CORE MODULE 3

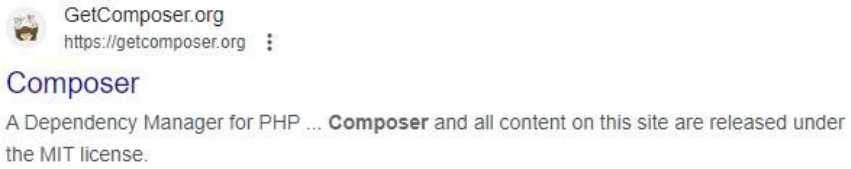
Backend Web Technologies & Frameworks

Installation of the Laravel in the System.

Install Composer

STEP1: Go to the browser and type Composer

https://getcomposer.org/ click this link to Download composer

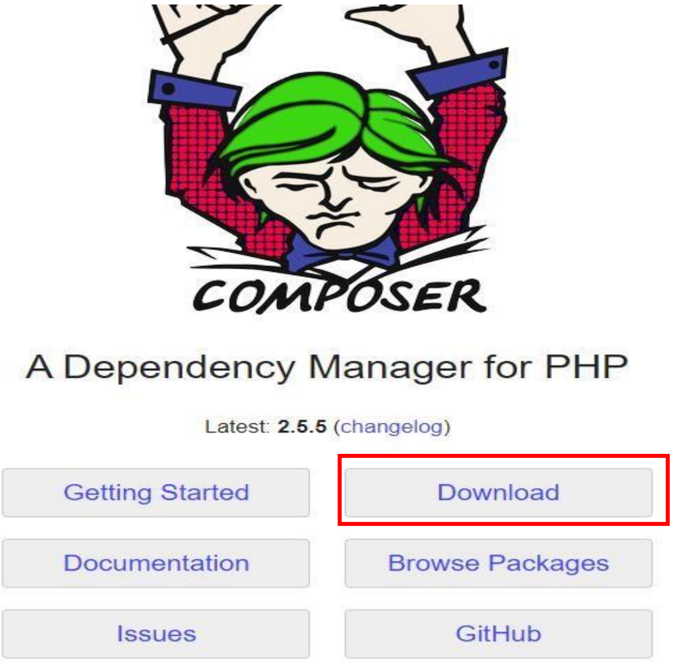


GetComposer.org
https://getcomposer.org

Composer

A Dependency Manager for PHP ... **Composer** and all content on this site are released under the MIT license.

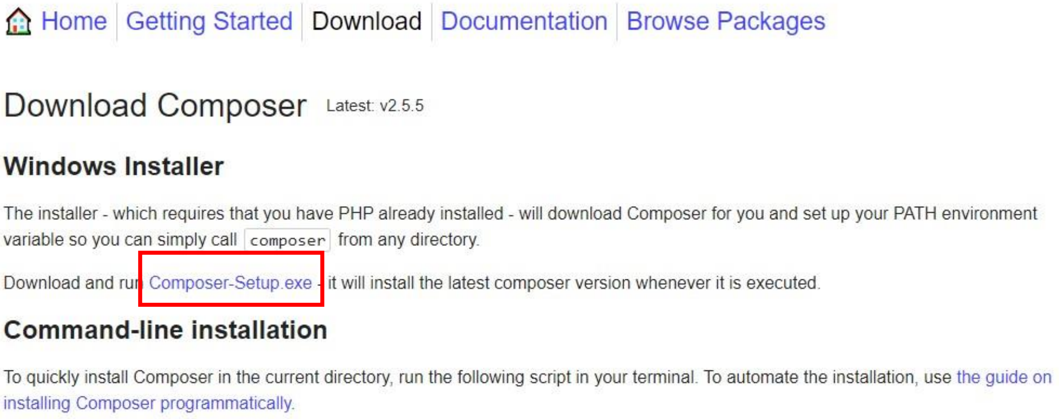
- Click on Download:



A Dependency Manager for PHP

Latest: 2.5.5 (changelog)

- Click on composerSetup.exe it.



Download Composer Latest: v2.5.5

Windows Installer

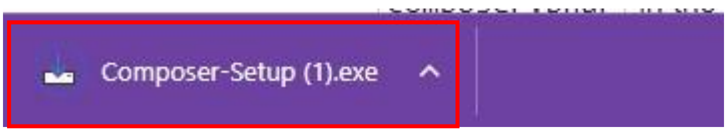
The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run Composer-Setup.exe - it will install the latest composer version whenever it is executed.

Command-line installation

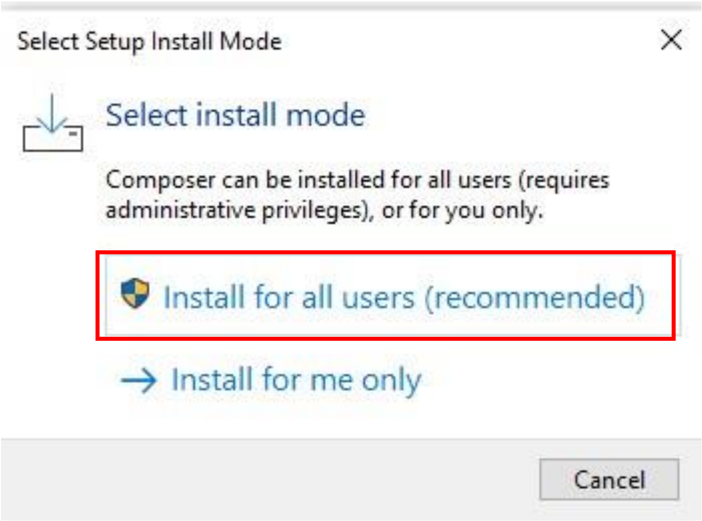
To quickly install Composer in the current directory, run the following script in your terminal. To automate the installation, use the guide on installing Composer programmatically.

- After download open .exe file



- Install Composer in your System:

i) Select install for all user:

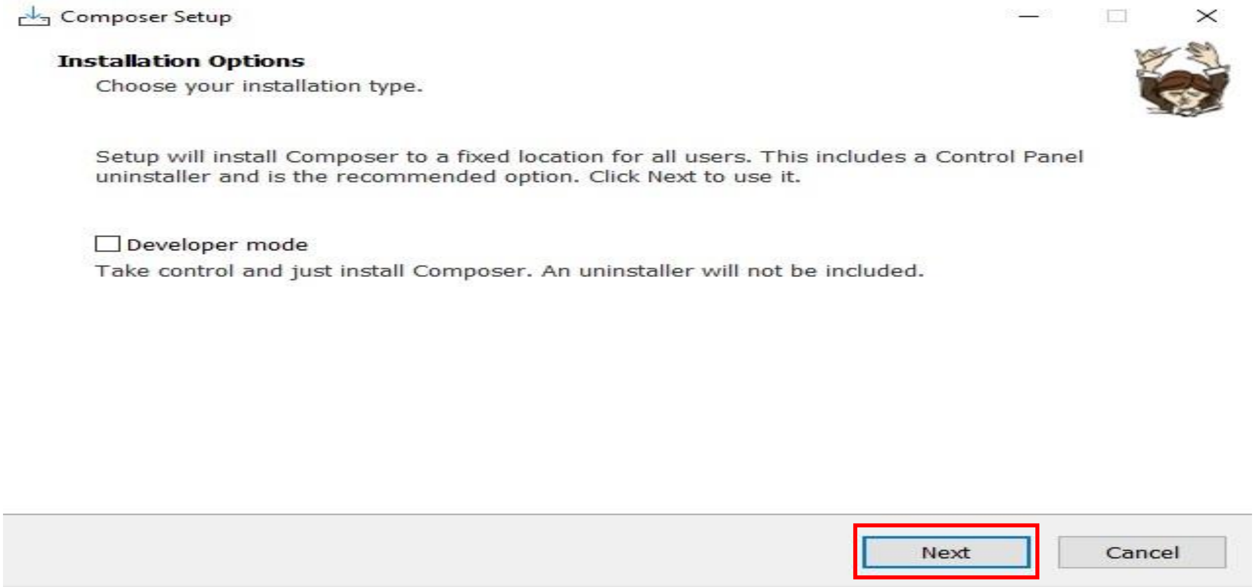


Select Setup Install Mode

Select install mode

Composer can be installed for all users (requires administrative privileges), or for you only.

ii) Click on Yes Composer setup. iii) Click on next.



Composer Setup

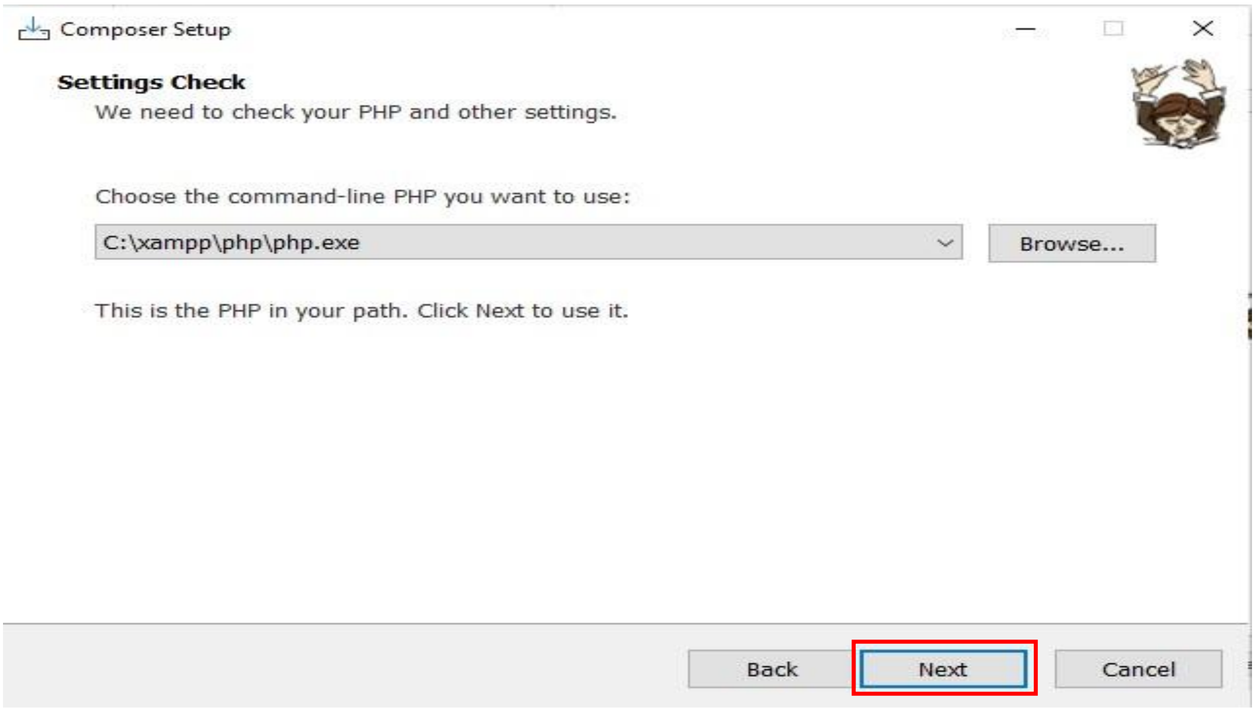
Installation Options
Choose your installation type.

Setup will install Composer to a fixed location for all users. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.

□ Developer mode
Take control and just install Composer. An uninstaller will not be included.

iv) Set path :

- Choose your php path then click on next.


Composer Setup

Settings Check

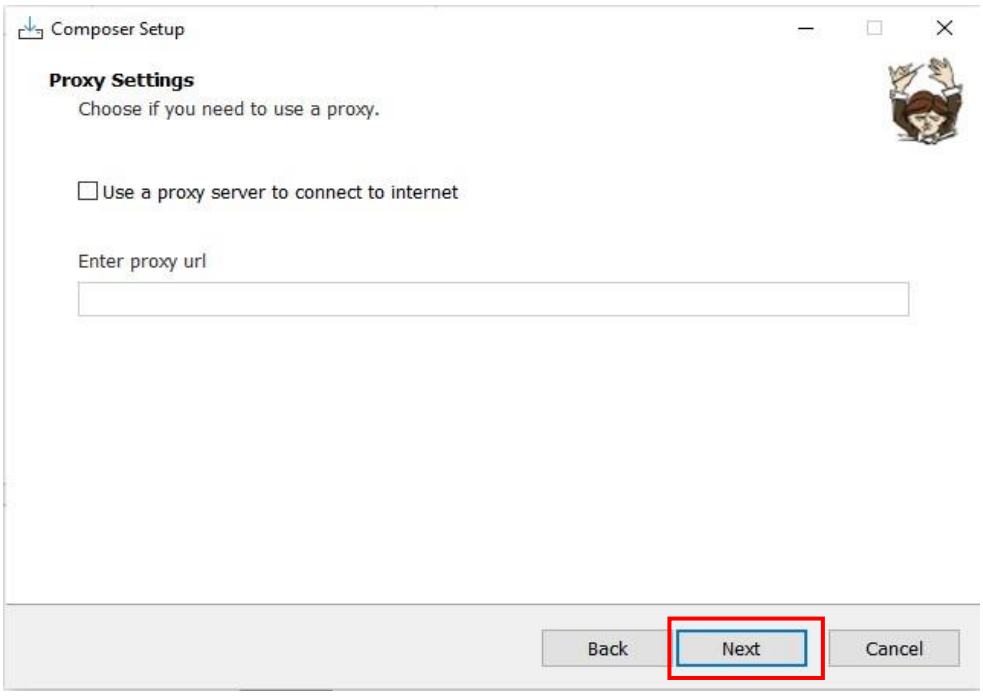
We need to check your PHP and other settings.

Choose the command-line PHP you want to use:

C:\xampp\php\php.exe

This is the PHP in your path. Click Next to use it.

v) Click on next.


Composer Setup

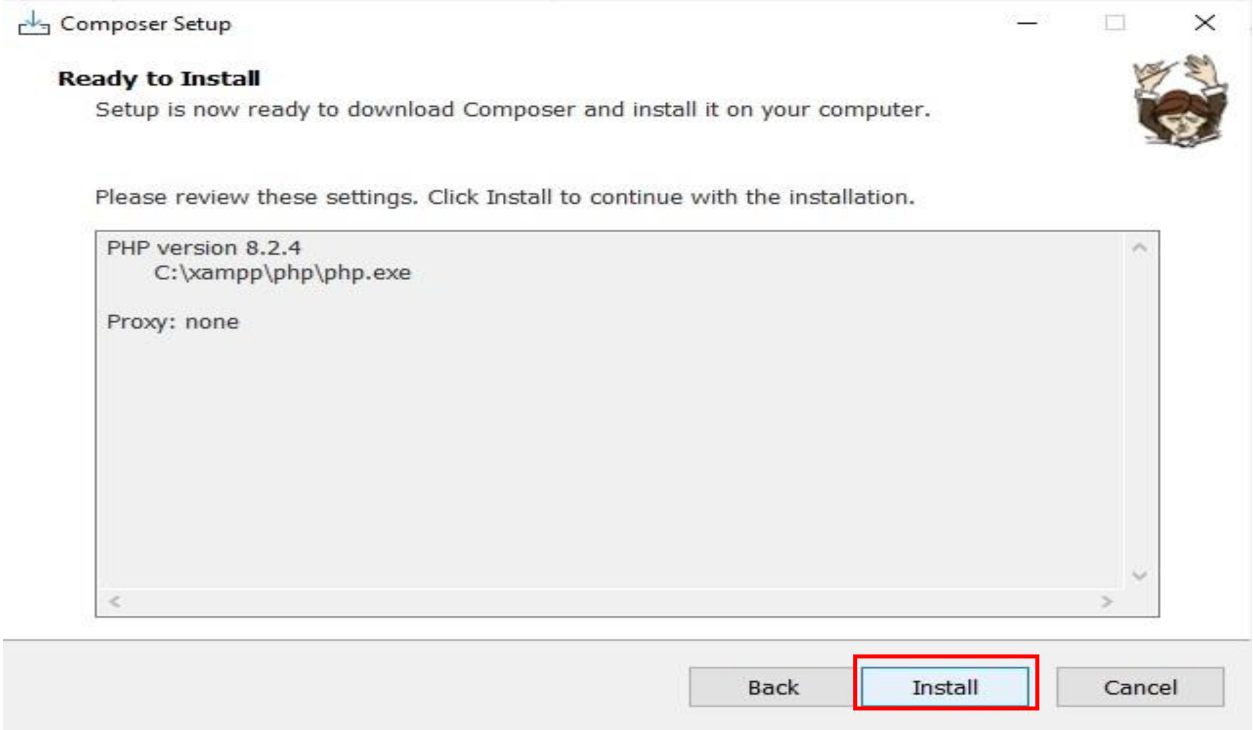
Proxy Settings

Choose if you need to use a proxy.

□ Use a proxy server to connect to internet

Enter proxy url

vi) Click on Install.



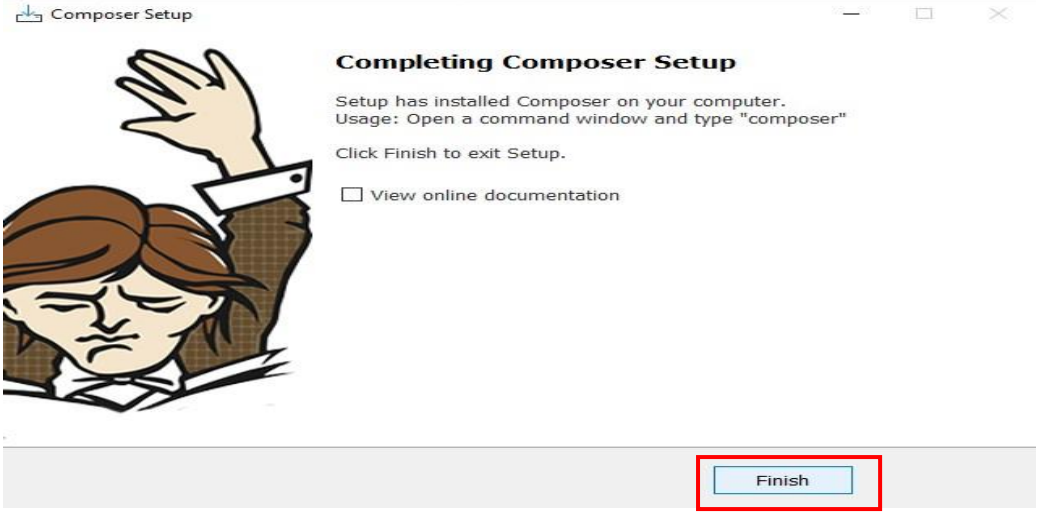
Ready to Install

Setup is now ready to download Composer and install it on your computer.

Please review these settings. Click Install to continue with the installation.

```
PHP version 8.2.4
    C:\xampp\php\php.exe

Proxy: none
```

vii) Click on Finish.



Completing Composer Setup

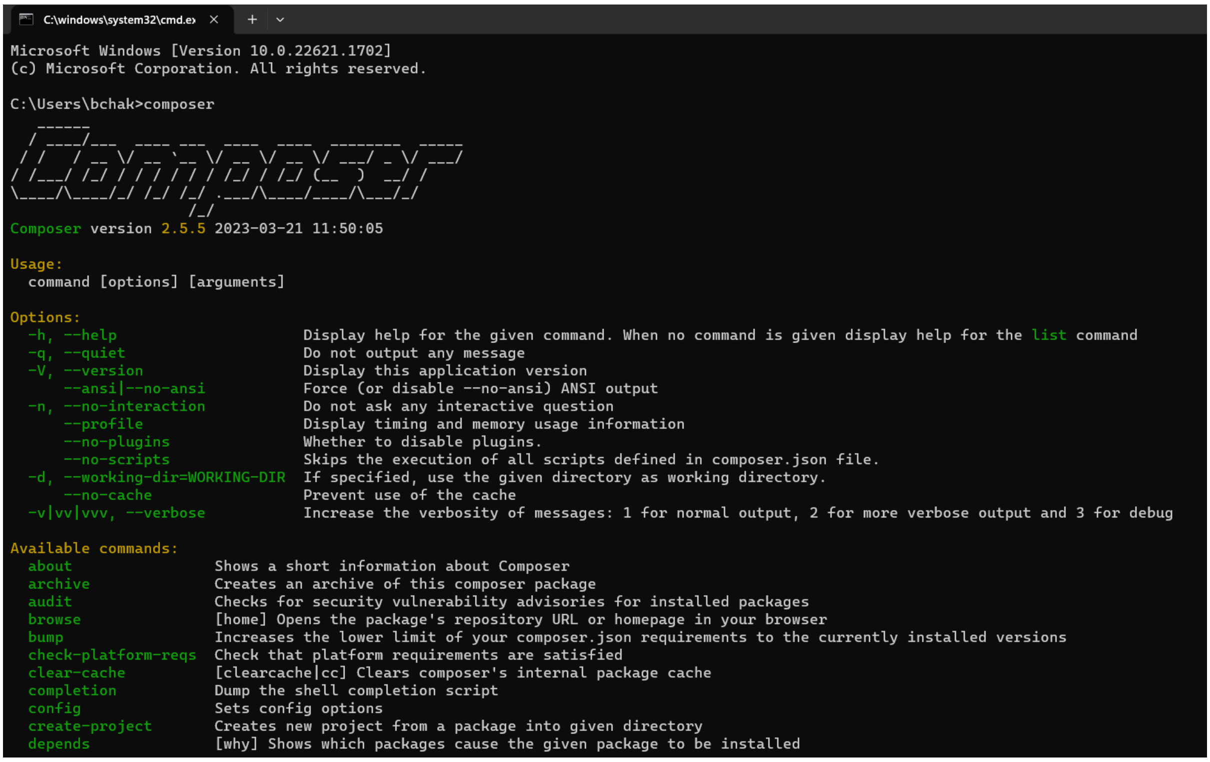
Setup has installed Composer on your computer.
Usage: Open a command window and type "composer"

Click Finish to exit Setup.

□ View online documentation

NOTE:

After that check Composer successfully installed or not! Go to Command Prompt(cmd)---------> then type composer---------> hit enter---------> then if you are able to see the Composer version that means you are successfully installed Composer.

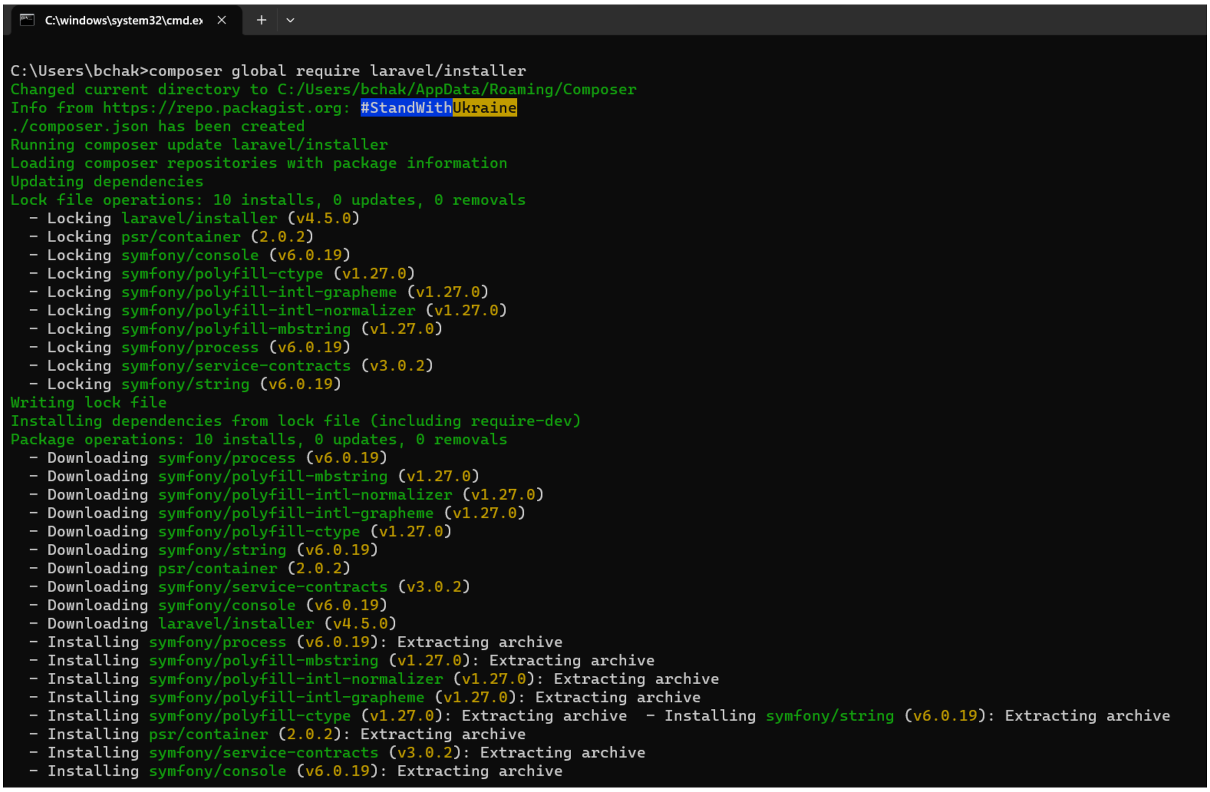


```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bchak>composer
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.5.5 2023-03-21 11:50:05

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display help for the given command. When no command is given display help for the list command
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi|--no-ansi           Force (or disable --no-ansi) ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directory.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  about                Shows a short information about Composer
  archive              Creates an archive of this composer package
  audit                Checks for security vulnerability advisories for installed packages
  browse               [home] Opens the package's repository URL or homepage in your browser
  bump                 Increases the lower limit of your composer.json requirements to the currently installed versions
  check-platform-reqs  Check that platform requirements are satisfied
  clear-cache          [clearcache|cc] Clears composer's internal package cache
  completion           Dump the shell completion script
  config               Sets config options
  create-project       Creates new project from a package into given directory
  depends              [why] Shows which packages cause the given package to be installed
```

Installing Laravel

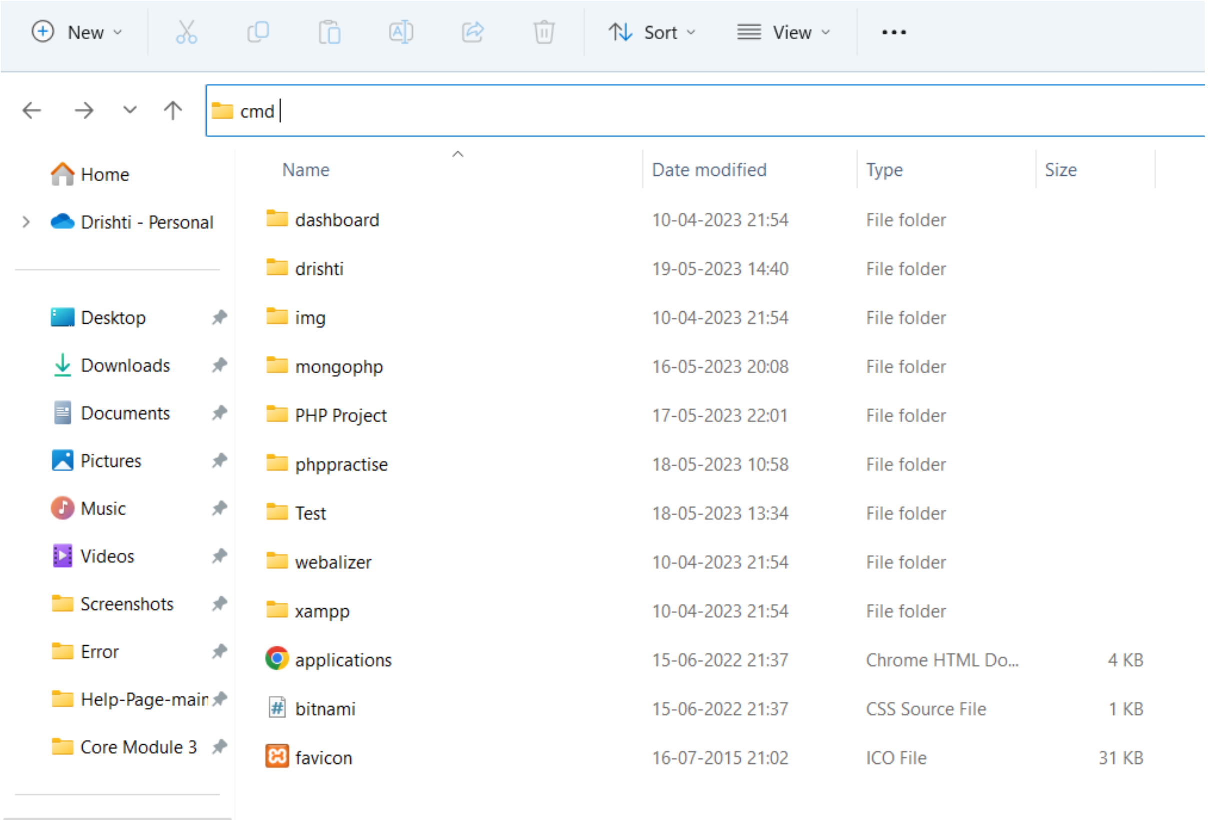
Type the following command in your command prompt window:

- **composer global require laravel/installer** in cmd.



```
C:\Users\bchak>composer global require laravel/installer
Changed current directory to C:/Users/bchak/AppData/Roaming/Composer
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been created
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 10 installs, 0 updates, 0 removals
  - Locking laravel/installer (v4.5.0)
  - Locking psr/container (2.0.2)
  - Locking symfony/console (v6.0.19)
  - Locking symfony/polyfill-ctype (v1.27.0)
  - Locking symfony/polyfill-intl-grapheme (v1.27.0)
  - Locking symfony/polyfill-intl-normalizer (v1.27.0)
  - Locking symfony/polyfill-mbstring (v1.27.0)
  - Locking symfony/process (v6.0.19)
  - Locking symfony/service-contracts (v3.0.2)
  - Locking symfony/string (v6.0.19)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 10 installs, 0 updates, 0 removals
  - Downloading symfony/process (v6.0.19)
  - Downloading symfony/polyfill-mbstring (v1.27.0)
  - Downloading symfony/polyfill-intl-normalizer (v1.27.0)
  - Downloading symfony/polyfill-intl-grapheme (v1.27.0)
  - Downloading symfony/polyfill-ctype (v1.27.0)
  - Downloading symfony/string (v6.0.19)
  - Downloading psr/container (2.0.2)
  - Downloading symfony/service-contracts (v3.0.2)
  - Downloading symfony/console (v6.0.19)
  - Downloading laravel/installer (v4.5.0)
  - Installing symfony/process (v6.0.19): Extracting archive
  - Installing symfony/polyfill-mbstring (v1.27.0): Extracting archive
  - Installing symfony/polyfill-intl-normalizer (v1.27.0): Extracting archive
  - Installing symfony/polyfill-intl-grapheme (v1.27.0): Extracting archive
  - Installing symfony/polyfill-ctype (v1.27.0): Extracting archive  - Installing symfony/string (v6.0.19): Extracting archive
  - Installing psr/container (2.0.2): Extracting archive
  - Installing symfony/service-contracts (v3.0.2): Extracting archive
  - Installing symfony/console (v6.0.19): Extracting archive
```

Now go to the **htdocs** folder in **C Drive** and type **cmd** in the **address bar.**

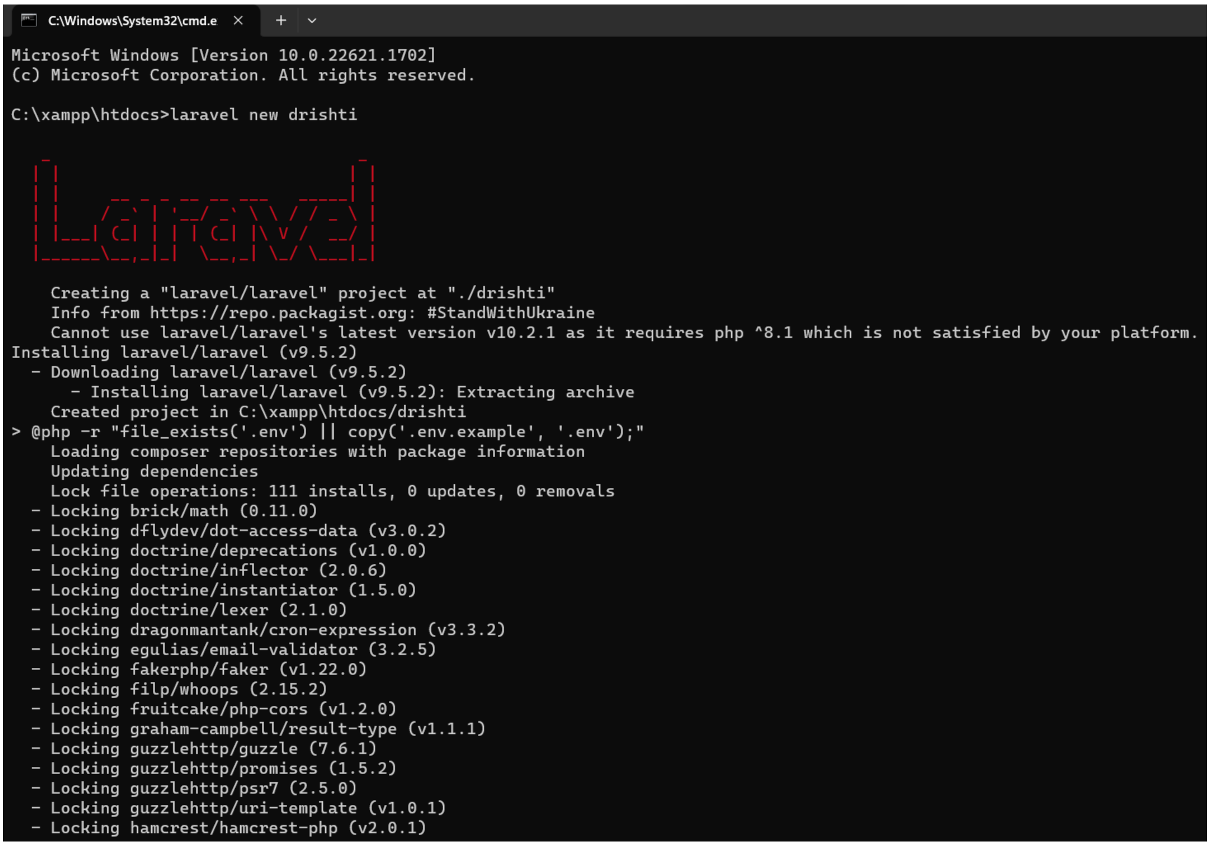


cmd

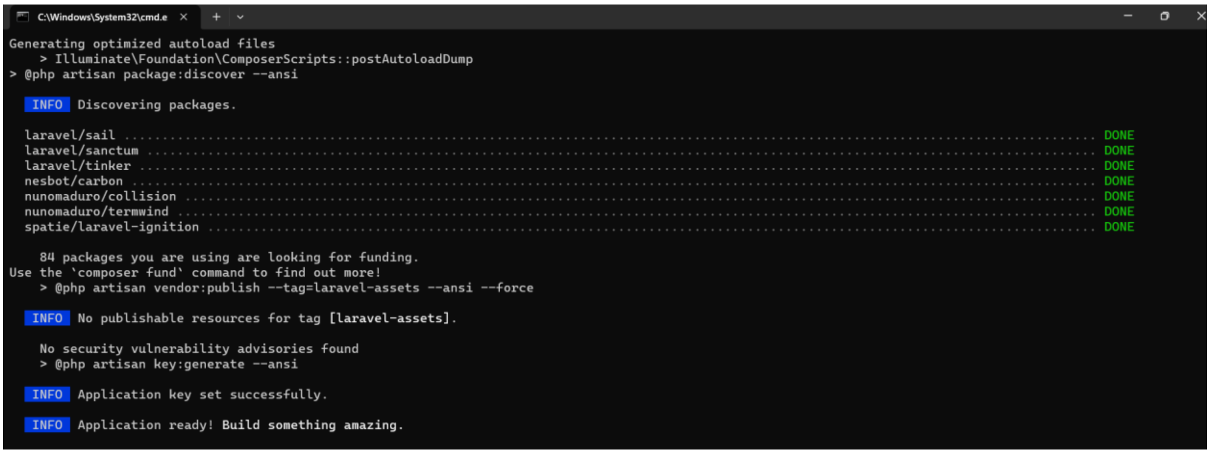
Name	Date modified	Type	Size
dashboard	10-04-2023 21:54	File folder	
drishti	19-05-2023 14:40	File folder	
img	10-04-2023 21:54	File folder	
mongophp	16-05-2023 20:08	File folder	
PHP Project	17-05-2023 22:01	File folder	
phppractise	18-05-2023 10:58	File folder	
Test	18-05-2023 13:34	File folder	
webalizer	10-04-2023 21:54	File folder	
xampp	10-04-2023 21:54	File folder	
applications	15-06-2022 21:37	Chrome HTML Do...	4 KB
bitnami	15-06-2022 21:37	CSS Source File	1 KB
favicon	16-07-2015 21:02	ICO File	31 KB

Now we will create a new project in Laravel.

laravel new Drishti

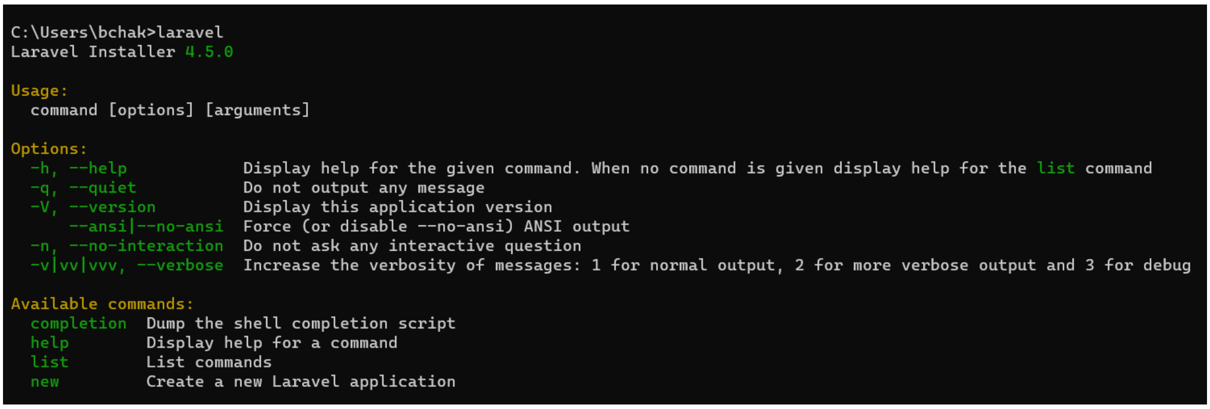


```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\xampp\htdocs>laravel new drishti

   _                               _
  | |                             | |
  | |     __ _ _ __ __ ___   _____| |
  | |    / _` | '__/ _` \ \ / / _ \ |
  | |___| (_| | | | (_| |\ V /  __/ |
  |______\__,_|_|  \__,_| \_/ \___|_|

    Creating a "laravel/laravel" project at "./drishti"
    Info from https://repo.packagist.org: #StandWithUkraine
    Cannot use laravel/laravel's latest version v10.2.1 as it requires php ^8.1 which is not satisfied by your platform.
Installing laravel/laravel (v9.5.2)
  - Downloading laravel/laravel (v9.5.2)
    - Installing laravel/laravel (v9.5.2): Extracting archive
    Created project in C:\xampp\htdocs/drishti
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
    Loading composer repositories with package information
    Updating dependencies
    Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking brick/math (0.11.0)
  - Locking dflydev/dot-access-data (v3.0.2)
  - Locking doctrine/deprecations (v1.0.0)
  - Locking doctrine/inflector (2.0.6)
  - Locking doctrine/instantiator (1.5.0)
  - Locking doctrine/lexer (2.1.0)
  - Locking dragonmantank/cron-expression (v3.3.2)
  - Locking egulias/email-validator (3.2.5)
  - Locking fakerphp/faker (v1.22.0)
  - Locking filp/whoops (2.15.2)
  - Locking fruitcake/php-cors (v1.2.0)
  - Locking graham-campbell/result-type (v1.1.1)
  - Locking guzzlehttp/guzzle (7.6.1)
  - Locking guzzlehttp/promises (1.5.2)
  - Locking guzzlehttp/psr7 (2.5.0)
  - Locking guzzlehttp/uri-template (v1.0.1)
  - Locking hamcrest/hamcrest-php (v2.0.1)
```



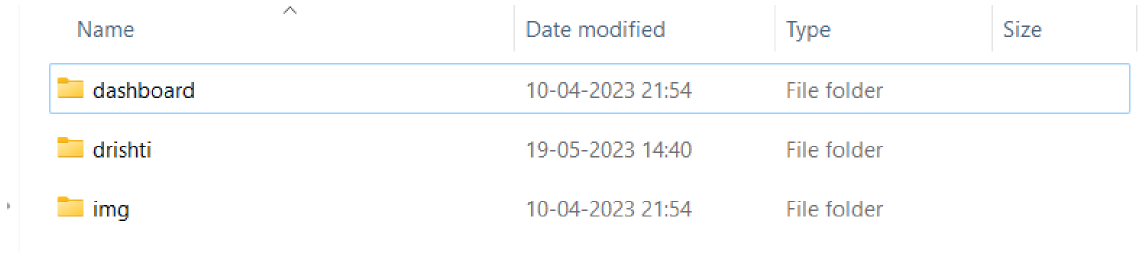
```
Generating optimized autoload files
    > Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

  laravel/sail ............................................................................................. DONE
  laravel/sanctum .......................................................................................... DONE
  laravel/tinker ........................................................................................... DONE
  nesbot/carbon ............................................................................................ DONE
  nunomaduro/collision ..................................................................................... DONE
  nunomaduro/termwind ...................................................................................... DONE
  spatie/laravel-ignition .................................................................................. DONE

    84 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
    > @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

    No security vulnerability advisories found
    > @php artisan key:generate --ansi

  INFO  Application key set successfully.

  INFO  Application ready! Build something amazing.
```

Now write **Laravel** in cmd to see the Laravel version.



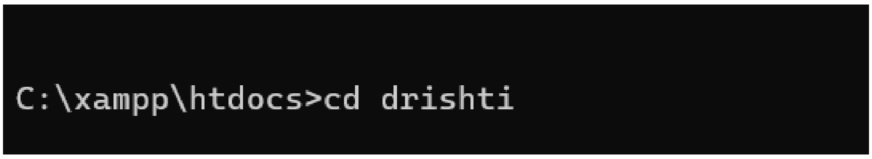
```
C:\Users\bchak>laravel
Laravel Installer 4.5.0

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display help for the given command. When no command is given display help for the list command
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi|--no-ansi  Force (or disable --no-ansi) ANSI output
  -n, --no-interaction  Do not ask any interactive question
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  completion  Dump the shell completion script
  help        Display help for a command
  list        List commands
  new         Create a new Laravel application
```

Now you will see that your folder **“drishti”** has been successfully created.



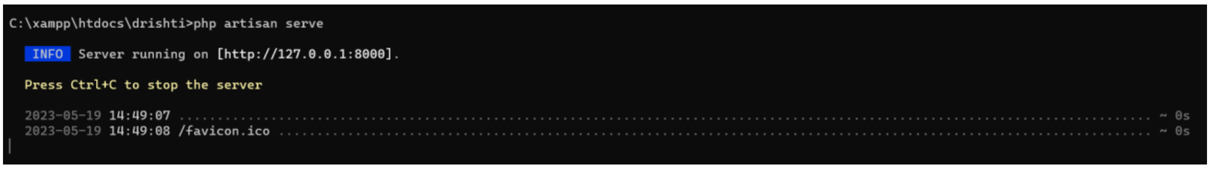
Name	Date modified	Type	Size
dashboard	10-04-2023 21:54	File folder	
drishti	19-05-2023 14:40	File folder	
img	10-04-2023 21:54	File folder	

Now switch to **cd drishti**



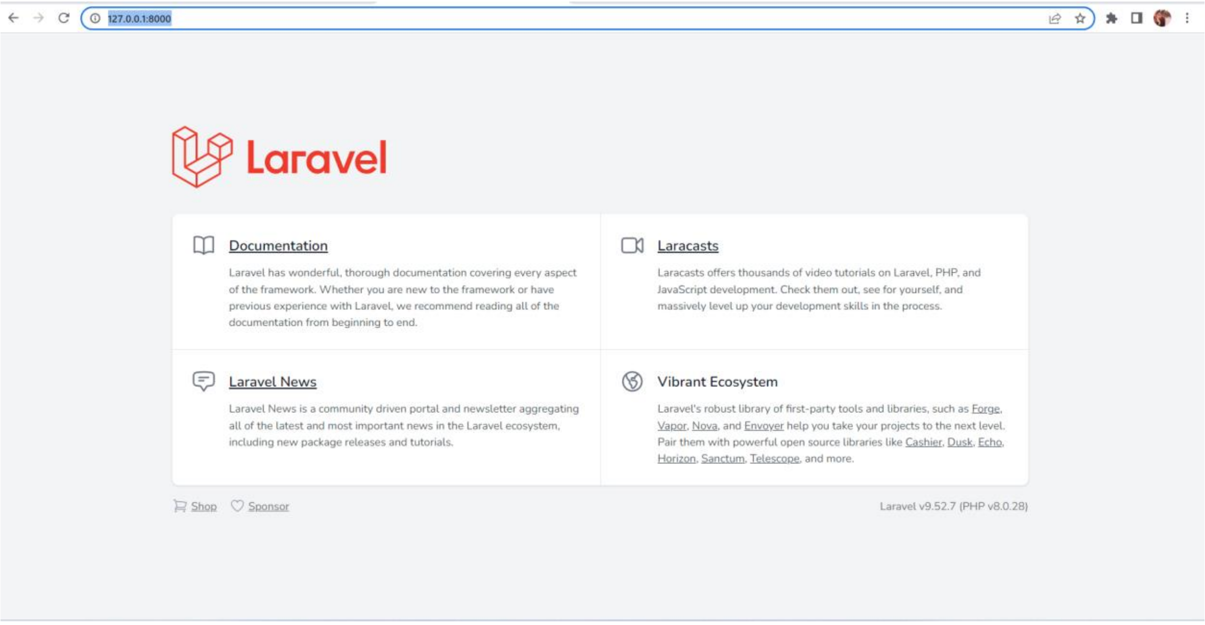
```
C:\xampp\htdocs>cd drishti
```

Now type :- **php artisan serve**



```
C:\xampp\htdocs\drishti>php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server

  2023-05-19 14:49:07 .......................................................................................... ~ 0s
  2023-05-19 14:49:08 /favicon.ico ............................................................................. ~ 0s
```

Now copy the given **url** and paste it on your **browser.**



127.0.0.1:8000

Laravel

Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

Shop Sponsor

Laravel v9.52.7 (PHP v8.0.28)

Laravel has been successfully install

Blade Conditional Directives

[@if, @elseif @else and @endif directives]

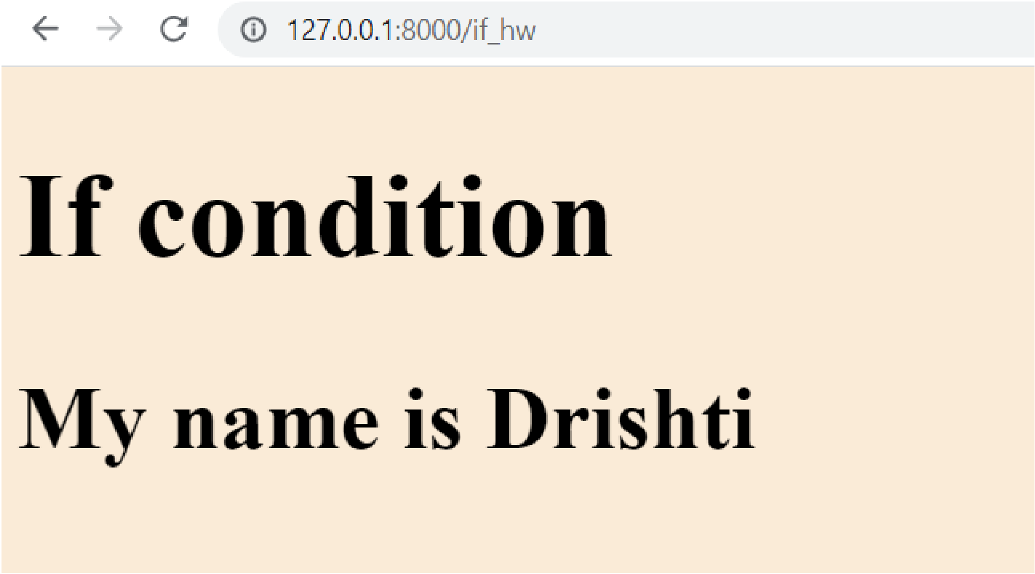
1) @if and @endif directives

Step 1: Create if.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>if condition</title>
</head>
<body>
    <h1>If condition</h1>
    @if($name=="Drishti")
    <h2>My name is {{$name}}</h2>
    @endif
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/if_hw’, function (){
return view('if_hw',["name"=>"Drishti"]);
});
```



If condition

My name is Drishti

2) @if, @else and @endif directives

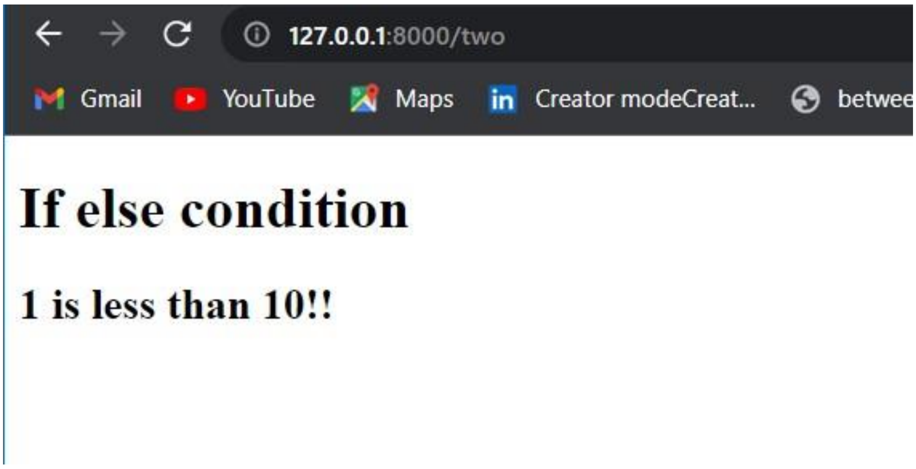
Step 1: Create ifelse.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>If else</title>
</head>
<body>
<h1>If else condition</h1>

    @if($a <= 10)
    <h2> {{$a}} is less than 10!! </h2>
    @else
    <h2> {{$a}} is greater than 10!! </h2>
    @endif
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/two', function (){

    return view('ifelse',["a"=>1]); });
```

Step 3: Output



If else condition

1 is less than 10!!

3) @if, @elseif, @else and @endif directives

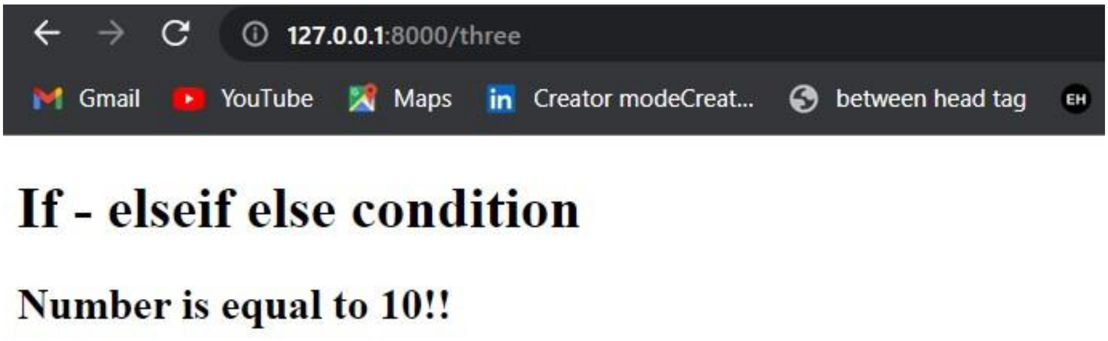
Step 1: Create ifelseif.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>If-elseif else</title>
</head>
<body>
<h1>If - elseif else condition</h1>

@if ($b < 10)
<h2>Number is less than 10!!</h2>
@elseif($b ==10)
<h2>Number is equal to 10!!</h2>
@else
<h2>Number is greater than 10!!</h2>
@endif
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/three', function(){
    return view('ifelseif',["b"=>10]);
});
```

Step 3: Output



If - elseif else condition

Number is equal to 10!!

ACTIVITY 2

Blade Looping Directives

[@for, @while @foreach directives]

1) @for and @endfor directives

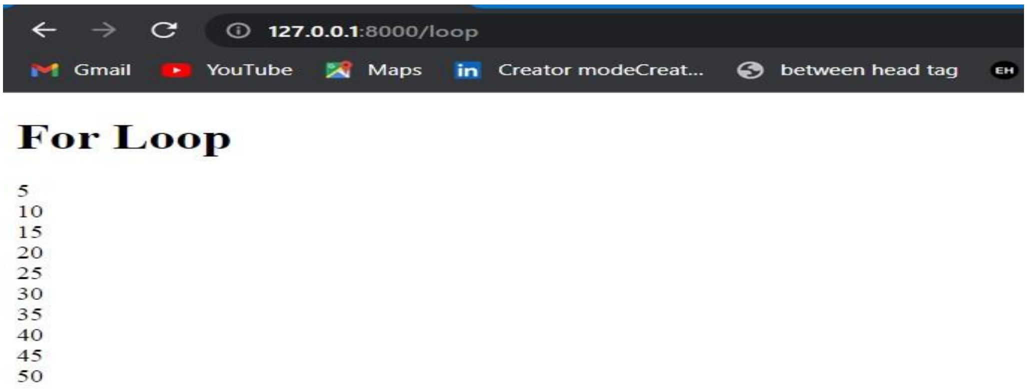
Step 1: Create for.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>FOR LOOP</title>
</head>
<body>
   <h1>For Loop</h1>
    @for($i=5;$i<=50;$i=$i+5)
    {{$i}}
    <br>
    @endfor
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/loop', function () {
    return view('for'); });
```

Step 3: Output



For Loop

```
5
10
15
20
25
30
35
40
45
50
```

2) @while and @endwhile directives

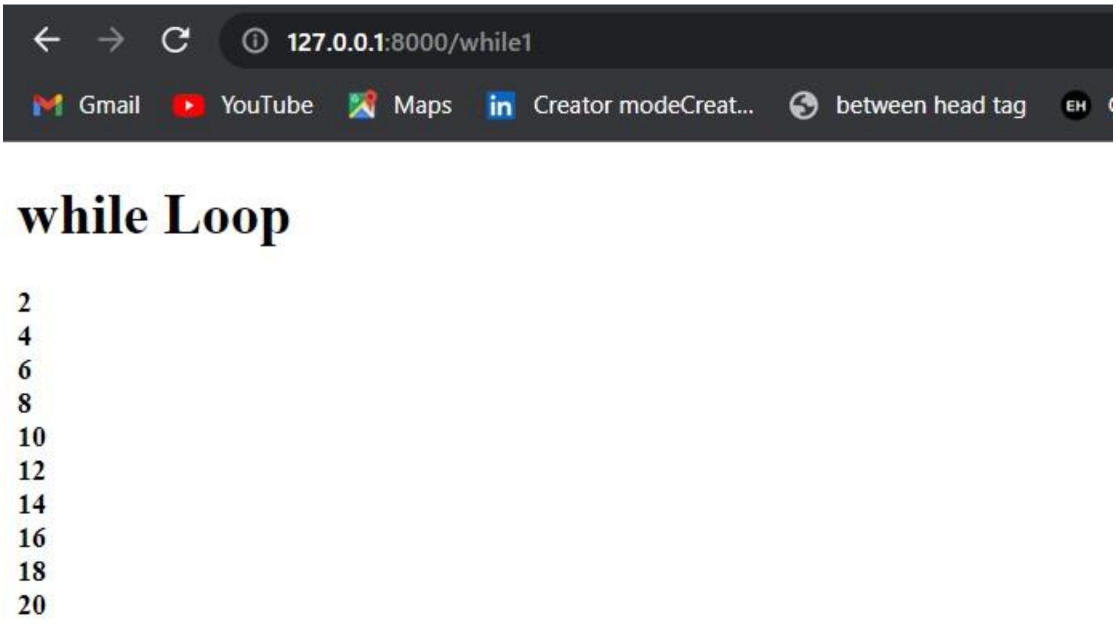
Step 1: Create while.blade.php file

```
<!DOCTYPE html>
<html>
<head>
<title>While Loop </title>
</head>
<body>
  <h1> while Loop</h1>
@while ($i <= 18)
<b>{{$i+=2}}</b>
<br>
@endwhile
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/while1', function(){
    return view('while',["i"=>0]); });
```

Step 3: Output



while Loop

```
2
4
6
8
10
12
14
16
18
20
```

3) @foreach and @endforeach directives tep 1:

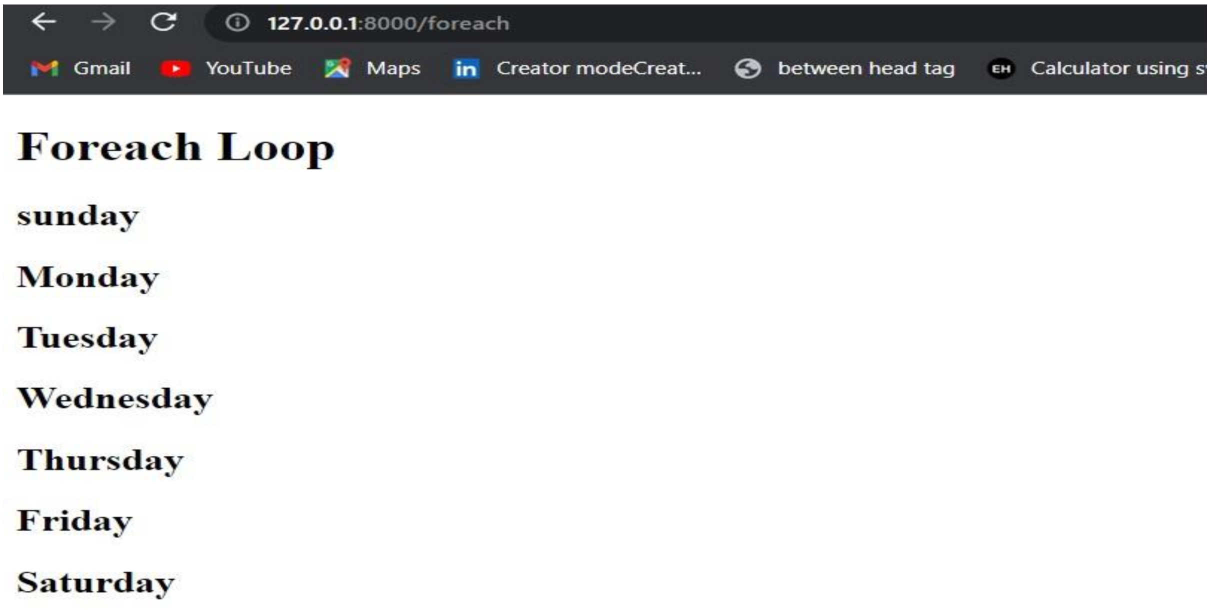
Create foreach.blade.php file

```
<!DOCTYPE html>
<html>
<head>
<title> Foreach Loop </title>
</head>
<body>
  <h1> Foreach Loop</h1>
@foreach ($arr as $i)
<h2> {{$i}} </h2>
@endforeach
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/foreach', function(){
    return
view('foreach',["arr"=>['sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday']]);
});
```

Step 3: Output



Foreach Loop

sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

ACTIVITY 3

[@break , @continue, Comments and @php and @endphp Directives]

1) @break directives

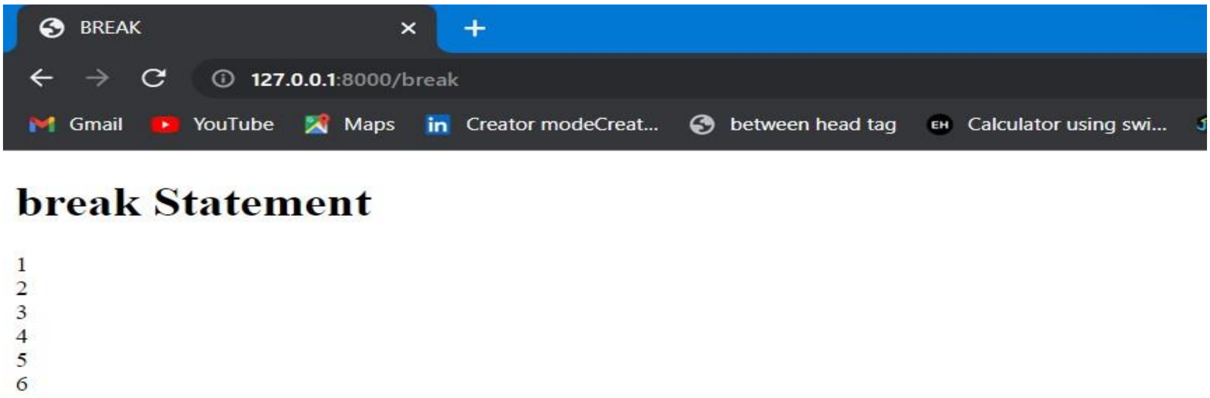
Step 1: Create break.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>BREAK</title>
</head>
<body>
   <h1>break Statement </h1>
@for($i=1;$i<=10;$i++)
    @if($i==7) @break @endif
    {{$i}}
    <br>
    @endfor
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/break',function(){
    return view('break');
});
```

Step 3: Output



```
break Statement

1
2
3
4
5
6
```

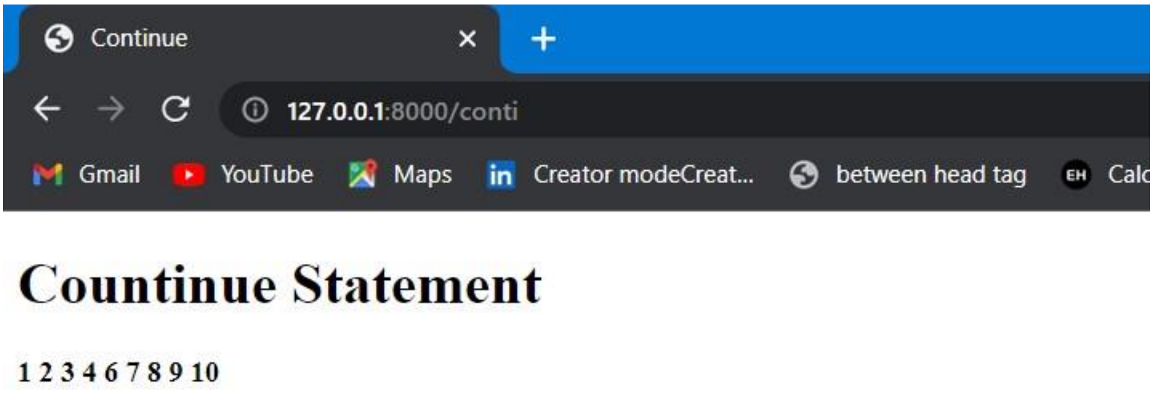
2) @continue directives tep 1: Create continue.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>Continue</title>
</head>
<body>
   <h1>Countinue Statement </h1>
    @for($i=1;$i<=10;$i++)
    @if($i==5) @continue @endif
   <b> {{$i}}</b>
    @endfor
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/conti',function(){
    return view('continue'); });
```

Step 3: Output



Countinue Statement

1 2 3 4 6 7 8 9 10

3) Comments in Blade Directives

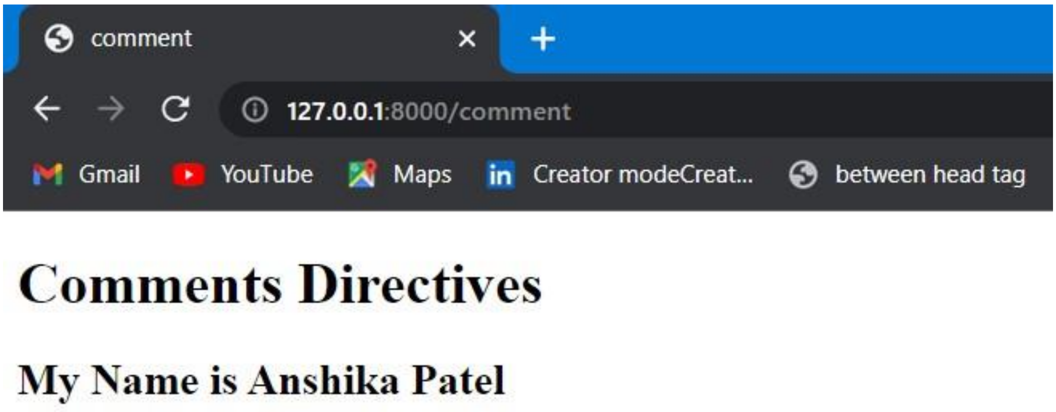
Step 1: Create comments.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>comment</title>
</head>
<body>
   <h1>Comments Directives</h1>

   {{--Print the normal string--}}
    <h2>My Name is

    {{--call the variable--}}
         {{$name}} </h2>
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/comment',function(){
    return view('comments',['name'=>'Anshika Patel']);
});
```

Step 3: Output



Comments Directives

My Name is Anshika Patel

4) **@php and @endphp directives** tep 1:

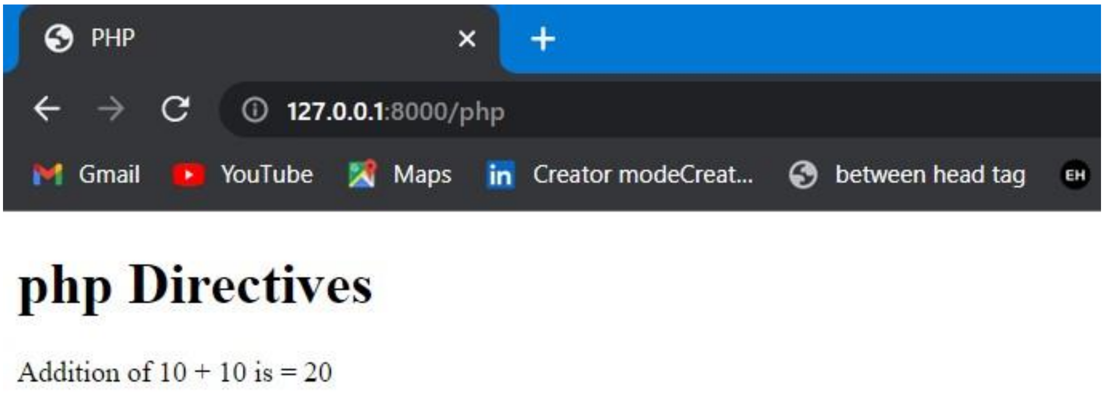
Create php.blade.php file

```
<!DOCTYPE html>
<html>
<head>
    <title>PHP</title>
</head>
<body>
   <h1>php Directives</h1>


   @php
   $a=10;
   $b=10;
   echo "Addition of $a + $b is = ". ($a+$b);
   @endphp
</body>
</html>
```

Step 2: Define the Route in web.php file

```
Route::get('/php',function(){
    return view('php');
});
```

Step 3: Output



php Directives

Addition of 10 + 10 is = 20

Controllers

Basic Controller

How we can create it:

```
Php artisan make:controller BasicController
```

How we can give the route in web.php:

```
Route::get(‘/name’,[BasicController::class,’Index’]);
```

Single Action Controller

How we can create it:

```
Php artisan make:controller SingleController  --
invokable
```

How we can give the route in web.php:

```
Route::get(‘/name’,SingleController::class);
```

Resource Controller

How we can create it:

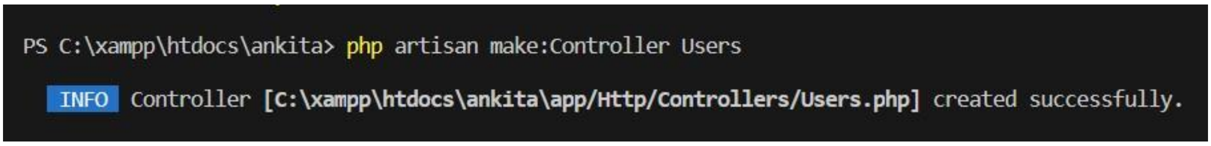
```
Php artisan make:controller ResourceController --r
```

How we can give the route in web.php:

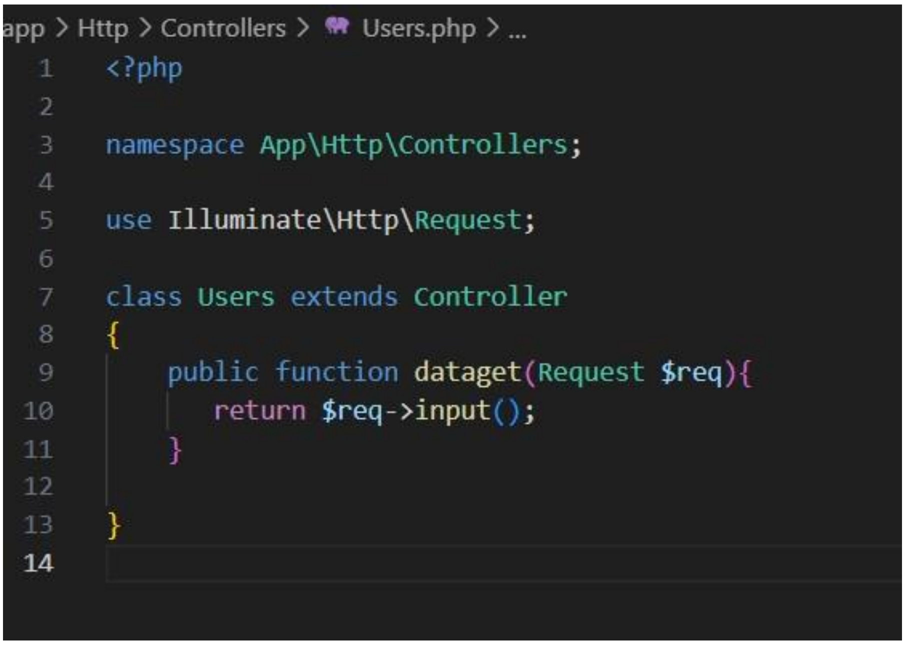
```
Route::resource(‘/Resource’,ResourceController::class);
```

How to fetch data from the form using POST method

<u>Step 1:-</u> create a controller:-

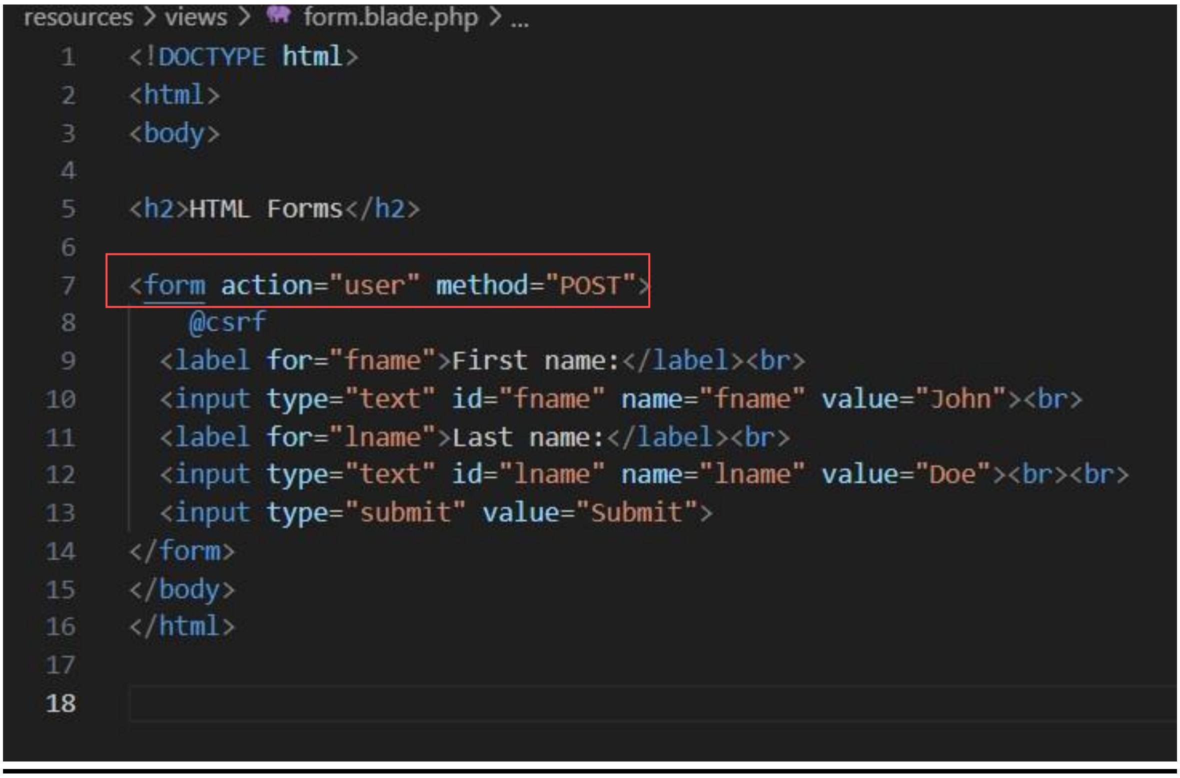


```
PS C:\xampp\htdocs\ankita> php artisan make:Controller Users

  INFO  Controller [C:\xampp\htdocs\ankita\app/Http/Controllers/Users.php] created successfully.
```



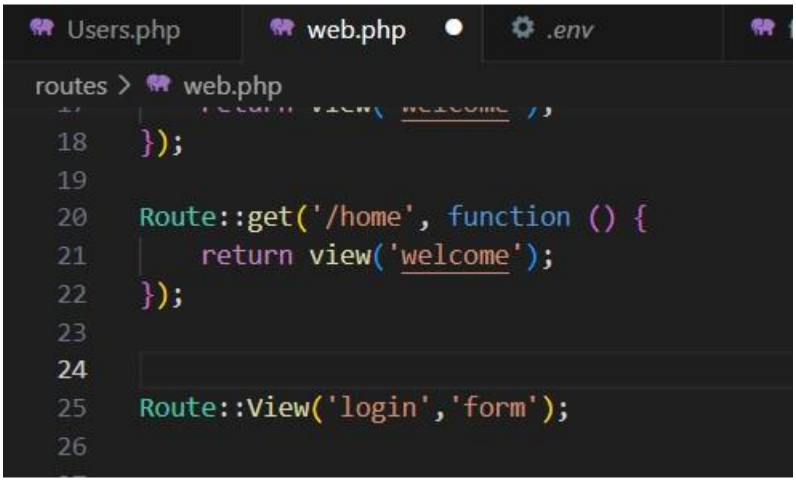
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Users extends Controller
{
    public function dataget(Request $req){
       return $req->input();
    }

}
```

<u>Step 2:-</u> Create a form using the POST method in view floder using form.blade.php and using the @csrf Hidden token



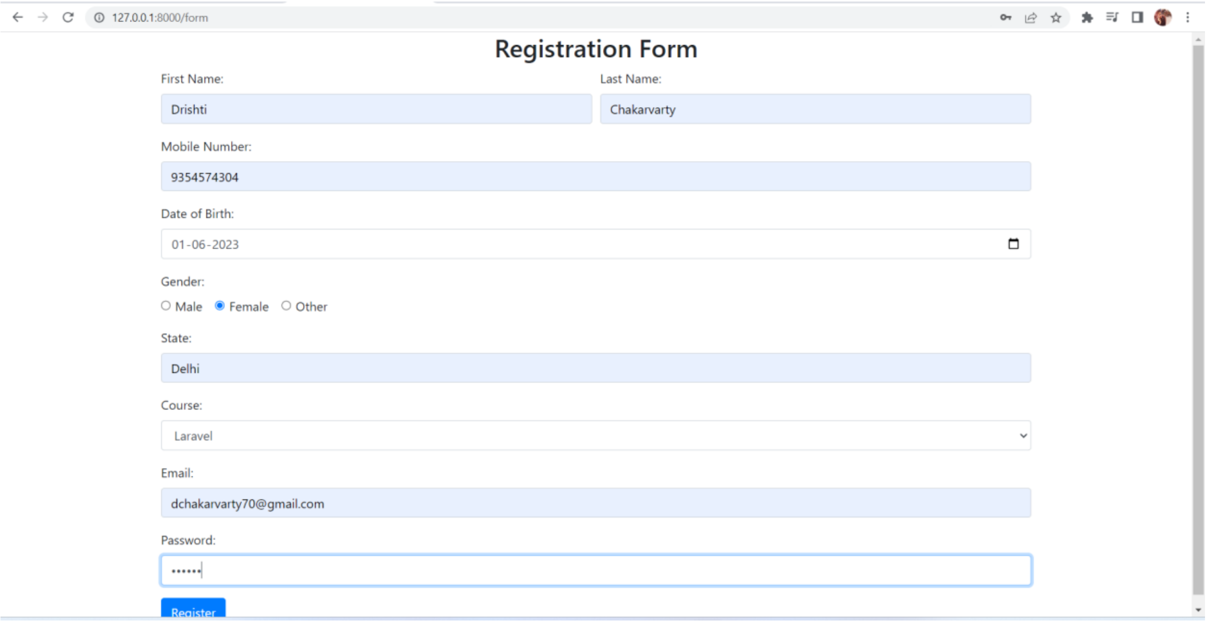
```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="user" method="POST">
    @csrf
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>

```

Step 3:- create routes for access the form file :-



```
    return view('welcome');
});

Route::get('/home', function () {
    return view('welcome');
});


Route::View('login','form');

```

Step 4:- now check the form in browser type login and then see:-

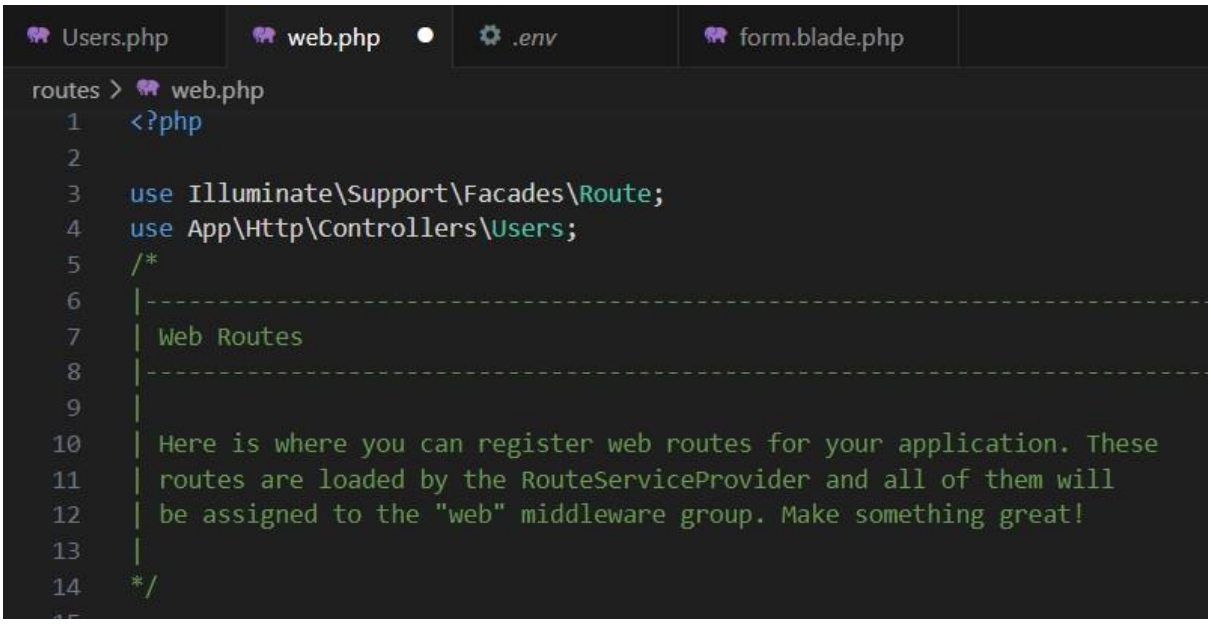


Registration Form

First Name:	Drishti
Last Name:	Chakarvarty
Mobile Number:	9354574304
Date of Birth:	01-06-2023
Gender:	○ Male ● Female ○ Other
State:	Delhi
Course:	Laravel
Email:	dchakarvarty70@gmail.com
Password:	••••••

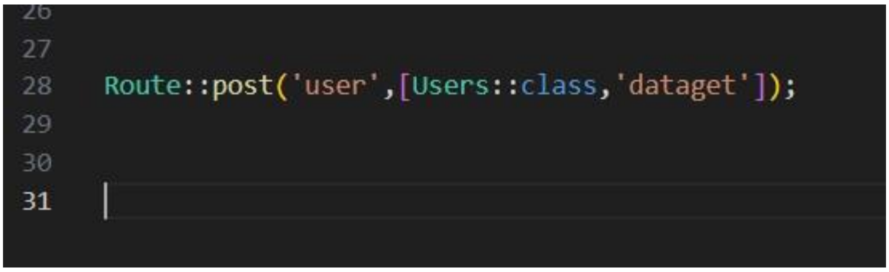
Step 5:- Create a Route for accessing the enter data in form

Add the controller namespace in web.php



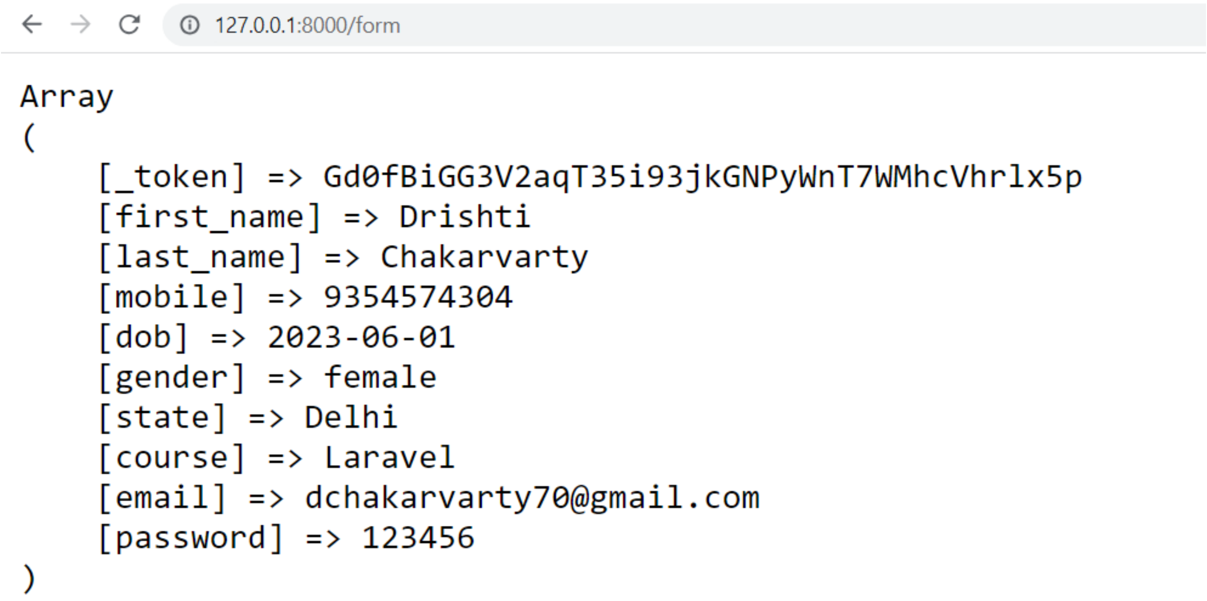
```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Users;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/
```

Step-6:- Create a route with post and use url path and define controller class



```
Route::post('user',[Users::class,'dataget']);
```

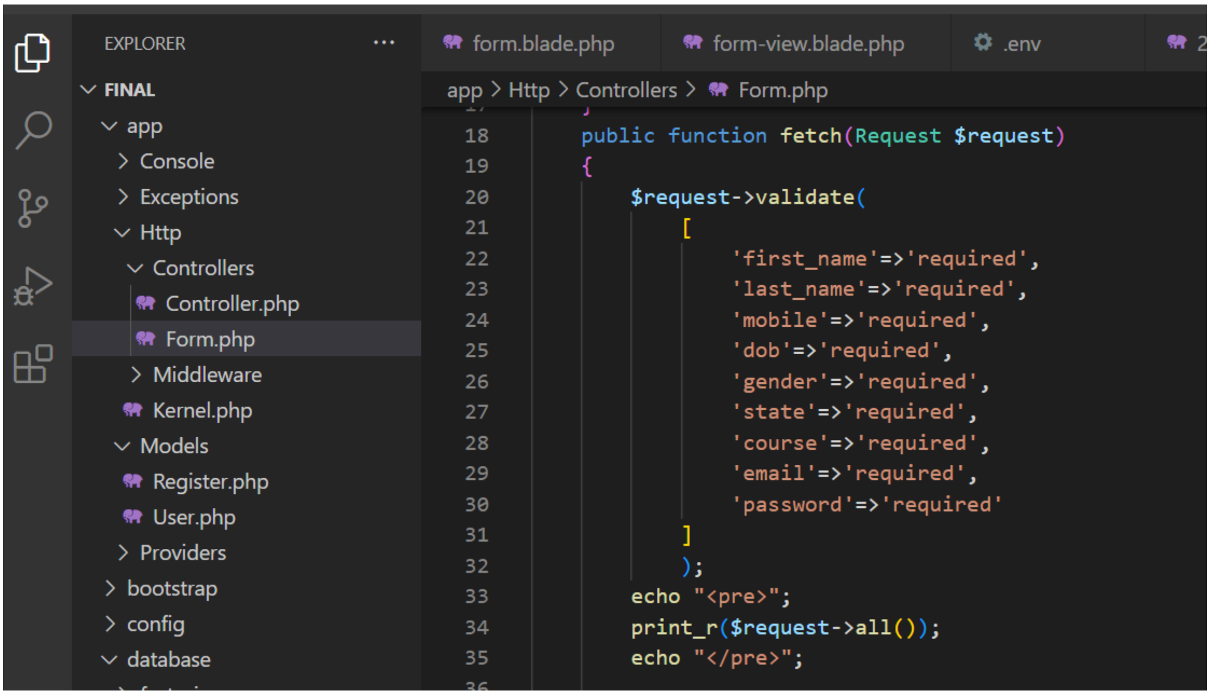
And final you see the output



```
Array
(
    [_token] => Gd0fBiGG3V2aqT35i93jkGNPyWnT7WMhcVhrlx5p
    [first_name] => Drishti
    [last_name] => Chakarvarty
    [mobile] => 9354574304
    [dob] => 2023-06-01
    [gender] => female
    [state] => Delhi
    [course] => Laravel
    [email] => dchakarvarty70@gmail.com
    [password] => 123456
)
```

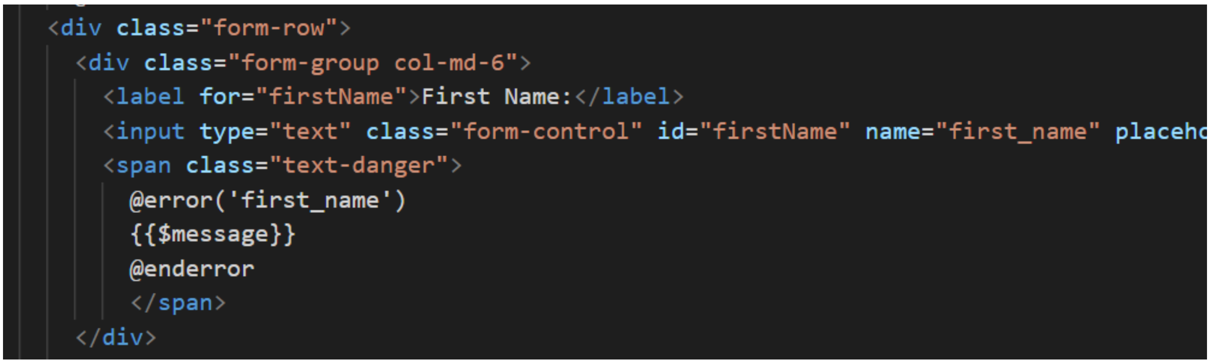
Apply Form Validation

Step-1 Go to the controller file where you will see public function that is already created to request there you have to apply validation.



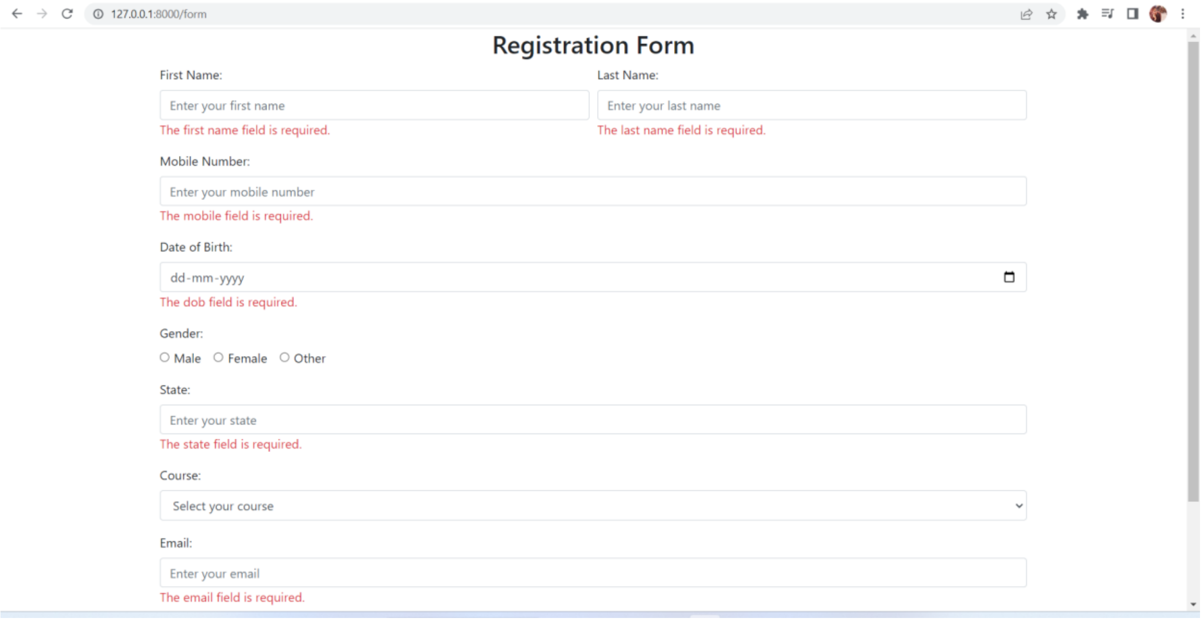
```
    public function fetch(Request $request)
    {
        $request->validate(
            [
                'first_name'=>'required',
                'last_name'=>'required',
                'mobile'=>'required',
                'dob'=>'required',
                'gender'=>'required',
                'state'=>'required',
                'course'=>'required',
                'email'=>'required',
                'password'=>'required'
            ]
            );
        echo "<pre>";
        print_r($request->all());
        echo "</pre>";
```

Step-2 Now go to the form page and put the span tag after every input to display the error msg(Validation).



```
<div class="form-row">
  <div class="form-group col-md-6">
    <label for="firstName">First Name:</label>
    <input type="text" class="form-control" id="firstName" name="first_name" placeho
    <span class="text-danger">
      @error('first_name')
      {{$message}}
      @enderror
      </span>
  </div>
```

Step-3 Now you will see the Validation has been applied.



127.0.0.1:8000/form

Registration Form

First Name:

Enter your first name

The first name field is required.

Last Name:

Enter your last name

The last name field is required.

Mobile Number:

Enter your mobile number

The mobile field is required.

Date of Birth:

dd-mm-yyyy

The dob field is required.

Gender:

○ Male ○ Female ○ Other

State:

Enter your state

The state field is required.

Course:

Select your course

Email:

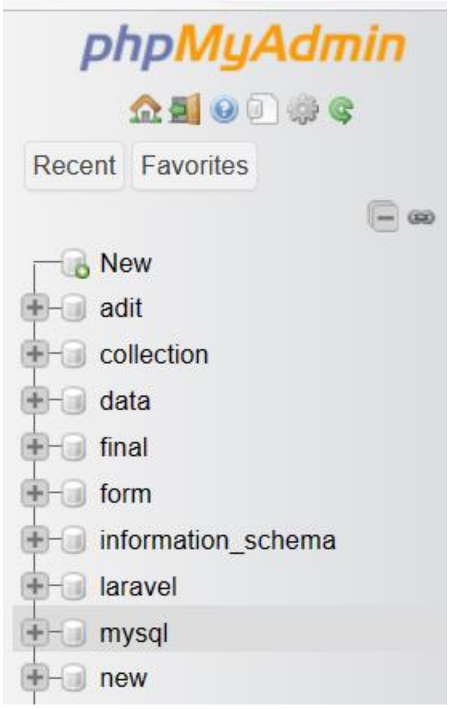
Enter your email

The email field is required.

Insert Query

Step-1: Take the Registration form that is already created.

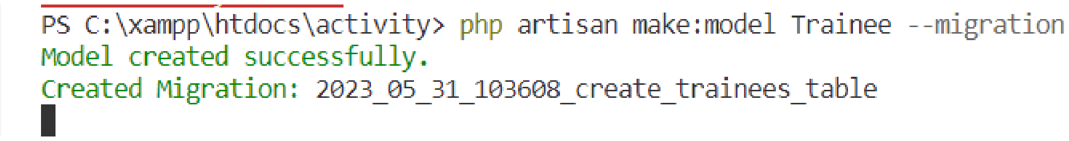
Step 2: Create a database on PHP MyAdmin



<u>Step 3:</u> Create a model and migration

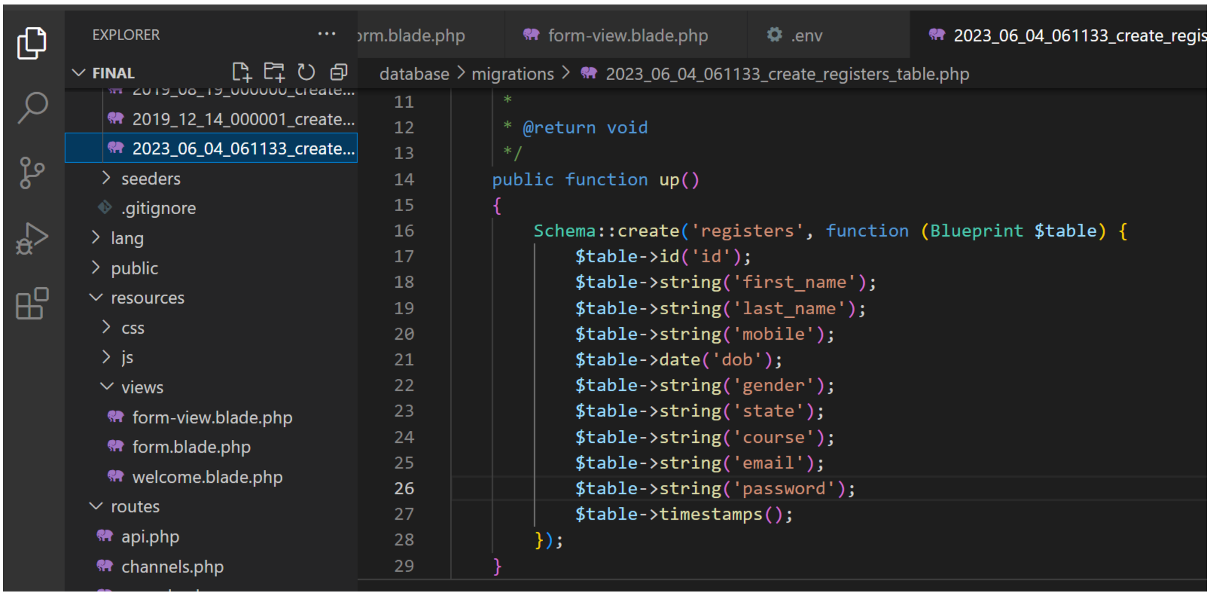
Command:

php artisan make:model Model_Name –migration



```
PS C:\xampp\htdocs\activity> php artisan make:model Trainee --migration
Model created successfully.
Created Migration: 2023_05_31_103608_create_trainees_table
```

<u>Step 4:</u> Add field to the created table(migration)

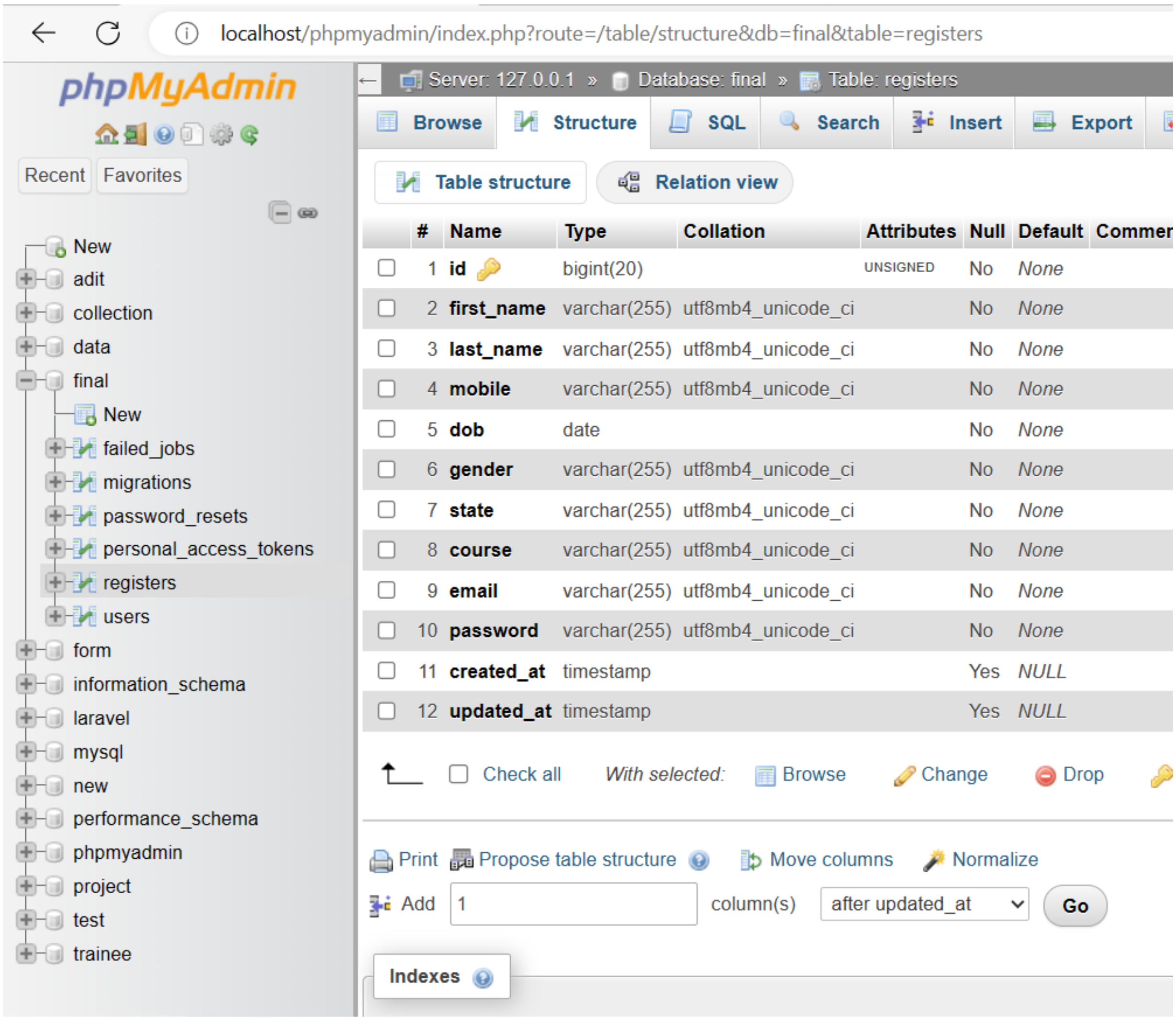


EXPLORER
FINAL
2019_08_19_000000_create...
2019_12_14_000001_create...
2023_06_04_061133_create...
seeders
.gitignore
lang
public
resources
css
js
views
form-view.blade.php
form.blade.php
welcome.blade.php
routes
api.php
channels.php
orm.blade.php
form-view.blade.php
.env
2023_06_04_061133_create_regis
database > migrations > 2023_06_04_061133_create_registers_table.php

```
     *
     * @return void
     */
    public function up()
    {
        Schema::create('registers', function (Blueprint $table) {
            $table->id('id');
            $table->string('first_name');
            $table->string('last_name');
            $table->string('mobile');
            $table->date('dob');
            $table->string('gender');
            $table->string('state');
            $table->string('course');
            $table->string('email');
            $table->string('password');
            $table->timestamps();
        });
    }
```

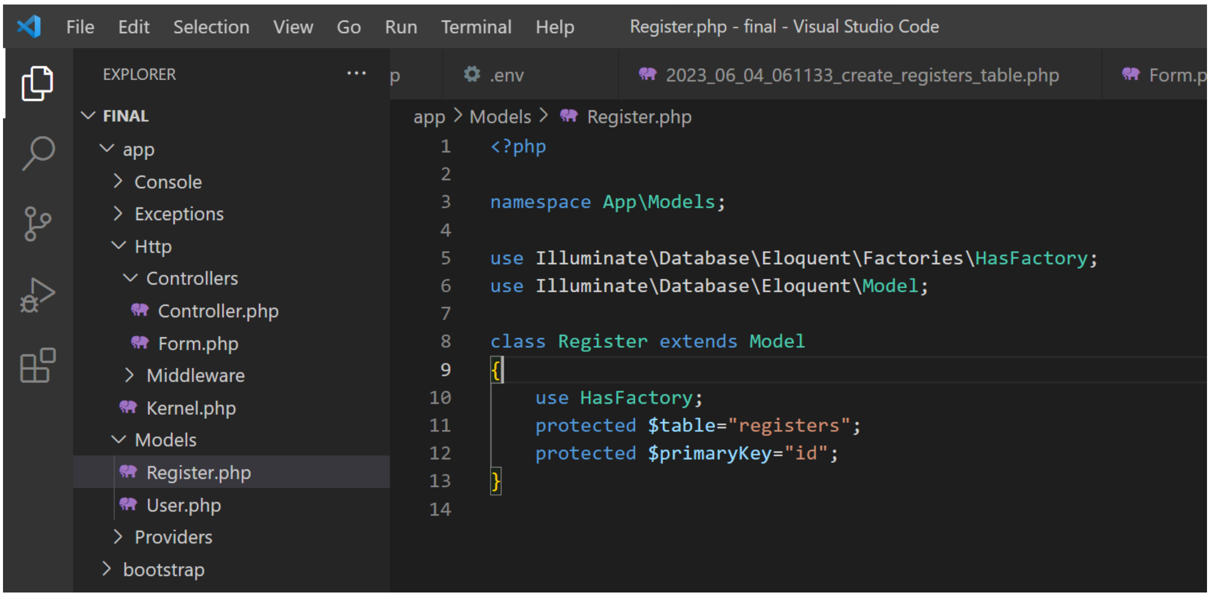
<u>Step 5:</u> Run command **php artisan migrate** to created table successfully

<u>Step 6:</u> go to PHP MyAdmin see the table created successfully



#	Name	Type	Collation	Attributes	Null	Default	Commer
1	id	bigint(20)		UNSIGNED	No	None	
2	first_name	varchar(255)	utf8mb4_unicode_ci		No	None	
3	last_name	varchar(255)	utf8mb4_unicode_ci		No	None	
4	mobile	varchar(255)	utf8mb4_unicode_ci		No	None	
5	dob	date			No	None	
6	gender	varchar(255)	utf8mb4_unicode_ci		No	None	
7	state	varchar(255)	utf8mb4_unicode_ci		No	None	
8	course	varchar(255)	utf8mb4_unicode_ci		No	None	
9	email	varchar(255)	utf8mb4_unicode_ci		No	None	
10	password	varchar(255)	utf8mb4_unicode_ci		No	None	
11	created_at	timestamp			Yes	NULL	
12	updated_at	timestamp			Yes	NULL	

Step 7: Go to Models folder then open model(Register.php). then define table name and primary Key

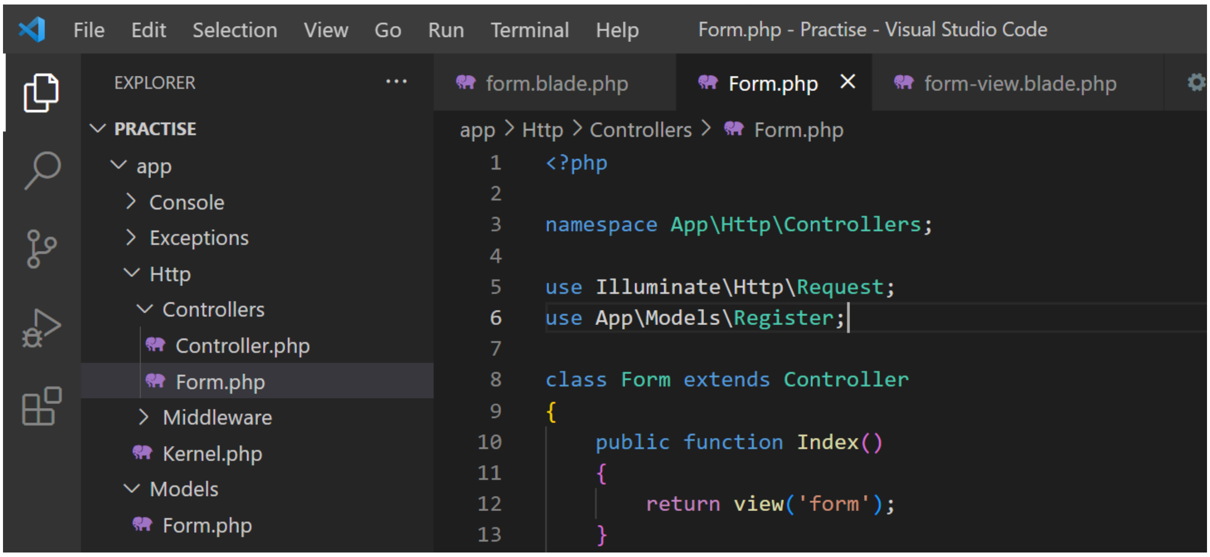


```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Register extends Model
{
    use HasFactory;
    protected $table="registers";
    protected $primaryKey="id";
}
```

Step 8: Now we need to include Model or we can say we use the model in controller file.

Command

App\Models\Register;



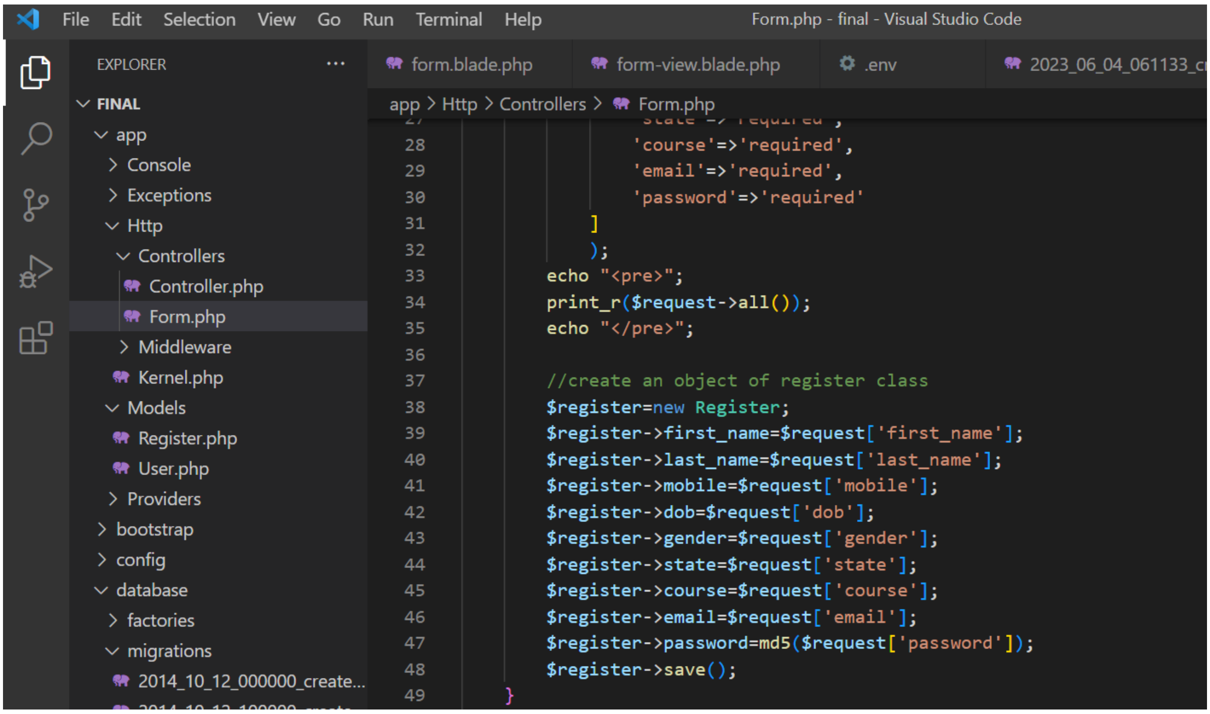
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Register;

class Form extends Controller
{
    public function Index()
    {
        return view('form');
    }
```

Step 9: then create an object of the class(Register) after that access the table field through array operator :

Create a variable then access the class

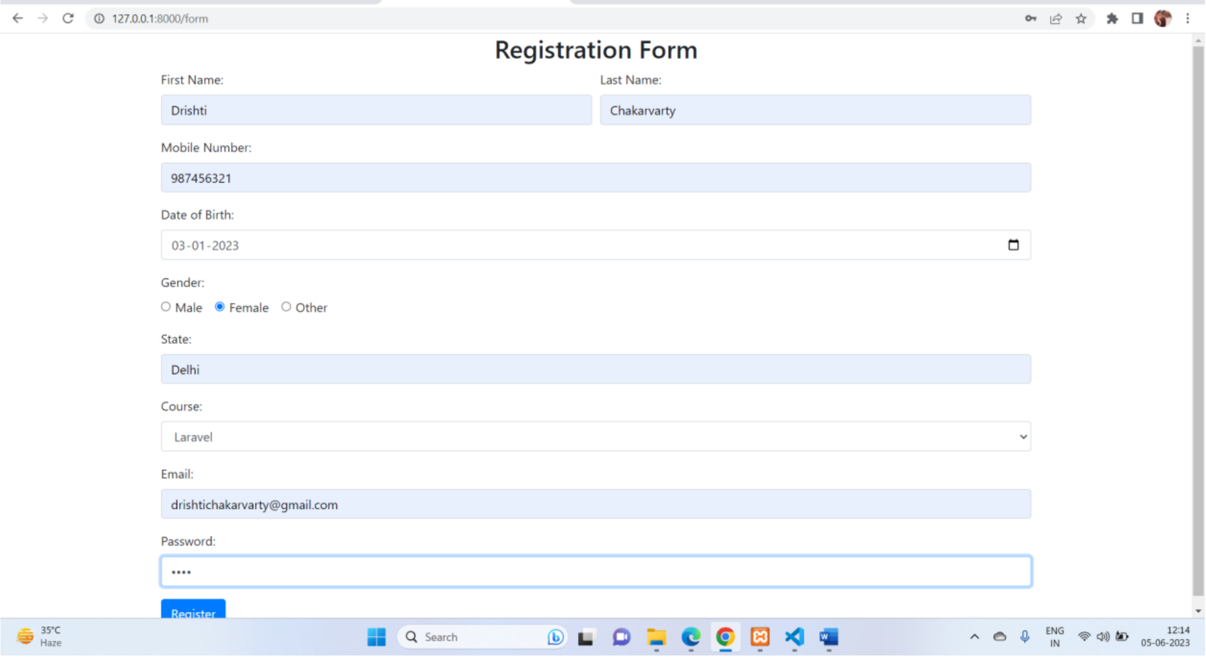
```
$register= new Register;
```

Then access the field through object



```php
                'state'=>'required',
                'course'=>'required',
                'email'=>'required',
                'password'=>'required'
            ]
            );
        echo "<pre>";
        print_r($request->all());
        echo "</pre>";

        //create an object of register class
        $register=new Register;
        $register->first_name=$request['first_name'];
        $register->last_name=$request['last_name'];
        $register->mobile=$request['mobile'];
        $register->dob=$request['dob'];
        $register->gender=$request['gender'];
        $register->state=$request['state'];
        $register->course=$request['course'];
        $register->email=$request['email'];
        $register->password=md5($request['password']);
        $register->save();
    }
```

Step 10: Now open the registration form and fill the details then submit it



Registration Form

First Name: Drishti

Last Name: Chakarvarty

Mobile Number: 987456321

Date of Birth: 03-01-2023

Gender: ○ Male ● Female ○ Other

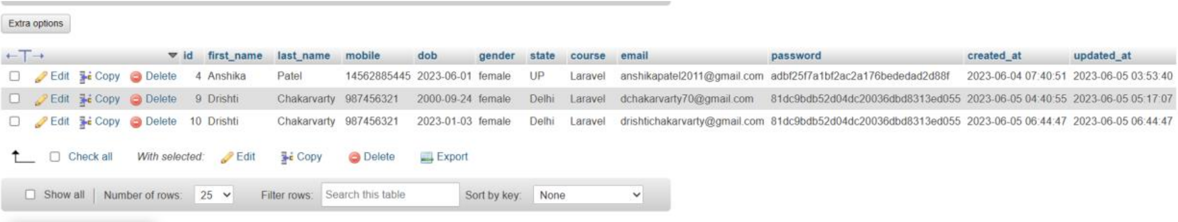
State: Delhi

Course: Laravel

Email: drishtichakarvarty@gmail.com

Password: ••••

Step 11: Go to PHP My Admin then click on browse to see the details are submitted.



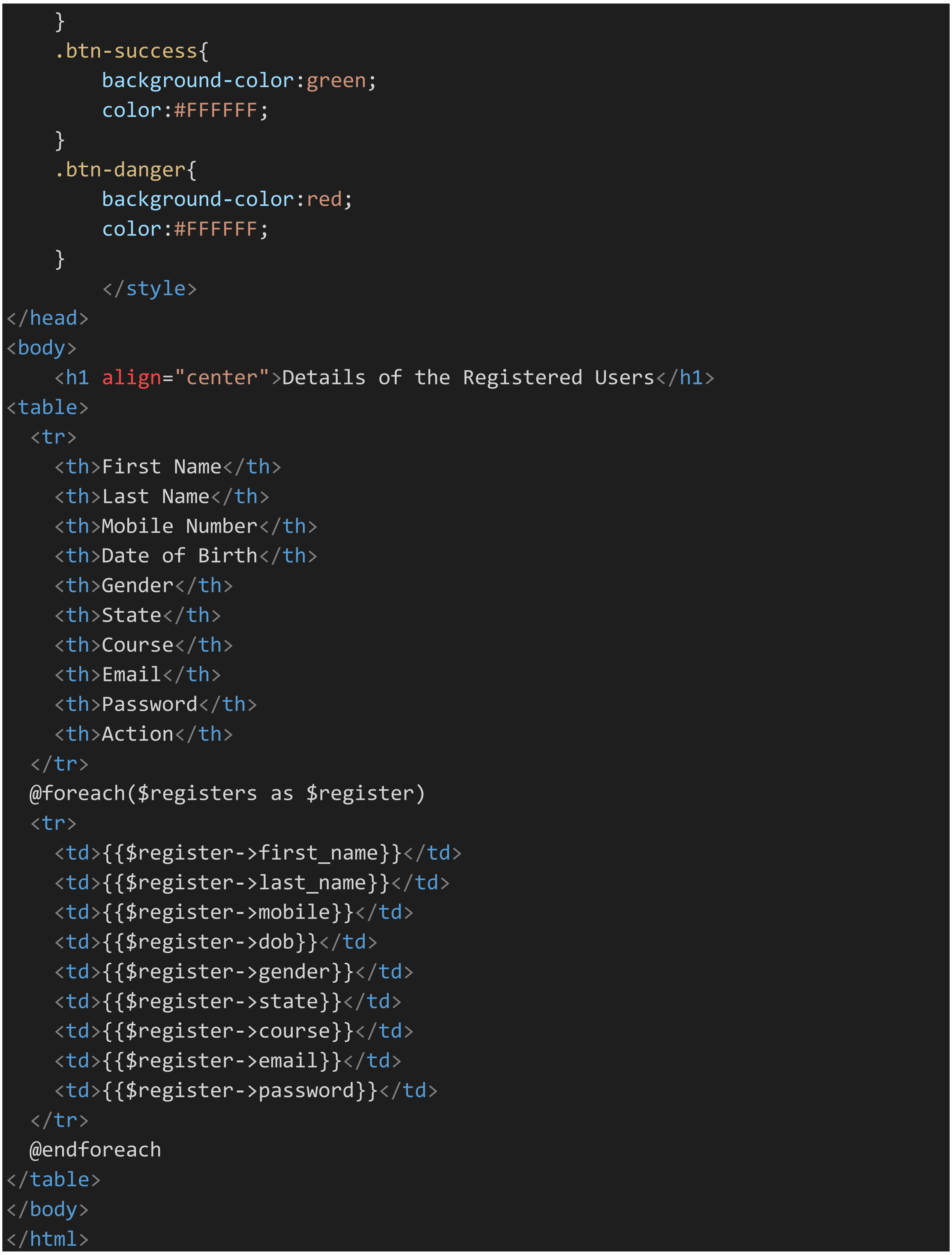
←T→			▼	id	first_name	last_name	mobile	dob	gender	state	course	email	password	created_at	updated_at
□	Edit	Copy	Delete	4	Anshika	Patel	14562885445	2023-06-01	female	UP	Laravel	anshikapatel2011@gmail.com	adbf25f7a1bf2ac2a176bededad2d88f	2023-06-04 07:40:51	2023-06-05 03:53:40
□	Edit	Copy	Delete	9	Drishti	Chakarvarty	987456321	2000-09-24	female	Delhi	Laravel	dchakarvarty70@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	2023-06-05 04:40:55	2023-06-05 05:17:07
□	Edit	Copy	Delete	10	Drishti	Chakarvarty	987456321	2023-01-03	female	Delhi	Laravel	drishtichakarvarty@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	2023-06-05 06:44:47	2023-06-05 06:44:47

Select Query

Step-1 Use the same Registration Form.

Step-2 Create a new file(form-view.blade.php) then create a table on this page.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Table</title>
    <style>
        table, th, td {
         border: 2px solid black;
            }
            .text-right {
      text-align: right;

    }
    .custom-button {
      background-color: #33E3FF;
      color: #FFFFFF;
    }
    .custom-button-lg {
      padding: 10px 20px;
      font-size: 1.2rem;
```

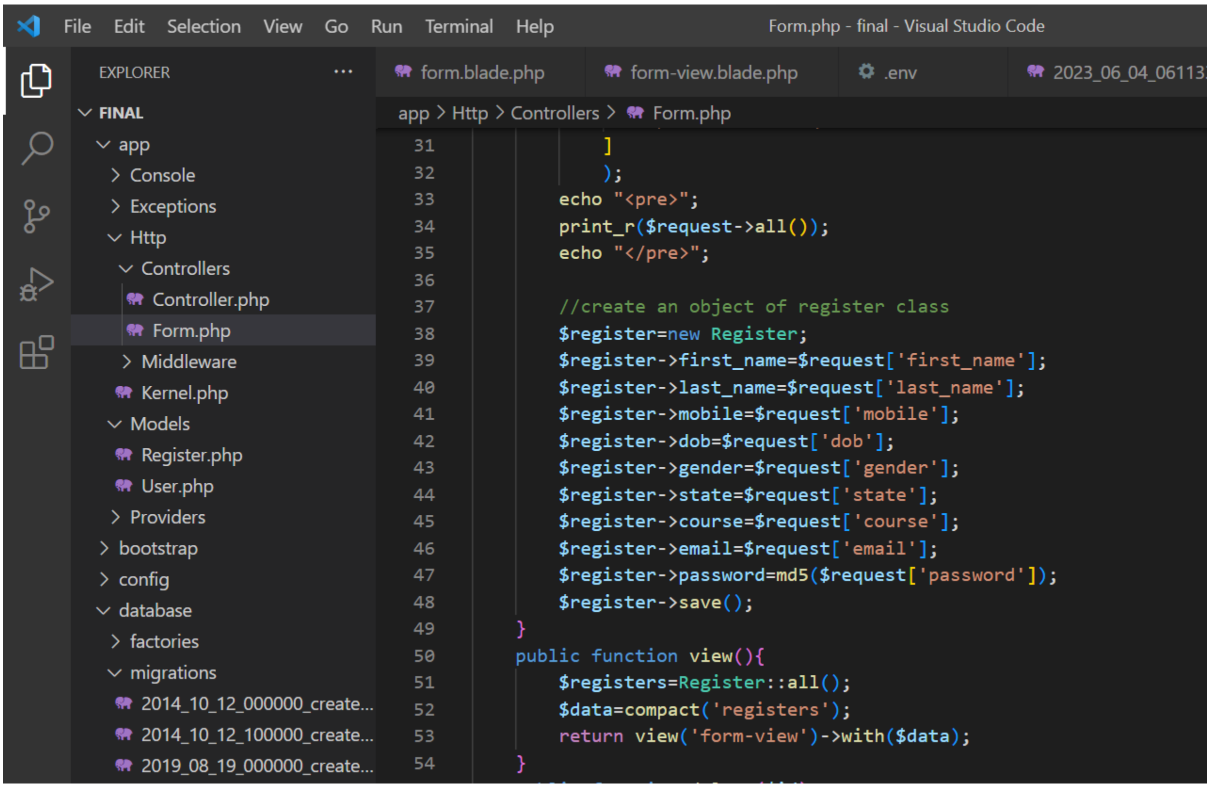


```
    }
    .btn-success{
        background-color:green;
        color:#FFFFFF;
    }
    .btn-danger{
        background-color:red;
        color:#FFFFFF;
    }
        </style>
</head>
<body>
    <h1 align="center">Details of the Registered Users</h1>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Mobile Number</th>
    <th>Date of Birth</th>
    <th>Gender</th>
    <th>State</th>
    <th>Course</th>
    <th>Email</th>
    <th>Password</th>
    <th>Action</th>
  </tr>
  @foreach($registers as $register)
  <tr>
    <td>{{$register->first_name}}</td>
    <td>{{$register->last_name}}</td>
    <td>{{$register->mobile}}</td>
    <td>{{$register->dob}}</td>
    <td>{{$register->gender}}</td>
    <td>{{$register->state}}</td>
    <td>{{$register->course}}</td>
    <td>{{$register->email}}</td>
    <td>{{$register->password}}</td>
  </tr>
  @endforeach
</table>
</body>
</html>
```

Step-3 Now go to the web.php file and give the route.

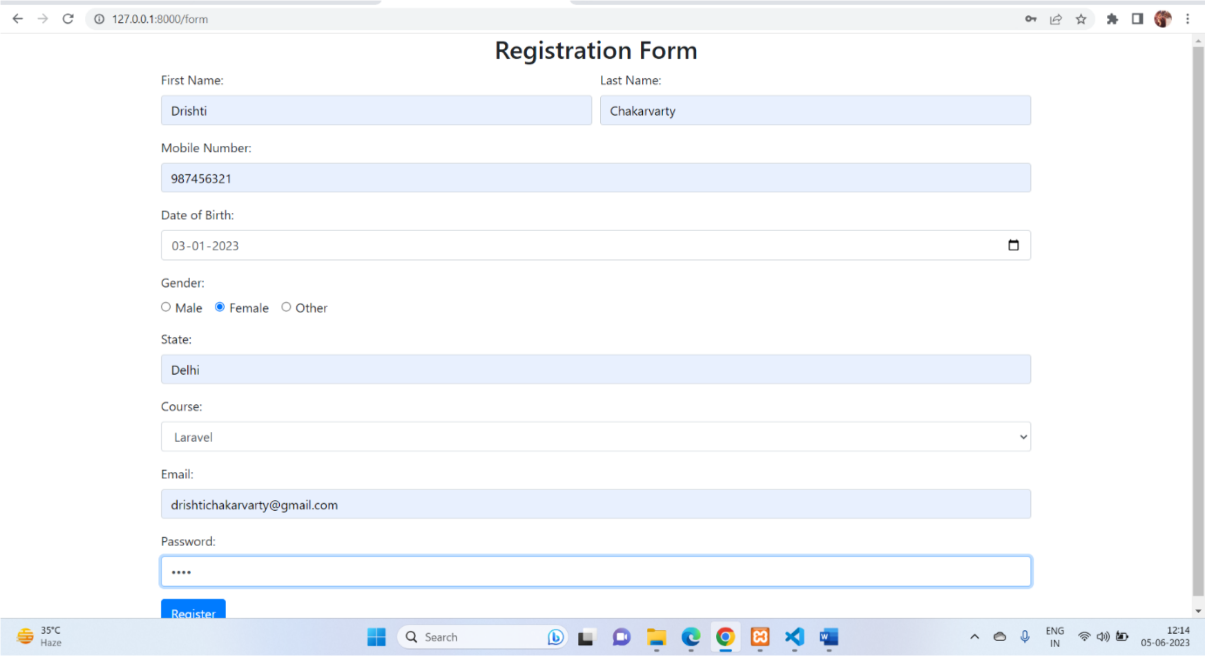
Route::get(‘/form/view’,[Form::class,’view’]);

Step-4 Now Open Controller then create function.



```
            ]
            );
        echo "<pre>";
        print_r($request->all());
        echo "</pre>";

        //create an object of register class
        $register=new Register;
        $register->first_name=$request['first_name'];
        $register->last_name=$request['last_name'];
        $register->mobile=$request['mobile'];
        $register->dob=$request['dob'];
        $register->gender=$request['gender'];
        $register->state=$request['state'];
        $register->course=$request['course'];
        $register->email=$request['email'];
        $register->password=md5($request['password']);
        $register->save();
    }
    public function view(){
        $registers=Register::all();
        $data=compact('registers');
        return view('form-view')->with($data);
    }
```

Step-5 Now Submit the details in the form then see the result.



Registration Form

First Name: Drishti

Last Name: Chakarvarty

Mobile Number: 987456321

Date of Birth: 03-01-2023

Gender:
○ Male ● Female ○ Other

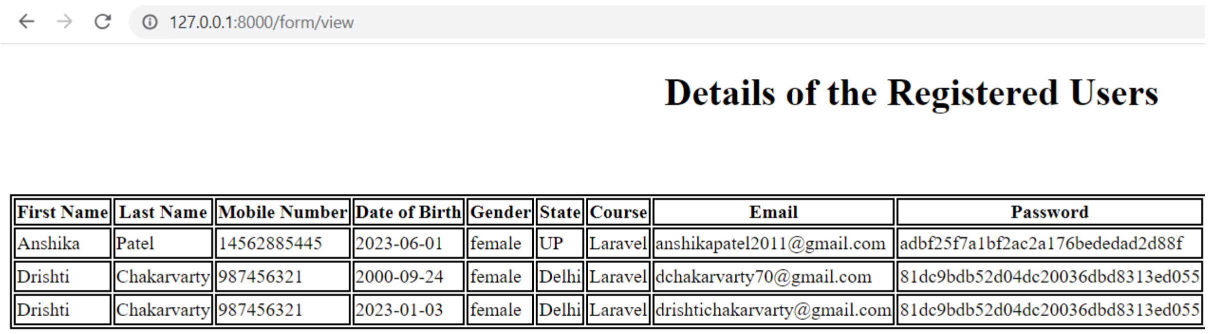
State: Delhi

Course: Laravel

Email: drishtichakarvarty@gmail.com

Password: ••••

Register

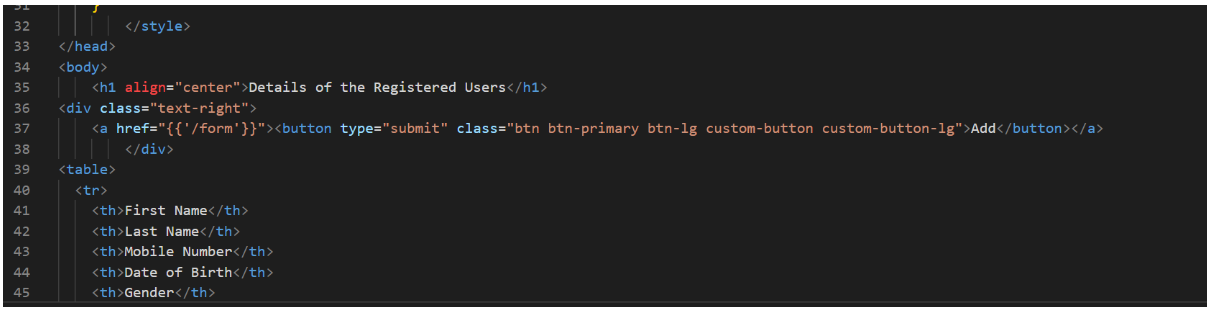


Details of the Registered Users

First Name	Last Name	Mobile Number	Date of Birth	Gender	State	Course	Email	Password
Anshika	Patel	14562885445	2023-06-01	female	UP	Laravel	anshikapatel2011@gmail.com	adbf25f7a1bf2ac2a176bededad2d88f
Drishti	Chakarvarty	987456321	2000-09-24	female	Delhi	Laravel	dchakarvarty70@gmail.com	81dc9bdb52d04dc20036dbd8313ed055
Drishti	Chakarvarty	987456321	2023-01-03	female	Delhi	Laravel	drishtichakarvarty@gmail.com	81dc9bdb52d04dc20036dbd8313ed055

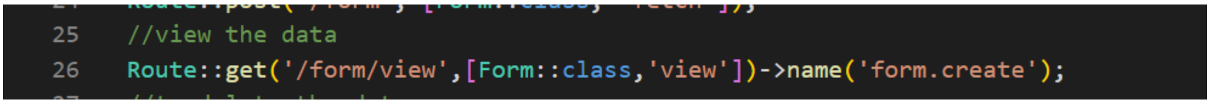
Routing through Buttons

<u>Step-1</u> Now go the form-view.blade.php file and create a button to add the data to the table.



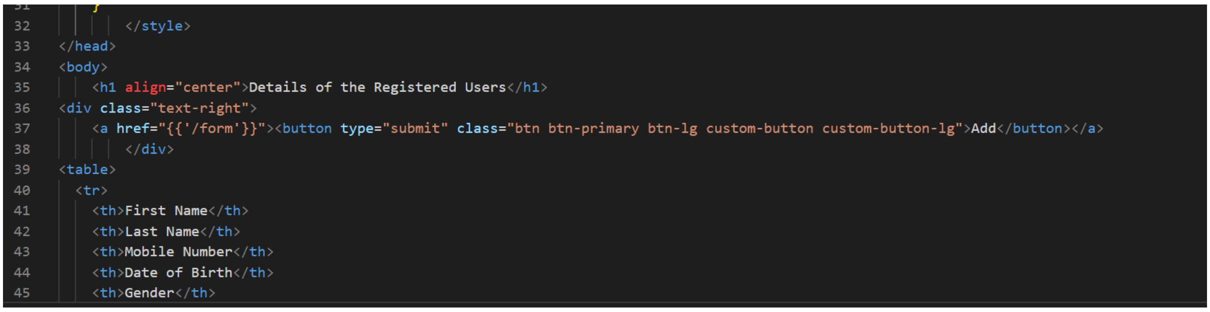
```
        </style>
</head>
<body>
    <h1 align="center">Details of the Registered Users</h1>
<div class="text-right">
    <a href="{{'/form'}}"><button type="submit" class="btn btn-primary btn-lg custom-button custom-button-lg">Add</button></a>
        </div>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Mobile Number</th>
    <th>Date of Birth</th>
    <th>Gender</th>
```

<u>Step-2</u> Now go the web.php file and provide the named route.



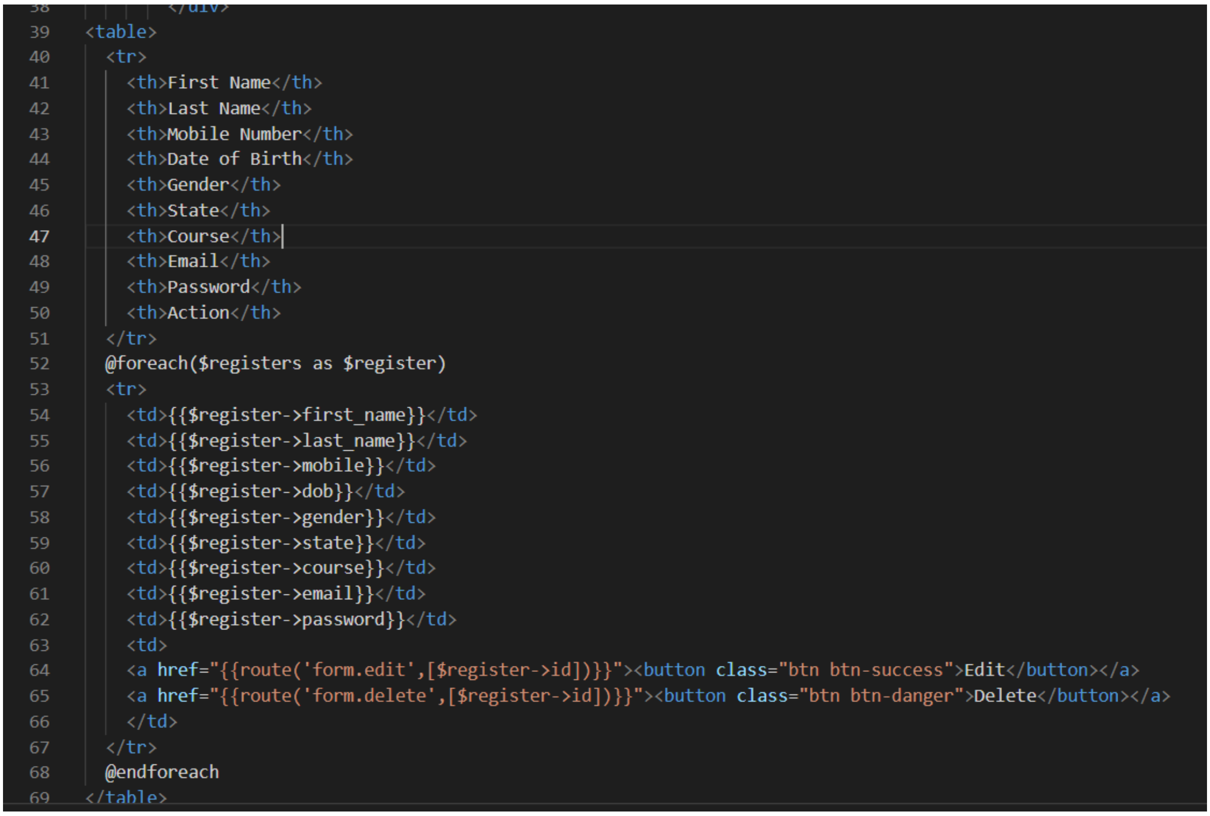
```
//view the data
Route::get('/form/view',[Form::class,'view'])->name('form.create');
```

<u>Step-3</u> Now again go to the form-view.blade.php file and add the route to the **"Add"** button.

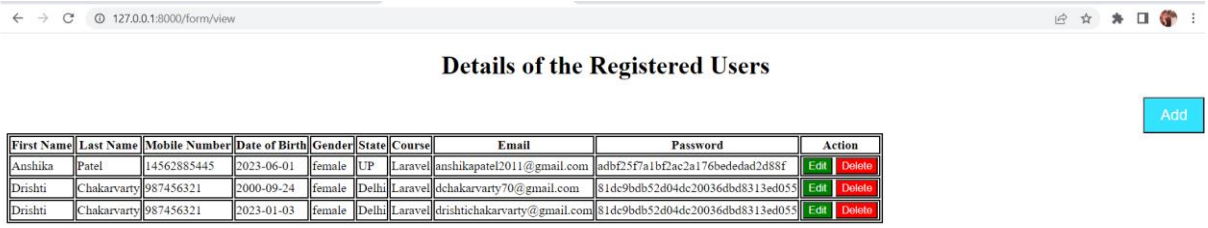


```
        </style>
</head>
<body>
    <h1 align="center">Details of the Registered Users</h1>
<div class="text-right">
    <a href="{{'/form'}}"><button type="submit" class="btn btn-primary btn-lg custom-button custom-button-lg">Add</button></a>
        </div>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Mobile Number</th>
    <th>Date of Birth</th>
    <th>Gender</th>
```

<u>Step-4</u> And also create two more buttons to the end of the table to **“Edit”** and “**Delete”** Data.



```
        </div>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Mobile Number</th>
    <th>Date of Birth</th>
    <th>Gender</th>
    <th>State</th>
    <th>Course</th>
    <th>Email</th>
    <th>Password</th>
    <th>Action</th>
  </tr>
  @foreach($registers as $register)
  <tr>
    <td>{{$register->first_name}}</td>
    <td>{{$register->last_name}}</td>
    <td>{{$register->mobile}}</td>
    <td>{{$register->dob}}</td>
    <td>{{$register->gender}}</td>
    <td>{{$register->state}}</td>
    <td>{{$register->course}}</td>
    <td>{{$register->email}}</td>
    <td>{{$register->password}}</td>
    <td>
    <a href="{{route('form.edit',[$register->id])}}"><button class="btn btn-success">Edit</button></a>
    <a href="{{route('form.delete',[$register->id])}}"><button class="btn btn-danger">Delete</button></a>
    </td>
  </tr>
  @endforeach
</table>
```

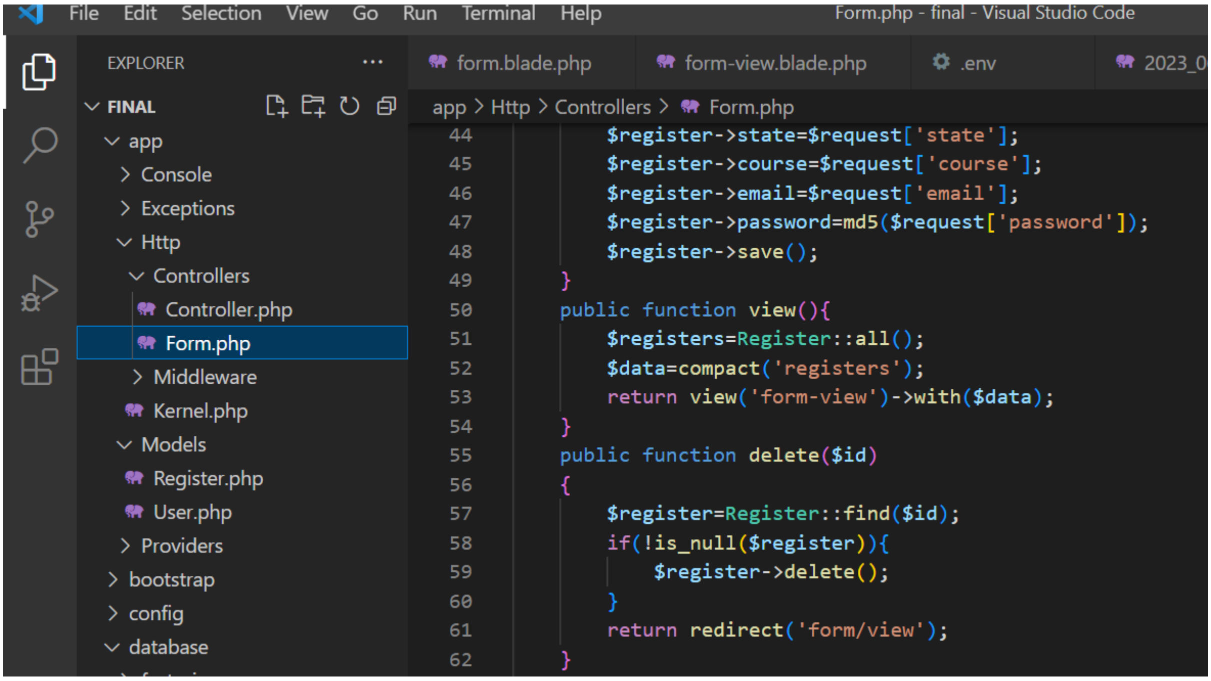


Details of the Registered Users

First Name	Last Name	Mobile Number	Date of Birth	Gender	State	Course	Email	Password	Action
Anshika	Patel	14562885445	2023-06-01	female	UP	Laravel	anshikapatel2011@gmail.com	adbf25f7a1bf2ac2a176bededad2d88f	Edit Delete
Drishti	Chakarvarty	987456321	2000-09-24	female	Delhi	Laravel	dchakarvarty70@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete
Drishti	Chakarvarty	987456321	2023-01-03	female	Delhi	Laravel	drishtichakarvarty@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete

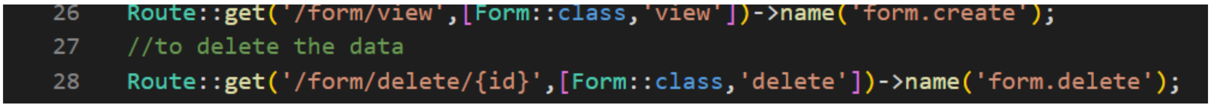
Delete Query

<u>Step-1</u> Go to the controller file and create a function.



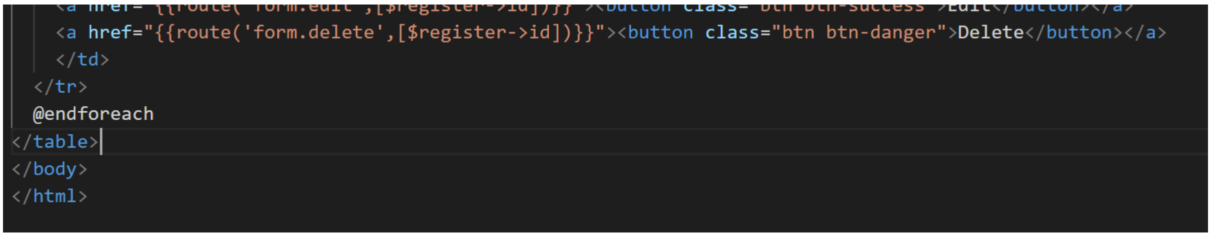
```php
        $register->state=$request['state'];
        $register->course=$request['course'];
        $register->email=$request['email'];
        $register->password=md5($request['password']);
        $register->save();
    }
    public function view(){
        $registers=Register::all();
        $data=compact('registers');
        return view('form-view')->with($data);
    }
    public function delete($id)
    {
        $register=Register::find($id);
        if(!is_null($register)){
            $register->delete();
        }
        return redirect('form/view');
    }
```

Step-2 Now go the web.php and create a route for delete.



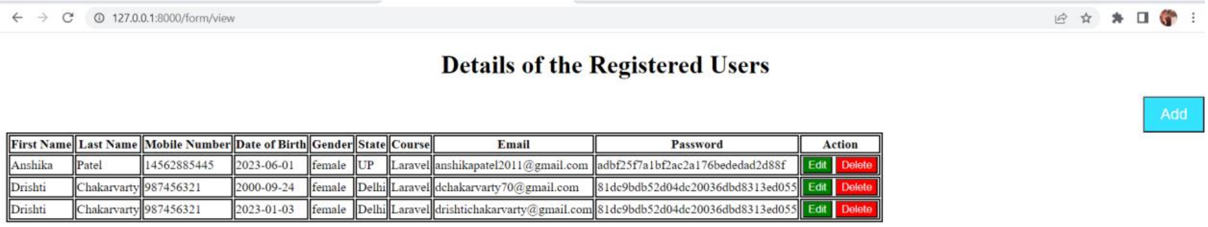
```php
Route::get('/form/view',[Form::class,'view'])->name('form.create');
//to delete the data
Route::get('/form/delete/{id}',[Form::class,'delete'])->name('form.delete');
```

Step-3 Now to the form-view.blade.php file and give the route to the Delete Button.



```html
        <a href="{{route('form.edit',[$register->id])}}"><button class="btn btn-success">Edit</button></a>
        <a href="{{route('form.delete',[$register->id])}}"><button class="btn btn-danger">Delete</button></a>
        </td>
      </tr>
    @endforeach
</table>
</body>
</html>
```

Step-4 Now if you click on the delete button you will see that your data will be deleted.

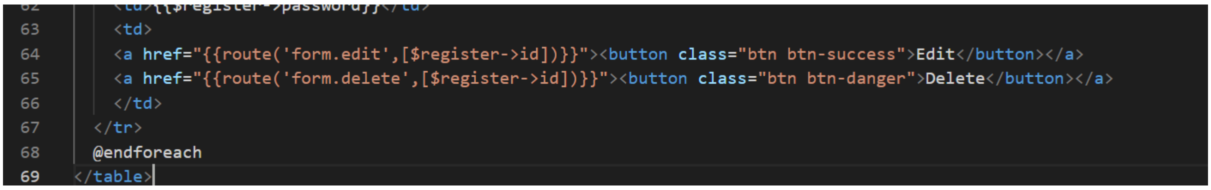


Details of the Registered Users

First Name	Last Name	Mobile Number	Date of Birth	Gender	State	Course	Email	Password	Action
Anshika	Patel	14562885445	2023-06-01	female	UP	Laravel	anshikapatel2011@gmail.com	adbf25f7a1bf2ac2a176bededad2d88f	Edit Delete
Drishti	Chakarvarty	987456321	2000-09-24	female	Delhi	Laravel	dchakarvarty70@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete
Drishti	Chakarvarty	987456321	2023-01-03	female	Delhi	Laravel	drishtichakarvarty@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete

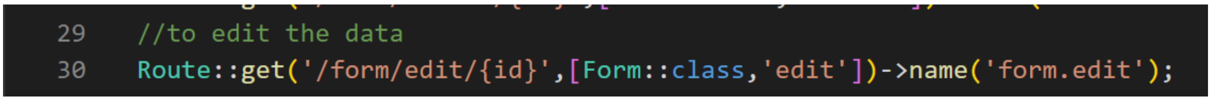
Update Query

Step-1 Now again go to the Form-view file and give the route to the **“edit”** button.



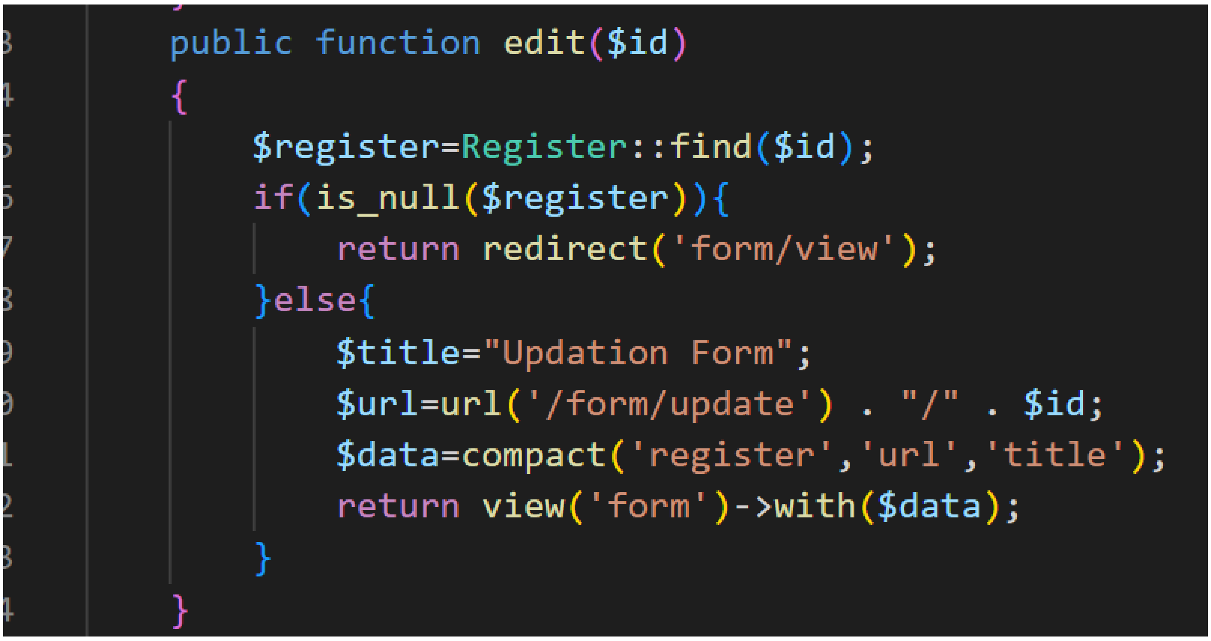
```
        <td>{{$register->password}}</td>
        <td>
        <a href="{{route('form.edit',[$register->id])}}"><button class="btn btn-success">Edit</button></a>
        <a href="{{route('form.delete',[$register->id])}}"><button class="btn btn-danger">Delete</button></a>
        </td>
      </tr>
      @endforeach
    </table>
```

Step-2 Go to the web.php file and give the route for edit.



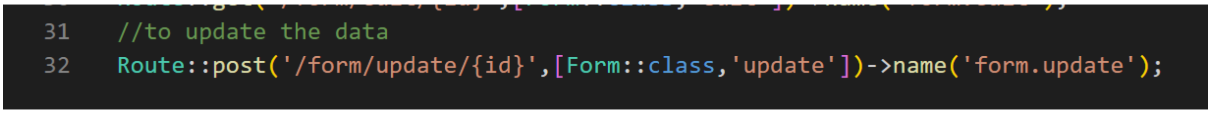
```
//to edit the data
Route::get('/form/edit/{id}',[Form::class,'edit'])->name('form.edit');
```

Step-3 Now go to the controller file and create a function.



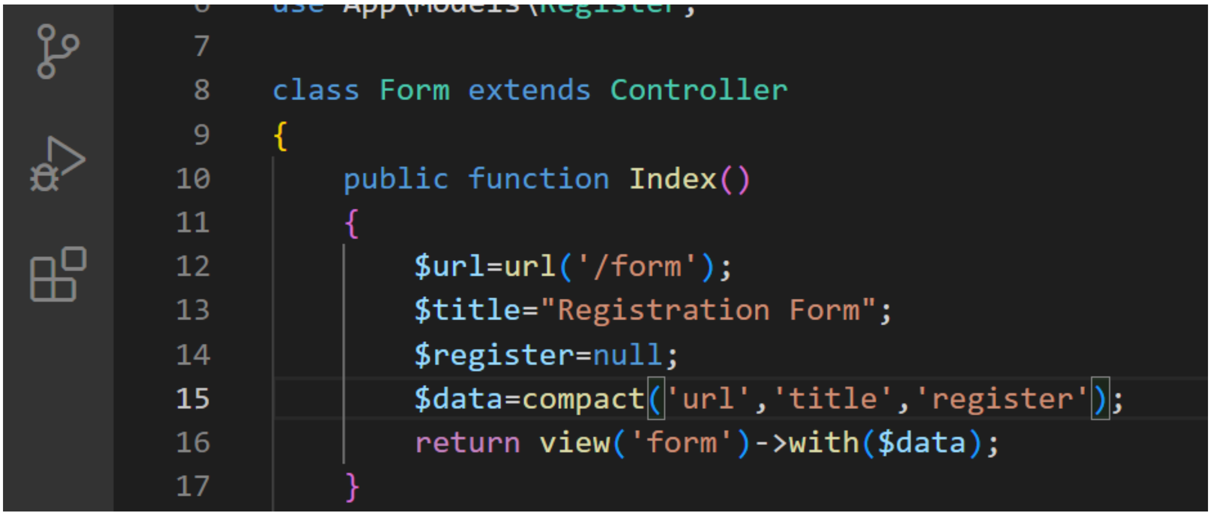
```
    public function edit($id)
    {
        $register=Register::find($id);
        if(is_null($register)){
            return redirect('form/view');
        }else{
            $title="Updation Form";
            $url=url('/form/update') . "/" . $id;
            $data=compact('register','url','title');
            return view('form')->with($data);
        }
    }
```

Step-4 Again go to the web.php file and now give the route for update using post method.



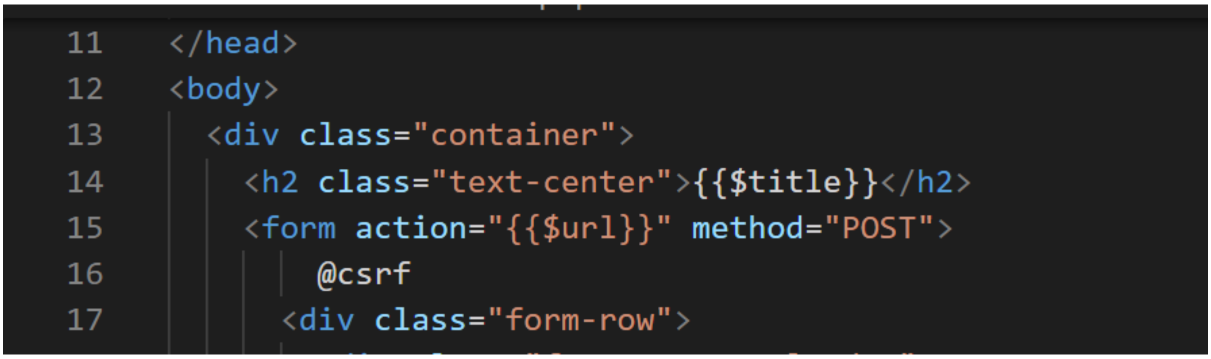
```
//to update the data
Route::post('/form/update/{id}',[Form::class,'update'])->name('form.update');
```

Step-5 In the same controller file go to the Index function and do some changes.



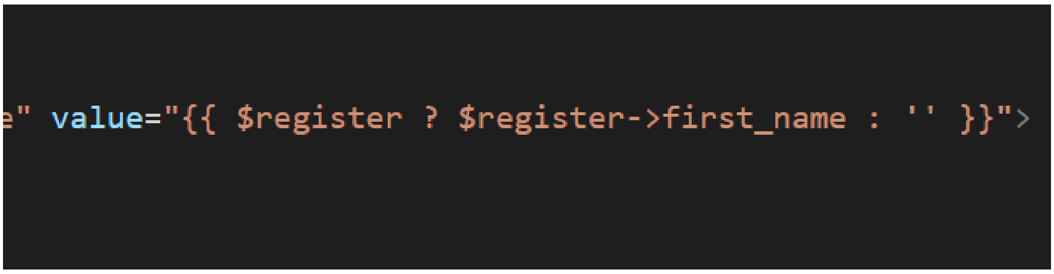
```
use App\Models\Register;

class Form extends Controller
{
    public function Index()
    {
        $url=url('/form');
        $title="Registration Form";
        $register=null;
        $data=compact('url','title','register');
        return view('form')->with($data);
    }
```

Step-6 Now go to the form.blade.php file and provide the url in form action and the title also.



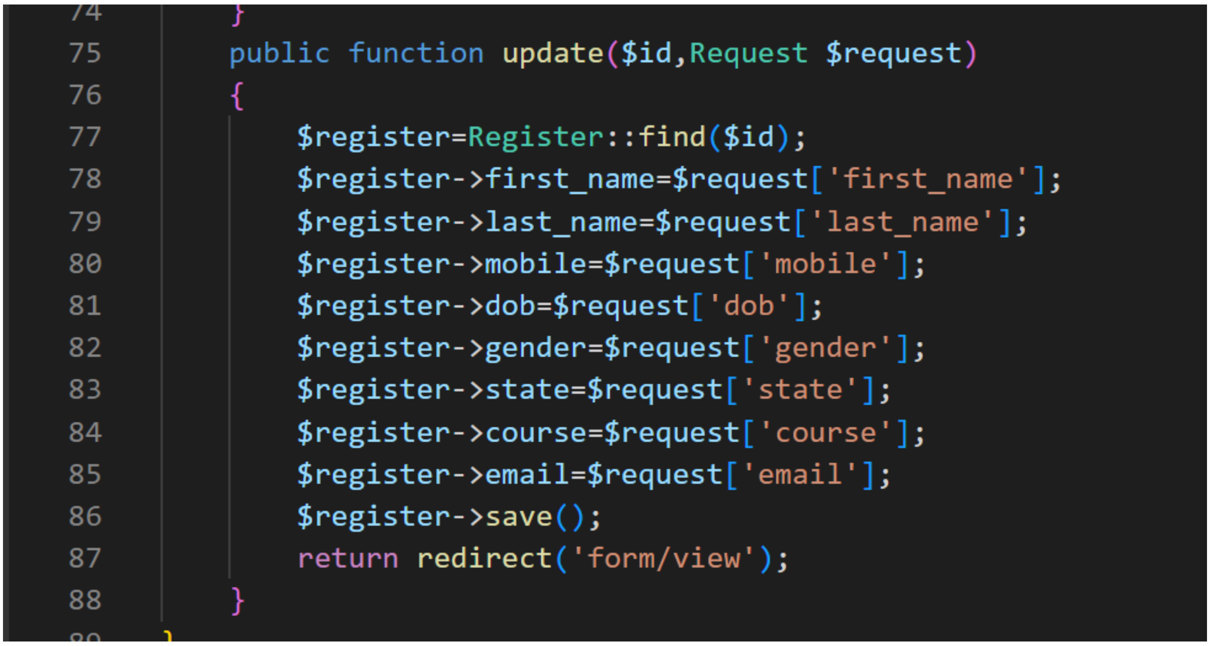
```
</head>
<body>
  <div class="container">
    <h2 class="text-center">{{$title}}</h2>
    <form action="{{$url}}" method="POST">
      @csrf
    <div class="form-row">
```

Step-7 In the same file you have to provide the value of the data which you want to update.



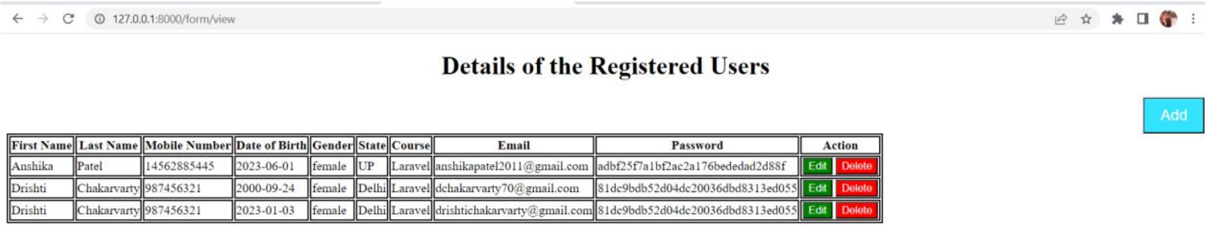
```
e" value="{{ $register ? $register->first_name : '' }}">
```

Step-8 Go to the Controller file and create a function to update.



```
    }
    public function update($id,Request $request)
    {
        $register=Register::find($id);
        $register->first_name=$request['first_name'];
        $register->last_name=$request['last_name'];
        $register->mobile=$request['mobile'];
        $register->dob=$request['dob'];
        $register->gender=$request['gender'];
        $register->state=$request['state'];
        $register->course=$request['course'];
        $register->email=$request['email'];
        $register->save();
        return redirect('form/view');
    }
```

<u>Step-9</u> Now you are able to Create, Read, Update and Delete data in Laravel.



Details of the Registered Users

First Name	Last Name	Mobile Number	Date of Birth	Gender	State	Course	Email	Password	Action
Anshika	Patel	14562885445	2023-06-01	female	UP	Laravel	anshikapatel2011@gmail.com	adbf25f7a1bf2ac2a176bededad2d88f	Edit Delete
Drishti	Chakarvarty	987456321	2000-09-24	female	Delhi	Laravel	dchakarvarty70@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete
Drishti	Chakarvarty	987456321	2023-01-03	female	Delhi	Laravel	drishtichakarvarty@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Edit Delete