shani singh
Posted on May 5, 2022

16 5 1 1 1

Laravel API Authentication Using LARAVEL SANCTUM

#laravel #tutorial #beginners #webdev

24 10 19

Make REST API AUTHENTICATION in LARAVEL 9 USING LARAVEL SANCTUM

Laravel Sanctum provides a featherweight authentication system for SPAs (single page applications), mobile applications, and simple, token based APIs.

Installation Steps

If you are not using LARAVEL 9 you need to install LARAVEL Sanctum Otherwise you can skip the installation step.

Step 1

Install via composer

```
composer require laravel/sanctum
```

Step 2

Publish the Sanctum Service Provider

```
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServ
```

Step 3

Migrate The Database

```
php artisan migrate
```

USING SANCTUM IN LARAVEL

User `HasApiTokens` Trait in `App\Models\User`

In Order to use Sanctum we need to use `HasApiTokens` Trait Class in User Model.
User Model should look like.

```
<?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',
    ];
}
```

API Authentication Routes

Create `AuthController` to handle all authentication realted to API

```
php artisan make:controller Api\\AuthController
```

In `routes\api.php` file update the API

```
Route::post('/auth/register', [AuthController::class, 'createUser'
Route::post('/auth/login', [AuthController::class, 'loginUser']);
```

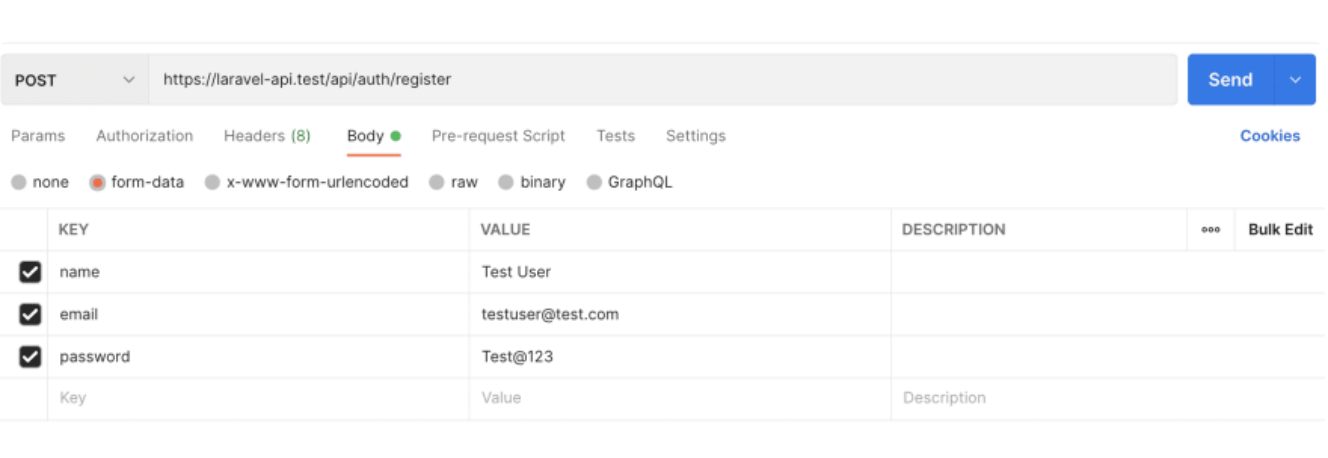
Now update `AuthContoller` with

```
<?php

namespace App\Http\Controllers\Api;

use App\Models\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class AuthController extends Controller
{
    /**
     * Create User
     * @param Request $request
     * @return User
     */
    public function createUser(Request $request)
    {
        try {
            //Validated
            $validateUser = Validator::make($request->all(),
            [
                'name' => 'required',
                'email' => 'required|email|unique:users,email',
                'password' => 'required'
            ]);

            if($validateUser->fails()){
                return response()->json([
                    'status' => false,
                    'message' => 'validation error',
                    'errors' => $validateUser->errors()
                ], 401);
            }

            $user = User::create([
                'name' => $request->name,
                'email' => $request->email,
                'password' => Hash::make($request->password)
            ]);

            return response()->json([
                'status' => true,
                'message' => 'User Created Successfully',
                'token' => $user->createToken("API TOKEN")->plainT
            ], 200);

        } catch (\Throwable $th) {
            return response()->json([
                'status' => false,
                'message' => $th->getMessage()
            ], 500);
        }
    }

    /**
     * Login The User
     * @param Request $request
     * @return User
     */
    public function loginUser(Request $request)
    {
        try {
            $validateUser = Validator::make($request->all(),
            [
                'email' => 'required|email',
                'password' => 'required'
            ]);

            if($validateUser->fails()){
                return response()->json([
                    'status' => false,
                    'message' => 'validation error',
                    'errors' => $validateUser->errors()
                ], 401);
            }

            if(!Auth::attempt($request->only(['email', 'password']
                return response()->json([
                    'status' => false,
                    'message' => 'Email & Password does not match
                ], 401);
            }

            $user = User::where('email', $request->email)->first()

            return response()->json([
                'status' => true,
                'message' => 'User Logged In Successfully',
                'token' => $user->createToken("API TOKEN")->plainT
            ], 200);

        } catch (\Throwable $th) {
            return response()->json([
                'status' => false,
                'message' => $th->getMessage()
            ], 500);
        }
    }
}
```

Protect API With Authentication we need to use `auth:sanctum` middleware.

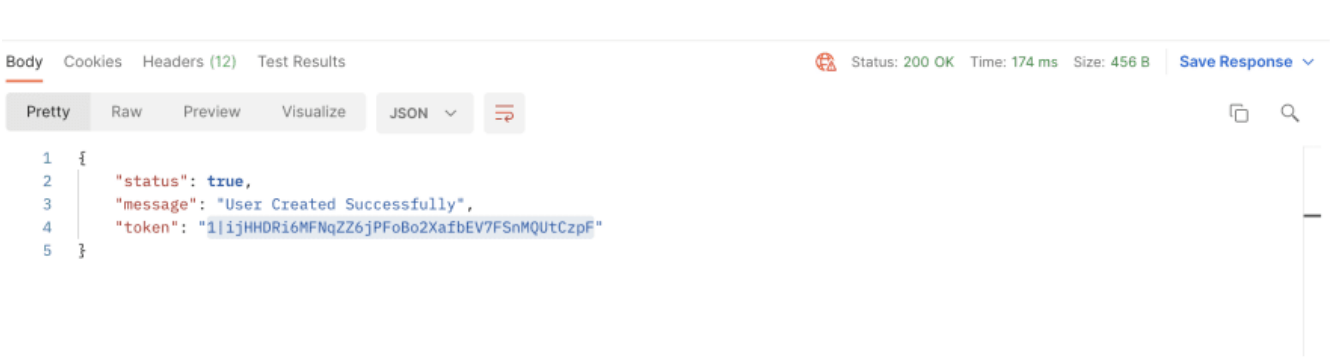
```
Route::apiResource('posts', PostController::class)->middleware('au
```

Here are the results.



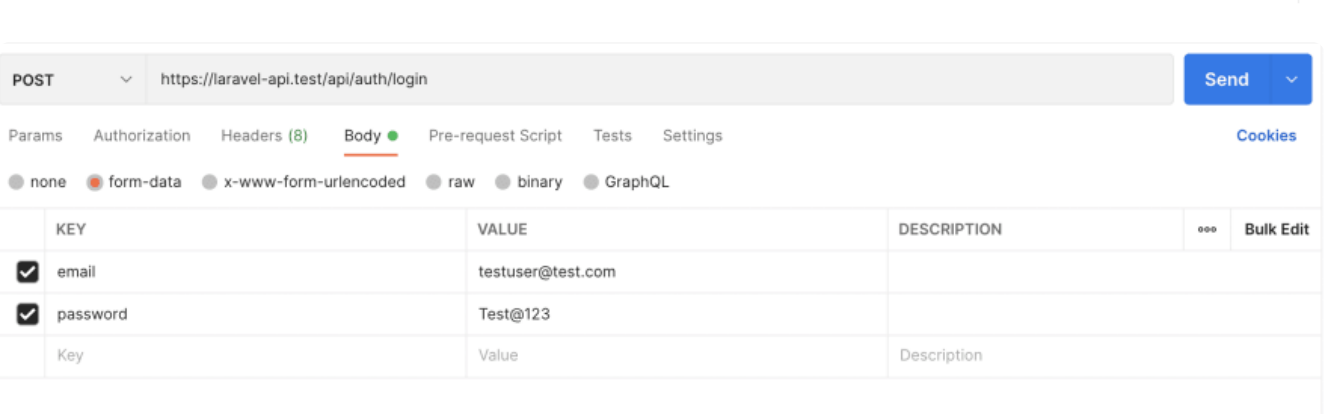
POST https://laravel-api.test/api/auth/register

KEY	VALUE	DESCRIPTION
☑ name	Test User	
☑ email	testuser@test.com	
☑ password	Test@123	
Key	Value	Description



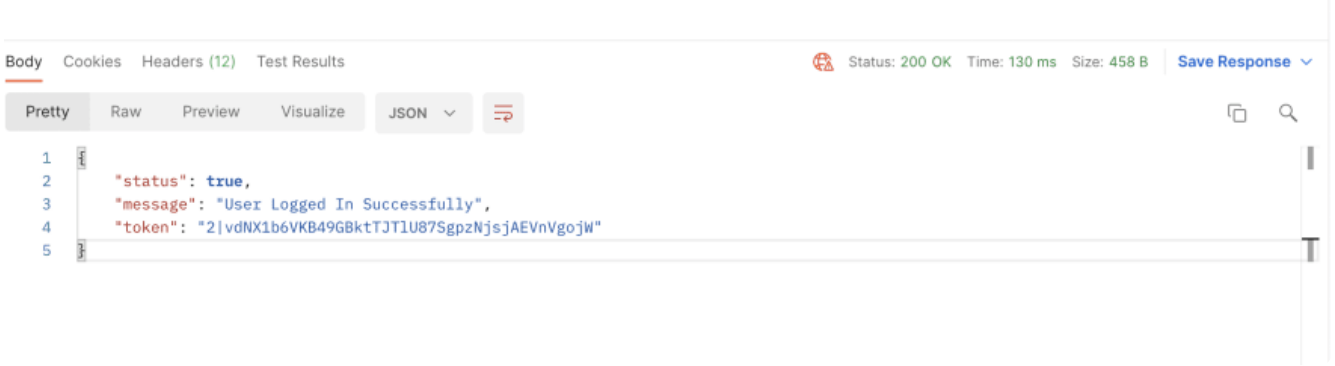
Status: 200 OK Time: 174 ms Size: 456 B

```
{
    "status": true,
    "message": "User Created Successfully",
    "token": "1|ijHHDRi6MFNqZZ6jPFoBo2XafbEV7FSnMQUtCzpF"
}
```



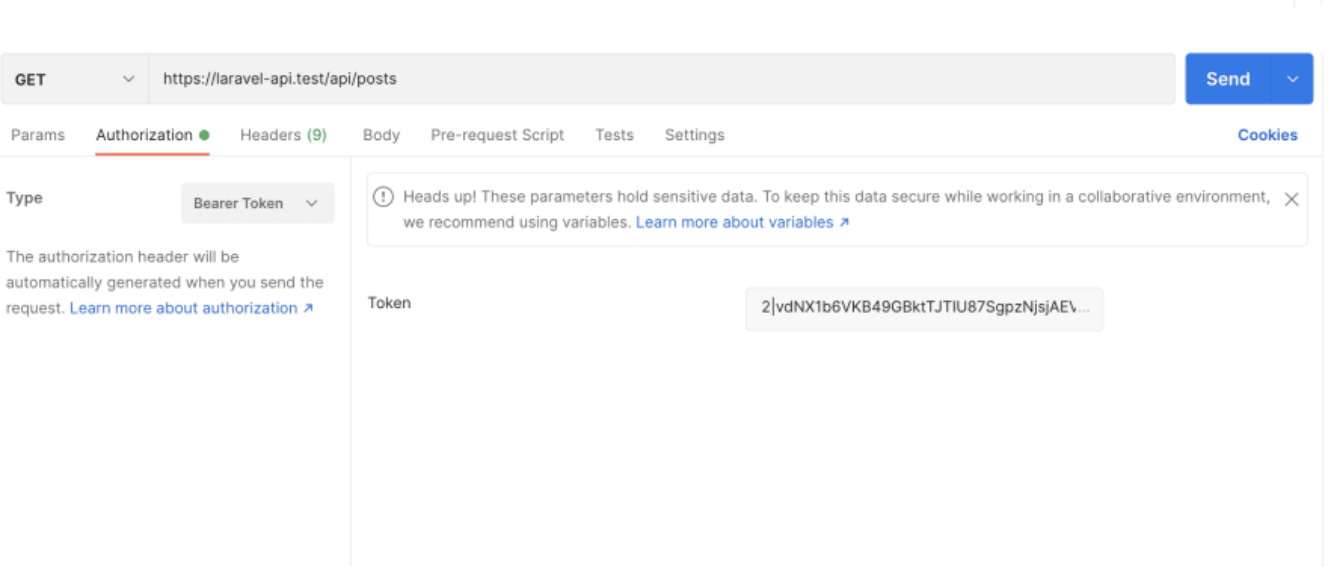
POST https://laravel-api.test/api/auth/login

KEY	VALUE	DESCRIPTION
☑ email	testuser@test.com	
☑ password	Test@123	
Key	Value	Description



Status: 200 OK Time: 130 ms Size: 458 B

```
{
    "status": true,
    "message": "User Logged In Successfully",
    "token": "2|vdNX1b6VKB49GBktTJTlU87SgpzNjsjAEVnVgojW"
}
```



GET https://laravel-api.test/api/posts

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

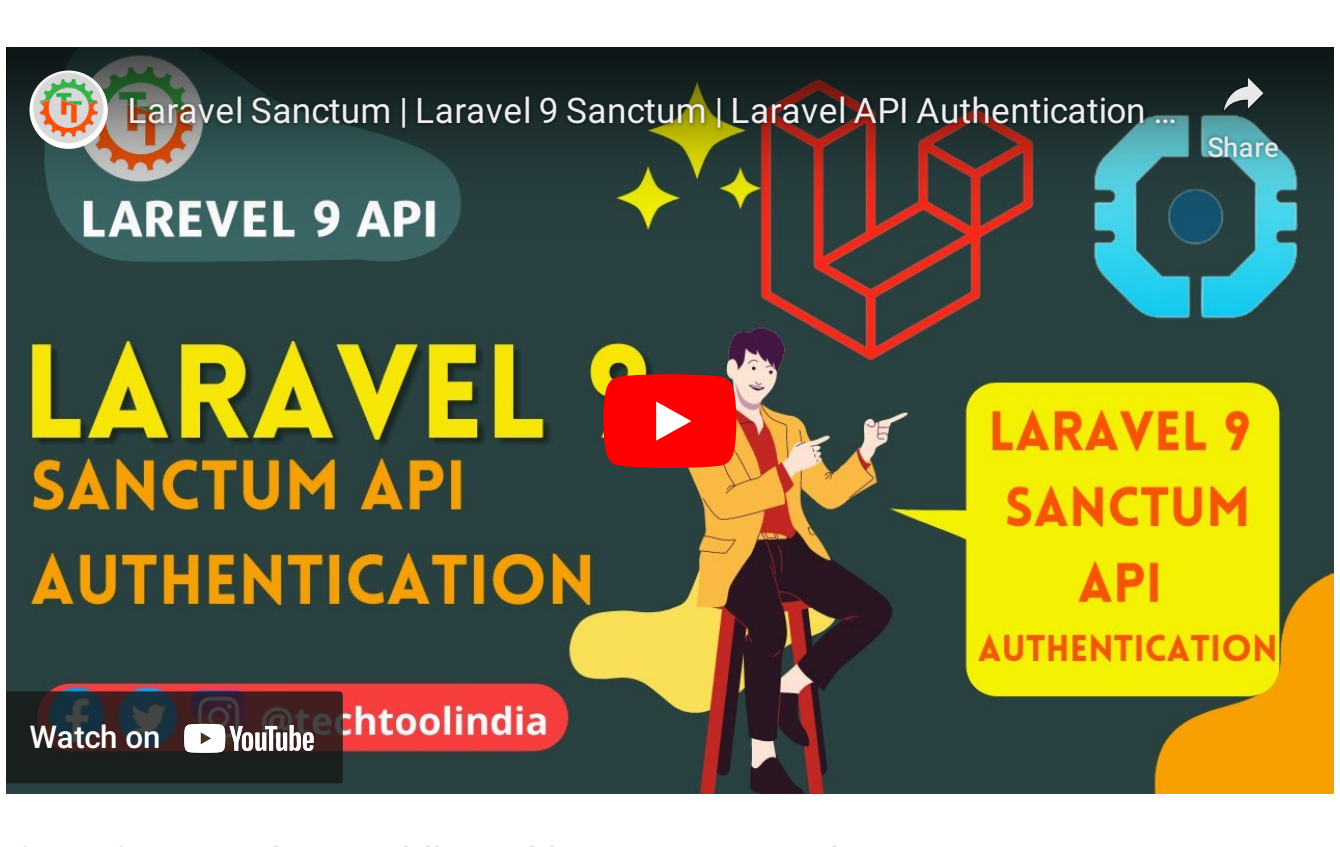
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token: 2|vdNX1b6VKB49GBktTJTlU87SgpzNjsjAEV...

Status: 200 OK Time: 40 ms Size: 691 B

```
{
    "status": true,
    "posts": [
        {
            "id": 2,
            "title": "This is first post",
            "description": "Test Post Only.",
            "created_at": "2022-03-30T07:56:18.000000Z",
            "updated_at": "2022-03-30T07:56:18.000000Z"
        },
        {
            "id": 3,
            "title": "This is first post 2",
            "description": "Test Post Only. 2",
            "created_at": "2022-03-30T07:56:57.000000Z",
            "updated_at": "2022-03-30T07:56:57.000000Z"
        }
    ]
}
```

The complete Tutorial is below in the video.



If you face any issue while making REST API, please comment your query.

Thank You for Reading

Reach Out To me.
[Twitter]
[Instagram]
[TechToolIndia YouTube Channel]

Top comments (10)

Subscribe

Add to the discussion

Umar Adil • Nov 24 '22

Hi Shani, thanks for this tutorial. I want to list the all users I have in DB and don't want this API to be public. How do I secure this api using sanctum. To access this DB no user logs in becuase this api is used to select the users only.

2 likes Reply

shani singh • Dec 13 '22

You can write a new middleware or use Auth Gaurd for Securing the API.

1 like Reply

Ahmad9250 • Jan 20

Hi can you make an login page on Laravel blade template and login through API post method with same email and password that was saved in db

2 likes Reply

shani singh • Jan 28

Hi @ahmad9250
You can connect me at my email shanisingh280795@gmail.com , We can discuss there.
Thanks

1 like Reply

Muhammad Faisal • Oct 28 '22

Hey, good article short and concise!

But may I ask about other uses of this personal access token?
If I may, I will ask point by point if you are willing to enlighten me..

1. We always generate token with code `createToken("API TOKEN")->plainTextToken` and input it in database table personal_access_tokens when we login/register right?

- 1.1. So, is the use of `auth:sanctum` only to provide a limit on `Route::middleware('auth:sanctum')->group(function () { });` ??
- 1.2. If not, what are the **other uses** of these `personal_access_tokens`?

1. After we successfully generate the token and send it as an authorization token in the header like in your Postman image, it will work and send the response as we want right?

- 2.1. If we send the wrong token code, this will give a response in the form of html code in Postman, which cannot be read in the API response. So, the question is **how to check and provide** a more understandable error response?

2 likes Reply

shani singh • Oct 29 '22

Hi
So use of Personal Access token is to identify users identity and based on that it gives you response, and to get JSON response you can define request type JSON then Laravel API will always give you JSON Response.
in Addition you can write exception handler condition and format the response as well.
hope this will clear your thoughts.

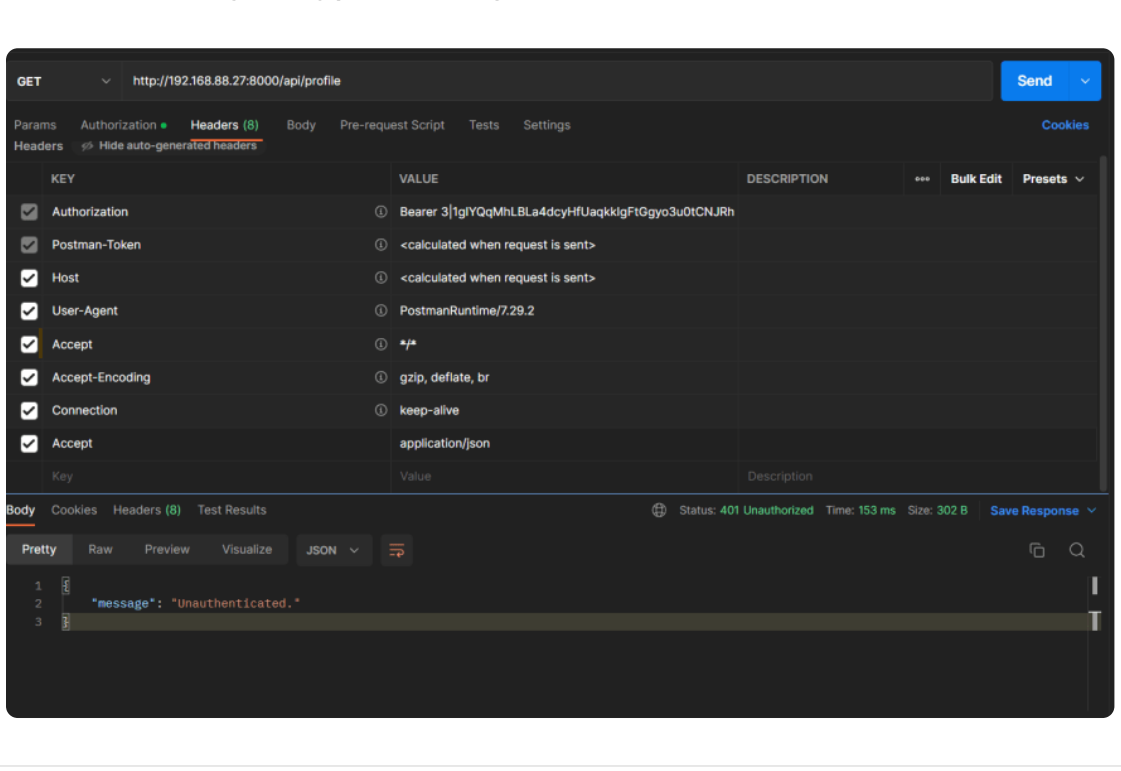
2 likes Reply

Muhammad Faisal • Nov 2 '22 • Edited on Nov 2

Whoa, thanks for the reply!
I've been trying for days to understand everything by reading the documentation from Laravel, also asking on Stackoverflow, but in the end I just understand that Personal Access Tokens are just User ID identifiers, and NOT work like a session security, right?

By the way, thanks for the explanation on the first point.

The second, thanks for pointing out about the json type acceptance and now i can json type message like this



GET http://192.168.88.27:8000/api/profile

KEY	VALUE	DESCRIPTION
☑ Authorization	Bearer 3|1gIYQqMhLBLa4dcyHfUaqkklgFtGgyo3u0tCNJRh	
☑ Postman-Token	<calculated when request is sent>	
☑ Host	<calculated when request is sent>	
☑ User-Agent	PostmanRuntime/7.29.2	
☑ Accept	*/*	
☑ Accept-Encoding	gzip, deflate, br	
☑ Connection	keep-alive	
☑ Accept	application/json	
Key	Value	Description

Status: 401 Unauthorized Time: 153 ms Size: 302 B

```
{
    "message": "Unauthenticated."
}
```

Reply

Rama-Eisawi • Jan 1 '23

Hello, thank you so much
I code exactly like you, but this error is shown when I try the login query in postman:"message": "SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'tokenable_id' cannot be null"
please help me :(

1 like Reply

Tanveer Qureshee • Nov 8 '22

Dear Shani singhn
First of all thank you for the great articles.

I have a question and if you can help me it will be great.
I am useing csrf-cookie base authentication not token based
But i am unable to test api from post man, or swagger due to header cookies check

have you any advice or solution for me?
Please reply me.

Reply

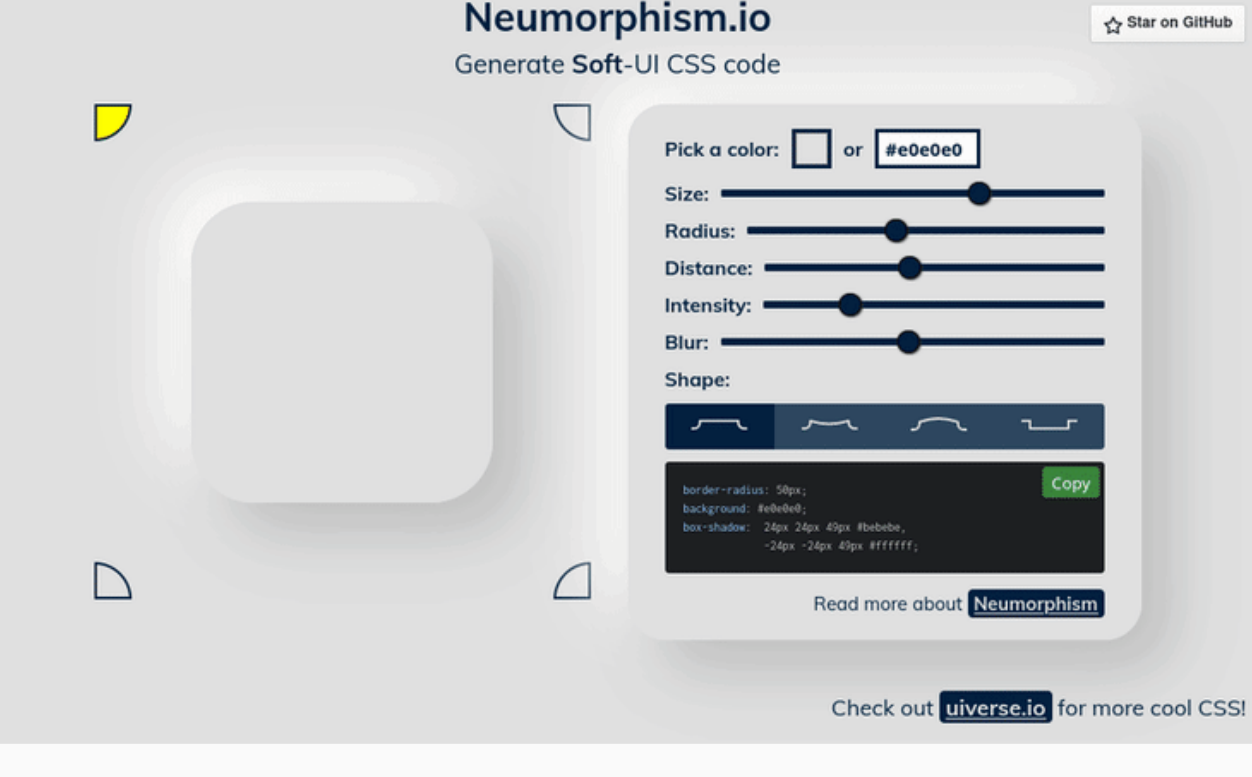
shani singh • Nov 15 '22

Hi Tanveer,
laravel.com/docs/9.x/sanctum#csrf-...

This can help you setting up csrf-cookie.
Thanks

1 like Reply

Code of Conduct • Report abuse



shani singh

Follow

FullStack Developer, PHP, LARAVEL, VueJs, React, Shopify, Shopify Plus, Flutter.

LOCATION
Nagpur, Maharashtra, India

EDUCATION
B.E.

WORK
CEO & MD at Sentriqo IT Solutions Pvt Ltd

JOINED
Feb 18, 2020

More from shani singh

- Laravel 9 - Eloquent Factory & Database Seeder Tutorial
  #laravel #php #webdev #programming
- Laravel Password less Login using Magic Link.
  #laravel #programming #tutorial #webdev
- How to login with OTP in Laravel 9 ?
  #laravel #webdev #beginners #tutorial

Read next

- **Next.js + MDX Blog | CodeSamples.dev**
  Shibu N - Apr 16
- **Weekend Reading**
  Simon Porter - Apr 15
- **How to Deploy a Next.js 13 App to AWS with Amplify Hosting**
  Femi Akinyemi - May 4
- **HTMX is the future**
  Chris James - May 5