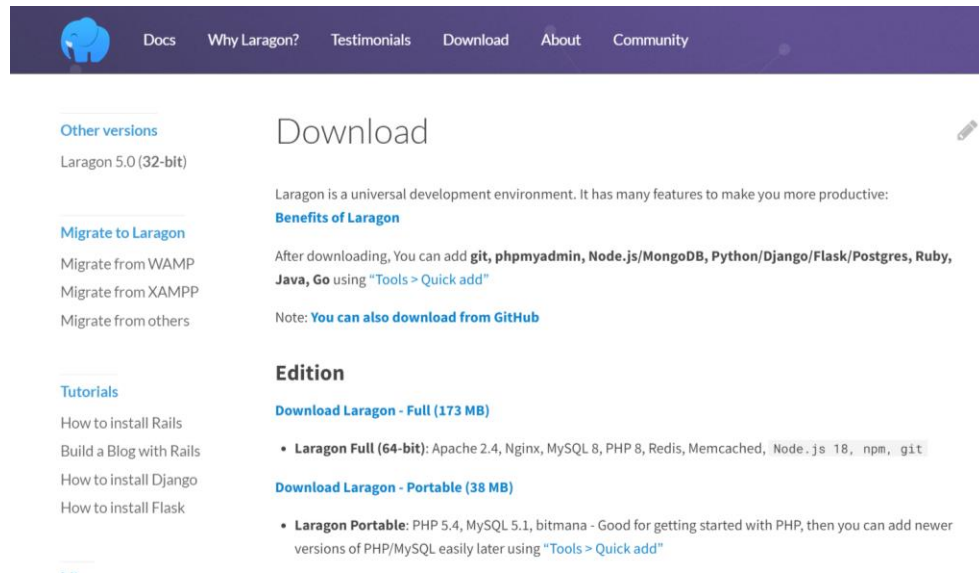


LARAVEL TUTORIAL

Step 1 – Installing Laragon

Go on Laragon's [download site](#) and download the latest version of the appropriate installer for your version of Windows. It is advisable to download the full edition of Laragon.



The screenshot shows the Laragon website's 'Download' page. The page has a dark purple header with the Laragon logo and navigation links: Docs, Why Laragon?, Testimonials, Download, About, and Community. The main content area is white. On the left, there are three sections: 'Other versions' (Laragon 5.0 (32-bit)), 'Migrate to Laragon' (Migrate from WAMP, XAMPP, and others), and 'Tutorials' (How to install Rails, Build a Blog with Rails, How to install Django, and How to install Flask). The main section is titled 'Download' and contains the following text: 'Laragon is a universal development environment. It has many features to make you more productive:'. Below this is a section titled 'Benefits of Laragon' which states: 'After downloading, You can add git, phpmyadmin, Node.js/MongoDB, Python/Django/Flask/Postgres, Ruby, Java, Go using "Tools > Quick add"'. A note follows: 'Note: You can also download from GitHub'. The 'Edition' section lists two options: 'Download Laragon - Full (173 MB)' and 'Download Laragon - Portable (38 MB)'. The 'Full' edition includes Apache 2.4, Nginx, MySQL 8, PHP 8, Redis, Memcached, Node.js 18, npm, and git. The 'Portable' edition includes PHP 5.4, MySQL 5.1, and bitnami, with a note that newer versions of PHP/MySQL can be added later using 'Tools > Quick add'.

Download

Laragon is a universal development environment. It has many features to make you more productive:

Benefits of Laragon

After downloading, You can add [git](#), [phpmyadmin](#), [Node.js/MongoDB](#), [Python/Django/Flask/Postgres](#), [Ruby](#), [Java](#), [Go](#) using "[Tools > Quick add](#)"

Note: [You can also download from GitHub](#)

Edition

Download Laragon - Full (173 MB)

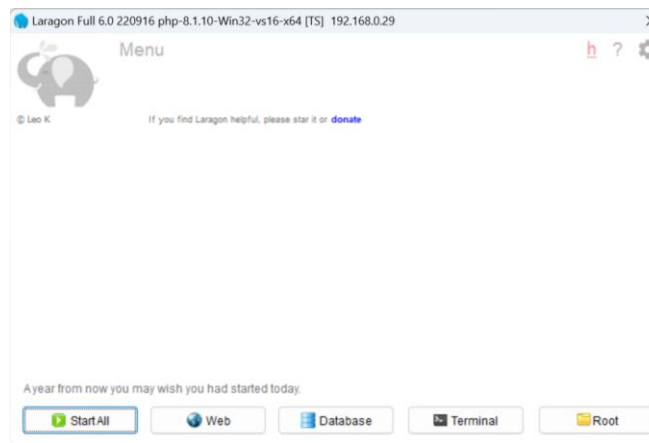
- **Laragon Full (64-bit):** Apache 2.4, Nginx, MySQL 8, PHP 8, Redis, Memcached, Node.js 18, npm, git

Download Laragon - Portable (38 MB)

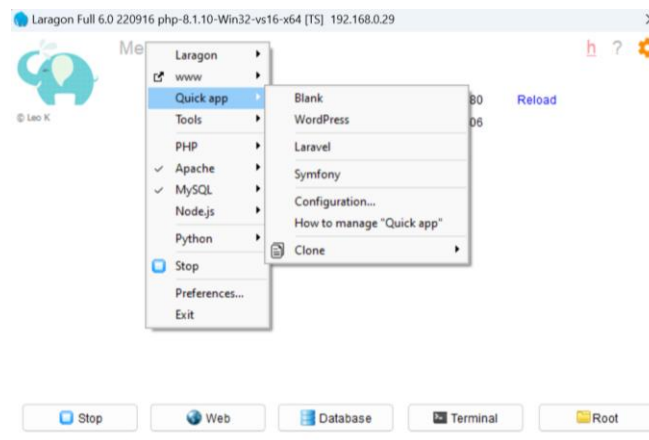
- **Laragon Portable:** PHP 5.4, MySQL 5.1, bitnami - Good for getting started with PHP, then you can add newer versions of PHP/MySQL easily later using "[Tools > Quick add](#)"

Step 2 — Installing Laravel

Now that you have Laragon installed, it's time to install Laravel using Laragon. Launch Laragon.



Go to Menu > Quick App > Laravel and give your project a name



A terminal window pops up and an automatic installation of Laravel and all its dependencies starts.

```
C:\Windows\SYSTEM32\cmd.exe - php...
Creating project: [student]. Please wait...

**** Database:
Created database: [student]

**** Hint: In Terminal, you can type:
-----
cd C:\laragon\www
composer create-project laravel/laravel student --prefer-dist
-----

Running.....
Creating a "laravel/laravel" project at "./student"
Installing laravel/laravel (v10.2.6)
- Installing laravel/laravel (v10.2.6): Extracting archive
Created project in C:\laragon\www\student
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inflector (2.0.8)
- Locking doctrine/lexer (3.0.0)
- Locking dragonmantank/cron-expression (v3.3.3)
- Locking egulias/email-validator (4.0.2)
- Locking fakerphp/faker (v1.23.0)
- Locking filp/whoops (2.15.3)
- Locking fruitcake/php-cors (v1.3.0)
```

```
C:\Windows\SYSTEM32\cmd.exe - C:\laragon\laragon reload

INFO Discovering packages.

laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE

82 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

INFO No publishable resources for tag [laravel-assets].

No security vulnerability advisories found
> @php artisan key:generate --ansi

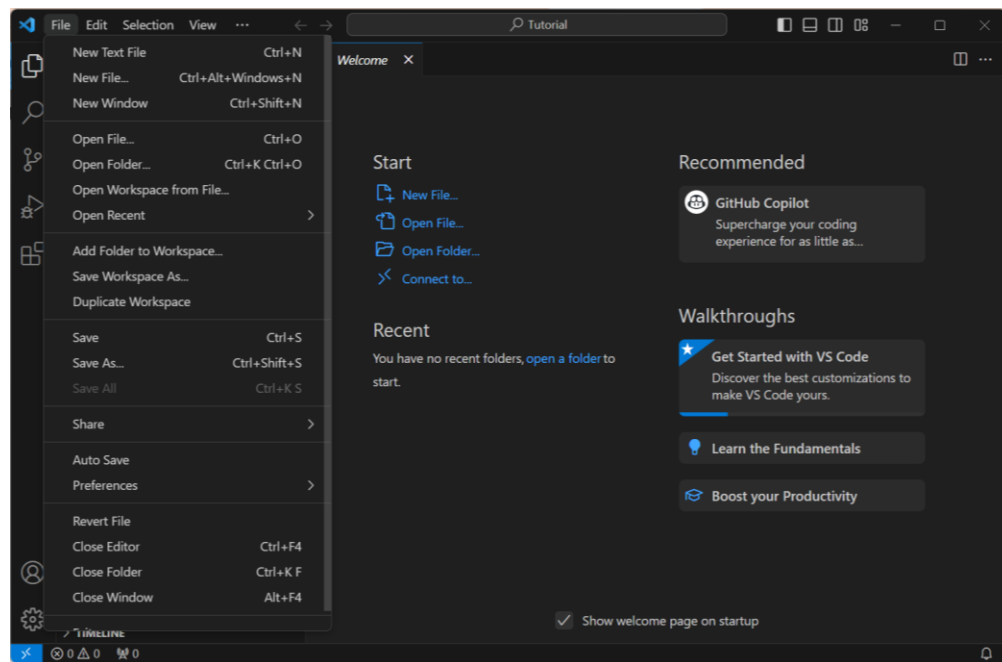
INFO Application key set successfully.

***** NOTE: Now, you can use pretty url for your awesome project :) *****
(Laragon) Project path: C:\laragon\www\student
(Laragon) Pretty url: http://student.test
-----
```

Step 3 — Installing Visual Studio Code and Node JS

You can download Node JS directly from its [download page](#) and Visual Studio Code (VS Code) directly from its [download page](#). For a detailed setup guide, click [here](#). If you have it already installed, you can just ignore this step.

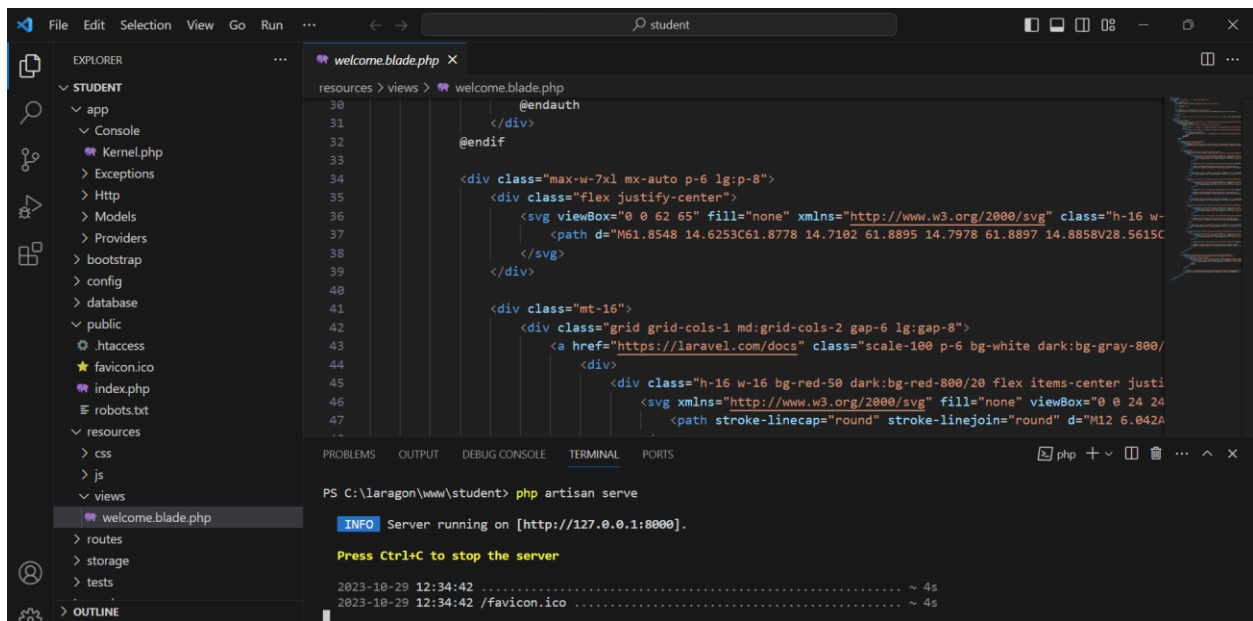
Once the installation is done, launch Visual Studio Code, click “File” on the menu bar and choose “Open Folder”



Navigate to the project path. (Project path: C:/laragon/www/your-project-name), select your project folder and you should have it opened

in Visual Code already. Now, you are set and ready to create magic with Laravel.

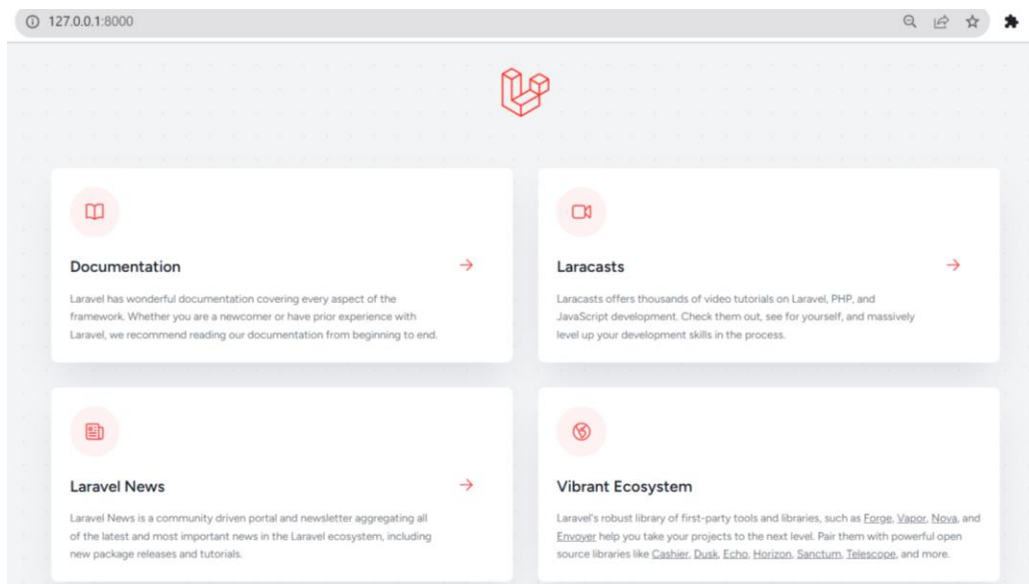
To run your project. Type **php artisan serve**



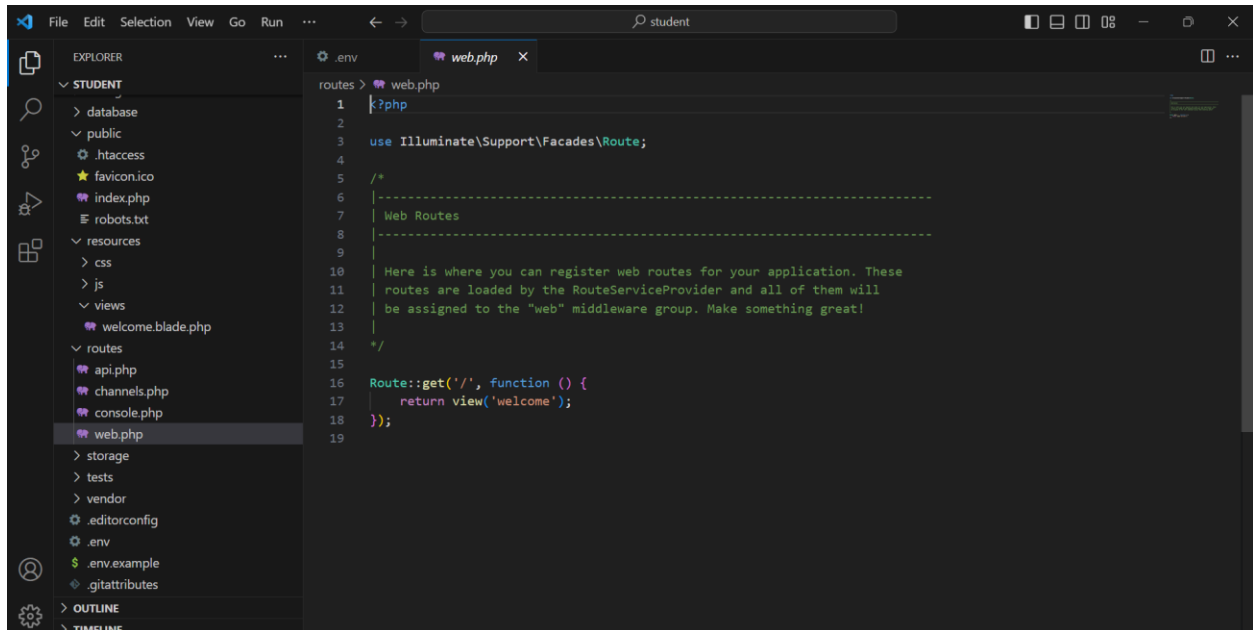
The screenshot shows the Visual Studio Code interface with the `welcome.blade.php` file open. The file contains Blade template syntax for a welcome page. The terminal at the bottom shows the command `php artisan serve` being executed, resulting in the server running on `http://127.0.0.1:8000`. The terminal also displays the command to stop the server (`Press Ctrl+C to stop the server`) and the time taken to serve the `favicon.ico` file.

```
resources > views > welcome.blade.php
30 @endauth
31 </div>
32 @endif
33
34 <div class="max-w-7xl mx-auto p-6 lg:p-8">
35   <div class="flex justify-center">
36     <svg viewBox="0 0 62 65" fill="none" xmlns="http://www.w3.org/2000/svg" class="h-16 w-16">
37       <path d="M61.8548 14.6253C61.8778 14.7102 61.8895 14.7978 61.8897 14.8858V28.5615C61.8897 28.5615 61.8897 28.5615 61.8897 28.5615" fill="none" stroke="black" stroke-width="1">
38     </svg>
39   </div>
40
41   <div class="mt-16">
42     <div class="grid grid-cols-1 md:grid-cols-2 gap-6 lg:gap-8">
43       <a href="https://laravel.com/docs" class="scale-100 p-6 bg-white dark:bg-gray-800/800" >
44         <div class="h-16 w-16 bg-red-50 dark:bg-red-800/20 flex items-center justify-center">
45           <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
46             <path stroke-linecap="round" stroke-linejoin="round" d="M12 6.042A" >
47           </svg>
48         </div>
49       </a>
50     </div>
51   </div>
52 </div>
```

```
PS C:\laragon\www\student> php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
2023-10-29 12:34:42 ..... ~ 4s
2023-10-29 12:34:42 /favicon.ico ..... ~ 4s
```



Every page should use blade template Example: **welcome.blade.php** or **home.blade.php** and save to resources/view folder. You may create a folder to organize everything. Request to access the page using “web.php”

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders like 'database', 'public', 'resources', 'views', and 'routes'. The 'routes' folder is expanded, showing 'web.php' selected. The main editor window displays the content of 'web.php', which includes a PHP docblock, a use statement for 'Illuminate\Support\Facades\Route', and a single GET route for the root path ('/') that returns a 'welcome' view.

```
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6   * Web Routes
7   *
8   * Here is where you can register web routes for your application. These
9   * routes are loaded by the RouteServiceProvider and all of them will
10  * be assigned to the "web" middleware group. Make something great!
11  */
12
13  Route::get('/', function () {
14      return view('welcome');
15  });
```

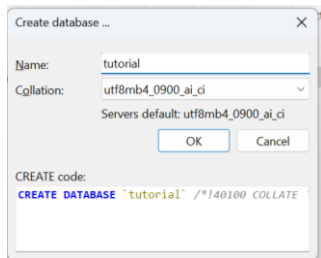
By default, this is the content of your web.php. Explanation: When you run your server using “php artisan serve” it will search for which file should be running.

Setup your Database

Open your project file folder using VS CODE then navigate/search for “.env file”. ENV file is your overall setup for your Laravel project.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tutorial
DB_USERNAME=root
DB_PASSWORD=
```

Go to your **Heidi SQL**



And then type **php artisan migrate**

```
PS C:\laragon\www\student> php artisan migrate

INFO: Preparing database.

Creating migration table ..... 121ms DONE

INFO: Running migrations.

2014_10_12_000000_create_users_table ..... 69ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 100ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 144ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 96ms DONE
```

Let's create new database table simple CRUD function.

Here, we will create a "books" table using laravel migration. so, let's use the following command to create a migration file.

php artisan make:migration create_products_table --create=products

```
PS C:\laragon\www\student> php artisan make:migration create_books_table --create=books

INFO: Migration [C:\laragon\www\student\database\migrations\2023_10_29_150723_create_books_table.php] created successfully.
```

After this command you will find one file in the following path "database/migrations" and you have to put below code in your migration file for creating the book table.

```

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('books', function (Blueprint $table) {
            $table->id();
            $table->string('book_name');
            $table->string('book_author');
            $table->string('year_published');
            $table->timestamps();
        });
    }
}

```

Now you have to run this migration by the following command: **php artisan migrate**

```

PS C:\laragon\www\student> php artisan migrate

INFO Running migrations.

2023_10_29_150723_create_books_table ..... 139ms DONE

```

Step: Create Controller and Model

In this step, now we should create new resource controller as BookController. So run below command and create new controller. Below controller for create resource controller.

```

PS C:\laragon\www\student> php artisan make:controller BookController --resource --model=Book

A App\Models\Book model does not exist. Do you want to generate it? (yes/no) [yes]
> yes

INFO Model [C:\laragon\www\student\app\Models\Book.php] created successfully.

INFO Controller [C:\laragon\www\student\app\Http\Controllers\BookController.php] created successfully.

```


Add this to your app/Models/Book.php

```
<?php

namespace App\Models;

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

class Book extends Model
{
    use HasFactory;
    protected $fillable = ['book_name', 'book_author', 'year_published'];
}
```

After the bellow command, you will find a new file in this path
"app/Http/Controllers/BookController.php".

In this controller will create seven methods by default as bellow methods:

So, let's write bellow code and put on BookController.php file.

1. Index()

```
<?php

namespace App\Http\Controllers;

use App\Models\Book;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\View\View;

class BookController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $books = Books::latest()->paginate(5);

        return view('books.index', compact('books'))->with('i', (request()->input('page', 1) - 1) * 5);
    }
}
```

2. Create()

```
/**
 * Show the form for creating a new resource.
 */
public function create(): View
{
    return view('books.create');
}
```

3. Store()

```
public function store(Request $request): RedirectResponse
{
    $request->validate([
        'book_name' => 'required',
        'book_author' => 'required',
        'year_published' => 'required'
    ]);
    $request->except(['_token', '_method']);

    Book::create($request->all());

    return redirect()->route('books.index')
        ->with('success','Book created successfully.');
```

4. Show()

```
public function show(Book $book): View
{
    return view('books.show',compact('book'));
}
```

5. Edit()

```
public function edit(Book $book): View
{
    return view('books.edit')->with('book', $book);
}
```

6. Update()

```
public function update(Request $request, Book $book): RedirectResponse
{
    $request->validate([
        'book_name' => 'required',
        'book_author' => 'required',
        'year_published' => 'required'
    ]);

    $book->update($request->all());

    return redirect()->route('books.index')
        ->with('success', 'Book updated successfully');
}
```

7. Destroy()

```
public function destroy(Book $book): RedirectResponse
{
    $book->delete();

    return redirect()->route('books.index')
        ->with('success', 'Book deleted successfully');
}
```

Add Resource route

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\BookController;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

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

Route::resource('books', BookController::class);
```

Add Blade Files

In last step. In this step we have to create just blade files. So mainly we have to create layout file and then create new folder "books" then create blade files of crud app. So finally, you have to create following bellow blade file:

1) layout.blade.php

2) index.blade.php

3) create.blade.php

4) edit.blade.php

5) show.blade.php

So, let's just create following file and write bellow code.

resources/views/books/layout.blade.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Laravel 10 CRUD Application</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
      rel="stylesheet">
  </head>
  <body>
    <div class="container">
      @yield('content')
    </div>
  </body>
</html>
```

resources/views/books/index.blade.php

```
@extends('books.layout')

@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
            <div class="pull-left">
                <h2>Laravel 10 CRUD</h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-success" href="{{ route('books.create') }}"> Create New Books</a>
            </div>
        </div>
    </div>

    @if ($message = Session::get('success'))
        <div class="alert alert-success">
            <p>{{ $message }}</p>
        </div>
    @endif

    <br>
```

```
<table class="table table-bordered">
    <tr>
        <th>No</th>
        <th>Book Name</th>
        <th>Book Author</th>
        <th>Year Published</th>
        <th width="280px">Action</th>
    </tr>
    @foreach ($books as $book)
        <tr>
            <td>{{ $book->id }}</td>
            <td>{{ $book->book_name }}</td>
            <td>{{ $book->book_author }}</td>
            <td>{{ $book->year_published}}</td>
            <td>
                <form action="{{ route('books.destroy',$book->id) }}" method="POST">
                    <a class="btn btn-info" href="{{ route('books.show',$book->id) }}">Show</a>
                    <a class="btn btn-primary" href="{{ route('books.edit',$book->id) }}">Edit</a>
                    @csrf
                    @method('DELETE')
                    <button type="submit" class="btn btn-danger">Delete</button>
                </form>
            </td>
        </tr>
    @endforeach
</table>
{!! $books->links() !!}
@endsection
```

resources/views/books/create.blade.php

```
@extends('books.layout')

@section('content')
<div class="row">
    <div class="col-lg-12 margin-tb">
        <div class="pull-left">
            <h2>Add New Book</h2>
        </div>
        <div class="pull-right">
            <a class="btn btn-primary" href="{{ route('books.index') }}"> Back</a>
        </div>
    </div>
</div>

@if ($errors->any())
    <div class="alert alert-danger">
        <strong>Whoops!</strong> There were some problems with your input.<br><br>
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
@endside
```

```
<form action="{{ route('books.store') }}" method="POST">
    @csrf
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Name:</strong>
                <input type="text" name="book_name" class="form-control" placeholder="Book Name">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Author:</strong>
                <input type="text" name="book_author" class="form-control" placeholder="Book Author">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Year Published:</strong>
                <input type="text" name="year_published" class="form-control" placeholder="Year Published">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12 text-center">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
    </div>
</form>
@endside
```

resources/views/books/edit.blade.php

```
@extends('books.layout')

@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
            <div class="pull-left">
                <h2>Edit Book</h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-primary" href="{{ route('books.index') }}"> Back</a>
            </div>
        </div>
    </div>

    @if ($errors->any())
        <div class="alert alert-danger">
            <strong>Whoops!</strong> There were some problems with your input.<br><br>
            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif
@endsection
```

```
<form action="{{ route('books.update',$book->id) }}" method="POST">
    @csrf
    @method('PUT')
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Name:</strong>
                <input type="text" name="book_name" value="{{ $book->book_name }}" class="form-control"
                placeholder="Book Name">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Author:</strong>
                <input type="text" name="book_author" value="{{ $book->book_author }}" class="form-control">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Year Published:</strong>
                <input type="text" name="year_published" value="{{ $book->year_published }}" class="form-control">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12 text-center">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
    </div>
</form>

@endsection
```

resources/views/books/show.blade.php

```
@extends('books.layout')
@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
            <div class="pull-left">
                <h2> Show Book</h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-primary" href="{{ route('books.index') }}"> Back</a>
            </div>
        </div>
    </div>
</div>
```

```
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Name:</strong>
                {{ $book->book_name }}
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Book Author:</strong>
                {{ $book->book_author }}
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Year Published</strong>
                {{ $book->year_published }}
            </div>
        </div>
    </div>
</div>
@endsection
```


After that run php artisan serve

List Book

Laravel 10 CRUD

Create New Books

No	Book Name	Book Author	Year Published	Action
3	The History of Tom Jones, a Foundling	Henry Fielding	1749	Show Edit Delete

Add New Book

Add New Book

[Back](#)

Book Name:

Pride and Prejudice

Book Author:

Jane Austen

Year Published:

1813

Submit

Edit Book

Edit Book

[Back](#)

Book Name:

Pride and Prejudice

Book Author:

Jane Austens

Year Published:

1813

Submit

Show Book

Show Book

[Back](#)

Book Name: Pride and Prejudice

Book Author: Jane Austens

Year Published 1813

Delete Book

Laravel 10 CRUD

Create New Books

Book deleted successfully

No	Book Name	Book Author	Year Published	Action
5	Pride and Prejudice	Jane Austens	1813	<a>Show <a>Edit <a>Delete