer dump-autoload
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
```

Now check Laratrust_Seeder file inside config folder.You can edit and create roles.

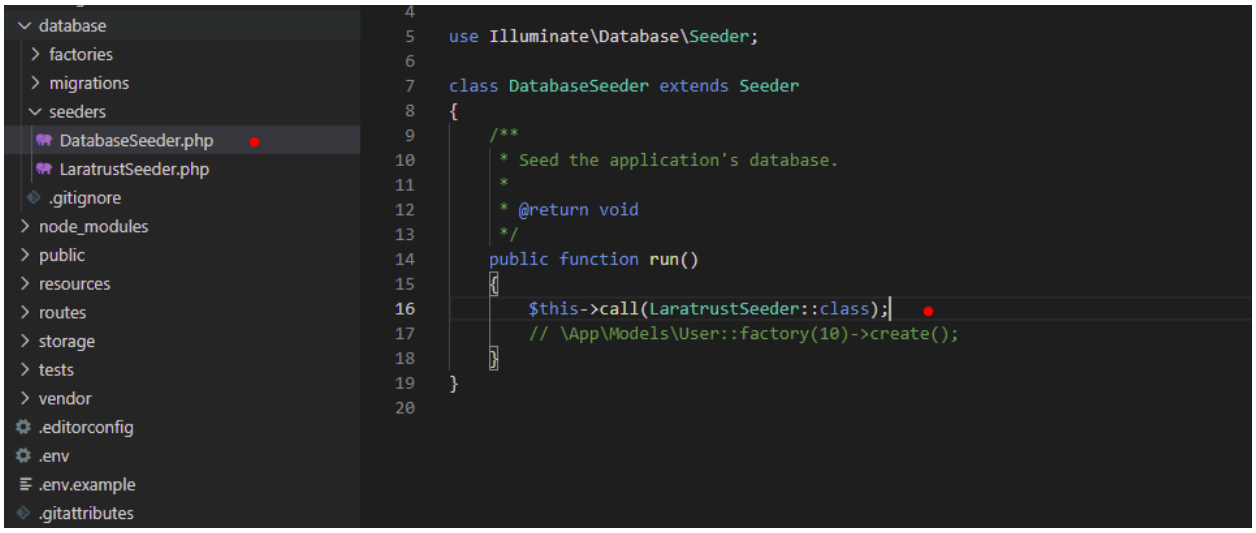
And make sure modify this ‘create_users’ to true if it is false.



```
bootstrap
config
  app.php
  auth.php
  broadcasting.php
  cache.php
  cors.php
  database.php
  filesystems.php
  hashing.php
  laratrust_seeder.php
  laratrust.php
  logging.php
  mail.php
  queue.php
  services.php
  session.php
  view.php
database
node_modules
public
resources
```

```
    /**
     * Control if the seeder should create a user per role while seeding the data.
     */
    'create_users' => true,

    /**
     * Control if all the laratrust tables should be truncated before running the seeder.
     */
    'truncate_tables' => true,

    'roles_structure' => [
        'student' => [
            'users' => 'c,r,u,d',
            'payments' => 'c,r,u,d',
            'profile' => 'r,u'
        ],
        'mailer' => [
            'users' => 'c,r,u,d',
            'profile' => 'r,u'
        ],
        'filer' => [
            'profile' => 'r,u',
        ],
        'role_name' => [
            'module_1_name' => 'c,r,u,d',
        ]
```

Now add this line in database >> seeder >> DatabaseSeeder.php >> inside run function.

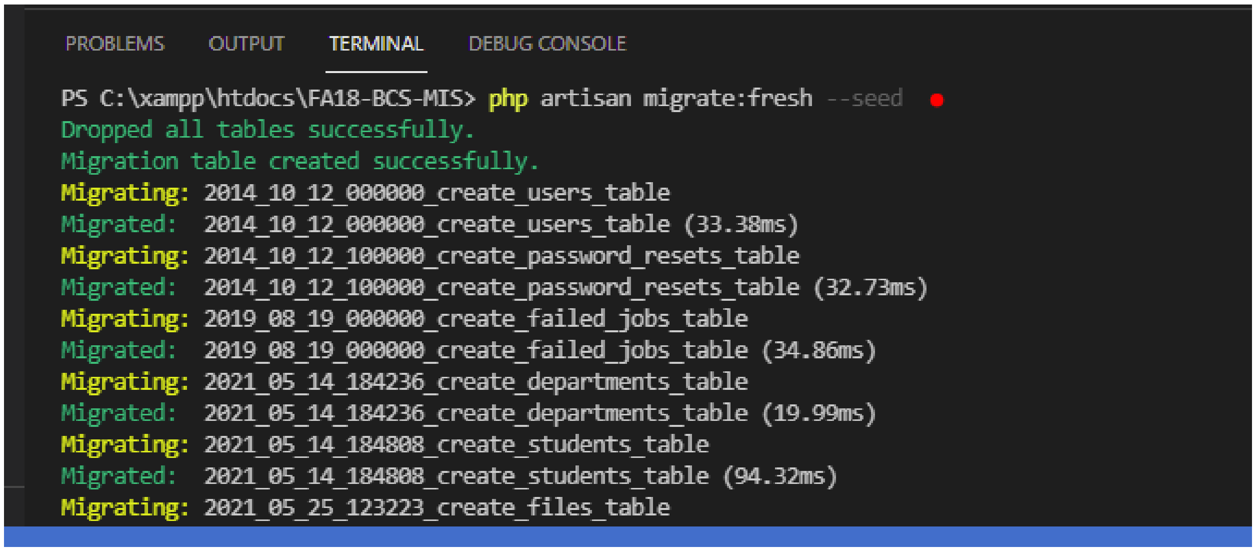


```
database
  factories
  migrations
  seeders
    DatabaseSeeder.php
    LaratrustSeeder.php
  .gitignore
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
```

```
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call(LaratrustSeeder::class);
        // \App\Models\User::factory(10)->create();
    }
}
```

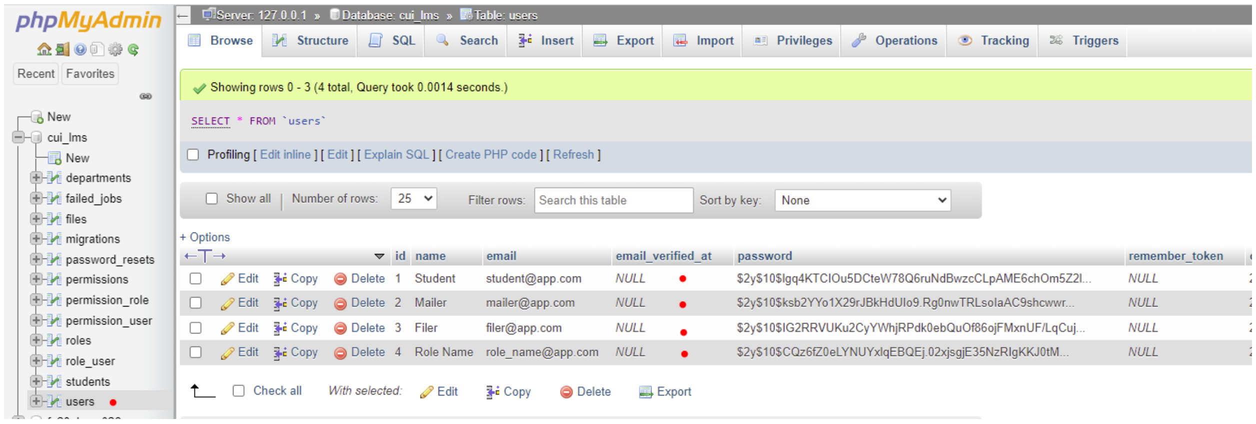
Now run the following command in terminal and make sure your xamp server in on.

```
php artisan migrate:fresh –seed
```



```
PS C:\xampp\htdocs\FA18-BCS-MIS> php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (33.38ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (32.73ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (34.86ms)
Migrating: 2021_05_14_184236_create_departments_table
Migrated:  2021_05_14_184236_create_departments_table (19.99ms)
Migrating: 2021_05_14_184808_create_students_table
Migrated:  2021_05_14_184808_create_students_table (94.32ms)
Migrating: 2021_05_25_123223_create_files_table
```

Now all roles and permission granted to users. Some login added in user table inside database due to seeding.

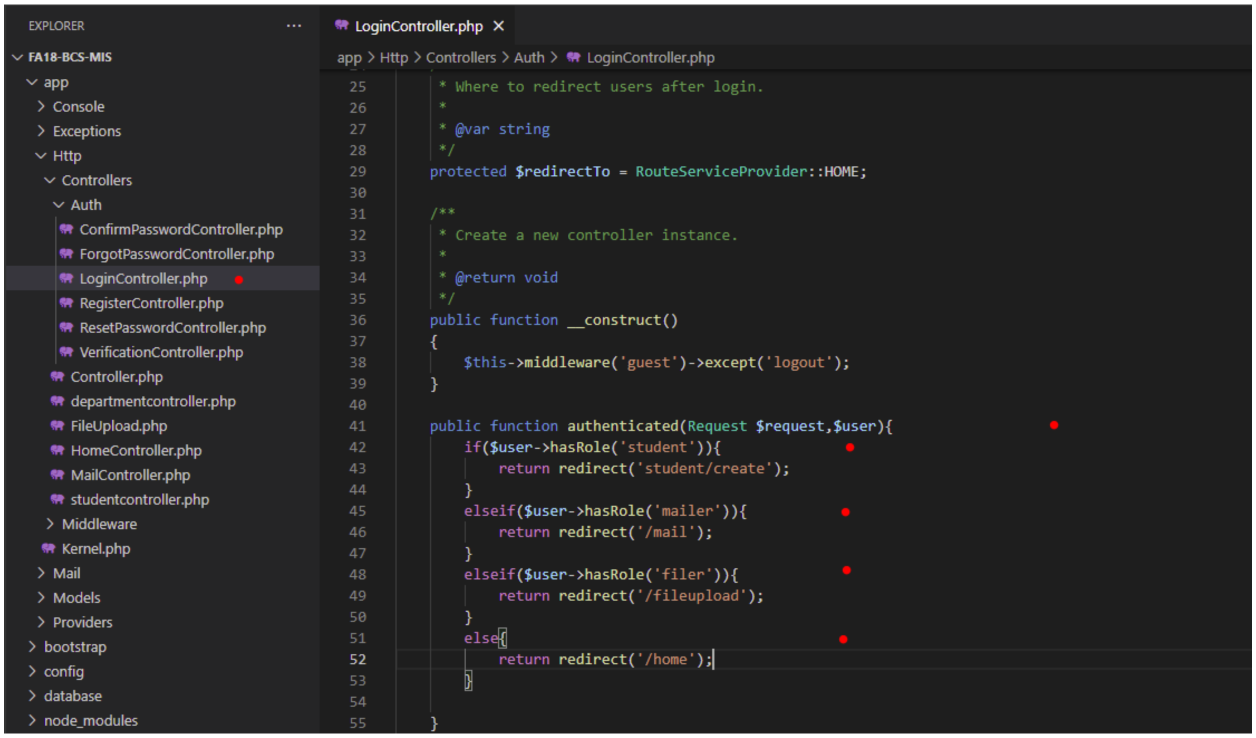


Showing rows 0 - 3 (4 total, Query took 0.0014 seconds.)

```
SELECT * FROM `users`
```

id	name	email	email_verified_at	password	remember_token
1	Student	student@app.com	NULL	$2y$10$lgq4KTClOu5DCteW78Q6ruNdBwzcCLpAME6chOm5Z2l...	NULL
2	Mailer	mailer@app.com	NULL	$2y$10$ksb2YYo1X29rJBkHdUlo9.Rg0nwTRLsolaAC9shcwwr...	NULL
3	Filer	filer@app.com	NULL	$2y$10$IG2RRVUKu2CyYWhjRPdk0ebQuOf86ojFMxnUF/LqCuj...	NULL
4	Role Name	role_name@app.com	NULL	$2y$10$CQz6fZ0eLYNUYxlqEBQEj.02xjsgjE35NzRlgKKJ0tM...	NULL

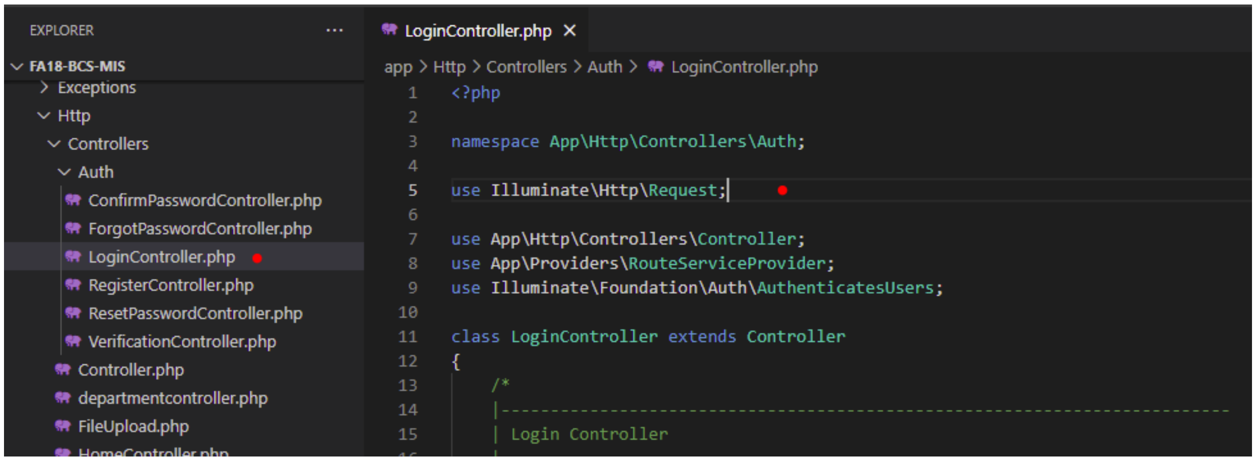
Now I want when student user login so it will redirect to student view. So inside Login controller write function. And make sure you have these routes in your project folder



```php
     * 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');
    }

    public function authenticated(Request $request,$user){
        if($user->hasRole('student')){
            return redirect('student/create');
        }
        elseif($user->hasRole('mailer')){
            return redirect('/mail');
        }
        elseif($user->hasRole('filer')){
            return redirect('/fileupload');
        }
        else{
            return redirect('/home');
        }

    }
```

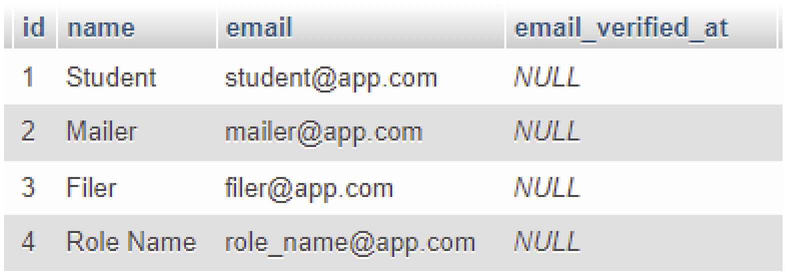
And import request in logincontroller

use Illuminate\Http\Request;

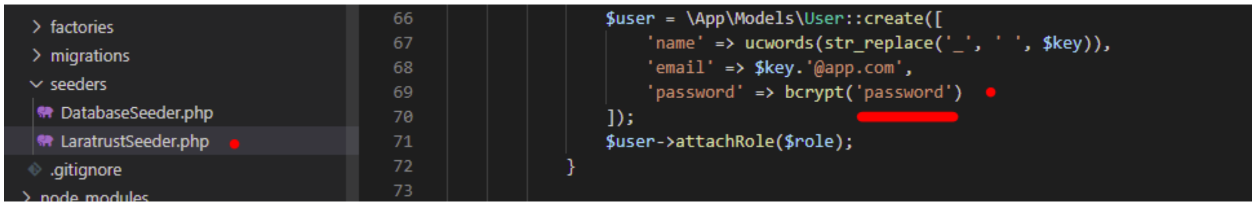


```php
<?php

namespace App\Http\Controllers\Auth;

use Illuminate\Http\Request;

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

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
```

In our case the default id/password of users which we generated in seeder file is



id	name	email	email_verified_at
1	Student	student@app.com	NULL
2	Mailer	mailer@app.com	NULL
3	Filer	filer@app.com	NULL
4	Role Name	role_name@app.com	NULL



```
            $user = \App\Models\User::create([
                'name' => ucwords(str_replace('_', ' ', $key)),
                'email' => $key.'@app.com',
                'password' => bcrypt('password')
            ]);
            $user->attachRole($role);
        }
```

For student

User id   : student@app.com

Password: password

For mailer

User id   : mailer@app.com

Password: password

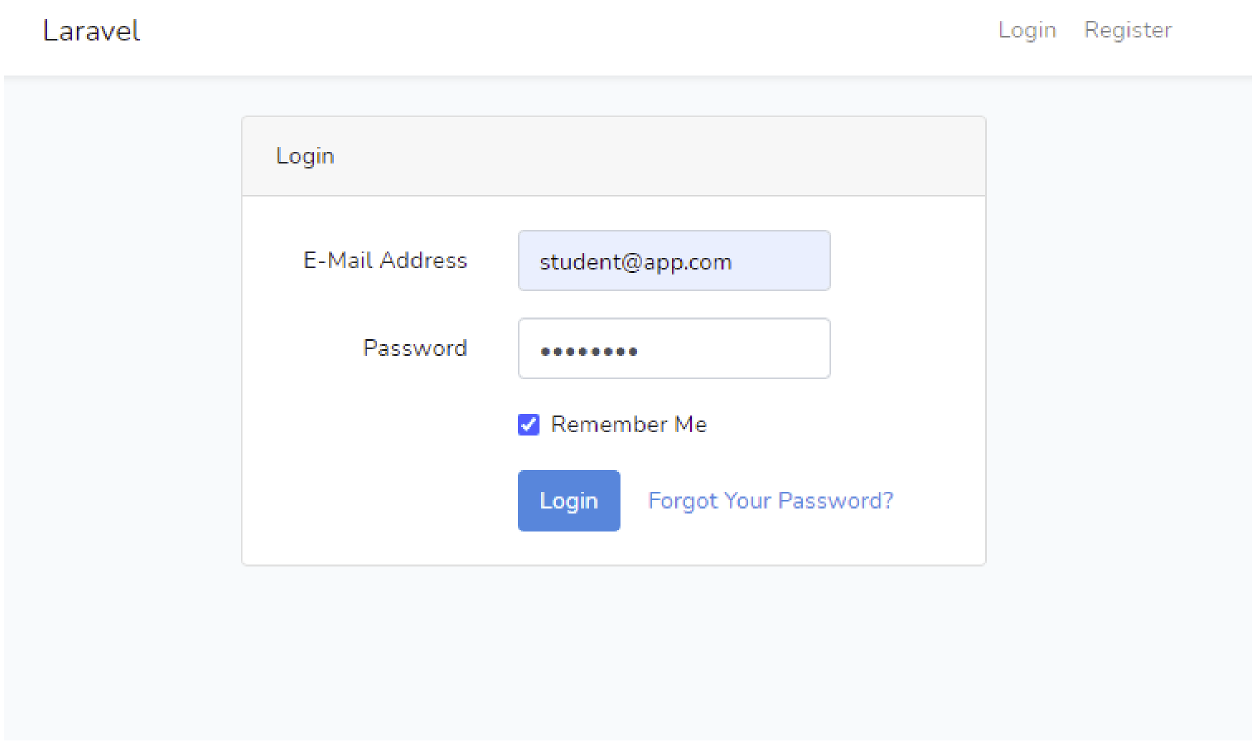
For filer

User id   : filer@app.com

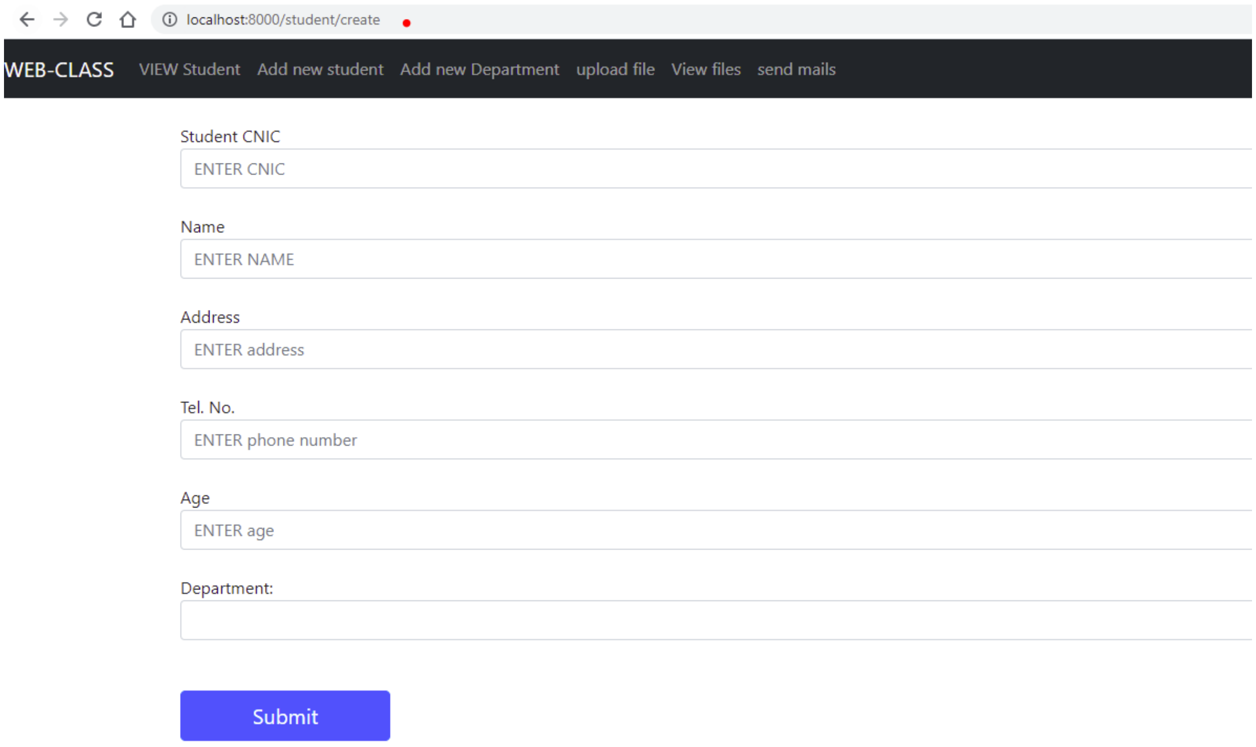
Password: password

Now let us try logging in.

Run the server using php artisan serve and go to the login page

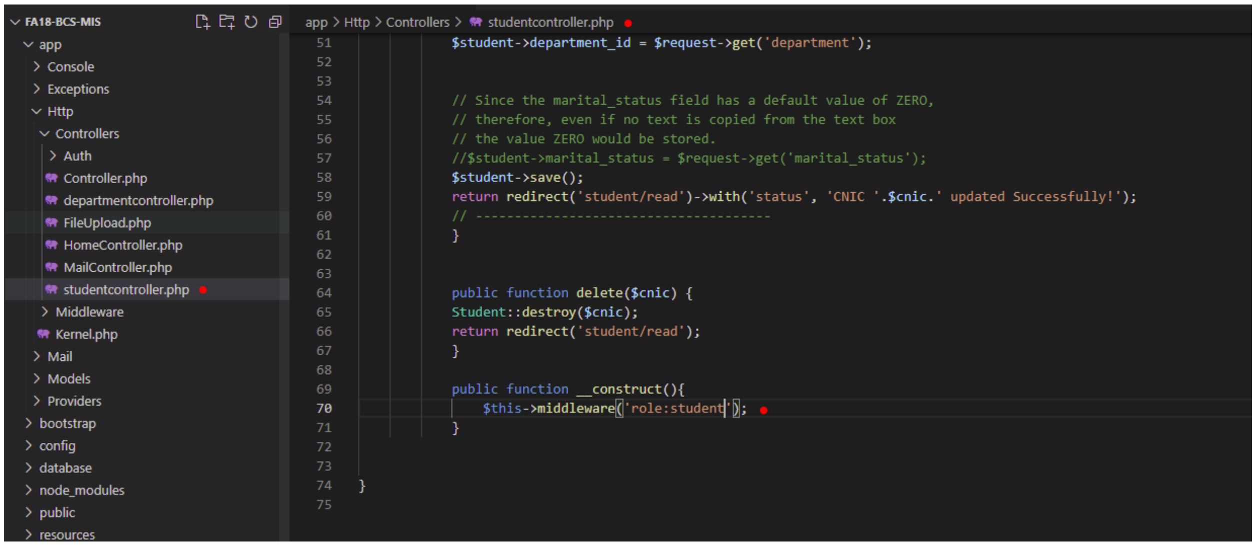


After login it will redirect us toward student/create route.



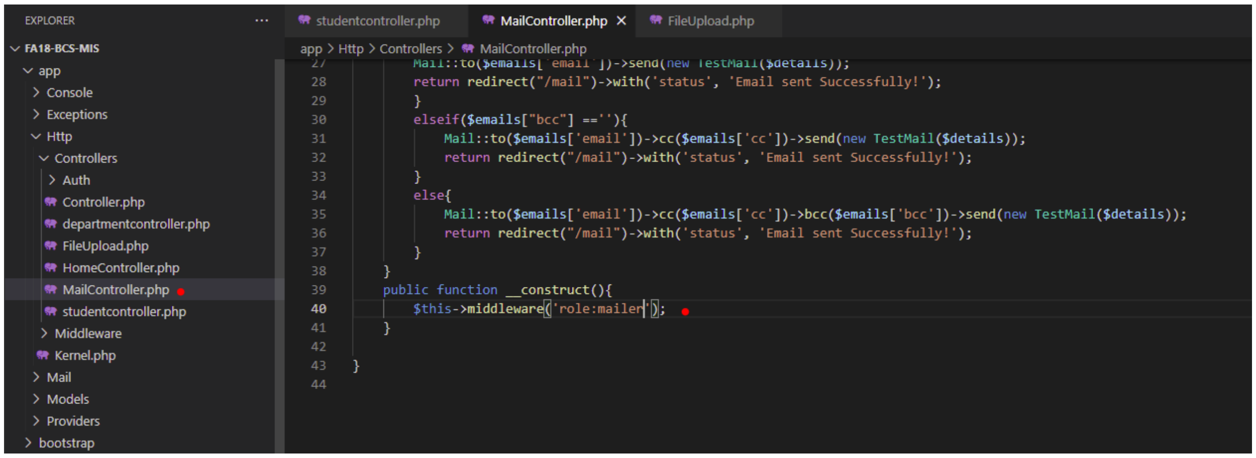
So, when I will login for mailer then I will be redirected toward mail view.

However now every role can access every page available in our website if we want to restrict webpages based on roles, we have to add following code in controllers.

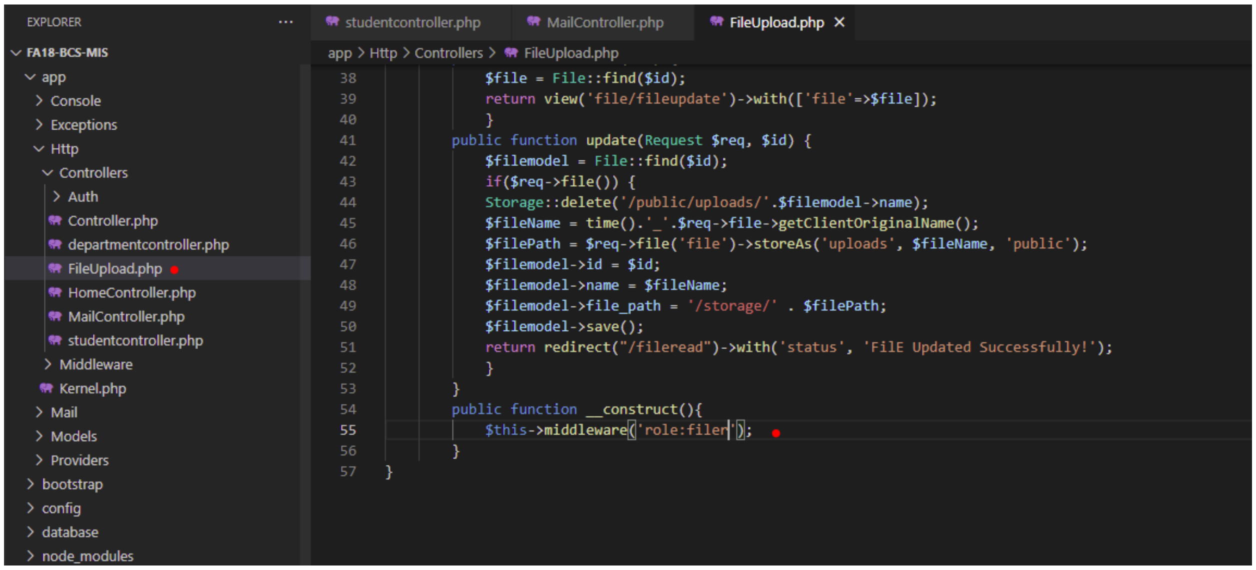


```
            $student->department_id = $request->get('department');


            // Since the marital_status field has a default value of ZERO,
            // therefore, even if no text is copied from the text box
            // the value ZERO would be stored.
            //$student->marital_status = $request->get('marital_status');
            $student->save();
            return redirect('student/read')->with('status', 'CNIC '.$cnic.' updated Successfully!');
            // ------------------------------------
            }


            public function delete($cnic) {
            Student::destroy($cnic);
            return redirect('student/read');
            }

            public function __construct(){
                $this->middleware('role:student');
            }


}
```

Same for other controllers

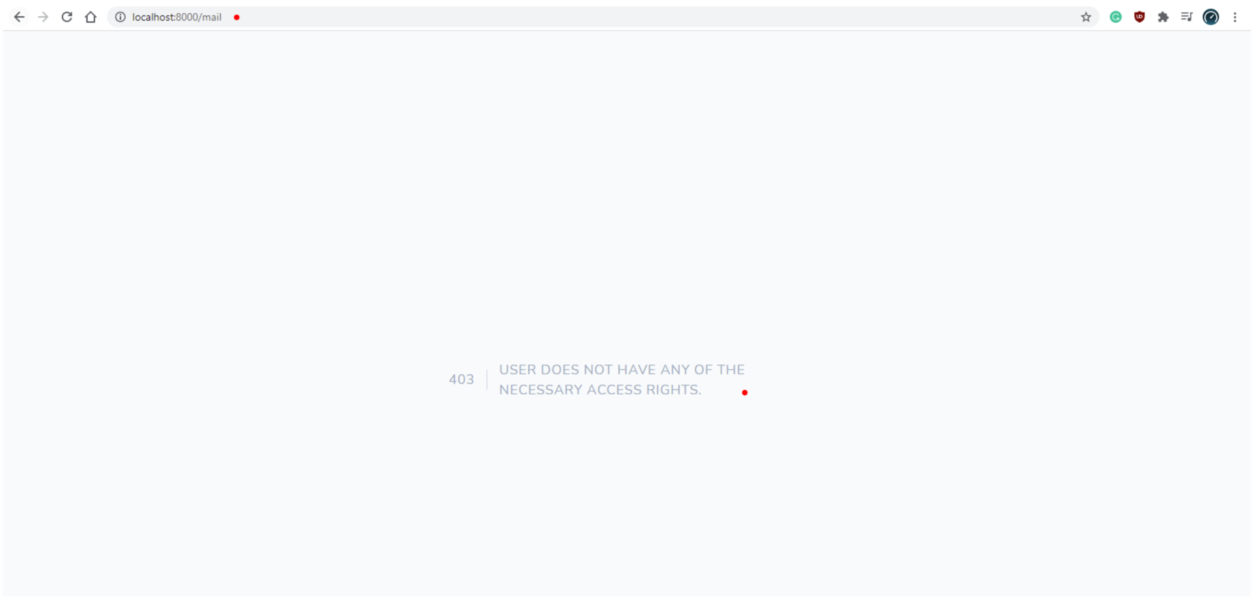


```
        Mail::to($emails['email'])->send(new TestMail($details));
        return redirect("/mail")->with('status', 'Email sent Successfully!');
        }
        elseif($emails["bcc"] ==''){
            Mail::to($emails['email'])->cc($emails['cc'])->send(new TestMail($details));
            return redirect("/mail")->with('status', 'Email sent Successfully!');
        }
        else{
            Mail::to($emails['email'])->cc($emails['cc'])->bcc($emails['bcc'])->send(new TestMail($details));
            return redirect("/mail")->with('status', 'Email sent Successfully!');
        }
    }
    public function __construct(){
        $this->middleware('role:mailer');
    }

}
```



```php
            $file = File::find($id);
            return view('file/fileupdate')->with(['file'=>$file]);
            }
        public function update(Request $req, $id) {
            $filemodel = File::find($id);
            if($req->file()) {
            Storage::delete('/public/uploads/'.$filemodel->name);
            $fileName = time().'_'.$req->file->getClientOriginalName();
            $filePath = $req->file('file')->storeAs('uploads', $fileName, 'public');
            $filemodel->id = $id;
            $filemodel->name = $fileName;
            $filemodel->file_path = '/storage/' . $filePath;
            $filemodel->save();
            return redirect("/fileread")->with('status', 'FilE Updated Successfully!');
            }
        }
        public function __construct(){
            $this->middleware('role:filer');
        }
}
```

Now if student try to access mailer role it will give an error.



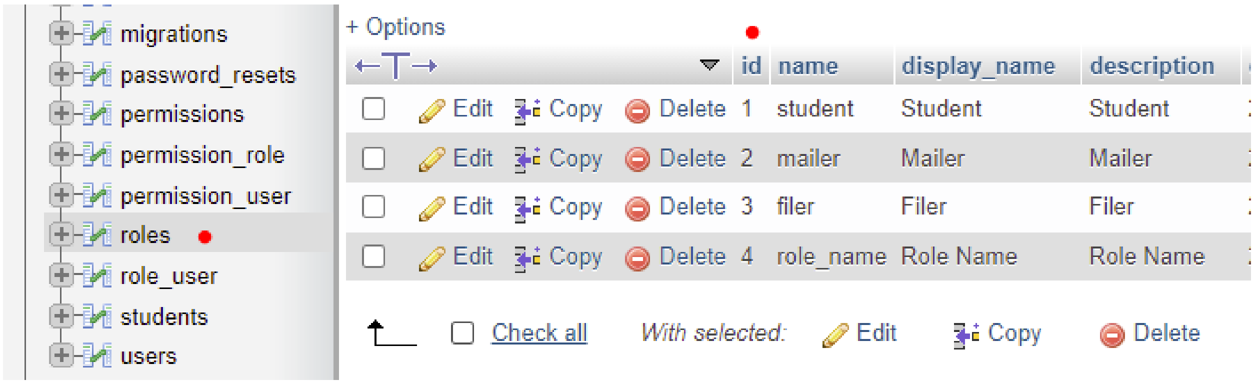
```
403 | USER DOES NOT HAVE ANY OF THE
      NECESSARY ACCESS RIGHTS.
```

The student is not authorized to access this view.

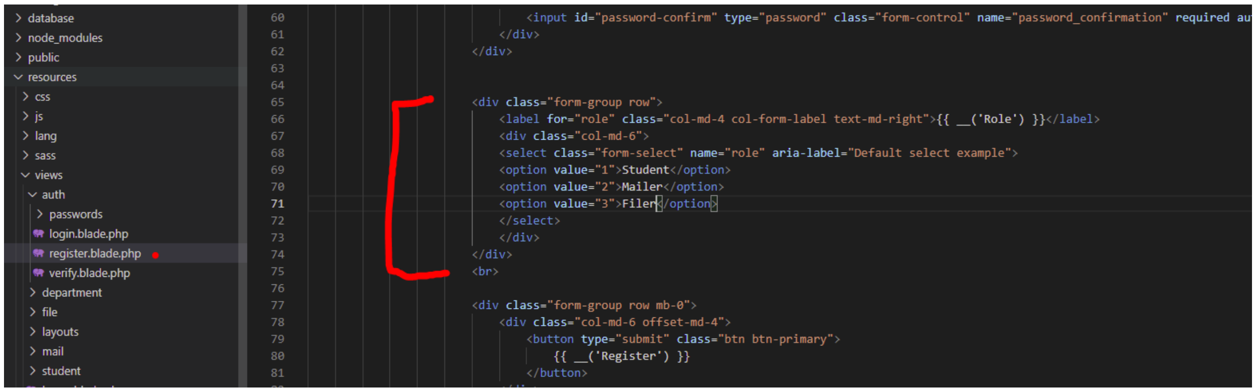
Adding roles field in register page:

add the following lines in resources>>views>>auth>>register.

Here we have used html select tag with role ids. You can check your role ids in role table of your database.

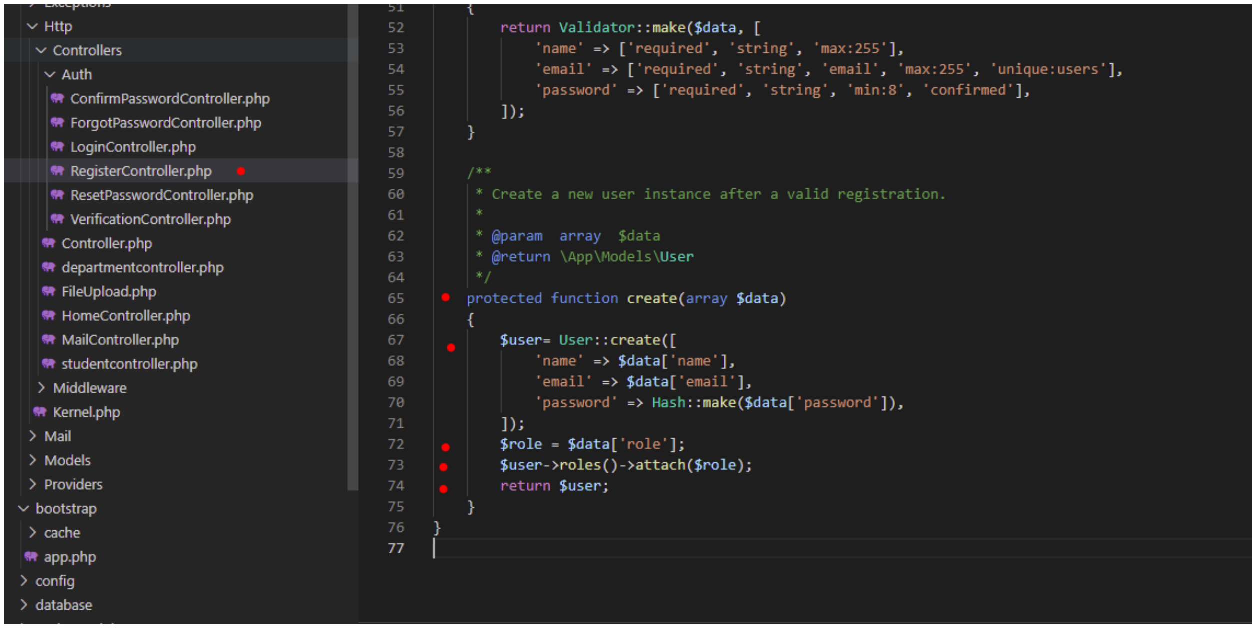


	id	name	display_name	description
Edit Copy Delete	1	student	Student	Student
Edit Copy Delete	2	mailer	Mailer	Mailer
Edit Copy Delete	3	filer	Filer	Filer
Edit Copy Delete	4	role_name	Role Name	Role Name



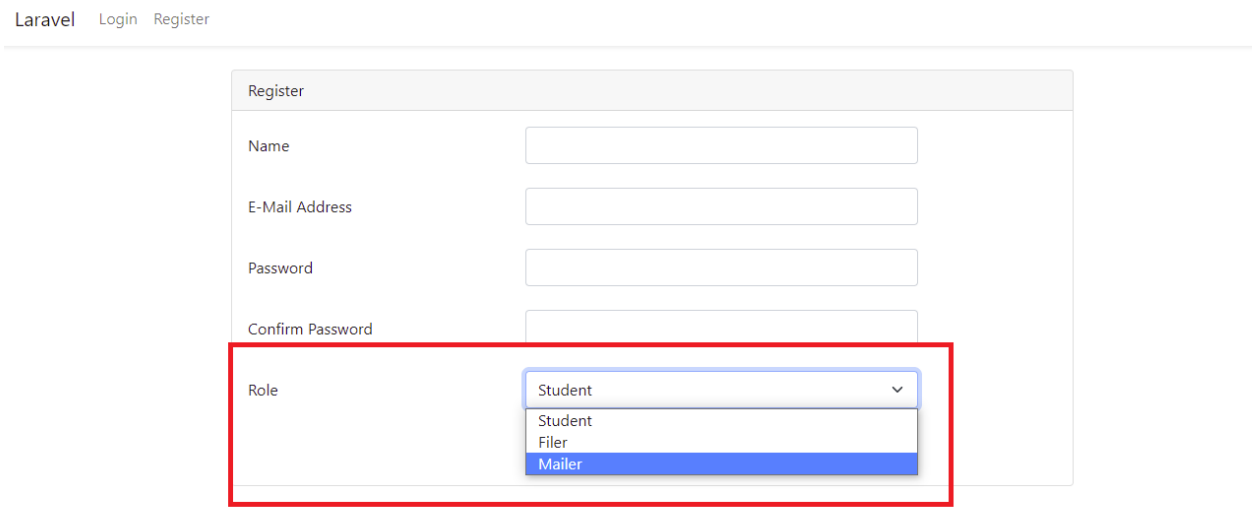
```
            <input id="password-confirm" type="password" class="form-control" name="password_confirmation" required au
        </div>
    </div>


    <div class="form-group row">
        <label for="role" class="col-md-4 col-form-label text-md-right">{{ __('Role') }}</label>
        <div class="col-md-6">
        <select class="form-select" name="role" aria-label="Default select example">
        <option value="1">Student</option>
        <option value="2">Mailer</option>
        <option value="3">Filer</option>
        </select>
        </div>
    </div>
    <br>

    <div class="form-group row mb-0">
        <div class="col-md-6 offset-md-4">
            <button type="submit" class="btn btn-primary">
                {{ __('Register') }}
            </button>
```

Then goto register controller and modify the following code.



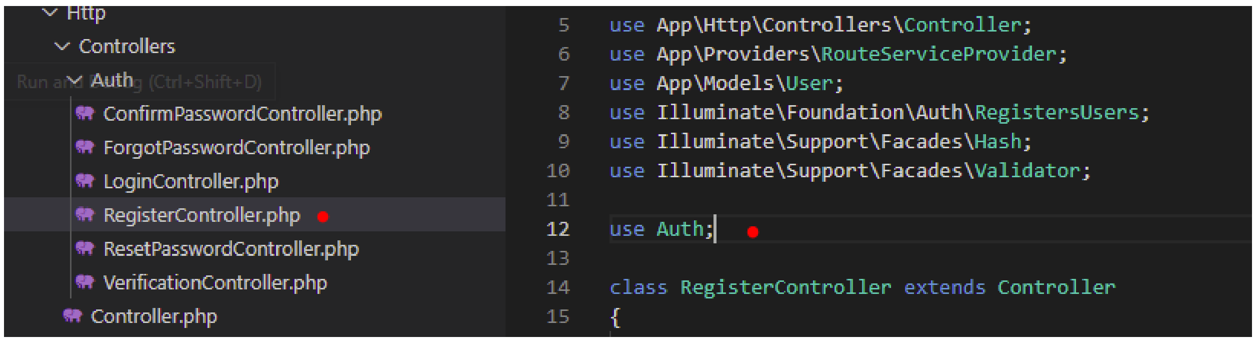
```
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }

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

Now you can register a user with role



REDIRECTION AFTER REGISTERATION BASED ON ROLE

Go into the register controller and import Auth;


Http
Controllers
Auth
ConfirmPasswordController.php
ForgotPasswordController.php
LoginController.php
RegisterController.php
ResetPasswordController.php
VerificationController.php
Controller.php

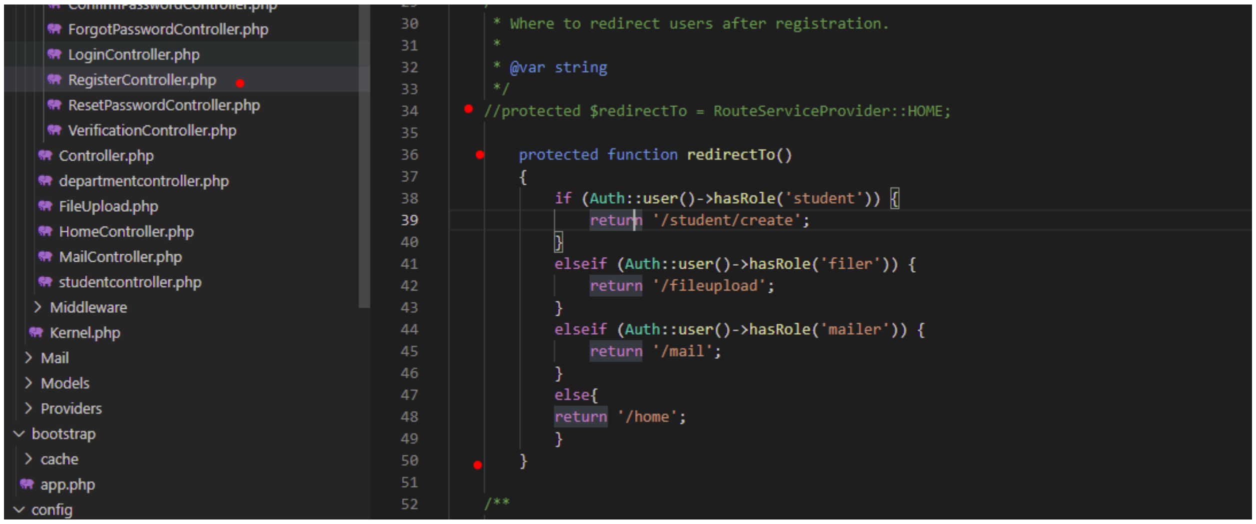
```
use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use App\Models\User;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

use Auth;

class RegisterController extends Controller
{
```

Then replace the following code

Replace this

```
protected $redirectTo = RouteServiceProvider::HOME;
```

With this



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

    protected function redirectTo()
    {
        if (Auth::user()->hasRole('student')) {
            return '/student/create';
        }
        elseif (Auth::user()->hasRole('filer')) {
            return '/fileupload';
        }
        elseif (Auth::user()->hasRole('mailer')) {
            return '/mail';
        }
        else{
        return '/home';
        }
    }

/**
```