**Day - 1**

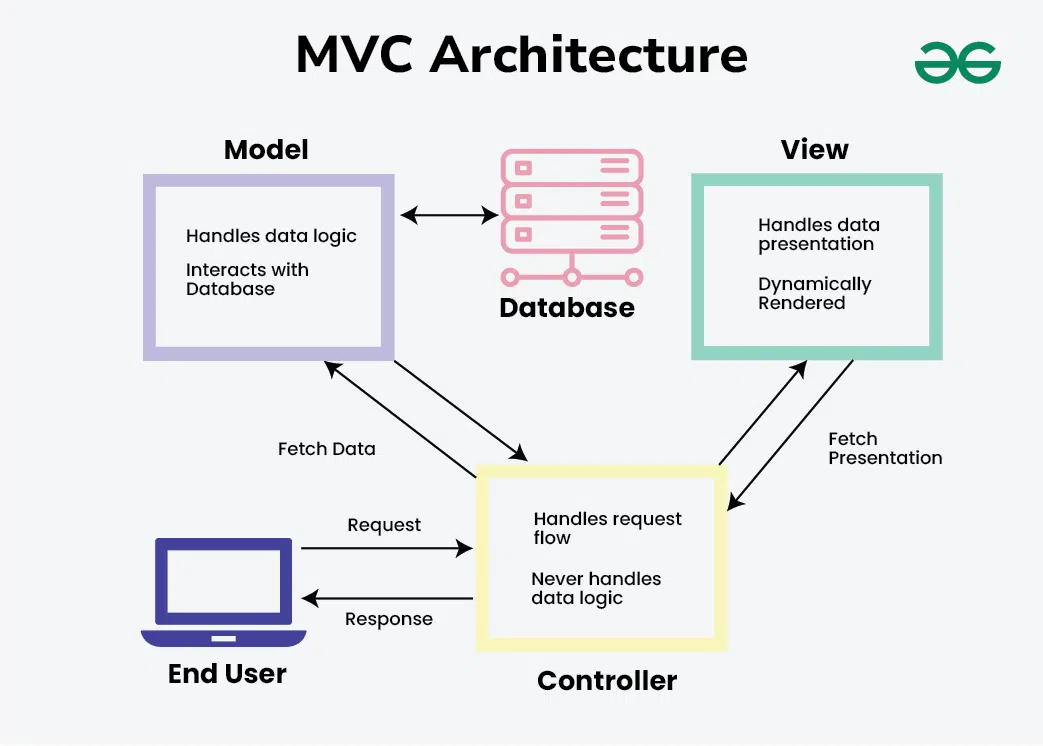
**Understanding Laravel Framework**

**Laravel’s MVC:**

The MVC (Model - View - Control) is the software architecture that is a widely adopted design pattern in web development, Laravel and a most popular PHP framework to achieve modularity, maintainability and scalability.   
 It separates the application into three interconnected components:

1. Model
2. View
3. controller

MVC ensures a clear and more organised way to build the application.



#### **Model:**

* Manages data, states, and rules of the application.
* Handles data and controls the application’s logic and rules.
* Utilizes Laravel’s Eloquent ORM for simplified database interactions.
* Supports CRUD operations on database records.
* Defines relationships between models for managing complex data associations.

#### **View:**

* Responsible for the presentation layer and user interface (UI).
* Presents data to users and receives user input.
* Uses Laravel's Blade templating engine for dynamic and reusable UI components.
* Contains HTML with embedded PHP code.
* Supports conditionals, loops, partials, and modular UI building.

#### **Controller:**

* Handles user input and interactions.
* Acts as a bridge between the Model and the View.
* Processes input, interacts with the Model, and updates the View.
* Defines methods (actions) for handling user requests.
* Manages request logic, input validation, and authorization checks before interacting with the Model.

### **MVC Workflow in Laravel:**

1. **User Request**: A user interacts with the application, sending a request to the server.
2. **Routing**: Laravel’s routing system maps the request to the appropriate Controller method based on defined routes.
3. **Controller**: Receives the request and handles data retrieval, validation, and manipulation. Interacts with the Model to fetch or update data.
4. **Model**: Manages database interactions such as querying or updating records based on Controller instructions.
5. **Data Preparation**: The Controller processes the data and passes it to the relevant View.
6. **View**:Receives the data and uses the Blade templating engine to generate the HTML output.
7. **User Response**: The rendered HTML, along with styles and scripts, is displayed to the user, completing the request-response cycle.

**Route**:

**Routing** in web development is the mechanism for mapping user requests (URLs) to specific actions or logic within a web application. In Laravel, **routing** is handled via the routes/web.php file for web interfaces and the routes/api.php file for APIs.

Laravel supports multiple HTTP methods (verbs) for routing:

| **Method** | **Description** |
| --- | --- |
| get | Retrieves resources (e.g., pages). |
| post | Submits data to the server. |
| put | Updates a resource. |
| patch | Partially updates a resource. |
| delete | Deletes a resource. |
| any | Responds to all HTTP methods. |
| match | Responds to specified methods. |

For example php:  
  
use App\Http\Controllers\PostController;

Route::get('/posts', [PostController::class, 'index']);

**Controller Action**:

Controllers serve as the C (Controller) in the MVC (Model-View-Controller) framework. They handle the logic for your application's requests and responses, bridging the gap between the Models (data) and Views (UI).

### **Basic Methods in Controllers**

A controller in Laravel typically consists of methods that correspond to specific actions or operations for your application. Here’s an overview:

1. **Index Method,** index(): Displays a list of resources.
2. **Show Method,** show(): Displays details of a single resource.
3. **Create Method,** create(): Shows the form to create a new resource.
4. **Store Method,** store(): Handles form submission and stores data in the database.
5. **Edit Method,** edit(): Shows the form to edit an existing resource.
6. **Update Method,** update(): Handles the update of a specific resource.
7. **Destroy Method,** destroy(): Deletes a resource from the database.

### **Types of Operations in Controllers**

#### **1. CRUD Operations**

* Create: create() and store() methods handle adding new records.
* Read: index() and show() methods retrieve and display data.
* Update: edit() and update() methods modify existing data.
* Delete: destroy() method removes data.

#### **2. Filtering and Searching**

**3. Pagination**

**4. Validation**

**Blade templating basics.**

Blade is Laravel's built-in templating engine. It provides a clean, simple syntax for building dynamic and reusable UI components.

Blade templates use the **.blade.php** file extension.

**Control Structures:**

1. **Conditional Statements:**

@if, @elseif, @else and @endif

1. **Loops:**

@foreach, @for, @while, @endforeach and @endwhile

**Template inheritance:**

* a layout using **@yield** placeholders in a master template.
* Extend layouts in child views using **@extends** and fill sections using **@section** and **@endsection.**

**Components:**

Create reusable components using **@component.**

**Slot:**

pass data using **@slot.**

**Including Views:**

Use **@include** to embed other Blade views.

**Raw PHP**:

Embed PHP code using **@php ... @endphp.**

**Comments**:

Use **{{-- Comment --}}** for Blade comments (not rendered in HTML).

**Data Passing**:

Pass data from the Controller to the View using the **compact()** function or as an array.

**Directives**:

Use custom Blade directives or predefined ones like **@csrf** (for form tokens) and **@method** (for HTTP verbs like PUT, DELETE).

1. **Debugging**:

* Use **@dump($variable)** or **@dd($variable)** for debugging.