# Learning Outcome

# After completing this module, the student should be able to understand Laravel framework

To meet the learning outcome, a student has to complete the following activities

1. Demonstrate csv file upload in Laravel, extract its data and insert into database (8 Hrs.)
2. Installation and usage of jet brains Live wire and its component (8 Hrs.)
3. Create a login and register Forms using Live wire (9 Hrs.)

# Activity 1

## Aim: Demonstrate csv file upload in Laravel, extract its data and insert into database

**Learning outcome:** Able to understand Laravel Framework

**Duration:** 8 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version

**Code/Program/Procedure (with comments):**

**Install Laravel then Create the new Laravel Project**

**Composer create-project --prefer-dist laravel/laravel csv2sql**

**Step1**: **Table structure**

CREATE TABLE `users` (`id` int(11) NOT NULL PRIMARY KEY AUTO\_INCREMENT, `first\_name` varchar(80) NOT NULL, `last\_name` varchar(80) NOT NULL, `gender` varchar(10) NOT NULL, `email` varchar(80) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Step 2: Database Configuration**

Open .env file.

**Specify the host, database name, username, and password.**

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=csv2sql

DB\_USERNAME=root

DB\_PASSWORD=

**Step 3: Model**

Create a Page Model.

**php artisan make: model csvData**

Inserted code to csvData.php

**Complete Code**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class CsvData extends Model

{

use HasFactory;

protected $fillable = [

'first\_name',

'last\_name',

'email',

'gender',

];

}

**Step 4: Controller**

Create a ImportController Controller.

**php artisan make:controller ImportController**

Add to the code

**Complete Code**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\CsvData;

use Illuminate\Support\Facades\Session;

class ImportController extends Controller

{

public function index(){

return view('index');

}

public function uploadCsv(Request $request){

if ($request->input('submit') != null ){

$file = $request->file('file');

// File Details

$filename = $file->getClientOriginalName();

$extension = $file->getClientOriginalExtension();

$tempPath = $file->getRealPath();

$fileSize = $file->getSize();

$mimeType = $file->getMimeType();

// Valid File Extensions

$valid\_extension = array("csv");

// 2MB in Bytes

$maxFileSize = 2097152;

// Check file extension

if(in\_array(strtolower($extension),$valid\_extension)){

// Check file size

if($fileSize <= $maxFileSize){

// File upload location

$location = 'uploads';

// Upload file

$file->move($location,$filename);

// Import CSV to Database

$filepath = public\_path($location."/".$filename);

// Reading file

$file = fopen($filepath,"r");

$dataRow = true;

while (($data = fgetcsv($file, 4000, ",")) !== FALSE) {

if (!$dataRow) {

CsvData::create([

"first\_name" => $data['1'],

"last\_name" => $data['2'],

"email" => $data['3'],

"gender" => $data['4']

]);

}

$dataRow = false;

Session::flash('message','Import Successful.');

}

fclose($file);

return redirect(url('/'));

}

else{

Session::flash('message','File too large. File must be less than 2MB.');}

}else{

Session::flash('message','Invalid File Extension.');

}

}

}

}

**Step 5: Routes**

 Open routes/web.php file.

Here, define two routes:

* **/ -** Root
* **/uploadFile –** This is post type route which uses on <form > action.

**Completed Code**

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ImportController;

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

Route::post('/uploadFile', [ImportController::class, 'uploadCsv']);

**Step 6: View**

Create a new index.blade.php file in resources/views/ directory.

If Session message variable if it is set then display value in <p>.

Create a <form > and pass '/uploadFile' in action attribute.

In <form > add {{ csrf\_field() }}, a file element and a submit button.

**Completed Code**

<!doctype html>

<html>

<head>

<title>Import CSV Data to MySQL database with Laravel</title>

</head>

<body>

<!-- Message -->

@if (Session::has('message'))

<p>{{ Session::get('message') }}</p>

@endif

<!-- Form -->

<form method='post' action='/uploadFile' enctype='multipart/form-data'>

@csrf

<input type='file' name='file'>

<input type='submit' name='submit' value='Import'>

</form>

</body>

</html>

**Step 7: Run Development Server**

Now we are set to run our example so run below command for quick run:

**php artisan serve**

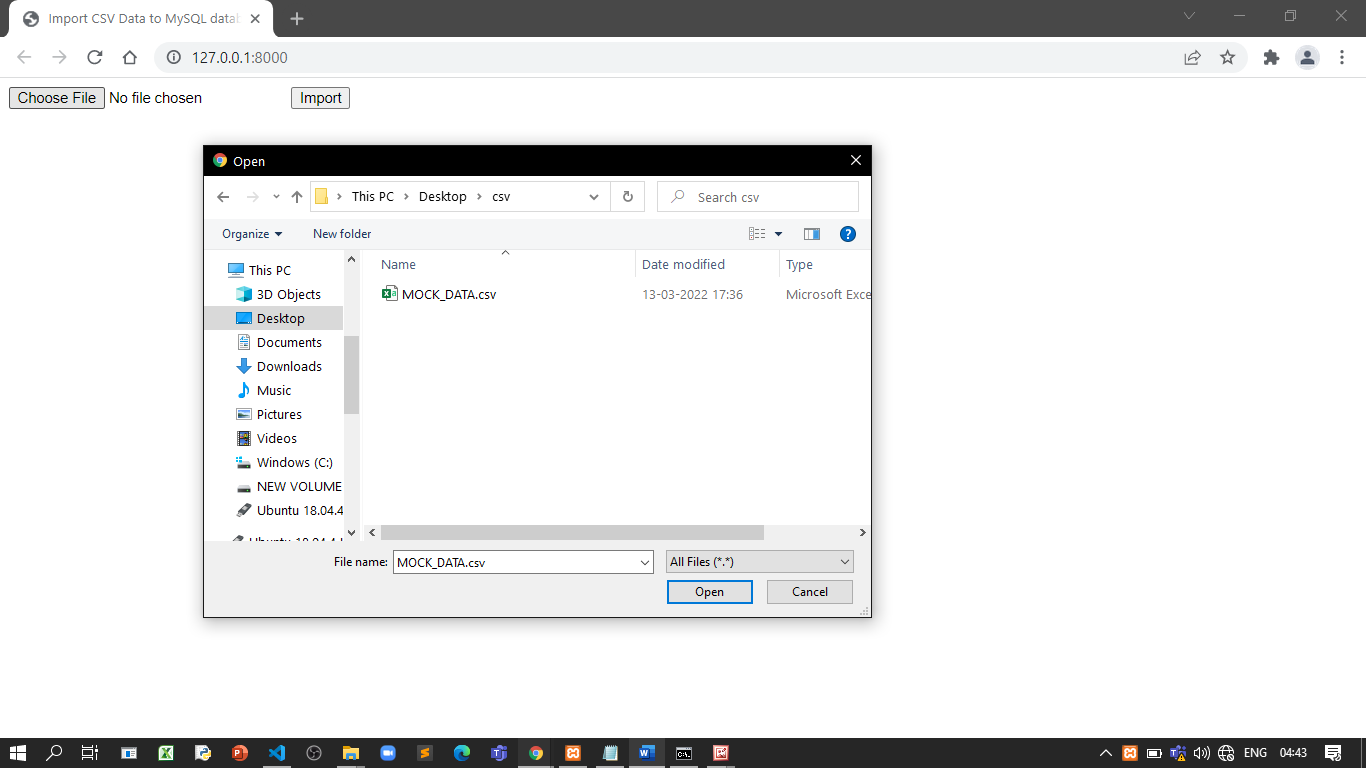
**Step 8:**

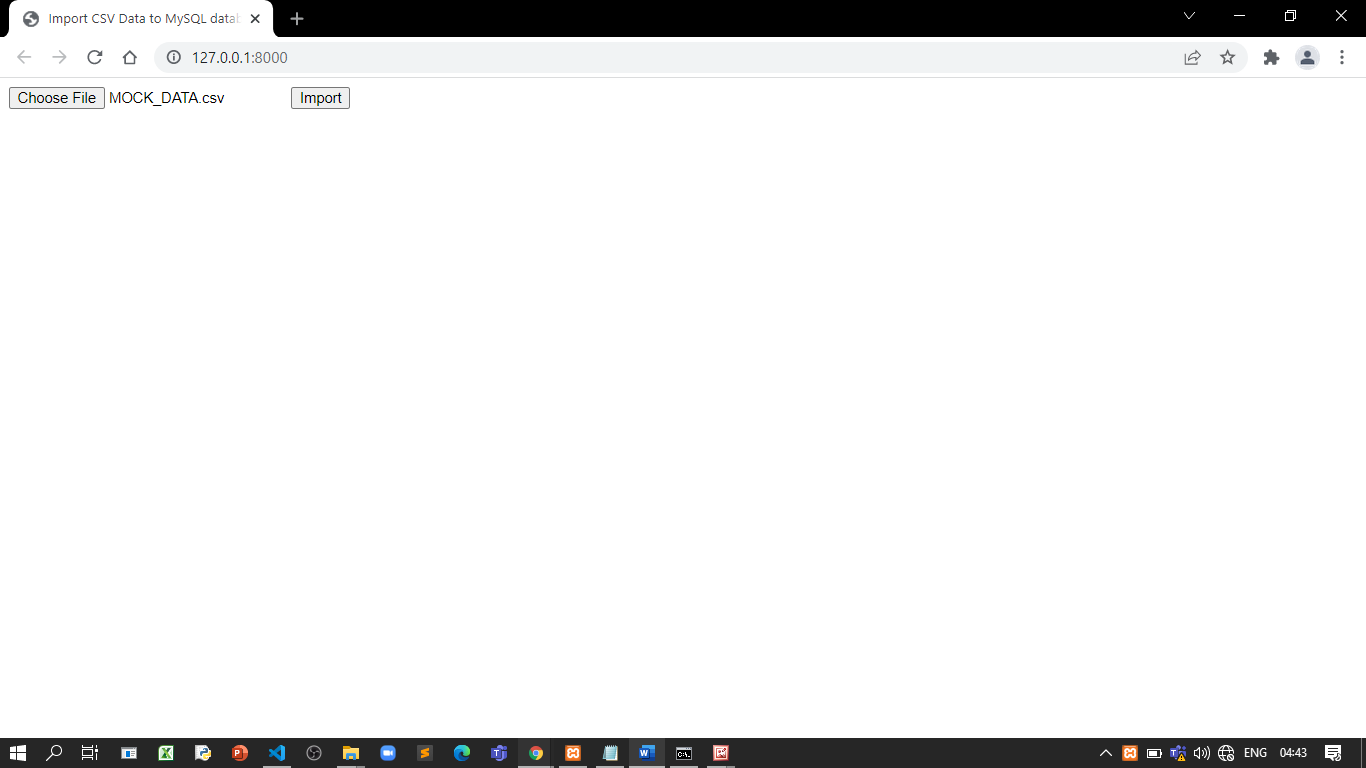
Now you can open bellow URL on your browser:

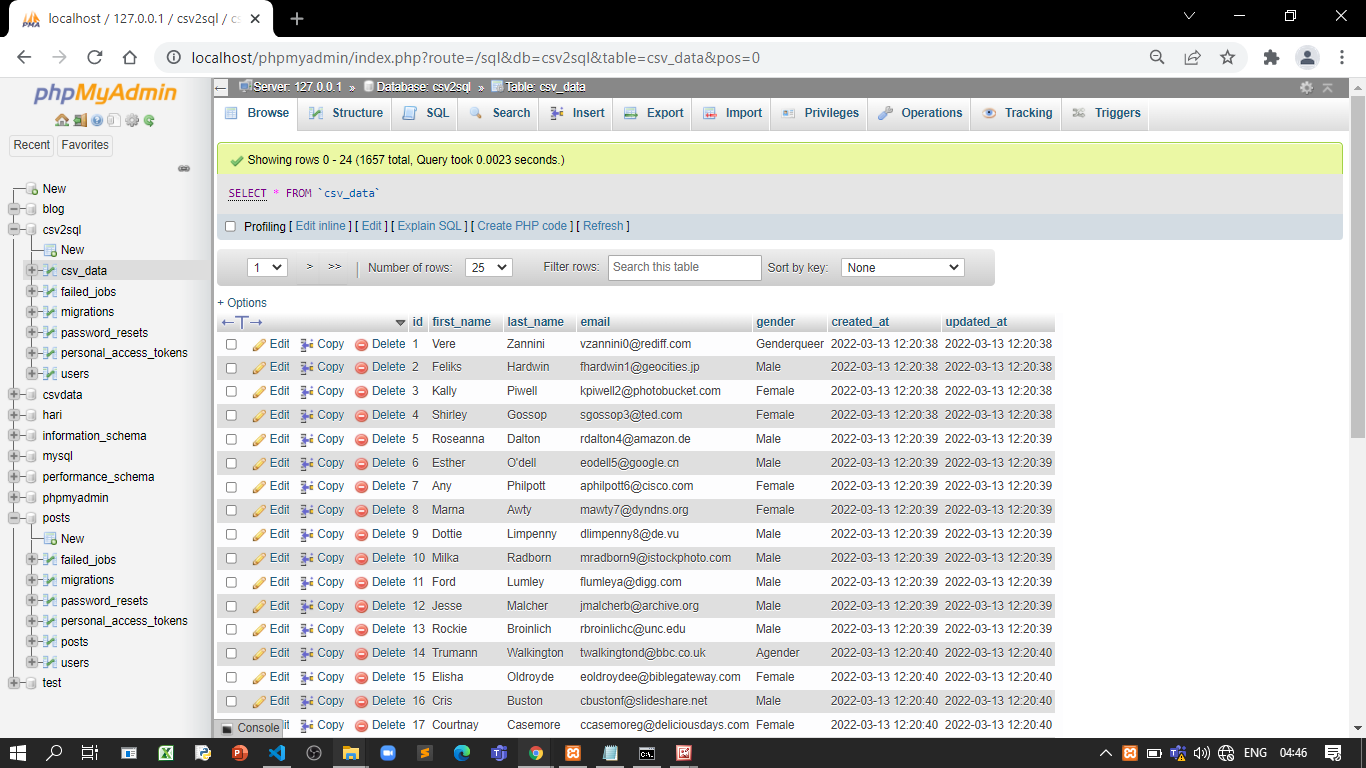
[**http://localhost:8000**](http://localhost:8000)

**Output/Results snippet:**



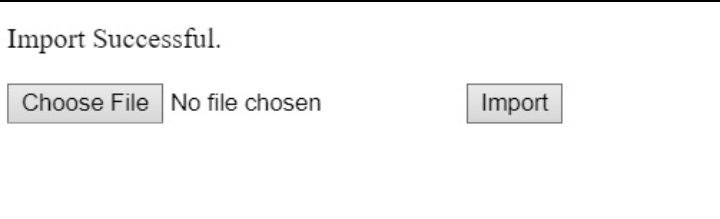








Mock\_data.csv



**References:**

* <https://makitweb.com/import-csv-data-to-mysql-database-with-laravel/>

# Activity 2

## Aim: Installation and usage of jet brains Live wire and its component

**Learning outcome:** Able to understand Laravel Framework

**Duration:** 8 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version
5. livewire

**Code/Program/Procedure (with comments):**

**Install Laravel then Create the new Laravel Project**

**Composer create-project --prefer-dist laravel/laravel livewirepro**

**Step 1:** **Create Routes**

Now we need to generate our routes. So, paste this below code in a web.php file.

**routes/web.php**

Route::view('contacts','users.contacts');

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

**Complete Code**

<?php

use App\Http\Controllers\PostController;

use Illuminate\Support\Facades\Route;

use App\Http\Livewire\Posts;

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

return view('welcome');

});

Route::view('contacts','users.contacts');

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

Here we used the view route and users.contacts shows blade view directories.

**Step 2:**

Now paste this following code to contacts.blade.php.

**resources/views/users/contacts.blade.php**

**Complete Code**

@extends('layouts.app')

@section('content')

<div class="flex justify-center">

@livewire('contact')

</div>

@endsection

**resources/views/livewire/users/contacts.blade.php**

**Complete Code**

@extends('layouts.app')

@section('content')

<div class="flex justify-center">

@livewire('contact')

</div>

@endsection

**resources/views/livewire/users/welcome..blade.php**

**Complete Code**

@extends('layouts.app')

@section('content')

<div class="antialiased">

<a class="button" href="{{ url('/contacts') }}">Post Lists</a>

@livewire('users')

</div>

@endsection

**Step 3: Create Migration and Model**

Here, we need to generate database migration for files table and also we will make a model for files table.

**php artisan make :model Post -m**

Now open Contact model and paste below code. **app/Models/Post..php**

**Complete Code**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

use HasFactory;

protected $fillable = [

'title',

'body',

];

}

**Step 4:**

Now, add these fields to your contacts table and run the migrate command to save it database.

**database/migrations/create\_posts\_table.php**

**Complete code**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('posts', function (Blueprint $table) {

$table->id();

$table->string('title');

$table->text('body');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('posts');

}

};

**Step 5:**

Then Run

**php artisan migrate**

**Step 6: Install Livewire**

**composer require livewire/livewire**

You have to set up livewire in your project for that you need to run below command

**resources/views/layouts/app.blade.php**

**Complete Code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<style>

.button {

background-color: #4CAF50;

/\* Green \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

}

.p {

margin-top: 50px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

}

.m {

margin: 20px;

padding: 10px;

}

</style>

</head>

<body>

@yield('content')

</body>

</html>

Add to the Controller-> app/Http/Controllers/PostController.php

**Complete Code**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Post;

use Illuminate\Support\Facades\Session;

use Illuminate\Support\Facades\Validator;

class PostController extends Controller

{

public function store(Request $request){

// dd($request);

$validator = Validator::make($request->all(), [

'title' => 'required',

'body' => 'required',

]);

if ($validator->fails()) {

Session::flash('message','Not valid Data');

return redirect()->back();

}else{

$post = $request->only([

'title' => 'title',

'body' => 'body',

]);

Post::create($post);

Session::flash('message','Succesfully added Post');

return redirect()->back();

}

}

}

**Step 7: Create Component**

You can create livewire component using bellow command, so let’s run below command to build user form component:

**php artisan make:livewire users**

**php artisan make:livewire contact**

Now they created files on both path:

**app/Http/Livewire/users.php**

**Complete Code**

<?php

namespace App\Http\Livewire;

use App\Models\Post;

use Livewire\Component;

use Illuminate\Support\Facades\Session;

class Users extends Component

{

public function render()

{

return view('livewire.users');

}

}

**app/Http/Livewire/Contact..php**

**Complete Code**

<?php

namespace App\Http\Livewire;

use Livewire\Component;

use App\Models\Post;

class Contact extends Component

{

public function render()

{

$this->posts = Post::all();

return view('livewire.contact');

}

public function destroy($id)

{

if ($id) {

$record = Post::where('id', $id);

$record->delete();

$this->render();

}

}

}

**Step 8:**

Add Route

We need to add a route for image form in Laravel livewire application. So open your “routes/web.php” file and add the following route in it.

routes/web.php

Route::view('users','livewire.contact');

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

**Step 9: Create Blade File**

Now paste the code to your blade file. Here we will create contact, create and update files to build our Laravel livewire demo example.

**resources/views/livewire/contact.blade.php**

**Complete Code**

<div>

@if (session()->has('message'))

<div class="alert alert-success">

{{ session('message') }}

</div>

@endif

<a class="button" href="{{ url('/') }}"> Create</a>

<table class="table table-bordered mt-5 p">

<thead>

<tr>

<th class="m">No.</th>

<th class="m">Title</th>

<th class="m">Body</th>

<!-- <th class="m" width="150px">Action</th> -->

</tr>

</thead>

<tbody>

@foreach ($posts as $post)

<tr>

<td>{{ $post->id }}</td>

<td>{{ $post->title }}</td>

<td>{{ $post->body }}</td>

<td>

<!-- <button wire:click="edit({{ $post->id }})" class="btn btn-primary btn-sm">Edit</button>

<button wire:click="delete({{ $post->id }})" class="btn btn-danger btn-sm">Delete</button> -->

</td>

</tr>

@endforeach

</tbody>

</table>

</div>

**resources/views/livewire/contact.blade.php**

@if (Session::has('message'))

<p>{{ Session::get('message') }}</p>

@endif

<form class="p" action="/store-post" method="post">

@csrf

<input type="text" name="title" placeholder="Title" />

<input type="text" name="body" placeholder="Body" />

<input class="button" type="submit">

</form>

**resources/views/livewire/users/contacts.blade.php**

**Complete Code**

@extends('layouts.app')

@section('content')

<div class="flex justify-center">

@livewire('contact')

</div>

@endsection

**resources/views/livewire/users/welcome..blade.php**

**Complete Code**

@extends('layouts.app')

@section('content')

<div class="antialiased">

<a class="button" href="{{ url('/contacts') }}">Post Lists</a>

@livewire('users')

</div>

@endsection

**Step 10: Run Development Server**

Now we are set to run our example so run below command for quick run:

**php artisan serve**

**Step 11:**

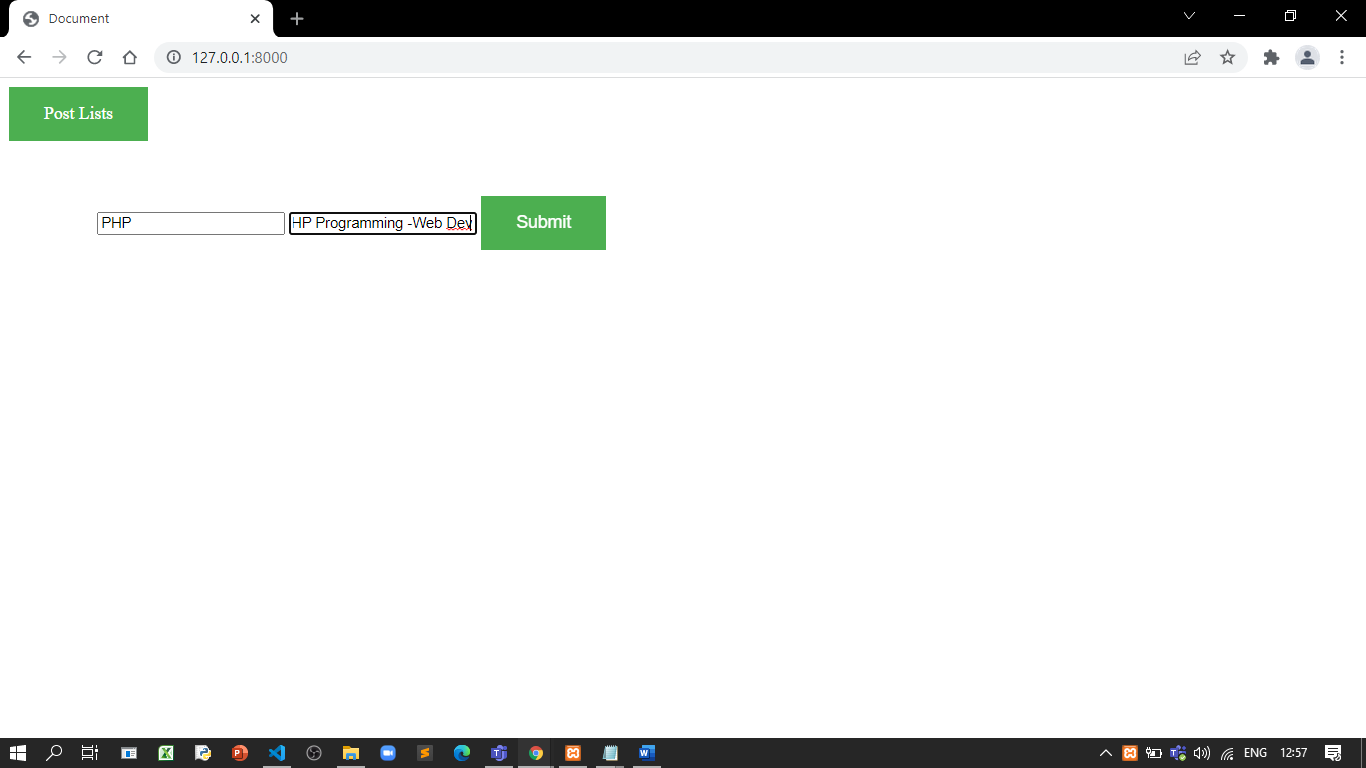
Now you can open bellow URL on your browser:

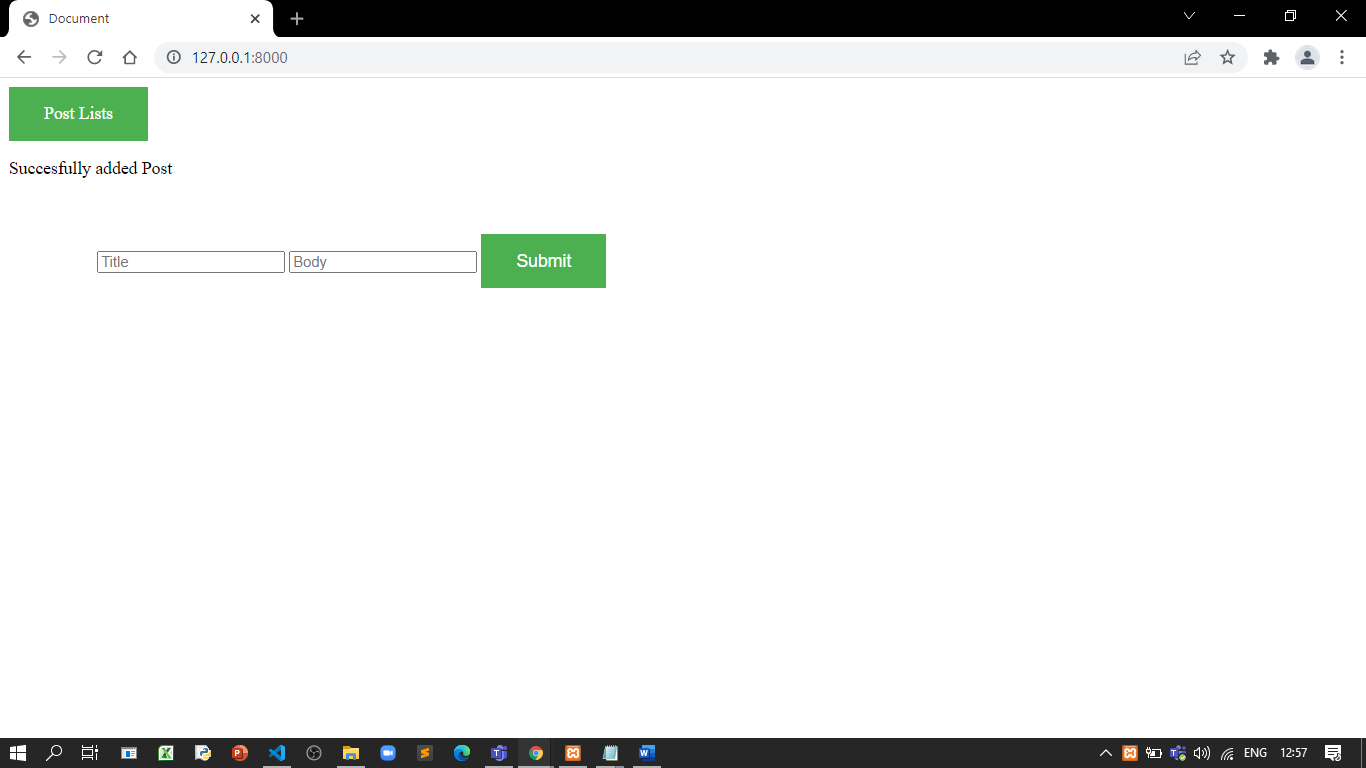
[**http://127.0.0.1:8000/contacts**](http://127.0.0.1:8000/contacts)

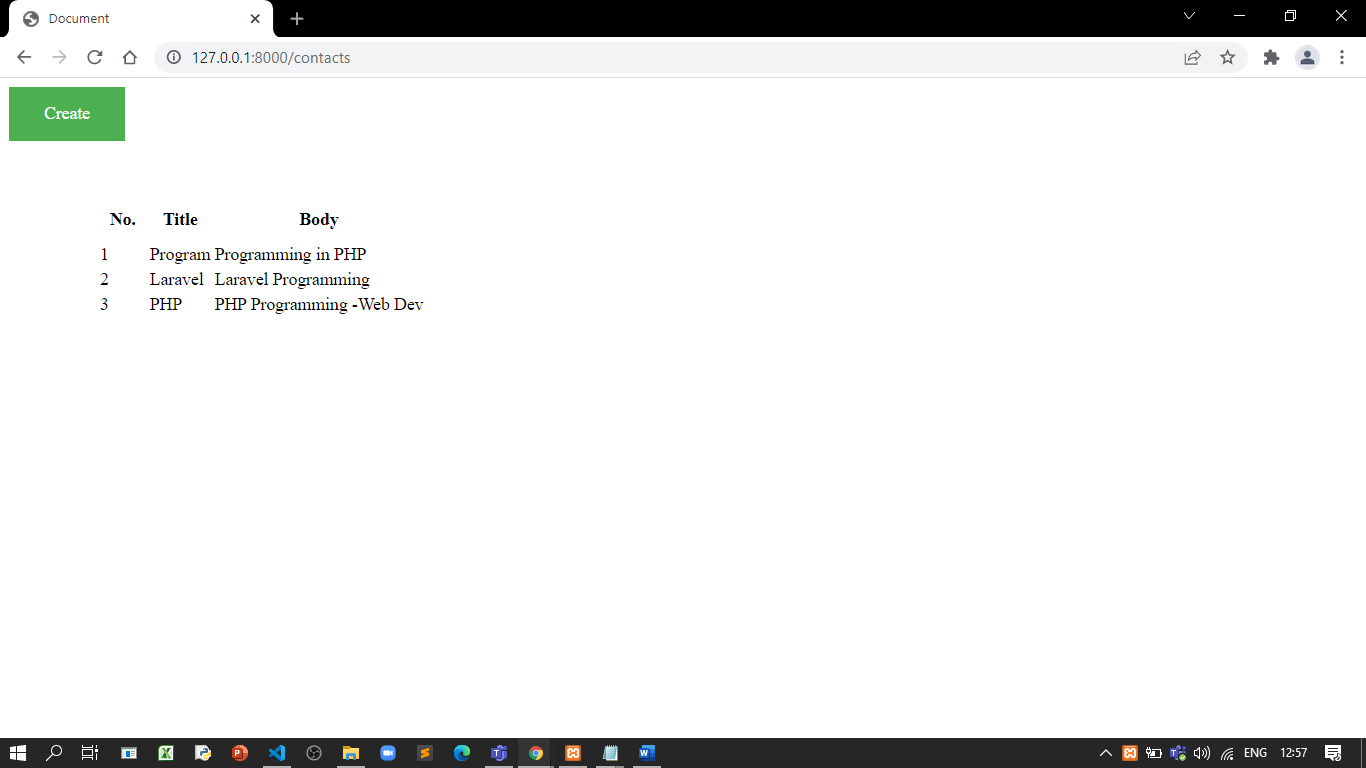
[**http://127.0.0.1:8000/**](http://127.0.0.1:8000/contacts)

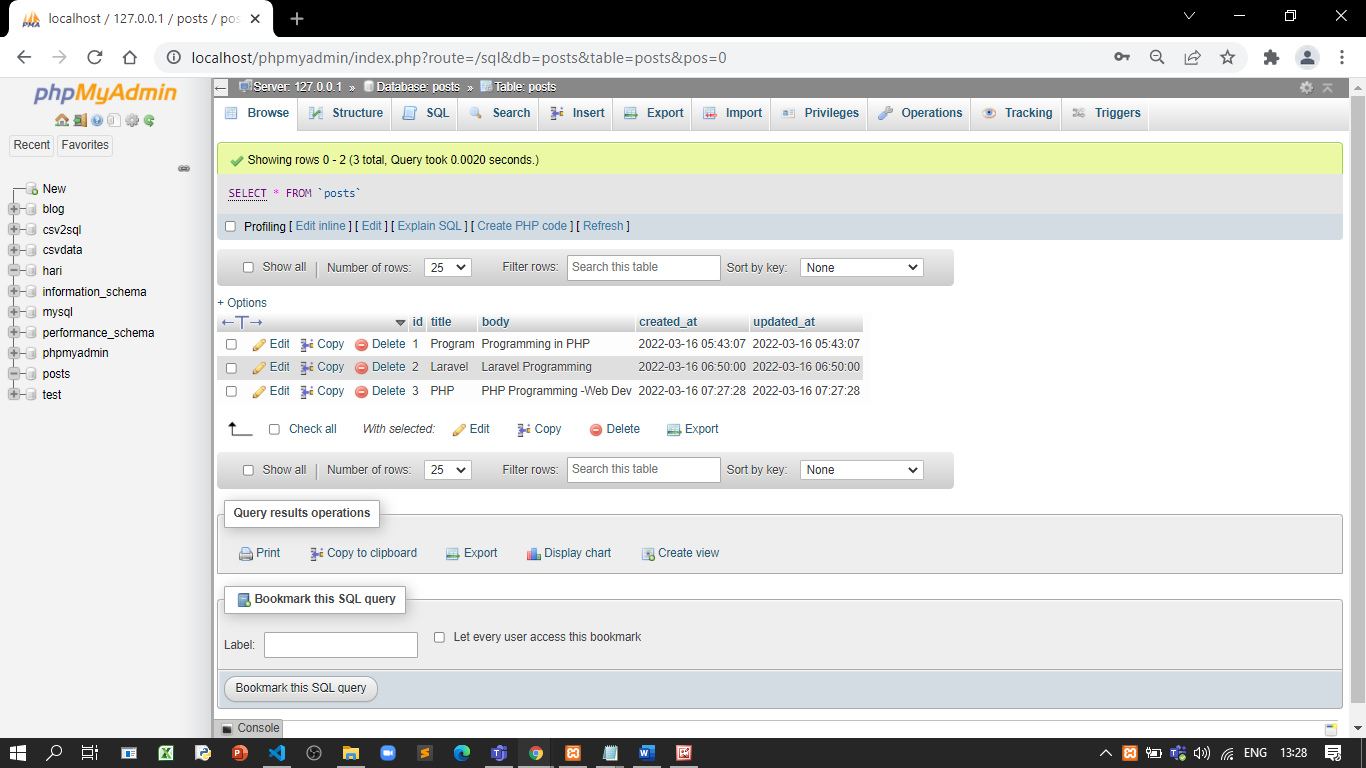
**Output/Results snippet:**











**References:**

* <https://medium.com/@avnshrathod/laravel-livewire-installation-and-demo-dfaf930f5f64>

# Activity 3

## Aim: Create a login and register Forms using Live wire

**Learning outcome:** Able to understand Laravel Framework

**Duration:** 9 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version
5. livewire

**Code/Program/Procedure (with comments):**

**Install Laravel then Create the new Laravel Project**

**Composer create-project --prefer-dist laravel/laravel livewirelogin**

**Step 1: Add Database Detail**

In this step, Add database credentials in the .env file. So, open your project root directory and find .env file. Then add database detail in .env file:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=here your database name here

DB\_USERNAME=here database username here

DB\_PASSWORD=here database password here

**Step 2:**

Run Migration Command

Now, Open command prompt and run the following command to create the table into your added database:

**php artisan migrate**

**Step 3:**

Install Livewire to Create Components

In this step, you need to install livewire to your laravel project using the following command:

**composer require livewire/livewire**

**Step 4:**

Generate Livewire Components for Login & Registration

In this step, create the livewire components for **creating a laravel livewire registration and login** components using the following command. So Open your cmd and run the following command:

**php artisan make:livewire registration-login**

This command will create the following components on the following path:

**app/Http/Livewire/RegistrationLogin.php**

**resources/views/livewire/registration-login.blade.php**

Now, Navigate to **app/Http/Livewire**folder and open**RegistrationLogin.php** file. Then add the following code into your **RegistrationLogin.php** file:

**Complete Code**

<?php

namespace App\Http\Livewire;

use Livewire\Component;

use Hash;

use App\Models\User;

class RegistrationLogin extends Component

{

public $users, $email, $password, $name;

public $registerForm = false;

public function render()

{

return view('livewire.registration-login');

}

private function resetInputFields(){

$this->name = '';

$this->email = '';

$this->password = '';

}

public function login()

{

$validatedDate = $this->validate([

'email' => 'required|email',

'password' => 'required',

]);

if(\Auth::attempt(array('email' => $this->email, 'password' => $this->password))){

session()->flash('message', "You have been successfully login.");

}else{

session()->flash('error', 'email and password are wrong.');

}

}

public function register()

{

$this->registerForm = !$this->registerForm;

}

public function registerStore()

{

$v = $this->validate([

'name' => 'required',

'email' => 'required|email',

'password' => 'required',

]);

$this->password = Hash::make($this->password);

$data = [ 'name' => $this->name,

'email' => $this->email,

'password' => $this->password

];

User::create($data);

session()->flash('message', 'You have been successfully registered.');

$this->resetInputFields();

}

}

After that, Navigate to **resources/views/livewire**folder and open **registration-login.blade.php** file. Then add the following code into your **registration-login.blade.php** file:

**Compelete code**

<div>

<div class="row">

<div class="col-md-12">

@if (session()->has('message'))

<div class="alert alert-success">

{{ session('message') }}

</div>

@endif

@if (session()->has('error'))

<div class="alert alert-danger">

{{ session('error') }}

</div>

@endif

</div>

</div>

@if($registerForm)

<form>

<div class="row">

<div class="col-md-12">

<div class="form-group">

<label>Name :</label>

<input type="text" wire:model="name" class="form-control">

@error('name') <span class="text-danger error">{{ $message }}</span>@enderror

</div>

</div>

<div class="col-md-12">

<div class="form-group">

<label>Email :</label>

<input type="text" wire:model="email" class="form-control">

@error('email') <span class="text-danger error">{{ $message }}</span>@enderror

</div>

</div>

<div class="col-md-12">

<div class="form-group">

<label>Password :</label>

<input type="password" wire:model="password" class="form-control">

@error('password') <span class="text-danger error">{{ $message }}</span>@enderror

</div>

</div>

<div class="col-md-12 text-center">

<button class="btn text-white btn-success" wire:click.prevent="registerStore">Register</button>

</div>

<div class="col-md-12">

<a class="text-primary" wire:click.prevent="register"><strong>Login</strong></a>

</div>

</div>

</form>

@else

<form>

<div class="row">

<div class="col-md-12">

<div class="form-group">

<label>Email :</label>

<input type="text" wire:model="email" class="form-control">

@error('email') <span class="text-danger error">{{ $message }}</span>@enderror

</div>

</div>

<div class="col-md-12">

<div class="form-group">

<label>Password :</label>

<input type="password" wire:model="password" class="form-control">

@error('password') <span class="text-danger error">{{ $message }}</span>@enderror

</div>

</div>

<div class="col-md-12 text-center">

<button class="btn text-white btn-success" wire:click.prevent="login">Login</button>

</div>

<div class="col-md-12">

Don't have account? <a class="btn btn-primary text-white" wire:click.prevent="register"><strong>Register Here</strong></a>

</div>

</div>

</form>

@endif

</div>

**Step 5: Create Route**

In this step, Navigate to routes folder and open web.php. Then add the following routes into your web.php file:

**Route::view('registration-login','livewire.home');**

**Step 6:**

Create View File

In this step, navigate to **resources/views/livewire** folder and create one blade view files that name **home.blade.php** file. Then add the following code into your **home.blade.php** file:

**Complete code**

<!DOCTYPE html>

<html lang="{{ str\_replace('\_', '-', app()->getLocale()) }}">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Laravel Livewire Registration and Login</title>

<!-- Fonts -->

<link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.1/css/bootstrap.min.css">

<!-- Styles -->

<style>

html, body {

background-color: #fff;

color: #636b6f;

font-family: 'Nunito', sans-serif;

font-weight: 200;

height: 100vh;

margin: 0;

}

.full-height {

height: 100vh;

}

.flex-center {

align-items: center;

display: flex;

justify-content: center;

}

.position-ref {

position: relative;

}

.top-right {

position: absolute;

right: 10px;

top: 18px;

}

.content {

text-align: center;

}

.title {

font-size: 84px;

}

.links > a {

color: #636b6f;

padding: 0 25px;

font-size: 13px;

font-weight: 600;

letter-spacing: .1rem;

text-decoration: none;

text-transform: uppercase;

}

.m-b-md {

margin-bottom: 30px;

}

</style>

</head>

<body>

<div class="container mt-5">

<div class="row mt-5 justify-content-center">

<div class="mt-5 col-md-8">

<div class="card">

<div class="card-header bg-primary text-white"><h3>Laravel Livewire Registration and login</h3></div>

<div class="card-body">

@livewire('registration-login')

</div>

</div>

</div>

</div>

</div>

@livewireScripts

</body>

</html>

**Step 7: Run Development Server**

Finally, you need to run the following **PHP artisan serve** command to start your laravel livewire registration and login app:

**php artisan serve**

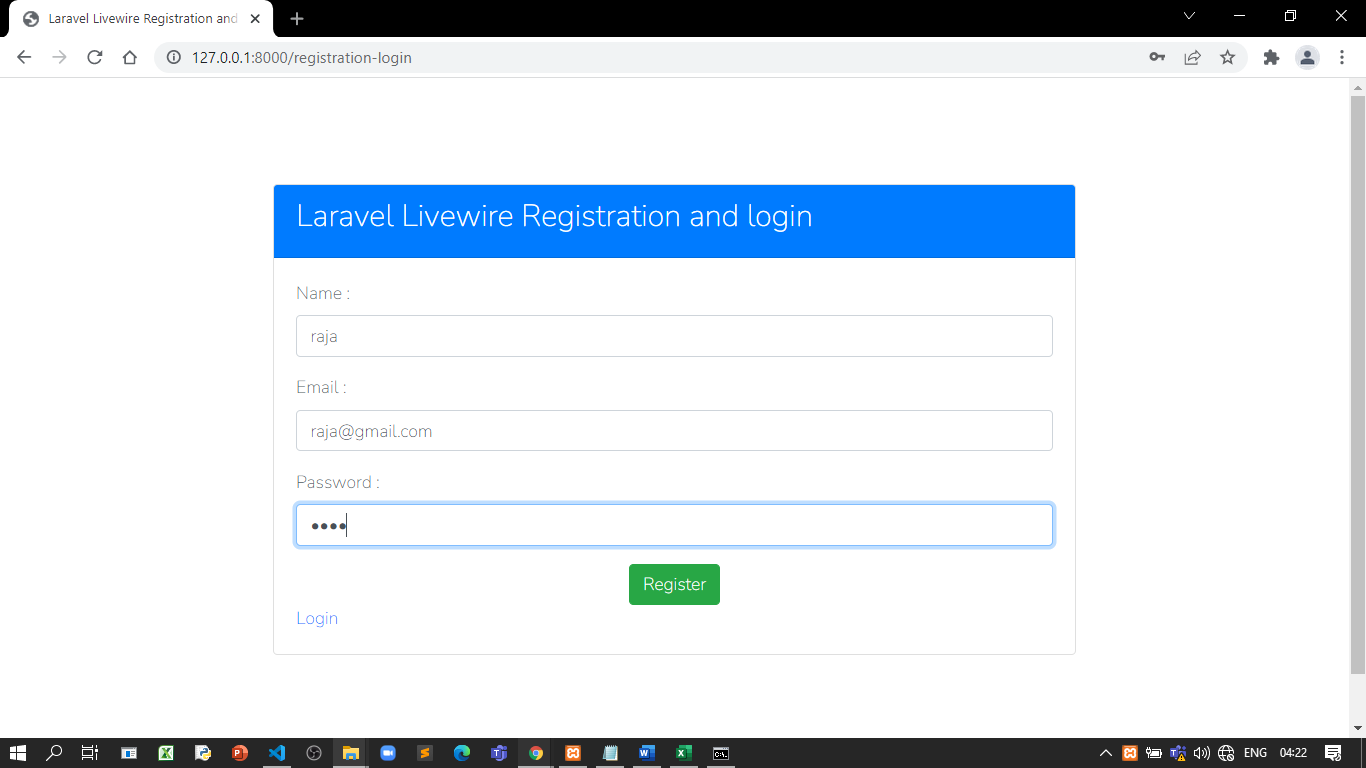
**Step 8:**

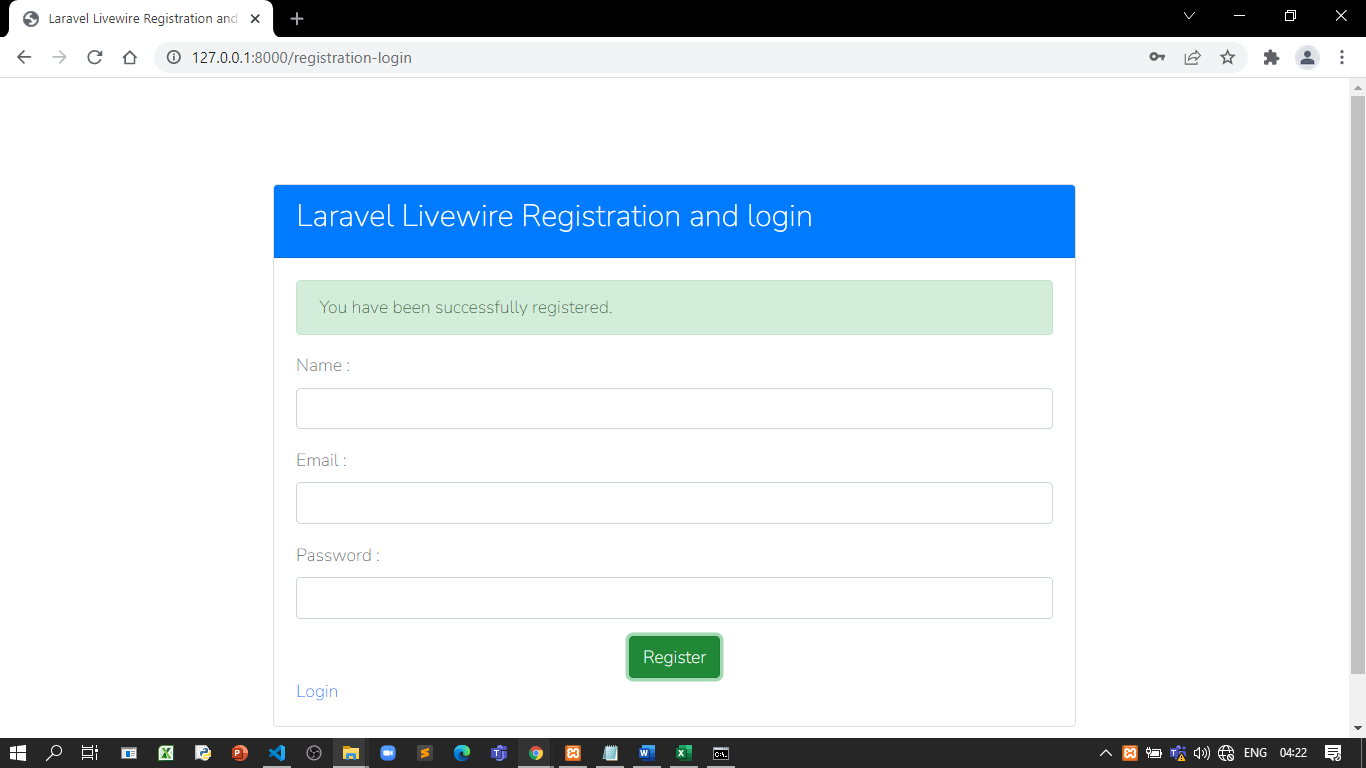
Now, you are ready to run laravel livewire registration and login app. So open your browser and hit the following URL into your browser:

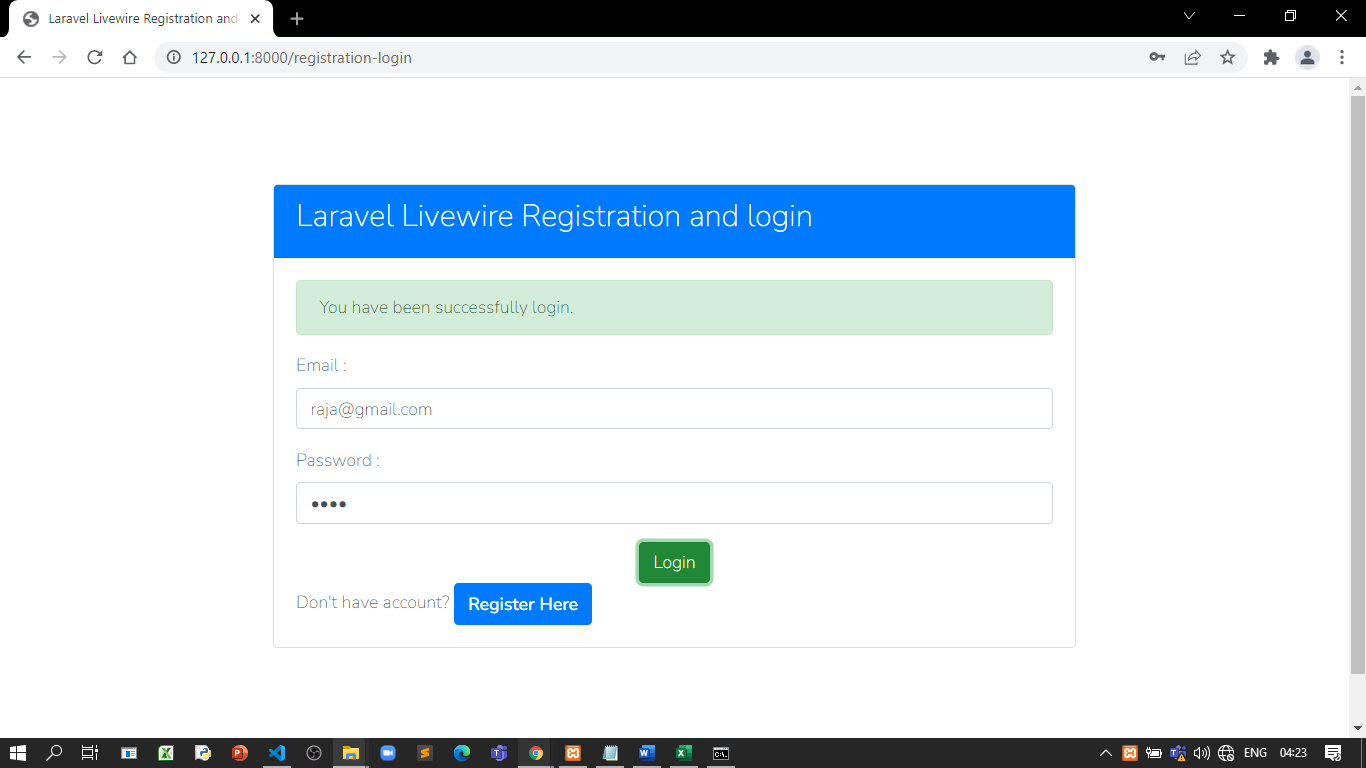
[**http://localhost:8000/registration-login**](http://localhost:8000/registration-login)

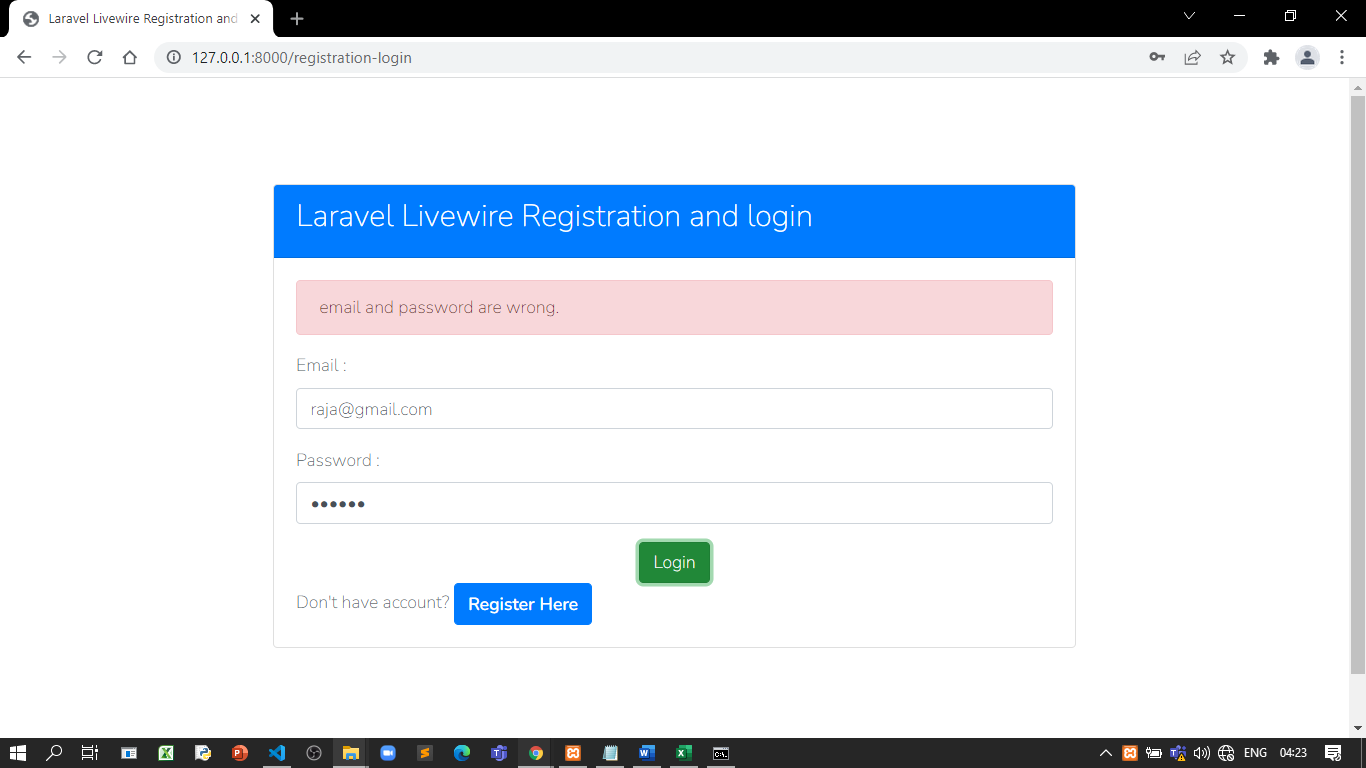
**Output/Results snippet:**

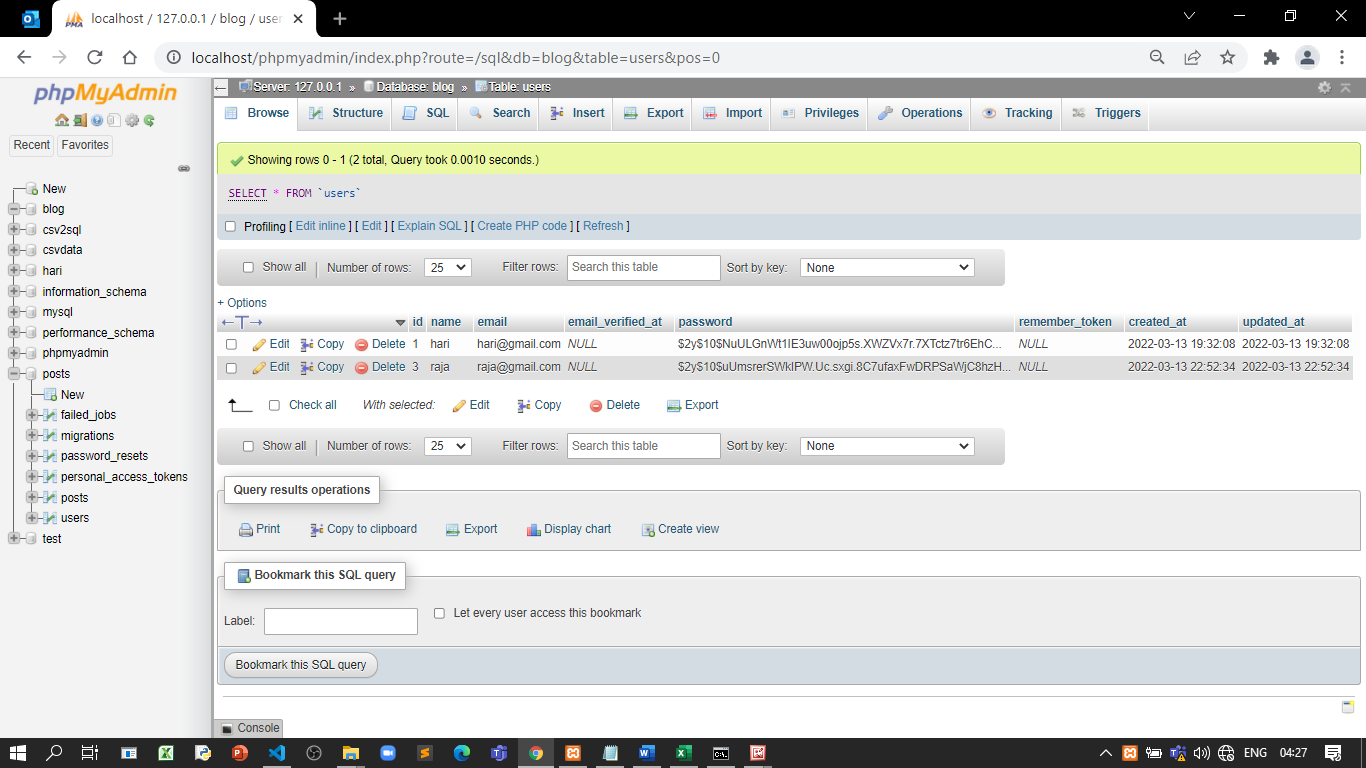












**References:**

* <https://www.nicesnippets.com/blog/laravel-livewire-login-register-example-tutorial>