التكنولوجيا التطبيقية
APPLIED TECHNOLOGY
وزارة التربية والتعليم والتعليم الفني

MINISTRY OF EDUCATION AND TECHNICAL EDUCATION
وزارة التربية والتعليم والتعليم الفنى

مدارس we للتكنولوجيا التطبيقية

IT Department
Web Programming

Laravel Framework
Unit 21
الصف الثالث

Laravel Framework
Practical Exercises

1st.

2024 - 2025

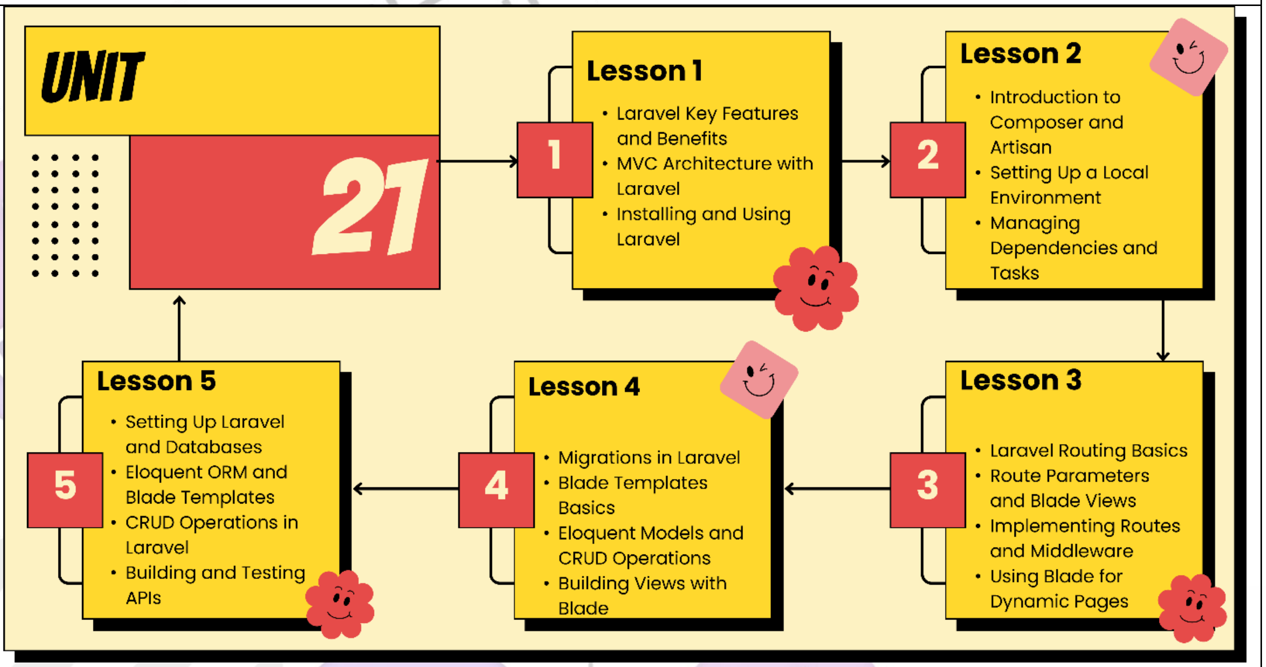
Laravel Framework
Unit 21
Practical Exercises

Unit	**21**
Name	**Laravel**
Goals / Outcomes	➢ **Remembering** 1. Recall the fundamental concepts and components of the Laravel framework, including its key features, tools, and directory structure. 2. Identify the purposes of core Laravel tools such as Composer, Artisan, Blade templates, and Eloquent ORM. 3. List the basic commands and syntax used for setting up Laravel projects, routes, controllers, and views. ➢ **Understanding** 1. Explain how Laravel integrates with PHP and its role in web application development. 2. Describe the MVC architecture and how Laravel implements this pattern. 3. Understand the purpose and usage of Laravel's routing system, Blade templating engine, and Eloquent ORM for managing database interactions. 4. Explain how Laravel handles dependencies and simplifies development tasks using Composer and Artisan commands. ➢ **Applying** 1. Install and configure a Laravel project, including setting up a local development environment and managing dependencies. 2. Write and implement Laravel scripts to manage routes, controllers, and views, and handle HTTP requests. 3. Utilize Laravel's Blade templating engine to create dynamic, reusable views. 4. Develop and manage database schemas using Laravel migrations and seeders. 5. Implement CRUD operations using Eloquent models to interact with databases. ➢ **Analyzing** 1. Compare Laravel with other PHP frameworks in terms of features, functionality, and performance.

	2. Analyze the benefits of using Laravel's routing, templating, and ORM systems for building scalable and maintainable applications. 3. Evaluate different methods for structuring code and organizing application components to enhance reusability and maintainability. 4. Assess the impact of using Laravel's tools (e.g., Composer, Artisan, Blade) on development efficiency and security. ➢ **Evaluating** 1. Evaluate the performance and efficiency of Laravel applications under different scenarios. 2. Judge the clarity, maintainability, and scalability of Laravel applications using best practices in coding and architecture. 3. Assess the robustness and security of Laravel applications, particularly in handling HTTP requests, data validation, and database interactions. ➢ **Creating** 1. Develop complete web applications using Laravel, demonstrating the integration of routes, controllers, models, and views. 2. Build dynamic web pages using Blade templates with layout inheritance, components, and conditional logic. 3. Create and manage database tables using migrations and implement RESTful APIs with Laravel to handle CRUD operations. 4. Design and implement scalable and maintainable web application structures following best practices in Laravel development.	
Knowledge	**Code**	**Description**
	TPK22	**Analyze and solve common web applications tasks by writing PHP programs**
Skill	**Code**	**Description**
	TPC5.7	**Using controllers and routes for APIs and URLs**
	TPC5.8	**Creating and using composer packages**

	TPC5.9	**Create restful services, use an Effect Dependency Array and how to hand errors in data requests**

Unit Preface


UNIT
21
1
Lesson 1
• Laravel Key Features and Benefits
• MVC Architecture with Laravel
• Installing and Using Laravel
2
Lesson 2
• Introduction to Composer and Artisan
• Setting Up a Local Environment
• Managing Dependencies and Tasks
3
Lesson 3
• Laravel Routing Basics
• Route Parameters and Blade Views
• Implementing Routes and Middleware
• Using Blade for Dynamic Pages
4
Lesson 4
• Migrations in Laravel
• Blade Templates Basics
• Eloquent Models and CRUD Operations
• Building Views with Blade
5
Lesson 5
• Setting Up Laravel and Databases
• Eloquent ORM and Blade Templates
• CRUD Operations in Laravel
• Building and Testing APIs

Create a Laravel CRUD (Create, Read, Update, Delete) application with image and search functionality for Simple E-commerce Product Management Application.

Create Laravel Project …………….

In Terminal :

```
composer create-project laravel/laravel lara-project
cd lara-project
code .
php artisan serve
```

Create Database lara-project and put it in .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lara-project
DB_USERNAME=root
DB_PASSWORD=
```

Go to the views folder from the resources folder and change the file name welcome.blade.php to home.blade.php and add any content

<html lang="en">

```
<head>
   <title>Home</title>
</head>
<body>
```

```
   <h1>Home Page</h1>
</body>
</html>
```

Go to the routes folder and in the web.php file put home is the default path

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

Add Bootstrap to the project

In the public folder, add a folder for css and another for js, and put the Bootstrap files inside it In the home.blade.php file:

```
   <title>Home</title>
   <link rel="stylesheet" href="{{ asset('css/bootstrap.min.css') }}">
</head>
<body>
   <h1 class="text-center">Home Page</h1>
   <script src="{{ asset('js/bootstrap.bundle.min.js') }}"></script>
</body>
```

Add a navbar from the Bootstrap site that contains the search and the logo
In the public folder, add a folder for images and put an image for the logo

In the home.blade.php file:

```
<body>
   <nav class="navbar navbar-expand-lg bg-body-tertiary">
      <div class="container ">
```

```
        <a class="navbar-brand" href="#"> <img src="{{
asset('images/logo.png') }}" height="50" alt=""> </a>
        <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse"
id="navbarSupportedContent">
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link active"  href="/">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#"> Products</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#" >Add Product</a>
            </li>
          </ul>
          <form class="d-flex" role="search">
            <input class="form-control me-2" type="search"
placeholder="Search" >
            <button class="btn btn-outline-primary"
type="submit">Search</button>
          </form>
        </div>
      </div>
    </nav>
  <h1 class="text-center mb-3 ">All Products </h1>
<section class="py-5">
      <div class="container">
          <div class="card p-3 mb-3">
              <h5 class="card-header"> - date</h5>
              <div class="card-body">
                <h5 class="card-title"> Product title</h5>
<p class="card-text"> text below as a natural lead-in to additional content.
</p>
                <a href="#" class="btn btn-primary">detail</a>
              </div> <!-- card-body -->
            </div> <!-- card -->
      </div> <!-- container  -->
    </section>
  <script src="{{ asset('js/bootstrap.bundle.min.js') }}"></script>
</body>
```

Create Product model with migration, controller, factory.

In Terminal:

```
php artisan make:model Product -mcrf
```

In Migration File:

```
Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->string('brand');
        $table->double('price');
```

```
            $table->string('image');
            $table->text('description');
        $table->foreignId('user_id')->constrained(table: 'users');
            $table->timestamps();
    });
```

Send migration to database as a table.

```
php artisan migrate
```

In Model File:

```
class Product extends Model{
   use HasFactory;
   // protected $fillable = ['title', 'brand','price', 'description' ] ;
}
```

Using Factory and Seeder

Note When not in use -f:

```
php artisan make:factory ProductFactory  --model=Product
```

In ProductFactory File:

```
      use App\Models\User;
return [
          'title' => fake()->sentence(),
          'brand' => fake()->word(),
          'price' => fake()->numberBetween(100,1000),
          'description' => fake()->paragraph(5),
          'image' => 'images/product.png',
         'user_id' => function () {
          return User::all()->random();
            }
     ];
```

In DatabaseSeeder File:

```
User::factory(5)->create();
\App\Models\Product::factory(50)->create();
```

In Terminal:

```
php artisan db:seed
```

Display Data in Views:

In ProductController add home() function:

```
   public function home(){
      $products = Product::all();
      // return view("home",compact("products"));
      return view("home",["products"=>$products]);
   }
```

In Web.php File:

```
use App\Http\Controllers\ProductController;
Route::get("/",[ProductController::class , 'home'] )->name('home');
```

In home.blade.php File:

```
        <div class="container">
@foreach ($products as $product)
<div class="card p-3 mb-3">
<h5 class="card-header">
Product : {{$product->id}} {{$product->created_at->format('Y-m-d')}}
</h5>
   <div class="card-body">
 <div class="row">
                  <div class="col-3">
<img src="{{ asset($product->image) }}"  class="img-fluid" alt="">
                  </div><!-- col-3 -->
                  <div class="col-9">
        <h3 class="card-title">{{$product->title}} </h3>
        <h5 class="card-title">{{$product->brand}} </h5>
<p class="card-text">{{\Str::limit($product->description , 100)}}</p>
        <a href="#" class="btn btn-primary">detail</a>
       </div><!-- col-9 -->
                </div><!-- row -->
</div> <!-- card-body -->
   </div> <!-- card -->
   @endforeach
      </div> <!-- container  -->
```

Using pagination

In ProductController in home() function :

```
   public function home(){
      $products = Product::paginate(5);
```

```
        return view("home",["products"=>$products]);
    }
```

In home.blade.php File:

```
    @endforeach
<div>{{$products->links()}}</div>
```

Using Bootstrap style for pagination

https://laravel.com/docs/11.x/pagination#using-bootstrap

in App\Providers\AppServiceProvider File:

```
use Illuminate\Pagination\Paginator;

    public function boot(): void{
        Paginator::useBootstrapFive();
    }
```

Using Blade Template

In the views add a folder named layouts and inside it a file named app.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>We Store |   @yield('title')  </title>
    <link rel="stylesheet" href="{{ asset('css/bootstrap.min.css') }}">
</head>
<body>

<nav>
……………………………………
</nav>

<section class="container py-5">
 @yield('content')
</section>
    <script src="{{ asset('js/bootstrap.bundle.min.js') }}"></script>
</body>
</html>
```

In home.blade.php we use @extends

```
@extends('layouts.app')
@section('title' , 'Home')
@section('content')
  @foreach ($products as $product)
                <div class="card p-3 mb-3">
                <h5 class="card-header">
                Product : {{$product->id}} {{$product->created_at->format('Y-
m-d')}} </h5>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-3">
```

```
<img src="{{ asset('images/'.$product->image) }}" class="img-fluid"
alt="">
                        </div><!-- col-3 -->
                        <div class="col-9">


                    <h3 class="card-title">{{$product->title}} </h3>
                    <h5 class="card-title">{{$product->brand}} </h5>
              <p class="card-text">{{\Str::limit($product->description ,
100)}}</p>
                    <a href="#" class="btn btn-primary">detail</a>
                  </div><!-- col-9 -->
              </div><!-- row -->
                 </div> <!-- card-body -->
                  </div> <!-- card -->
                  @endforeach
                  <div>{{$products->links()}}</div>
```

Working With Views

In Views Folder Add new folder products In it, add files (index,create,show,edit,search).blade.php

Add new Product

In Web.php File:

```
Route::get("/products/create",[ProductController::class , 'create'] )->name('products.create');
Route::post("/products/store",[ProductController::class , 'store'] ) ->name('products.store');
```

In create.blade.php

```
@extends('layouts.app')
@section('title' , 'Add New Product')
@section('content')

<div class="card w-50 mx-auto p-3 mb-3">
  <h5 class="card-header">Add New Product </h5>
  <div class="card-body">
     <form action="{{route('products.store')}}" method="post"
enctype="multipart/form-data" >
        @csrf
         <div class="mb-3">
           <label for="">Product Title</label>
           <input type="text" name="title" class="form-control" >
         </div> <!-- Title -->

         <div class="mb-3">
           <label for="">Product Brand</label>
       <input type="text" name="brand" class="form-control" >
         </div> <!-- Brand -->

         <div class="mb-3">
           <label for="">Product Price</label>
           <input type="text" name="price" class="form-control" >
         </div> <!-- Price -->

         <div class="mb-3">
           <label for="">Product Image</label>
```

```
            <input type="file" name="image" class="form-control form-control-file" >
          </div> <!-- Image -->

          <div class="mb-3">
            <label for="">Product Description</label>
            <textarea name="description" rows="3" class="form-control" ></textarea>
          </div> <!-- Description -->

          <div class="mb-3">
            <button type="submit" class="btn btn-success" >Add Product</button>
          </div> <!-- submit -->

        </form>
      </div> <!-- card-body -->
    </div> <!-- card -->
@endsection
```

In ProductController in create() function :

```
  public function create(){
      return view("products.create");
   }

   public function store(Request $request){
      return $request;
     //dd($request->all()) ;
   }
```

Add button In app.blade.php

```
     <section class="container py-5">
       <div  class="mb-3">
 <a href="/products/create" class="btn btn-success">
Add New Product </a>
       </div>
```

Edit Navbar links:

```
  <a class="navbar-brand" href="/"> <img src="{{
asset('images/logo.png') }}" height="50" alt=""> </a>

<a class="nav-link" href="{{ route('products.index')
}}">products</a>

<a class="nav-link" href="{{route('products.create')}}" >Add
product</a>
```

Working With Validation

```
 public function store(Request $request){
       $request->validate([
          'title'=>"required|string|min:5|max:120",
          'brand'=>'required','string', 'min:2',
          'price'=>'required','numeric','between:100,1000',
'image' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048']) ;
   }
```

https://laravel.com/docs/11.x/validation#quick-displaying-the-validation-errors

On the create page, above the form crown, we add the validation code to show errors.

```
            @if ($errors->any())
            <div class="alert alert-danger">
              <ul>
                 @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                 @endforeach
              </ul>
            </div>
        @endif
```

To keep old values we use old().

```
 <input type="text" name="title" value="{{old('title')}}" class="form-control" >
<input type="text" name="brand" value="{{old('brand')}}" class="form-control">
<input type="text" name="price" value="{{old('price')}}" class="form-control" >
<input type="file" name="image" class="form-control form-control-file" value="{{old('image')}}" >
<textarea name="description" rows="3" class="form-control" >
{{old('description')}} </textarea>
```

Insert into database

```
  public function store(Request $request){
     $request->validate([
        'title'=>"required|string|min:5|max:120",
        'brand'=>'required','string', 'min:2',
        'price'=>'required','numeric','between:100,1000',
'image' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048',
        ]) ;

$imageName = time().'.'.$request->image->extension();
$request->image->move(public_path('images'), $imageName);

     // $product->user_id =1;
$product->user_id = User::all()->random()->id;
        $product = new Product();
        $product->title = $request->title;
        $product->brand = $request->brand;
        $product->price = $request->price;
        $product->image = 'images/'.$imageName;
        $product->description = $request->description;
        $product->save();
        //return back()->with('success','Product Added Successfully');
        return redirect()->route('products.index')-
>with('success','Product Added Successfully');

    }//store
```

In index.blade.php

```
@extends('layouts.app')
@section('title' , 'All products')
@section('content')
<h2 class="text-center mb-3">All products</h2>
<table class="table table-bordered table-striped table-hover w-75 mx-
auto">
    <thead>
```

```
        <tr>
            <th class="text-center">ID</th>
            <th class="text-center">Title</th>
            <th class="text-center">Brand</th>
            <th class="text-center">Image</th>
            <th class="text-center">Show</th>
            <th class="text-center">Edit</th>
            <th class="text-center">Delete</th>
        </tr>
    </thead>
    <tbody>
        <tr>
             <td class="text-center">1</td>
            <td>Product one</td>
            <td class="text-center">Dell</td>
<td><img src="{{ asset('images/product.png') }}" width="60"
 alt=""></td>
<td class="text-center"><a href="#" class="btn btn-primary">
Show</a></td>
<td class="text-center"><a href="#" class="btn btn-warning">
Edit</a></td>
<td class="text-center"><a href="#" class="btn btn-danger">
Delete</a></td>
        </tr>
    </tbody>
</table>
@endsection
```

In Web.php File:

```
Route::get("/products",[ProductController::class , 'index'] ) -
>name('products.index');
```

In ProductController in index() function :

```
 public function index(){
     //$products = Product::all();
     $products = Product::paginate(5);
     return view("products.index",compact("products"));
   }
```

In index.blade.php

```
   <tbody>
@foreach ($products as $product)
        <tr>
              <td class="text-center">{{$loop->iteration}}</td>
             <td>{{$product->title}} </td>
             <td class="text-center">{{$product->brand}} </td>
<td><img src="{{ asset($product->image) }}" alt="{{ $product->name }}"
width="60"> </td>

  <td class="text-center"><a href="#" class="btn btn-primary">
Show</a></td>
             <td class="text-center"><a href="#" class="btn btn-warning">
Edit</a></td>
<td class="text-center"><a href="#" class="btn btn-danger">
Delete</a></td>
        </tr>
@endforeach
```

```
    </tbody>
</table>
<div>{{$products->links()}}</div>
@endsection
```

Add Flash message

In the index.blade.php file there is a message stating that the addition was successful.

```
<h2 class="text-center mb-3">All products</h2>
  @if (session('success'))
  <div class="alert alert-success text-center p-3" role="alert">
    <h3> {{ session('success') }}</h1>
  </div>
@endif
```

Show single Product

In web.php

```
Route::get("/products/show/{product}",[ProductController::class ,
'show'] ) ->name('products.show');
```

In ProductController in show() function :

```
   public function show(Product $product){
      return view("products.show",compact('product'));
   }
```

In home.blade.php

```
<a href="{{ route('products.show',$product->id) }}" class="btn btn-primary">detail</a>
```

In index.blade.php

```
<td class="text-center"><a href="{{ route('products.show',$product->id) }}" class="btn btn-primary"> Show</a></td>
```

In show.blade.php

```
@extends('layouts.app')
@section('title' , 'Home')
@section('content')
<div class="card p-3 mb-3">
   <h5 class="card-header">Product : {{$product->id}} {{$product->created_at->format('Y-m-d')}} </h5>
   <div class="card-body">
      <h3 class="card-title">{{$product->title}} </h3>
<img src="{{ asset($product->image) }}" alt="{{ $product->name }}" class="w-50 " >
      <h5 class="card-title">Brand : {{$product->brand}} </h5>
      <h5 class="card-title">Price : {{$product->price}} </h5>
      <p class="card-text">{{$product->description }}</p>
     </div> <!-- card-body -->
   </div> <!-- card -->
@endsection
```

Update Product

In web.php

```
Route::get("/products/edit/{product}",[ProductController::class,'edit']
) ->name('products.edit');
Route::put("/products/update/{product}",[ProductController::class ,
'update'] )->name('products.update');
```

In ProductController in edit() function :

```
  public function edit(Product $product){
      return view("products.edit",compact('product'));
  }
```

In index.blade.php

```
<td class="text-center"><a href="{{ route('products.edit',$product-
>id) }}" class="btn btn-warning"> Edit</a></td>
```

In edit.blade.php

```
@extends('layouts.app')
@section('title' , 'Add New Product')
@section('content')

<div class="card w-50 mx-auto p-3 mb-3">
    <h5 class="card-header">Add New Product </h5>
    @if (session('success'))
    <div class="alert alert-success text-center p-3" role="alert">
      <h1> {{ session('success') }}</h1>
    </div>
```

```
 @endif

    <div class="card-body">
        @if ($errors->any())
        <div class="alert alert-danger">
            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

<form action="{{route('products.update',$product->id)}}" method="post"
enctype='multipart/form-data' >
          @csrf
          @method('PUT')
           <div class="mb-3">
             <label for="">Product Title</label>
             <input type="text" name="title" value="{{$product->title}}"
class="form-control" >
           </div> <!-- Title -->

           <div class="mb-3">
             <label for="">Product Brand</label>
<input type="text" name="brand" value="{{$product->brand}}"
class="form-control" >
           </div> <!-- Brand -->
```

```
            <div class="mb-3">
              <label for="">Product Price</label>
<input type="text" name="price" value="{{$product->price}}"
class="form-control" >
            </div> <!-- Price -->

<div class="mb-3">
    <img src="{{ asset($product->image) }}" alt="{{ $product->name }}"
class="w-25 " >
</div> <!-- Image -->

<div class="mb-3">
<label for="">Product Image</label>
<input type="file" name="image"  class="form-control form-control-file" >
</div> <!-- Image -->

            <div class="mb-3">
<label for="">Product Description</label>
<textarea name="description" rows="3" class="form-control" >
{{$product->description}} </textarea>
            </div> <!-- Description -->

<div class="mb-3">
<button type="submit" class="btn btn-success" >Update Product</button>
</div> <!-- submit -->
        </form>
      </div> <!-- card-body -->
    </div> <!-- card -->
@endsection
```

In ProductController in update() function :

```
public function update(Request $request, Product $product)
   {
     // Handle the image upload if a new image is provided
   if ($request->hasFile('image')) {
      $imageName = time().'.'.$request->image->extension();
      $request->image->move(public_path(), $imageName);
      $product->image = $imageName;
   }else{
      $product->image = $product->image;
   }
```

// Update the product details

```
   $product->user_id = User::all()->random()->id;
   $product->title = $request->title;
   $product->brand = $request->brand;
   $product->price = $request->price;
   $product->description = $request->description;

   // Save the updated product
   $product->save();

   // Redirect back with a success message
   return redirect()->route('products.index')->with('success', 'Product
Updated Successfully');
   }//
```

In index.blade.php

```
@if (session('success'))
  <div class="alert alert-success text-center p-3" role="alert">
    <h3> {{ session('success') }}</h1>
  </div>
@endif
```

Delete Product

In web.php

```
Route::delete("/products/destroy/{product}",[ProductController::class
, 'destroy'] )->name('products.destroy');
```

In ProductController in destroy() function :

```
   public function destroy(Product $product){
      $product->delete();
      //return back()->with('success','Product deleted Successfully');
      return redirect()->route('products.index')->with('success','Product
deleted Successfully');
   }
```

In index.blade.php

```
          <td class="text-center">
<form action="{{route('products.destroy',$product->id)}}"
method="post">
                 @csrf
                 @method('DELETE')
<button type="submit" class="btn btn-danger" >Delete</button>
               </form>
```

```
        </td>
```

Search

In web.php

```
Route::post("/products/search",[ProductController::class , 'search'] )-
>name('products.search');
```

In app.blade.php

```
<form method="POST" action="{{ route('products.search') }}" class="d-
flex" role="search">
@csrf
<input name="query" type="search" placeholder="Search" class="form-
control me-2"  >
<button class="btn btn-outline-primary" type="submit">Search</button>
</form>
```

In ProductController in search () function :

```
   public function search(Request $request){
      $query = $request->input('query');
      $products = Product::where('title','like','%'.$query.'%')-
>orWhere('description','like','%'.$query.'%')->get();
   //    return  $products;
   return view('products.search',compact('products'));
   }
```

In search.blade.php

The content of the search page is the same as the content of the home page without the pagination links

```
@extends('layouts.app')
@section('title' , 'Home')
@section('content')
@foreach ($products as $product)

<div class="card p-3 mb-3">
   <h5 class="card-header">Product : {{$product->id}} {{$product->created_at->format('Y-m-d')}} </h5>
   <div class="card-body">
<img src="{{ asset($product->image) }}" alt="{{ $product->name }}" class="w-50 " >

      <h3 class="card-title">{{$product->title}} </h3>
      <h5 class="card-title">{{$product->brand}} </h5>
      <p class="card-text">{{\Str::limit($product->description , 100)}}</p>
      <a href="{{ route('products.show',$product->id) }}" class="btn btn-primary">detail</a>
     </div> <!-- card-body -->
   </div> <!-- card -->
   @endforeach
@endsection
```

Notes

From phpmyadmin you can change the database name and table name

Ctrl+Alt+I inserts the namespace

Add fill dummy data to the browser fill all inputs extension

How To Make Laravel 11 REST API ?

Contents

Step 1: Install Laravel 11 using Composer

Run this command on Terminal or CMD to install:

```
composer create-project laravel/laravel laravel-11-rest-api
```

or via Laravel Installer:

```
laravel new laravel-11-rest-api
```

Step 2: Setup Database Configuration

Inside the project root folder open the file .env and put the configuration for the database.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your database name(laravel_11_rest_api)
```

DB_USERNAME=your database username(root)
DB_PASSWORD=your database password(root)

Step 3: Create a Model with Migration

A model is a class that represents a table on a database.

Migration is like a version of your database.

Run this command on Terminal or CMD:

php artisan make:model Project --migration

After running this command you will find a file in this path *"database/migrations"* and update the code in that file.

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('projects', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('projects');
    }
};
```

Run the migration by executing the migrate Artisan command:

```
php artisan migrate
```

Step 4: Enable API and Create an API Resource Controller

By default, laravel 11 API route is not enabled in laravel 11. We will enable the API:

```
php artisan install:api
```

After we enable the API, we will now create our controller. The controller will be responsible for handling HTTP incoming requests.

Run this command to create an API Resource Controller:

```
php artisan make:controller ProjectController --api
```

This command will generate a controller at ***"app/Http/Controllers/ProjectController.php"***. It contains methods for each of the available resource operations. Open the file and insert these codes:

app/Http/Controllers/ProjectController.php

```
<?php
```

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Project;
class ProjectController extends Controller{

 public function index() {
      $projects = Project::get();
      return response()->json($projects);
   }

   public function store(Request $request){
      $project = new Project();
      $project->name = $request->name;
      $project->description = $request->description;
      $project->save();
      return response()->json($project);
   }

   public function show(string $id){
      $project = Project::find($id);
      return response()->json($project);
   }

   public function update(Request $request, string $id)
   {
      $project = Project::find($id);
      $project->name = $request->name;
```

```
        $project->description = $request->description;
        $project->save();
        return response()->json($project);
    }

    public function destroy(string $id){
        Project::destroy($id);
        return response()->json(['message' => 'Deleted']);
    }
}
```

Step 5: Add an API resource route

We will be using the route file **routes/api.php** since we are creating an API. The routes inside **routes/api.php** are stateless and use the **API** middleware group.

When creating an API resource route, you must use the ***apiResource*** method to exclude the route that represents ***create*** and ***edit*** html templates.

Now we register the API resource routes:

routes/api.php

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProjectController;
```

```
Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');

Route::apiResource('projects', ProjectController::class);
```

Step 6: Run the Laravel App

Run this command to start the Laravel App:

```
php artisan serve
```

After successfully running your app, open this URL in your browser:

http://localhost:8000

Test the API:

We will be using Postman for testing our API, but you can use you preferred tool.