



Outlines of Discussion

- ❖ Database Query Builder
- ❖ Eloquent ORM
- ❖ Route Methods in Laravel
- ❖ HTTP Request Class
- ❖ Session in laravel

Database Query Builder

- Laravel's Query Builder provides a convenient, fluent **interface** for creating and running database queries. It is a **layer between raw SQL queries** and **Eloquent ORM**, allowing you to work with databases without writing raw SQL.
- The Query Builder supports all **major database** systems, including MySQL, PostgreSQL, SQLite, and SQL Server.

Cont...

- The **Database Query Builder** in Laravel is a powerful and flexible interface for building and executing database queries in a structured, fluent, and programmatic way.
- It allows developers to interact with the database without writing **raw SQL queries**, making the code cleaner, readable, and secure.

Key Features

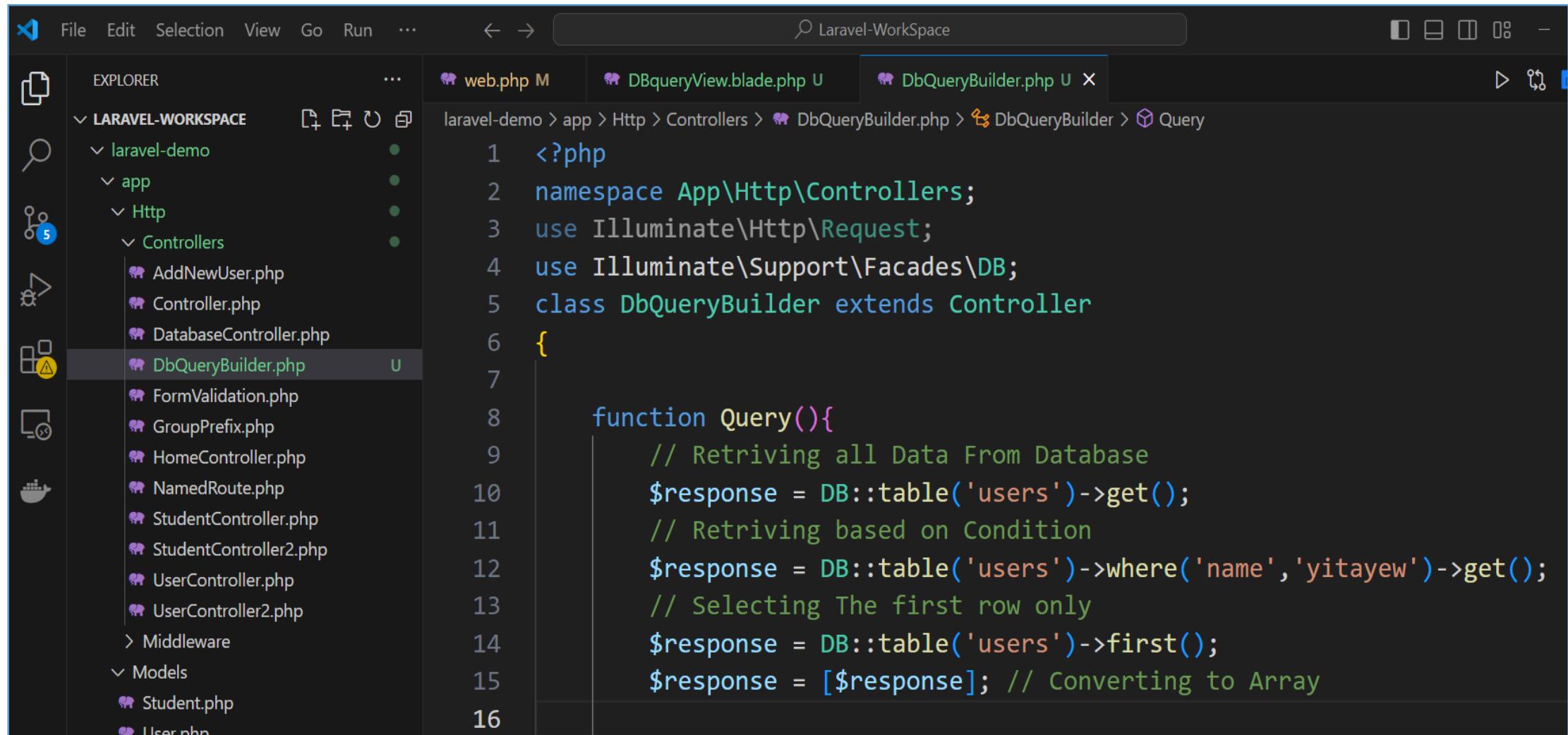
- ❖ **Chainable Methods:** Query Builder methods are chainable, making queries more readable and easier to build.
- ❖ **Protection Against SQL Injection:** Automatically binds parameters, making queries safer.
- ❖ **Supports CRUD Operations:** Includes methods for all basic database operations.

Example

```
> ✓ TERMINAL
⚡ yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:controller DbQueryBuilder
    INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\DbQueryBuilder.php] created successfully.

● yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
$ php artisan make:view DBqueryView
    INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\DBqueryView.blade.php] created successfully.
```

Data Manipulation Using Laravel

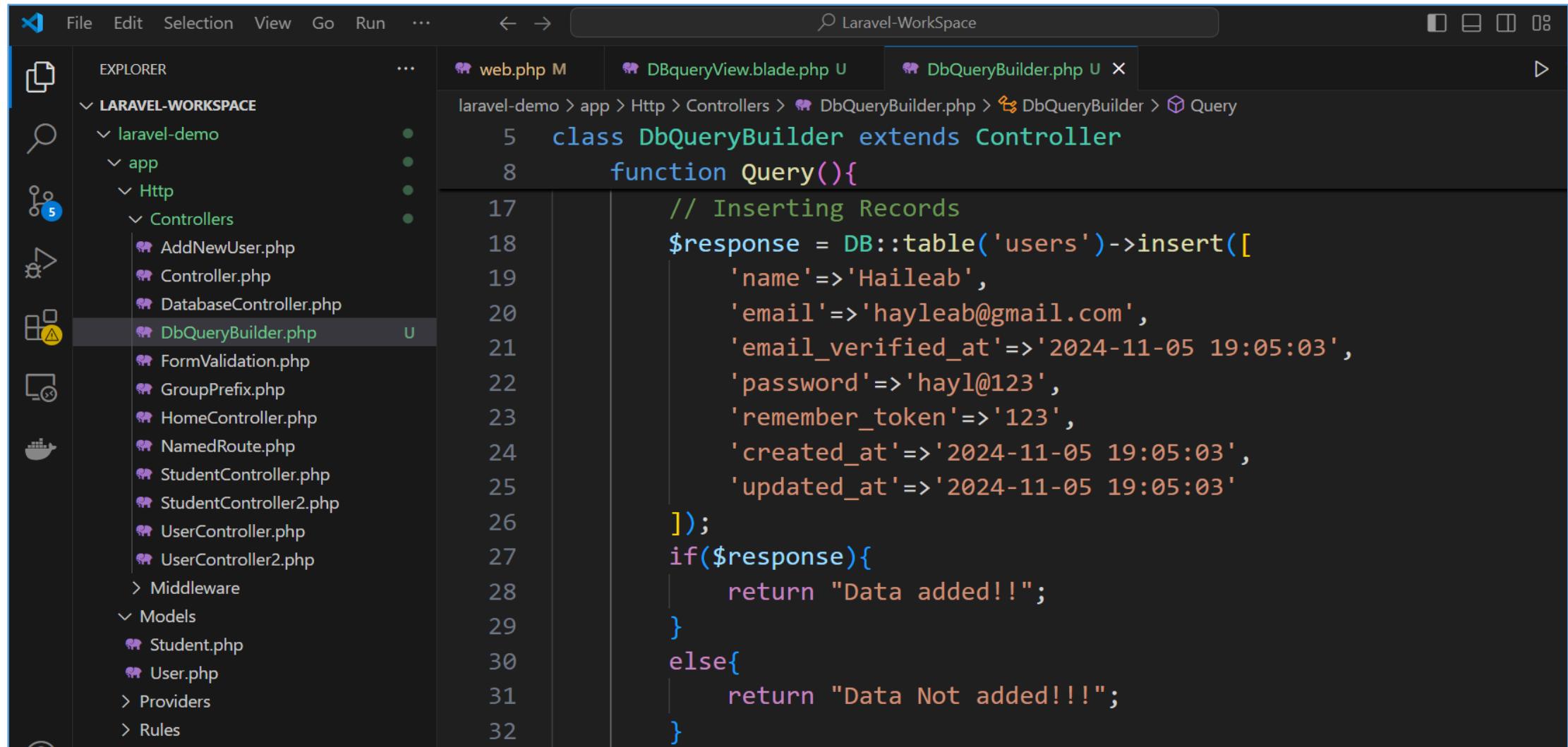


The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Laravel-WorkSpace
- Toolbars:** Standard icons for file operations.
- Left Sidebar (EXPLORER):** Shows the project structure:
 - LARAVEL-WORKSPACE
 - laravel-demo
 - app
 - Http
 - Controllers
 - AddNewUser.php
 - Controller.php
 - DatabaseController.php
 - DbQueryBuilder.php** (selected)
 - FormValidation.php
 - GroupPrefix.php
 - HomeController.php
 - NamedRoute.php
 - StudentController.php
 - StudentController2.php
 - UserController.php
 - UserController2.php
 - Middleware
 - Models
 - Student.php
 - User.php
 - Central Area:** Displays the content of **DbQueryBuilder.php**. The code is as follows:

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\DB;
5 class DbQueryBuilder extends Controller
6 {
7
8     function Query(){
9         // Retrieving all Data From Database
10        $response = DB::table('users')->get();
11        // Retrieving based on Condition
12        $response = DB::table('users')->where('name','yitayew')->get();
13        // Selecting The first row only
14        $response = DB::table('users')->first();
15        $response = [$response]; // Converting to Array
16    }
}
```

Inserting Records



The screenshot shows a code editor interface with the title bar "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The current file is "DbQueryBuilder.php" located in the "Http\Controllers" directory. The code in the editor is as follows:

```
5 class DbQueryBuilder extends Controller
8     function Query(){
17         // Inserting Records
18         $response = DB::table('users')->insert([
19             'name'=>'Haileab',
20             'email'=>'hayleab@gmail.com',
21             'email_verified_at'=>'2024-11-05 19:05:03',
22             'password'=>'hayl@123',
23             'remember_token'=>'123',
24             'created_at'=>'2024-11-05 19:05:03',
25             'updated_at'=>'2024-11-05 19:05:03'
26         ]);
27         if($response){
28             return "Data added!!";
29         }
30         else{
31             return "Data Not added!!!";
32         }
}
```

Update and Delete Records

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The current file, 'DbQueryBuilder.php', is selected and shown in the main editor area. The code implements a controller method to update or delete a user record based on their name.

```
class DbQueryBuilder extends Controller
{
    function Query(){
        // Updatting Record
        $response = DB::table('users')->where('name', 'haileab')->update(
            ['password'=>'hayl@1234']);
        if($response){
            return "Data Updated!!";
        }
        else{
            return "Data Not Updated!!!";
        }
        // Deleting Record
        $response = DB::table('users')->where('name', 'haileab')->delete();
        if($response){
            return "Data Deleted!!";
        }
        else{
            return "Data Not Deleted!!!";
        }
        return view('DBqueryView',[ 'users'=>$response]);
    }
}
```

Route

```
// Model in Laravel
use App\Http\Controllers\StudentController2;
Route::get('students',[StudentController2::class,'getstudent']);

// Database Class Quiry Builder
use App\Http\Controllers\DbQueryBuilder;
Route::get('querybuilder',[DbQueryBuilder::class,'Query']);✖

// Eleouquent Model Query Builder
use App\Http\Controllers\EmodelQueryBuilder;
Route::get('emqb',[EmodelQueryBuilder::class,'equery']);
```

Cont. ...

A screenshot of a web browser window displaying a table of user data. The table has columns for Name, Email, Email Verified At, Password, Remember Token, Created At, and Updated At. The data shows one row for a user named 'yitayew' with the email 'yitayew@gmail.com'. The 'Email Verified At' column is empty. The 'Created At' and 'Updated At' columns show the same timestamp: '2024-10-31 19:05:03'.

Name	Email	Email Verified At	Password	Remember Token	Created At	Updated At
yitayew	yitayew@gmail.com		1234	123	2024-10-31 19:05:03	2024-10-31 19:05:03

A screenshot of a web browser window showing a success message: 'Data added!!'.

Summary



Here's a summary of CRUD operations using Laravel's Database Query Builder in table format:

Operation	Description	Example
Create	Insert a single or multiple records into a database table.	<code>DB::table('users')->insert(['name' => 'John Doe', 'email' => 'johndoe@example.com']);</code>
	Insert multiple records.	<code>DB::table('users')->insert([['name' => 'Jane'], ['name' => 'Mark']]));</code>
	Insert and get the ID of the new record.	<code>\$id = DB::table('users')->insertGetId(['name' => 'Alice']);</code>
Read	Retrieve all records from a table.	<code>\$users = DB::table('users')->get();</code>
	Retrieve a single record.	<code>\$user = DB::table('users')->where('id', 1)->first();</code>
	Retrieve specific columns.	<code>\$names = DB::table('users')->pluck('name');</code>
	Filter records using <code>where</code> .	<code>\$users = DB::table('users')->where('status', 'active')->get();</code>

Cont. ...

Update	Update a record.	<code>DB::table('users')->where('id', 1)->update(['name' => 'Updated Name']);</code>
	Increment a numeric column.	<code>DB::table('posts')->where('id', 1)->increment('views');</code>
	Decrement a numeric column.	<code>DB::table('posts')->where('id', 1)->decrement('views', 2);</code>
Delete	Delete a specific record.	<code>DB::table('users')->where('id', 1)->delete();</code>
	Delete all records in a table.	<code>DB::table('users')->delete();</code>
	Truncate the table (delete all records and reset IDs).	<code>DB::table('users')->truncate();</code>

This table highlights CRUD functions commonly used with Laravel's Query Builder, with sample syntax for each operation.

Eloquent Model Query Builder

- Laravel's Eloquent ORM (Object-Relational Mapping) provides a powerful, expressive way to interact with databases by allowing developers to represent database tables as classes and rows as objects.
- With Eloquent, each model represents a table, where class methods translate into SQL commands. For instance, an User model corresponds to a users table, and interacting with this model feels intuitive, using simple syntax like `$users = User::all()` to retrieve all records, rather than needing to write complex SQL.

Cont. ...

- The Eloquent Query Builder takes Eloquent further by allowing more complex queries with chainable methods, making it easy to build and customize database requests without writing SQL.
- Eloquent supports a range of methods like **where()**, **orderBy()**, and **select()**, which can be combined in logical chains to create precise queries.

Cont. ...

- Relationships are another key feature, where methods like `hasMany`, `belongsTo`, and `belongsToMany` allow developers to define and query relationships between tables, reflecting the structure of relational databases directly in code.
- Eloquent also includes built-in support for CRUD operations, with methods like `create()`, `update()`, and `delete()` for simple database interactions.

Creating(Controller, View and Model)

```
> ▼ TERMINAL
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:controller EmodelQueryBuilder

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

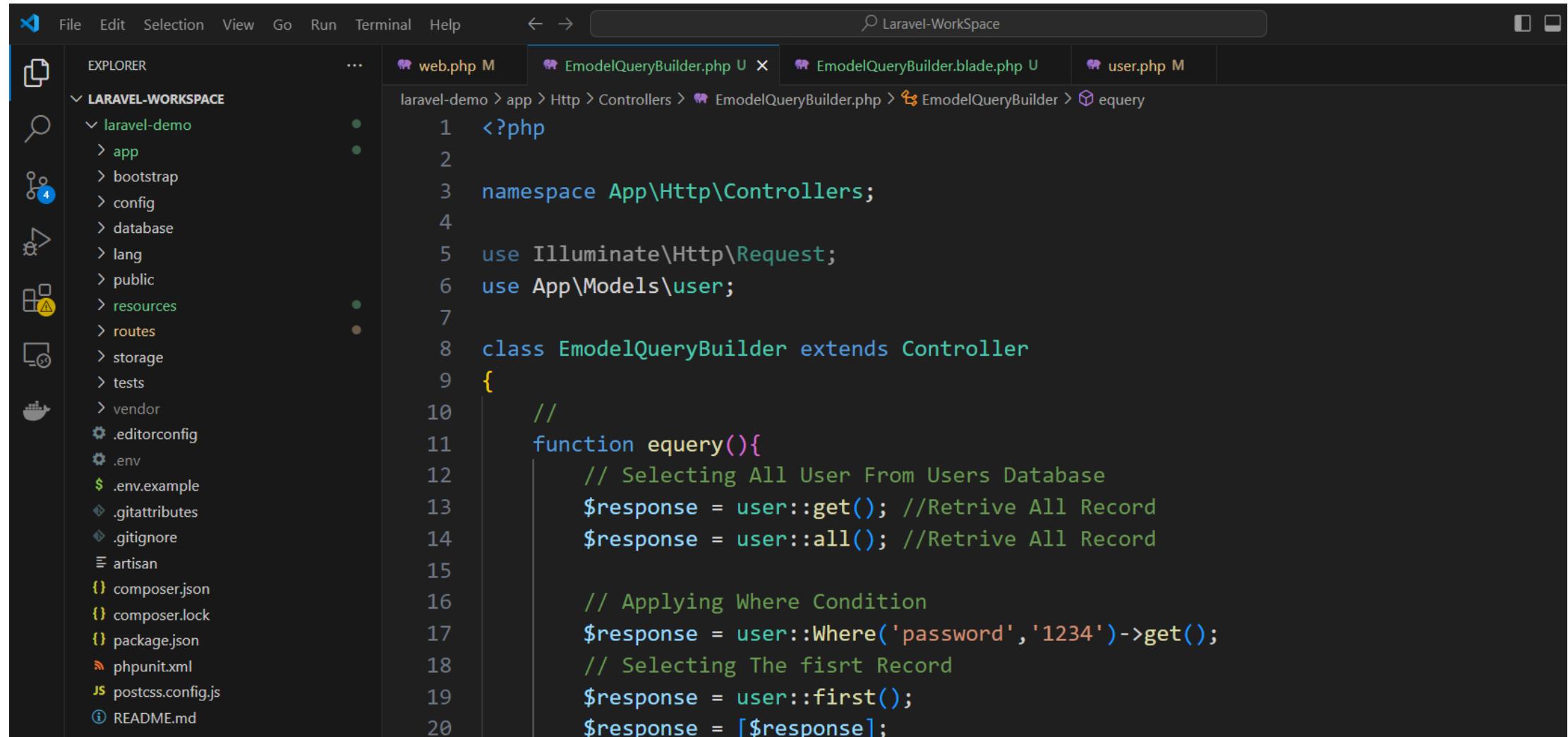
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view EmodelQueryBuilder

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

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:model user

[INFO] Model [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Models\user.php] created successfully.
```

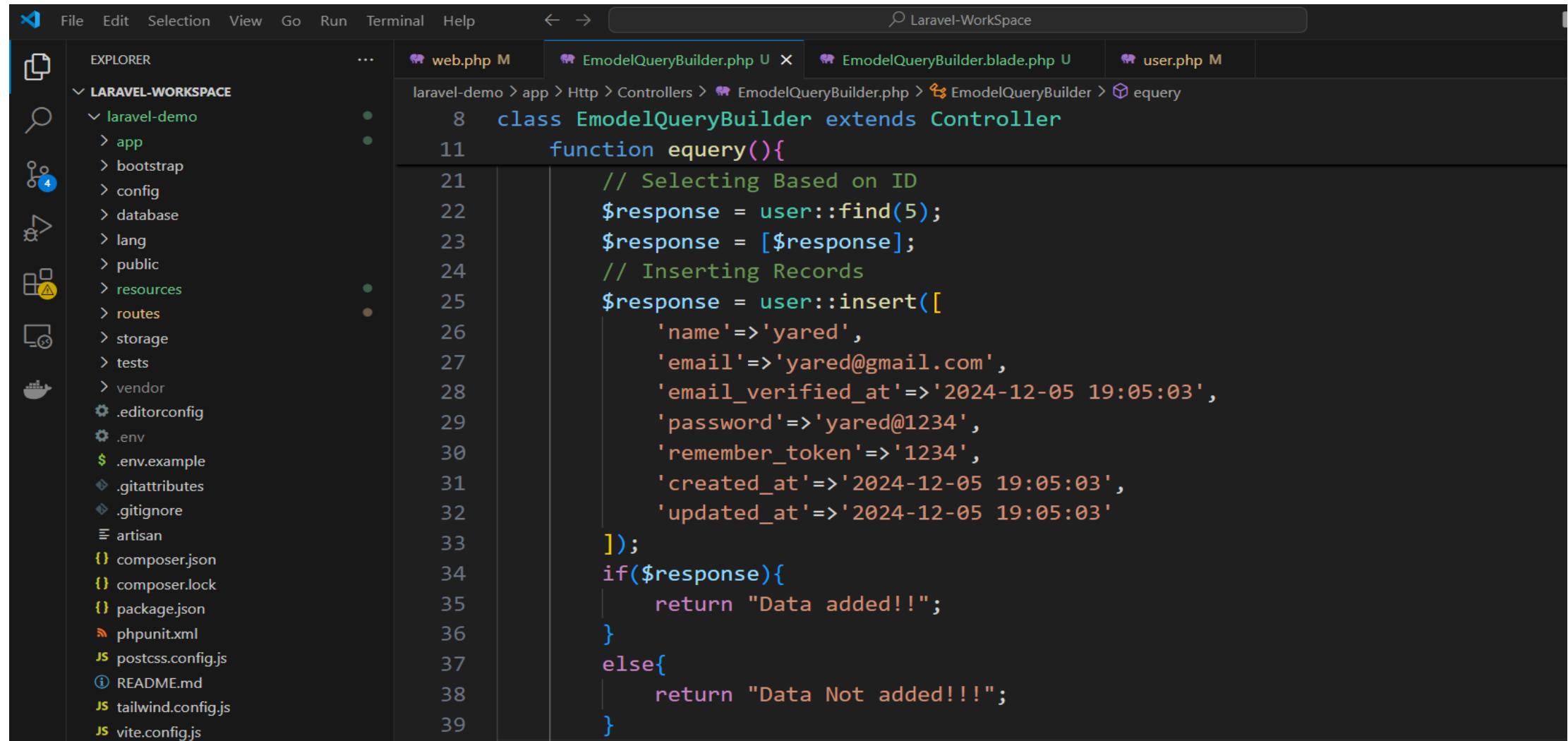
Controller



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure of a Laravel workspace named 'laravel-demo'. The structure includes app, bootstrap, config, database, lang, public, resources, routes, storage, tests, vendor, .editorconfig, .env, .env.example, .gitattributes, .gitignore, artisan, composer.json, composer.lock, package.json, phpunit.xml, postcss.config.js, and README.md. The main editor area displays a PHP file named 'EmodelQueryBuilder.php' located at laravel-demo/app/Http/Controllers. The code implements a custom controller class 'EmodelQueryBuilder' that extends the standard Controller class. It contains a single method 'equery()' which performs database queries using the User model. The code uses Illuminate\Http\Request and App\Models\User.

```
<?php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
use App\Models\User;  
  
class EmodelQueryBuilder extends Controller  
{  
    //  
    function equery(){  
        // Selecting All User From Users Database  
        $response = user::get(); //Retrive All Record  
        $response = user::all(); //Retrive All Record  
  
        // Applying Where Condition  
        $response = user::Where('password','1234')->get();  
        // Selecting The fisrt Record  
        $response = user::first();  
        $response = [$response];  
    }  
}
```

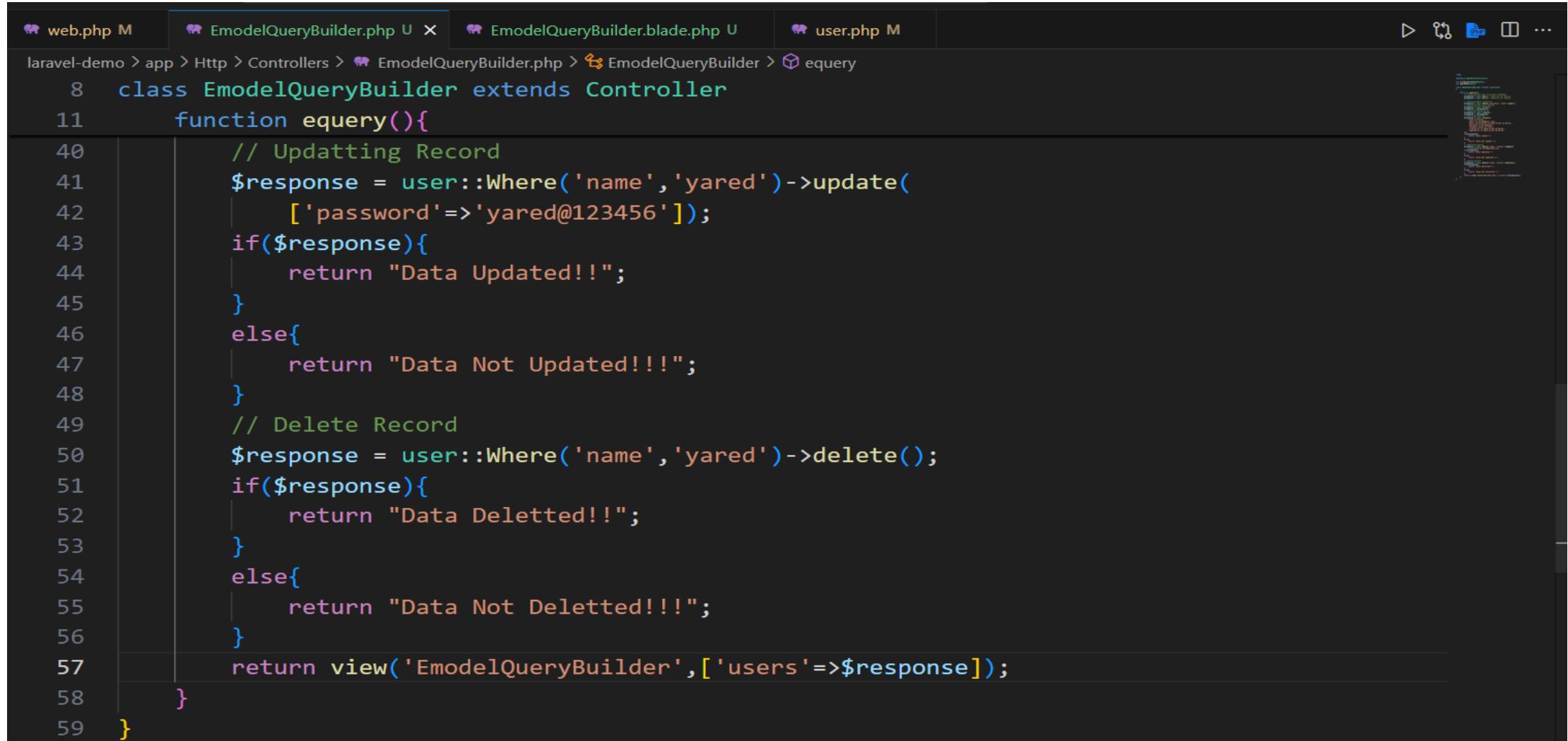
Cont. ...



The screenshot shows a dark-themed interface of the Visual Studio Code code editor. The title bar reads "Laravel-WorkSpace". The left sidebar, titled "EXPLORER", shows the project structure under "LARAVEL-WORKSPACE". The "laravel-demo" folder contains "app", "bootstrap", "config", "database", "lang", "public", "resources", "routes", "storage", "tests", "vendor", ".editorconfig", ".env", ".env.example", ".gitattributes", ".gitignore", "artisan", "composer.json", "composer.lock", "package.json", "phpunit.xml", "postcss.config.js", "README.md", "tailwind.config.js", and "vite.config.js". The main editor area displays the "EmodelQueryBuilder.php" file, which is part of the "Http\Controllers" namespace. The code implements a custom query builder:

```
8 class EmodelQueryBuilder extends Controller
11 function equery(){
21     // Selecting Based on ID
22     $response = user::find(5);
23     $response = [$response];
24     // Inserting Records
25     $response = user::insert([
26         'name'=>'yared',
27         'email'=>'yared@gmail.com',
28         'email_verified_at'=>'2024-12-05 19:05:03',
29         'password'=>'yared@1234',
30         'remember_token'=>'1234',
31         'created_at'=>'2024-12-05 19:05:03',
32         'updated_at'=>'2024-12-05 19:05:03'
33     ]);
34     if($response){
35         return "Data added!!";
36     }
37     else{
38         return "Data Not added!!!";
39     }
}
```

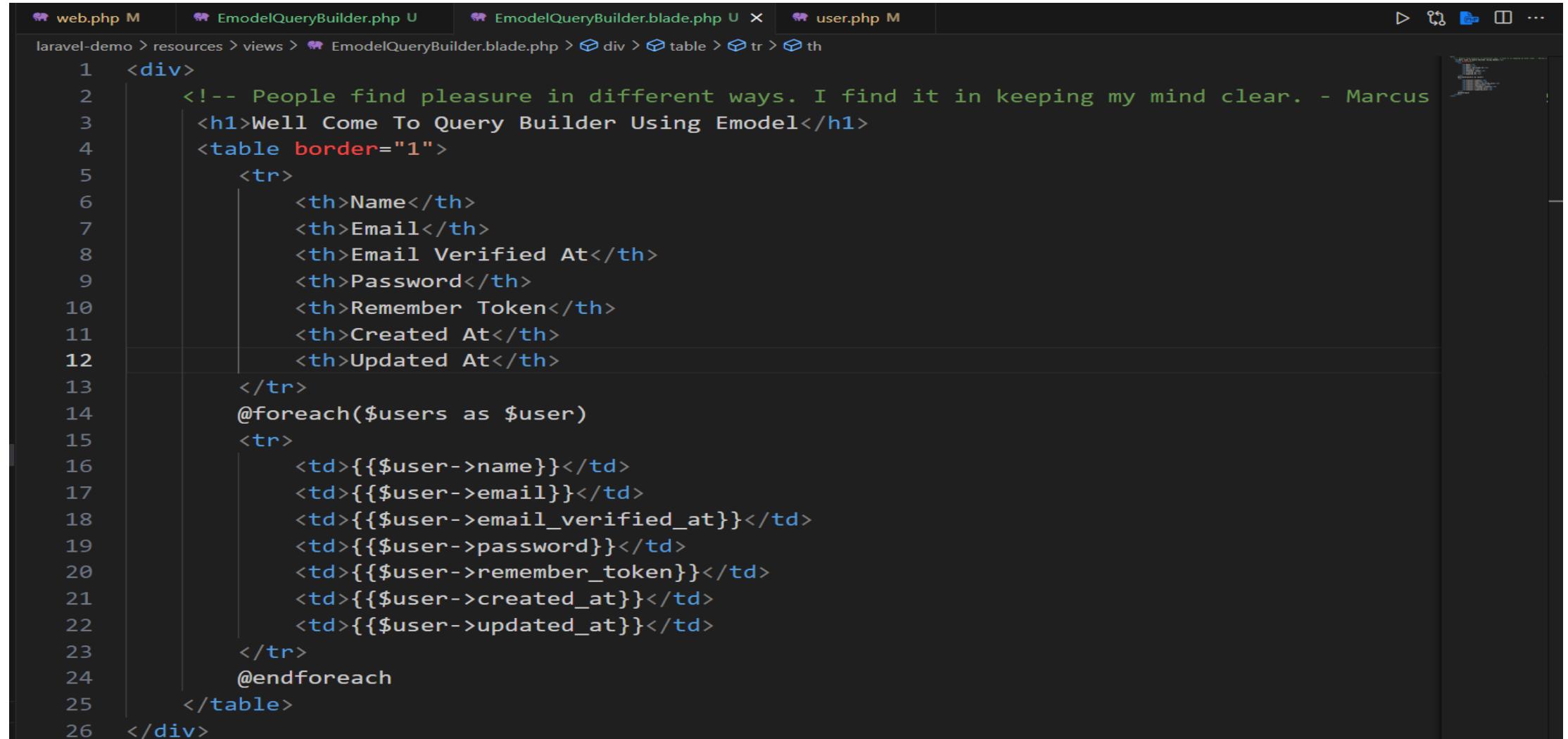
Cont. ...



The screenshot shows a code editor with several tabs at the top: web.php M, EmodelQueryBuilder.php U X, EmodelQueryBuilder.blade.php U, and user.php M. The current file is EmodelQueryBuilder.php. The code is as follows:

```
8 class EmodelQueryBuilder extends Controller
11 function equery(){
40     // Updatting Record
41     $response = user::Where('name', 'yared')->update(
42         ['password'=>'yared@123456']);
43     if($response){
44         return "Data Updated!!";
45     }
46     else{
47         return "Data Not Updated!!!";
48     }
49     // Delete Record
50     $response = user::Where('name', 'yared')->delete();
51     if($response){
52         return "Data Deleted!!";
53     }
54     else{
55         return "Data Not Deleted!!!";
56     }
57     return view('EmodelQueryBuilder', ['users'=>$response]);
58 }
59 }
```

View



The screenshot shows a code editor with a dark theme displaying a Laravel blade template. The file path in the header is laravel-demo > resources > views > EmodelQueryBuilder.blade.php. The code itself is:

```
1 <div>
2     <!-- People find pleasure in different ways. I find it in keeping my mind clear. - Marcus
3     <h1>Well Come To Query Builder Using Emodel</h1>
4     <table border="1">
5         <tr>
6             <th>Name</th>
7             <th>Email</th>
8             <th>Email Verified At</th>
9             <th>Password</th>
10            <th>Remember Token</th>
11            <th>Created At</th>
12            <th>Updated At</th>
13        </tr>
14        @foreach($users as $user)
15            <tr>
16                <td>{{$user->name}}</td>
17                <td>{{$user->email}}</td>
18                <td>{{$user->email_verified_at}}</td>
19                <td>{{$user->password}}</td>
20                <td>{{$user->remember_token}}</td>
21                <td>{{$user->created_at}}</td>
22                <td>{{$user->updated_at}}</td>
23            </tr>
24        @endforeach
25    </table>
26 </div>
```

Model

web.php M EmodelQueryBuilder.php U EmodelQueryBuilder.blade.php U user.php M X

laravel-demo > app > Models > user.php > ...

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class user extends Model
8 {
9     //
10    public $timestamps=false;
11 }
```

Route

The screenshot shows a Laravel development environment in a code editor. The left sidebar displays the project structure under 'Laravel-Workspace' with files like 'middleware1.blade.php', 'middleware2.blade.php', etc., in the 'resources/views' directory, and 'console.php' and 'web.php' in the 'routes' directory. The main editor window shows the 'web.php' file with the following code:

```
119 // Elequent Model Query Builder
120 use App\Http\Controllers\EmodelQueryBuilder;
121 Route::get('emqb',[EmodelQueryBuilder::class,'equery']);
```

The status bar at the bottom indicates the URL `127.0.0.1:8000/emqb`. A modal dialog box is open in the foreground displaying the message **Data Deleted!!**.

Summary

Here's a table summarizing CRUD operations using Laravel's Eloquent Model and Query Builder methods:

Operation	Eloquent Model	Query Builder	Description
Create	<pre>\$user = new User(); \$user->name = 'John'; \$user->save(); or User::create(['name' => 'John']);</pre>	<pre>DB::table('users')->insert(['name' => 'John']);</pre>	Inserts new records into the database.
Read	<pre>User::all(); User::find(1); User::where('status', 'active')->get();</pre>	<pre>DB::table('users')->get(); DB::table('users')->where('status', 'active')->first();</pre>	Retrieves records from the database.

Cont. ...

Update	<pre>\$user = User::find(1); \$user->name = 'Jane'; \$user->save(); or User::where('id', 1)->update(['name' => 'Jane']);</pre>	<pre>DB::table('users')->where('id', 1)->update(['name' => 'Jane']);</pre>	Updates existing records in the database.
Delete	<pre>\$user = User::find(1); \$user->delete(); or User::where('id', 1)->delete();</pre>	<pre>DB::table('users')->where('id', 1)->delete();</pre>	Deletes records from the database.

This table shows how to perform basic CRUD operations with both Eloquent Models and Query Builder methods in Laravel. Eloquent provides an object-oriented approach, while Query Builder is a more flexible option for more complex queries.

Eloquent ORM vs Query Builder in Laravel

- Laravel provides **two primary** ways to interact with the database: **Eloquent ORM** and **Query Builder**. While both serve the purpose of querying and manipulating data, they differ in their **approach, features, and use cases**.

Eloquent ORM

Eloquent ORM

1. Definition:

Eloquent is Laravel's **Object-Relational Mapping (ORM)**, which provides an object-oriented and model-centric approach to interact with the database. It maps database tables to PHP classes and rows to objects, enabling developers to use models for querying and manipulating data.

2. Features:

- **Model-Based:** Each database table corresponds to a model.
- **Active Record Implementation:** Data is manipulated through model instances.
- **Relationships:** Supports defining and working with database relationships like one-to-one, one-to-many, and many-to-many.
- **Eloquent Methods:** Provides methods like `find()`, `save()`, and `delete()`.

Example

3. Example:

Fetching all users:

```
php
```

 Copy code

```
$users = User::all();
```

Fetching a user by ID:

```
php
```

 Copy code

```
$user = User::find(1);
```

4. Best For:

- Applications with complex relationships.
- When leveraging Laravel's built-in ORM features like relationships, scopes, and mutators.

QueryBuilder

QueryBuilder

1. Definition:

QueryBuilder is a more **direct and procedural approach** to interact with the database. It allows developers to build database queries programmatically without relying on models or object-oriented design.

2. Features:

- **Fluent API:** Methods are chainable and readable.
- **No Models Required:** Operates directly on database tables.
- **Highly Flexible:** Can be used for custom and complex SQL queries.
- **Parameter Binding:** Helps prevent SQL injection.

Example

3. Example:

Fetching all users:

```
php
```

 Copy code

```
$users = DB::table('users')->get();
```

Fetching a user by ID:

```
php
```

 Copy code

```
$user = DB::table('users')->where('id', 1)->first();
```

4. Best For:

- Lightweight operations or raw SQL-like queries.
- When no model is required, e.g., for custom tables or views.

Difference

Key Differences

Feature	Eloquent ORM	Query Builder
Approach	Object-Oriented (Model-centric)	Procedural (Query-centric)
Ease of Use	Easier for complex relationships	Flexible for custom SQL queries
Performance	May have more overhead due to models	Generally faster for simple queries
Relationships	Built-in support (e.g., <code>hasOne</code> , <code>belongsTo</code>)	Manual implementation required
Use Case	CRUD operations with defined models	Custom queries or lightweight tasks
Code Readability	Cleaner and more descriptive	Explicit but less structured

When to use

When to Use

- **Eloquent ORM:**

Use when working with structured data models and their relationships. Ideal for most Laravel applications.

- **Query Builder:**

Use for raw queries, dynamic SQL, or when working with tables that don't have an associated model.

Both Eloquent ORM and Query Builder are powerful, and the choice depends on the specific needs of your application. Often, developers combine both for optimal results.



Route Methods in Laravel

In Laravel, route methods are used to define how your application responds to various HTTP requests. Here's a breakdown of common route methods, with descriptions and examples:

Route Method	Description	Example
GET	Retrieves data without making changes, typically used for displaying views or data.	<code>Route::get('/users', [UserController::class, 'index']);</code>
POST	Used to submit data to the server, often for creating new resources.	<code>Route::post('/users', [UserController::class, 'store']);</code>
PUT	Replaces an entire resource, typically used for updating existing records.	<code>Route::put('/users/{id}', [UserController::class, 'update']);</code>
PATCH	Partially updates a resource, often for modifying specific fields of an existing record.	<code>Route::patch('/users/{id}', [UserController::class, 'updatePartial']);</code>

Cont. ...

OPTIONS	Describes communication options for the target resource, commonly used for CORS preflight requests in APIs.	<pre>Route::options('/users', function () { return response()->json(['options' => 'GET, POST']);});</pre>
ANY	Responds to any HTTP method for the specified route.	<pre>Route::any('/users', function () { /* logic */});</pre>
MATCH	Allows you to specify an array of HTTP methods to respond to on a single route.	<pre>Route::match(['get', 'post'], '/users', [UserController::class, 'handleRequest']);</pre>

Examples of Route Methods in a Controller Context

1. GET - Display all users:

```
php
```

 Copy code

```
Route::get('/users', [UserController::class, 'index']);
```

2. POST - Add a new user:

```
php
```

 Copy code

```
Route::post('/users', [UserController::class, 'store']);
```

3. PUT - Update an existing user:

```
php
```

 Copy code

```
Route::put('/users/{id}', [UserController::class, 'update']);
```

Cont. ...

4. PATCH - Partially update a user's info:

php

 Copy code

```
Route::patch('/users/{id}', [UserController::class, 'updatePartial']);
```

5. DELETE - Delete a user:

php

 Copy code

```
Route::delete('/users/{id}', [UserController::class, 'destroy']);
```

6. ANY - Accept any HTTP request type:

php

 Copy code

```
Route::any('/users', [UserController::class, 'handleAny']);
```

Examples

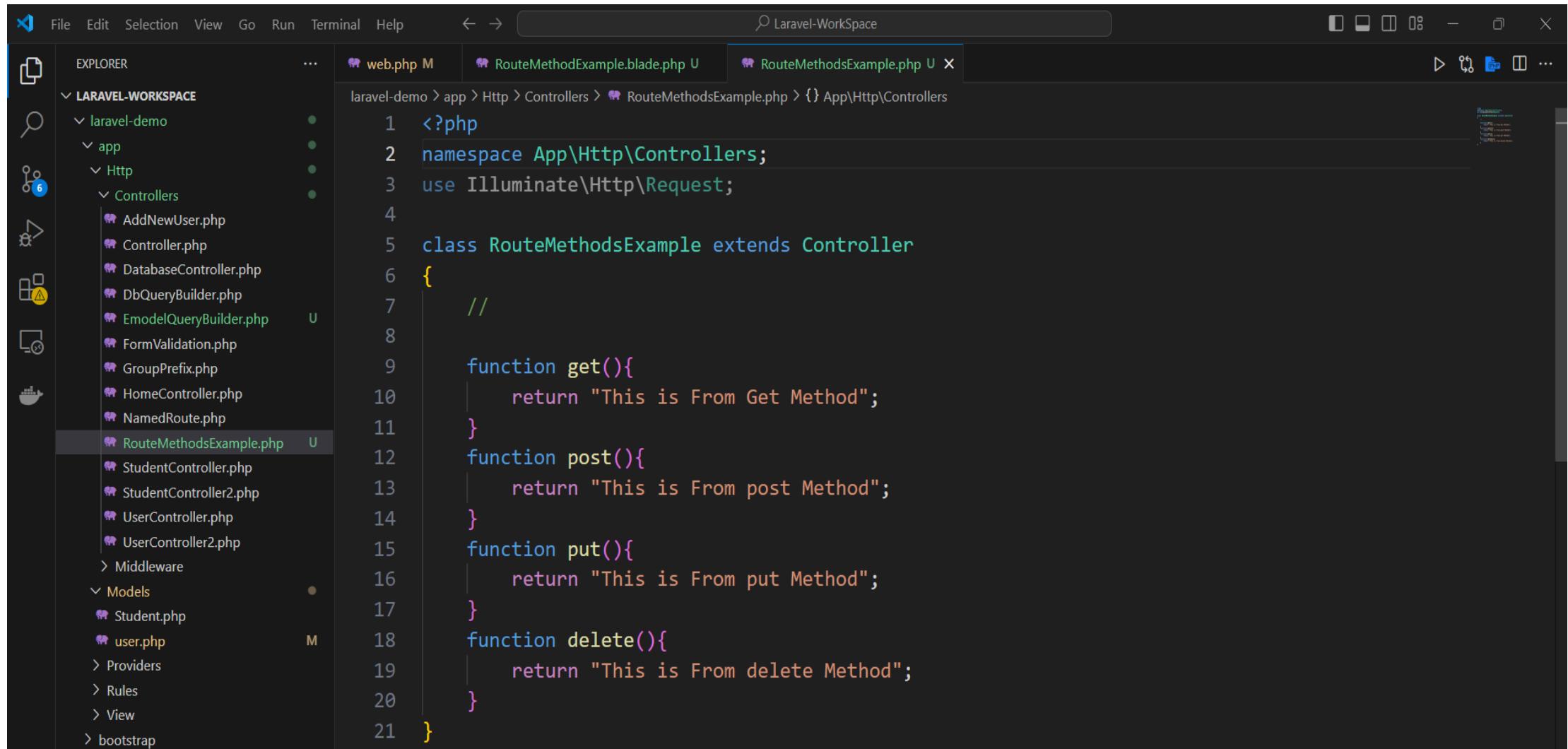
▼ TERMINAL

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

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

```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
○ $ php artisan make:controller RouteMethodsExample
```

Controller (RouteMethodExample)



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The 'app' directory contains 'Http' and 'Controllers' sub-directories. Inside 'Controllers', there are several files: AddNewUser.php, Controller.php, DatabaseController.php, DbQueryBuilder.php, EmodelQueryBuilder.php, FormValidation.php, GroupPrefix.php, HomeController.php, NamedRoute.php, RouteMethodsExample.php (which is currently selected), StudentController.php, StudentController2.php, UserController.php, and UserController2.php. The 'Http' directory also contains 'Middleware', 'Models', 'Providers', 'Rules', 'View', and 'bootstrap'. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled 'Laravel-WorkSpace'. The main editor area displays the contents of the 'RouteMethodsExample.php' file:

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4
5 class RouteMethodsExample extends Controller
6 {
7     //
8
9     function get(){
10        return "This is From Get Method";
11    }
12    function post(){
13        return "This is From post Method";
14    }
15    function put(){
16        return "This is From put Method";
17    }
18    function delete(){
19        return "This is From delete Method";
20    }
21 }
```

View (RouteMethodExample)

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The 'resources/views' folder contains several blade files, including 'RouteMethodExample.blade.php' (which is currently selected), 'DBqueryView.blade.php', 'Demo.blade.php', 'EmodelQueryBuilder.blade.php', 'formvalidation.blade.php', 'grouprefix.blade.php', 'home.blade.php', 'home2.blade.php', 'middleware.blade.php', 'middleware2.blade.php', 'middleware3.blade.php', 'middleware4.blade.php', 'middleware5.blade.php', 'namedroute.blade.php', 'RouteMethodExample.blade.php', 'Students.blade.php', 'url1.blade.php', 'url2.blade.php', 'userform.blade.php', 'userform2.blade.php', and 'welcome.blade.php'. The 'routes' folder contains 'console.php' and 'web.php'. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The title bar says 'Laravel-WorkSpace'. The main editor area displays the contents of 'RouteMethodExample.blade.php'.

```
<div>
    <!-- It is quality rather than quantity that matters. - Lucius Annaeus Seneca -->
    <h1>Well Come To User Form</h1>
    <form action="methodsexample" method="post">
        <!-- <input type="hidden" name='_method' value="PUT"> -->
        <input type="hidden" name='_method' value="DELETE">

        @csrf
        <label for="name">User Name:</label>
        <input type="text" name='user' placeholder="enter user">
        <br>
        <br>
        <label for="password">Password:</label>
        <input type="password" name='password' placeholder="enter password">
        <br>
        <br>
        <button>submit</button>
        <br>
        <br>
    </form>
</div>
```

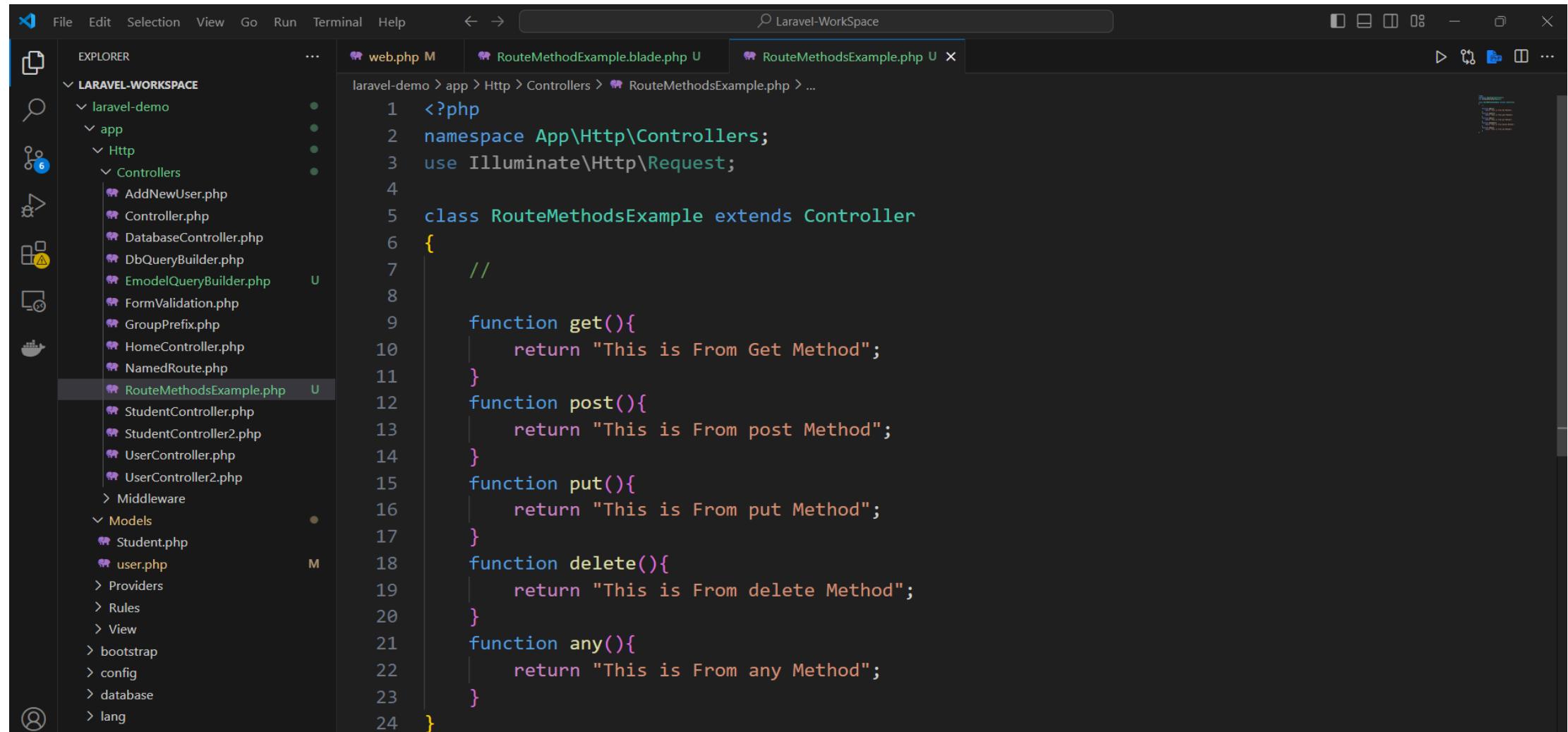
Route

The screenshot shows a Visual Studio Code (VS Code) interface with an orange border. The title bar says "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems, and others. The Explorer view shows a file tree under "LARAVEL-WORKSPACE" with "laravel-demo" expanded, showing "resources" and "views" folders containing various blade files like DBqueryView.blade.php, Demo.blade.php, etc. Below "views" is a "routes" folder containing "console.php" and "web.php". The main editor area shows "web.php" with code:

```
123 // Route Methods Examples
124 use App\Http\Controllers\RouteMethodsExample;
125 Route::get('methodsexample',[RouteMethodsExample::class,'get']);
126 Route::post('methodsexample',[RouteMethodsExample::class,'post']);
127 Route::put('methodsexample',[RouteMethodsExample::class,'put']);
128 Route::delete('methodsexample',[RouteMethodsExample::class,'delete']);
129
130 // Rout For RouteMethodExample blade
131 Route::view('form','RouteMethodExample');
132
133
134
135
136
137
138
139
140
141
```

The status bar at the bottom shows "12/17/2024" and "40". A browser preview window is open, showing the URL "127.0.0.1:8000/methodsexample" and the text "This is From delete Method".

Cont. ... any Method



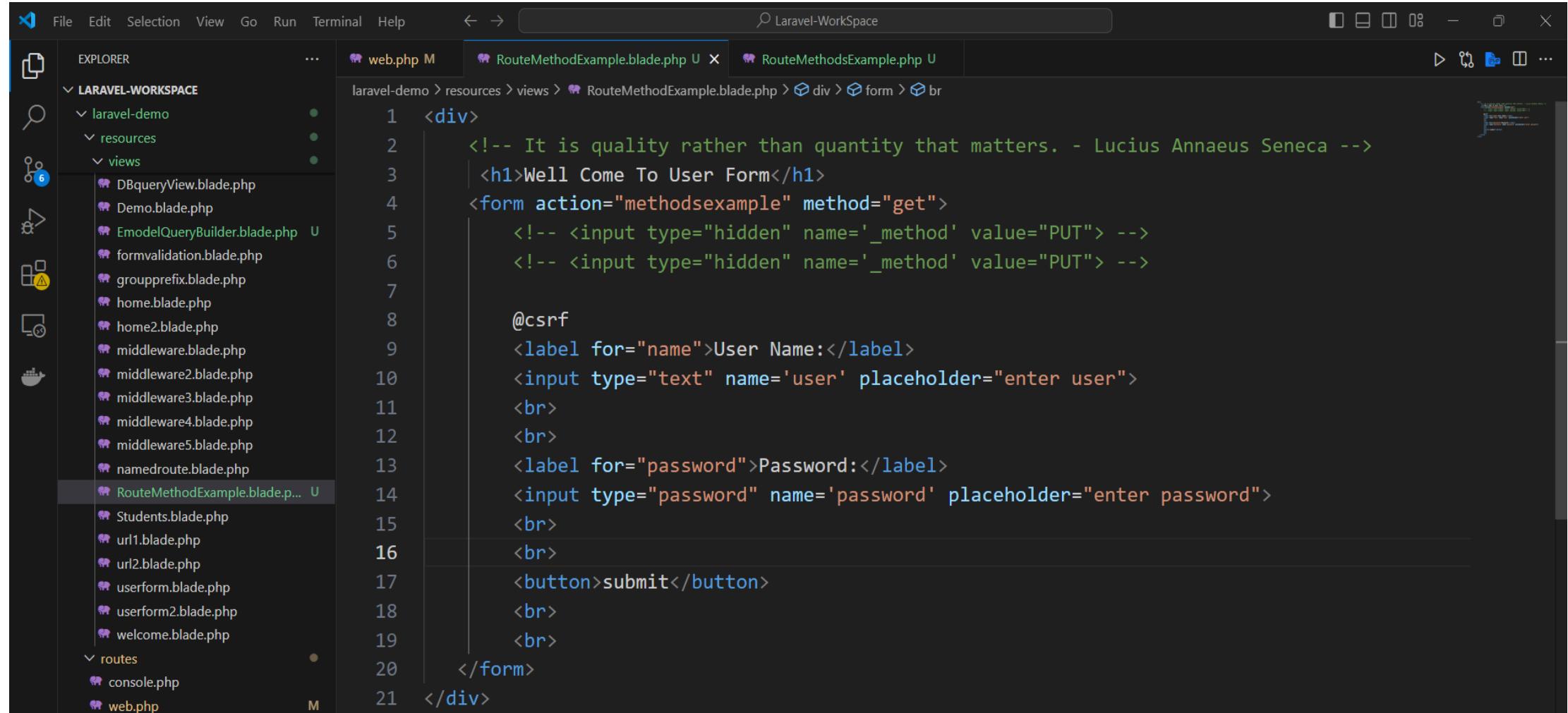
The screenshot shows the Visual Studio Code interface with the title bar "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The "app/Http/Controllers" folder contains several files, and "RouteMethodsExample.php" is currently selected and highlighted in the list. The main editor area displays the code for "RouteMethodsExample.php". The code defines a class "RouteMethodsExample" that extends "Controller". It includes methods for GET, POST, PUT, DELETE, and ANY requests, each returning a specific string response.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class RouteMethodsExample extends Controller
{
    //

    function get(){
        return "This is From Get Method";
    }
    function post(){
        return "This is From post Method";
    }
    function put(){
        return "This is From put Method";
    }
    function delete(){
        return "This is From delete Method";
    }
    function any(){
        return "This is From any Method";
    }
}
```

Cont. ...



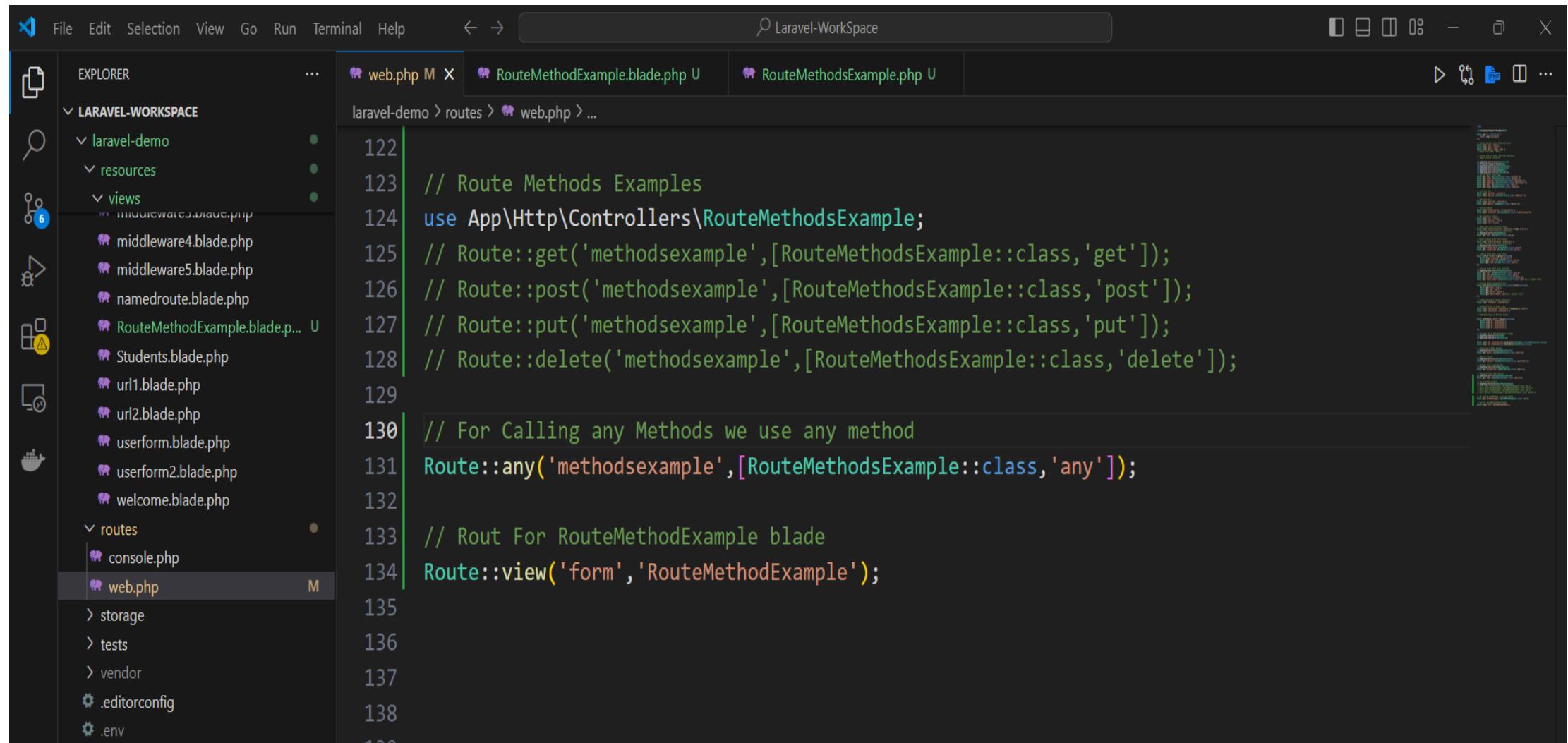
The screenshot shows a code editor interface with the title bar "Laravel-WorkSpace". The left sidebar is titled "EXPLORER" and shows a tree view of a "LARAVEL-WORKSPACE" folder named "laravel-demo". Inside "laravel-demo", there are "resources" and "views" folders. The "views" folder contains several Blade template files: "DbqueryView.blade.php", "Demo.blade.php", "EmodelQueryBuilder.blade.php", "formvalidation.blade.php", "groupprefix.blade.php", "home.blade.php", "home2.blade.php", "middleware.blade.php", "middleware2.blade.php", "middleware3.blade.php", "middleware4.blade.php", "middleware5.blade.php", "namedroute.blade.php", "RouteMethodExample.blade.php", "Students.blade.php", "url1.blade.php", "url2.blade.php", "userform.blade.php", "userform2.blade.php", and "welcome.blade.php". Below "views" is a "routes" folder containing "console.php" and "web.php". The "EXPLORER" sidebar also includes icons for file search, file history, and file status.

The main editor area displays the content of the "RouteMethodExample.blade.php" file. The code is as follows:

```
<div>
    <!-- It is quality rather than quantity that matters. - Lucius Annaeus Seneca -->
    <h1>Well Come To User Form</h1>
    <form action="methodsexample" method="get">
        <!-- <input type="hidden" name='_method' value="PUT"> -->
        <!-- <input type="hidden" name='_method' value="PUT"> -->

        @csrf
        <label for="name">User Name:</label>
        <input type="text" name='user' placeholder="enter user">
        <br>
        <br>
        <label for="password">Password:</label>
        <input type="password" name='password' placeholder="enter password">
        <br>
        <br>
        <br>
        <button>submit</button>
        <br>
        <br>
        <br>
    </form>
</div>
```

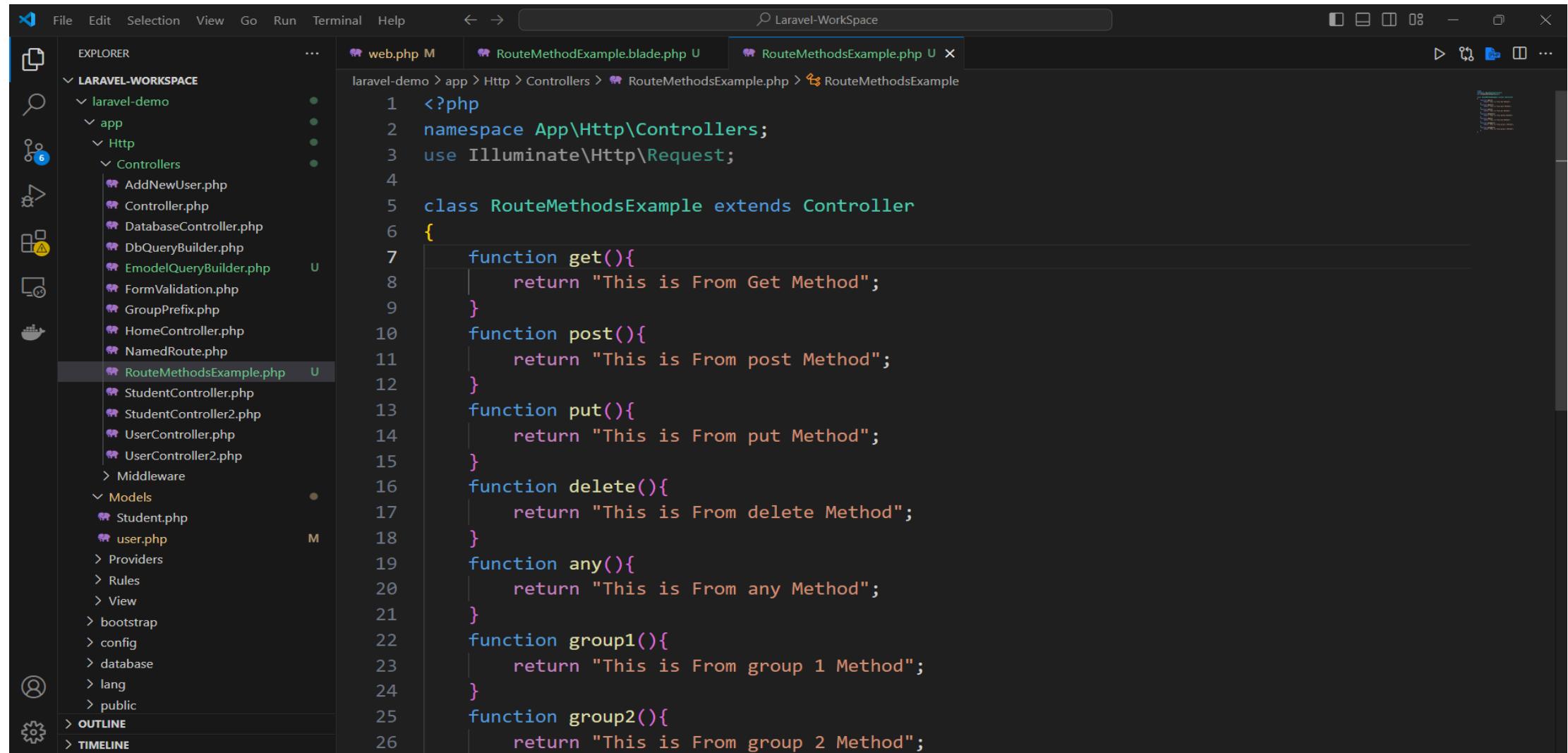
Cont. ...



The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. At the top, a navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A search bar is positioned above the main workspace area, which is titled "Laravel-WorkSpace". The left side features a sidebar with various icons: a file icon, a magnifying glass, a gear with a "6" (indicating six notifications), a double arrow, a square with a circle, a monitor with a gear, and a ship. Below these are sections for "EXPLORER", "LARAVEL-WORKSPACE", and "TERMINAL". The "LARAVEL-WORKSPACE" section displays a tree view of a Laravel project structure under "laravel-demo": resources (views, middleware, blade files like middleware4.blade.php, middleware5.blade.php, namedroute.blade.php, RouteMethodExample.blade.php, Students.blade.php, url1.blade.php, url2.blade.php, userform.blade.php, userform2.blade.php, welcome.blade.php), routes (console.php, web.php), storage, tests, vendor, .editorconfig, and .env. The "web.php" file is currently selected and highlighted in the tree view. The main workspace shows the content of "web.php":

```
122 // Route Methods Examples
123 use App\Http\Controllers\RouteMethodsExample;
124 // Route::get('methodsexample',[RouteMethodsExample::class,'get']);
125 // Route::post('methodsexample',[RouteMethodsExample::class,'post']);
126 // Route::put('methodsexample',[RouteMethodsExample::class,'put']);
127 // Route::delete('methodsexample',[RouteMethodsExample::class,'delete']);
128
129
130 // For Calling any Methods we use any method
131 Route::any('methodsexample',[RouteMethodsExample::class,'any']);
132
133 // Rout For RouteMethodExample blade
134 Route::view('form','RouteMethodExample');
```

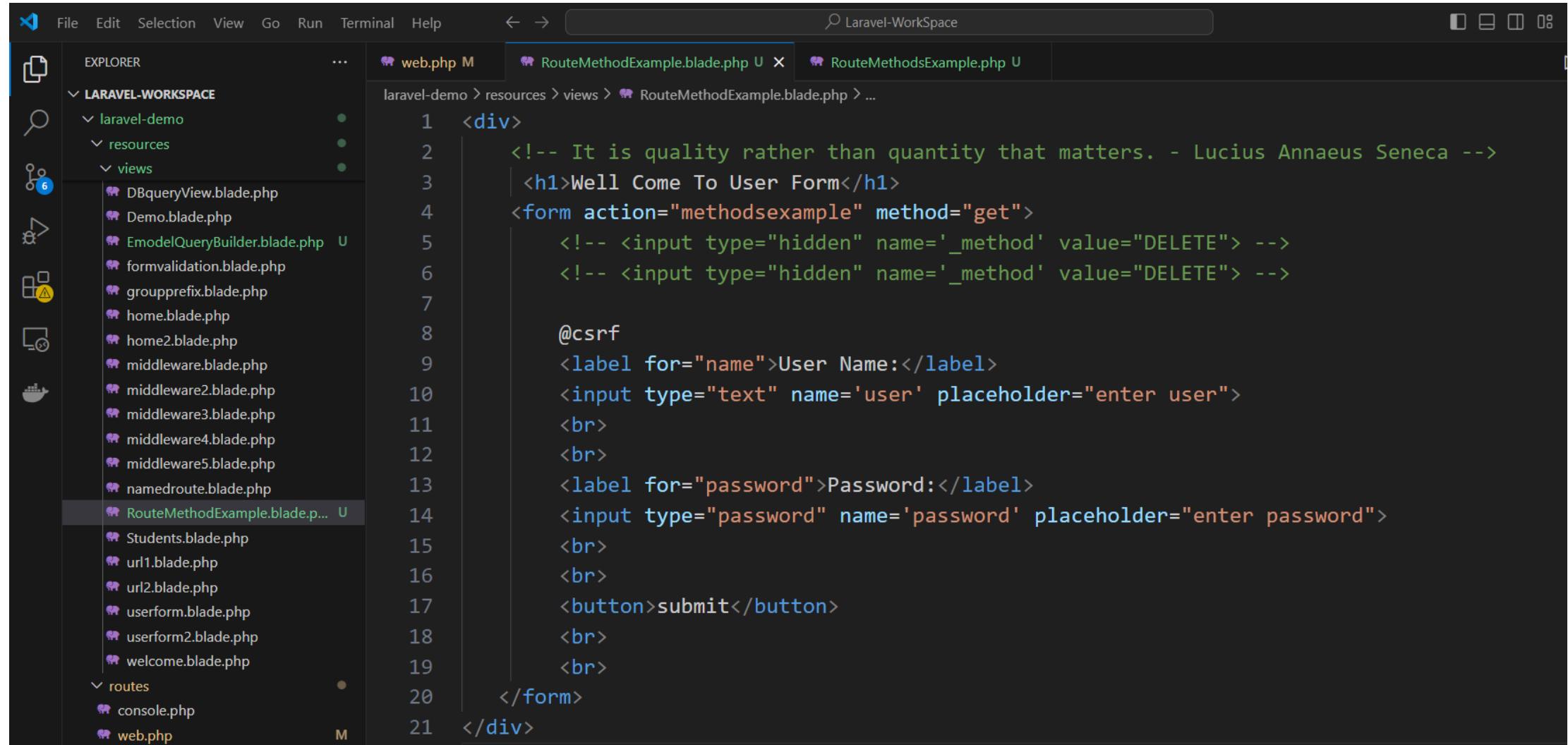
Cont. ... match method



The screenshot shows a dark-themed interface of Visual Studio Code (VS Code) with an orange border around the main content area. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar displays "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems, Editor, and others. The Explorer panel shows a tree view of a Laravel workspace named "laravel-demo". Under "app/Http/Controllers", the file "RouteMethodsExample.php" is selected and highlighted in blue. The code editor shows the following PHP code:

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
  
class RouteMethodsExample extends Controller  
{  
    function get()  
    {  
        return "This is From Get Method";  
    }  
    function post()  
    {  
        return "This is From post Method";  
    }  
    function put()  
    {  
        return "This is From put Method";  
    }  
    function delete()  
    {  
        return "This is From delete Method";  
    }  
    function any()  
    {  
        return "This is From any Method";  
    }  
    function group1()  
    {  
        return "This is From group 1 Method";  
    }  
    function group2()  
    {  
        return "This is From group 2 Method";  
    }  
}
```

Cont. ...



The screenshot shows the Visual Studio Code interface with the title bar "Laravel-WorkSpace". The left sidebar displays the file structure of a Laravel project named "laravel-demo". The "EXPLORER" view shows various files under "LARAVEL-WORKSPACE" such as DBqueryView.blade.php, Demo.blade.php, EmodelQueryBuilder.blade.php, formvalidation.blade.php, groupprefix.blade.php, home.blade.php, home2.blade.php, middleware.blade.php, middleware2.blade.php, middleware3.blade.php, middleware4.blade.php, middleware5.blade.php, namedroute.blade.php, RouteMethodExample.blade.php, Students.blade.php, url1.blade.php, url2.blade.php, userform.blade.php, userform2.blade.php, and welcome.blade.php. Below these are files in the "routes" folder: console.php and web.php. The "ROUTE" tab in the Explorer sidebar has a count of 6. The main editor area shows the content of "RouteMethodExample.blade.php". The code is as follows:

```
1 <div>
2   <!-- It is quality rather than quantity that matters. - Lucius Annaeus Seneca -->
3   <h1>Well Come To User Form</h1>
4   <form action="methodsexample" method="get">
5     <!-- <input type="hidden" name='_method' value="DELETE"> -->
6     <!-- <input type="hidden" name='_method' value="DELETE"> -->
7
8     @csrf
9     <label for="name">User Name:</label>
10    <input type="text" name='user' placeholder="enter user">
11    <br>
12    <br>
13    <label for="password">Password:</label>
14    <input type="password" name='password' placeholder="enter password">
15    <br>
16    <br>
17    <button>submit</button>
18    <br>
19    <br>
20  </form>
21 </div>
```

Cont. ...

The screenshot shows a Visual Studio Code interface with the title bar "Laravel-WorkSpace". The left sidebar is the Explorer view, showing the project structure under "LARAVEL-WORKSPACE". The main editor area displays a PHP file named "web.php" containing route definitions. The browser preview window at the bottom shows the output of a route call, indicating successful execution of a method from a controller.

```
122 // Route Methods Examples
123 use App\Http\Controllers\RouteMethodsExample;
124 // Route::get('methodsexample',[RouteMethodsExample::class,'get']);
125 // Route::post('methodsexample',[RouteMethodsExample::class,'post']);
126 // Route::put('methodsexample',[RouteMethodsExample::class,'put']);
127 // Route::delete('methodsexample',[RouteMethodsExample::class,'delete']);
128
129
130 // For Calling any Methods we use any method
131 // Route::any('methodsexample',[RouteMethodsExample::class,'any']);
132
133 // For Calling match method we use match methods
134 Route::match(['get','post'],'methodsexample',[RouteMethodsExample::class,'group1']);
135 Route::match(['put','delete'],'methodsexample',[RouteMethodsExample::class,'group2']);
136
137 // Rout For RouteMethodExample blade
138 Route::view('form','RouteMethodExample');
139
140
141
142
```

127.0.0.1:8000/methodsexample?_token=Z09iqSsBfipwEgDH0q63RHl5OCHdjAt98AhfzlCr&user=&password=

This is From group 1 Method

HTTP Request Class

- The HTTP Request class in Laravel is a powerful component used to interact with **incoming HTTP requests**. It provides methods to retrieve input data, query parameters, cookies, files, and more.
- This class is automatically injected into controller methods and can be used to access and validate request data conveniently.

Basic Usage of the Request Class

To use the `Request` class, you need to import it in your controller or route:

php

```
use Illuminate\Http\Request;
```

 Copy code

Example

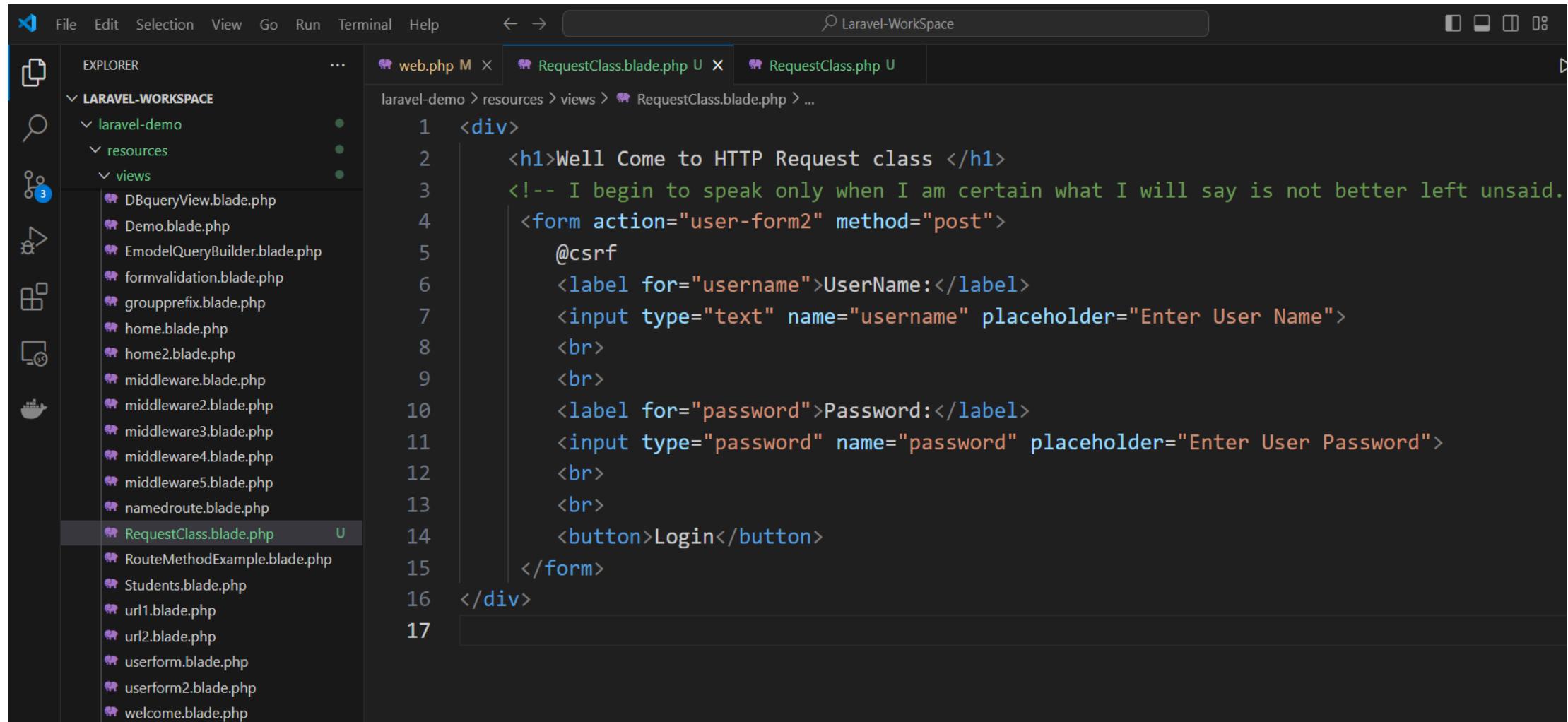
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
> ✓ TERMINAL
❯ $ php artisan make:controller RequestClass
INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\RequestClass.php] created successfully.

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
❯ $ php artisan make:view RequestClass
INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\RequestClass.blade.php] created successfully.

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
❯ $ 
```

In 2 Col 22 Spaces: 4 UTF-8 LF

View (RequestClass)

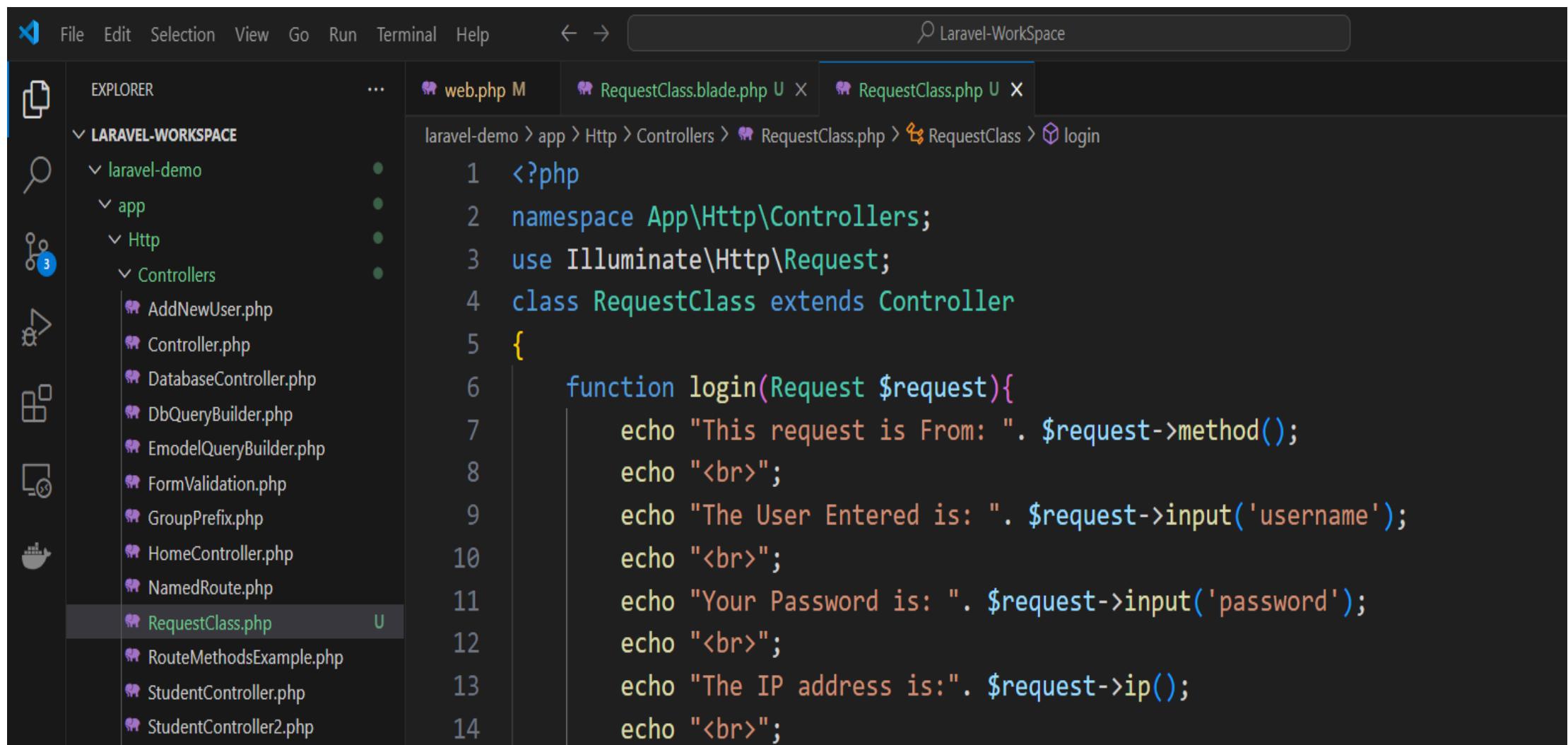


The screenshot shows a dark-themed interface of Visual Studio Code (VS Code) with an orange header bar. The title bar says "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing a tree structure of a Laravel workspace named "laravel-demo". Under "resources/views", there are many blade files like DBqueryView.blade.php, Demo.blade.php, etc., and one file named "RequestClass.blade.php" which is currently selected and has a green status bar icon indicating it's been modified.

The main editor area displays the code for "RequestClass.blade.php". The code is as follows:

```
1 <div>
2   <h1>Well Come to HTTP Request class </h1>
3   <!-- I begin to speak only when I am certain what I will say is not better left unsaid. -->
4   <form action="user-form2" method="post">
5     @csrf
6     <label for="username">UserName:</label>
7     <input type="text" name="username" placeholder="Enter User Name">
8     <br>
9     <br>
10    <label for="password">Password:</label>
11    <input type="password" name="password" placeholder="Enter User Password">
12    <br>
13    <br>
14    <button>Login</button>
15  </form>
16 </div>
17 
```

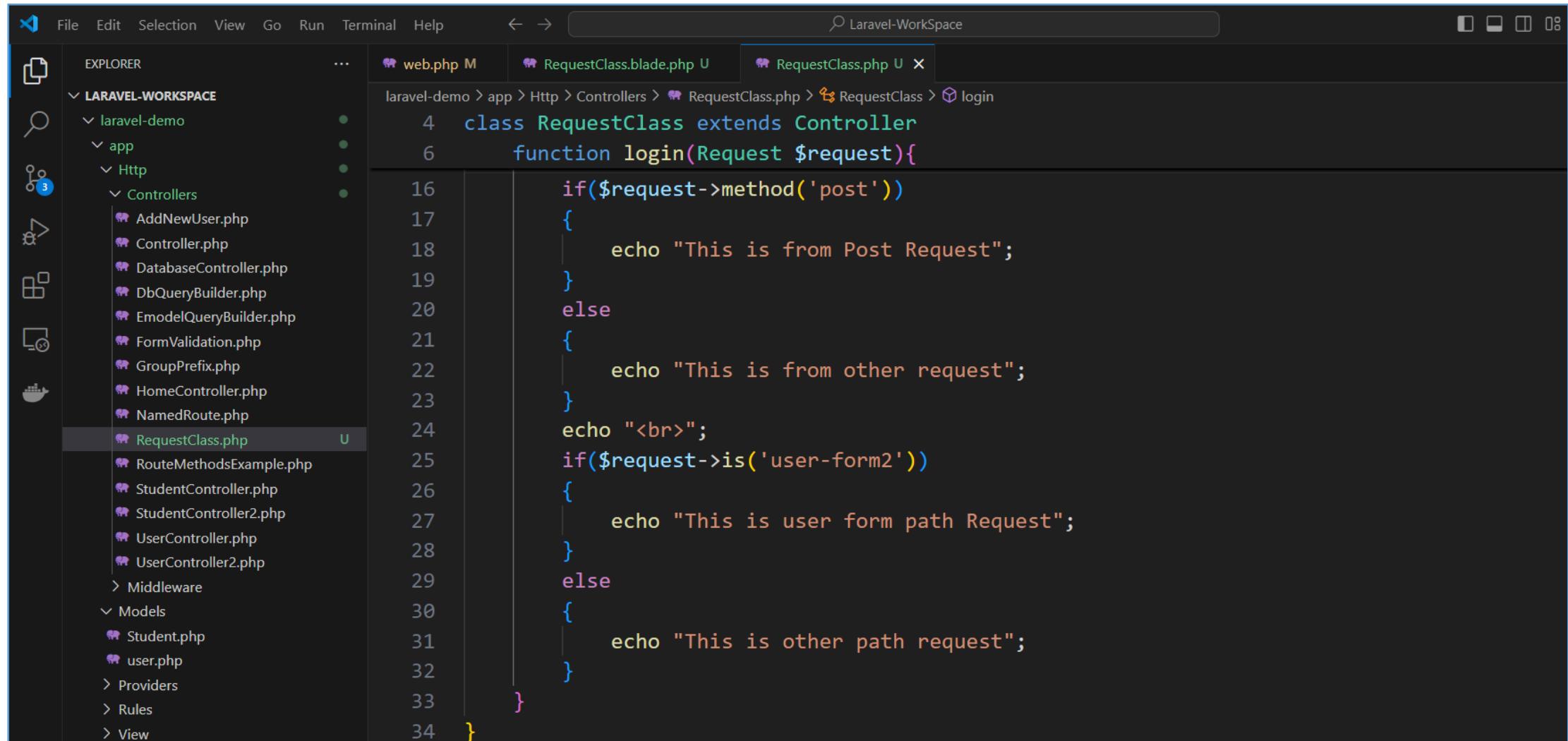
Controller (RequestClass)



The screenshot shows a dark-themed interface of Visual Studio Code. In the top bar, there are icons for File, Edit, Selection, View, Go, Run, Terminal, and Help. To the right of the menu is a search bar containing "Laravel-WorkSpace". Below the menu is the Explorer sidebar, which displays a tree structure of a Laravel workspace named "laravel-demo". Under "app/Http/Controllers", several files are listed: AddNewUser.php, Controller.php, DatabaseController.php, DbQueryBuilder.php, EmodelQueryBuilder.php, FormValidation.php, GroupPrefix.php, HomeController.php, NamedRoute.php, RequestClass.php (which is currently selected and has a green outline), RouteMethodsExample.php, StudentController.php, and StudentController2.php. The status bar at the bottom shows "12/17/2024" on the left and "Laravel Framework" on the right. The main code editor area contains the following PHP code:

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 class RequestClass extends Controller
5 {
6     function login(Request $request){
7         echo "This request is From: ". $request->method();
8         echo "<br>";
9         echo "The User Entered is: ". $request->input('username');
10        echo "<br>";
11        echo "Your Password is: ". $request->input('password');
12        echo "<br>";
13        echo "The IP address is:". $request->ip();
14        echo "<br>";
```

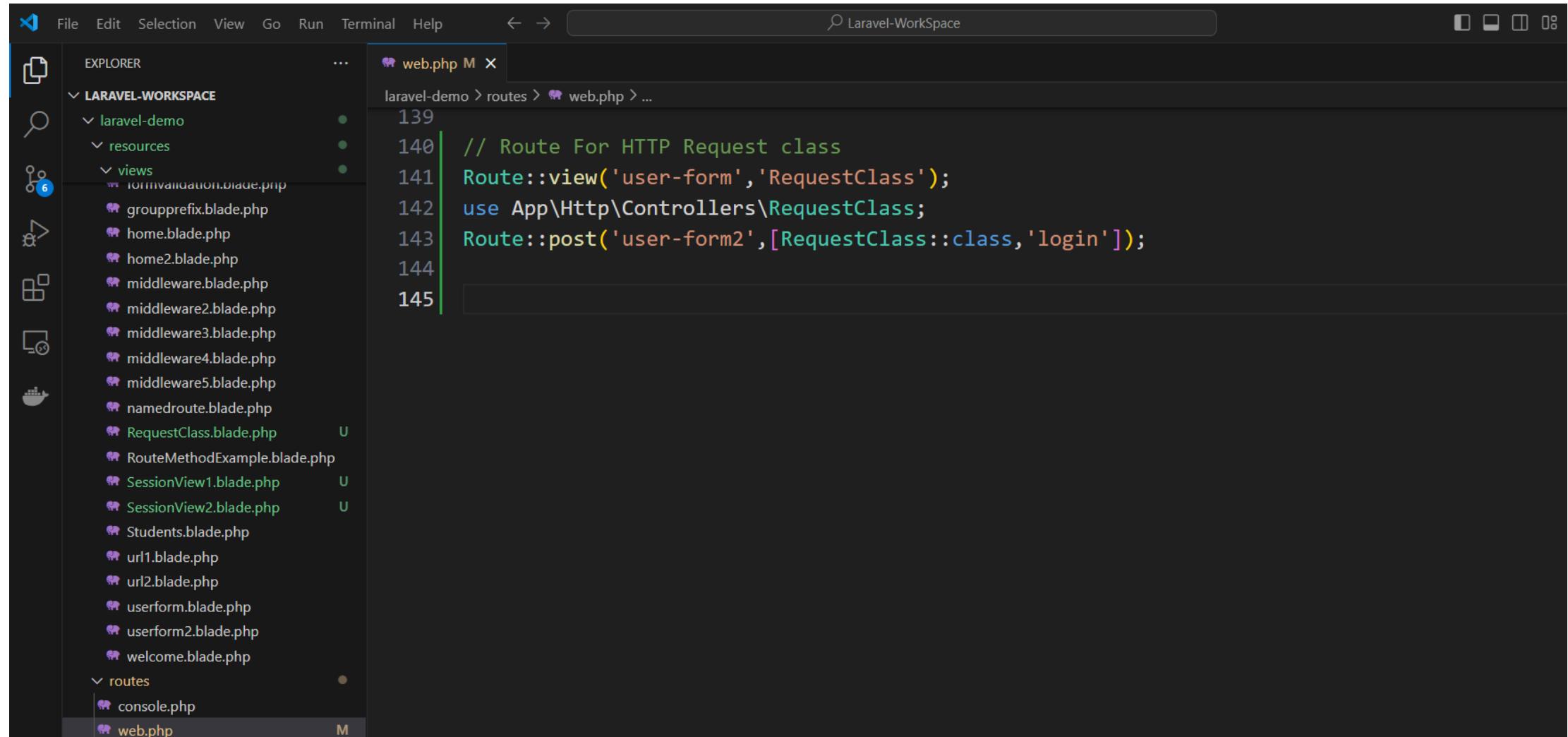
Cont. ...



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The current file is 'RequestClass.php' located in the 'Http\Controllers' directory. The main pane displays the following PHP code:

```
laravel-demo > app > Http > Controllers > RequestClass.php > RequestClass > login
4 class RequestClass extends Controller
6     function login(Request $request){
16         if($request->method('post'))
17         {
18             echo "This is from Post Request";
19         }
20         else
21         {
22             echo "This is from other request";
23         }
24         echo "<br>";
25         if($request->is('user-form2'))
26         {
27             echo "This is user form path Request";
28         }
29         else
30         {
31             echo "This is other path request";
32         }
33     }
34 }
```

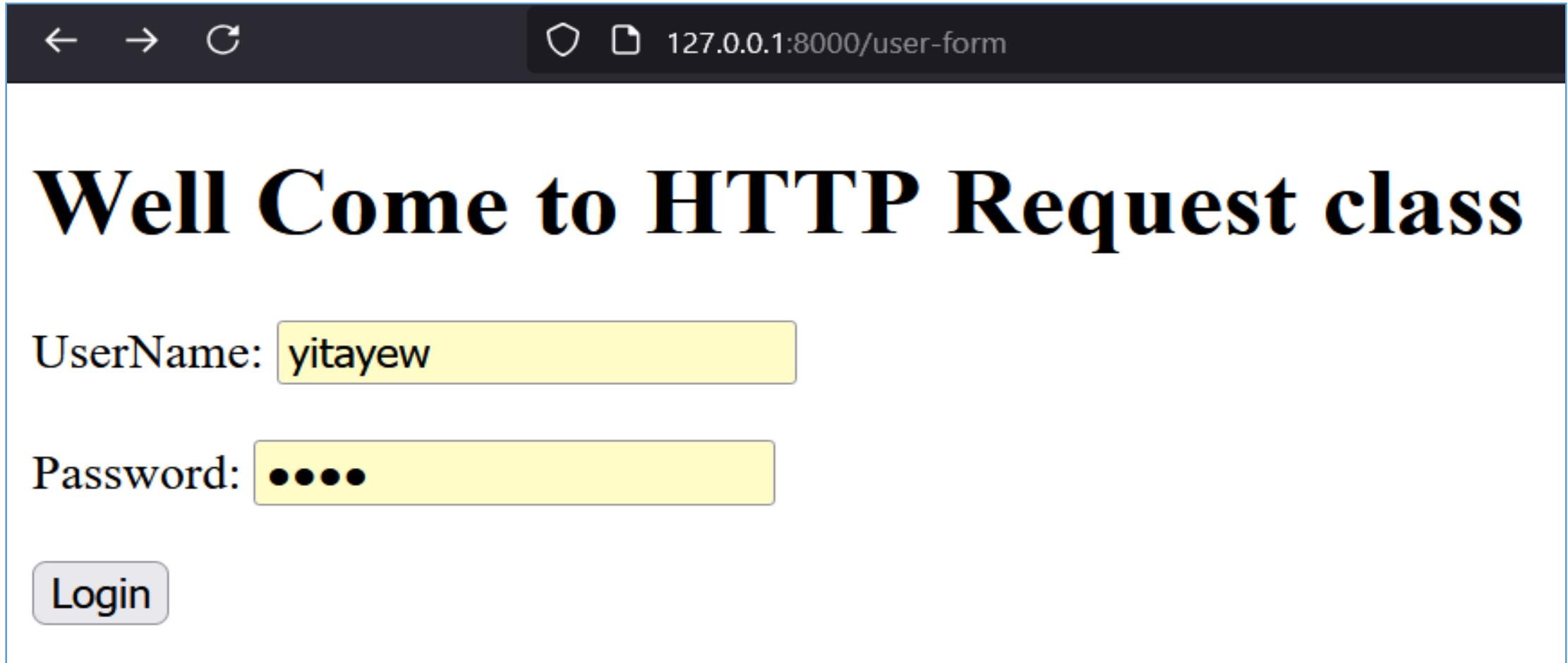
Route



The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. At the top, a navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A search bar on the right contains the text "Laravel-WorkSpace". The left sidebar features a "File Explorer" icon and a tree view of the "LARAVEL-WORKSPACE". Under "laravel-demo", the "resources" folder is expanded, showing "views" (with files like "formvalidation.blade.php", "groupprefix.blade.php", "home.blade.php", "home2.blade.php", "middleware.blade.php", "middleware2.blade.php", "middleware3.blade.php", "middleware4.blade.php", "middleware5.blade.php", "namedroute.blade.php", "RequestClass.blade.php" (marked with a 'U'), "RouteMethodExample.blade.php" (marked with a 'U'), "SessionView1.blade.php" (marked with a 'U'), "SessionView2.blade.php" (marked with a 'U'), "Students.blade.php", "url1.blade.php", "url2.blade.php", "userform.blade.php", "userform2.blade.php", and "welcome.blade.php"), and "routes" (with files like "console.php" and "web.php" both marked with an 'M'). The main editor area displays the "web.php" file under the "routes" directory. The code in the editor is:

```
139
140 // Route For HTTP Request class
141 Route::view('user-form', 'RequestClass');
142 use App\Http\Controllers\RequestClass;
143 Route::post('user-form2', [RequestClass::class, 'login']);
144
145
```

Form(user-form)



The screenshot shows a web browser window with the URL `127.0.0.1:8000/user-form` in the address bar. The page content is displayed below the header.

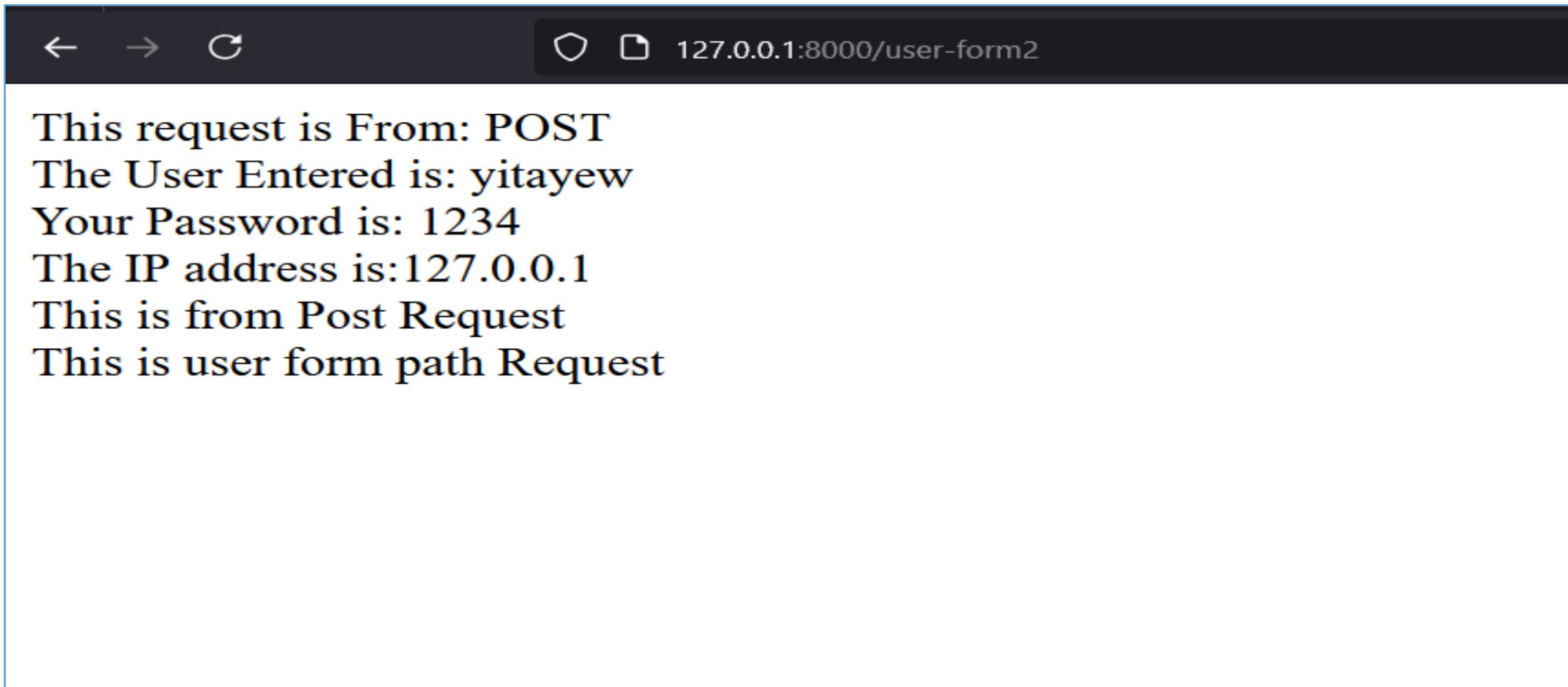
Well Come to HTTP Request class

User Name:

Password:

Login

Output



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:8000/user-form2`. The main content area of the browser displays the following text:

```
This request is From: POST
The User Entered is: yitayew
Your Password is: 1234
The IP address is:127.0.0.1
This is from Post Request
This is user form path Request
```

Common Methods in the Request Class

Common Methods in the Request Class

Method	Description	Example
<code>all()</code>	Retrieves all input data as an array.	<code>\$data = \$request->all();</code>
<code>input('key')</code>	Retrieves a specific input value by key.	<code>\$name = \$request->input('name');</code>
<code>only(['key1', 'key2'])</code>	Retrieves only specific inputs from the request.	<code>\$data = \$request->only(['name', 'email']);</code>
<code>except(['key1', 'key2'])</code>	Retrieves all input data except specified keys.	<code>\$data = \$request->except(['password']);</code>
<code>has('key')</code>	Checks if a particular key is present in the request.	<code>if (\$request->has('name')) { /* logic */ }</code>
<code>filled('key')</code>	Checks if an input is present and not empty.	<code>if (\$request->filled('name')) { /* logic */ }</code>
<code>query('key')</code>	Retrieves a query parameter from the URL.	<code>\$page = \$request->query('page');</code>

Cont. ...

<code>cookie('cookieName')</code>	Retrieves a cookie value.	<code>\$cookie = \$request->cookie('cookieName');</code>
<code>file('fileKey')</code>	Retrieves an uploaded file.	<code>\$file = \$request->file('avatar');</code>
<code>method()</code>	Retrieves the HTTP method of the request (e.g., GET, POST).	<code>\$method = \$request->method();</code>
<code>isMethod('post')</code>	Checks if the request method matches the specified method.	<code>if (\$request->isMethod('post')) { /* logic */ }</code>
<code>header('headerName')</code>	Retrieves a value from the request headers.	<code>\$userAgent = \$request->header('User-Agent');</code>
<code>ip()</code>	Retrieves the client's IP address.	<code>\$ipAddress = \$request->ip();</code>
<code>ajax()</code>	Checks if the request was made via AJAX.	<code>if (\$request->ajax()) { /* logic */ }</code>
<code>json()</code>	Retrieves JSON data from the request, if available.	<code>\$data = \$request->json()->all();</code>

Advantage of Request Class

Key Benefits

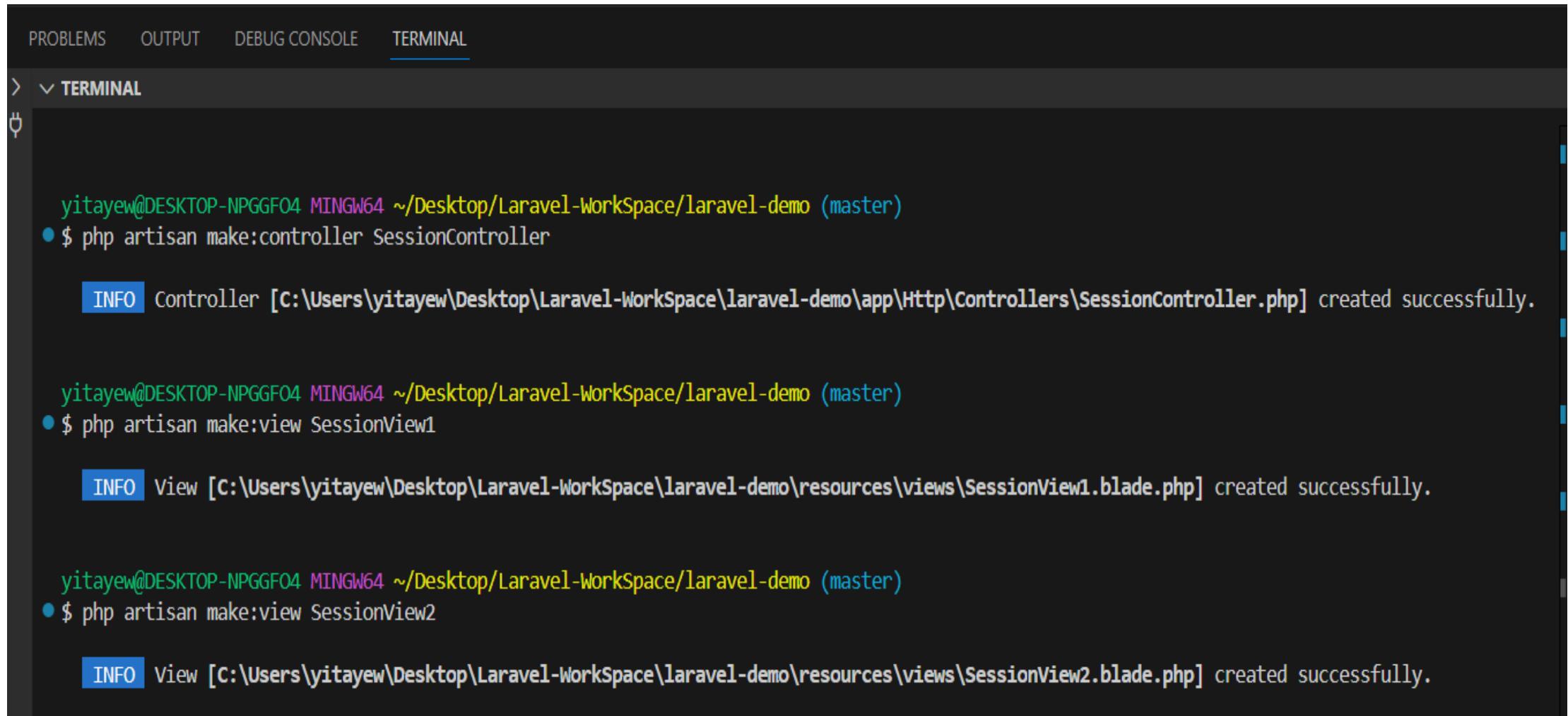
1. **Convenient Access:** The `Request` class provides easy access to all aspects of the HTTP request.
2. **Validation Integration:** Request validation can be performed directly on the incoming data.
3. **Flexible Input Handling:** Request inputs, files, headers, and more can be accessed, making it flexible for handling user data.

The `Request` class is foundational for processing HTTP requests in Laravel, making it straightforward to work with user input, files, and headers in a unified way.

Session in laravel

- In Laravel, a session is a way to **store information** about a user across multiple requests, allowing data to persist between pages. Sessions are commonly used for tasks such as storing **user authentication** data, **flash messages**, form data, and other **temporary** user information.
- Laravel provides a robust session management system, which is configured in the config/session.php file, and supports several backends for session storage, such as file storage, cookies, database, Redis, and memory-based stores like Memcached.

Example (Creating Controller and View)



The screenshot shows a terminal window within a dark-themed IDE interface. The terminal tab is selected at the top. The output shows three separate command executions:

- The first execution creates a controller:

```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:controller SessionController
```

INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\SessionController.php] created successfully.
- The second execution creates a view:

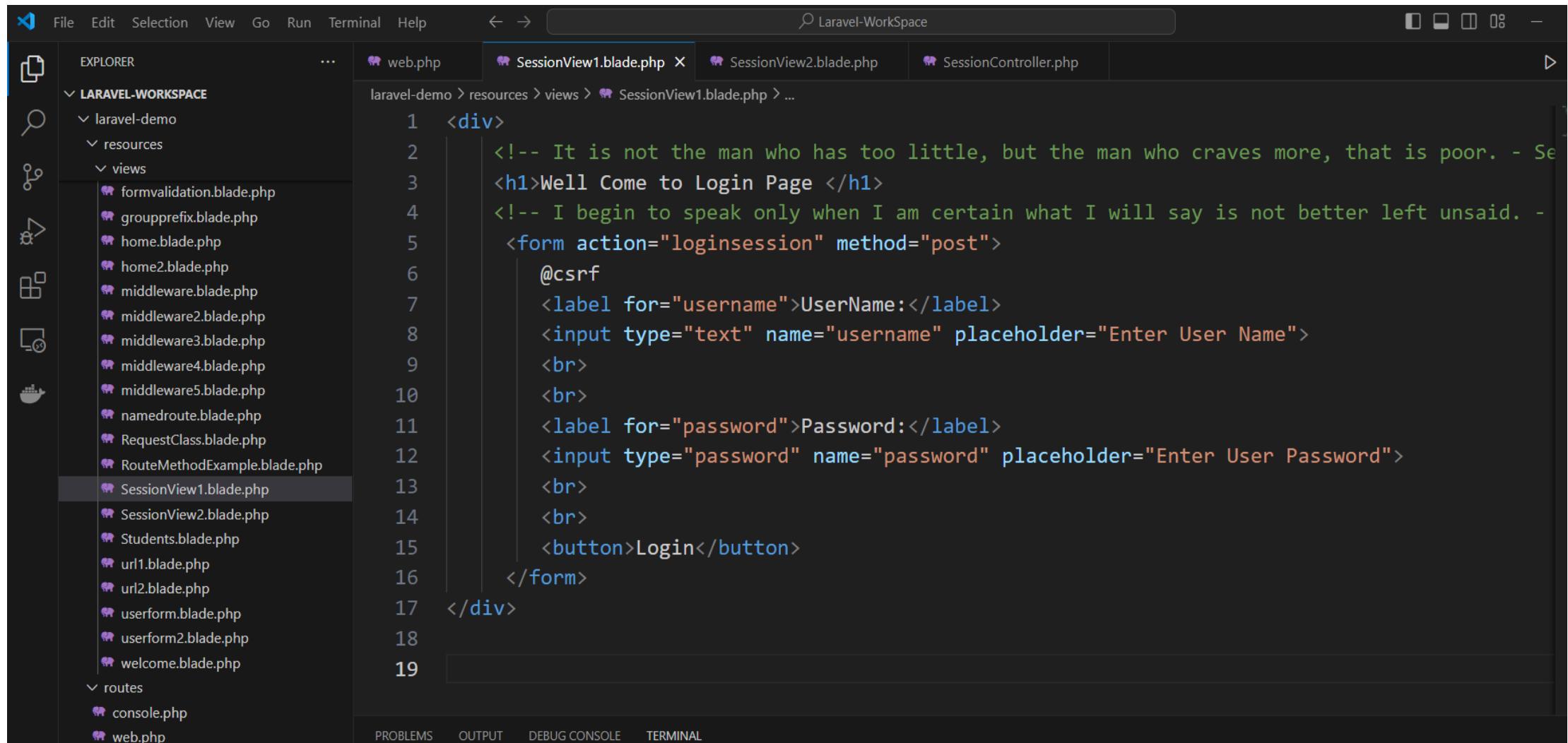
```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view SessionView1
```

INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\SessionView1.blade.php] created successfully.
- The third execution creates another view:

```
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view SessionView2
```

INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\SessionView2.blade.php] created successfully.

Example: View(SessionView1)



The screenshot shows the Visual Studio Code interface with the following details:

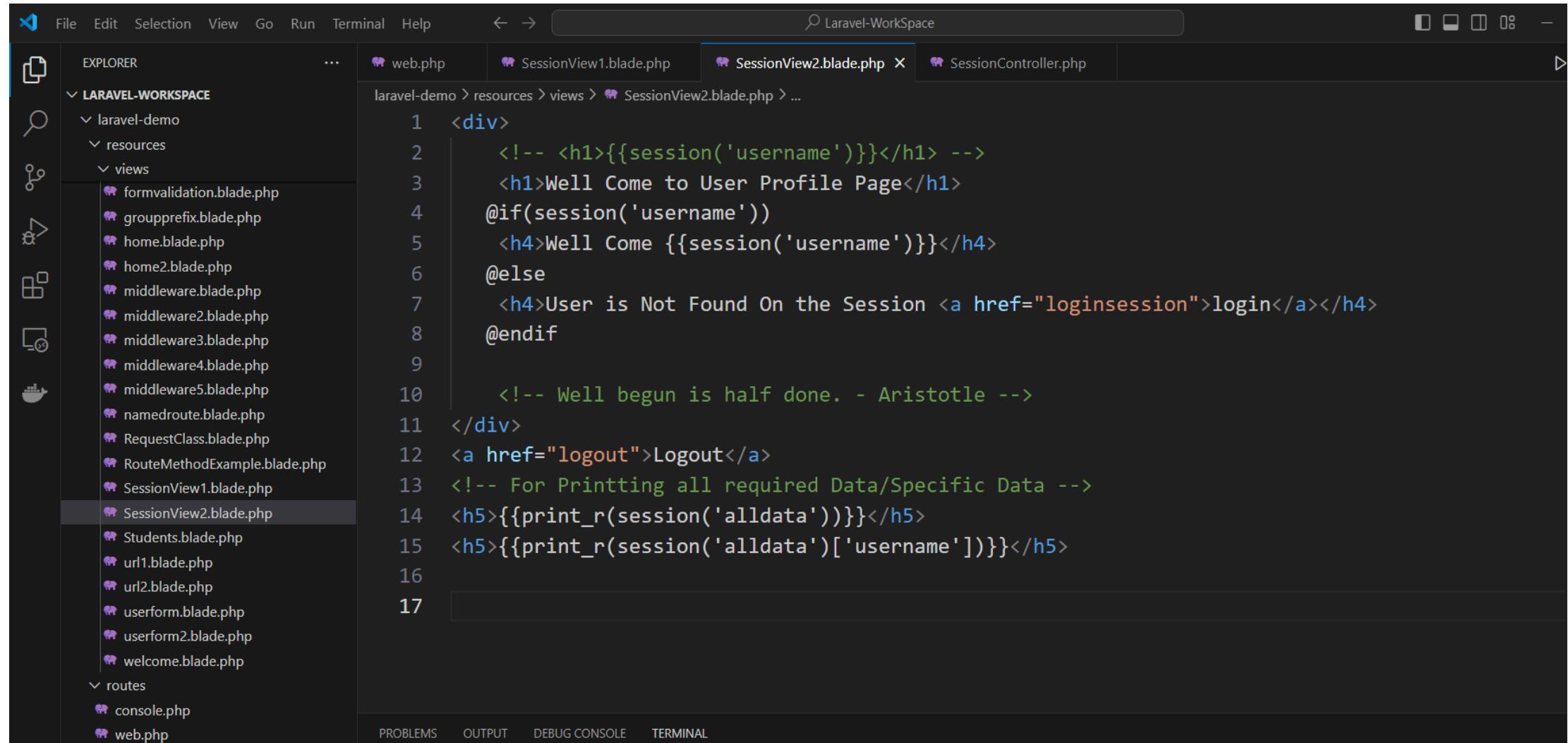
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Laravel-WorkSpace
- Explorer:** Shows the project structure under LARAVEL-WORKSPACE / laravel-demo / resources / views. The file SessionView1.blade.php is currently selected.
- Editor:** Displays the contents of SessionView1.blade.php. The code is as follows:

```
1 <div>
2   <!-- It is not the man who has too little, but the man who craves more, that is poor. - Se
3   &lt;h1&gt;Well Come to Login Page &lt;/h1&gt;
4   &lt;!-- I begin to speak only when I am certain what I will say is not better left unsaid. -
5   &lt;form action="loginsession" method="post"&gt;
6     @csrf
7     &lt;label for="username"&gt;UserName:&lt;/label&gt;
8     &lt;input type="text" name="username" placeholder="Enter User Name"&gt;
9
10    &lt;br&gt;
11    &lt;br&gt;
12    &lt;label for="password"&gt;Password:&lt;/label&gt;
13    &lt;input type="password" name="password" placeholder="Enter User Password"&gt;
14
15    &lt;br&gt;
16    &lt;br&gt;
17    &lt;button&gt;Login&lt;/button&gt;
18
19</pre>

At the bottom, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL.


```

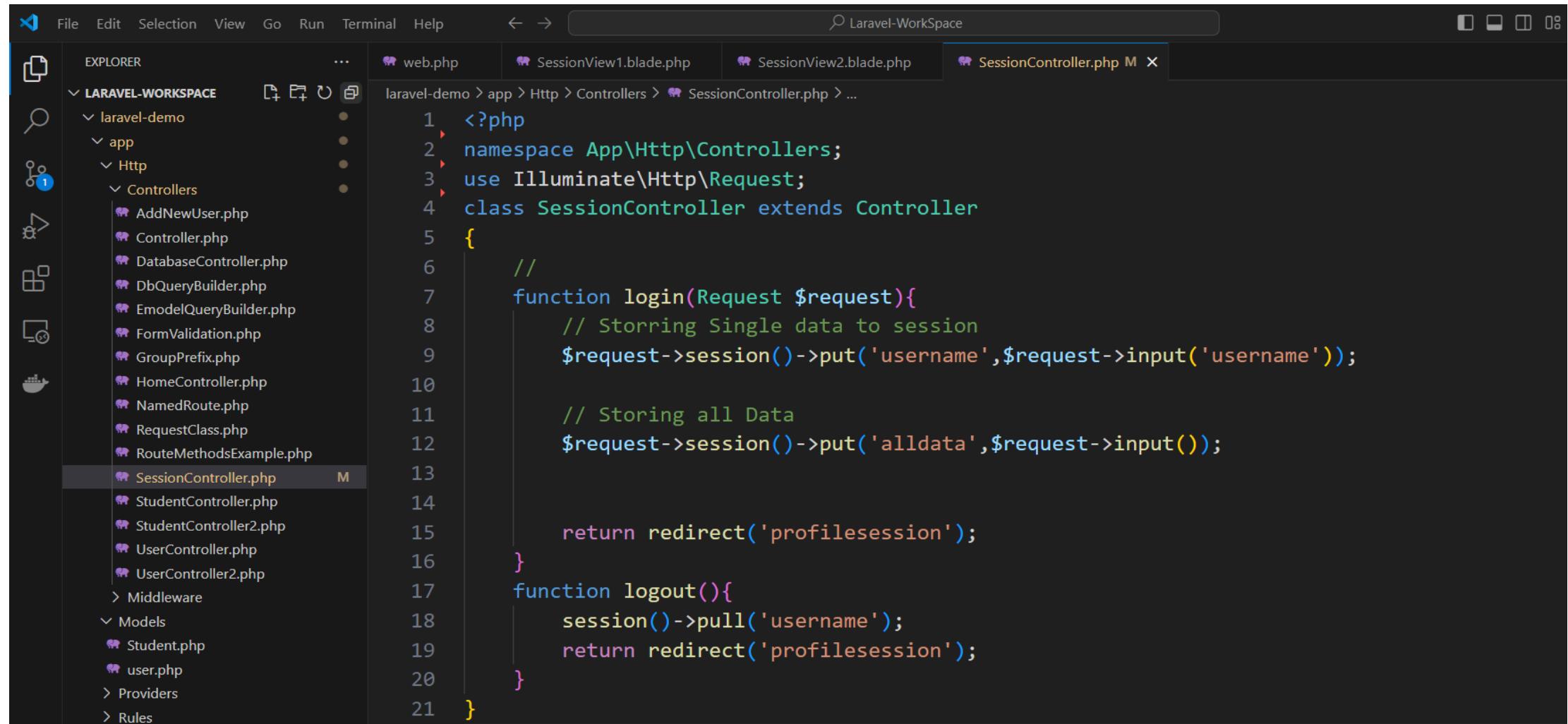
View (SessionView2)



The screenshot shows a dark-themed interface of the Visual Studio Code code editor. The title bar reads "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the file structure of a Laravel project named "laravel-demo". Under "resources/views", several blade files are listed, including "formvalidation.blade.php", "grouprefix.blade.php", "home.blade.php", "home2.blade.php", "middleware.blade.php", "middleware2.blade.php", "middleware3.blade.php", "middleware4.blade.php", "middleware5.blade.php", "namedroute.blade.php", "RequestClass.blade.php", "RouteMethodExample.blade.php", "SessionView1.blade.php", "SessionView2.blade.php" (which is currently selected), "Students.blade.php", "url1.blade.php", "url2.blade.php", "userform.blade.php", "userform2.blade.php", and "welcome.blade.php". Below these, there are "routes", "console.php", and "web.php". The main editor area displays the contents of "SessionView2.blade.php". The code uses Blade templating syntax to display session data and provide navigation links.

```
1 <div>
2   <!-- <h1>{{session('username')}}</h1> -->
3   <h1>Well Come to User Profile Page</h1>
4   @if(session('username'))
5     <h4>Well Come {{session('username')}}</h4>
6   @else
7     <h4>User is Not Found On the Session <a href="loginsession">login</a></h4>
8   @endif
9
10  <!-- Well begun is half done. - Aristotle -->
11 </div>
12 <a href="logout">Logout</a>
13 <!-- For Printting all required Data/Specific Data -->
14 <h5>{{print_r(session('alldata'))}}</h5>
15 <h5>{{print_r(session('alldata')['username'])}}</h5>
16
17
```

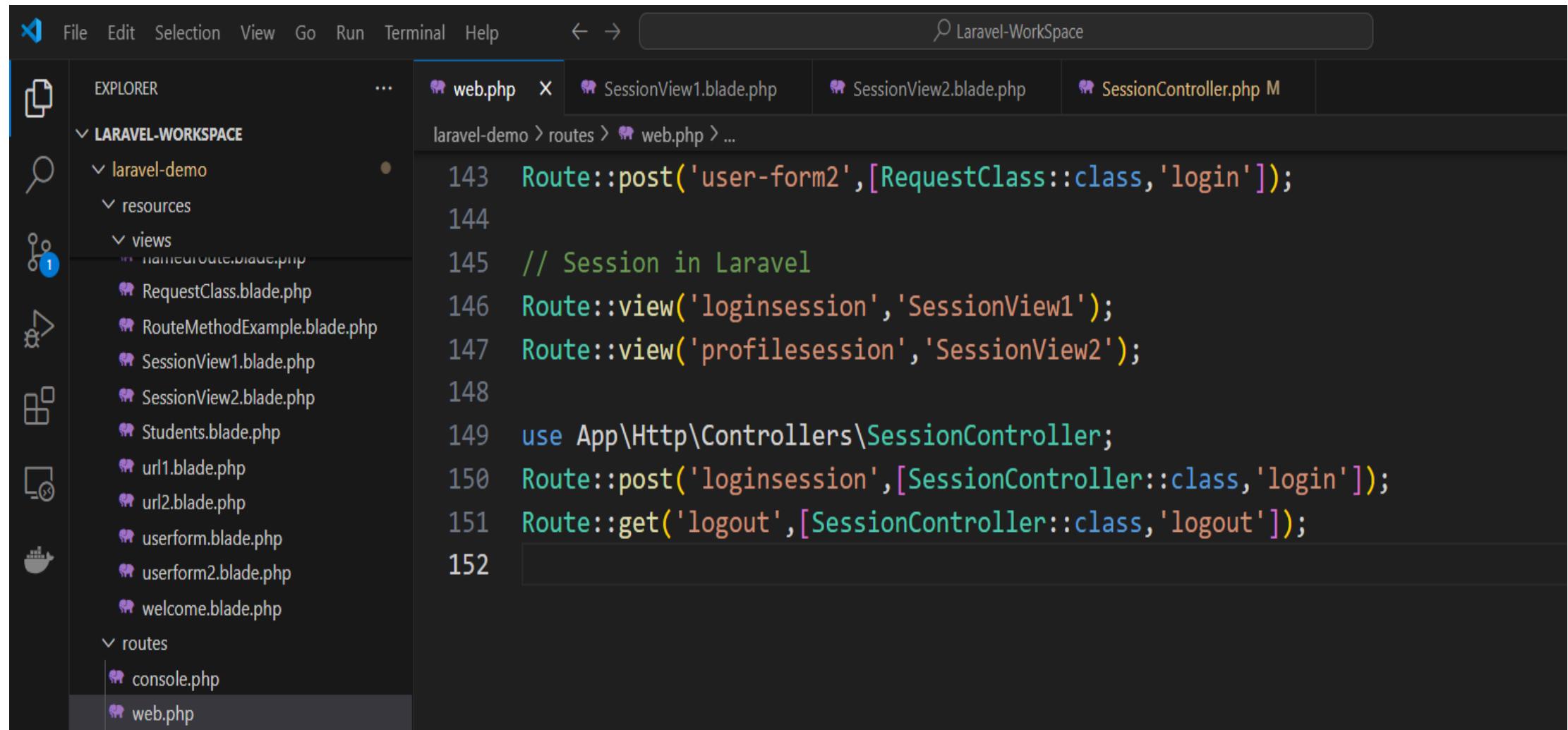
Controller (SessionController)



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons and a tree view of the project structure under 'LARAVEL-WORKSPACE'. The main area displays the content of 'SessionController.php'.

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 class SessionController extends Controller
5 {
6     //
7     function login(Request $request){
8         // Storing Single data to session
9         $request->session()->put('username',$request->input('username'));
10
11        // Storing all Data
12        $request->session()->put('alldata',$request->input());
13
14
15        return redirect('profilesession');
16    }
17    function logout(){
18        session()->pull('username');
19        return redirect('profilesession');
20    }
21 }
```

Route



The screenshot shows a code editor interface with a dark theme. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A search bar on the right contains the text "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems, Files, and Diff. The Explorer panel shows a project structure under "LARAVEL-WORKSPACE": "laravel-demo" > "routes" > "web.php". The main editor area displays the "web.php" file content:

```
143 Route::post('user-form2',[RequestClass::class,'login']);  
144  
145 // Session in Laravel  
146 Route::view('loginsession','SessionView1');  
147 Route::view('profilesession','SessionView2');  
148  
149 use App\Http\Controllers\SessionController;  
150 Route::post('loginsession',[SessionController::class,'login']);  
151 Route::get('logout',[SessionController::class,'logout']);  
152
```

Output

Well Come to Login Page

UserName:

Password:

[Login](#)

Well Come to User Profile Page

User is Not Found On the Session [login](#)

[Logout](#)

```
Array ( [_token] => vGESZXrCkzxyP70EJf4DQbXM6Zis5biNOTuVBOQq [username] => yared [password] => 123 ) 1
```

yared1

Well Come to User Profile Page

Well Come yitayew

[Logout](#)

```
Array ( [_token] => vGESZXrCkzxyP70EJf4DQbXM6Zis5biNOTuVBOQq [username] => yitayew [password] => 12346 ) 1
```

yitayew1

Well Come to Login Page

UserName:

Password:

[Login](#)

Upload file | Upload and Display Image

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> ✓ TERMINAL
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view UploadFile

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

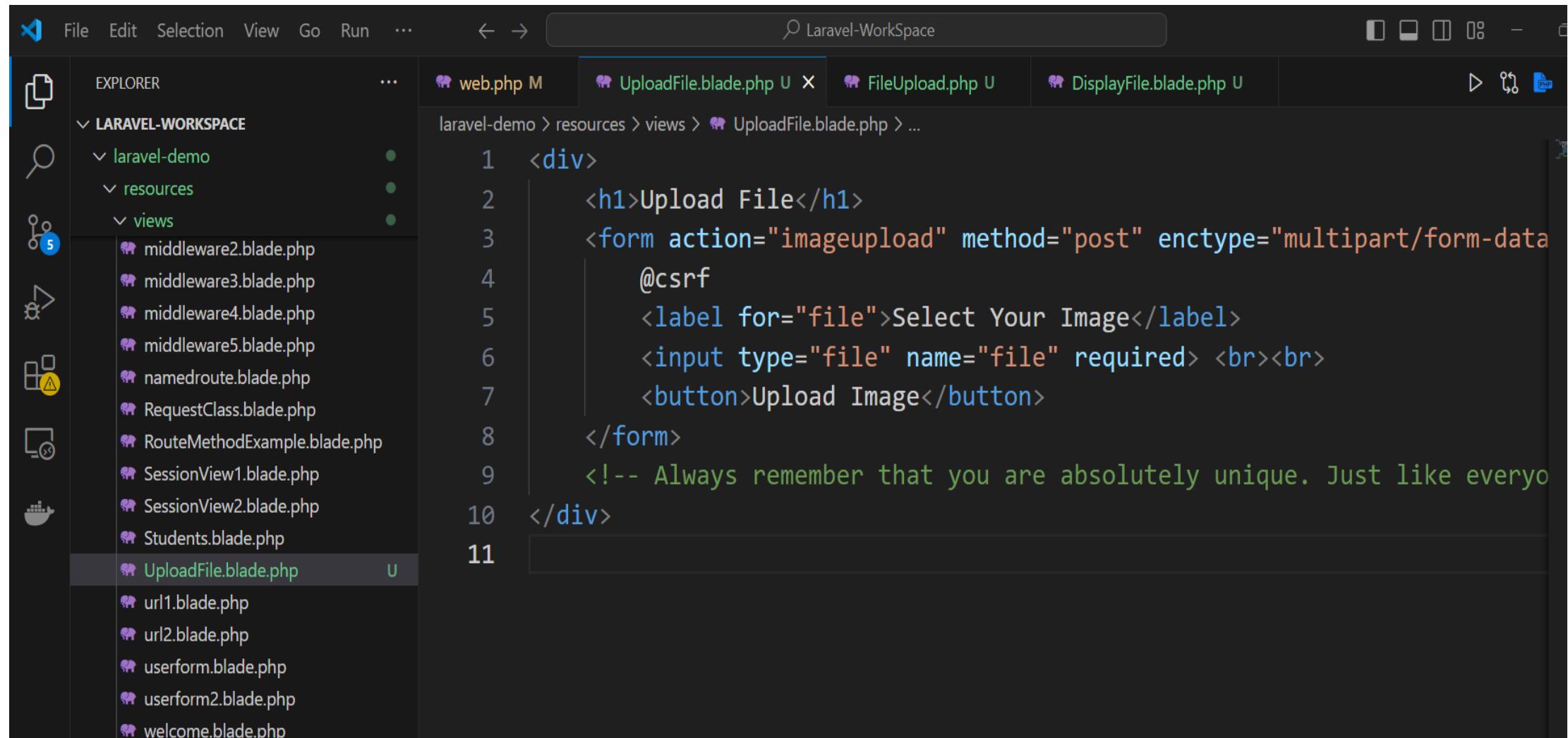
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view DisplayFile

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

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:controller FileUpload

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

View (FileUpload)



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists a project structure under 'LARAVEL-WORKSPACE'. The 'views' folder contains several Blade files, including 'UploadFile.blade.php', which is currently selected and highlighted in green. The main editor area displays the contents of 'UploadFile.blade.php'. The code is as follows:

```
1 <div>
2   <h1>Upload File</h1>
3   <form action="imageupload" method="post" enctype="multipart/form-data">
4     @csrf
5     <label for="file">Select Your Image</label>
6     <input type="file" name="file" required> <br><br>
7     <button>Upload Image</button>
8   </form>
9   <!-- Always remember that you are absolutely unique. Just like everyone-->
10 </div>
```

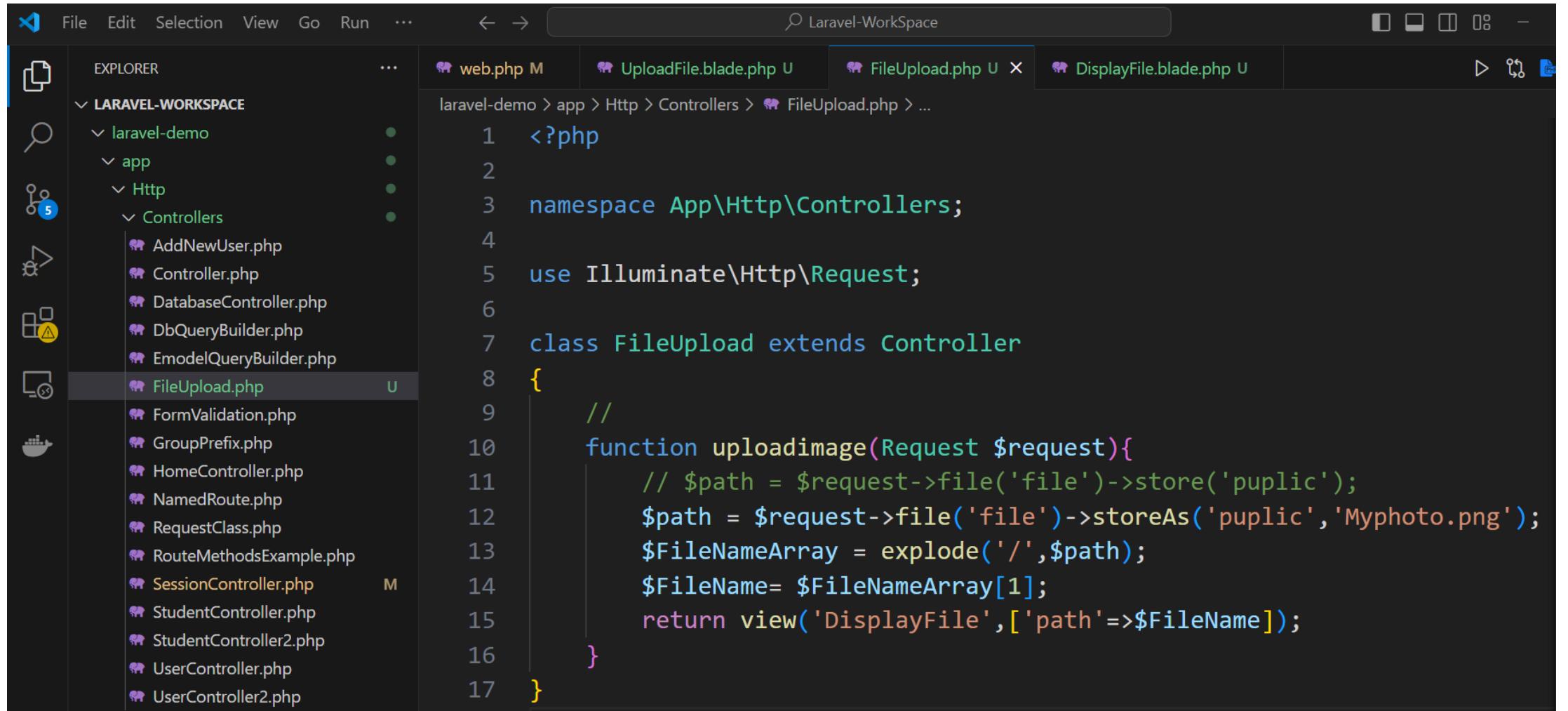
View(DisplayFile)

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Laravel-WorkSpace
- Toolbars:** Standard icons for file operations.
- Left Sidebar (EXPLORER):** Shows the project structure:
 - LARAVEL-WORKSPACE
 - laravel-demo
 - public
 - resources
 - css
 - js
 - views
 - admin
 - components
 - layouts
 - about.blade.php
 - databaseuser.blade.php
 - DBqueryView.blade.php
 - Demo.blade.php
 - DisplayFile.blade.php** (highlighted)
 - EmodelQueryBuilder.blade.php
 - formvalidation.blade.php
 - grouprefix.blade.php
 - home.blade.php
 - home2.blade.php
 - middleware.blade.php
 - Central Area:** Displays the content of the selected file, `DisplayFile.blade.php`.

```
1 <div>
2   
3   <!-- You must be the change you wish to see in the world. - Mahatma
4 </div>
5 
```

Controller (FileUpload)



The screenshot shows a code editor interface with the title bar "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The "laravel-demo" folder contains an "app" folder, which has an "Http" folder containing a "Controllers" folder. Inside "Controllers", there are several files: AddNewUser.php, Controller.php, DatabaseController.php, DbQueryBuilder.php, EmodelQueryBuilder.php, FileUpload.php (which is currently selected and highlighted in blue), FormValidation.php, GroupPrefix.php, HomeController.php, NamedRoute.php, RequestClass.php, RouteMethodsExample.php, SessionController.php (marked with a "M" icon), StudentController.php, StudentController2.php, UserController.php, and UserController2.php. The main editor area displays the "FileUpload.php" code:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class FileUpload extends Controller
8 {
9     //
10    function uploadimage(Request $request){
11        // $path = $request->file('file')->store('puplic');
12        $path = $request->file('file')->storeAs('puplic', 'Myphoto.png');
13        $FileNameArray = explode('/', $path);
14        $FileName= $FileNameArray[1];
15        return view('DisplayFile', ['path'=>$FileName]);
16    }
17 }
```

Route

The screenshot shows a development environment for a Laravel application named "laravel-demo". The application structure is visible in the Explorer sidebar, including files like middleware2.blade.php, routes/console.php, and routes/web.php. The main code editor displays routes/web.php with the following code:

```
// File Uploading | Image
Route::view('imageupload','UploadFile');
use App\Http\Controllers\FileUpload;
Route::post('imageupload',[FileUpload::class,'uploadimage']);
```

The browser window shows a "Upload File" page with a file input field labeled "Select Your Image" and a "Browse..." button. The file "photo.jpg" is selected. Below the input is a "Upload Image" button.

Thank you!

Appreciate your action.