Laravel Installation and Folder Structure Assignment

<u>Part-1:</u>

<u>Laravel Installation</u>

Installation Prerequisite:

1. Installing PHP 8.1 or updating existing PHP version into PHP 8.1 since it is required to have PHP 8.1 installed in our pc to install Laravel-10. Also remember to uncomment ;extension=gd ;extension=zip commands located in the php.ini file.

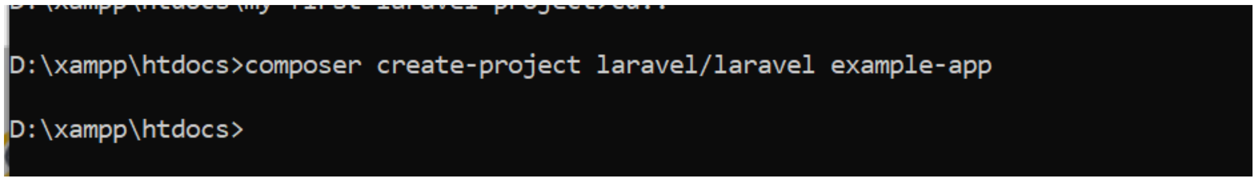
2. Installing PHP package manager[Composer]

Steps to install the composer:

1. Go to the website https://getcomposer.org/download/
2. Download windows installer if the CLI does not work for you. In my case I had to do it using windows installer since CLI gave me several error.
3. Install the composer into xampp/php/php.exe since it is a PHP package manager.
4. Accept the installation steps by complying with next command and finally finish

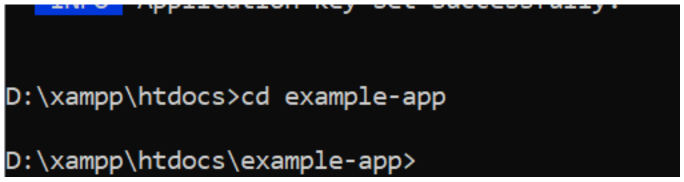
Installing Laravel using CLI:

1. Go to the directory where Laravel needs to be installed.
2. Run the command **composer create-project laravel/laravel example-app** on the command



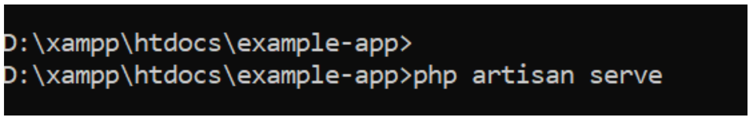
```
D:\xampp\htdocs\my first laravel project>cd..

D:\xampp\htdocs>composer create-project laravel/laravel example-app

D:\xampp\htdocs>
```

3. prompt and it will start installing Laravel into the directory.
4. Then change the directory to the project directory using **cd example-app**



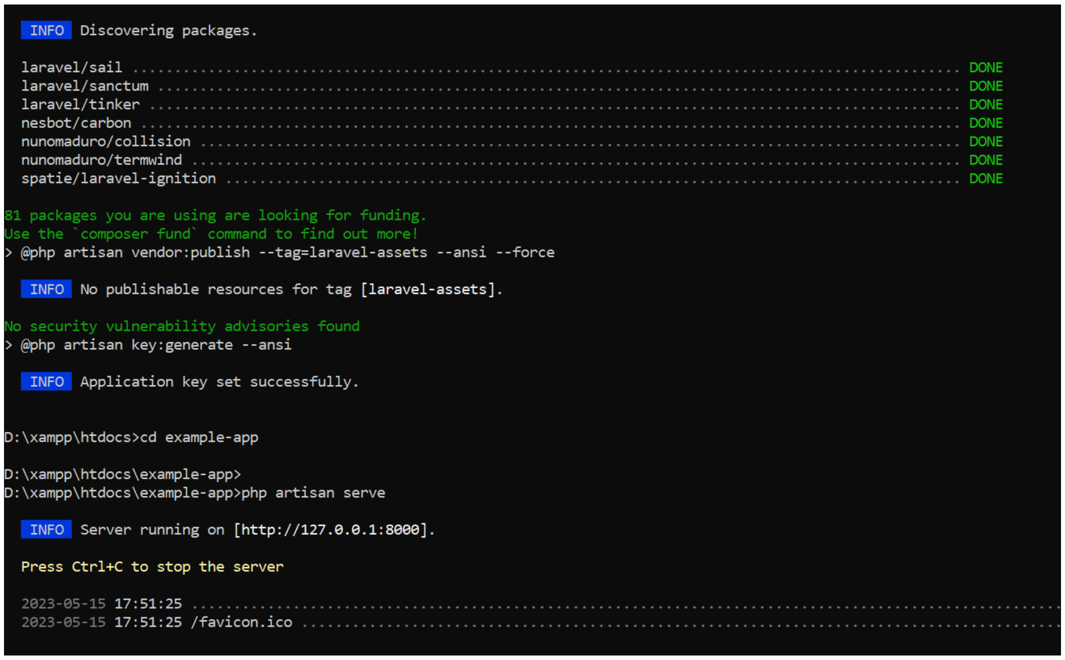
```
  INFO  Application key set successfully.


D:\xampp\htdocs>cd example-app

D:\xampp\htdocs\example-app>
```

5. Then start Laravel's local development server using the Laravel's Artisan CLI serve command like this: **php artisan serve**



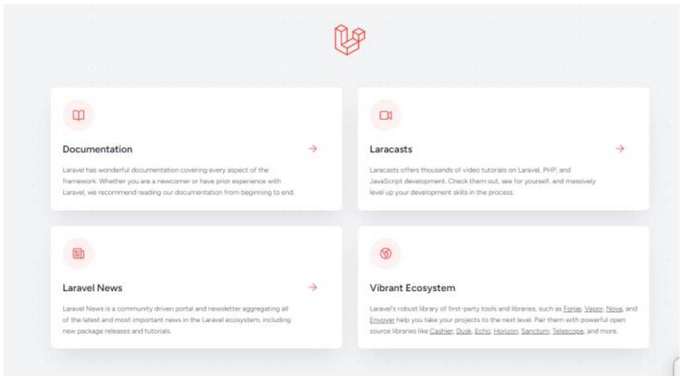
```
D:\xampp\htdocs\example-app>
D:\xampp\htdocs\example-app>php artisan serve
```

6. After successful installation the CLI will show up like the following screenshot:



```
  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

81 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.


D:\xampp\htdocs>cd example-app

D:\xampp\htdocs\example-app>
D:\xampp\htdocs\example-app>php artisan serve

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

  Press Ctrl+C to stop the server

  2023-05-15 17:51:25 ..........................................................................................
  2023-05-15 17:51:25 /favicon.ico .............................................................................
```

7. Running project at http://127.0.0.1: 8000 will look like the following



Documentation

Laravel has wonderful documentation covering every aspect of the framework. Whether you are a newcomer or have prior experience with Laravel, we recommend reading our 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.

Part-2

Describe the Purpose of Different Folders Located in the Laravel Project:

App: The app directory contains the core code of our application. Almost all of the classes of our application will be in this directory.

Bootstrap: The bootstrap directory contains the app.php file which bootstraps the framework. This directory also includes a cache directory which contains framework generated files for performance optimization such as the route and services cache files. We don't need to modify any file within this directory.

Config: The config directory, as the name implies, contains all of our application's configuration files.

Database: The database directory contains our database migrations, model factories, and seeds. If we wish, you may also use this directory to hold an SQLite database.

Public: The public directory contains the index.php file, which is the entry point for all requests entering our application and configures autoloading. This directory also includes our assets such as images, JavaScript, and CSS.

Resources: The resources directory contains our views as well as our raw, un-compiled assets such as CSS or JavaScript.

Routes: The routes directory contains all of the route definitions for the application. By default, several route files are included with Laravel: web.php, api.php, console.php, and channels.php.

> **web.php:** The web.php file contains routes that the RouteServiceProvider places in the web middleware group, which provides session state, CSRF protection, and cookie encryption.
>
> **api.php:** The api.php file contains routes that the RouteServiceProvider places in the api middleware group. These routes are intended to be stateless, so requests entering the application through these routes are intended to be authenticated via tokens and will not have access to session state.
>
> **console.php:** this console.php file is where we may define all of our closure based console commands. Each closure is bound to a command instance allowing a simple approach to interacting with each command's IO methods. Even though this file does not define HTTP routes, it defines console based entry points (routes) into our application.
>
> **channels.php:** The channels.php file is where we may register all of the event broadcasting channels that our application supports.

Storage: The storage directory contains our logs, compiled Blade templates, file based sessions, file caches, and other files generated by the framework. This directory is segregated into the following directories:

1. App

2. Framework
3. Logs

App: The app directory may be used to store any files generated by our application.

Framework: The framework directory is used to store framework generated files and caches.

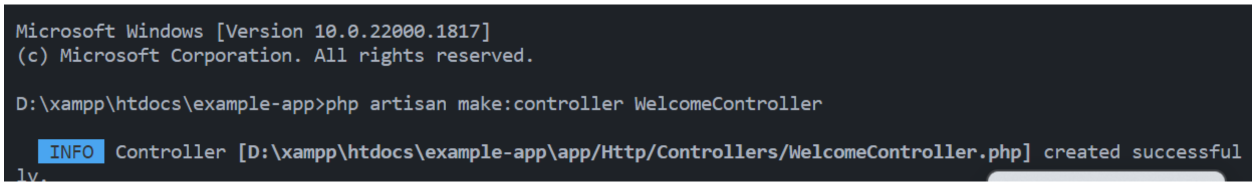
Logs: The logs directory contains our application's log files.

Tests: The tests directory contains the automated tests. Example PHPUnit unit tests, feature tests etc. We may need to work with this folder when we want to perform testing of a software project.

Vendor: The vendor directory contains the Composer dependencies. Files in here are auto generated during the installation of Laravel, as a developer we don't need to work with this folder.

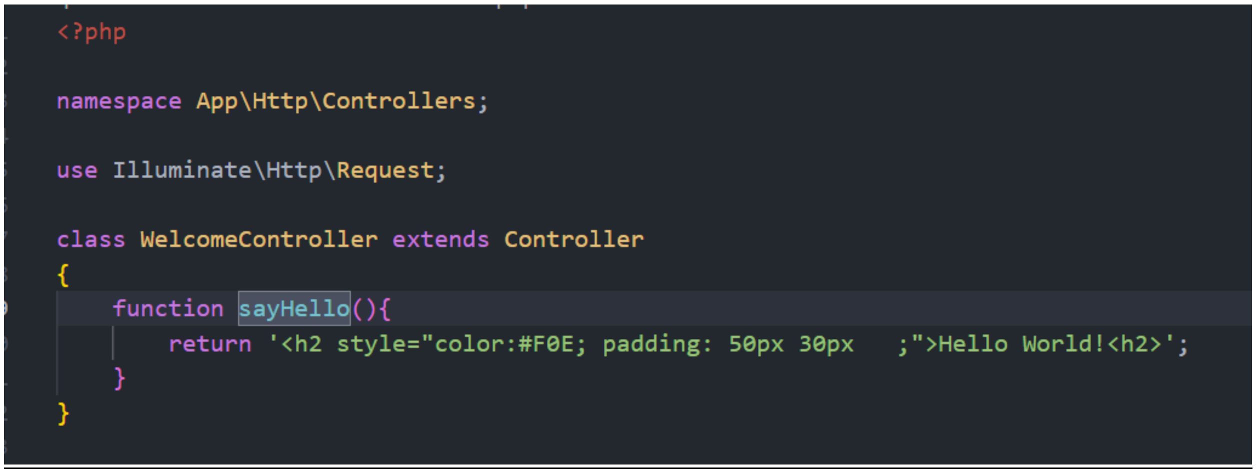
<u>Create a new route in your Laravel project that displays a simple "Hello, World!" message. Take a screenshot of the running route.</u>

1. To create a new route I have made a a controller name wise WelcomeController.



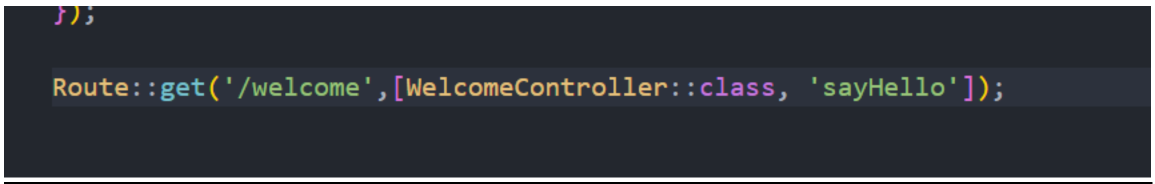
```
Microsoft Windows [Version 10.0.22000.1817]
(c) Microsoft Corporation. All rights reserved.

D:\xampp\htdocs\example-app>php artisan make:controller WelcomeController

  INFO  Controller [D:\xampp\htdocs\example-app\app/Http/Controllers/WelcomeController.php] created successful
ly.
```

2. <u>Then I have created a public function called sayHello() under the WelcomeControler class which returns "Hello World!" message</u>



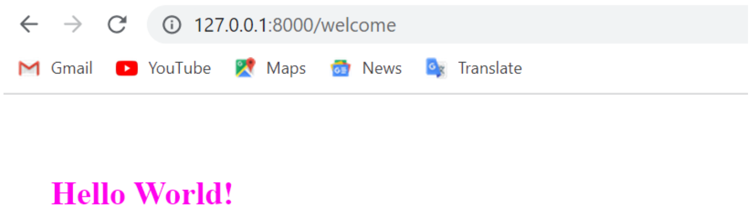
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
    function sayHello(){
        return '<h2 style="color:#F0E; padding: 50px 30px   ;">Hello World!<h2>';
    }
}
```

3. **<u>Then I have created a route under routes/web.php like the following:</u>**



```
});

Route::get('/welcome',[WelcomeController::class, 'sayHello']);
```

4. On the website when visiting the route http://127.0.0.1:8000/welcome it show up like the following:



Hello World!