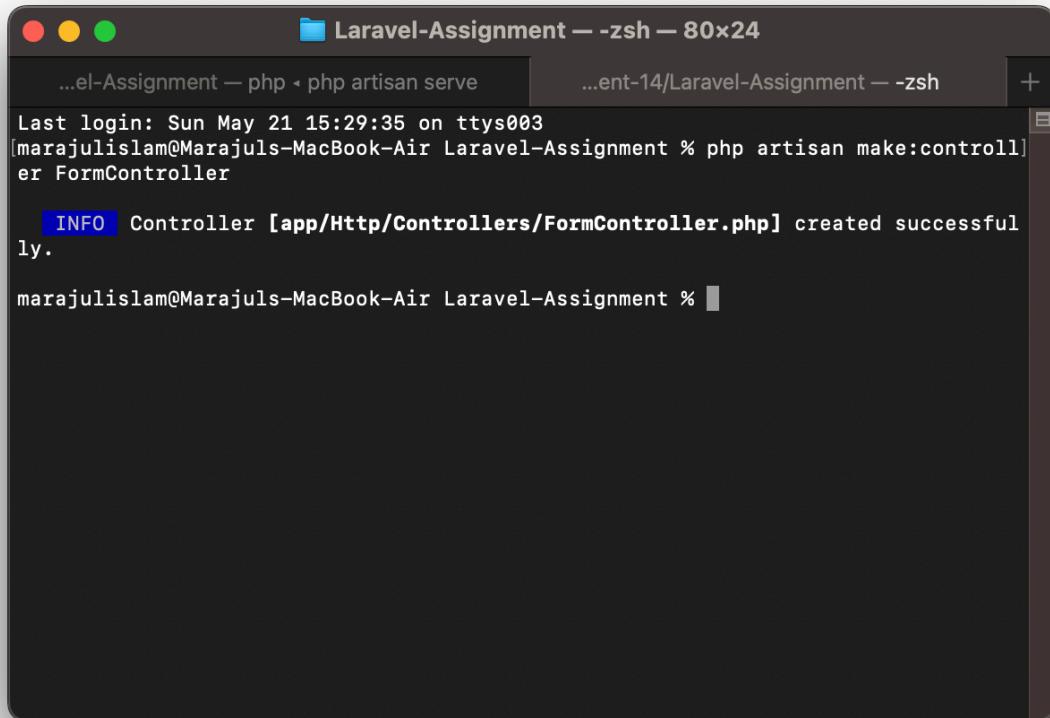


## **Module-14 Laravel Foundation Assignment:**

**Question 1: You have a Laravel application with a form that submits user information using a POST request. Write the code to retrieve the 'name' input field value from the request and store it in a variable called \$name.**

### **Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command: `php artisan make:controller FormController`



```
Laravel-Assignment — zsh — 80x24
...el-Assignment — php < php artisan serve      ...ent-14/Laravel-Assignment — zsh
Last login: Sun May 21 15:29:35 on ttys003
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller FormController
INFO Controller [app/Http/Controllers/FormController.php] created successfully.

marajulislam@Marajuls-MacBook-Air Laravel-Assignment %
```

### **Step 2: Implement the controller method**

A screenshot of a code editor window titled "FormController.php — Practice Laravel". The file contains the following PHP code:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class FormController extends Controller
8 {
9     public function submitForm(Request $request)
10    {
11        $name = $request->input('name');
12
13        return response()->json(['name' => value($name)]);
14    }
15 }
16
```

### Step 3: Define the route

A screenshot of a code editor window titled "web.php — Practice Laravel". The file contains the following PHP code:

```
21 //Submits user information:
22 Route::post('/submit-form', [FormController::class, 'submitForm']);
```

### Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

### Step 5: Test the route using Postman

The screenshot shows the Postman application interface. A test collection named 'Laravel / Test' is selected. A POST request is being made to the URL `http://127.0.0.1:8000/submit-form`. The request body is set to 'form-data' and contains a single field 'name' with the value 'Junaid'. The response status is 200 OK, and the response body is displayed as:

```
1 "name": "Junaid"
```

**Question 2: In your Laravel application, you want to retrieve the value of the 'User-Agent' header from the current request. Write the code to accomplish this and store the value in a variable called \$ userAgent.**

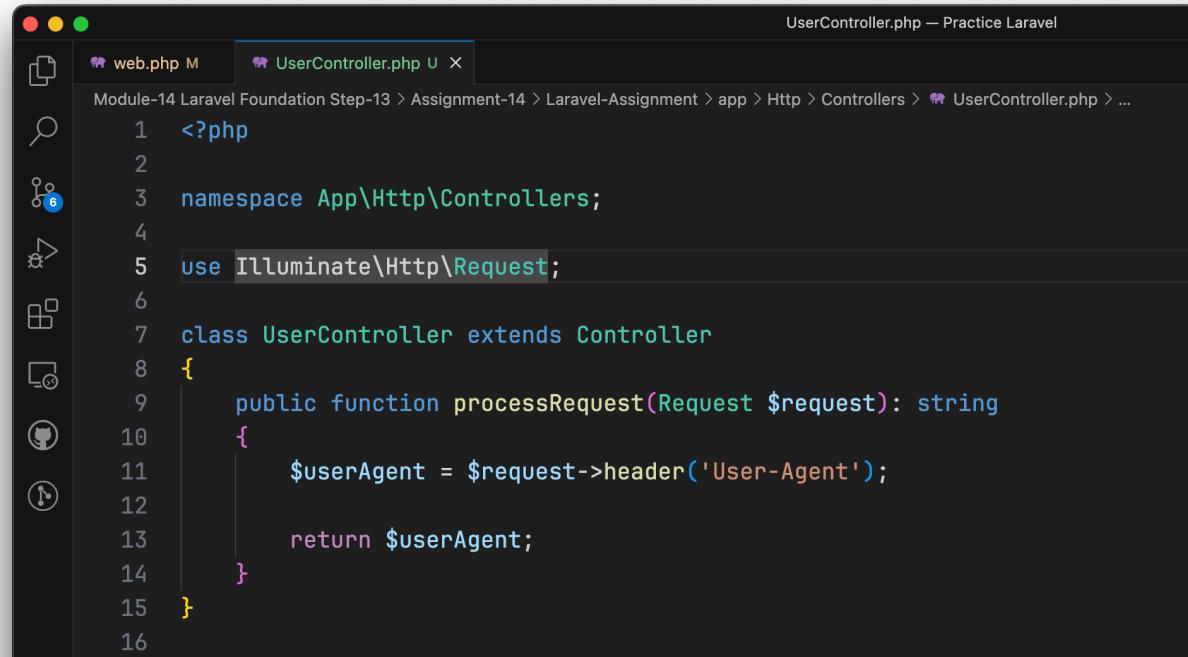
### **Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command:`php artisan make:controller UserController`

```
Last login: Sun May 21 15:29:48 on ttys003
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller UserController
INFO Controller [app/Http/Controllers/UserController.php] created successfully.

marajulislam@Marajuls-MacBook-Air Laravel-Assignment % ]
```

## Step 2: Implement the controller method



The screenshot shows a code editor window titled "UserController.php — Practice Laravel". The file content is as follows:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class UserController extends Controller
8 {
9     public function processRequest(Request $request): string
10    {
11        $userAgent = $request->header('User-Agent');
12
13        return $userAgent;
14    }
15 }
16
```

### Step 3: Define the route



The screenshot shows a code editor window titled "web.php — Practice Laravel". The file content is as follows:

```
24
25 //User-Agent:
26 Route::post('/process-request', [UserController::class, 'processRequest']);
27
```

### Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

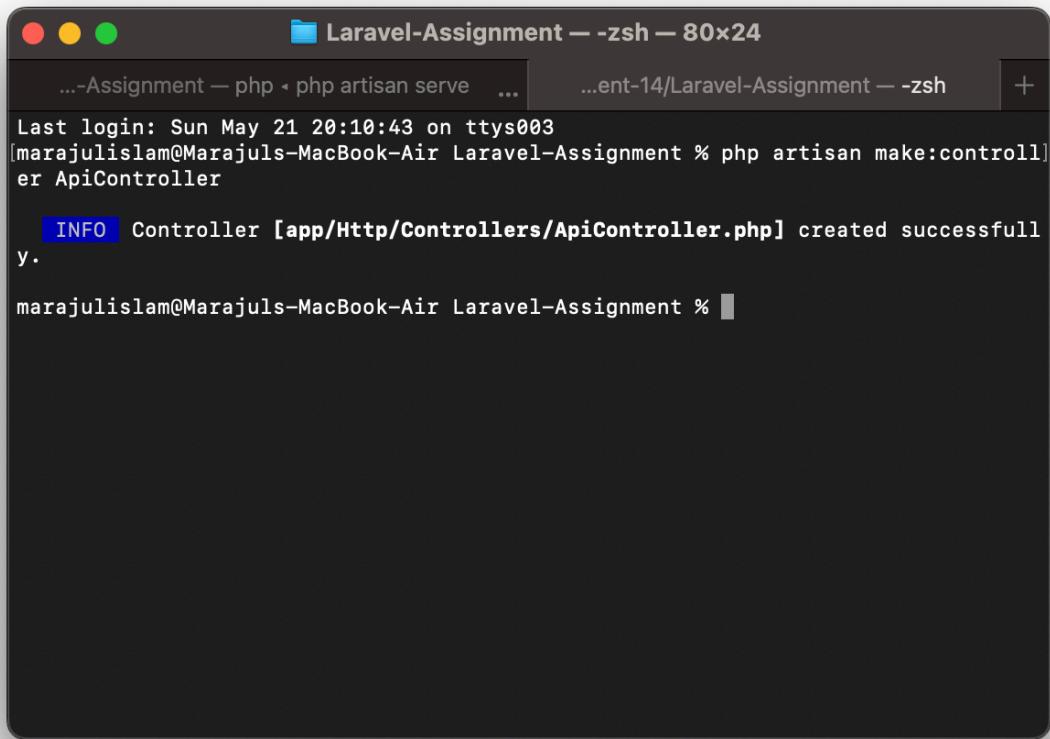
### Step 5: Test the route using Postman

The screenshot shows the Postman application interface. A test collection named "Laravel / Test" is selected. A POST request is being prepared to the endpoint `http://127.0.0.1:8000/process-request`. In the Headers tab, there is one explicitly defined header: `User-Agent` with the value `MyCustomUserAgent`. The Body tab shows a simple JSON payload with one key, `1`, which has the value `MyCustomUserAgent`. The status bar at the bottom indicates a successful `200 OK` response with a time of `51 ms` and a size of `707 B`.

**Question 3: You are building an API endpoint in Laravel that accepts a GET request with a 'page' query parameter. Write the code to retrieve the value of the 'page' parameter from the current request and store it in a variable called \$page. If the parameter is not present, set \$page to null.**

#### **Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command:`php artisan make:controller ApiController`



The screenshot shows a macOS terminal window titled "Laravel-Assignment — zsh — 80x24". The window has three tabs: "...Assignment — php & php artisan serve ...", "...ent-14/Laravel-Assignment — zsh", and a new tab indicator (+). The terminal output is as follows:

```
Last login: Sun May 21 20:10:43 on ttys003
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller ApiController
INFO Controller [app/Http/Controllers/ApiController.php] created successfully.

marajulislam@Marajuls-MacBook-Air Laravel-Assignment %
```

## Step 2: Implement the controller method

```
ApiController.php — Practice Laravel
api.php M ApiController.php U X
Module-14 Laravel Foundation Step-13 > Assignment-14 > Laravel-Assignment > app > Http > Controllers > ApiController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ApiController extends Controller
8 {
9     public function myEndpoint(Request $request)
10    {
11        $page = $request->query('page', null);
12
13        return response()->json(['page' => $page]);
14    }
15 }
16
```

### Step 3: Define the route

```
api.php — Practice Laravel
api.php M ApiController.php U X
Module-14 Laravel Foundation Step-13 > Assignment-14 > Laravel-Assignment > routes > api.php > ...
22
23 //Api Endpoint:
24 Route::get('/my-api', [ApiController::class, 'myEndpoint']);
25
```

### Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

### Step 5: Test the route using Postman

1. Without Parameter:

GET Test

Laravel / Test

GET http://127.0.0.1:8000/api/my-api

Headers (7)

Key	Value	Description
Key	Value	Description

Body

Pretty	Raw	Preview	Visualize	JSON
1	"page": null			
2				
3				

Status: 200 OK Time: 40 ms Size: 320 B

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 "page": null
```

Runner Capture requests Cookies Trash

## 1. With Parameter:

GET Test

Laravel / Test

GET http://127.0.0.1:8000/api/my-api?page=2

Headers (7)

Key	Value	Description
Key	Value	Description

Body

Pretty	Raw	Preview	Visualize	JSON
1	"page": "2"			
2				
3				

Status: 200 OK Time: 39 ms Size: 319 B

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 "page": "2"
```

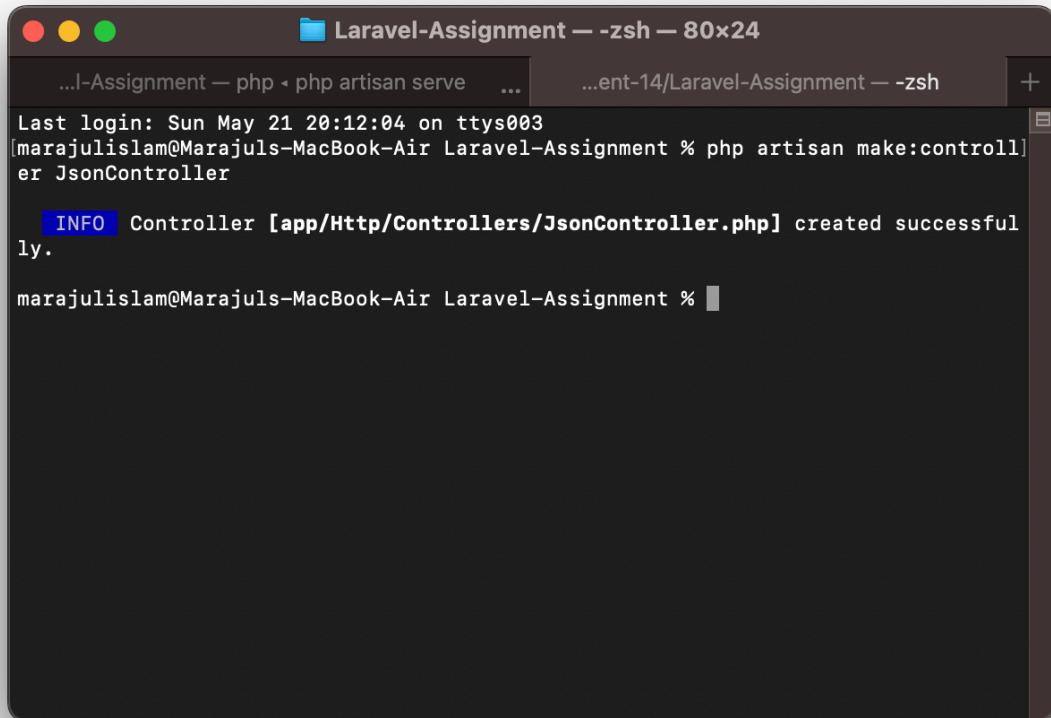
Runner Capture requests Cookies Trash

**Question 4: Create a JSON response in Laravel with the following data:**

```
{  
  "message": "Success",  
  "data": {  
    "name": "John Doe",  
    "age": 25  
  }  
}
```

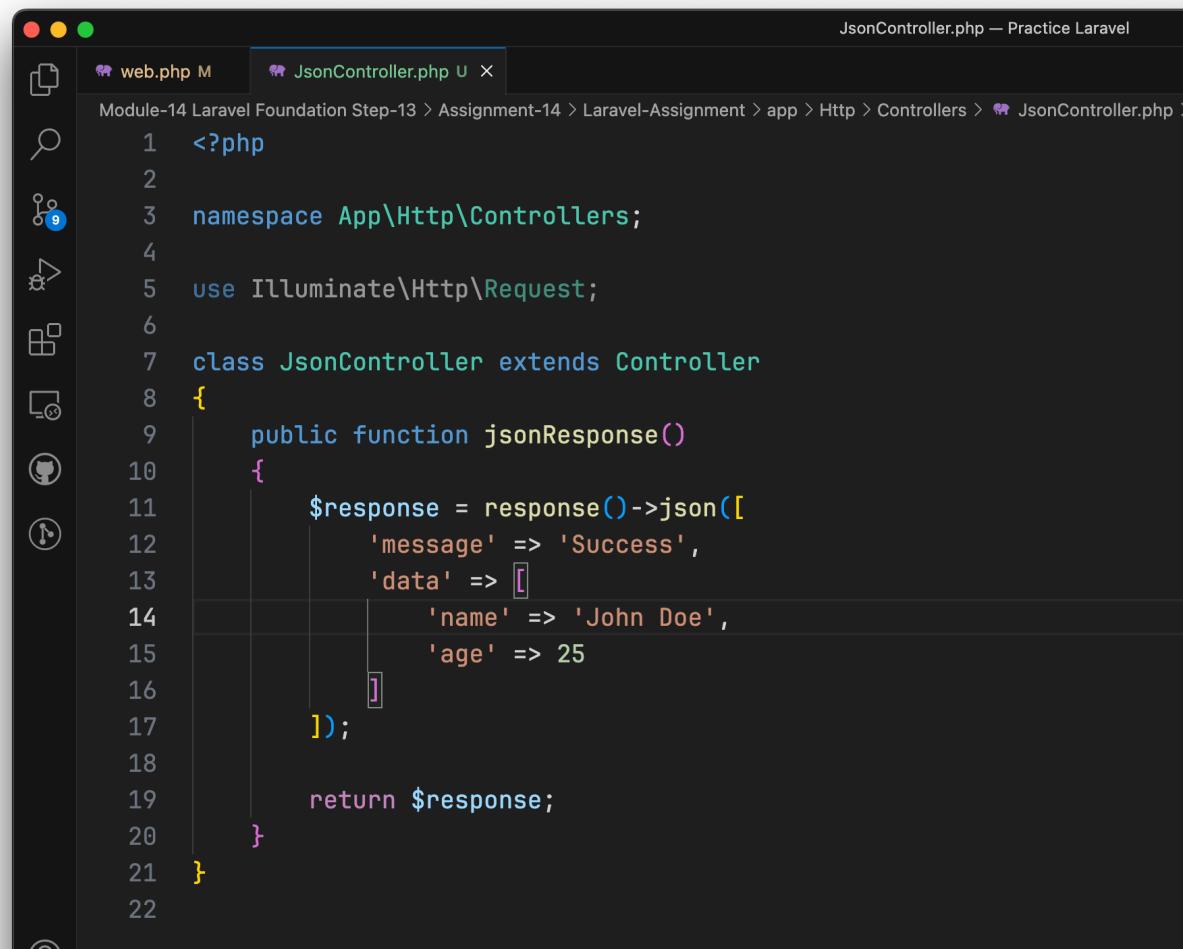
**Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command:`php artisan make:controller JsonController`



```
Laravel-Assignment — zsh — 80x24  
...l-Assignment — php artisan serve ... ...ent-14/Laravel-Assignment — zsh +  
Last login: Sun May 21 20:12:04 on ttys003  
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller JsonController  
INFO Controller [app/Http/Controllers/JsonController.php] created successfully.  
marajulislam@Marajuls-MacBook-Air Laravel-Assignment %
```

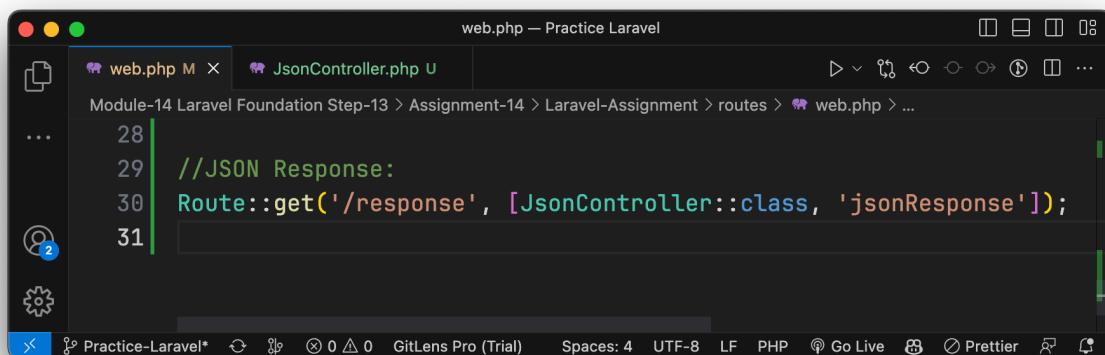
## Step 2: Implement the controller method



A screenshot of a code editor showing the `JsonController.php` file. The code defines a controller class that returns a JSON response.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class JsonController extends Controller
{
    public function jsonResponse()
    {
        $response = response()->json([
            'message' => 'Success',
            'data' => [
                'name' => 'John Doe',
                'age' => 25
            ]
        ]);
        return $response;
    }
}
```

## Step 3: Define the route



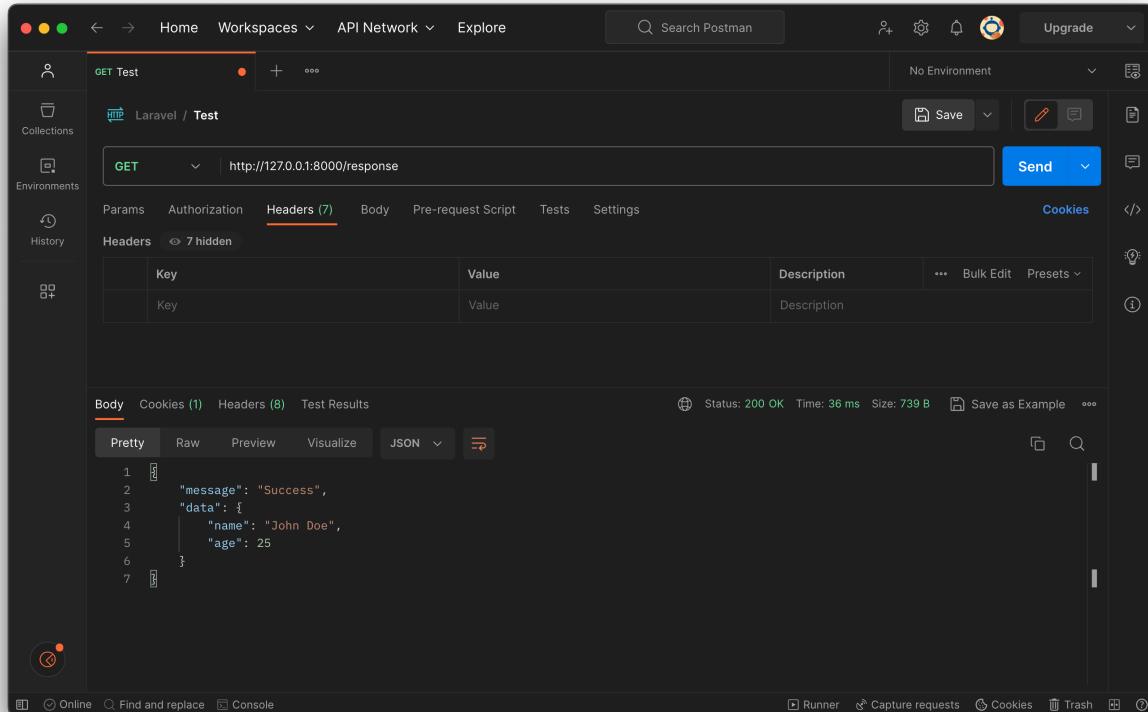
A screenshot of a code editor showing the `web.php` file. It contains a route definition for a JSON response.

```
//JSON Response:
Route::get('/response', [JsonController::class, 'jsonResponse']);
```

## Step 4: Start the Laravel development server

1. To start a development server, use the new command:php artisan serve

## Step 5: Test the route using Postman



The screenshot shows the Postman application interface. A test named "GET Test" is selected. The request URL is set to "http://127.0.0.1:8000/response". The "Headers" tab is active, showing a single header entry: "Key" under "Key" and "Value" under "Value". The "Body" tab is also visible. Below the request details, the response status is shown as "Status: 200 OK", "Time: 36 ms", and "Size: 739 B". The response body is displayed in "Pretty" format:

```
1  "message": "Success",
2  "data": {
3      "name": "John Doe",
4      "age": 25
5  }
```

**Question 5: You are implementing a file upload feature in your Laravel application. Write the code to handle a file upload named 'avatar' in the current request and store the uploaded file in the 'public/uploads' directory. Use the original filename for the uploaded file.**

## Step 1: Create a new controller

1. Open Terminal.
2. Run the following command:php artisan make:controller UploadController

```
Laravel-Assignment — zsh — 80x24
...el-Assignment — php < php artisan serve      ...ent-14/Laravel-Assignment — zsh +
```

```
Last login: Sun May 21 22:25:58 on ttys003
[mrajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller UploadController]
```

```
[INFO] Controller [app/Http/Controllers/UploadController.php] created successfully.
```

```
mrajulislam@Marajuls-MacBook-Air Laravel-Assignment %
```

## Step 2: Implement the controller method

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
class UploadController extends Controller  
{  
    public function handleUpload(Request $request)  
    {  
        if ($request->hasFile('avatar')) {  
            $file = $request->file('avatar');  
            $file->move(public_path('uploads'), $file->getClientOriginalName());  
            return response()->json(['message' => 'File uploaded successfully']);  
        } else {  
            return response()->json(['message' => 'No file uploaded']);  
        }  
    }  
}
```

Ln 21, Col 1 Spaces: 4 UTF-8 LF PHP Go Live Prettier

## Step 3: Define the route

```
...  
32 //Upload-File:  
33 Route::post('/upload', [UploadController::class, 'handleUpload']);  
34  
35  
36 |
```

Spaces: 4 UTF-8 LF PHP Go Live Prettier

## Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

## Step 5: Test the route using Postman

The screenshot shows the Postman application interface. A test named "POST Test" is selected. The request URL is "http://127.0.0.1:8000/upload". The "Headers" tab is active, showing a table with one row: "Key" and "Value". The "Body" tab is also visible, showing a JSON response with the following content:

```
1  "message": "No file uploaded"
```

**Question 6: Retrieve the value of the 'remember\_token' cookie from the current request in Laravel and store it in a variable called \$rememberToken. If the cookie is not present, set \$rememberToken to null.**

### **Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command:`php artisan make:controller ManageController`

```
Last login: Mon May 22 12:03:11 on ttys000
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller ManageController

    INFO Controller [app/Http/Controllers/ManageController.php] created successfully.

marajulislam@Marajuls-MacBook-Air Laravel-Assignment % ]
```

## Step 2: Implement the controller method

A screenshot of the VS Code interface showing the `ManageController.php` file. The code defines a controller class `ManageController` that retrieves a remember token from a cookie and returns it as JSON.

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
class ManageController extends Controller  
{  
    public function retrieveRememberToken(Request $request): array|string|null  
    {  
        $rememberToken = $request->cookie('remember_token', null);  
        return response()->json(['rememberToken' => $rememberToken]);  
    }  
}
```

### Step 3: Define the route

A screenshot of the VS Code interface showing the `routes/web.php` file. It contains a route definition for retrieving a cookie.

```
//Retrieve-Cookies:  
Route::get('/retrieve-cookie', [ManageController::class, 'retrieveRememberToken']);
```

### Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

### Step 5: Test the route using Postman

The screenshot shows the Postman application interface. A GET request is being made to the URL `http://127.0.0.1:8000/retrieve-cookie`. The Headers tab is selected, showing one header entry: `Key` and `Value`. The Body tab shows the response content, which is a JSON object with the following structure:

```
1 HTTP/1.0 200 OK
2 Cache-Control: no-cache, private
3 Content-Type: application/json
4 Date: Mon, 22 May 2023 06:03:57 GMT
5
6 {"rememberToken":null}
```

**Question 7: Create a route in Laravel that handles a POST request to the '/submit' URL. Inside the route closure, retrieve the 'email' input parameter from the request and store it in a variable called \$email. Return a JSON response with the following data:**

```
{
```

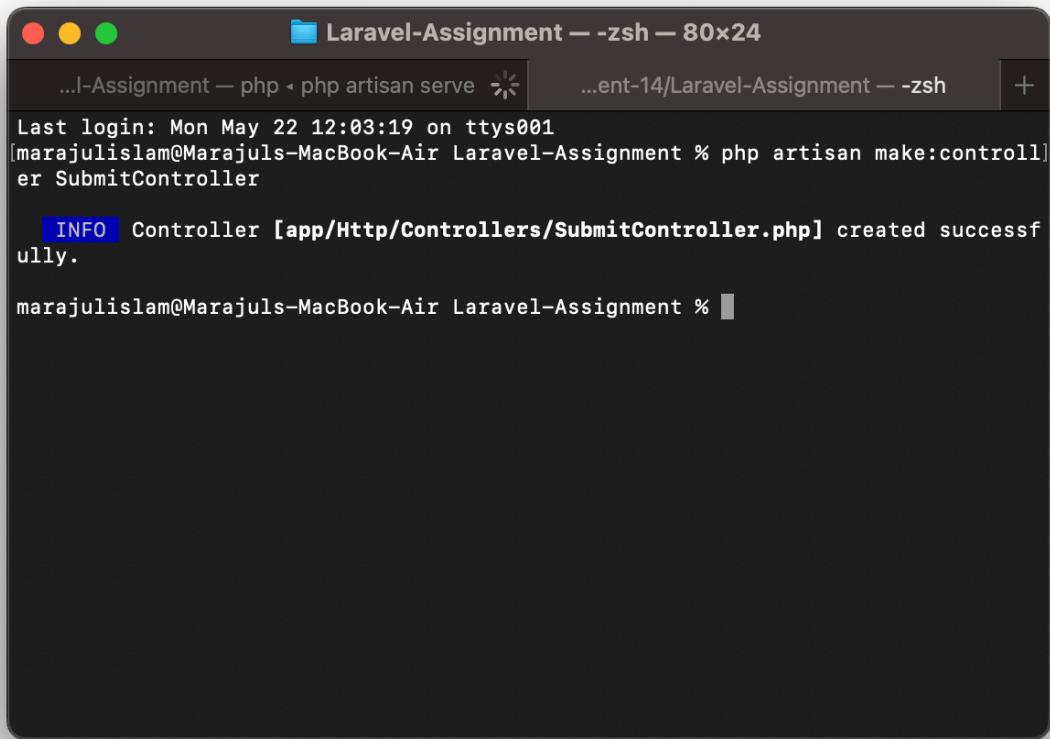
```
  "success": true,
```

```
  "message": "Form submitted successfully."
```

```
}
```

### **Step 1: Create a new controller**

1. Open Terminal.
2. Run the following command:`php artisan make:controller SubmitController`



```
...l-Assignment — php -S php artisan serve ...ent-14/Laravel-Assignment — zsh +  
Last login: Mon May 22 12:03:19 on ttys001  
[marajulislam@Marajuls-MacBook-Air Laravel-Assignment % php artisan make:controller SubmitController]  
INFO Controller [app/Http/Controllers/SubmitController.php] created successfully.  
marajulislam@Marajuls-MacBook-Air Laravel-Assignment %
```

## Step 2: Implement the controller method

A screenshot of a code editor window titled "SubmitController.php — Practice Laravel". The editor shows the following PHP code:

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
class SubmitController extends Controller  
{  
    public function submit(Request $request)  
    {  
        $email = $request->input('email');  
        return response()->json([  
            'success' => true,  
            'message' => 'Form submitted successfully.'  
        ]);  
    }  
}
```

The code editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom.

### Step 3: Define the route

A screenshot of a code editor window titled "web.php — Practice Laravel". The editor shows the following PHP code:

```
...  
40 //Submit:  
41 Route::post('/submit', [SubmitController::class, 'submit']);  
42  
43
```

The code editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom.

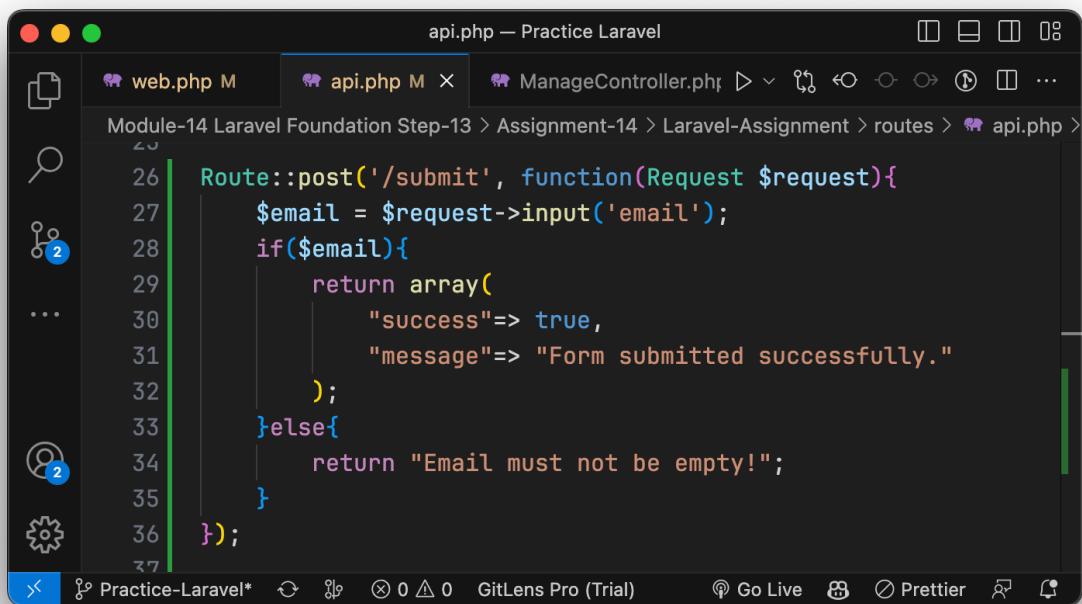
## Step 4: Start the Laravel development server

1. To start a development server, use the new command: `php artisan serve`

## Step 5: Test the route using Postman

The screenshot shows the Postman application interface. A test named "Laravel / Test" is selected. The request method is set to "POST" and the URL is "http://127.0.0.1:8000/submit". The "Headers" tab is active, showing one header entry: "Key" with the value "Value". Below the request details, the response status is "Status: 200 OK", time is "Time: 30 ms", and size is "Size: 739 B". The response body is displayed in "Pretty" format, showing the JSON response: "success": true, "message": "Form submitted successfully.".

Another way:



The screenshot shows a code editor window titled "api.php — Practice Laravel". The file content is as follows:

```
Route::post('/submit', function(Request $request){  
    $email = $request->input('email');  
    if($email){  
        return array(  
            "success"=> true,  
            "message"=> "Form submitted successfully."  
        );  
    }else{  
        return "Email must not be empty!";  
    }  
});
```

The code defines a POST route for '/submit' that checks if the 'email' input is present. If it is, it returns an array with 'success' set to true and a success message. If it's not, it returns a string indicating the email is required.

**Author: Marajul Islam**