Simple Laravel 11 CRUD Application Tutorial

In this tutorial, we will learn about **Laravel 11 CRUD application** by developing a complete **Laravel 11 CRUD application system** with step by step and simple guide.

Although, this CRUD operations tutorial is for Laravel 11. However, it will also work on the previous versions of Laravel such as Laravel 10, Laravel 9 and Laravel 8 as well.

In this example, we will use the **Laravel 11** with the latest **Bootstrap 5.3.2** version.

To install Laravel 11, you will need at least **PHP 8.2** version, if you do not have PHP installed then you can download them from their respective source links.

PHP – https://windows.php.net
Apache – https://apachelounge.com
MySQL and Workbench – https://dev.mysql.com
Composer – https://getcomoser.org

Prerequisites of Laravel 11 CRUD App

To develop Laravel 11 CRUD application, ensure that you have already installed the following:

1. PHP 8.2 installed (you can check your php version by typing php –v on the commad line)
2. Apache Server is installed and configured to load php.
3. MySQL database server.
4. MySQL Workbench
5. Composer

Application Flow:

In this Laravel 11 CRUD example, we will create a simple **products** table and perform all CRUD operations in it.

In this tutorial, we will store all product details in 5 columns which are product **code** (varchar), **name** (varchar), **quantity** (int), **price** (decimal) and **description** (text).

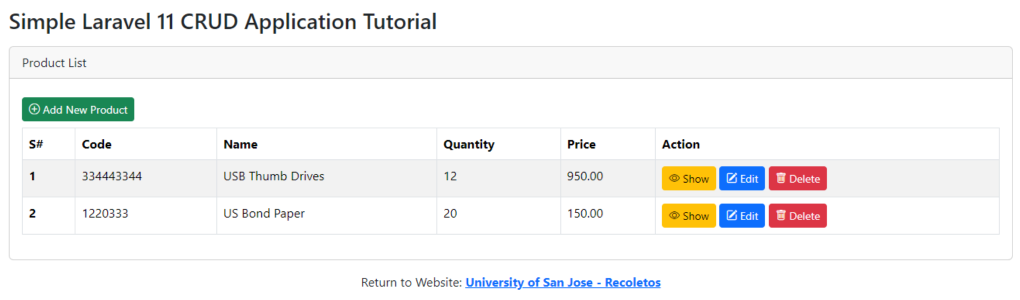
We can use 1 or 2 column table but we are using 5 columns tables intentionally to show you how to add different data types of records in table.

So that you can understand how to create migration of different data types and store data of these specific data types.

We will add, view, update and delete products from database.

Following are the screenshots of the **Laravel 11 CRUD application** which store, view, update and delete product from **products** table.

Product List Page



Simple Laravel 11 CRUD Application Tutorial

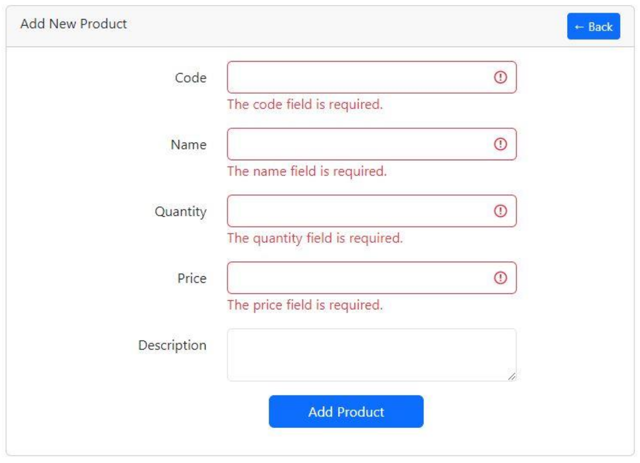
Product List

Add New Product

S#	Code	Name	Quantity	Price	Action
1	334443344	USB Thumb Drives	12	950.00	Show Edit Delete
2	1220333	US Bond Paper	20	150.00	Show Edit Delete

Return to Website: University of San Jose - Recoletos

Add New Product Page

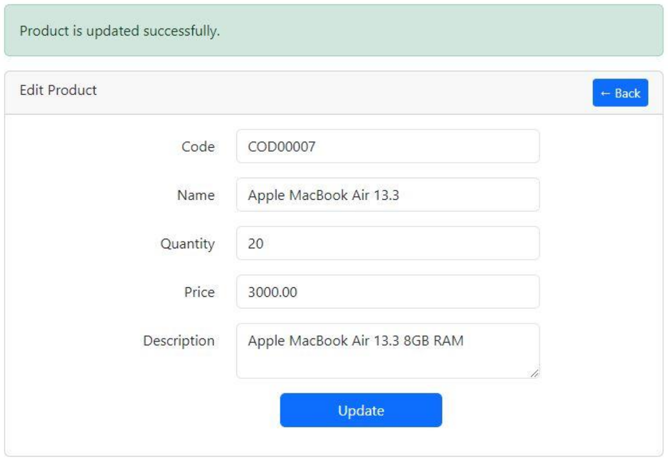


Add New Product ← Back

Code	The code field is required.
Name	The name field is required.
Quantity	The quantity field is required.
Price	The price field is required.
Description	

Add Product

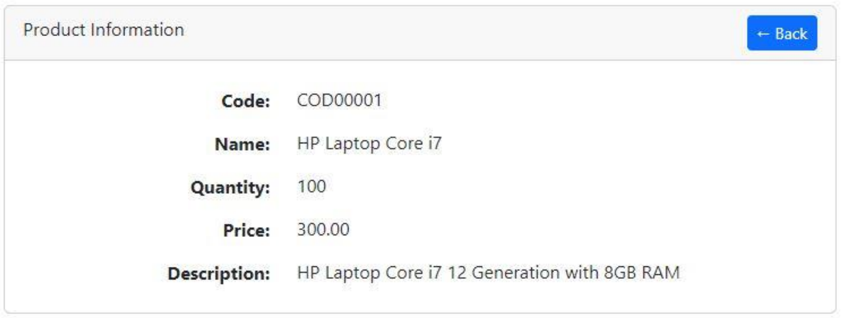
Edit Product Page


Product is updated successfully.

Edit Product

Code	COD00007
Name	Apple MacBook Air 13.3
Quantity	20
Price	3000.00
Description	Apple MacBook Air 13.3 8GB RAM

Show Product Page


Product Information

Code:	COD00001
Name:	HP Laptop Core i7
Quantity:	100
Price:	300.00
Description:	HP Laptop Core i7 12 Generation with 8GB RAM

Steps to Create Laravel 11 CRUD Application

Follow the below step by step guide to create a Laravel 11 CRUD application.

1. Install Laravel 11 App
2. Create and Configure Database Credentials
3. Create a Model with Migration, Resource Controller and Requests for Validation
4. Update Product Migration
5. Migrate Tables to Database
6. Define Product Resource Routes
7. Update Code in Product Model
8. Update Code in Product Controller
9. Update Code in Product Store and Update Requests
10. Enable Bootstrap 5 in AppServiceProvider
11. Create Layout and Product Resource Blade View Files

12. Run Laravel Development Server

1. Install Laravel 11 App

Open the command prompt/terminal window and go to the directory where you want to install the Laravel 11 CRUD application. For example like below.

```
D: <Enter>

cd \ <Enter>

md laravel <Enter>

cd laravel <Enter>
```

Your current directory should look like below.

```
D:\laravel>
```

Now install the Laravel 11 application with name **laravel_11_crud** by running the below command on your terminal window or command prompt.

```
composer create-project laravel/laravel laravel_11_crud --prefer-dist
```

After Laravel 11 installation, navigate to the newly created directly by running the below command.

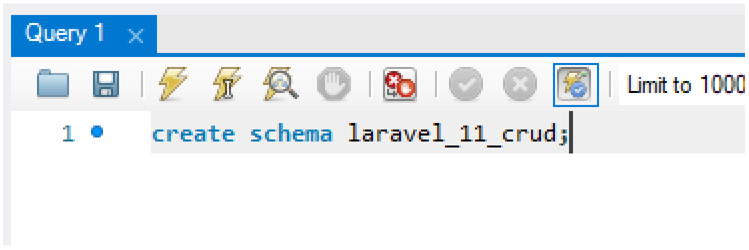
```
cd laravel_11_crud <Enter>

D:\laravel\laravel_11_crud>
```

Open your project folder using **Visual Studio Code**.

2. Create and Configure Database Credentials

In this step, first we need to create a database and then configure database credentials for Laravel 11 CRUD application.

Open your MySQL Workbench. Once you have connected to your Database server and in the query box type in the following command.

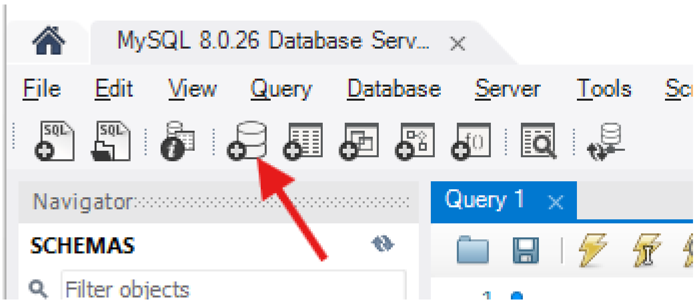


```
create schema laravel_11_crud;
```

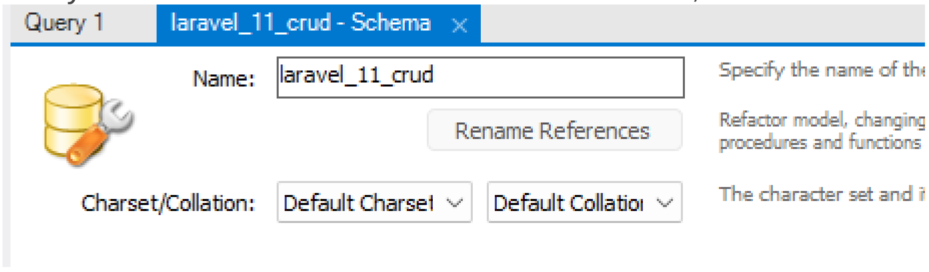
This will create the schema that will be used for the app that we will be creating.

Alternatively, you can also select the creation wizard as follows:

1. Click on the create new schema icon.



2. The you will be asked to enter the schema name,



Query 1 | laravel_11_crud - Schema

Name: laravel_11_crud

Rename References

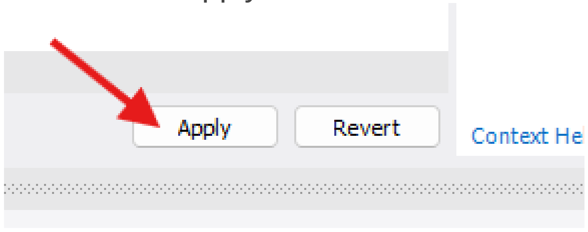
Charset/Collation: Default Charset | Default Collation

Specify the name of the

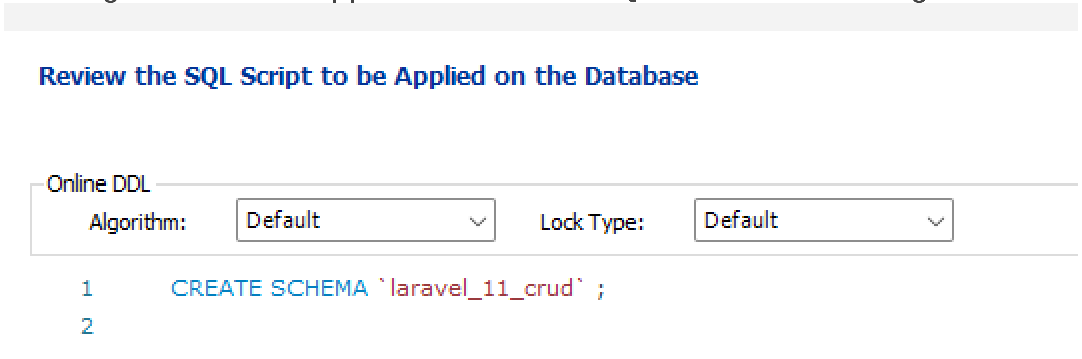
Refactor model, changing procedures and functions

The character set and i

3. Click on the Apply button.



4. A dialog box will then appear to show the SQL statement that is generated.

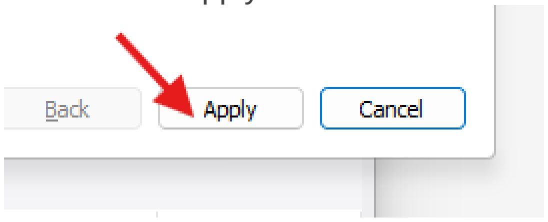


Review the SQL Script to be Applied on the Database

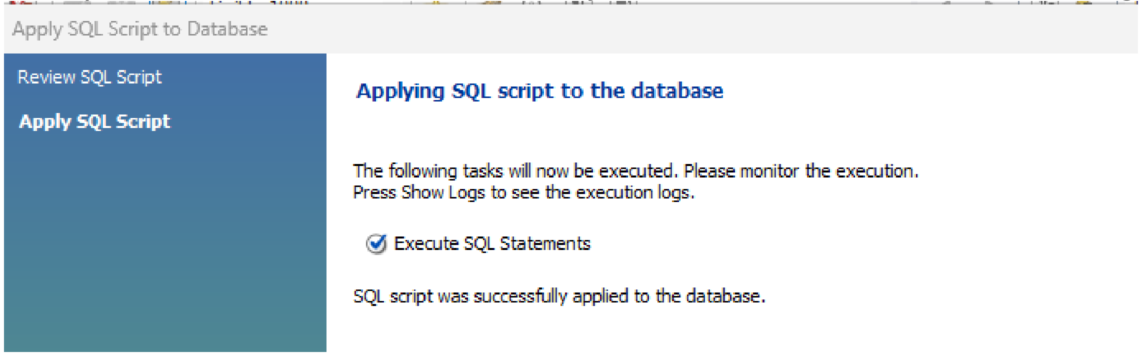
Online DDL
Algorithm: Default   Lock Type: Default

```
CREATE SCHEMA `laravel_11_crud` ;
```

5. Click on the Apply button.



6. It will proceed to create the schema then you will receive a confirmation message



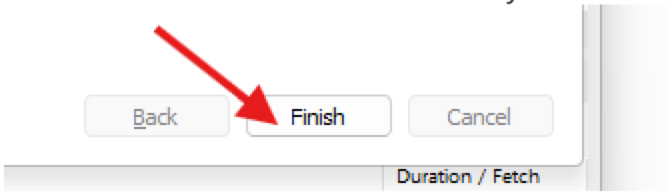
Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

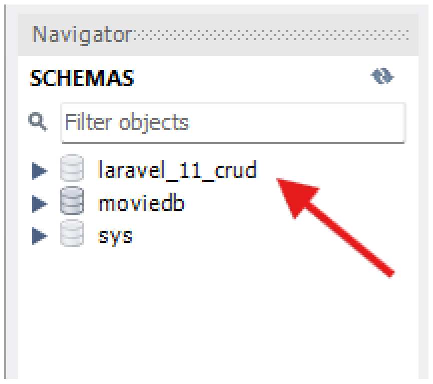
✓ Execute SQL Statements

SQL script was successfully applied to the database.

7. Click on the Finish button and then you are done.



You should now see the new schema on the schemas pane of Workbench.



After creating the schema we will now proceed to configuring Laravel to connect to our database server, the specified schema and the user credentials needed.

From you editor look for the .env file and open it in your editor window and enter your own database credentials details.

The default user of MySQL in computer system is **root** and **password** is empty.

Database Credentials:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your_db_name
DB_USERNAME=your_db_username
DB_PASSWORD=your_db_password
DB_COLLATION=utf8mb4_unicode_ci
```

By default Laravel 11 comes with the database collation as **utf8mb4_0900_ai_ci**, which might create an issue when migrating tables to database therefore we also added the **DB_COLLATION** with value as **utf8mb4_unicode_ci** in our database credentials to avoid errors.

3. Create a Model with Migration, Resource Controller and Requests for Validation

In this step, we need to create a Model, Migration, Resource Controller and Requests for Validation of **products** table.

Although, we can create them individually, but we can also create them using a single artisan command.

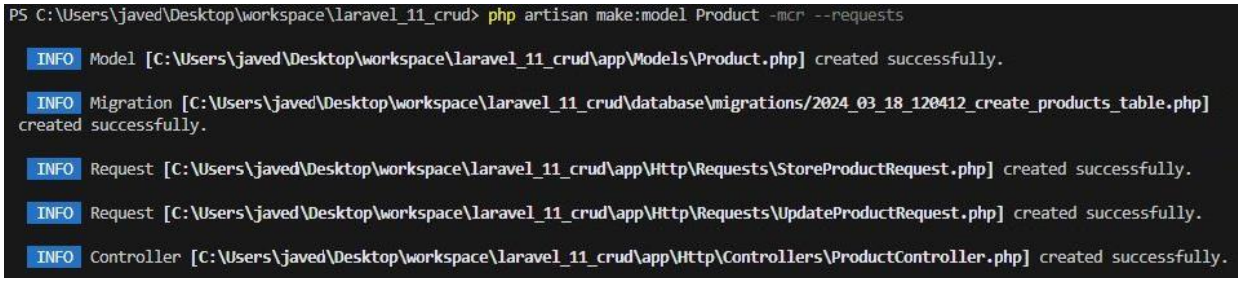
Just run the below command on your terminal:

```
php artisan make:model Product -mcr --requests
```

In the above command **–m** flag refers to a migration of the Model, **cr** flag refers to a resource controller, and **— requests** flag refers to the custom requests for resource controller.

It will create two requests, store and update requests for validation.

You will get the following message on your terminal after running the above command.



```
PS C:\Users\javed\Desktop\workspace\laravel_11_crud> php artisan make:model Product -mcr --requests

  INFO  Model [C:\Users\javed\Desktop\workspace\laravel_11_crud\app\Models\Product.php] created successfully.

  INFO  Migration [C:\Users\javed\Desktop\workspace\laravel_11_crud\database\migrations/2024_03_18_120412_create_products_table.php]
 created successfully.

  INFO  Request [C:\Users\javed\Desktop\workspace\laravel_11_crud\app\Http\Requests\StoreProductRequest.php] created successfully.

  INFO  Request [C:\Users\javed\Desktop\workspace\laravel_11_crud\app\Http\Requests\UpdateProductRequest.php] created successfully.

  INFO  Controller [C:\Users\javed\Desktop\workspace\laravel_11_crud\app\Http\Controllers\ProductController.php] created successfully.
```

4. Update Product Migration

Now, we need to update a product migration file, just go to the directory **laravel_11_crud\database\migrations** and there you will see the product migration file like below.

YYYY_MM_DD_TIMESTAMP_create_products_table.php

Open this file and paste the following code in it.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration

{
    /**
      * Run the migrations.
      */

    public function up(): void {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('code')->unique();
            $table->string('name');
            $table->integer('quantity');
            $table->decimal('price', 8, 2);
            $table->text('description')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('products');
    }
};
```

5. Migrate Tables to Database

Once the product migration file is updated, now we need to migrate all tables into our database.

Run the below artisan command on the terminal to migrate all tables into database.

```
php artisan migrate
```

6. Define Product Resource Routes

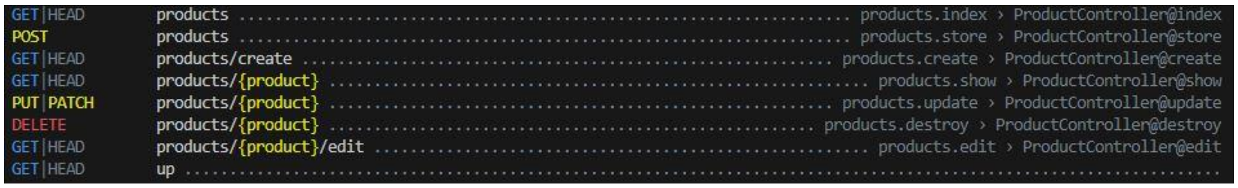
Now in this step, we will need to define our application product resource routes in **web.php**. Just copy and paste the below code in your **routes/web.php** file.

```
<?php

use Illuminate\Support\Facades\Route;
```

```php
use App\Http\Controllers\ProductController;

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

Route::resource('products', ProductController::class);
```

After defining the application routes, check the application routes by running the below artisan command on the terminal.

By running above artisan command, you will see all your routes like below screenshot:



```
GET|HEAD    products ............................................................. products.index › ProductController@index
POST        products ............................................................. products.store › ProductController@store
GET|HEAD    products/create ................................................... products.create › ProductController@create
GET|HEAD    products/{product} ..................................................... products.show › ProductController@show
PUT|PATCH   products/{product} ................................................. products.update › ProductController@update
DELETE      products/{product} ............................................... products.destroy › ProductController@destroy
GET|HEAD    products/{product}/edit ................................................ products.edit › ProductController@edit
GET|HEAD    up ................................................................................................................
```

7. Update Code in Product Model

Now, we need to allow mass assignment in Product Model file, just go to the **app\Models\Product.php** and update the following code in **Product.php** file.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;

    protected $fillable = [
        'code',
        'name',
        'quantity',
        'price',
        'description'
    ];
}
```

In the above code, we are passing column names to protected **$fillable** array to enable mass assignment of these fields, otherwise Laravel will not add records into **products** table if we use the Laravel Eloquent to add data.

8. Update Code in Product Controller

Now in this step, we need to update our **Product Resource Controller** which contains **seven (7)** different methods to perform Laravel eloquent CRUD operations in our Laravel 11 CRUD application.

By default, the resource controller comes with the following methods. We will use these all methods for our Laravel 11 CRUD App.

1. **index()** -> To list or display all of our products from products table.
2. **create()** -> To display add new product form.
3. **store()** -> To store our new product form request data into our products table.
4. **show()** -> To display a single product with details.
5. **edit()** -> To display a product edit form.
6. **update()** -> To update the product data into our products table.
7. **destroy()** -> To delete the product from the database.

Simply copy and paste the below code in **app\Http\Controllers\ProductController.php** file.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use App\Http\Requests\StoreProductRequest;
use App\Http\Requests\UpdateProductRequest;
use Illuminate\View\View;
use Illuminate\Http\RedirectResponse;

class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index() : View
    {
        return view('products.index', [
            'products' => Product::latest()->paginate(4)
        ]);
    }
    /**
     * Show the form for creating a new resource.
     */
    public function create() : View
    {
        return view('products.create');
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StoreProductRequest $request) :
RedirectResponse
    {
        Product::create($request->validated());

        return redirect()->route('products.index')
                ->withSuccess('New product is added successfully.');
    }

    /**
     * Display the specified resource.
     */
    public function show(Product $product) : View
    {
        return view('products.show', compact('product'));
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Product $product) : View
    {
        return view('products.edit', compact('product'));
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(UpdateProductRequest $request, Product
$product) : RedirectResponse
    {
        $product->update($request->validated());

        return redirect()->back()
                ->withSuccess('Product is updated successfully.');
```

```php
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Product $product) : RedirectResponse
    {
        $product->delete();

        return redirect()->route('products.index')
                ->withSuccess('Product is deleted successfully.');
    }
}
```

As you can see that the above code is self explanatory. However, an additional comment is added with each method to explain the working of that method.

9. Update Code in Product Store and Update Requests

In this step, we will update code in our product **store** and **update** requests.

Copy and paste the below code in **app\Http\Requests\StoreProductRequest.php** file.

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreProductRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string,
\Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'code' => 'required|string|max:50|unique:products,code',
            'name' => 'required|string|max:250',
            'quantity' => 'required|integer|min:1|max:10000',
            'price' => 'required',
            'description' => 'nullable|string'
        ];
    }
}
```

After that copy and paste the below code in **app\Http\Requests\UpdateProductRequest.php** file.

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateProductRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string,
\Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'code' =>
'required|string|max:50|unique:products,code,'.$this->product->id,
            'name' => 'required|string|max:250',
            'quantity' => 'required|integer|min:1|max:10000',
            'price' => 'required',
            'description' => 'nullable|string'
        ];
    }
}
```

These both form request files are responsible to validate data before adding and updating record in our database’s **products** table.

Alternatively, we can also validate data in our controller class but as a good practice, we should follow the SOLID principle and make a separate validation classes to follow the single responsibility principle.

10. Enable Bootstrap 5 in AppServiceProvider

Since, we are using **Bootstrap v5.3.2** via CDN but some of its feature will not work such as pagination until you explicitly call it in **App\Providers\AppServiceProviders.php** file.

We need to call **Paginator::useBootstrapFive();** in **boot()** method, all we need is to copy and paste the below code in **AppServiceProviders.php** file.

```php
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Pagination\Paginator;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     */
    public function register(): void
    {
        //
    }

    /**
     * Bootstrap any application services.
     */
    public function boot(): void
    {
        Paginator::useBootstrapFive();
    }
}
```

11. Create Layout and Product Resource Blade View Files

In this step, we need to create **/layouts** and **/products** directories in **resources/views/** directory and then create blade files in it.

We can create them either manually or using PHP artisan command. We will create them using artisan commands.

So just run the below artisan commands and all 5 blade view files will be created in the relevant directory.

I stand corrected on this part due to the fact that in the earlier versions of Laravel make:view was not yet available as an artisan feature.

```
php artisan make:view layouts.app
php artisan make:view products.index
php artisan make:view products.create
php artisan make:view products.edit
php artisan make:view products.show
```

The above commands will create the following files:

1. app.blade.php
2. index.blade.php
3. create.blade.php
4. edit.blade.php
5. show.blade.php

Now, we need to update each blade view files with the following code.

app.blade.php is an application main layout view file of Laravel 11 CRUD, just copy and paste the below code in the file **resources/views/layouts/app.blade.php**

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<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>Simple Laravel 11 CRUD Application Tutorial</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.
min.css">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.1/font/bootstrap-icons.css">
</head>
<body>

    <div class="container">
        <h3 class=" mt-3">Simple Laravel 11 CRUD Application
Tutorial</h3>
            @yield('content')
            <div class="row justify-content-center text-center mt-3">
                <div class="col-md-12">
                  <p>
                      Return to Website: <a
href="https://www.usjr.edu.ph/"><strong>University of San Jose -
Recoletos</strong></a>
                  </p>

                </div>
            </div>
    </div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bu
ndle.min.js"></script>
</body>
</html>
```

index.blade.php is the Laravel 11 CRUD application main landing page which will list all products from database table along with pagination.

Copy and paste the below code in the file **resources/views/products/index.blade.php**

```
@extends('layouts.app')

@section('content')

<div class="row justify-content-center mt-3">
    <div class="col-md-12">

        @session('success')
            <div class="alert alert-success" role="alert">
                {{ $value }}
            </div>
        @endsession

        <div class="card">
            <div class="card-header">Product List</div>
            <div class="card-body">
                <a href="{{ route('products.create') }}" class="btn
btn-success btn-sm my-2"><i class="bi bi-plus-circle"></i> Add New
Product</a>
                <table class="table table-striped table-bordered">
                    <thead>
                      <tr>
                        <th scope="col">S#</th>
                        <th scope="col">Code</th>
                        <th scope="col">Name</th>
                        <th scope="col">Quantity</th>
                        <th scope="col">Price</th>
                        <th scope="col">Action</th>
                      </tr>
                    </thead>
                    <tbody>
                        @forelse ($products as $product)
                        <tr>
                            <th scope="row">{{ $loop->iteration
}}</th>
                            <td>{{ $product->code }}</td>
                            <td>{{ $product->name }}</td>
                            <td>{{ $product->quantity }}</td>
                            <td>{{ $product->price }}</td>
                            <td>
                                <form action="{{
route('products.destroy', $product->id) }}" method="post">
                                    @csrf
                                    @method('DELETE')

                                    <a href="{{ route('products.show',
$product->id) }}" class="btn btn-warning btn-sm"><i class="bi bi-
eye"></i> Show</a>

                                    <a href="{{ route('products.edit',
$product->id) }}" class="btn btn-primary btn-sm"><i class="bi bi-
pencil-square"></i> Edit</a>

                                    <button type="submit" class="btn
btn-danger btn-sm" onclick="return confirm('Do you want to delete this
product?');"><i class="bi bi-trash"></i> Delete</button>
```

```
                                </form>
                            </td>
                        </tr>
                        @empty
                            <td colspan="6">
                                <span class="text-danger">
                                    <strong>No Product Found!</strong>
                                </span>
                            </td>
                        @endforelse
                    </tbody>
                  </table>

                  {{ $products->links() }}

            </div>
        </div>
    </div>
</div>

@endsection
```

create.blade.php is a file to add new product blade view file, just copy and paste the below code in the file **resources/views/products/create.blade.php**

```
@extends('layouts.app')

@section('content')

<div class="row justify-content-center mt-3">
    <div class="col-md-8">

        <div class="card">
            <div class="card-header">
                <div class="float-start">
                    Add New Product
                </div>
                <div class="float-end">
                    <a href="{{ route('products.index') }}" class="btn
btn-primary btn-sm">&larr; Back</a>
                </div>
            </div>
            <div class="card-body">
                <form action="{{ route('products.store') }}"
method="post">
```

```
                    @csrf

                    <div class="mb-3 row">
                        <label for="code" class="col-md-4 col-form-
label text-md-end text-start">Code</label>
                        <div class="col-md-6">
                          <input type="text" class="form-control
@error('code') is-invalid @enderror" id="code" name="code" value="{{
old('code') }}">
                            @error('code')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <label for="name" class="col-md-4 col-form-
label text-md-end text-start">Name</label>
                        <div class="col-md-6">
                          <input type="text" class="form-control
@error('name') is-invalid @enderror" id="name" name="name" value="{{
old('name') }}">
                            @error('name')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <label for="quantity" class="col-md-4 col-
form-label text-md-end text-start">Quantity</label>
                        <div class="col-md-6">
                          <input type="number" class="form-control
@error('quantity') is-invalid @enderror" id="quantity" name="quantity"
value="{{ old('quantity') }}">
                            @error('quantity')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
```

```
                        <label for="price" class="col-md-4 col-form-
label text-md-end text-start">Price</label>
                        <div class="col-md-6">
                          <input type="number" step="0.01"
class="form-control @error('price') is-invalid @enderror" id="price"
name="price" value="{{ old('price') }}">
                            @error('price')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <label for="description" class="col-md-4 col-
form-label text-md-end text-start">Description</label>
                        <div class="col-md-6">
                            <textarea class="form-control
@error('description') is-invalid @enderror" id="description"
name="description">{{ old('description') }}</textarea>
                            @error('description')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <input type="submit" class="col-md-3 offset-
md-5 btn btn-primary" value="Add Product">
                    </div>

                </form>
            </div>
        </div>
    </div>
</div>

@endsection
```

edit.blade.php is a product edit view file, just copy paste the below code in the file **resources/views/products/edit.blade.php**

```
@extends('layouts.app')

@section('content')
```

```
<div class="row justify-content-center mt-3">
    <div class="col-md-8">

        @session('success')
            <div class="alert alert-success" role="alert">
                {{ $value }}
            </div>
        @endsession

        <div class="card">
            <div class="card-header">
                <div class="float-start">
                    Edit Product
                </div>
                <div class="float-end">
                    <a href="{{ route('products.index') }}" class="btn
btn-primary btn-sm">&larr; Back</a>
                </div>
            </div>
            <div class="card-body">
                <form action="{{ route('products.update', $product-
>id) }}" method="post">
                    @csrf
                    @method("PUT")

                    <div class="mb-3 row">
                        <label for="code" class="col-md-4 col-form-
label text-md-end text-start">Code</label>
                        <div class="col-md-6">
                          <input type="text" class="form-control
@error('code') is-invalid @enderror" id="code" name="code" value="{{
$product->code }}">
                            @error('code')
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <label for="name" class="col-md-4 col-form-
label text-md-end text-start">Name</label>
                        <div class="col-md-6">
```

```
             <input type="text" class="form-control
@error('name') is-invalid @enderror" id="name" name="name" value="{{
$product->name }}">
              @error('name')
                <span class="text-danger">{{ $message
}}</span>
              @enderror
            </div>
          </div>

          <div class="mb-3 row">
            <label for="quantity" class="col-md-4 col-
form-label text-md-end text-start">Quantity</label>
            <div class="col-md-6">
             <input type="number" class="form-control
@error('quantity') is-invalid @enderror" id="quantity" name="quantity"
value="{{ $product->quantity }}">
              @error('quantity')
                <span class="text-danger">{{ $message
}}</span>
              @enderror
            </div>
          </div>

          <div class="mb-3 row">
            <label for="price" class="col-md-4 col-form-
label text-md-end text-start">Price</label>
            <div class="col-md-6">
             <input type="number" step="0.01"
class="form-control @error('price') is-invalid @enderror" id="price"
name="price" value="{{ $product->price }}">
              @error('price')
                <span class="text-danger">{{ $message
}}</span>
              @enderror
            </div>
          </div>

          <div class="mb-3 row">
            <label for="description" class="col-md-4 col-
form-label text-md-end text-start">Description</label>
            <div class="col-md-6">
              <textarea class="form-control
@error('description') is-invalid @enderror" id="description"
name="description">{{ $product->description }}</textarea>
              @error('description')
```

```
                                <span class="text-danger">{{ $message
}}</span>
                            @enderror
                        </div>
                    </div>

                    <div class="mb-3 row">
                        <input type="submit" class="col-md-3 offset-
md-5 btn btn-primary" value="Update">
                    </div>

                </form>
            </div>
        </div>
    </div>
</div>

@endsection
```

show.blade.php is a single product view file, simply copy paste the below code in the file **resources/views/products/show.blade.php**

```
@extends('layouts.app')

@section('content')

<div class="row justify-content-center mt-3">
    <div class="col-md-8">

        <div class="card">
            <div class="card-header">
                <div class="float-start">
                    Product Information
                </div>
                <div class="float-end">
                    <a href="{{ route('products.index') }}" class="btn
btn-primary btn-sm">&larr; Back</a>
                </div>
            </div>
            <div class="card-body">

                    <div class="row">
                        <label for="code" class="col-md-4 col-form-
label text-md-end text-start"><strong>Code:</strong></label>
```

```
                        <div class="col-md-6" style="line-height:
35px;">
                            {{ $product->code }}
                        </div>
                    </div>

                    <div class="row">
                        <label for="name" class="col-md-4 col-form-
label text-md-end text-start"><strong>Name:</strong></label>
                        <div class="col-md-6" style="line-height:
35px;">
                            {{ $product->name }}
                        </div>
                    </div>

                    <div class="row">
                        <label for="quantity" class="col-md-4 col-
form-label text-md-end text-start"><strong>Quantity:</strong></label>
                        <div class="col-md-6" style="line-height:
35px;">
                            {{ $product->quantity }}
                        </div>
                    </div>

                    <div class="row">
                        <label for="price" class="col-md-4 col-form-
label text-md-end text-start"><strong>Price:</strong></label>
                        <div class="col-md-6" style="line-height:
35px;">
                            {{ $product->price }}
                        </div>
                    </div>

                    <div class="row">
                        <label for="description" class="col-md-4 col-
form-label text-md-end text-
start"><strong>Description:</strong></label>
                        <div class="col-md-6" style="line-height:
35px;">
                            {{ $product->description }}
                        </div>
                    </div>

            </div>
        </div>
    </div>
```

```
</div>

@endsection
```

12. Run Laravel Development Server

Finally we have completed all the steps of our **Laravel 11 CRUD application example tutorial**, now it is the time to test the application.

Just run the Laravel development server by running the below artisan command.

```
php artisan serve
```

After starting the development server, visit the below link to test our Laravel 11 CRUD application.

```
http://127.0.0.1:8000/products
```