# Laravel – User Auth and CRUD application Of Specific data to specific user tutorial with Example

## **Step - 1 : Install Laravel**

In the very first step, we need to create laravel 8 application in your local system using the following composer command in your terminal.

composer create-project --prefer-dist laravel/laravel:^7.0 blog

## **Step - 2 : Database Configuration**

now, just open your laravel applications. env file and make the following database related changes in it.

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=laravelcrud // your database name

DB\_USERNAME=root // your database user name

DB\_PASSWORD=root // your database password

## **Step - 3 : User Authentication**

Authentication is the process of identifying the user credentials. In web applications, authentication is managed by sessions which take the input parameters such as email or username and password, for user identification. If these parameters match, the user is said to be authenticated.

**Command**

Laravel uses the following command to create forms and the associated controllers to perform authentication −

Php artisan make:auth

**Controller**

The controller which is used for the authentication process is **HomeController**.

<?php

namespace App\Http\Controllers;

use App\Http\Requests;

use Illuminate\Http\Request;

class HomeController extends Controller{

/\*\*

\* Create a new controller instance.

\*

\* @return void

\*/

public function \_\_construct() {

$this->middleware('auth');

}

/\*\*

\* Show the application dashboard.

\*

\* @return \Illuminate\Http\Response

\*/

public function index() {

return view('home');

}

}

As a result, the scaffold application generated creates the login page and the registration page for performing authentication. They are as shown below –

**Login Page And Registration page**

## Manually Authenticating Users

Laravel uses the **Auth** façade which helps in manually authenticating the users. It includes the **attempt** method to verify their email and password.

Consider the following lines of code for **LoginController** which includes all the functions for authentication –

<?php

// Authentication mechanism

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Auth;

class LoginController extends Controller{

/\*\*

\* Handling authentication request

\*

\* @return Response

\*/

public function authenticate() {

if (Auth::attempt(['email' => $email, 'password' => $password])) {

// Authentication passed...

return redirect()->intended('dashboard');

}

}

}

## **Step - 4 : PostController**

<?php

namespace App\Http\Controllers;

use App\Models\Post;

use App\Models\Users;

use Illuminate\Http\Request;

use Auth;

class PostController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        // return view('insert');

        $posts=Post::all();

        $user=Users::find(Auth::user()->id);

        if($user->group->name!='admin'){

            $posts = $posts->where('created\_by', Auth::user()->id);

        }

        $work="index";

        return view('dummy',compact('work','posts','user'));

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $posts=new Post;

        $posts->post\_title=$request->get('title');

        $posts->post\_author=$request->get('author');

        $posts-> save();

        // $data='<h1>Data Added Successfully.....</h1>';

        // return view('show',compact('posts'));

        // return redirect('/show');

        // return view('insert',compact('data'));

        // echo "<h1>Data Added Successfully.....</h1>";

        // $page\_data=>$data

        return redirect('home');

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  \App\Models\Post  $post

     \* @return \Illuminate\Http\Response

     \*/

    public function show(Post $post)

    {

        // $posts=Post::all();

        // // return view('show',['posts'=>$posts]);

        // $work="show";

        // return view('dummy',compact('work'));

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*

     \* @param  \App\Models\Post  $post

     \* @return \Illuminate\Http\Response

     \*/

    public function edit($id)

    {

        // print\_r($id);

        // exit();

        $post=Post::find($id);

        // return view('edit',['posts'=>$posts]);

        $posts=Post::all();

        $posts = $posts->where('created\_by', Auth::user()->id);

        $work="edit";

        return view('dummy',compact('work','posts','post'));

    }

    /\*\*

     \* Update the specified resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  \App\Models\Post  $post

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, Post $post,$id)

    {

        $posts=Post::find($id);

        $posts->post\_title=$request->get('title');

        $posts->post\_author=$request->get('author');

        $posts-> update();

        // return redirect('/show');

        // $work="update";

        // return view('dummy',compact('work'));

        return redirect('home');

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*

     \* @param  \App\Models\Post  $post

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy(Post $post,$id)

    {

        $post=Post::find($id);

        $post->delete();

        // return redirect('show');

        return redirect('home');

    }

}

## **Step - 5 : Route: Web.php**

<?php

use App\Http\Controllers\PostController;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

Route::get('',[PostController::class,'index']);

Route::post('store',[PostController::class,'store']);

// Route::get('show',[PostController::class,'show']);

Route::get('delete/{id}',[PostController::class,'destroy']);

Route::get('edit/{id}',[PostController::class,'edit']);

Route::post('update/{id}',[PostController::class,'update']);

## **Step – 6 : View: dummy.php**

<!DOCTYPE html>

<html>

<head>

    <title></title>

    <!-- CSS only -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

</head>

<body>

<div class="container">

<h1> {{$work}} </h1>

{{--

@if($work=='index')

--}}

    <h1 class="text-center">Add Data</h1>

    <form method="POST" action=" @if($work=='index')

    /store

    @elseif($work=='edit')

    /update/{{$post->id}}

    @endif">

        @csrf

        <div class="mb-3">

        <label><b>Post Title:</b></label>

        <input type="text" name="title" class="form-control" value="@if($work=='edit'){{$post->post\_title}}@endif">

        </div>

        <div class="mb-3">

        <label><b>Post Author:</b></label>

        <input type="text" name="author" class="form-control" value="@if($work=='edit'){{$post->post\_author}}@endif">

        </div>

        <input type="submit" name="insert" value="Insert" class="btn btn-primary">

    </form>

{{--

    @elseif($work=='edit')

    <h1 class="text-center">Update Data</h1>

    <form method="POST" action="/update/{{$post->id}}">

        csrf

        <div class="mb-3">

        <label><b>Post Title:</b></label>

        <input type="text" name="title" class="form-control" value="{{$post->post\_title}}">

        </div>

        <div class="mb-3">

        <label><b>Post Author:</b></label>

        <input type="text" name="author" class="form-control" value="{{$post->post\_author}}    ">

        </div>

        <input type="submit" name="update" value="update" class="btn btn-success">

    </form>

@endif

--}}

<h1 class="text-center">Inserted Data</h1>

    <table class="table table-bordered shadow text-center table-striped">

    <tr>

        <th>Post Id</th>

        <th>Post Title</th>

        <th>Post Author</th>

        <th>Post Delete</th>

        <th>Post Edit</th>

    </tr>

    @foreach($posts as $post\_item)

    <tr>

        <td>{{$post\_item->id}}</td>

        <td>{{$post\_item->post\_title}}</td>

        <td>{{$post\_item->post\_author}}</td>

        <td><a href="/delete/{{$post\_item->id}}" class="btn btn-danger">DELETE</a></td>

        <td><a href="/edit/{{$post\_item->id}}" class="btn btn-success">EDIT</a></td>

    </tr>

    @endforeach

    </table>

</div>

</body>

</html>

## **Step – 7 : Model: Post.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model

{

    use SoftDeletes;

    use HasFactory;

    protected $table="posts";

    protected $fillable=['post\_title','post\_author'];

}

## **Step – 8 : Model: 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;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

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

    ];

}