Module-13 Assignment (Part 1)

1. Start by installing PHP on my computer.

2. Next, download and install Composer, which is a dependency manager for PHP.

3. Setting up Laravel:

- Open terminal or command line interface (CLI).

- Use Composer to run the following command and install Laravel globally:

  ✓ **`composer global require laravel/installer`**

4. Creating a New Laravel Project:

- Specify the location where i want to create my Laravel project.

- Execute the following command to start a fresh Laravel project:

  ✓ **`laravel new project-name`**

5. Accessing the Laravel Application:

- Change my directory to the project folder using the following command:

  ✓ **`cd project-name`**

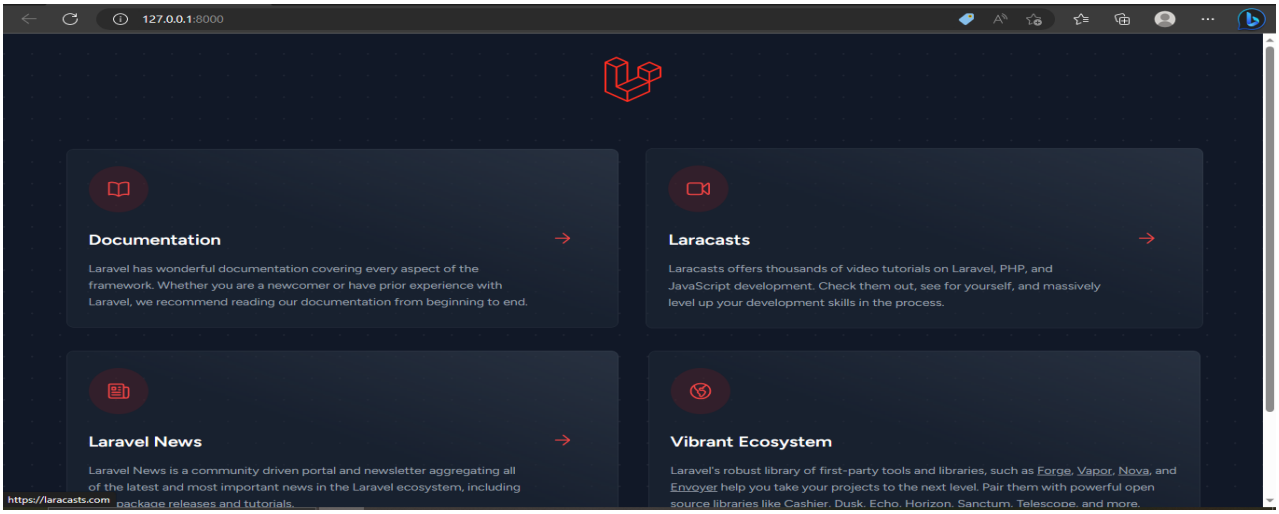
6. Open the project folder in your preferred code editor, for example, Visual Studio Code, using the following command:

  ✓ **`code .`**

7. Finally, I have launch the development server and serve my Laravel application by running the following command:

  ✓ **`php artisan serve`**

My development server is running as seen on the screen below.



Part 2 : Identifying each folder's function in a Laravel project:

- **App:** **The Laravel application's main component is the app folder.**

It includes the models, controllers, middleware, and other PHP classes for the application.

We specify the business logic for your application in the app folder.

- **Bootstrap :** **The Laravel framework's bootstrapping files are located in the bootstrap subdirectory.**

It also contains the app.php file, which launches the application and loads the required files.

Specifically, setting up the application environment and using the bootstrap folder

carrying out any initialization activities.

- **Database :** **The Laravel application's database-related files are kept in the database folder.**

It contains migration files that specify how our database tables should be organized.

Files that fill the database with test data are located in the seeds directory.

To produce fictitious data for testing, we may also establish factories in the factories directory.

- **Public :** **The document root of the web server and the starting point for all HTTP requests is**

the public folder. It includes the index.php file, which acts as the application's front controller.

The public folder is often where static files like CSS, JavaScript, and pictures are kept, making

making them open to the public.

- **Resources :** **Views, language files, and other non-PHP resources utilized by the program are**

kept under the resources folder. The Blade templates in the views directory establish the design and layout of our

HTML pages of an application.

Resources may also contain other directories that house assets, CSS, and JavaScript files, as well as language files for localization.

- **Routes :** The route definitions for our application are located in the routes folder.

Routes for responding to HTTP requests sent by web browsers are defined in the web.php file.

Routes for API endpoints are defined in the api.php file.

It is possible to structure and manage routes for various components of our application using additional route files.

- **Storage :** The application's created files are kept in the storage folder.

It contains directories for transient or dynamically created content, logs, cache files, session files, and other items.

Additionally, Laravel stores uploaded files, such user-created photos or documents, in the storage folder.

- **Tests :** For the Laravel application, there are automated tests in the tests subdirectory.

It contains test cases and test suites that aid in ensuring the efficiency and accuracy of the code.

A testing framework is offered by Laravel that enables the creation of unit tests, integration tests, and more.

- **Vendor :** The Laravel framework is among the dependencies installed using Composer that

can be found in the vendor folder. It contains all of the third-party libraries and software packages needed by the program.

Since, Composer creates and maintains the vendor folder, we shouldn't directly alter its contents.

In my Laravel project, I've added a new route that shows a straightforward "Hello, World!" message.

```
Route::get('/hello',function () {
    return "Hello, World!";
});
```

Screenshot of this route in server