SACHIN HESHAN MUNASINGHE

Documentation for Testing

Applying for the position of Senior Software Engineer

Capricon Solutions (Private) Limited

Table of Contents

Why Testing.

Software testing is a critical process that involves evaluating and ensuring that a software product or application functions as intended. Its benefits are substantial, encompassing bug prevention, cost reduction, and performance enhancement. Several key reasons highlight the significance of testing for software companies:

- ➢ Early Issue Detection:

Testing allows for the early detection of potential issues or defects in the software. By identifying problems at their nascent stages, developers can rectify them more easily and cost-effectively. This early detection also contributes to a more efficient development process.

- ➢ Enhanced Product Quality:

Testing plays a pivotal role in improving the quality of both the Minimum Viable Product (MVP) and the final product. It helps ensure that the software meets its intended functionality, adheres to quality standards, and performs reliably. This quality enhancement is crucial for building trust among users.

- ➢ Flaw Identification:

Testing assists in the identification and resolution of flaws within specific software components or the entire system. It helps in pinpointing weaknesses, vulnerabilities, and inconsistencies that may compromise the software's performance or security. Addressing these flaws is vital for delivering a robust and dependable product.

Testing can be a time-consuming process, and its approach may vary depending on the size and complexity of the software. For smaller projects, manual or ad-hoc testing may suffice. However, for larger and more complex systems, automation tools are commonly employed. Automated testing offers several advantages, such as the ability to execute various test scenarios, assess the impact of changes (e.g., migrating components to a cloud environment), and provide rapid feedback on the software's functionality and reliability. This automation significantly contributes to the efficiency and effectiveness of the testing process, especially in the context of modern software development where continuous integration and delivery are prevalent.

What is Laravel Unit Testing

PHPUnit is a widely recognized and highly optimized unit testing framework for PHP. It is a popular choice among developers for identifying and addressing various development issues in applications. Its primary purpose is to facilitate thorough unit testing of PHP code, allowing developers to scrutinize their code in detail. However, PHPUnit's flexibility extends beyond just unit testing, making it a versatile tool for a range of testing functions. It's especially well-regarded for its compatibility with major PHP frameworks like Laravel, and many experienced developers recommend it for Laravel unit testing due to its robust testing standards

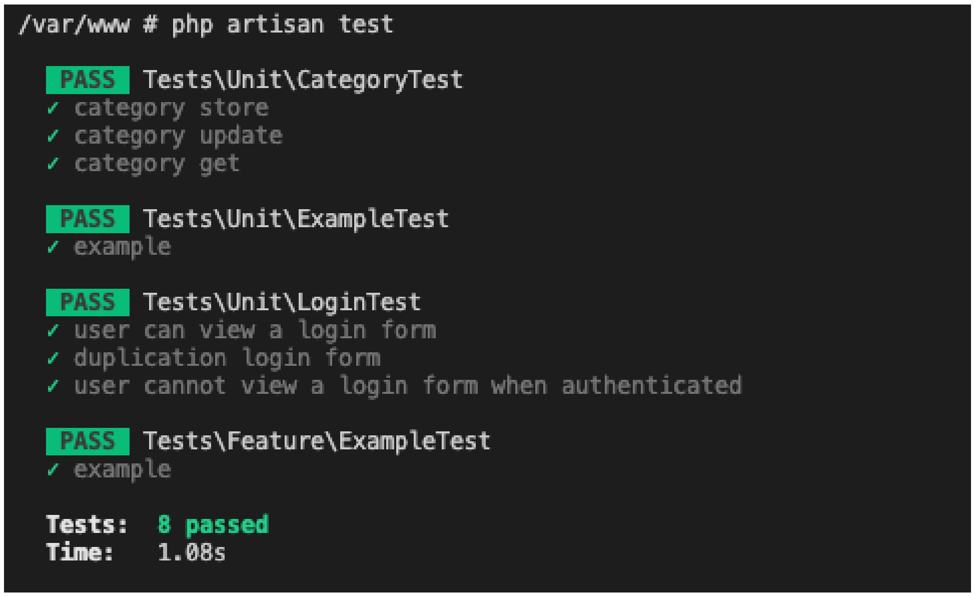
In summary, PHPUnit is a powerful tool for PHP developers, offering a strong focus on unit testing and straightforward assertions. It's highly regarded for its compatibility with PHP frameworks, such as Laravel, and its ability to thoroughly test individual code units. However, when dealing with more complex components, developers may need to invest additional effort in setting up comprehensive tests.

What is Postman Testing

API testing is the practice of validating that an Application Programming Interface (API) functions as intended. APIs are essential for enabling communication and data exchange between different software components, and API testing ensures that these interactions are reliable and error-free. In summary, API testing is essential for verifying the functionality and reliability of APIs. It can be performed manually or automated, and various types of tests address different aspects of the API's behavior. The shift-left approach to API testing promotes early detection and resolution of issues, enabling teams to deliver more robust and reliable APIs while maintaining a faster development cycle.

Testing results for the project using Laravel Unit Testing

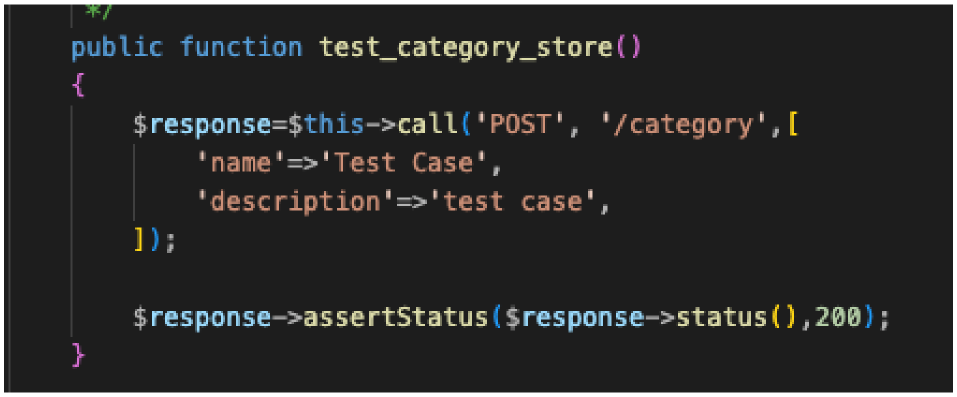
Testing all the functions created in the project and 8 tests are passed.



```
/var/www # php artisan test

   PASS  Tests\Unit\CategoryTest
  ✓ category store
  ✓ category update
  ✓ category get

   PASS  Tests\Unit\ExampleTest
  ✓ example

   PASS  Tests\Unit\LoginTest
  ✓ user can view a login form
  ✓ duplication login form
  ✓ user cannot view a login form when authenticated

   PASS  Tests\Feature\ExampleTest
  ✓ example

  Tests:  8 passed
  Time:   1.08s
```

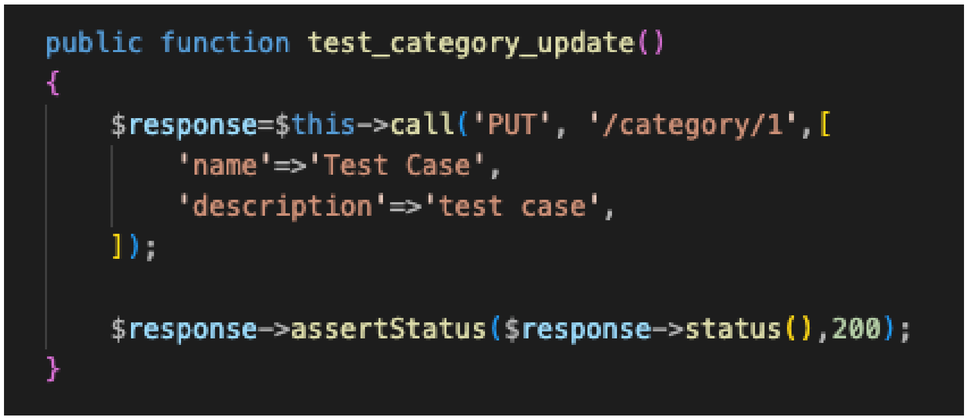
And I have attached all the test cases for all the functions that coded. And every test case is compiled.

- Insert of the blogs-



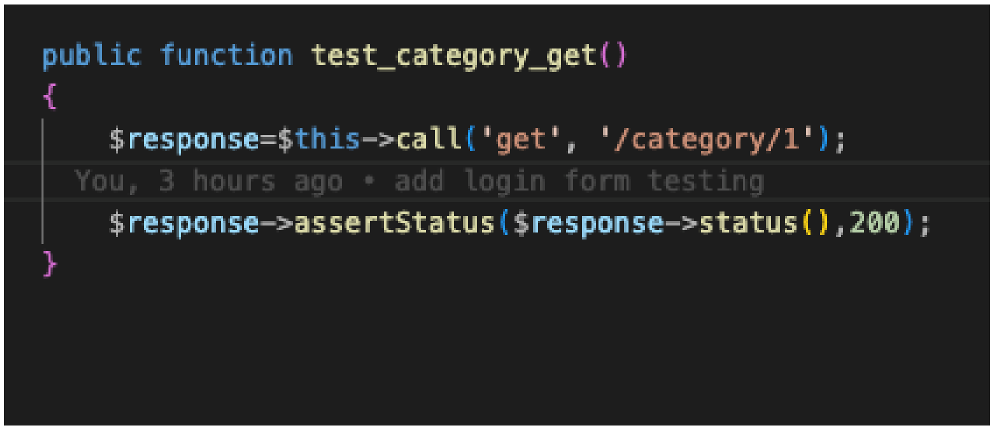
```
    */
    public function test_category_store()
    {
        $response=$this->call('POST', '/category',[
            'name'=>'Test Case',
            'description'=>'test case',
        ]);

        $response->assertStatus($response->status(),200);
    }
```

- Update of the blog that already added to the DB –



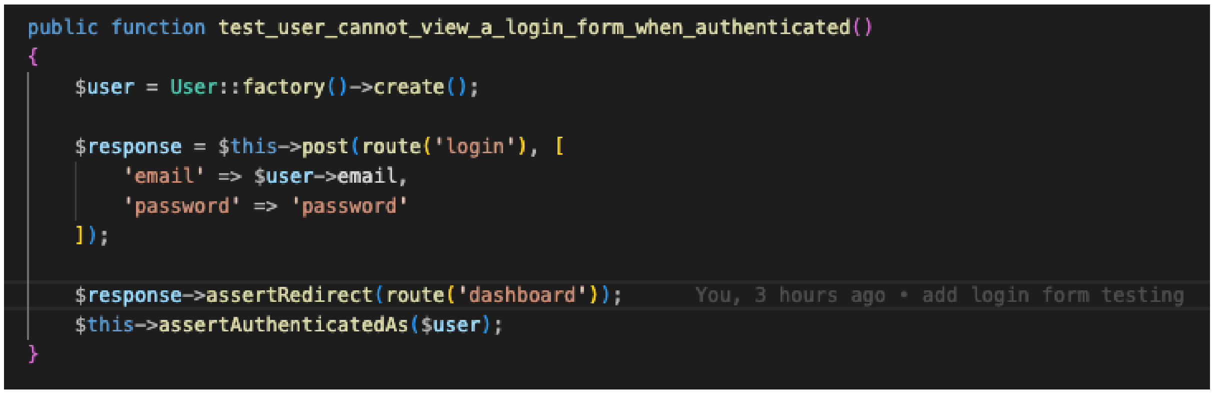
```
public function test_category_update()
{
    $response=$this->call('PUT', '/category/1',[
        'name'=>'Test Case',
        'description'=>'test case',
    ]);

    $response->assertStatus($response->status(),200);
}
```

- Retrieve the Blogs from the DB –



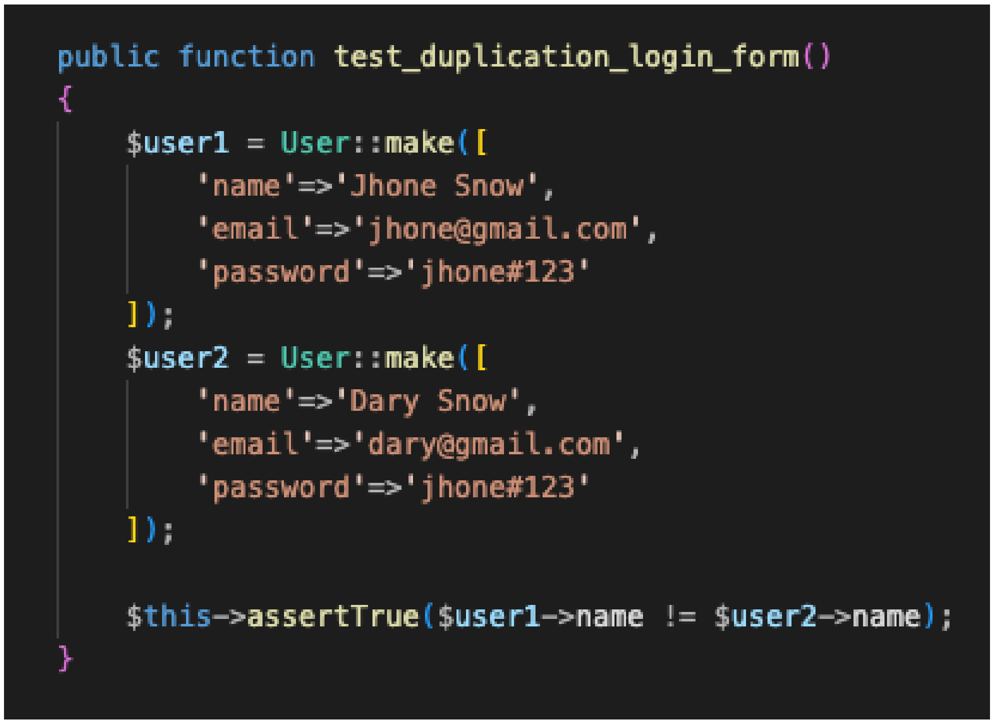
```
public function test_category_get()
{
    $response=$this->call('get', '/category/1');
  You, 3 hours ago • add login form testing
    $response->assertStatus($response->status(),200);
}
```

- Check whether when the login form is authenticated user can not view it –



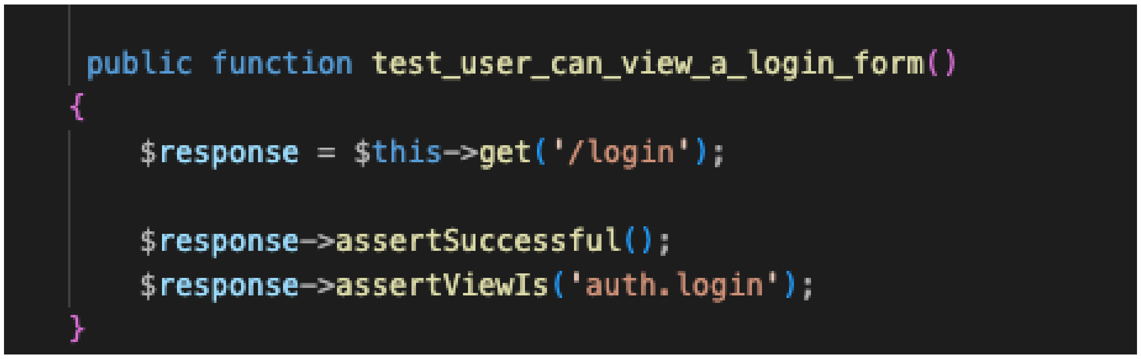
```
public function test_user_cannot_view_a_login_form_when_authenticated()
{
    $user = User::factory()->create();

    $response = $this->post(route('login'), [
        'email' => $user->email,
        'password' => 'password'
    ]);

    $response->assertRedirect(route('dashboard'));       You, 3 hours ago • add login form testing
    $this->assertAuthenticatedAs($user);
}
```

- Check whether the duplicate data is going to enter the user –



```
public function test_duplication_login_form()
{
    $user1 = User::make([
        'name'=>'Jhone Snow',
        'email'=>'jhone@gmail.com',
        'password'=>'jhone#123'
    ]);
    $user2 = User::make([
        'name'=>'Dary Snow',
        'email'=>'dary@gmail.com',
        'password'=>'jhone#123'
    ]);

    $this->assertTrue($user1->name != $user2->name);
}
```

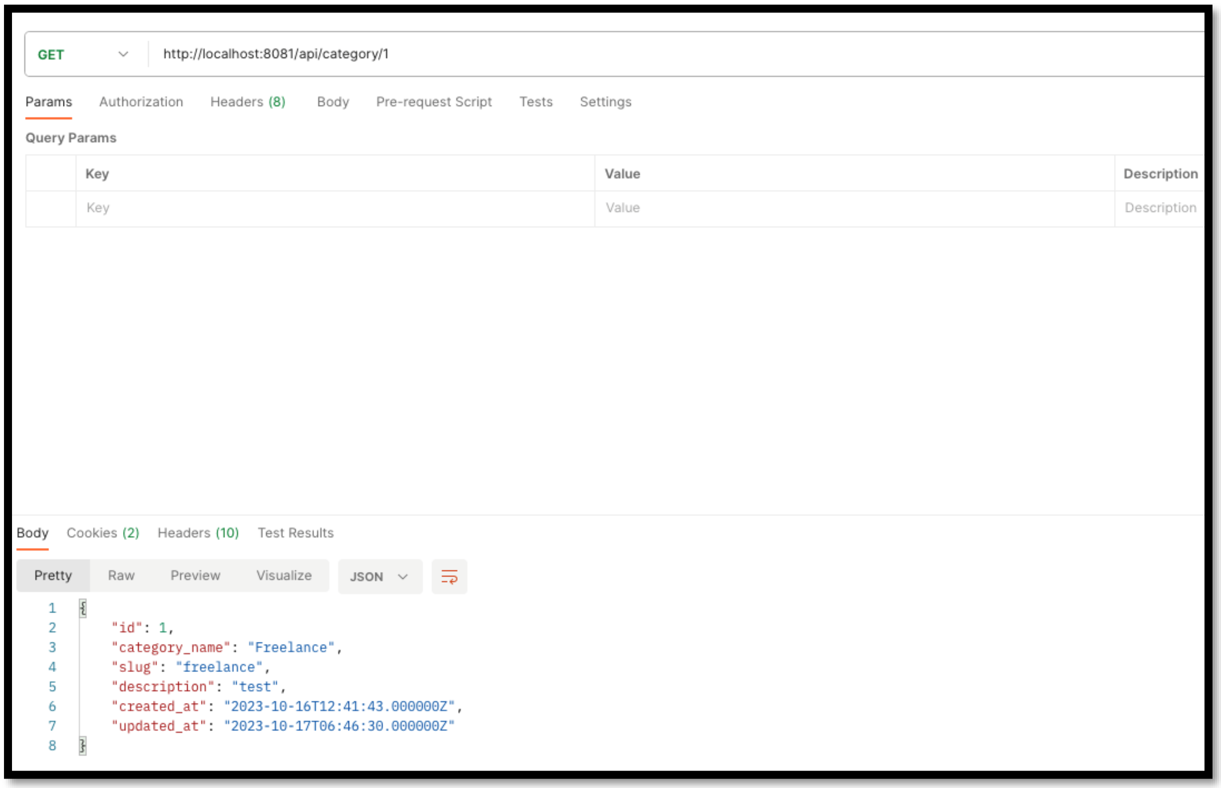
- Check whether the user can view the login form



```
 public function test_user_can_view_a_login_form()
{
    $response = $this->get('/login');

    $response->assertSuccessful();
    $response->assertViewIs('auth.login');
}
```

Testing results for the project using Postman Testing

Test whether the blogs are updating through the backend using postman and its success.



GET http://localhost:8081/api/category/1

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

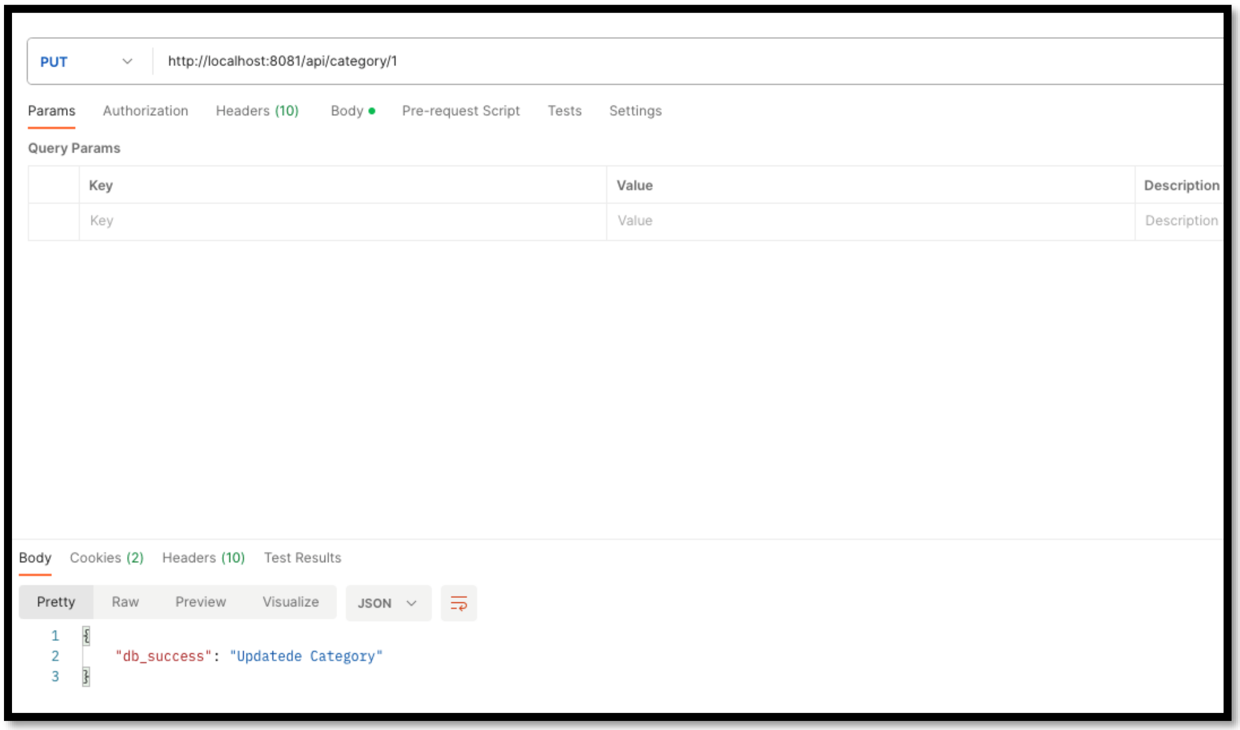
Query Params

	Key	Value	Description
	Key	Value	Description

Body | Cookies (2) | Headers (10) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": 1,
    "category_name": "Freelance",
    "slug": "freelance",
    "description": "test",
    "created_at": "2023-10-16T12:41:43.000000Z",
    "updated_at": "2023-10-17T06:46:30.000000Z"
}
```



PUT http://localhost:8081/api/category/1

Params | Authorization | Headers (10) | Body ● | Pre-request Script | Tests | Settings

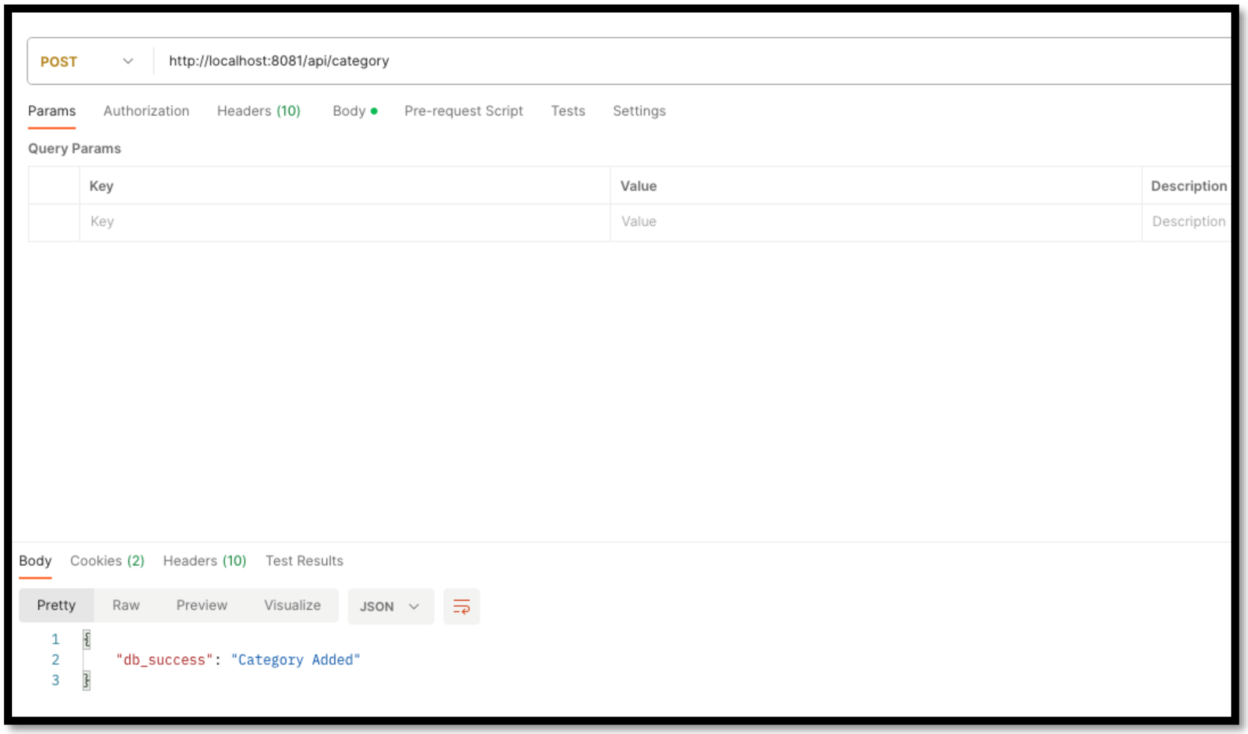
Query Params

	Key	Value	Description
	Key	Value	Description

Body | Cookies (2) | Headers (10) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "db_success": "Updatede Category"
}
```


POST http://localhost:8081/api/category

Params | Authorization | Headers (10) | Body ● | Pre-request Script | Tests | Settings

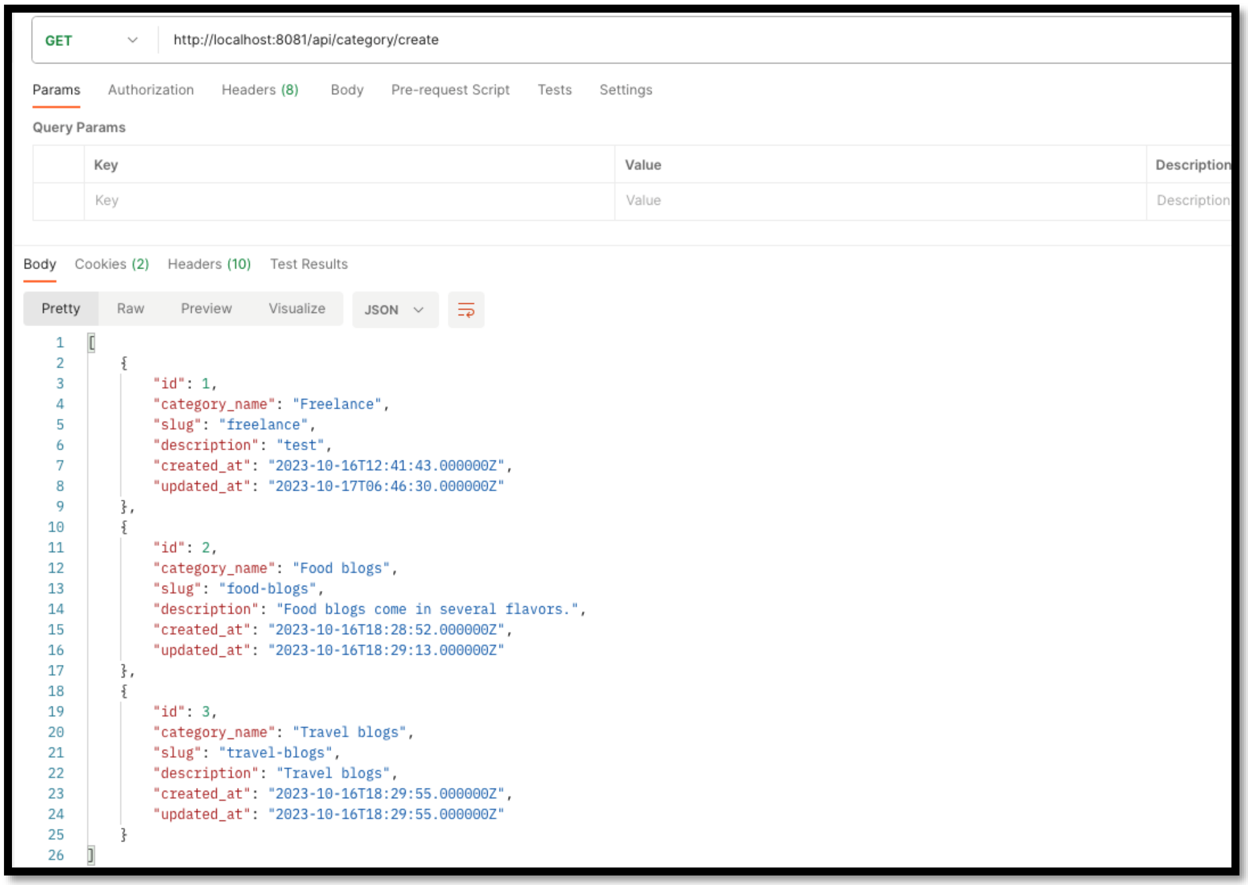
Query Params

	Key	Value	Description
	Key	Value	Description

Body | Cookies (2) | Headers (10) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "db_success": "Category Added"
}
```


GET http://localhost:8081/api/category/create

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body | Cookies (2) | Headers (10) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "id": 1,
        "category_name": "Freelance",
        "slug": "freelance",
        "description": "test",
        "created_at": "2023-10-16T12:41:43.000000Z",
        "updated_at": "2023-10-17T06:46:30.000000Z"
    },
    {
        "id": 2,
        "category_name": "Food blogs",
        "slug": "food-blogs",
        "description": "Food blogs come in several flavors.",
        "created_at": "2023-10-16T18:28:52.000000Z",
        "updated_at": "2023-10-16T18:29:13.000000Z"
    },
    {
        "id": 3,
        "category_name": "Travel blogs",
        "slug": "travel-blogs",
        "description": "Travel blogs",
        "created_at": "2023-10-16T18:29:55.000000Z",
        "updated_at": "2023-10-16T18:29:55.000000Z"
    }
]
```

THANK YOU!!