**LARAVEL**

**Tools for Start Working:**

* **xampp**: For Apache & MySQL;
* **composer**: Php Package Manager**;**
* **Sequence**: First xampp then composer, composer depends on xampp php.
* **Php path location**: **E:/xampp/php/php.exe** ; specify path location for composer.
* **Code Editor.**

**Installation Command:**

* **Project\_install\_command**: composer create-project --prefer-dist laravel/laravel folder\_name -> enter; **(may take a few munites).**
* **If use space inside folder name use quotes: “laravel crud”**
* **Development server create**: **php artisan serve ->enter** ;
* **Directory\_change**: **E:/xampp/php cd abc;// set into abc;**

**MVC (MODEL VIEW CONTROLLER):**

* **Model**: Data Structure.
* **View**: Visible Part of Application.
* **Controller**: Logics of Application.
* **Three Part of Application:**
  + Data & Data Structure of Application.
  + Logics of Application.
  + Visible Part of Application.
* **Data Insert:** 
  + View -> Controller -> Model -> Database;
  + Controller like a bridge connection with view and model;
* **Data Select:** 
  + Database -> Model -> Controller -> View;

**Essential PHP Artisan command:**

**Most Important:**

* **php artisan serve**: Run Laravel development Server.
* **php artisan serve –port=4000**: Development Server in Different Port.
* **php artisan make:Controller ControllerName** : Create Controller/Class.
* **php artisan make:controller** **resourceCrud** **--resource** **--model=crudModel**
* **php artisan make:Model ModelName** : Create Model.
* **php artisan make:Middleware MiddlewareName**: Create Middleware.
* **php artisan make:Migration migration\_name**: Create Migration/TableDB.
* **Php artisan make**:**migration blog\_table –create=blog Create with Table**
* **php artisan migrate**: Run Migration.
* **php artisan make:seeder seederName; location: Database\seeds😉**

**Less Important: (if code not worked or before deploying real server)**

* **php artisan cache:clear** : Clear Application Cache.
* **php artisan route:clear** : Clear Application Route Cache.
* **php artisan config:clear** : Clear Application Config Cache.
* **php artisan view:clear** : Clear Complied View Files.

**Multi Command together:**

* php artisan **make:model CrudModel** **-m/model & migration together.**
* **Automatically import use class in PHP STORM:** 
  + Select Alt+Enter (import);

**Routing (Set Post URL):**

**(Return a text By Coll Back Function).**

* **Route::get('/welcome', function(){return "Pro nazmul";});**

**(Return a view By Coll Back Function).**

* **Route::get('/', function(){return view(‘home\_page’);});**

**(Return a controller method in url).**

* **Route::get('/address', ‘controller\_name@method\_name’;});**
* **Standard Way**: Return view from controller & always route controller.

**(Route with href link).**

<a **href= “{{url(‘/’) }}”**>HOME </a>.

<a **href= “{{url(‘contract’) }}”**>/contract</a>.

**(Routing Parameter: Data Pass From Route To Controller).**

* Single Parameter: **Route::get('/address/{name}', ‘controller\_name@try’});**
* Multiple Parameter: **Route::get('/address/{name}/{age}', ‘controller @try’});**
* Routing Parameter name and method argument name should be same.

**(Routing Parameter: Data Pass From Route -> Controller -> View).**

* **Route::get('/address/{name}/{age}', ‘controller @try’});**
* **Public function try ($name,$age) {return view(‘page’,** **['name'=>$name, 'age'=>$age); }; //**Argument value pass to view by associative array **& echo in view {{$name}}**

**Create Route Group:**

**(Route::group(): Route group to use common features to all).**

**Route::group(['prefix'=>'admin'], function(){**

**Route::get('/registration', function(){**

**return "This is Admin Registration Page"; });**

**Route::get('/login', function(){**

**return "This is Admin login Page"; });**

**Route::get('/dashboard', function(){**

**return "This is Admin dashboard Page"; });**

**Route::get('/profile', function(){**

**return "This is Admin Profile Page"; });**

**});**

**Types of Laravel controller:**

* **Custom\_controller**: Controller create as you want.
* **Single\_action\_controller**: Controller creates only for specific job.
* **Resource\_controller**: Http request handles & CRUD operation.
* **Middleware\_controller**: Use inside request && Response.

**(Custom Controller).**

* **public function Anything(){};**
* **Route::get('/address', ‘controller\_name@Anything’;});**

**(Single Action Controller).**

* **public function \_\_invoke(){}; // Method execute automatically.**
* **Route::get('/address', ‘controller\_name’}); Need only Controller Name**.

**(Resource controller).**

* **Action**: Can Handle All types of http request.
* **Command**: php artisan make:Controller C\_name –resource (enter)
* **Route::resource('/address', ‘controller\_name’;}); only controller name**
* **Auto Generated Methods**:
  + **Index()**
  + **Create()**
  + **Store(**Request $request**)**
  + **Show(**$id**)**
  + **Edit(**$id**)**
  + **Update(**Request $request, $id**)**
  + **Destroy(**$id**)**

|  |  |  |  |
| --- | --- | --- | --- |
| Verb | URI | Action / Method | Route Name |
| GET | /name | Index() | name.index |
| GET | /name/create | Create() | name.create |
| GET | /name/id | Show() | name.show |
| GET | /name/id/edit | Edit() | name.edit |
| POST | /name | Store() | name.store |
| PUT/PATCH | /name/id | Update() | name.update |
| DELETE | /name/id | Destroy() | name.destroy |

**Blade Template:**

**Type PHP inside blade view**

**@php**

**Echo “PHP Code Will Goes Here”;**

**@endphp**

* **{{ Php echo goes here }}: Prevent XSS && Filter htmlspecialchar();**
* **{!! Php echo goes here !!}: Not Prevent XSS && no;**

**For Loop inside Blade**

* **Syntax :**

**@for($i=1; $i<20; $i++)**

**Code to be execute**

**@endfor**

**Foreach Loop inside Blade**

* **Syntax :**

**@foreach($keys as $values)**

**Code to be execute**

**@endforeach**

**Forelse Loop inside Blade**

* **Syntax :**

**@forelse($keys as $values)**

**Code to be execute**

**@empty**

**Code to be execute**

**@endforelse**

**Loop Special Properties**

**{{$loop -> count}}: Count Loop Numbers.**

**Syntax :**

**@foreach($keys as $values)**

**{{$loop -> count}}**

**@break**

**@endforeach**

**{{$loop -> index}}: Show array indexes.**

**Syntax :**

**@foreach($keys as $values)**

**{{$loop -> index}}**

**@endforeach**

**{{$loop -> odd, even, first, last}}: Show odd/even/first/last indexes.**

**Syntax :**

**@foreach($keys as $values)**

**@if($loop -> odd)**

**{{$values}}**

**@endif**

**@endforeach**

**Include a file in Blade PHP**

* **@include('templates.Header')**

**Data Pass By include:**

* **@include('templates.Header',['head'=> 'NazmulPro'])**

**includeIf Uses: If file not exist won’t generate error.**

* **@includeIf('templates.Header',['head'=> 'NazmulPro'])**

**includeFirst Uses: include first file.**

* **@includeFirst([‘template.header’, ‘template.footer’])**

**includeWhen Uses: include When first bol value is true.**

* **@includeFirst(true, 'templates.Header')**

**includeUnless Uses: include When first bol value is false.**

* **@includeUnless(false, 'templates.Header')**

Include Parent Layout to child in Blade PHP

* **@extends('layout.MasterLayout')**

**Data sent to parent:**

* **@section('key', 'value')**

**Data Received by parent:**

* **@yield('key')**

**Multiple Data sent to parent:**

**Syntax:**

**@section('key')**

**Child codes will go here**

**@endSection**

**@yield('key'): Receive in desired place same way.**

**Link CSS, JS, IMG, SASS, Fonts From public folder:**

* **href= “{{asset(‘css/custom.css’)}}”**
* **href= “{{asset(‘js/custom.js’)}}”**

**Parent Page Demo Layout:**

**Syntax:**

**<head>**

**<title>@yield('title')</title>**

**<link rel="stylesheet" type="text/css" href="{{asset('css/bootstrap.min.css')}} ">**

**</head>**

**<body>**

**@include('layout.Menu')**

**@yield('content')**

**@include('layout.Footer')**

**<script type="text/javascript" src="{{asset('js/jquery-3.4.1.min.js')}}"></script>**

**</body>**

**Child Page Demo Layout:**

**Syntax:**

**@extends('layout.App')**

**@section ('title','Service Page')**

**@section('content')**

**Pages Content will go here**

**@endsection**

**Form Validation & Hash Password**

**Hash Password:**

* **Required Class: use Illuminate\Support\Facades\Hash;**
* **Hash::make($request->input('password'));**
* **Hash::check** **('plain-text', $hashedPassword);**

**Laravel Migration:**

**Is a Process to Manage table from Laravel template**

* **Step 1: Create A Database;**
* **Step 2: Set Database information in .env file;**
* **Step 3:** **Command:**

**Two Methods in Migration:**

* + **Up method**: Uses for constructive works.
  + **Down method**: Uses For destructive Works. Works when Rollback;

**Table Create Syntax:**

**Schema::create('students\_info', function(Blueprint $coll){**

**$coll -> increments('id');**

**$coll -> string('name', 100);**

**$coll -> smallint('roll');**

**$coll -> string('email', 60);**

**$coll -> string('cell', 20);**

**$coll -> timestamps();**

**});**

**Operation column:**

* + **Syntax: Size Change:**

**Schema::table('students\_info', function(Blueprint $table){**

**$table -> string('name', 150) -> change();**

**});**

* + **Syntax: Column Rename:**

**Schema::table('students\_info', function(Blueprint $table){**

**$table -> renameColumn('name', ‘st\_name’);**

**});**

* + **Syntax: Column Drop:**

**Schema::table('students\_info', function(Blueprint $table){**

**$table -> dropColumn('name');// for single column;**

**$table -> dropColumn(['name', 'email']);// for multiple column;**

**});**

* **Update Operation Table:**
  + **Command: composer require doctrine/dbal**
  + **Syntax: Table Rename:**

**Schema::rename('students\_info', 'students' );**

* + **Syntax: Table Drop:**

**Schema::drop( 'students' );**

**Schema::dropIfExist( 'students' ); //Safe way**

**Roll Back Commands: (Down Method works while rollback)**

|  |
| --- |
| * + php artisan migrate:rollback |
| * + php artisan migrate:rollback --step=2 |
| * + php artisan migrate:reset |
| * + php artisan migrate:refresh |
| * + php artisan migrate:refresh --step=3 |

Laravel Raw Query vs **Laravel Query Builder**

|  |  |
| --- | --- |
| Laravel Raw Query | Laravel Raw Query Builder |
| Query Builder Use: DB::table(‘table\_name’) & Use: (use Illuminate\Support\Facades\DB;) | |
| SELECT Using Query Builder (No need Json encode) | |
| DB::select("SELECT \* FROM user"); | **DB::table('user') -> get();** |
| DB::select('SELECT \* FROM user WHERE name=‘ ’ '); | **DB::table('user')-> where('name', 'nazmul') -> get();** |
| DB::select("SELECT name FROM user"); | **DB::table('user') -> select('name') -> get();** |
| DB::select("SELECT name, cell FROM user"); | **DB::table('user') -> select('name', 'cell') -> get();** |
| DB::select('SELECT distinct name FROM user'); | **DB::table('user')-> distinct()-> select('name')-> get()** |
| Check Row if exist or not | **DB::table('user') -> where('name' ,'Nazmul') -> count();** |
| Data Insert into Database | |
| DB::insert('INSERT INTO `student`(` name`, `cell`) VALUES (?,?,?)', [$name, $cell]); | **DB::table('user') -> insert(['name' => '', 'cell' => '' ]);** |
| If Duplicate value insert generate Error | **DB::table('user') -> insert(['id' => '','n' => '', 'c' => '' ]);** |
| It Duplicate value doesn’t create error | **DB::table('user') -> insertOrIgnore(['id' => '','n' => '', 'c' => '' ]);** |
| Delete DB Data | |
| DB::delete('SELECT \* FROM user WHERE name=‘ ’ '); | **DB::table('user') -> where('id', '10') -> delete();** |
| Delete \* Row (Do not reset id) | **DB::table('user') -> delete();** |
| Delete \* Row (Reset id) | **DB::table('user') -> truncate();** |
| Update Database Data | |
| DB::update(UPDATE user SET n=? WHERE id = ? '); | **DB::table('user') -> where('id','1') -> update(['n' => '']);** |
| Aggregates or Calculation | |
|  | **DB::table('user') -> count();** |
|  | **DB::table('user') -> max('id');** |
|  | **DB::table('user') -> min('id');** |
|  | **DB::table('user') -> avg('id');** |
|  | **DB::table('user') -> sum('id');** |
| Marge (Join Multi result) | |
| Get two table data no Unique data required | **$student = DB::table('student') -> get();**  **$user = DB::table('user') -> get();**  **$student -> merge($user);** |
| Join (Left Or Right) Need Foreign key | |
| All right uncommon Left | **DB::table('user') -> rightJoin('st','user.cell','st.cell') -> get();** |
| All Left uncommon Right | **DB::table('user') -> lefttJoin('st','user.cell','st.cell') -> get();** |

Eloquent ORM **(Object Relational Model**)

**\*\* Model will be created for each table\*\***

**Model Required Fixed Properties:**

**Protected $table = ‘table\_name’;**

**Protected $primaryKey = ‘id’;**

**Protected $keyType = ‘int’;**

**Public $incrementing = true/ false;**

**Public $timestamps = true/false;**

**Protected $dateFormat = ‘U’;**

**Protected $connection = ‘connection\_name’;**

|  |  |
| --- | --- |
| ORM (object relational Model use): studentModel::get(); | |
| studentModel::get();/all(); | |
| studentModel::where('name','=','nazmul') -> get(); | |
| studentModel::select('name') -> get(); | |
| studentModel::orderBy('roll', ‘desc) ->take(3)-> get(); | |
| studentModel::orderBy('roll', ‘desc) ->limit(3)-> get(); | |
| studentModel::select('name', 'cell') -> get(); | |
| studentModel::distinct()-> select('name')-> get() | |
| studentModel::where('name','Nazmul') -> count(); | |
| loginModel::where('name', $name)->value('name'); //Echo Single Data | |
| studentModel::where(['name' => $name, 'cell' =>$cell ]) -> count(); //Check Multiple | |
| Data Insert into Database | |
| studentModel::insert(['name' => '', 'cell' => '' ]); | |
| studentModel::insert(['id' => '','n' => '', 'c' => '' ]); | |
| studentModel::insertOrIgnore(['id' => '','n' => '', 'c' => '' ]); | |
| Delete DB Data | |
| studentModel::where('id','10') -> delete(); | |
| studentModel::delete(); | |
| studentModel::truncate(); (Delete with id History) | |
| Update Database Data | |
|  | studentModel::**where('id','1') -> update(['n' => '']);** |
| Aggregates or Calculation | |
|  | studentModel::**count();** |
|  | studentModel::**max('id');** |
|  | studentModel::**min('id');** |
|  | studentModel::**avg('id');** |
| Join | |
|  |  |

Database Seeding (**Create Random Data**)

**Commands:**

* + **Requirement: composer dump-autoload**
  + **SeederCreate: php artisan make:seeder seederName**
  + **RunSeeder: php artisan db:seed --class= seederName**

**Requirements:**

* + **use Illuminate\Support\Facades\DB;**

**syntax: (single Seeder file)**

**DB::table('user')->insert([**

**'name' => Str::random(10),**

**'email' => Str::random(10).'@gmail.com',**

**'cell' => Str::random(5)];**

**syntax: (Multiple Seeder File)**

**\*\*First create another seeder file**

**$this -> call([**

**firstSeeder::class,**

**secondSeeder::class**

**]);**

**\*\*Run total seeder file;**

**Commands:**

* + **Requirement: composer dump-autoload**
  + **SeederCreate: php artisan make:seedar seederName**
  + **RunSeeder: php artisan db:seed --class= seederName**

**Laravel** Http Client 😉

**Requirements :**

* **Class: use Illuminate\Support\Facades\Http;**
* **$response = Http::get('http://apishooter.com/getSimpleString');**
* **return $response->headers();**

Frontend Scaffolding

* **Laravel uses NPM(Node Package Manager) to install Frontend:**

**#Laravel UI Composer Package Install:**

* **Command: composer require laravel/ui**

**Once the laravel/ui package has been installed, you may install the frontend scaffolding using the ui Artisan command:**

**# Generate basic scaffolding.**

* **php artisan ui bootstrap**
* **php artisan ui vue**
* **php artisan ui react**

**# Generate scaffolding with login / registration.**

* **php artisan ui bootstrap --auth**
* **php artisan ui vue --auth**
* **php artisan ui react –auth**

**# Install NPM.**

* **Firstly Install node.js app on your pc.**
* **Command: install npm**
* **Run Development npm to convert resurce sass to public css.**
* **Command: npm run dev**
* **Transfer Resource Custom.scss as public custom.css using npm run dev**
* **Step 1: create .scss file inside resource/sass.**
* **Step 2: create mix.sass(‘’, ‘’) constant inside webpack.mix.js**
* **Syntax 1: mix.sass('resources/sass/custom.scss', 'public/css');//Convert**
* **Syntax 2: mix.copy('resources/sass/custom.css, 'public/css);//Transfer**
* **Step 3: link css in app.blade.php**
* **Step 4: npm run dev (cmd command)**
* **Transfer Resource Custom.js as public custom.js using npm run dev**
* **Step 1: create .js file inside resource/js.**
* **Step 2: create mix.js(‘’, ‘’) constant inside webpack.mix.js**
* **Syntax: mix.js('resources/sass/custom.js, 'public/js);**
* **Step 3: link js in app.blade.php**
* **Step 4: npm run dev (cmd command)**

**Another Way Compile with Public/js/app.js:**

* **Step 1: create custom.js file inside resource/js.**
* **Step 2: go to resource/js/app.js**
* **Syntax: require('./custom');**
* **Step 3: npm run dev (cmd command)**
* **Laravel With Bootstrap Scaffolding 😊**

**#Installation Commands:**

* **composer create-project –prefer-dist laravel/laravel project**
* **composer require laravel/ui**
* **php artisan ui bootstrap**

**install node.js app on your pc**

* **install npm**

**create custom.js & custom.scss file, connected in app file**

**synchronize jquery, popper & bootstrap in resource/bootstrap.js**

* **npm run dev**
* **Laravel With React Scaffolding 😐**

**#Installation Commands:**

* **composer create-project –prefer-dist laravel/laravel project**
* **composer require laravel/ui**
* **php artisan ui react**

**install node.js app on your pc**

* **install npm**
* **npm install react-bootstrap bootstrap**

**create custom.js & custom.scss file, connected in app file**

**synchronize jquery, popper & bootstrap in resource/bootstrap.js**

* **npm run dev**

**Laravel** file Upload Using JQuery & Axious

**Axios Handle Laravel CSRF Protection**

**Syntax:**

**//Receive File Data**

**$('#upload').click(function(){**

**let img = $('#img').prop('files'); //Get Object file list;**

**let img = $('#img').prop('files')[0]; //Get single Object file;**

**//Get All Data**

**let img = $('#img').prop('files')[0];**

**let imgName = img.name;**

**let imgSize = img.size;**

**let imgExt = imgName.split('.').pop().toLowerCase();**

**});**

**//Transfer By Axios**

**let url = '/upload';**

**let data = new FormData();**

**data.append(‘key’, file);**

**let config = {headers:{'content-type': 'multipart/form-data'}}**

**axios.post(url, data, config).then(function(response){**

**if (response.status==200) {**

**$('#upload').html('Upload Success');**

**$('#img').val('');**

**setTimeout(function(){**

**$('#upload').html('Upload Photo');**

**},2000);**

**}else{**

**$('#upload').html('Upload Failed');**

**$('#img').val('');**

**setTimeout(function(){**

**$('#upload').html('Upload Photo');**

**},2000);**

**}**

**}).catch(function(error){ });**

**//Set Config & File progress Check:**

**let config = {**

**headers:{'content-type': 'multipart/form-data'},**

**onUploadProgress: function(progressEvent){**

**let progress = Math.round((progressEvent.loaded\*100)/progressEvent.total);**

**let progressbar= "<div class='progress mt-3'><div class='progress-bar'**

**style='width:"+progress+"%'>"+progress+"%</div></div>";**

**$('#response').html(progressbar);**

**let totalMB = (progressEvent.total)/(1028\*1028);**

**let loadedMB = (progressEvent.loaded)/(1028\*1028);**

**let dueMB = totalMB - loadedMB;**

**$('#response').html(**

**'Total Size : '+ totalMB.toFixed(2) + 'MB db<br>**

**Uploaded : '+ loadedMB.toFixed(2) + 'MB <br>**

**Due : '+ dueMB.toFixed(2) + 'MB <br>**

**Persentage : '+ progress + '%'**

**);}**

**//Table Data Set By jQuery Using Axios:**

**axios.get('/serviceData')**

**.then(function(response){**

**let service =response.data;**

**$.each(service,function(i,item){**

**$('<tr>').html(**

**"<td><img class='table-img' src="+service[i].image+"></td>"+**

**"<td>"+service[i].title+"</td>"+**

**"<td>"+service[i].description+"</td>"+**

**"<td><a href='' ><i class='fas fa-edit'></i></a></td>"+**

**"<td><a href='' ><i class='fas fa-trash-alt'></i></td>"**

**).appendTo('#service\_table');**

**});**

**}).catch(function(error){**

**})**

**Laravel** file Upload Using Laravel Storage

**# Why.**

* **Publicly Accessible From anywhere**
* **By Default, Public Disk uses Local Driver**

**# Requirement.**

* **use Illuminate\Support\Facades\Storage;**
* **Command: php artisan storage:link**
* **Asset{{‘storage/file\_name’}};**

**# Laravel Store Fucntions.**

* **Store: $request ->file('key') ->store('images');**
* **Download: Storage::download($path.'/'.$name);**
* **Delete: Storage::delete($path.'/'.$name);**

Login Authentication Using Middleware & Session 😉

**# Set Value in Session:**

**Set By Request Class & Session Method.**

* **$request-> session()->put(‘key’, ‘value’);**

**Set By Session Class**

**use Illuminate\Support\Facades\Session;**

**Session::put(‘key’, ‘value’);**

**# Setup MiddleWare:**

* **Step:1 - Php artisan make:middleware loginMiddleware;**
* **Step:2- Go in handle() method & Check The Session.**

**if ($request->session()->has('user')) {return $next($request);}**

**else{return redirect('/login');}**

* **Step:3- Karne.php File & Set a route mIddleware:**
  + **'loginCheck'=> \App\Http\Middleware\loginCheckMiddleware::class,**
* **Step:4- Use Middleware:**
  + **Route::get('/', 'mainController')->middleware('loginCheck');**

**# Set Session Value:**

* **$request-> session()-> flush();**

**# Set Session in Blade View:**

* **{{Session::get('name')}}**

**# Also, Can Set Middleware in Controller:**

* **Create –construct() function in controller Class;**
* **Syntax:**

**function –construc(){**

**$this->middleware('check');**

**}**

Laravel Session & Cookie😉

**# SESSION:**

* **Create file in a temporary directory on the server.**
* **Inside file session var & their value will be stored.**
* **Var & value will be available to all pages on the side during visit that.**
* **Session will be closed when user close the browser.**
* **Server will terminate session after a predetermined period of time.**

**# COOKIE:**

* **Cookie are text files stored on client’s computer.**
* **Server script sent a set of cookies on browser.**
* **Browser store information on local computer for future use.**

**# SET VALUE IN SESSION:**

**Set, Retrieve, Clear, Destroy Session Data:**

* **$request-> session()->put(‘key’, ‘value’); //Set Session Data**
* **$request-> session()->get(‘key’); // Retrieve Session Data.**
* **$request-> session()->forget(‘key’); // Clear single Session Data.**
* **$request-> session()->flush(); // Destroy Session Data**

**Do all above using session class:**

**use Illuminate\Support\Facades\Session;**

**Session::put(‘key’, ‘value’);**

**# SET VALUE IN COOKIE:**

**Set, Retrieve, Clear, Destroy Cookie Data:**

**Required Class : use Illuminate\Support\Facades\Cookie;**

* **Cookie::queue(‘key’, ‘value’, ‘10’); //Set Data in Cookie time in min.**
* **Cookie::get(‘key) //Get Cookie Data.**
* **Cookie::queue(Cookie::forget(‘key’) ); //Clear Cookie Data.**

Laravel Social Media Login (Github)

**# composer Package:**

* **composer require laravel/socialite**

**# Login With GitHub:**

* **Step:1 – Create Auth Application in Github.**
* **Step:2- Go to Config -> app.php (Add Class in Provider & Aliases)**

**'providers' => [ Laravel\Socialite\SocialiteServiceProvider::class, ],**

**'aliases' => [ 'Socialite' => Laravel\Socialite\Facades\Socialite::class, ],**

* **Step:3- Go to Config -> services.php (Third Perty Services)**

**'github' => [**

**'client\_id' => '422294b83512a7ea6af2',**

**'client\_secret' => 'a9a0ee91c43558ec2053272fefed19fd7ce4d09c',**

**'redirect' => 'http://127.0.0.1:8000/gitHubCallBack',**

**],**

* **Step:4- Create Controller (method) Redirect To Provider:**

**use Laravel\Socialite\Facades\Socialite;**

**return Socialite::driver('github')->redirect();**

* **Step:4- Create Controller (method) Provider Call Back:**

**$user = Socialite::driver('github')->stateless()->user();**

$token = $user->token;

$name = $user->getName();

$nikName = $user->getNickname();

$email = $user->getEmail();

$avatar = $user->getAvatar();

* **Step:5- Setup Middleware for Login Security:**
  + - **Create New MiddleWare -> Handle Method -> check if Session Created or Not**
    - **Set MiddleWare in Karnel.php & use Middleware in route.**