



Laravel



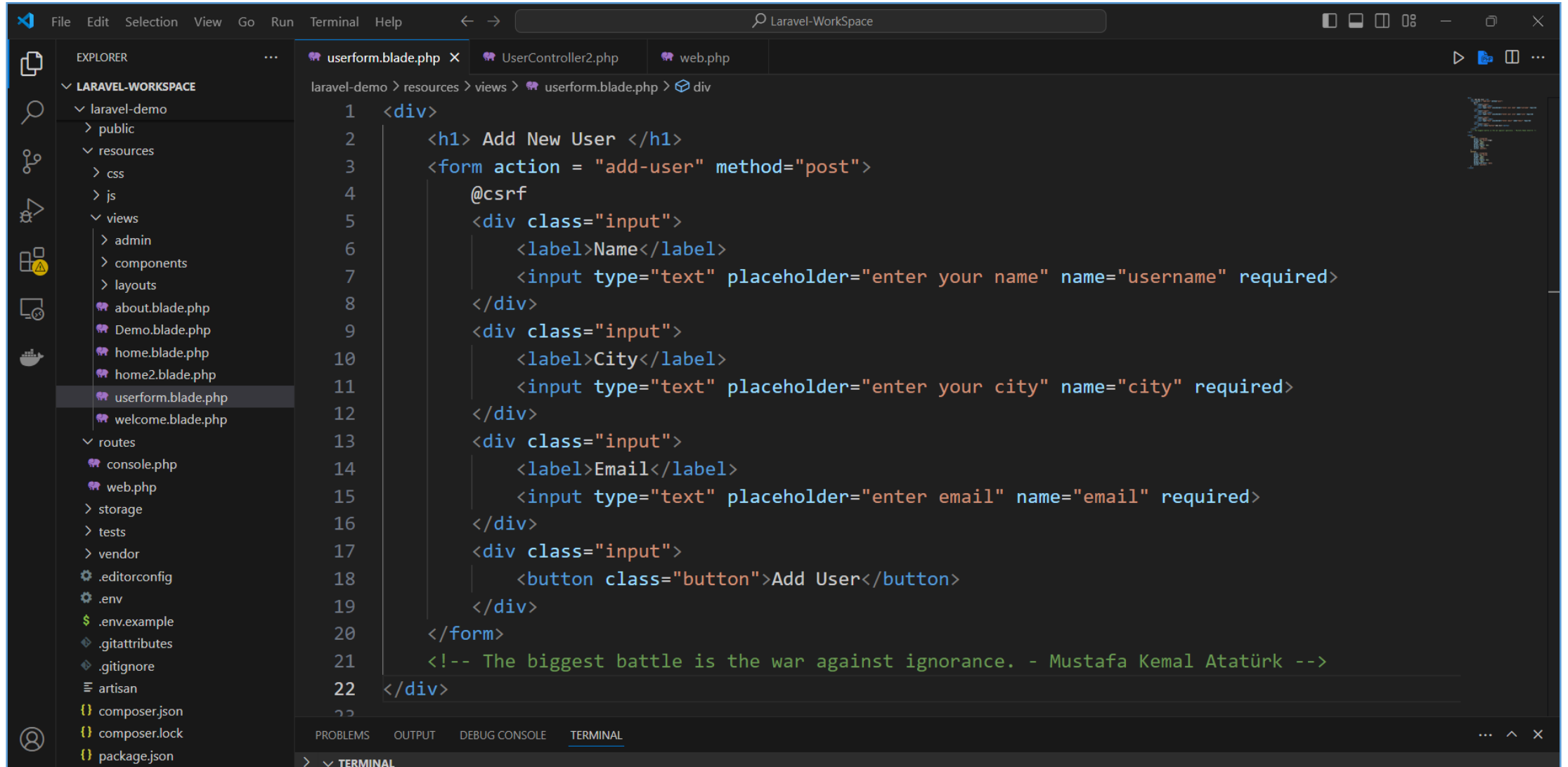
Outlines of Discussions

- ❖ **Form handling in laravel**
- ❖ **Form Validation**
- ❖ **Named Route in Laravel**
- ❖ **Route Grouping**

Form handling in laravel

- In Laravel, forms and input fields can be created using **standard HTML** or **Laravel's Blade** syntax.
- Laravel doesn't have its own syntax for creating forms, but it does offer various **tools** and **helper** methods to make handling forms more manageable.

Cont. ...



```
1 <div>
2   <h1> Add New User </h1>
3   <form action = "add-user" method="post">
4     @csrf
5     <div class="input">
6       <label>Name</label>
7       <input type="text" placeholder="enter your name" name="username" required>
8     </div>
9     <div class="input">
10      <label>City</label>
11      <input type="text" placeholder="enter your city" name="city" required>
12    </div>
13    <div class="input">
14      <label>Email</label>
15      <input type="text" placeholder="enter email" name="email" required>
16    </div>
17    <div class="input">
18      <button class="button">Add User</button>
19    </div>
20  </form>
21  <!-- The biggest battle is the war against ignorance. - Mustafa Kemal Atatürk -->
22 </div>
```

Cont. ...

Add New User

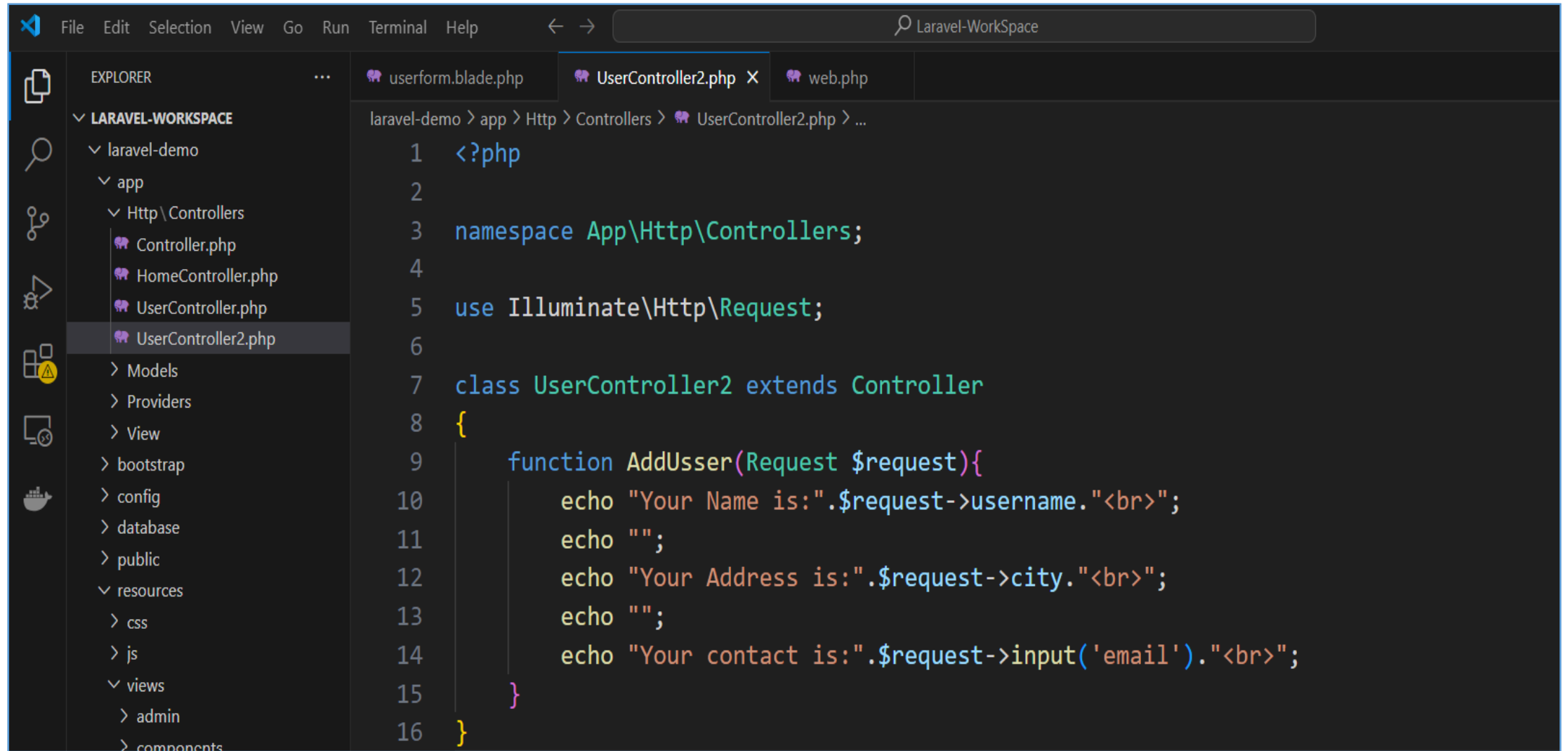
Name

City

Email

Add User

Cont. ...



The image shows a Visual Studio Code editor interface with a dark theme. The Explorer sidebar on the left displays the project structure for 'Laravel-Workspace'. The file 'UserController2.php' is selected and open in the editor. The breadcrumb navigation shows the path: 'laravel-demo > app > Http > Controllers > UserController2.php > ...'. The code in the editor is a PHP controller class named 'UserController2' that extends 'Controller'. It contains a single method 'AddUser' that takes a 'Request' object and echoes the user's name, address, and contact information.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class UserController2 extends Controller
8  {
9      function AddUser(Request $request){
10         echo "Your Name is:". $request->username."<br>";
11         echo "";
12         echo "Your Address is:". $request->city."<br>";
13         echo "";
14         echo "Your contact is:". $request->input('email')."<br>";
15     }
16 }
```

Cont. ...

```
File Edit Selection View Go Run Terminal Help
```

Laravel-WorkSpace

EXPLORER

laravel-demo

- resources
 - css
 - js
 - views
 - admin
 - components
 - layouts
- about.blade.php
- Demo.blade.php
- home.blade.php
- home2.blade.php
- userform.blade.php
- welcome.blade.php
- routes
 - console.php
 - web.php

laravel-demo > routes > web.php > ...

```
30 Route::view('userform', 'userform');
31 Route::Post('add-user', [UserController2::class, 'AddUsser']);
32
33
34
35
36
37
38
39
40
41
42
```

Cont. ...

Add New User

Name

Yitayew Solomon

City

Addis Abeba

Email

yitayewsolomon3@gmail.com

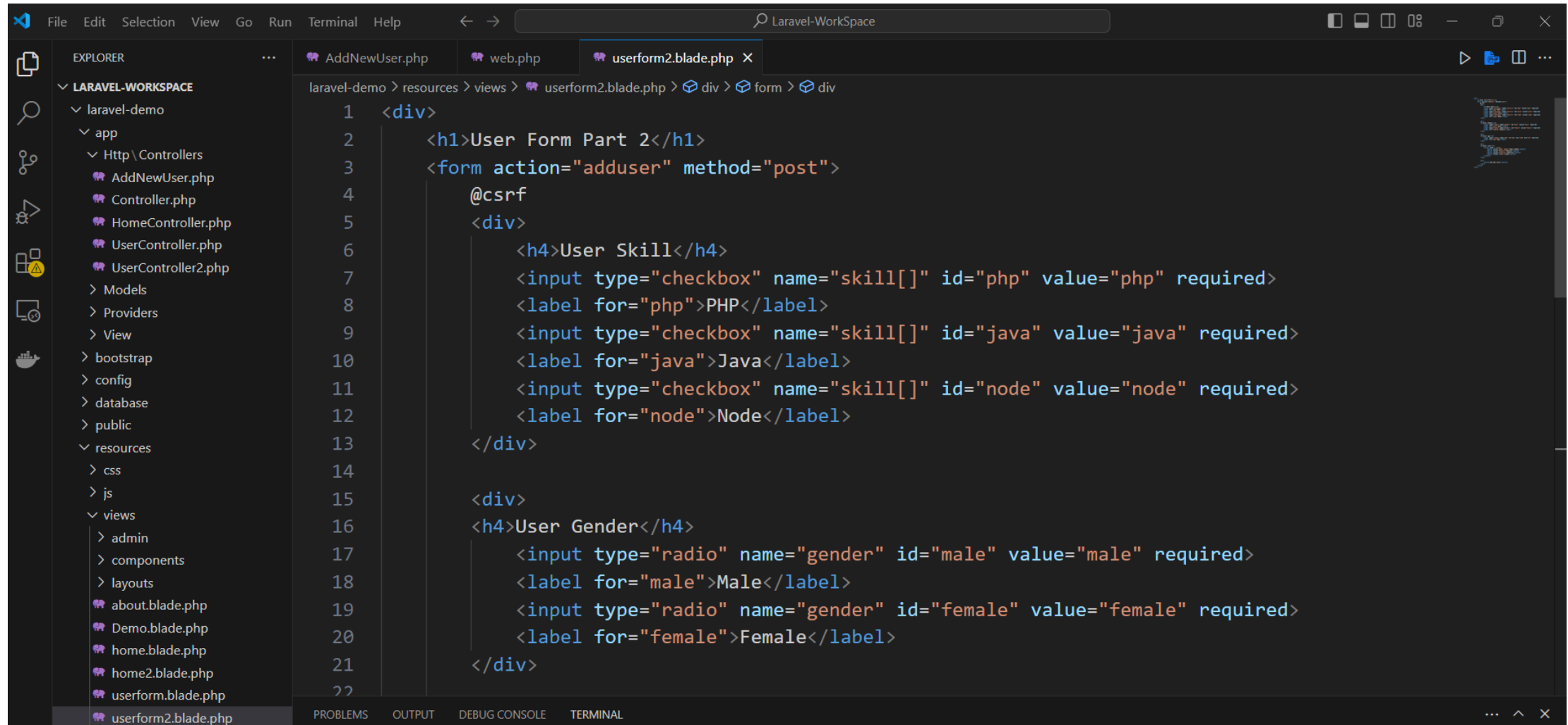
Add User

Your Name is:Yitayew Solomon

Your Address is:Addis Abeba

Your contact is:yitayewsolomon3@gmail.com

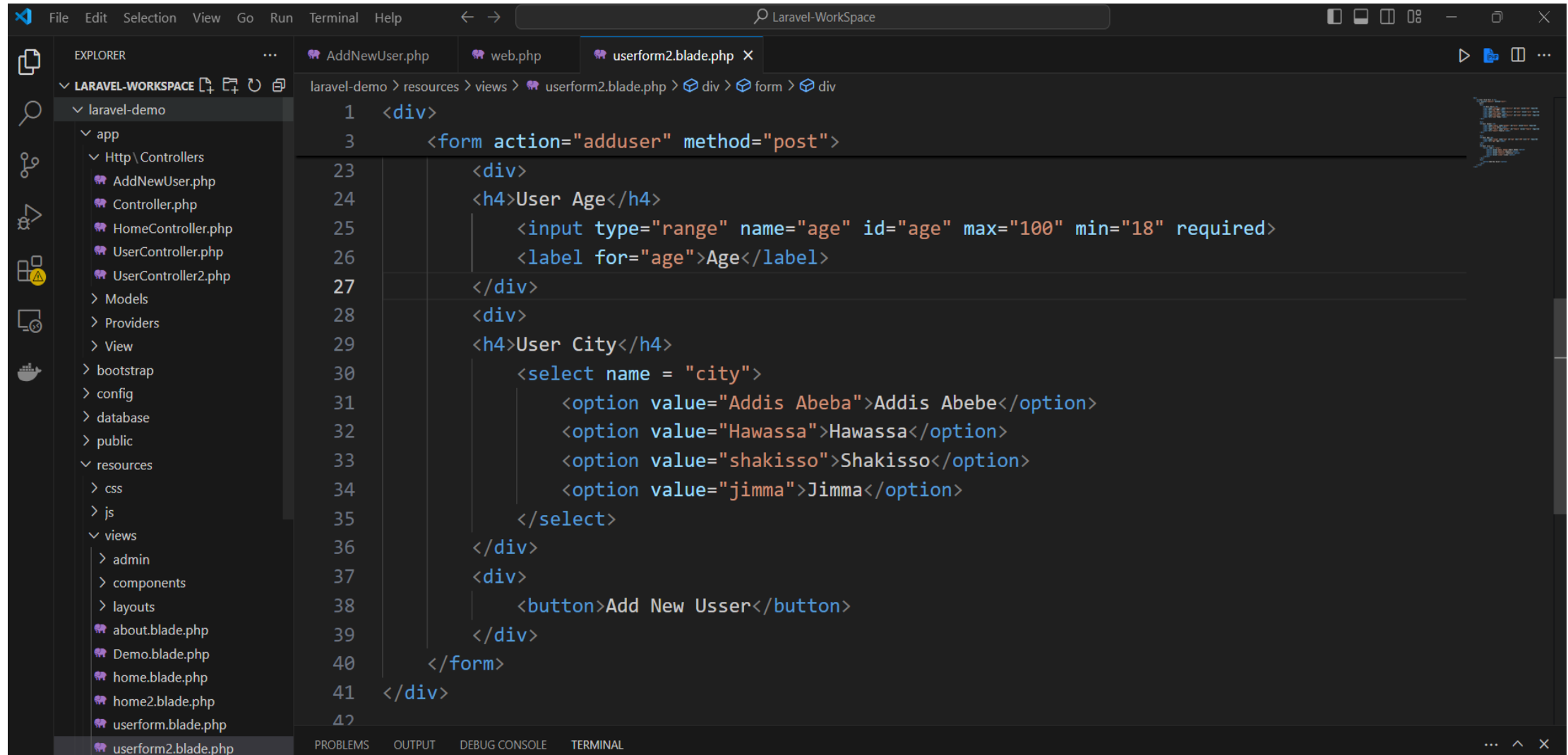
More on Forms



The image shows a Visual Studio Code editor window with a Laravel project. The Explorer sidebar on the left displays the project structure, including the 'resources' directory with 'views' and 'userform2.blade.php'. The main editor area shows the content of 'userform2.blade.php', which is a Blade template for a user form. The template includes a form with a CSRF token, a skill selection section with three checkboxes (PHP, Java, Node), and a gender selection section with two radio buttons (Male, Female).

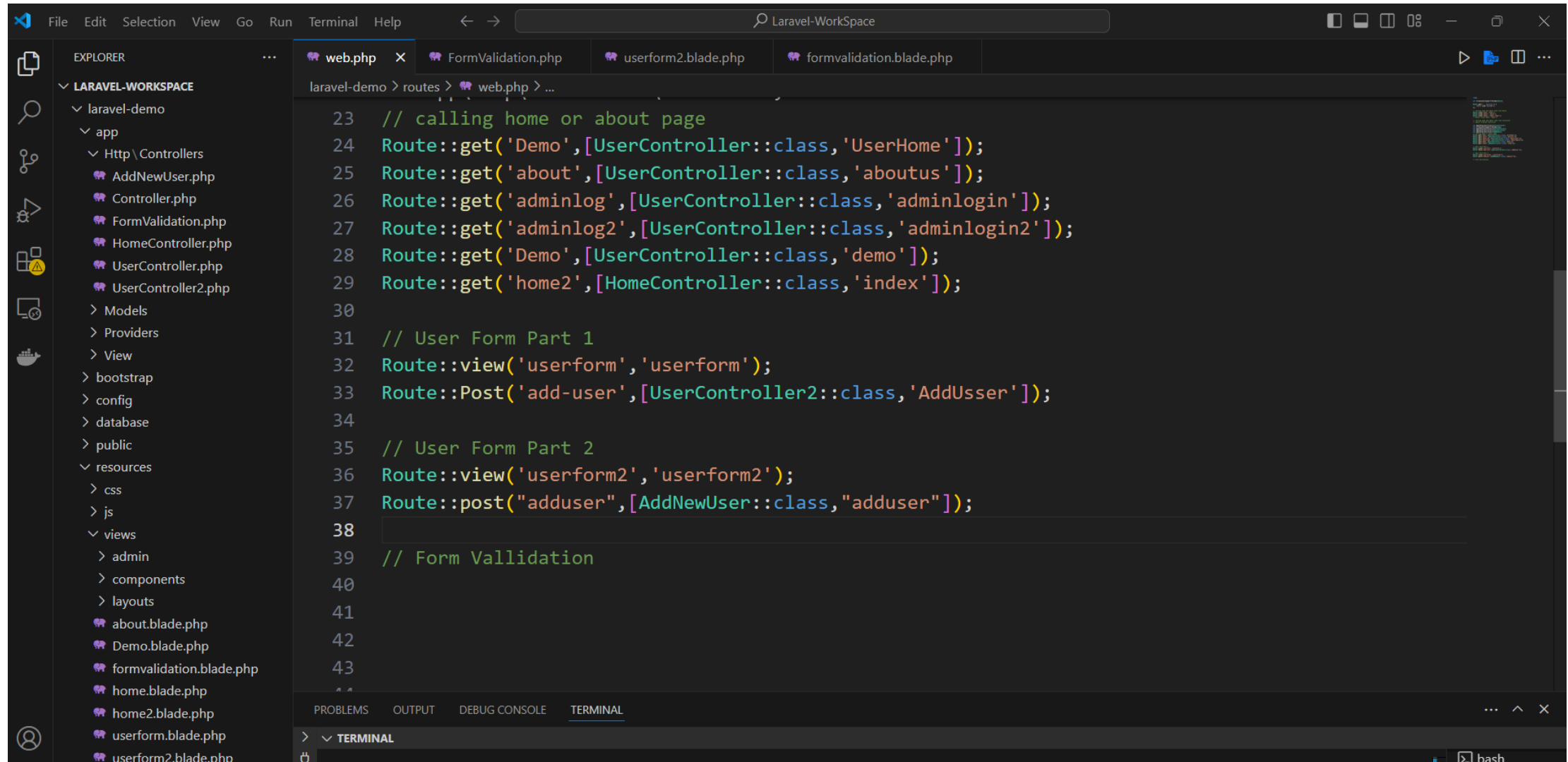
```
1 <div>
2   <h1>User Form Part 2</h1>
3   <form action="adduser" method="post">
4     @csrf
5     <div>
6       <h4>User Skill</h4>
7       <input type="checkbox" name="skill[]" id="php" value="php" required>
8       <label for="php">PHP</label>
9       <input type="checkbox" name="skill[]" id="java" value="java" required>
10      <label for="java">Java</label>
11      <input type="checkbox" name="skill[]" id="node" value="node" required>
12      <label for="node">Node</label>
13    </div>
14
15    <div>
16      <h4>User Gender</h4>
17      <input type="radio" name="gender" id="male" value="male" required>
18      <label for="male">Male</label>
19      <input type="radio" name="gender" id="female" value="female" required>
20      <label for="female">Female</label>
21    </div>
22  </form>
23 </div>
```

Cont. ...



```
1 <div>
3 <form action="adduser" method="post">
23 <div>
24 <h4>User Age</h4>
25 <input type="range" name="age" id="age" max="100" min="18" required>
26 <label for="age">Age</label>
27 </div>
28 <div>
29 <h4>User City</h4>
30 <select name = "city">
31 <option value="Addis Abeba">Addis Abebe</option>
32 <option value="Hawassa">Hawassa</option>
33 <option value="shakisso">Shakisso</option>
34 <option value="jimma">Jimma</option>
35 </select>
36 </div>
37 <div>
38 <button>Add New Usser</button>
39 </div>
40 </form>
41 </div>
42
```

Cont. ...



The screenshot shows the Visual Studio Code interface with a Laravel project workspace. The Explorer panel on the left shows the project structure, including controllers, models, providers, views, and resources. The main editor displays the `web.php` file in the `routes` directory, which defines several routes for the application. The routes include a home/about page, an admin login page, a demo page, and a home page. There are also routes for user form parts and a form validation page. The bottom panel shows the Terminal with a `bash` prompt.

```
23 // calling home or about page
24 Route::get('Demo',[UserController::class,'UserHome']);
25 Route::get('about',[UserController::class,'aboutus']);
26 Route::get('adminlog',[UserController::class,'adminlogin']);
27 Route::get('adminlog2',[UserController::class,'adminlogin2']);
28 Route::get('Demo',[UserController::class,'demo']);
29 Route::get('home2',[HomeController::class,'index']);
30
31 // User Form Part 1
32 Route::view('userform','userform');
33 Route::Post('add-user',[UserController2::class,'AddUsser']);
34
35 // User Form Part 2
36 Route::view('userform2','userform2');
37 Route::post("adduser",[AddNewUser::class,"adduser"]);
38
39 // Form Vallidation
40
41
42
43
```

Cont. ...

127.0.0.1:8000/userform2

140% ☆

User Form Part 2

User Skill

☐ PHP ☐ Java ☐ Node

User Gender

☒ Male ☐ Female

User Age

Age

User City

Addis Abebe ▾

Add New Usser

127.0.0.1:8000/adduser

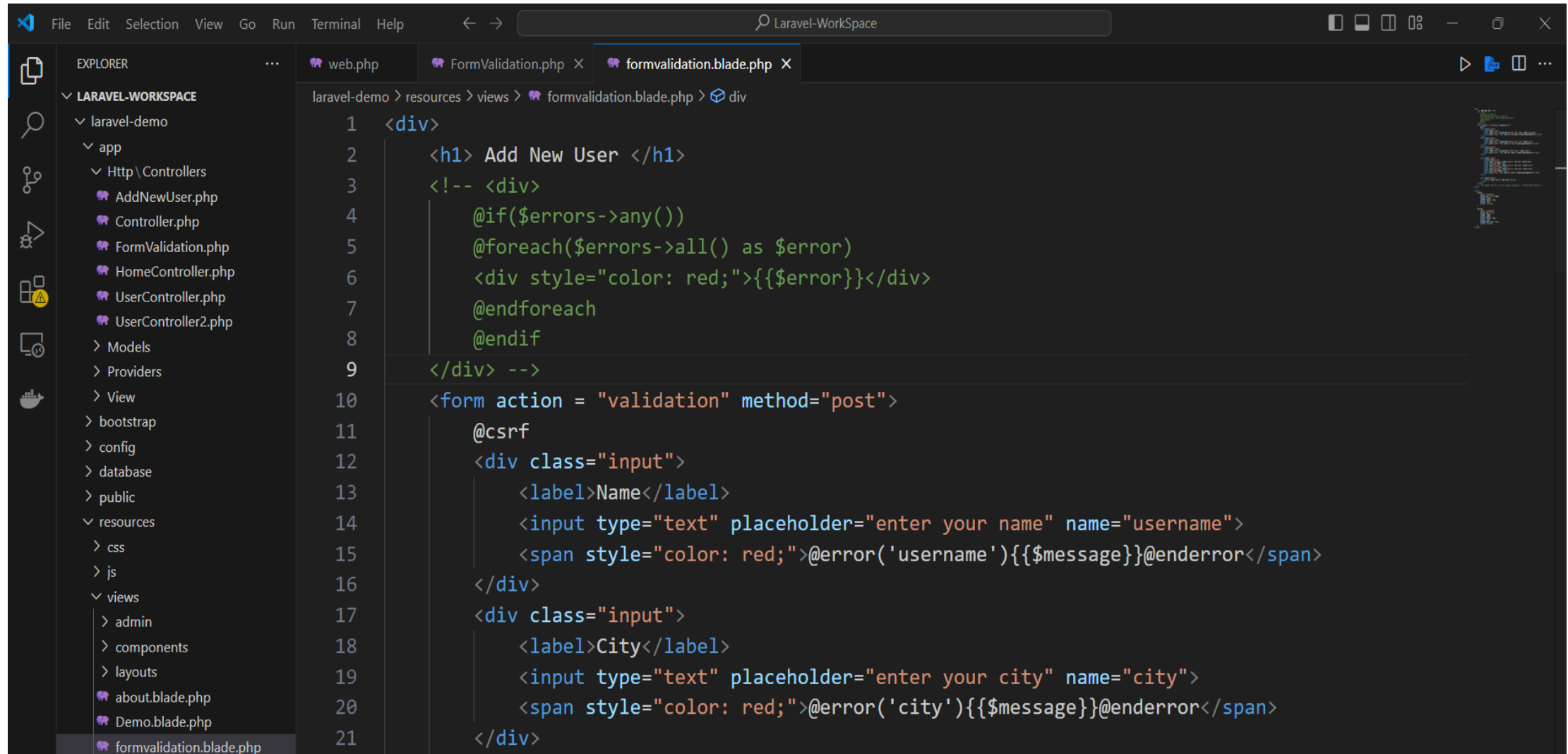
your skils are: Array ([0] => php [1] => java [2] => node)

Your Address is: Addis Abeba

Your Age is: 38

your Gender is: male

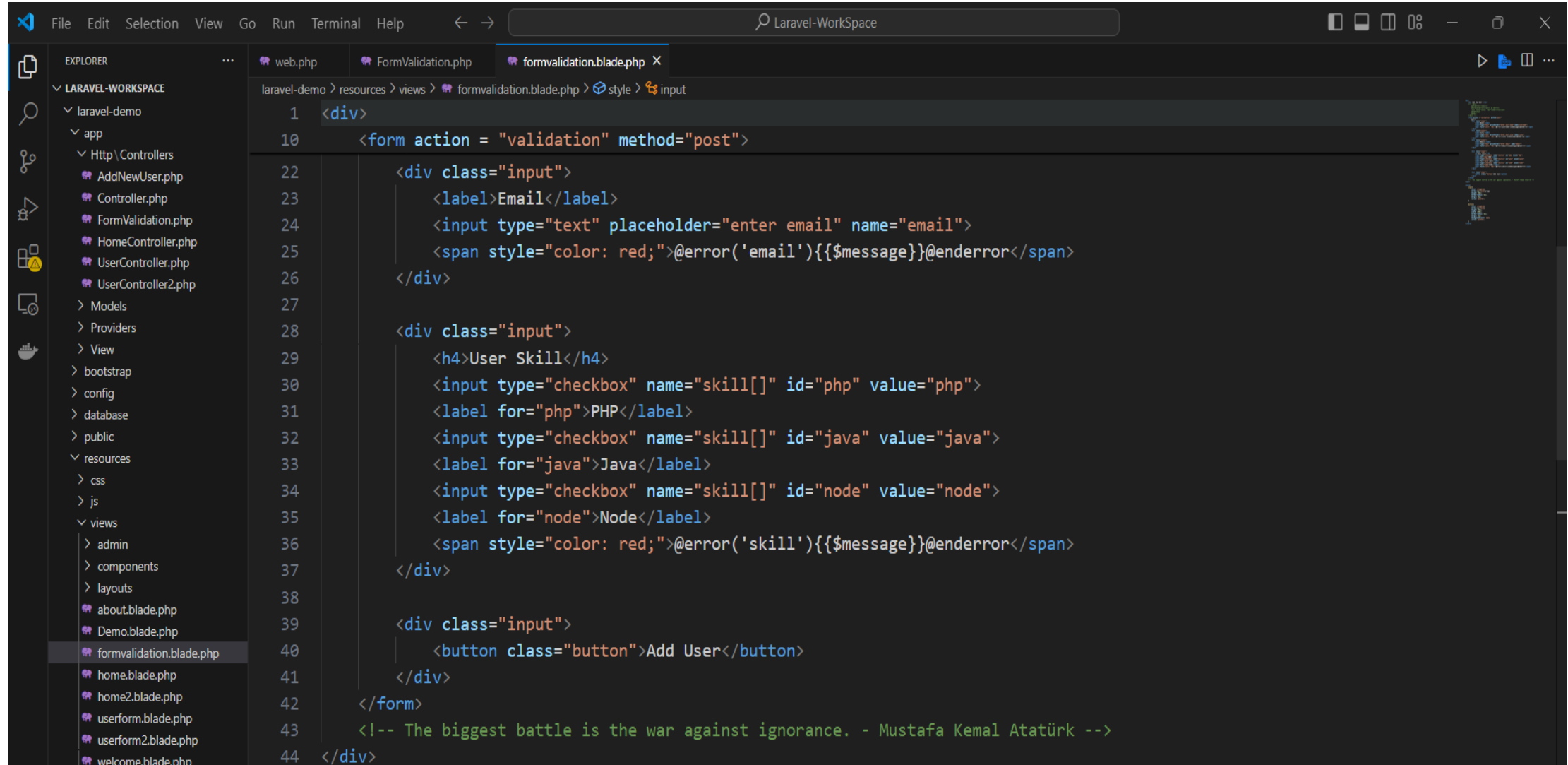
Form Validation



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows the project structure for 'laravel-demo', including 'app', 'Http\ Controllers', 'Models', 'Providers', 'View', 'bootstrap', 'config', 'database', 'public', 'resources', and 'views'. The 'formvalidation.blade.php' file is selected in the Explorer and also in the editor tabs. The editor displays the following Blade template code:

```
1 <div>
2   <h1> Add New User </h1>
3   <!-- <div>
4     @if($errors->any())
5       @foreach($errors->all() as $error)
6         <div style="color: red;">{{ $error }}</div>
7       @endforeach
8     @endif
9   </div> -->
10  <form action = "validation" method="post">
11    @csrf
12    <div class="input">
13      <label>Name</label>
14      <input type="text" placeholder="enter your name" name="username">
15      <span style="color: red;">@error('username'){{ $message }}@enderror</span>
16    </div>
17    <div class="input">
18      <label>City</label>
19      <input type="text" placeholder="enter your city" name="city">
20      <span style="color: red;">@error('city'){{ $message }}@enderror</span>
21    </div>
```

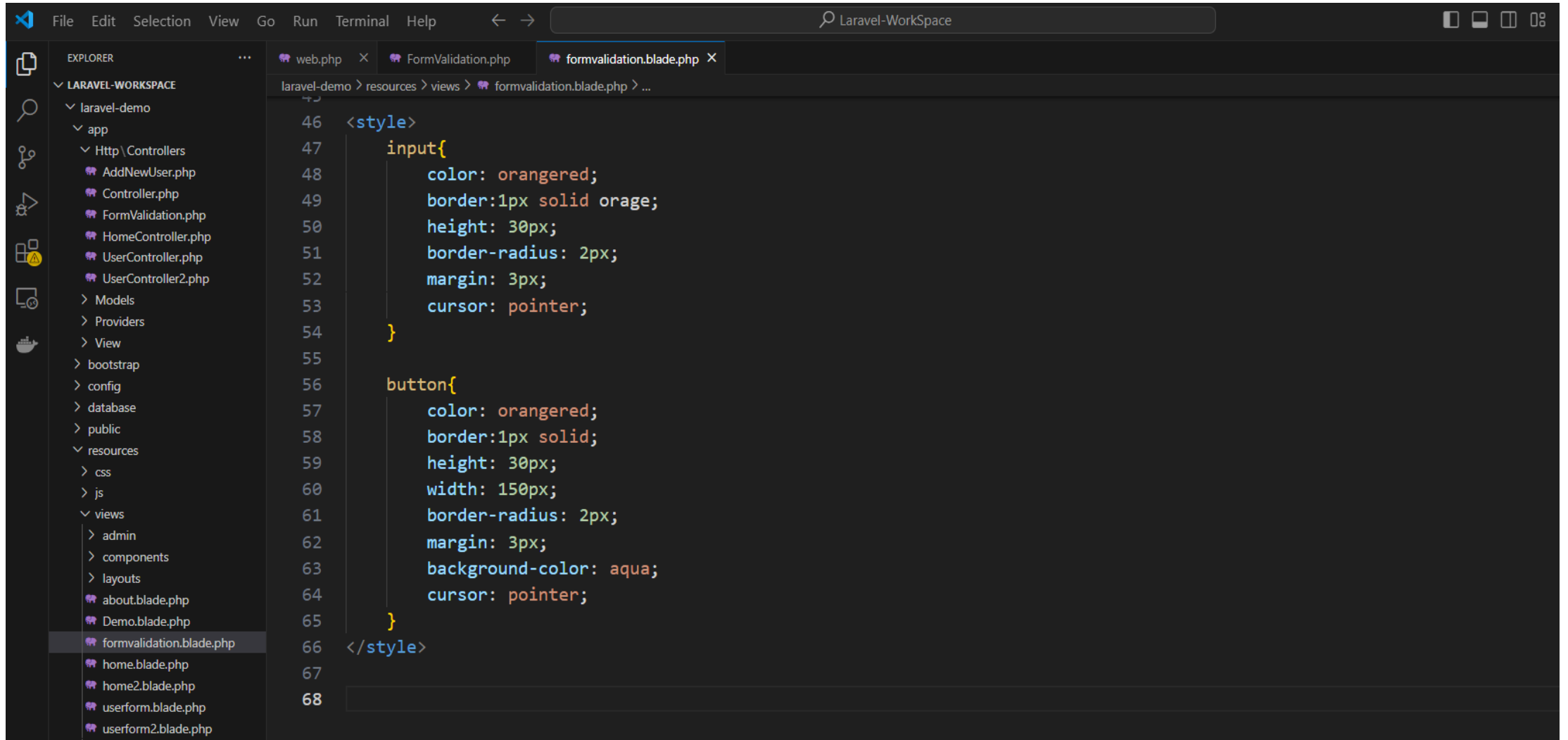
Cont. ...



The screenshot shows a Visual Studio Code editor interface with a Laravel project workspace. The Explorer panel on the left shows the project structure, including the 'resources' directory and the 'views' directory. The 'formvalidation.blade.php' file is selected in the Explorer and is also open in the editor. The editor displays the following code:

```
1 <div>
10 <form action = "validation" method="post">
22 <div class="input">
23 <label>Email</label>
24 <input type="text" placeholder="enter email" name="email">
25 <span style="color: red;">@error('email'){{$message}}@enderror</span>
26 </div>
27
28 <div class="input">
29 <h4>User Skill</h4>
30 <input type="checkbox" name="skill[]" id="php" value="php">
31 <label for="php">PHP</label>
32 <input type="checkbox" name="skill[]" id="java" value="java">
33 <label for="java">Java</label>
34 <input type="checkbox" name="skill[]" id="node" value="node">
35 <label for="node">Node</label>
36 <span style="color: red;">@error('skill'){{$message}}@enderror</span>
37 </div>
38
39 <div class="input">
40 <button class="button">Add User</button>
41 </div>
42 </form>
43 <!-- The biggest battle is the war against ignorance. - Mustafa Kemal Atatürk -->
44 </div>
```

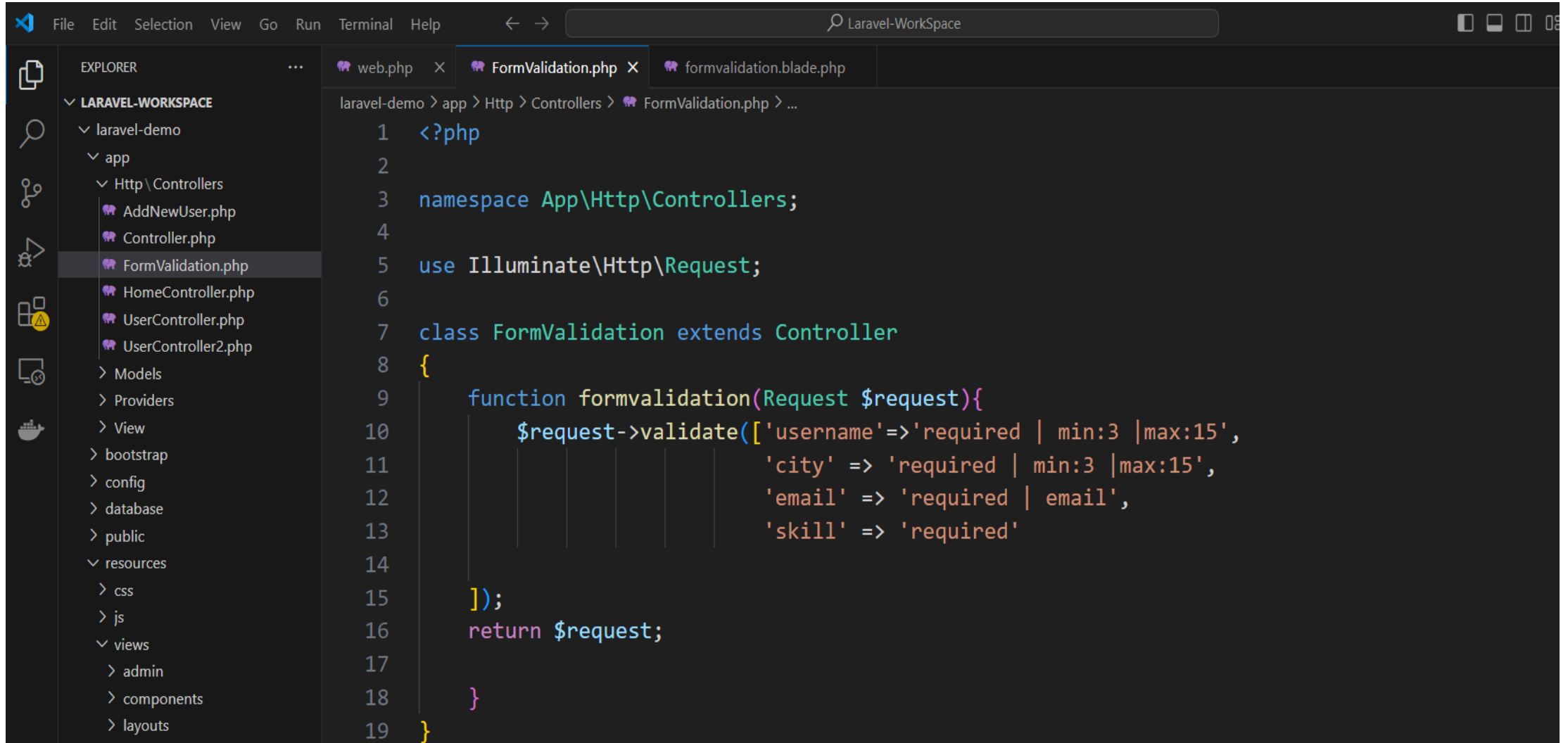
Cont. ...



The image shows a Visual Studio Code editor window with a dark theme. The title bar at the top reads "Laravel-WorkSpace". The Explorer sidebar on the left shows a project structure for "LARAVEL-WORKSPACE". The "resources" folder is expanded, showing subfolders "css", "js", and "views". The "views" folder is further expanded, listing several Blade templates, with "formvalidation.blade.php" selected and highlighted. The main editor area displays the content of "formvalidation.blade.php", which contains CSS styles for an input field and a button. The code is as follows:

```
46 <style>
47     input{
48         color: orangered;
49         border:1px solid orange;
50         height: 30px;
51         border-radius: 2px;
52         margin: 3px;
53         cursor: pointer;
54     }
55
56     button{
57         color: orangered;
58         border:1px solid;
59         height: 30px;
60         width: 150px;
61         border-radius: 2px;
62         margin: 3px;
63         background-color: aqua;
64         cursor: pointer;
65     }
66 </style>
67
68
```

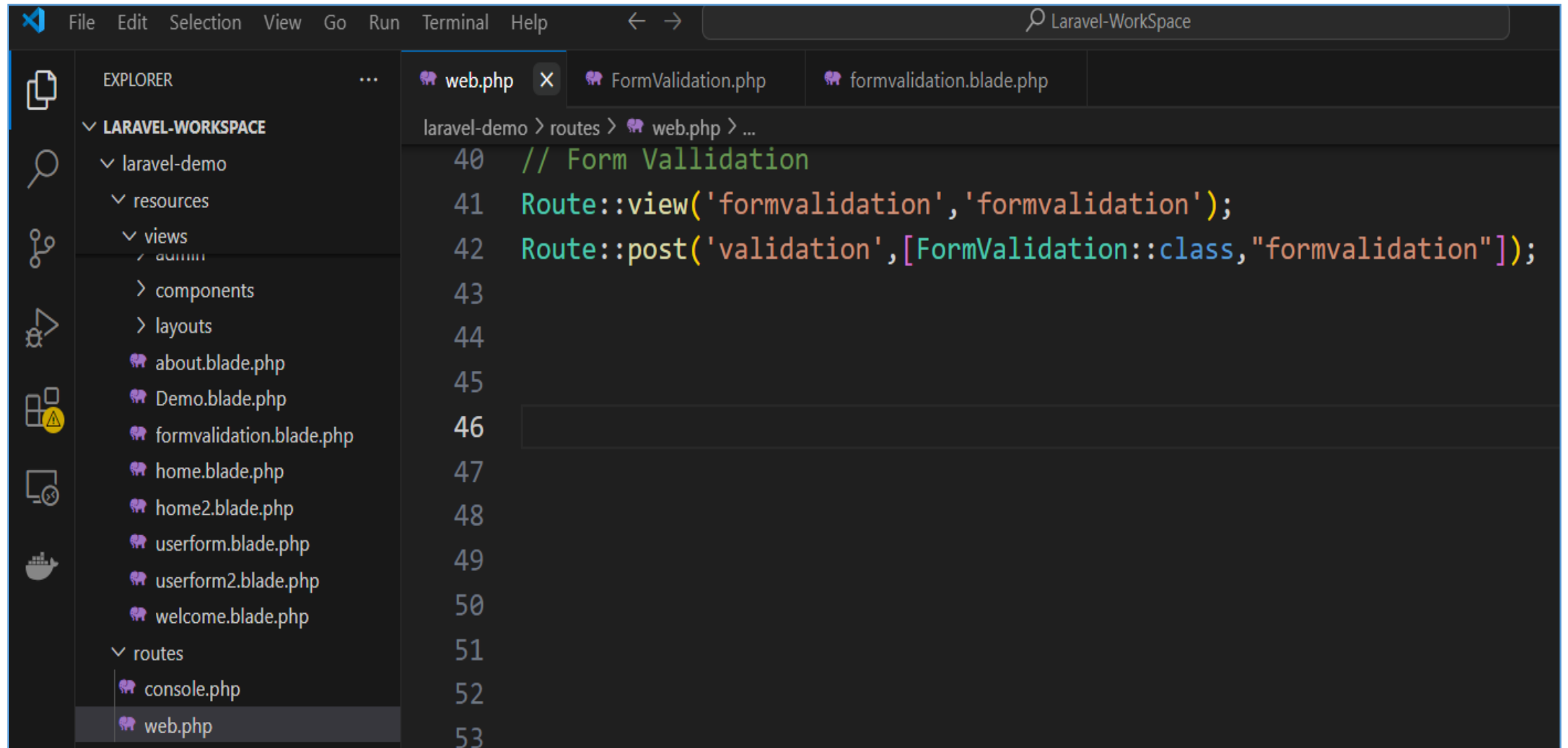
Cont. ...



The screenshot shows the Visual Studio Code interface with a Laravel project. The Explorer panel on the left displays the project structure, including the `FormValidation.php` file in the `Http\Controllers` directory. The main editor area shows the code for `FormValidation.php`, which includes a namespace declaration, a use statement for `Illuminate\Http\Request`, and a `formvalidation` method that validates user input.

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class FormValidation extends Controller
8 {
9     function formvalidation(Request $request){
10         $request->validate(['username'=>'required | min:3 | max:15',
11                             'city' => 'required | min:3 | max:15',
12                             'email' => 'required | email',
13                             'skill' => 'required'
14
15         ]);
16         return $request;
17
18     }
19 }
```


Cont. ...



```
File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace

EXPLORER
LARAVEL-WORKSPACE
  laravel-demo
    resources
      views
        admin
          components
          layouts
        about.blade.php
        Demo.blade.php
        formvalidation.blade.php
        home.blade.php
        home2.blade.php
        userform.blade.php
        userform2.blade.php
        welcome.blade.php
      routes
        console.php
        web.php

web.php
40 // Form Vallidation
41 Route::view('formvalidation','formvalidation');
42 Route::post('validation',[FormValidation::class,"formvalidation"]);
43
44
45
46
47
48
49
50
51
52
53
```

Cont. ...

127.0.0.1:8000/formvalidation

Add New User

Name

enter your name

The username field is required.

City

enter your city

The city field is required.

Email

enter email

The email field is required.

User Skill

☐ PHP

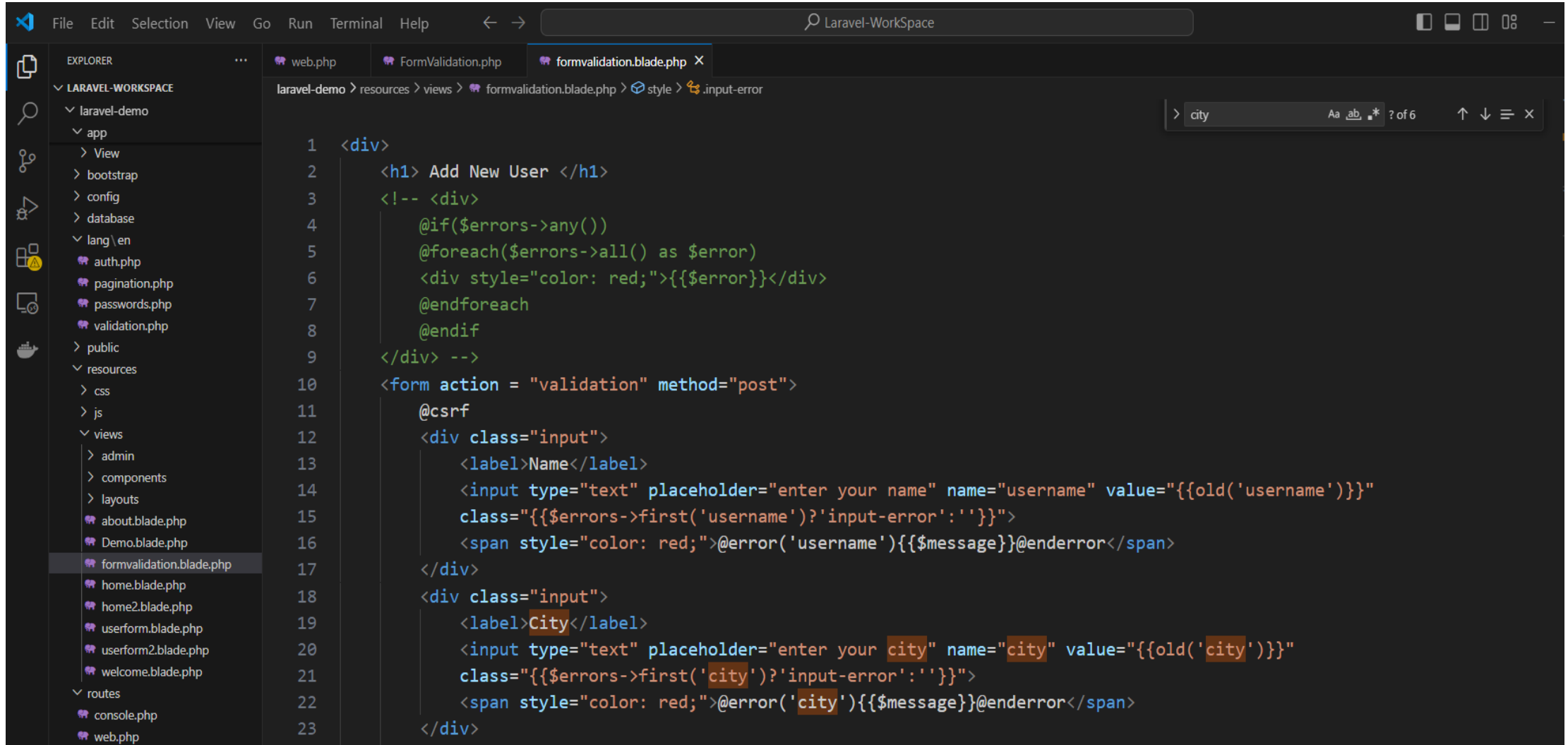
☐ Java

☐ Node

The skill field is required.

Add User

Custom Validation Message



The screenshot shows a Visual Studio Code editor window titled "Laravel-WorkSpace". The Explorer sidebar on the left displays the project structure for "laravel-demo", including folders like "app", "bootstrap", "config", "database", "lang", "public", "resources", and "views". The "formvalidation.blade.php" file is selected in the Explorer and also in the editor's tab bar. The breadcrumb navigation at the top of the editor shows the path: "laravel-demo > resources > views > formvalidation.blade.php > style > .input-error". The main editor area displays the content of "formvalidation.blade.php" with line numbers 1 through 23. The code is a Blade template that includes a form for adding a new user. It features a conditional block to display validation errors for the 'username' field, followed by an input field for the username. Below this, there is another input field for the 'city' field, which also has a custom validation error message displayed. The code uses Laravel's Blade templating engine syntax, including @if, @foreach, @endif, @csrf, and @error directives. The validation error messages are displayed in red text within a span element with the class "input-error".

```
1 <div>
2 <h1> Add New User </h1>
3 <!-- <div>
4     @if($errors->any())
5         @foreach($errors->all() as $error)
6             <div style="color: red;">{{$error}}</div>
7         @endforeach
8     @endif
9 </div> -->
10 <form action = "validation" method="post">
11     @csrf
12     <div class="input">
13         <label>Name</label>
14         <input type="text" placeholder="enter your name" name="username" value="{{old('username')}}"
15             class="{{ $errors->first('username')?'input-error':'' }}">
16         <span style="color: red;">@error('username'){{$message}}@enderror</span>
17     </div>
18     <div class="input">
19         <label>City</label>
20         <input type="text" placeholder="enter your city" name="city" value="{{old('city')}}"
21             class="{{ $errors->first('city')?'input-error':'' }}">
22         <span style="color: red;">@error('city'){{$message}}@enderror</span>
23     </div>
```

Cont. ...

The screenshot shows the Visual Studio Code interface with a Laravel project workspace. The Explorer sidebar on the left lists the following files and directories:

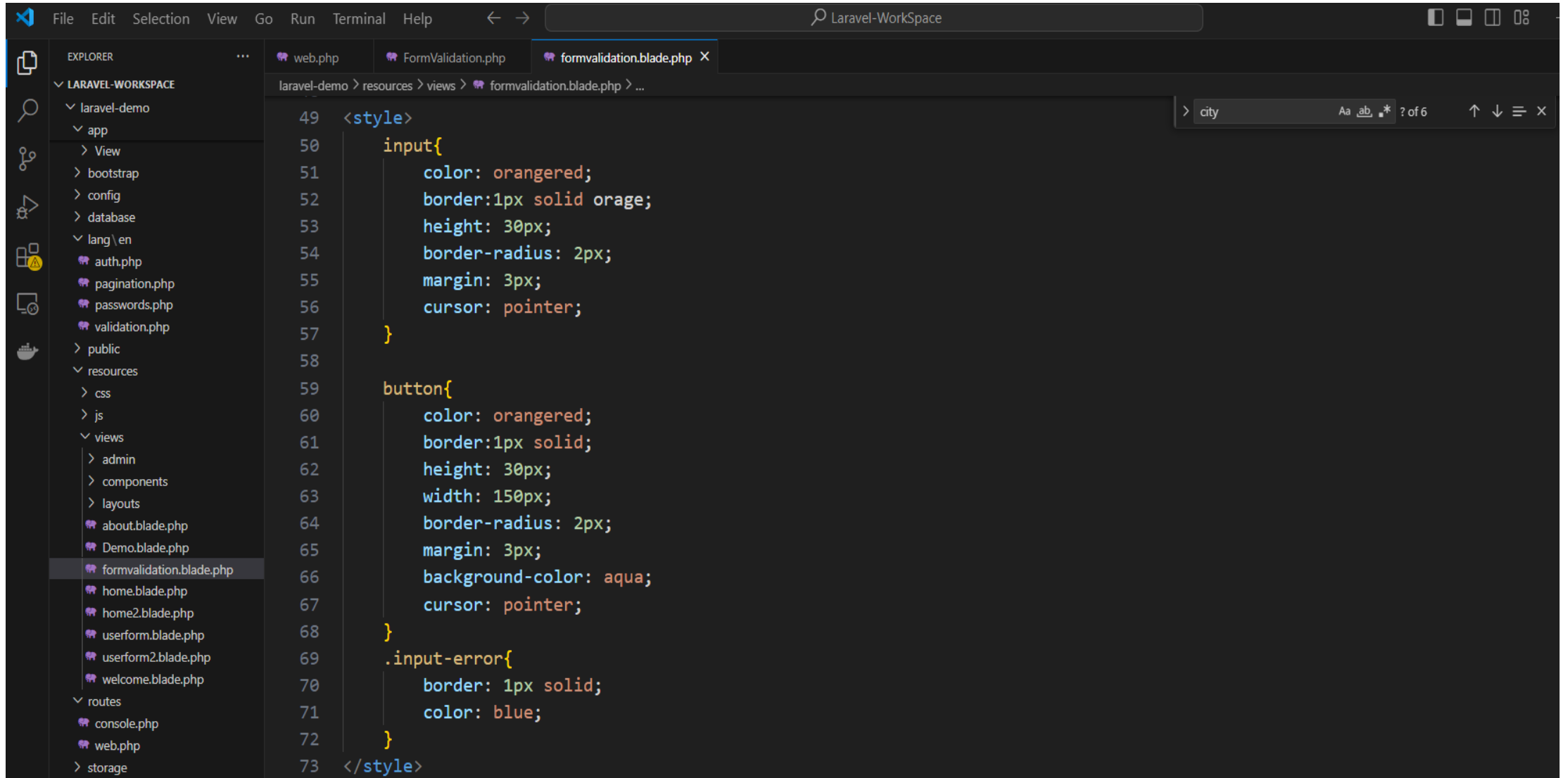
- LARAVEL-WORKSPACE
 - laravel-demo
 - app
 - View
 - bootstrap
 - config
 - database
 - lang\en
 - auth.php
 - pagination.php
 - passwords.php
 - validation.php
 - public
 - resources
 - css
 - js
 - views
 - admin
 - components
 - layouts
 - about.blade.php
 - Demo.blade.php
 - formvalidation.blade.php**
 - home.blade.php
 - home2.blade.php
 - userform.blade.php
 - userform2.blade.php
 - welcome.blade.php
 - routes
 - console.php
 - web.php
 - storage
 - tests

The main editor area displays the contents of `formvalidation.blade.php` at line 10. The code defines a form action named "validation" with the method "post". It contains two input fields: an email field and a skill selection field. The email field has a placeholder "enter email" and a class that dynamically adds the "input-error" class if there are errors. The skill field consists of three checkboxes labeled "PHP", "Java", and "Node", each with its own label and a dynamic class that adds "input-error" if there are errors. A button labeled "Add User" is also present.

```

1 <div>
10 <form action = "validation" method="post">
    24 <div class="input">
    25     <label>Email</label>
    26     <input type="text" placeholder="enter email" name="email" value="{{old('email')}}"
    27         class="{{ $errors->first('email')?'input-error':'' }}">
    28     <span style="color: red;">@error('email'){{$message}}@enderror</span>
    29 </div>
    30
    31 <div class="input">
    32     <h4>User Skill</h4>
    33     <input type="checkbox" name="skill[]" id="php" value="php">
    34     <label for="php">PHP</label>
    35     <input type="checkbox" name="skill[]" id="java" value="java">
    36     <label for="java">Java</label>
    37     <input type="checkbox" name="skill[]" id="node" value="node">
    38     <label for="node">Node</label>
    39     <span style="color: red;">@error('skill'){{$message}}@enderror</span>
    40 </div>
    41
    42 <div class="input">
    43     <button class="button">Add User</button>
    44 </div>
    45 </form>
    46 <!-- The biggest battle is the war against ignorance. - Mustafa Kemal Atatürk -->
    47 </div>
```

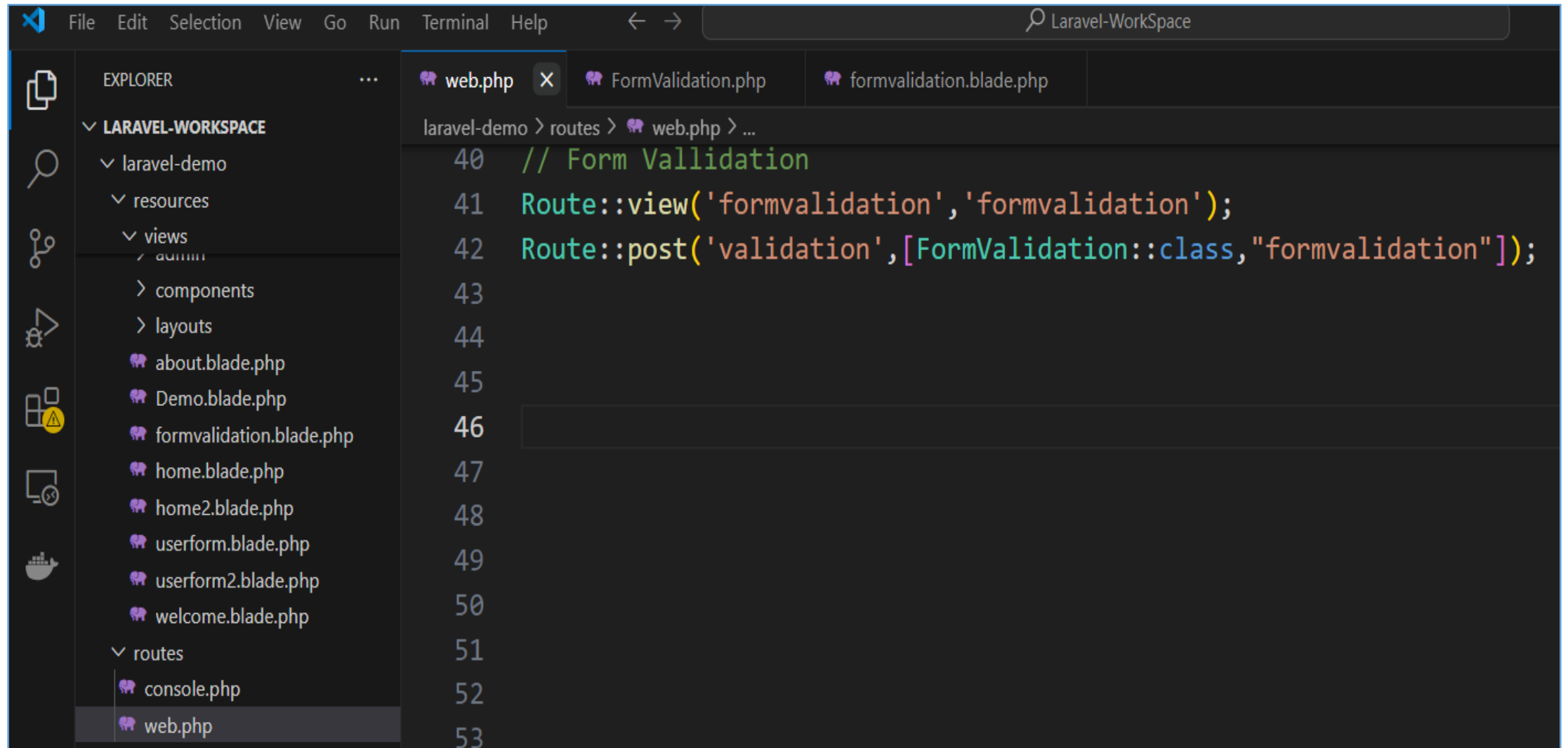
Cont. ...



The screenshot shows the Visual Studio Code interface with a Laravel project. The Explorer sidebar on the left displays the project structure, including the 'resources' directory with 'views' and 'formvalidation.blade.php' files. The main editor area shows the content of 'formvalidation.blade.php', which contains CSS styles for an input field and a button. The input field has an orange border and pointer cursor. The button has an orange border, aqua background, and pointer cursor. There is also a class for input errors with a blue border.

```
49 <style>
50     input{
51         color: orangered;
52         border:1px solid orage;
53         height: 30px;
54         border-radius: 2px;
55         margin: 3px;
56         cursor: pointer;
57     }
58
59     button{
60         color: orangered;
61         border:1px solid;
62         height: 30px;
63         width: 150px;
64         border-radius: 2px;
65         margin: 3px;
66         background-color: aqua;
67         cursor: pointer;
68     }
69     .input-error{
70         border: 1px solid;
71         color: blue;
72     }
73 </style>
```

Cont. ...



The screenshot shows the Visual Studio Code interface with a Laravel workspace. The Explorer sidebar on the left displays the project structure, including a 'resources' directory with 'views' and 'components' subdirectories, and a 'routes' directory. The main editor area shows the 'web.php' file in the 'routes' directory, containing PHP code for form validation. The code includes a comment, a GET route for 'formvalidation', and a POST route for 'validation' that uses the 'FormValidation' class. The breadcrumb navigation at the top of the editor indicates the current file path: 'laravel-demo > routes > web.php > ...'.

```
File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace

EXPLORER
LARAVEL-WORKSPACE
  laravel-demo
    resources
      views
      components
      layouts
      about.blade.php
      Demo.blade.php
      formvalidation.blade.php
      home.blade.php
      home2.blade.php
      userform.blade.php
      userform2.blade.php
      welcome.blade.php
    routes
      console.php
      web.php

web.php
FormValidation.php
formvalidation.blade.php

laravel-demo > routes > web.php > ...
40 // Form Vallidation
41 Route::view('formvalidation','formvalidation');
42 Route::post('validation',[FormValidation::class,"formvalidation"]);
43
44
45
46
47
48
49
50
51
52
53
```

Cont. ...

← → ↻ 🛡️ 📄 127.0.0.1:8000/formvalidation

Add New User

Name

User Field can Not Be Empty!!!

City

Address can Not Be Empty!!!

Email

The E-mail Should be Vallid!!!

User Skill

☐ PHP

☐ Java

☐ Node

Please Select at least One Skill!!!

Add User

User Data Are Saved

← → ↻ 127.0.0.1:8000/formvalidation

Add New User

Name

City

Email The E-mail Should be Vallid!!!

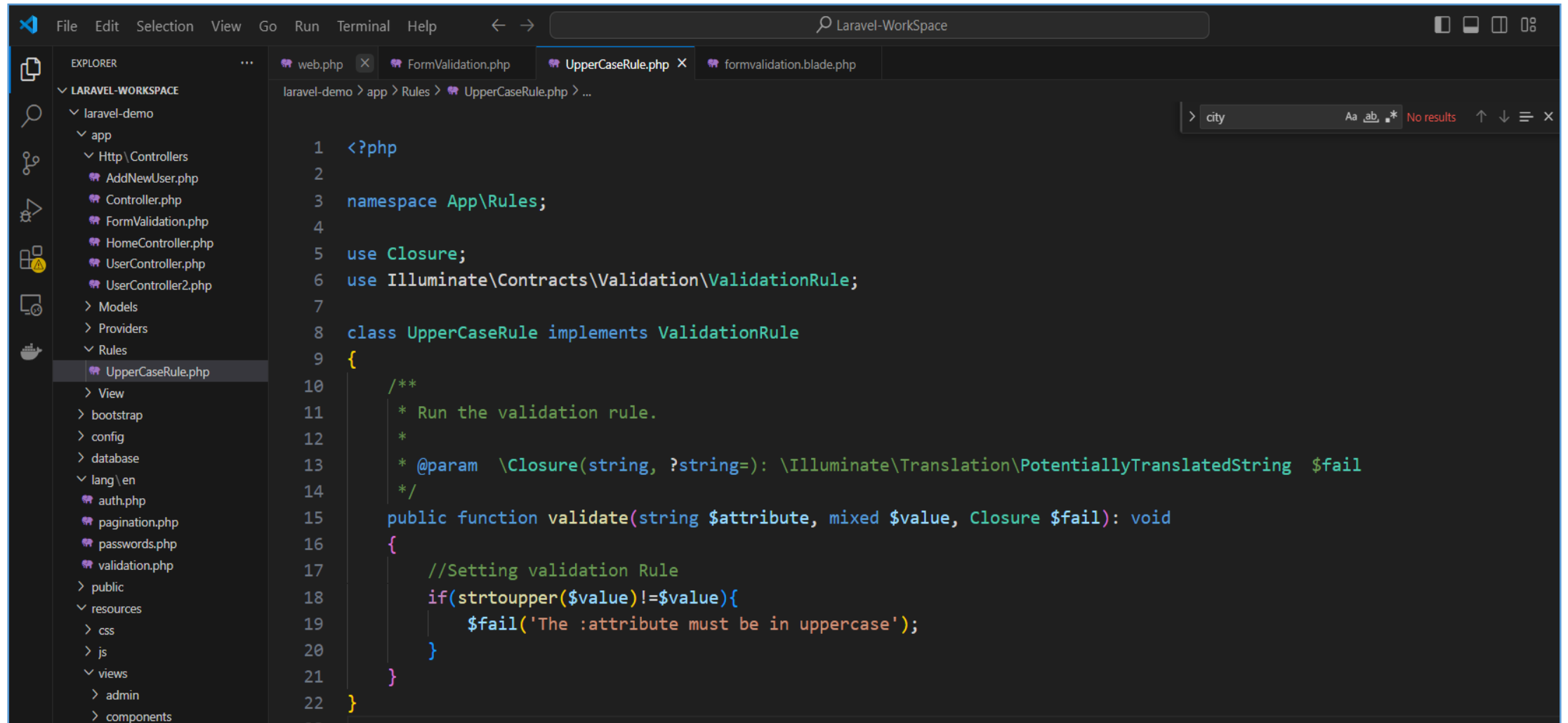
User Skill

☐ PHP ☐ Java ☐ Node Please Select at least One Skill!!!

Make Own Validation

- In Laravel, you can create custom validation rules to handle scenarios that are not covered by the **default validation rules**. This involves creating a **custom validation class** or using **closures** in your validation logic.

Cont. ...

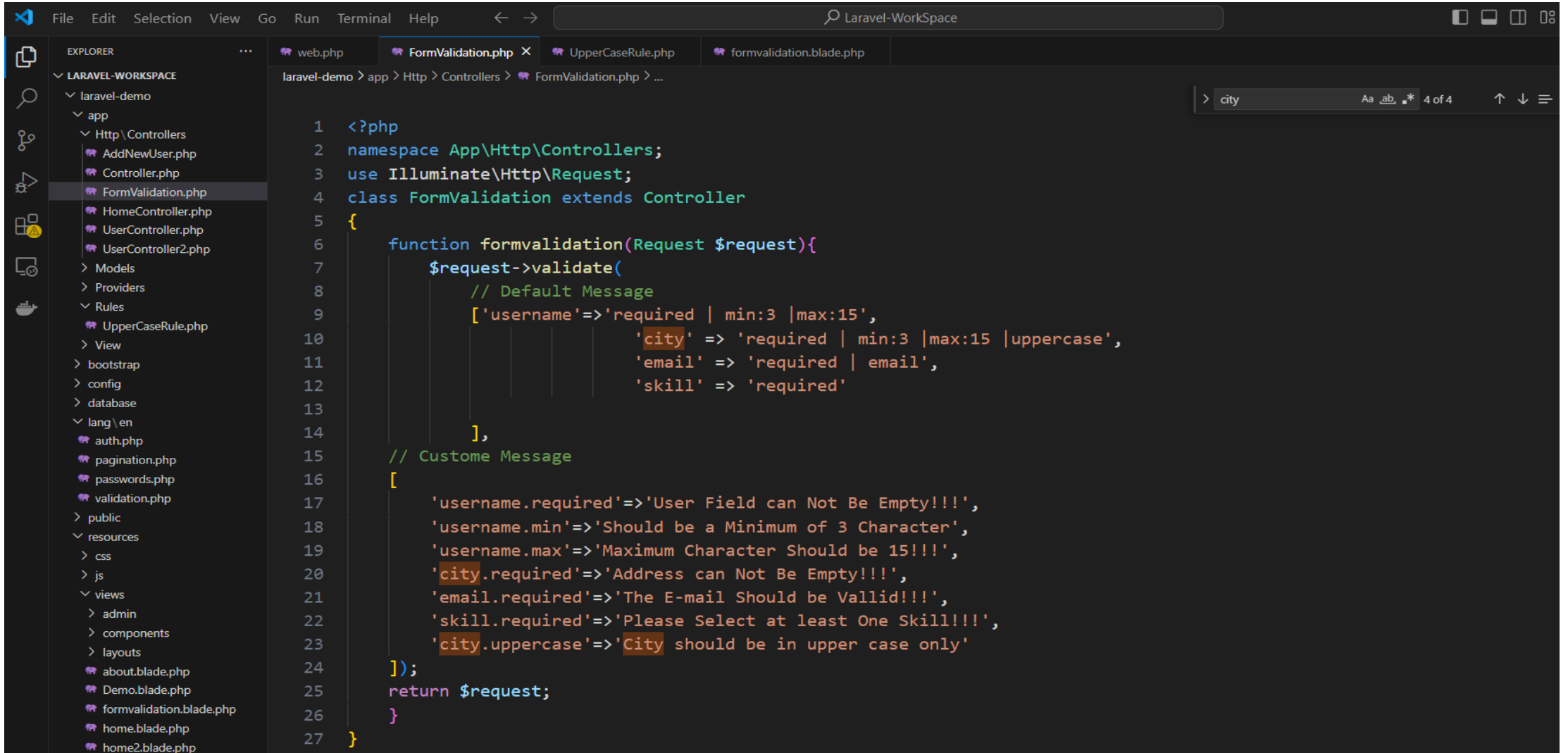


The screenshot shows the Visual Studio Code interface with a Laravel project workspace named 'Laravel-WorkSpace'. The Explorer panel on the left shows the project structure, including the 'app' directory with 'Rules' and 'UpperCaseRule.php' selected. The main editor area displays the code for 'UpperCaseRule.php'.

```
1 <?php
2
3 namespace App\Rules;
4
5 use Closure;
6 use Illuminate\Contracts\Validation\ValidationRule;
7
8 class UpperCaseRule implements ValidationRule
9 {
10     /**
11      * Run the validation rule.
12      *
13      * @param \Closure(string, ?string=): \Illuminate\Translation\PotentiallyTranslatedString $fail
14      */
15     public function validate(string $attribute, mixed $value, Closure $fail): void
16     {
17         //Setting validation Rule
18         if(strtoupper($value)!=$value){
19             $fail('The :attribute must be in uppercase');
20         }
21     }
22 }
```

A search bar in the top right corner shows the text 'city' with 'No results' displayed.

Cont. ...



```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 class FormValidation extends Controller
5 {
6     function formvalidation(Request $request){
7         $request->validate(
8             // Default Message
9             ['username'=>'required | min:3 |max:15',
10              'city' => 'required | min:3 |max:15 |uppercase',
11              'email' => 'required | email',
12              'skill' => 'required'
13             ],
14             // Custome Message
15             [
16                 'username.required'=>'User Field can Not Be Empty!!!',
17                 'username.min'=>'Should be a Minimum of 3 Character',
18                 'username.max'=>'Maximum Character Should be 15!!!',
19                 'city.required'=>'Address can Not Be Empty!!!',
20                 'email.required'=>'The E-mail Should be Vallid!!!',
21                 'skill.required'=>'Please Select at least One Skill!!!',
22                 'city.uppercase'=>'City should be in upper case only'
23             ]
24         );
25         return $request;
26     }
27 }
```

Cont. ...

← → ↻ 127.0.0.1:8000/formvalidation

Add New User

Name

City City should be in upper case only

Email The E-mail Should be Vallid!!!

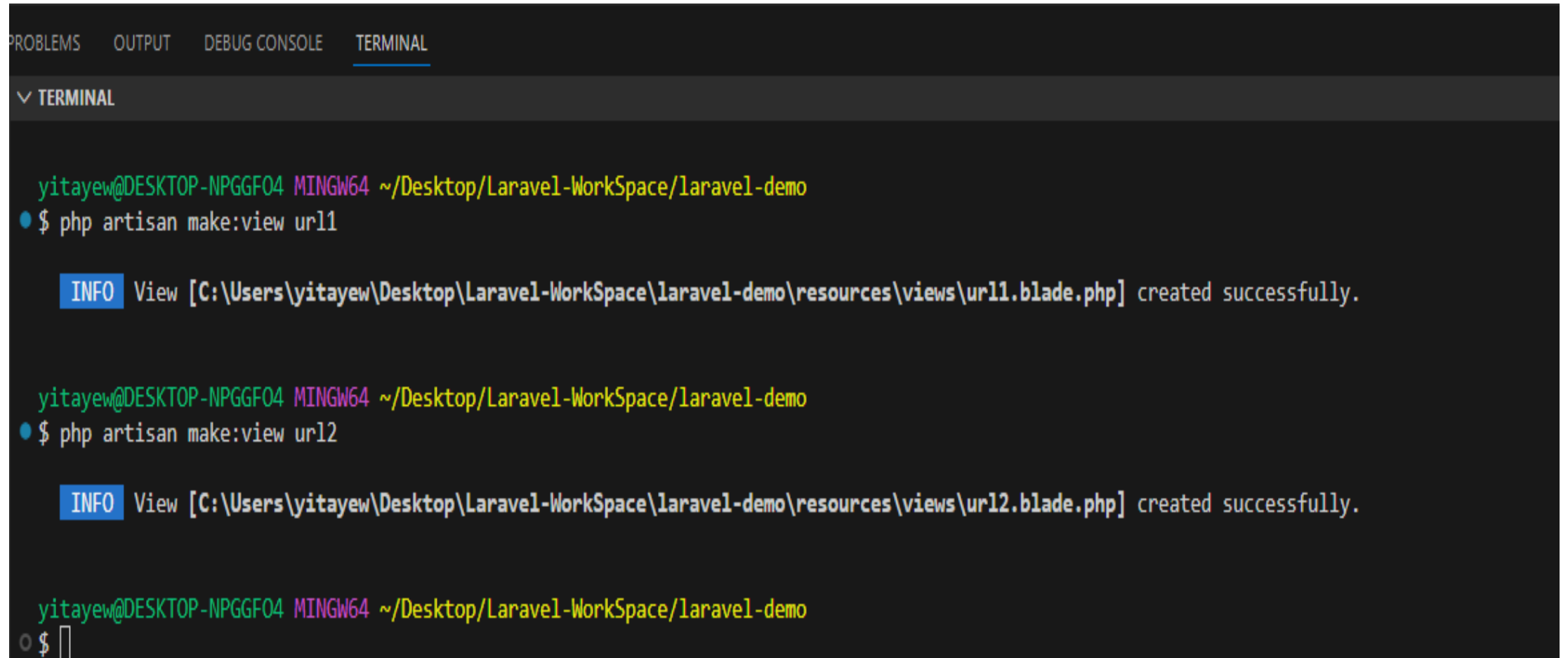
User Skill

☐ PHP ☐ Java ☐ Node Please Select at least One Skill!!!

URL Generation

- In Laravel, URL generation is a core feature that allows you to create URLs dynamically within your application, making it easier to manage links and routes.
- Laravel provides several **helper functions** and **methods** to generate URLs.

Creating Views



The screenshot shows a terminal window with the following content:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
```

▼ TERMINAL

```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
● $ php artisan make:view url1
```

INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\url1.blade.php] created successfully.

```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
● $ php artisan make:view url2
```

INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\url2.blade.php] created successfully.

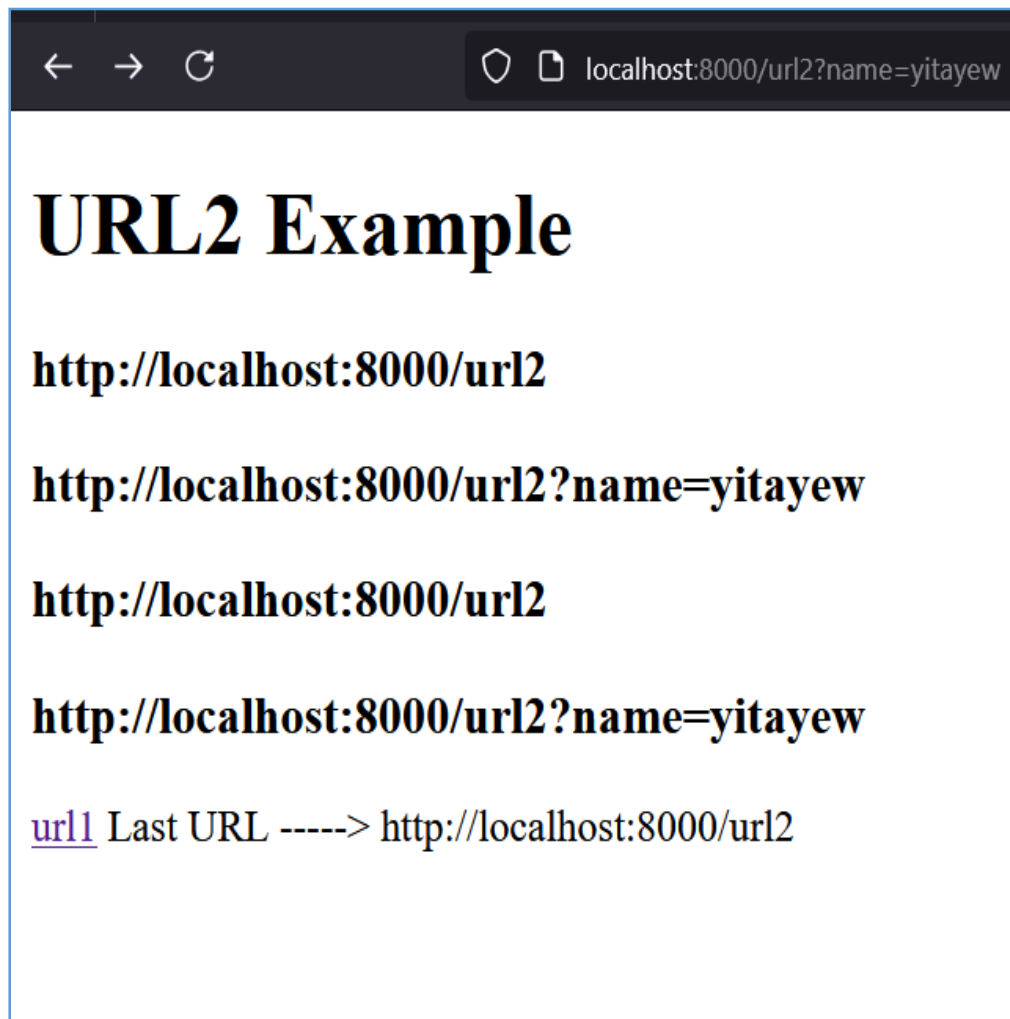
```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
○ $
```

Cont. ...

```
web.php X url1.blade.php X url2.blade.php
laravel-demo > resources > views > url1.blade.php > ...
1 <div>
2     <!-- Do what you can, with what you have,
3     <h1>URL one Example</h1>
4
5     <!-- For Checking Last URL -->
6     <a href="url2">url2</a>
7     Last URL ----> {{URL::previous()}}
8 </div>
9
10 44 // URL Generation Example
11 45 Route::view('url1','url1');
    46 Route::view('userurl1','url1');
    47 Route::view('url2','url2');
```

```
web.php X url1.blade.php X url2.blade.php X
laravel-demo > resources > views > url2.blade.php > div > a
1 <div>
2     <!-- Do what you can, with what you have,
3     <h1>URL2 Example</h1>
4     <!-- Accessesing Current URL -->
5     <h3>{{URL::current()}}</h3>
6     <!-- full URL -->
7     <h3>{{URL::full()}}</h3>
8
9     <!-- Alternative Method Same Output-->
10    <h3>{{url()->current()}}</h3>
11    <h3>{{url()->full()}}</h3>
12
13    <!-- For checking Last URL -->
14    <a href="url1">url1</a>
15    Last URL ----> {{URL::previous()}}
16 </div>
```

Cont. ...



A screenshot of a web browser window. The address bar shows 'localhost:8000/url2?name=yitayew'. The main content area has the title 'URL2 Example' and a list of four URLs: 'http://localhost:8000/url2', 'http://localhost:8000/url2?name=yitayew', 'http://localhost:8000/url2', and 'http://localhost:8000/url2?name=yitayew'. At the bottom, there is a link labeled 'url1' followed by the text 'Last URL -----> http://localhost:8000/url2'.

URL2 Example

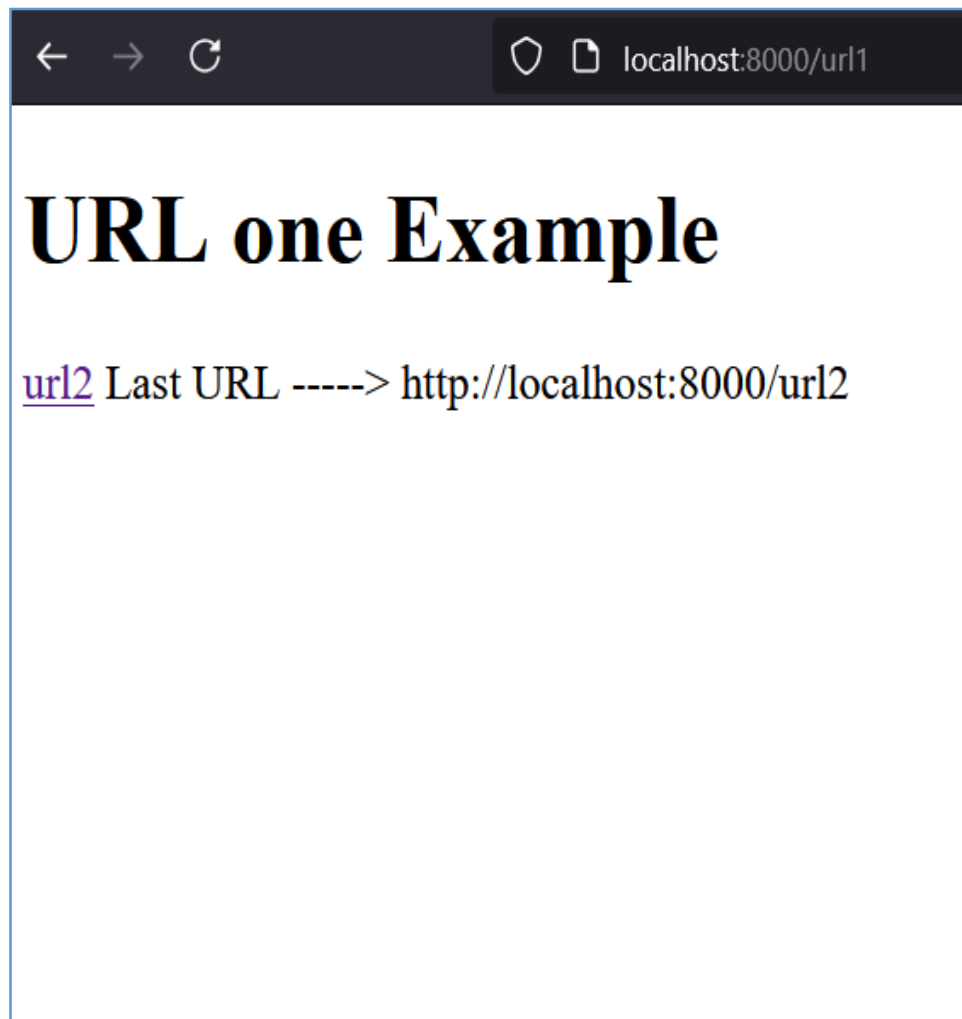
`http://localhost:8000/url2`

`http://localhost:8000/url2?name=yitayew`

`http://localhost:8000/url2`

`http://localhost:8000/url2?name=yitayew`

[url1](#) Last URL -----> `http://localhost:8000/url2`



A screenshot of a web browser window. The address bar shows 'localhost:8000/url1'. The main content area has the title 'URL one Example' and a link labeled 'url2' followed by the text 'Last URL -----> http://localhost:8000/url2'.

URL one Example

[url2](#) Last URL -----> `http://localhost:8000/url2`

Named Route in Laravel

- A named route in Laravel is a route with a **specific name assigned** to it. Named routes allow you to reference the route by its **name** rather than by its URL path, which makes your code more flexible and easier to maintain.
- Instead of hardcoding URLs throughout your application, you can use named routes to generate URLs or redirects by referring to the route name.

Why Use Named Routes?

- ❖ **Maintainability:** If the URL of a route changes, you only need to update the route definition instead of updating every instance of the URL in your application.
- ❖ **Readability:** Named routes make it easier to understand the purpose of a route.
- ❖ **Flexibility:** Named routes support generating URLs, handling redirects, and passing parameters dynamically.


Defining Named Route

Defining Named Routes

To define a named route in Laravel, use the `name` method on the route definition.

Example

php

 Copy code

```
Route::get('/user/profile', [UserController::class, 'showProfile'])->name('profile');
```

In this example:

- The route `'/user/profile'` is named `profile`.
- You can now reference this route anywhere in your application by the name `profile` instead of the full URL path.


Cont. ...

Using Named Routes with Parameters

If a route requires parameters, you can pass them as an associative array in the `route()` helper.

Example

php

 Copy code


```
Route::get('/user/{id}/profile', [UserController::class, 'showProfile'])->name('profile');

// Generating URL with a parameter
$url = route('profile', ['id' => 1]); // Output: /user/1/profile
```

Using Named Routes in Blade Templates

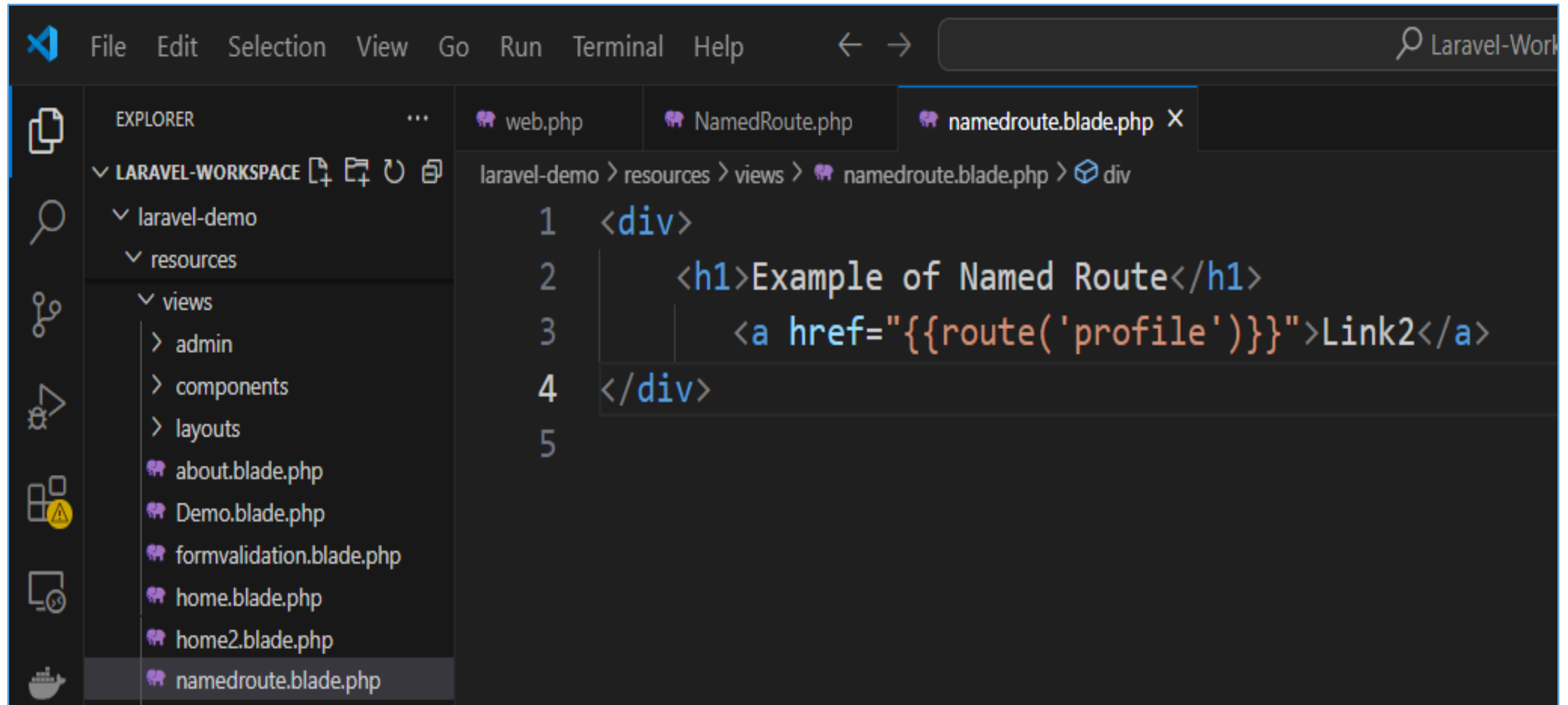
In Blade templates, you can use named routes for links or form actions.

blade

 Copy code

```
<a href="{{ route('profile') }}">Profile</a>
```

Example

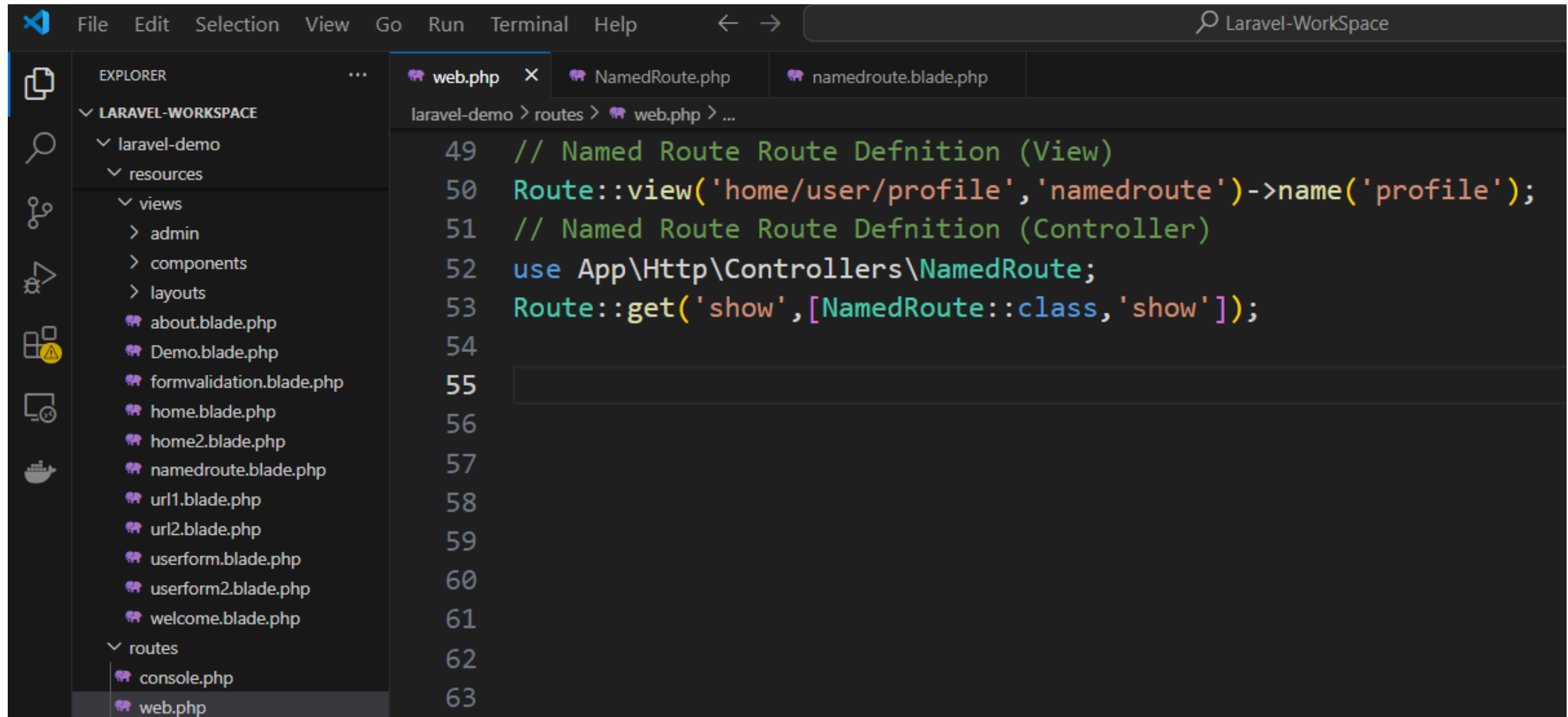


The screenshot shows the Visual Studio Code interface with a Laravel workspace. The Explorer panel on the left shows the project structure: **LARAVEL-WORKSPACE** > **laravel-demo** > **resources** > **views**. The **views** folder is expanded, showing subfolders **admin**, **components**, and **layouts**, and several Blade template files. The file **namedroute.blade.php** is selected and highlighted in the Explorer. The main editor area shows the content of **namedroute.blade.php** with the following code:

```
1 <div>
2     <h1>Example of Named Route</h1>
3     <a href="{{route('profile')}}">Link2</a>
4 </div>
5
```

The breadcrumb navigation at the top of the editor indicates the path: **laravel-demo > resources > views > namedroute.blade.php > div**. The file tabs at the top show **web.php**, **NamedRoute.php**, and **namedroute.blade.php** (which is the active file).

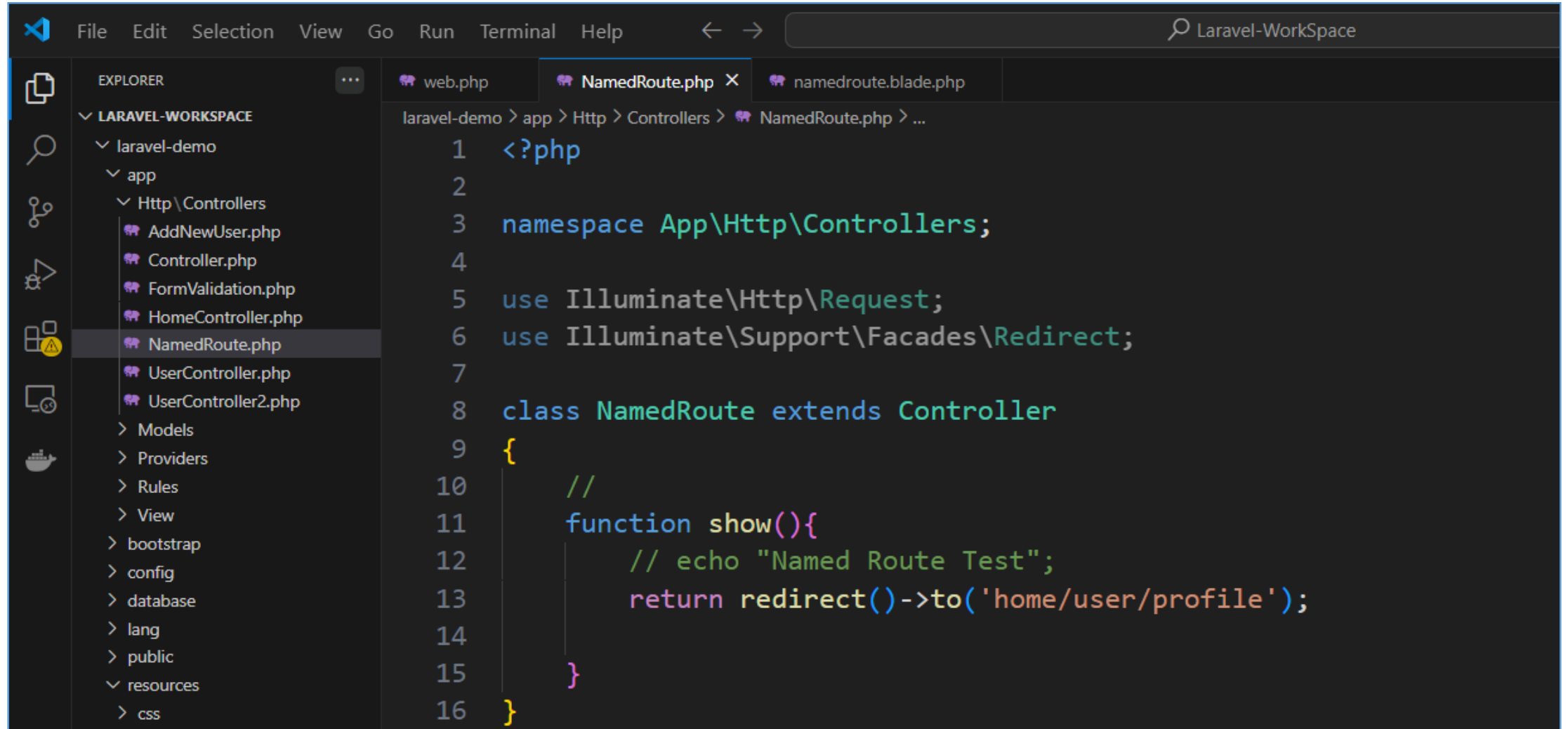
Cont. ...



The screenshot displays the Visual Studio Code interface for a Laravel project. The Explorer sidebar on the left shows the project structure, including the 'resources' folder, 'views' subfolder, and the 'routes' folder. The main editor area shows the 'web.php' file in the 'routes' folder, containing the following code:

```
49 // Named Route Route Definition (View)
50 Route::view('home/user/profile','namedroute')->name('profile');
51 // Named Route Route Definition (Controller)
52 use App\Http\Controllers\NamedRoute;
53 Route::get('show',[NamedRoute::class,'show']);
54
55
56
57
58
59
60
61
62
63
```

Cont. ...



```
File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace

EXPLORER
LARAVEL-WORKSPACE
  laravel-demo
    app
      Http\Controllers
        AddNewUser.php
        Controller.php
        FormValidation.php
        HomeController.php
        NamedRoute.php
        UserController.php
        UserController2.php
    Models
    Providers
    Rules
    View
    bootstrap
    config
    database
    lang
    public
    resources
      css

web.php NamedRoute.php x namedroute.blade.php
laravel-demo > app > Http > Controllers > NamedRoute.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Redirect;
7
8  class NamedRoute extends Controller
9  {
10     //
11     function show(){
12         // echo "Named Route Test";
13         return redirect()->to('home/user/profile');
14     }
15 }
16 }
```

Cont. ...



Route Group with a Prefix


- In Laravel, a route group with a prefix allows you to apply a common URL prefix to multiple routes, **simplifying route management**.
- This is useful for **organizing routes** with similar structures, such as routes for the admin panel, API, or user profile sections.

Cont. ...

Example

Let's say we want to group all routes related to the "admin" section under the `/admin` prefix.

php

 Copy code

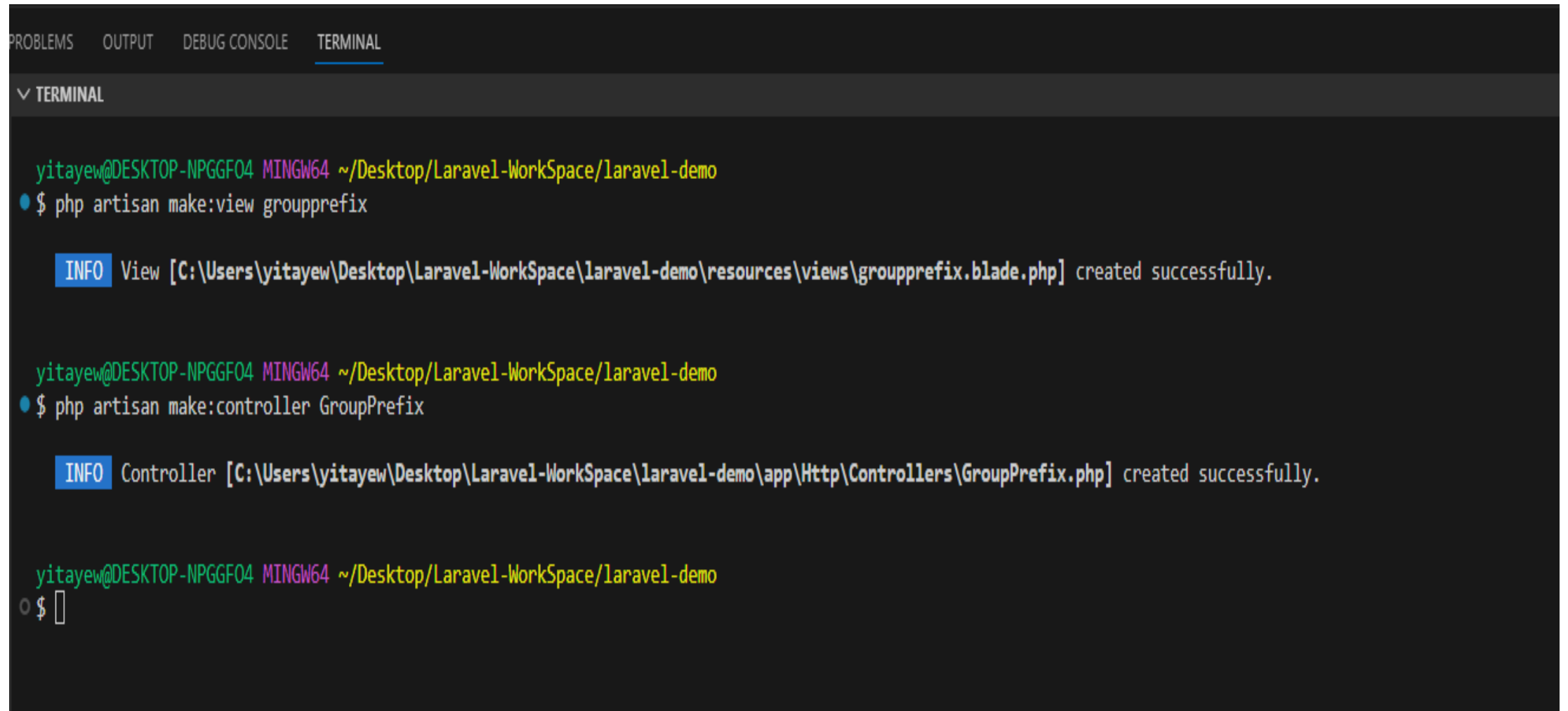
```
// routes/web.php

Route::prefix('admin')->group(function () {
    Route::get('/dashboard', [AdminController::class, 'dashboard'])->name('admin.dashboard');
    Route::get('/users', [AdminController::class, 'users'])->name('admin.users');
    Route::get('/settings', [AdminController::class, 'settings'])->name('admin.settings');
});
```

In this example:

- The routes `/admin/dashboard`, `/admin/users`, and `/admin/settings` will be created.
- Each route has a name prefixed with `admin.` (e.g., `admin.dashboard`), making it easier to manage within the application.

Example



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal displays three separate command sessions. Each session starts with a prompt indicating the user is 'yitayew@DESKTOP-NPGGF04' in a 'MINGW64' environment at the directory '~/Desktop/Laravel-WorkSpace/laravel-demo'. The first session shows the command 'php artisan make:view groupprefix' followed by an 'INFO' message stating that the view file 'groupprefix.blade.php' was created successfully. The second session shows the command 'php artisan make:controller GroupPrefix' followed by an 'INFO' message stating that the controller file 'GroupPrefix.php' was created successfully. The third session shows a prompt with a cursor, indicating the user is ready to enter another command.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
▼ TERMINAL

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
• $ php artisan make:view groupprefix

INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\groupprefix.blade.php] created successfully.

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
• $ php artisan make:controller GroupPrefix

INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\GroupPrefix.php] created successfully.

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
o $
```

Cont. ...

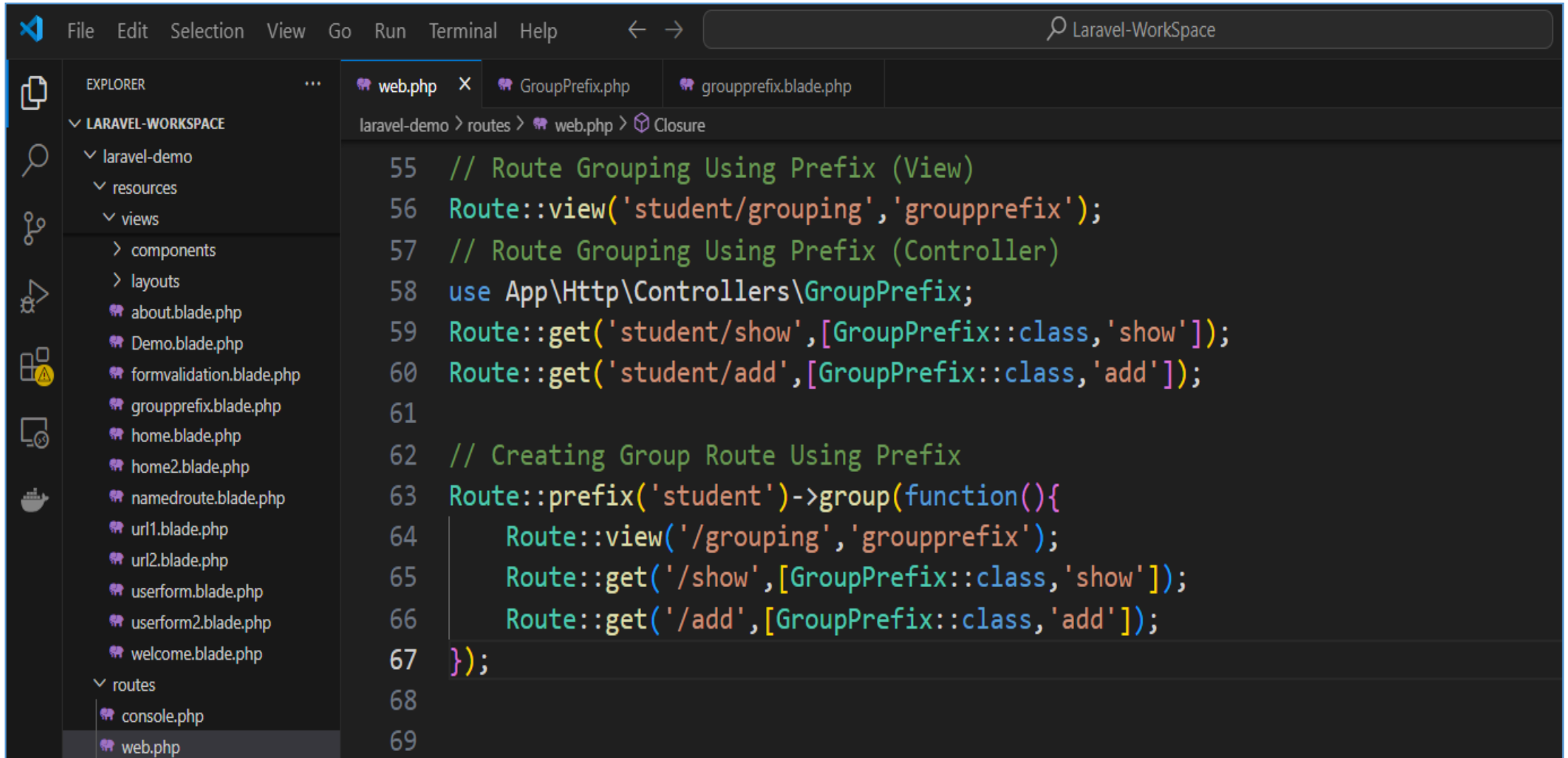
```
File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace

EXPLORER
LARAVEL-WORKSPACE
  laravel-demo
    app
      Http\ Controllers
        AddNewUser.php
        Controller.php
        FormValidation.php
        GroupPrefix.php
        HomeController.php
        NamedRoute.php
        UserController.php
        UserController2.php
    Models
    Providers
    Rules
    View
    bootstrap
    config
    database
    lang
    public
    resources
      css
      js
      views
        admin
        components
        layouts
        about.blade.php
        Demo.blade.php
        formvalidation.blade.php
        groupprefix.blade.php

laravel-demo > app > Http > Controllers > GroupPrefix.php > GroupPrefix > add
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class GroupPrefix extends Controller
8  {
9      //
10     function show(){
11         echo "show student list";
12     }
13     function add(){
14
15         return "Students are added to the list";
16     }
17 }
18

laravel-demo > resources > views > groupprefix.blade.php > ...
1  <div>
2      <!-- Well begun is half done. - Aristotle -->
3      <h1>Route Grouping With Prefix</h1>
4  </div>
```

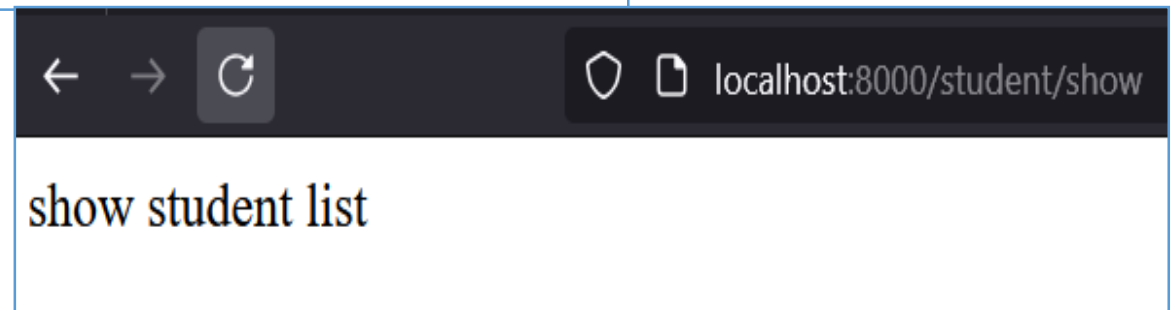
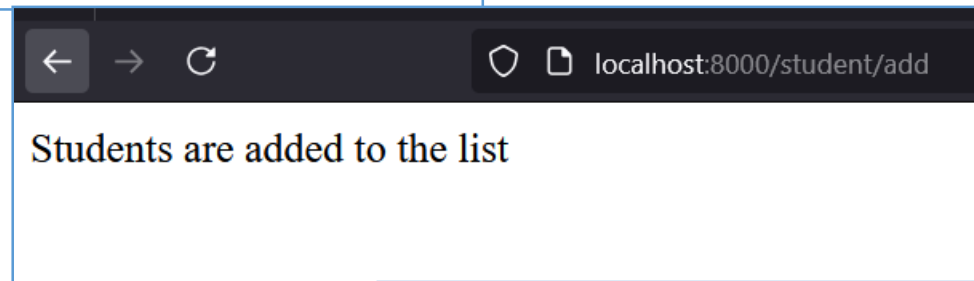
Cont. ...



The screenshot shows the Visual Studio Code interface with a Laravel project. The Explorer sidebar on the left shows the file structure: **LARAVEL-WORKSPACE** > **laravel-demo** > **resources** > **views**. Under **views**, there are subfolders **components** and **layouts**, and a list of Blade templates including **about.blade.php**, **Demo.blade.php**, **formvalidation.blade.php**, **groupprefix.blade.php**, **home.blade.php**, **home2.blade.php**, **namedroute.blade.php**, **url1.blade.php**, **url2.blade.php**, **userform.blade.php**, **userform2.blade.php**, and **welcome.blade.php**. The **routes** folder is expanded, showing **console.php** and **web.php**. The **web.php** file is open in the editor, showing the following PHP code:

```
55 // Route Grouping Using Prefix (View)
56 Route::view('student/grouping','groupprefix');
57 // Route Grouping Using Prefix (Controller)
58 use App\Http\Controllers\GroupPrefix;
59 Route::get('student/show',[GroupPrefix::class,'show']);
60 Route::get('student/add',[GroupPrefix::class,'add']);
61
62 // Creating Group Route Using Prefix
63 Route::prefix('student')->group(function(){
64     Route::view('/grouping','groupprefix');
65     Route::get('/show',[GroupPrefix::class,'show']);
66     Route::get('/add',[GroupPrefix::class,'add']);
67 });
68
69
```

Cont. ...



Route Group With Controller


- In Laravel, you can create route groups with a common controller to streamline routing for multiple methods within the same controller. This is especially useful when multiple actions in a controller share a common path or prefix.
- ❖ **Defining a Route Group with a Controller:** To define a route group with a controller, use the `controller` method within a route group. This method assigns the specified controller to all routes in the group, simplifying the code and reducing redundancy.

Example

Example

Suppose you have a `UserController` handling actions like showing a profile, updating user information, and deleting a user. You can set up a route group with a controller to handle these actions:

php

 Copy code

```
// routes/web.php

use App\Http\Controllers\UserController;

Route::controller(UserController::class)->group(function () {
    Route::get('/profile', 'showProfile')->name('user.profile');
    Route::post('/update', 'updateProfile')->name('user.update');
    Route::delete('/delete', 'deleteUser')->name('user.delete');
});
```


Cont. ...

The screenshot shows the Visual Studio Code interface with a Laravel project. The Explorer sidebar on the left shows the project structure, with the `StudentController.php` file selected under `app/Http/Controllers`. The main editor displays the code for `StudentController.php`, which includes namespace declarations, a use statement for `Illuminate\Http\Request`, and a class `StudentController` extending `Controller`. The class contains four methods: `show()`, `add()`, `delete()`, and `about($name)`. The bottom panel shows the TERMINAL output, indicating that the controller was created successfully.

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4
5 class StudentController extends Controller
6 {
7     //
8     function show(){
9         return "Student List are added!!";
10    }
11    function add(){
12        return "Students are added!!";
13    }
14    function delete(){
15        return "Students are deleted!!";
16    }
17    function about($name){
18        return "Your Name is: ".$name;
19    }
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

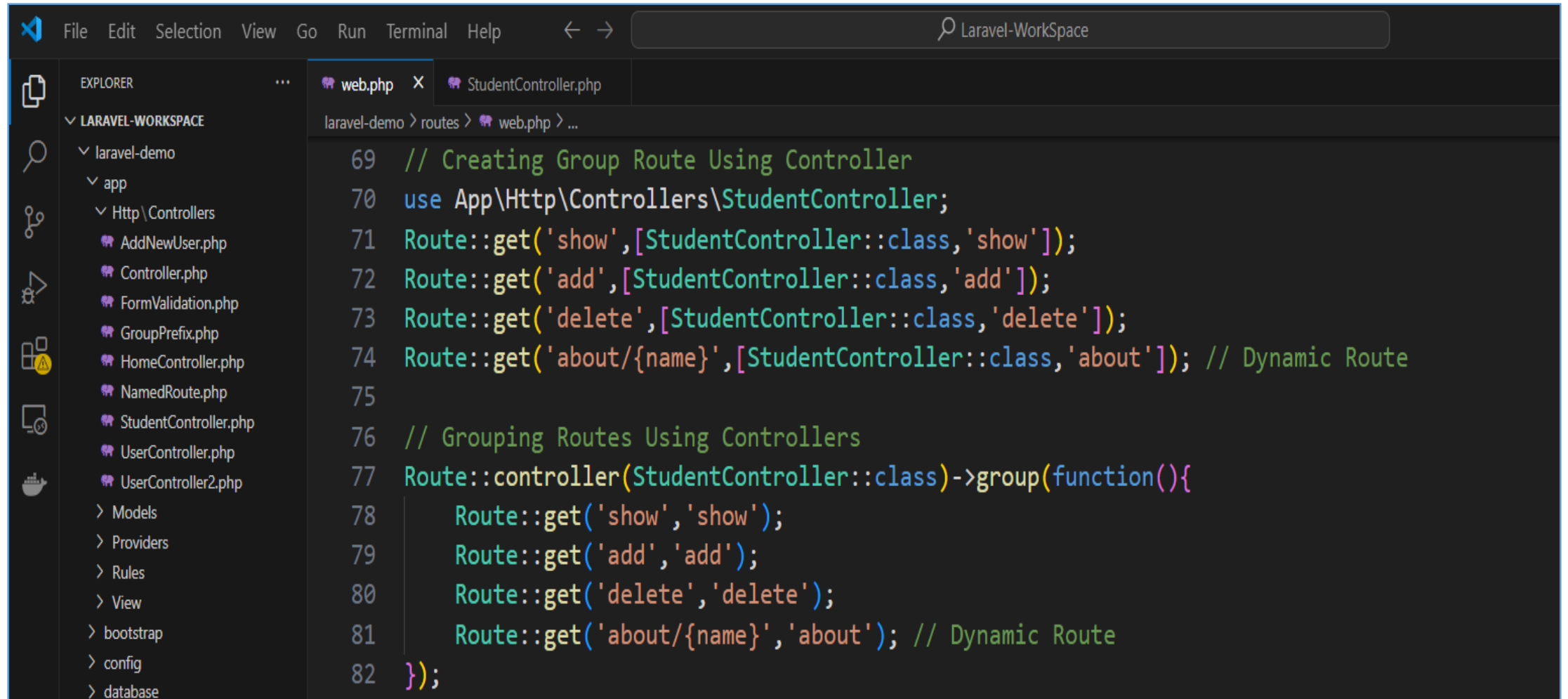
> TERMINAL

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo

• \$ php artisan make:controller StudentController

INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\StudentController.php] created successfully.

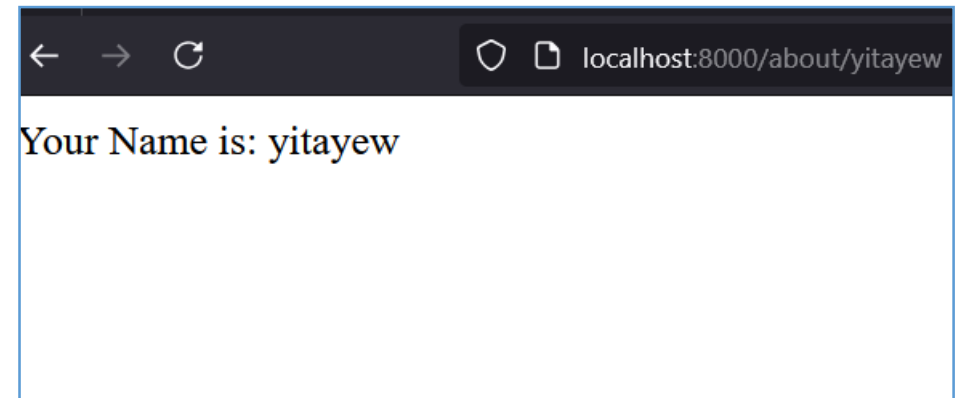
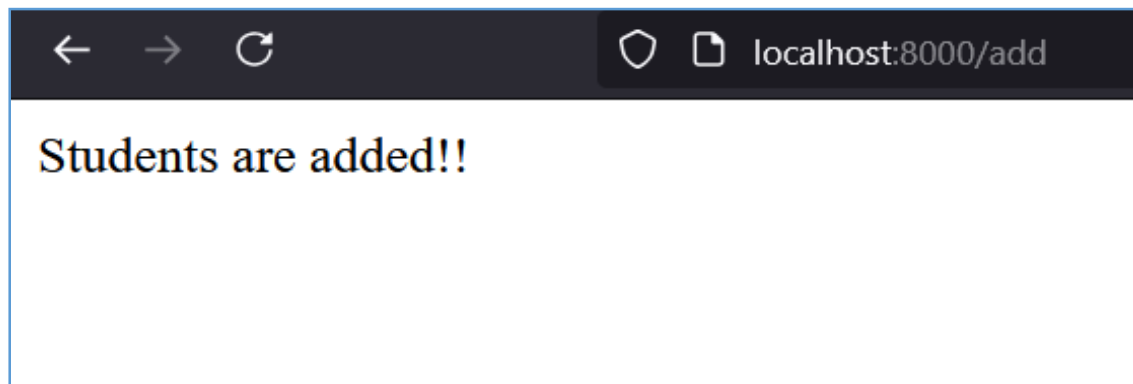
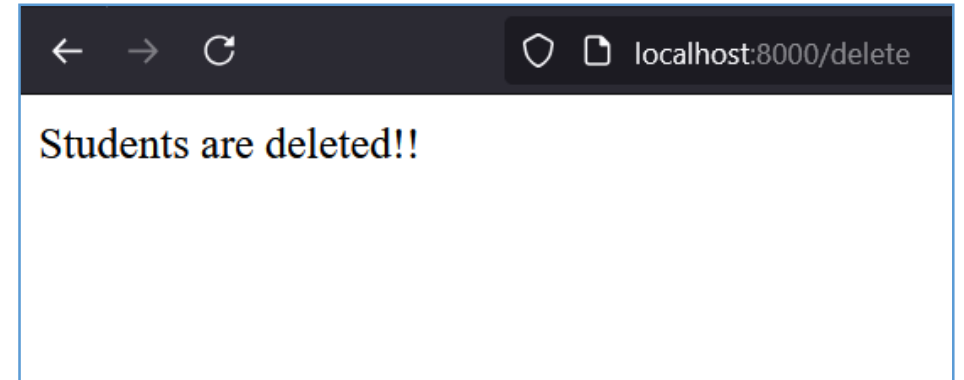
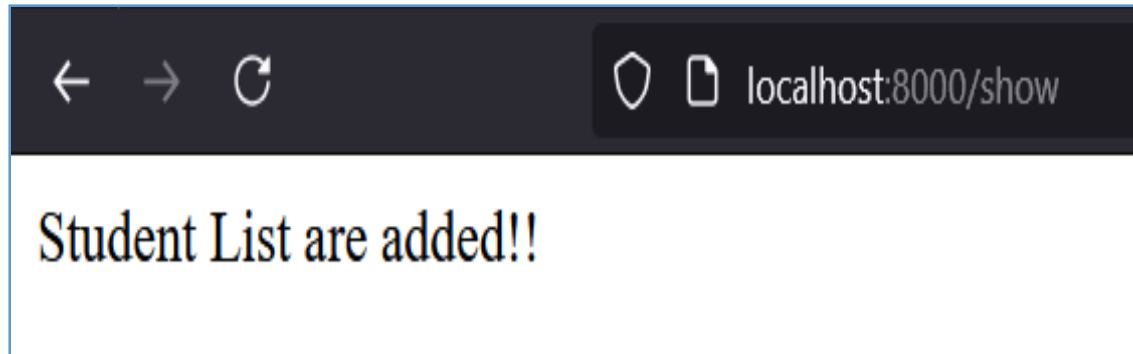
Cont. ...



The screenshot shows a code editor with a dark theme. The Explorer sidebar on the left shows a project structure for 'laravel-demo' with folders 'app', 'bootstrap', 'config', and 'database'. The 'app' folder is expanded, showing 'Http\Controllers' with files like 'AddNewUser.php', 'Controller.php', 'FormValidation.php', 'GroupPrefix.php', 'HomeController.php', 'NamedRoute.php', 'StudentController.php', 'UserController.php', and 'UserController2.php'. The main editor area shows the 'web.php' file in the 'routes' directory. The code defines routes for a 'StudentController' using both individual route definitions and a grouped controller route.

```
69 // Creating Group Route Using Controller
70 use App\Http\Controllers\StudentController;
71 Route::get('show',[StudentController::class,'show']);
72 Route::get('add',[StudentController::class,'add']);
73 Route::get('delete',[StudentController::class,'delete']);
74 Route::get('about/{name}',[StudentController::class,'about']); // Dynamic Route
75
76 // Grouping Routes Using Controllers
77 Route::controller(StudentController::class)->group(function(){
78     Route::get('show','show');
79     Route::get('add','add');
80     Route::get('delete','delete');
81     Route::get('about/{name}','about'); // Dynamic Route
82 });
```

Cont. ...



Thank you!

Appreciate your action.