Part 1:

To install Laravel, I followed these steps:

1. Installing Laravel:
    a. Opened my command line interface (CLI) or terminal.
    b. Runned the following command to install Laravel globally using Composer:
    c. **composer global require laravel/installer**
2. Creating a New Laravel Project:

    a. Navigated to the directory where i wanted to create my Laravel project.

    b. Run the following command to create a new Laravel project:

    c. **laravel new project-name**

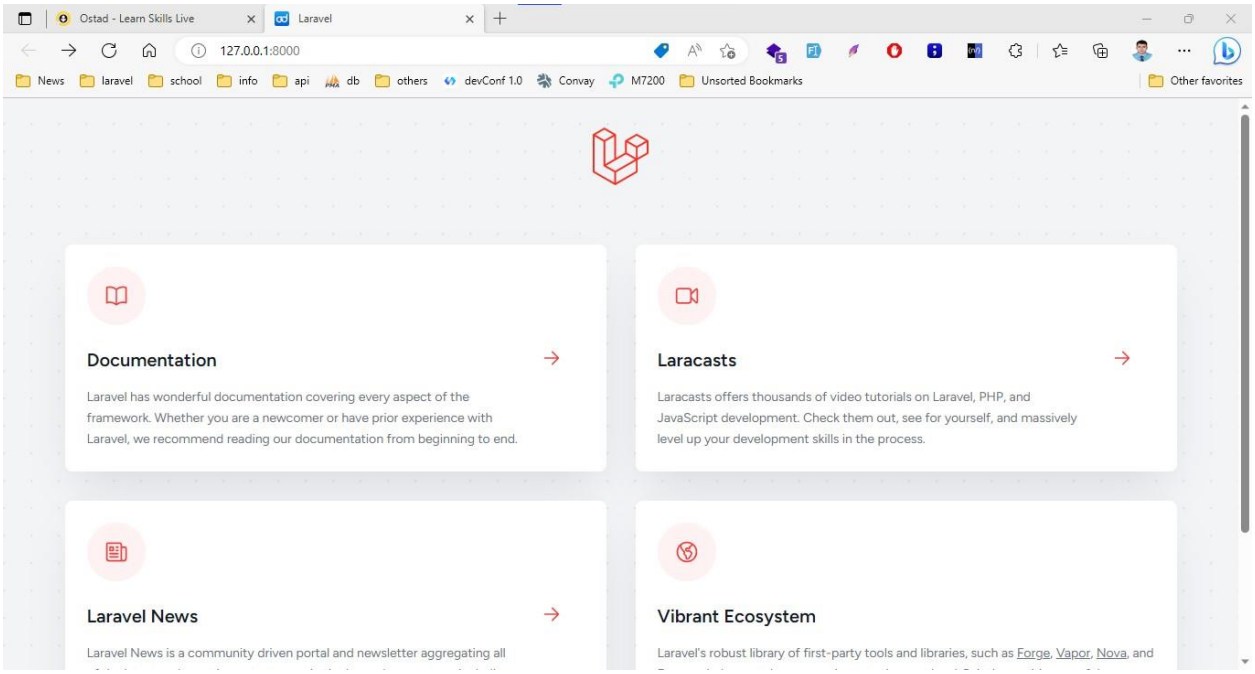
3. After Creating Success Laravel Application:
    a. Change into the project directory: using this command
    b. **cd project-name**
4. Then opened the folder into my VS CODE code editor my using this command:

    code .

5. Finally, To start the development server and serve my Laravel application, I run the following command:

    php artisan serve

Below Screen sort shows the Running of my development Server



Part 2:

Describing the purpose of each folder in a Laravel Project:

- **app:**

  The "app" folder in Laravel contains the core logic and application code of a Laravel project. It is one of the most important folders in a Laravel project as it houses the "Models," "Controllers," "Providers," and other related files.

  Here's a brief description of what each sub-folder in the "app" folder does:

  Console: This folder contains all the Artisan console commands for the application.

  Exceptions: This folder contains custom exception handlers for the application.

  Http: This folder contains the controllers, middleware, and requests classes for handling incoming HTTP requests.

  Models: This folder contains the Eloquent models for the application's database tables.

  Providers: This folder contains service providers for the application.

  Overall, the "app" folder serves as the central hub for the Laravel application's core logic and is where most of the application's functionality is built and organized.

- **bootstrap:**

  The bootstrap folder in Laravel contains the essential files and folders required to bootstrap the Laravel framework and start the application. It includes the app.php file, which initializes the Laravel application and sets up the service container, event dispatcher, and other components.

  The cache folder contains the cached configuration files, routes, and views, which improve the application's performance by reducing the number of disk and database reads.

  The config folder contains the application's configuration files, such as database, session, and mail settings. The routes folder contains the web and API routes defined by the application, while the storage folder contains the application's log files, cache, and other runtime files.

  Additionally, the public folder, which is located outside of the bootstrap folder but is closely related to it, contains the entry point for the application, the index.php file, and other publicly accessible files such as images, JavaScript, and CSS files.

  Overall, the bootstrap folder in Laravel is a critical part of the application's infrastructure, providing the necessary files and folders to initialize and configure the framework and start the application.

- **config:**

  In Laravel, the config folder is used to store various configuration files for the application. These files contain key-value pairs that are used to configure different aspects of the application, such as the database settings, cache settings, mail settings, and much more.

  The purpose of the config folder is to provide a central location for managing application configuration. Instead of hardcoding configuration values throughout the application, developers can define them in separate configuration files and access them using Laravel's configuration system.

The config folder contains several files by default, such as app.php, database.php, and cache.php. However, developers can create their own configuration files and store them in this folder as well.

To access configuration values within the application, developers can use the config helper function, which accepts the name of the configuration file and the key of the value they want to retrieve. For example, to retrieve the value of the APP_NAME key from the app.php configuration file, developers can use the following code:

Overall, the config folder in Laravel serves an important role in managing application configuration, making it easier to modify and maintain the settings of the application.

- **database:**

In Laravel, the database folder serves as a directory where you can store all the files related to database management. It typically contains the following sub-folders:

migrations: This folder contains the database migration files. A migration is a version control system for database schemas. It allows you to modify your database schema without having to recreate it from scratch. The migration files are created using Laravel's built-in migration generator, and they contain instructions for creating or modifying database tables, columns, indexes, and other database structures.

seeds: This folder contains the database seed files. Seeders allow you to populate your database with test data, which can be useful for testing or for creating demo data. The seed files are also created using Laravel's built-in seeder generator.

factories: This folder contains the model factories. Model factories are used to generate fake data for testing and seeding purposes. You can use them to create a large number of model instances with random data, which can be useful when you need to test how your application handles large amounts of data.

Overall, the database folder in Laravel serves as a central location for managing your application's database-related files, making it easier to organize and maintain your code.

- **public:**

In Laravel, the "public" folder is the web server's document root, which means it contains all the files and assets that will be publicly accessible through the web browser.

The purpose of the public folder in Laravel is to store all the assets such as images, CSS, JavaScript, and other publicly accessible files that are required to run the Laravel application. When a request is made to the Laravel application, the web server looks for the requested file in the public folder.

In other words, any file or asset that needs to be accessed by the user's browser should be stored in the "public" directory of a Laravel application. All the other files, such as configuration files, views, and controllers, should be stored outside of the public directory to keep them private and secure.

By default, Laravel routes all incoming requests to the index.php file located in the "public" directory. This file is responsible for bootstrapping the Laravel application and routing the request to the appropriate controller or view.

Therefore, it is crucial to ensure that sensitive files or data are not stored in the "public" directory, as they could be accessed by anyone who knows the URL of the file.

- **resources:**

  In Laravel, the resources folder is a directory that contains all the assets and resources used to build the application's user interface. The primary purpose of the resources folder is to organize and store the various resources and assets that are required for building the user interface, such as views, language files, assets like CSS, JavaScript, images, and more.

  The **resources/views** directory contains the HTML templates that are used to generate the application's user interface. These views are typically built using the Blade templating engine, which provides a simple and easy-to-use syntax for creating reusable components and layouts

  Overall, the resources folder is an essential part of any Laravel application, providing a central location for organizing and managing all the resources and assets needed to build the user interface. Other directories within resources may include language files for localization, CSS and JavaScript files, and other assets.

- **routes:**

  In Laravel, the "routes" folder is used to define the routes of your application.

  The purpose of the "routes" folder is to provide a centralized location for defining the URL routes that the application will respond to. In Laravel, routes are defined in the "routes/web.php" file for web requests and "routes/api.php" file for API requests. These files define the endpoints that can be accessed by users and how they should be handled.

  Routes are the entry point of your application and determine how incoming requests are processed and what response is sent back to the client. The "routes" folder helps organize the routes into separate files and makes it easier to maintain and manage your application's endpoints.

  By separating the routes into different files based on their purpose (e.g., web or API), you can keep your codebase organized and easier to maintain. This makes it easier to add or modify routes as your application evolves, without having to search through a single large file to find the route you need.

  Overall, the "routes" folder is an important component of Laravel's routing system, allowing developers to define the application's endpoints in a clear and organized way.

- **storage:**

  In Laravel, the "storage" folder serves as a centralized location for storing various types of data that the application needs to access at runtime. This folder is generally not meant to be directly accessible to the public, and therefore, it is located outside of the web root directory.

  The "storage" folder in Laravel has several subfolders, each with a specific purpose:

  app: This folder stores application-specific files such as cached data, configuration files, and session files.

  framework: This folder contains files related to the Laravel framework, including cached views, session data, and application logs.

  logs: This folder contains log files generated by the application. These files can be useful for debugging purposes.

  Overall, the "storage" folder in Laravel plays an important role in ensuring that the application's data is organized, secure, and easily accessible.

- **tests:**

  In Laravel, the tests folder is used to store automated tests for your application. The purpose of this folder is to provide a standardized way to test your code and ensure that it works as expected. Tests are essential for maintaining the quality of your code, catching bugs early, and preventing regressions as you make changes to your codebase.

  The tests folder contains several subfolders and files, including:

  Unit: This folder contains tests that focus on individual units of code, such as classes, methods, or functions. Unit tests are typically used to ensure that a specific piece of code behaves correctly under different scenarios.

  Feature: This folder contains tests that focus on the behavior of your application as a whole, including interactions between different parts of the codebase. Feature tests are typically used to ensure that your application works correctly from a user's perspective.

  Overall, the tests folder in Laravel provides a comprehensive set of tools for testing your code and ensuring that it works as expected, both during development and after deployment.

- **vendor:**

  In Laravel, the vendor folder contains all the third-party dependencies or packages used by the application. It's created by Composer, which is a dependency manager used in PHP applications to manage the installation and updating of packages.

  When you install a new package using Composer, it downloads the required package files and dependencies to the vendor directory. This directory is excluded from version control since it's considered to be generated code, and the contents are managed by Composer.

  The **vendor** folder contains many subdirectories, each corresponding to a package installed via Composer. These subdirectories contain the package's source code, configuration files, and other necessary files.

  The purpose of the **vendor** folder is to simplify the process of managing dependencies and their versions. It allows developers to easily add, remove, and update packages without having to manually download and include them in the project.

  Overall, the **vendor** folder is an important part of the Laravel ecosystem and is essential for managing dependencies and packages in Laravel applications.

I have Created a new route in my Laravel project that displays a simple "Hello World!" message.

The route is:

```
1  Route::get('/hello',function(){
2      return 'Hello World!';
3  });
```

Secreensort of this route in server: