# Learning Outcome

To meet the learning outcome, a student has to complete the following activities

# Create a Form which inserts the data in the database (7 hrs)

# Demonstrate all the CRUD operation using (DB) from controllers (7 hrs)

# Create a Model and Demonstrate all CRUD operations (8 hrs)

# Creating a complete end-to–end solution which demonstrate session and database connectivity (8 hrs)

# Activity 1

## Aim: Create a Form which inserts the data in the database

**Learning outcome:** Create a Form which inserts the data in the database

**Duration:** 7 hrs

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

**Code/Program/Procedure (with comments):**

// First, creating table with SQL query:

CREATE TABLE student\_details

(

id int NOT NULL AUTO\_INCREMENT,

first\_name varchar(50),

last\_name varchar(50),

city\_name varchar(50),

email varchar(50),

PRIMARY KEY (id)

);

// Now, create three files for insert data in Laravel.

Step 1. Create a controller name as StudInsertController.php.

The file location is : (app/Http/Controllers/StudInsertController.php)

Step 2. Create view page name as stud\_create.php

The file location is: (resources/views/stud\_create.php)

// Then put this code in your StudInsertController.php.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use DB;

use App\Http\Requests;

use App\Http\Controllers\Controller;

class StudInsertController extends Controller

{

//

public function insertform(){

return view('stud\_create');

}

public function insert(Request $request){

$first\_name = $request->input('first\_name');

$last\_name = $request->input('last\_name');

$city\_name = $request->input('city\_name');

$email = $request->input('email');

$data=array('first\_name'=>$first\_name,"last\_name"=>$last\_name,"city\_name"=>$city\_name,"email"=>$email);

DB::table('student\_details')->insert($data);

echo "Record inserted successfully.<br/>";

echo '<a href = "/insert">Click Here</a> to go back.';

}

}

// Then put this code in your stud\_create.blade.php.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Student Management</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">

</script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js">

</script>

</head>

<body>

<div class="container">

<h2 class="text-center">Student Management | Add</h2><br>

<form action = "/create" method = "post" class="form-group" style="width:70%; margin-left:15%;" action="/action\_page.php">

<input type = "hidden" name = "\_token" value = "<?php echo csrf\_token(); ?>">

<input type = "hidden" name = "\_token" value = "<?php echo csrf\_token(); ?>">

<label class="form-group">First Name:</label>

<input type="text" class="form-control" placeholder="First Name" name="first\_name">

<label>Last Name:</label>

<input type="text" class="form-control" placeholder="Last Name" name="last\_name">

<label>City Name:</label>

<select class="form-control" name="city\_name">

<option value="bhubaneswar">Bhubaneswar</option>

<option value="cuttack">Cuttack</option>

</select>

<label>Email:</label>

<input type="text" class="form-control" placeholder="Enter Email" name="email"><br>

<button type="submit" value = "Add student" class="btn btn-primary">Submit</button>

</form>

</div>

</body>

</html>

Step 3. Then go to routes as web.php and put this code.

The file location is : (routes/web.php)

Route::get('insert','StudInsertController@insertform');

Route::post('create','StudInsertController@insert');

// Then, you can see the data is inserted.

// So, the data is inserted in the database, then we need to retrieve a record or data from the MySQL database.

Step 4. Create a controller for view name as StudViewController.php.

The file location is : (app/Http/Controllers/StudViewController.php)

Step 5. Create a view page name as stud\_view.blade.php.

The file location is : (resources/views/stud\_view.blade.php).

// Then put this code StudViewController.php.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use DB;

use App\Http\Controllers\Controller;

class StudViewController extends Controller

{

//

public function index(){

$users = DB::select('select \* from student\_details');

return view('stud\_view',['users'=>$users]);

}

}

// Then, retrieve the students data then put this code stud\_view.blade.php.

<!DOCTYPE html>

<html lang="en">

<head>

<title>View Student Records</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">

</script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js">

</script>

</head>

<body>

<div class="container">

<h2 class="text-center">View Student Records</h2>

<table class="table table-bordered table-striped">

<thead>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>City Name</th>

<th>Email</th>

</tr>

</thead>

<tbody>

@foreach ($users as $user)

<tr>

<td>{{ $user->id }}</td>

<td>{{ $user->first\_name }}</td>

<td>{{ $user->last\_name }}</td>

<td>{{ $user->city\_name }}</td>

<td>{{ $user->email }}</td>

</tr>

@endforeach

</tbody>

</table>

</div>

</body>

</html>

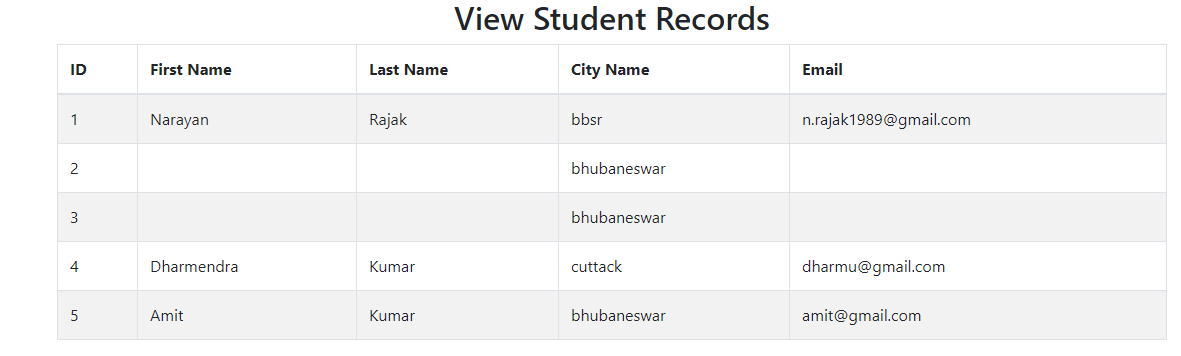
Step 6. Then go to routes as web.php and put this code.

The file location is (routes/web.php)

Route::get('view-records','StudViewController@index');

// Then, you can see the view page.

**Output/Results snippet:**



**References:**

* <https://www.devopsschool.com/blog/how-to-insert-and-retrieve-data-in-database-laravel-framework/>

# 

# Activity 2

## Aim: Demonstrate all the CRUD operation using (DB) from controllers

**Learning outcome:** Demonstrate all the CRUD operation using (DB) from controllers

**Duration:** 7 hrs.

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

**Code/Program/Procedure (with comments):**

**Step 1 -** Setup Laravel project

**// U**sing the Composer create-project command.

composer create-project --prefer-dist laravel/laravel laravel\_crud

**Step 2 - C**reate a database named laravel\_crud and configure it.

**//** Open the .env file in your root path and setup the username and password.

LOG\_CHANNEL=stack

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=laravel\_crud

DB\_USERNAME=root

DB\_PASSWORD=

**// C**reate a table named articles. You can so it creating a migration:

php artisan make:migration create\_articles\_table --create=articles

# Check the file created in database/migrations/ and make this edition:

# public function up()

# {

# Schema::create('articles', function (Blueprint $table) {

# $table->increments('id');

# $table->string('topic');

# $table->text('description');

# $table->string('categorie');

# $table->timestamps();

# });

# }

# // Migrate it using

# php artisan migrate

# /\*Sometimes it can give an error about maxLength, collection, or the charset. Therefore, if it is your case open AppServiceProvider.php in app/Providers/ and make this file looks like this:\*/

# <?php

# namespace App\Providers;

# use Illuminate\Support\ServiceProvider;

# use Illuminate\Support\Facades\Schema;

# class AppServiceProvider extends ServiceProvider

# {

# /\*\*

# \* Register any application services.

# \*

# \* @return void

# \*/

# public function register()

# {

# //

# }

# /\*\*

# \* Bootstrap any application services.

# \*

# \* @return void

# \*/

# public function boot()

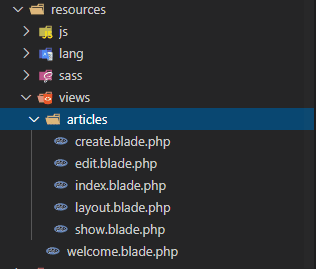
# {

# Schema::defaultStringLength(191);

# }

# }

# Step 3 - Create the views.



**//** Setup a Resource Router. Open routes/web.php and add this line:

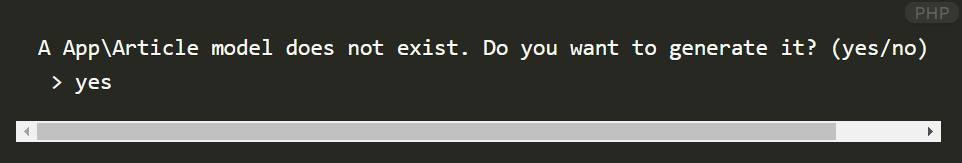
Route::resource('articles', 'ArticleController');

# Step 4 - Create Controller and Model

# // We need to create a new controller as ArcticleController as well as its Model.

# php artisan make:controller ArticleController--resource --model=Article

# // Enter yes to the question that will appear.

****

**//** Open your model Article.php and create these variables:

# <?php

# namespace App;

# use Illuminate\Database\Eloquent\Model;

# class Article extends Model

{

protected $fillable = [

# 'topic','description', 'categorie',

# ];

# }

# // Open ArticleController.php:

# // index() lists table data and make a pagination with three elements on each page.

# public function index()

# {

# $articles = DB::table('articles')->paginate(3);

# return view('articles.index', ['articles' => $articles]);

# }

# // create() – Returns the View to create a table item.

# public function create()

# {

# return view('articles.create');

# }

# // store() – Saves the new item to the table

# public function store(Request $request)

# {

# $request->validate([

# 'topic' => 'required',

# 'description' => 'required',

# 'categorie' => 'required',

# ]);

# Article::create($request->all());

# return redirect()->route('articles.index')

# ->with('success','Article created successfully.');

# }

# // show() – Shows a specific item.

# public function show(Article $article)

# {

# return view('articles.show',compact('article'));

# }

# // edit() – Returns the View for editing the data

# public function edit(Article $article)

# {

# return view('articles.edit',compact('article'));

# }

# // update() – Save data update.

# public function update(Request $request, Article $article)

# {

# $request->validate([

# 'topic' => 'required',

# 'description' => 'required',

# 'categorie' => 'required',

# ]);

# $article->update($request->all());

# return redirect()->route('articles.index')

# ->with('success','Article updated successfully');

# }

# // delete() – Removes data.

# public function destroy(Article $article)

# {

# $article->delete();

# return redirect()->route('articles.index')

# ->with('success','Article deleted successfully');

# }

# Execute:

# php artisan serve

# And go to: <http://127.0.0.1:8000/articles>

# // Setup the Layout

# // Open layout.blade.php

# <!DOCTYPE html>

# <html lang="en">

# <head>

# <meta charset="UTF-8">

# <meta name="viewport" content="width=device-width, initial-scale=1.0">

# <title>Layout</title>

# <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

# </head>

# <body>

# <div class="container">

# <!-- Content here -->

# @yield('content')

# </div>

# </body>

# </html>

# // READ Operation

# // Open index.blade.php

# @extends('articles.layout')

# @section('content')

# ...

# ...

# @endsection

# Inside the section content add:

# Button to add a new article

# <div class="row">

# <div class="col-lg-12 margin-tb">

# <div class="pull-right">

# <a class="btn btn-primary" href="{{ route('articles.create') }}"> Create new article</a>

# </div>

# </div>

# </div>

# Section to display a success message

# @if ($message = Session::get('success'))

# <div class="alert alert-success">

# <p>{{ $message }}</p>

# </div>

# @endif

# And a table to display the articles:

# <table class="table table-striped table-bordered">

# <thead>

# <tr>

# <th scope="col">#</th>

# <th scope="col">Topic</th>

# <th scope="col">Description</th>

# <th scope="col">Categorie</th>

# <th scope="col"></th>

# </tr>

# </thead>

# <tbody>

# @foreach ($articles as $article)

# <tr>

# <th scope="row">{{ $article->id }}</th>

# <td>{{ $article->topic }}</td>

# <td>{{ $article->description }}</td>

# <td>{{ $article->categorie }}</td>

# <td>

# <form action="{{ route('articles.destroy',$article->id) }}" method="POST">

# <a class="btn btn-info" href="{{ route('articles.show',$article->id) }}">Show</a>

# <a class="btn btn-primary" href="{{ route('articles.edit',$article->id) }}">Edit</a>

# @csrf

# @method('DELETE')

# <button type="submit" class="btn btn-danger">Delete</button>

# </form>

# </td>

# </tr>

# @endforeach

# </tbody>

# </table>

# Finally, the links for the pagination:

# {{ $articles->links() }}

# // CREATE Operation

# // Bellow, we have a button to return to the home/initial page, a section to display an error message and the form.

# @extends('articles.layout')

# @section('content')

# <div class="row">

# <div class="col-lg-12 margin-tb">

# <div class="pull-left">

# <h2>Create Ne w Article</h2>

# </div>

# <div class="pull-right">

# <a class="btn btn-info" href="{{ route('articles.index') }}"> Back</a>

# </div>

# </div>

# </div>

# @if ($errors->any())

# <div class="alert alert-danger">

# <strong>Warning!</strong> Please check your fields<br><br>

# <ul>

# @foreach ($errors->all() as $error)

# <li>{{ $error }}</li>

# @endforeach

# </ul>

# </div>

# @endif

# <form action="{{ route('articles.store') }}" method="POST">

# @csrf

# <div class="form-group">

# <label for="topic">Topic title</label>

# <input type="text" class="form-control" placeholder="Enter Topic" name ="topic">

# </div>

# <div class="form-group">

# <label for="description">Description</label>

# <textarea class="form-control" rows="3"placeholer ="Enter description" name="description"></textarea>

# </div>

# <div class="form-group">

# <label for="categorie">Categorie</label>

# <input type="text" class="form-control” placeholder="Enter Categorie" name ="categorie">

# </div>

# <button type="submit" class="btn btn-primary">Submit</button>

# </form>

# @endsection

# // UPDATE Operation

# // A form to update the article:

# @extends('articles.layout')

# @section('content')

# <div class="row">

# <div class="col-lg-12 margin-tb">

# <div class="pull-left">

# <h2>Edit Article</h2>

# </div>

# <div class="pull-right">

# <a class="btn btn-info" href="{{ route('articles.index') }}"> Back</a>

# </div>

# </div>

# </div>

# @if ($errors->any())

# <div class="alert alert-danger">

# <strong>Warning!</strong> Please check your fields.<br><br>

# <ul>

# @foreach ($errors->all() as $error)

# <li>{{ $error }}</li>

# @endforeach

# </ul>

# </div>

# @endif

# <form action="{{ route('articles.update',$article->id) }}" method="POST">

# @csrf

# @method('PUT')

# <div class="form-group">

# <label for="topic">Topic title</label>

# <input type="text" class="form-control" value="{{ $article->topic }}" placeholder="Enter Topic" name ="topic">

# </div>

# <div class="form-group">

# <label for="description">Description</label>

# <textarea class="form-control" rows="3" name="description">{{ $article->description }}</textarea>

# </div>

# <div class="form-group">

# <label for="categorie">Categorie</label>

# <input type="text" class="form-control" value="{{ $article->categorie }}" placeholder="Enter Categorie" name ="categorie">

# </div>

# <button type="submit" class="btn btn-primary">Submit</button>

# </form>

# @endsection

# // DELETE Operaton

# // On index.blade.php you saw that we have these lines:

# @method('DELETE')

# <button type="submit" class="btn btn-danger">Delete</button>

**References:**

* <https://codesource.io/build-a-complete-laravel-crud-application-with-mysql/#Create-Controller-and-Model>

# Activity 3

**Aim:** Create a Model and Demonstrate all CRUD operations

**Learning outcome:** Create a Model and Demonstrate all CRUD operations

**Duration:** 8 hrs.

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS

2. Php , MySQL , XAMMP or LAMP Server, composer, Laravel 8

**Code/Program/Procedure (with comments):**

Step 1 — Installing Laravel 8 using composer

// Open a new command-line interface and run the following command:

$ composer create-project laravel/laravel=8.0 laravel8app --prefer-dist

// Wait for composer to install the dependencies and set you your project

Step 2 — Setting up a MySQL Database

// In your terminal, run the following command to run the mysql client:

$ mysql -u root -p

// When prompted, enter the password for your MySQL server when you've installed it.

// Next, run the following SQL statement to create a db database:

mysql> create database db;

// Open the .env file and update the credentials to access your MySQL database:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=db

DB\_USERNAME=root

DB\_PASSWORD=\*\*\*\*\*\*

// You need to provide the database name, the username and password.

// run the migrate command to create your database and a bunch of SQL tables needed by Laravel:

$ php artisan migrate

Note: You can run the migrate command at any other points of your development to add other SQL tables in your database or to later your database if you need to add any changes later.

Step 3 — Creating a Database Migration

// Open your terminal and run the following commands:

$ cd laravel8app

$ php artisan make:migration create\_products\_table --create=products

A migration file will be created inside the database/migrations folder of your product, next we need to add the fields to our database table. A product will have a name, description, price, date created, and date updated.

Open the create\_products\_table.php file that contains the migration class and update it as follows:

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('products', function (Blueprint $table) {

$table->id();

$table->string('name', 255)->nullable();

$table->string('description', 500)->nullable();

$table->decimal('price', 22)->nullable()->default(0.00);

$table->timestamp('created\_at')->useCurrent();

$table->timestamp('updated\_at')->nullable();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('products');

}

}

To avoid running into errors, you need to specify the default string length before running your migration.

Open the app/Providers/AppServiceProvider.php file and add Schema::defaultstringLength(191) as follows:

<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

use Illuminate\Support\Facades\Schema;

class AppServiceProvider extends ServiceProvider

{

/\*\*

\* Register any application services.

\*

\* @return void

\*/

public function register()

{

//

}

/\*\*

\* Bootstrap any application services.

\*

\* @return void

\*/

public function boot()

{

Schema::defaultStringLength(191);

}

}

Open your terminal and run the following command:

$ php artisan migrate

This will add the fields to our database table.

Step 4 — Adding a Resource Route

// Add routes for our CRUD operations.

Open the routes\web.php file and add our resource route as follows:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ProductController;

Route::get('/', function () {

return view('welcome');

});

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

Step 5 — Adding a Laravel 8 Controller and Model

Next, we need to create a Laravel controller and model by running the following command:

$ php artisan make:controller ProductController --resource --model=Product

You'll be prompted if you want to create the Product model because it does not exist. Type yes and it will create the model and controller.

Open the app/Http/Controllers/ProductController.php file and update it as follows:

<?php

namespace App\Http\Controllers;

use App\Models\Product;

use Illuminate\Http\Request;

class ProductController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

$products = Product::latest()->paginate(5);

return view('products.index', compact('products'))

->with('i', (request()->input('page', 1) - 1) \* 5);

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

return view('products.create');

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$request->validate([

'name' => 'required',

'description' => 'required',

'price' => 'required'

]);

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

return redirect()->route('products.index')

->with('success', 'Product created successfully.');

}

/\*\*

\* Display the specified resource.

\*

\* @param \App\Models\Product $product

\* @return \Illuminate\Http\Response

\*/

public function show(Product $product)

{

return view('products.show', compact('product'));

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param \App\Models\Product $product

\* @return \Illuminate\Http\Response

\*/

public function edit(Product $product)

{

return view('products.edit', compact('product'));

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param \App\Models\product $product

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, Product $product)

{

$request->validate([

'name' => 'required',

'description' => 'required',

'price' => 'required'

]);

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

return redirect()->route('products.index')

->with('success', 'Product updated successfully');

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param \App\Models\Product $product

\* @return \Illuminate\Http\Response

\*/

public function destroy(Product $product)

{

$product->delete();

return redirect()->route('products.index')

->with('success', 'Product deleted successfully');

}

}

Laravel 8 makes use of a Models folder for storing model files.

Open the app/Models/Product.php, add the following functions and the fillable, the fillable are the fields in the database that a user can fill:

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

use HasFactory;

protected $table = 'products';

public $timestamps = true;

protected $casts = [

'price' => 'float'

];

protected $fillable = [

'name',

'description',

'price',

'created\_at'

];

}

Step 6 — Adding your Laravel 8 Blade Views

Laravel makes use of the blade templating system for views.

Inside the resources/views folder, create two Layouts and Products folders.

Update the Layouts/App.blade.php file with the following content:

<html>

<head>

<title>App Name - @yield('title')</title>

<!-- Bootstrap -->

<link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0 alpha/css/bootstrap.css" rel="stylesheet">

<!-- Font Awesome JS -->

<script defer src="https://use.fontawesome.com/releases/v5.0.13/js/solid.js"

integrity="sha384-tzzSw1/Vo+0N5UhStP3bvwWPq+uvzCMfrN1fEFe+xBmv1C/AtVX5K0uZtmcHitFZ" crossorigin="anonymous">

</script>

<script defer src="https://use.fontawesome.com/releases/v5.0.13/js/fontawesome.js"integrity="sha384-6OIrr52G08NpOFSZdxxz1xdNSndlD4vdcf/q2myIUVO0VsqaGHJsB0RaBE01VTOY" crossorigin="anonymous">

</script>

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #9C27B0;

color: white;

text-align: center;

}

</style>

</head>

<body>

@section('sidebar')

@show

<div class="container">

@yield('content')

</div>

</body>

</html>

// Open the Index.blade.php and update it as follows:

@extends('layouts.app')

@section('content')

<div class="row">

<div class="col-lg-12 margin-tb">

<div class="pull-left">

<h2>Laravel 8 CRUD Example </h2>

</div>

<div class="pull-right">

<a class="btn btn-success" href="" title="Create a product"> <i class="fas fa-plus-circle"></i>

</a>

</div>

</div>

</div>

@if ($message = Session::get('success'))

<div class="alert alert-success">

<p></p>

</div>

@endif

<table class="table table-bordered table-responsive-lg">

<tr>

<th>No</th>

<th>Name</th>

<th>description</th>

<th>Price</th>

<th>Date Created</th>

<th>Actions</th>

</tr>

@foreach ($products as $product)

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td>

<form action="" method="POST">

<a href="" title="show">

<i class="fas fa-eye text-success fa-lg"></i>

</a>

<a href="">

<i class="fas fa-edit fa-lg"></i>

</a>

@csrf

@method('DELETE')

<button type="submit" title="delete" style="border: none; background-color:transparent;">

<i class="fas fa-trash fa-lg text-danger"></i>

</button>

</form>

</td>

</tr>

@endforeach

</table>

{!! $products->links() !!}

@endsection

// Open the create.blade.php file and update it as follows:

@extends('layouts.app')

@section('content')

<div class="row">

<div class="col-lg-12 margin-tb">

<div class="pull-left">

<h2>Add New Product</h2>

</div>

<div class="pull-right">

<a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward "></i> </a>

</div>

</div>

</div>

@if ($errors->any())

<div class="alert alert-danger">

<strong>Error!</strong>

<ul>

@foreach ($errors->all() as $error)

<li></li>

@endforeach

</ul>

</div>

@endif

<form action="" method="POST" >

@csrf

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Name:</strong>

<input type="text" name="name" class="form-control" placeholder="Name">

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Description:</strong>

<textarea class="form-control" style="height:50px" name="introduction"

placeholder="description"></textarea>

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Price:</strong>

<input type="number" name="price" class="form-control" placeholder="Put the price">

</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

// Open the edit.blade.php file and update it as follows:

@extends('layouts.app')

@section('content')

<div class="row">

<div class="col-lg-12 margin-tb">

<div class="pull-left">

<h2>Edit Product</h2>

</div>

<div class="pull-right">

<a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward "></i> </a>

</div>

</div>

</div>

@if ($errors->any())

<div class="alert alert-danger">

<strong>Error!</strong>

<ul>

@foreach ($errors->all() as $error)

<li></li>

@endforeach

</ul>

</div>

@endif

<form action="" method="POST">

@csrf

@method('PUT')

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Name:</strong>

<input type="text" name="name" value="" class="form-control" placeholder="Name">

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Description</strong>

<textarea class="form-control" style="height:50px" name="description"

placeholder="description"></textarea>

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Price</strong>

<input type="number" name="price" class="form-control" placeholder=""

value="">

</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

// Open the show.blade.php file and update it as follows:

@extends('layouts.app')

@section('content')

<div class="row">

<div class="col-lg-12 margin-tb">

<div class="pull-left">

<h2> </h2>

</div>

<div class="pull-right">

<a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward "></i> </a>

</div>

</div>

</div>

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Name:</strong>

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Description</strong>

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Price</strong>

</div>

</div>

<div class="col-xs-12 col-sm-12 col-md-12">

<div class="form-group">

<strong>Date Created</strong>

</div>

</div>

</div>

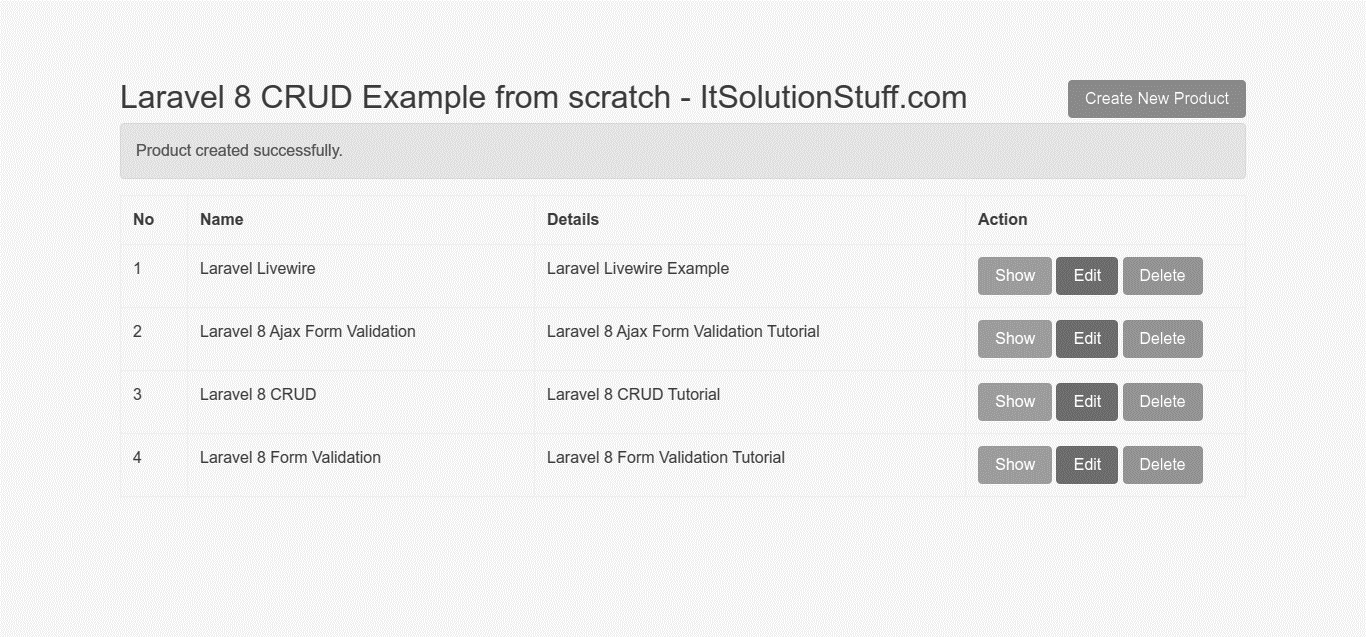
@endsection

You can serve your Laravel 8 application using the following command:

$ php artisan serve

You can access your app from <http://127.0.0.1:8000>.

**Output:**



**References:**

* <https://www.itsolutionstuff.com/post/laravel-8-crud-application-tutorial-for-beginnersexample.html>

**Activity 4**

## Aim: Creating a complete end-to–end solution which demonstrate session and database connectivity

**Learning outcome:** Creating a complete end-to–end solution which demonstrate session and database connectivity

**Duration:** 8 hrs.

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Create Laravel App

composer create-project --prefer-dist laravel/laravel laravel\_demo\_app

Next, head over to app folder:

cd laravel\_demo\_app

Connect to Database

// To add database name, username, and password into the .env configuration file to connect the laravel app to the database:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=database\_name

DB\_USERNAME=database\_user\_name

DB\_PASSWORD=database\_password

The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

php artisan migrate

Set Up Auth Controller

php artisan make:controller CustomAuthController

Thereafter, open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Hash;

use Session;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class CustomAuthController extends Controller

{

public function index()

{

return view('auth.login');

}

public function customLogin(Request $request)

{

$request->validate([

'email' => 'required',

'password' => 'required',

]);

$credentials = $request->only('email', 'password');

if (Auth::attempt($credentials)) {

return redirect()->intended('dashboard')

->withSuccess('Signed in');

}

return redirect("login")->withSuccess('Login details are not valid');

}

public function registration()

{

return view('auth.registration');

}

public function customRegistration(Request $request)

{

$request->validate([

'name' => 'required',

'email' => 'required|email|unique:users',

'password' => 'required|min:6',

]);

$data = $request->all();

$check = $this->create($data);

return redirect("dashboard")->withSuccess('You have signed-in');

}

public function create(array $data)

{

return User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::make($data['password'])

]);

}

public function dashboard()

{

if(Auth::check()){

return view('dashboard');

}

return redirect("login")->withSuccess('You are not allowed to access');

}

public function signOut() {

Session::flush();

Auth::logout();

return Redirect('login');

}

}

Create Auth Routes

This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\CustomAuthController;

Route::get('dashboard', [CustomAuthController::class, 'dashboard']);

Route::get('login', [CustomAuthController::class, 'index'])->name('login');

Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');

Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');

Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');

Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');

Create Auth Blade View Files

You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

@extends('app')

@section('content')

<main class="login-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Login</h3>

<div class="card-body">

<form method="POST" action="{{ route('login.custom') }}">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email" class="form-control" name="email" required

autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control" name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label>

<input type="checkbox" name="remember"> Remember Me

</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Signin</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

@extends('app')

@section('content')

<main class="signup-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Register User</h3>

<div class="card-body">

<form action="{{ route('register.custom') }}" method="POST">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Name" id="name" class="form-control" name="name"

required autofocus>

@if ($errors->has('name'))

<span class="text-danger">{{ $errors->first('name') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email\_address" class="form-control"

name="email" required autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control"

name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label><input type="checkbox" name="remember"> Remember Me</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Sign up</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

Home » Laravel Tutorials » Laravel 8 Custom Auth Login and Registration Tutorial

Laravel 8 Custom Auth Login and Registration Tutorial

Last updated on: January 31, 2022 by Digamber

Laravel 8 custom login and registration tutorial; In this tutorial, we will explain how to create custom authentication login and registration in the Laravel application.

Laravel is a top-notch PHP framework that is a boon for web developers, and it offers numerous packages and plugins to build any type of functionality.

When we talk about the authentication feature, you can install and use the Laravel JetStream package.

Having said that, we are going to share with you the traditional method through which you can create custom authentication in laravel.

This quick guide, bit by bit, describes the simple method to build custom login, registration and dashboard pages or templates.

Laravel 8 Custom Auth Login and Registration Example

Step 1: Create Laravel App

Step 2: Connect to Database

Step 3: Set Up Auth Controller

Step 4: Create Auth Routes

Step 5: Create Auth Blade View Files

Step 6: Run Laravel Development Server

Create Laravel App

We assume you have already configured Composer on your system, run the following command to install the new laravel app. However, you can skip this step if the app is already installed.

composer create-project --prefer-dist laravel/laravel laravel\_demo\_app

Bash

Next, head over to app folder:

cd laravel\_demo\_app

Bash

Connect to Database

Now, you have to add database name, username, and password into the .env configuration file to connect the laravel app to the database:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=database\_name

DB\_USERNAME=database\_user\_name

DB\_PASSWORD=database\_password

Bash

The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

php artisan migrate

Bash

Set Up Auth Controller

Next, type the suggested command on the command prompt and execute the command to generate a new controller file by the name of CustomAuthController.

php artisan make:controller CustomAuthController

Bash

Thereafter, open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Hash;

use Session;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class CustomAuthController extends Controller

{

public function index()

{

return view('auth.login');

}

public function customLogin(Request $request)

{

$request->validate([

'email' => 'required',

'password' => 'required',

]);

$credentials = $request->only('email', 'password');

if (Auth::attempt($credentials)) {

return redirect()->intended('dashboard')

->withSuccess('Signed in');

}

return redirect("login")->withSuccess('Login details are not valid');

}

public function registration()

{

return view('auth.registration');

}

public function customRegistration(Request $request)

{

$request->validate([

'name' => 'required',

'email' => 'required|email|unique:users',

'password' => 'required|min:6',

]);

$data = $request->all();

$check = $this->create($data);

return redirect("dashboard")->withSuccess('You have signed-in');

}

public function create(array $data)

{

return User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::make($data['password'])

]);

}

public function dashboard()

{

if(Auth::check()){

return view('dashboard');

}

return redirect("login")->withSuccess('You are not allowed to access');

}

public function signOut() {

Session::flush();

Auth::logout();

return Redirect('login');

}

}

PHP

Create Auth Routes

This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\CustomAuthController;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

\*/

Route::get('dashboard', [CustomAuthController::class, 'dashboard']);

Route::get('login', [CustomAuthController::class, 'index'])->name('login');

Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');

Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');

Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');

Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');

PHP

Create Auth Blade View Files

You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

@extends('app')

@section('content')

<main class="login-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Login</h3>

<div class="card-body">

<form method="POST" action="{{ route('login.custom') }}">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email" class="form-control" name="email" required

autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control" name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label>

<input type="checkbox" name="remember"> Remember Me

</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Signin</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

PHP

You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

@extends('app')

@section('content')

<main class="signup-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Register User</h3>

<div class="card-body">

<form action="{{ route('register.custom') }}" method="POST">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Name" id="name" class="form-control" name="name"

required autofocus>

@if ($errors->has('name'))

<span class="text-danger">{{ $errors->first('name') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email\_address" class="form-control"

name="email" required autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control"

name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label><input type="checkbox" name="remember"> Remember Me</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Sign up</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

Head over to resources/views/ folder, then create the new dashboard.blade.php file, then add the given below code in the resources/views/dashboard.blade.php file:

<!DOCTYPE html>

<html>

<head>

<title>Custom Auth in Laravel</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<nav class="navbar navbar-light navbar-expand-lg mb-5" style="background-color: #e3f2fd;">

<div class="container">

<a class="navbar-brand mr-auto" href="#">PositronX</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"

aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

@guest

<li class="nav-item">

<a class="nav-link" href="{{ route('login') }}">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{{ route('register-user') }}">Register</a>

</li>

@else

<li class="nav-item">

<a class="nav-link" href="{{ route('signout') }}">Logout</a>

</li>

@endguest

</ul>

</div>

</div>

</nav>

@yield('content')

</body>

</html>

Laravel 8 Custom Auth Login and Registration Example

Step 1: Create Laravel App

Step 2: Connect to Database

Step 3: Set Up Auth Controller

Step 4: Create Auth Routes

Step 5: Create Auth Blade View Files

Step 6: Run Laravel Development Server

Create Laravel App

// We assume you have already configured Composer on your system, run the following command to install the new laravel app. However, you can skip this step if the app is already installed.

composer create-project --prefer-dist laravel/laravel laravel\_demo\_app

Head over to app folder:

cd laravel\_demo\_app

Connect to Database

// To add database name, username, and password into the .env configuration file to connect the laravel app to the database:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=database\_name

DB\_USERNAME=database\_user\_name

DB\_PASSWORD=database\_password

// The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

php artisan migrate

Set Up Auth Controller

// Type the suggested command on the command prompt and execute the command to generate a new controller file by the name of CustomAuthController.

php artisan make:controller CustomAuthController

// open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Hash;

use Session;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class CustomAuthController extends Controller

{

public function index()

{

return view('auth.login');

}

public function customLogin(Request $request)

{

$request->validate([

'email' => 'required',

'password' => 'required',

]);

$credentials = $request->only('email', 'password');

if (Auth::attempt($credentials)) {

return redirect()->intended('dashboard')

->withSuccess('Signed in');

}

return redirect("login")->withSuccess('Login details are not valid');

}

public function registration()

{

return view('auth.registration');

}

public function customRegistration(Request $request)

{

$request->validate([

'name' => 'required',

'email' => 'required|email|unique:users',

'password' => 'required|min:6',

]);

$data = $request->all();

$check = $this->create($data);

return redirect("dashboard")->withSuccess('You have signed-in');

}

public function create(array $data)

{

return User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::make($data['password'])

]);

}

public function dashboard()

{

if(Auth::check()){

return view('dashboard');

}

return redirect("login")->withSuccess('You are not allowed to access');

}

public function signOut() {

Session::flush();

Auth::logout();

return Redirect('login');

}

}

Create Auth Routes

// This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\CustomAuthController;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

\*/

Route::get('dashboard', [CustomAuthController::class, 'dashboard']);

Route::get('login', [CustomAuthController::class, 'index'])->name('login');

Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');

Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');

Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');

Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');

Create Auth Blade View Files

// You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

@extends('app')

@section('content')

<main class="login-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Login</h3>

<div class="card-body">

<form method="POST" action="{{ route('login.custom') }}">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email" class="form-control" name="email" required

autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control" name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label>

<input type="checkbox" name="remember"> Remember Me

</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Signin</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

// You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

@extends('app')

@section('content')

<main class="signup-form">

<div class="cotainer">

<div class="row justify-content-center">

<div class="col-md-4">

<div class="card">

<h3 class="card-header text-center">Register User</h3>

<div class="card-body">

<form action="{{ route('register.custom') }}" method="POST">

@csrf

<div class="form-group mb-3">

<input type="text" placeholder="Name" id="name" class="form-control" name="name"

required autofocus>

@if ($errors->has('name'))

<span class="text-danger">{{ $errors->first('name') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="text" placeholder="Email" id="email\_address" class="form-control"

name="email" required autofocus>

@if ($errors->has('email'))

<span class="text-danger">{{ $errors->first('email') }}</span>

@endif

</div>

<div class="form-group mb-3">

<input type="password" placeholder="Password" id="password" class="form-control"

name="password" required>

@if ($errors->has('password'))

<span class="text-danger">{{ $errors->first('password') }}</span>

@endif

</div>

<div class="form-group mb-3">

<div class="checkbox">

<label><input type="checkbox" name="remember"> Remember Me</label>

</div>

</div>

<div class="d-grid mx-auto">

<button type="submit" class="btn btn-dark btn-block">Sign up</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</main>

@endsection

// Head over to resources/views/ folder, then create the new dashboard.blade.php file, then add the given below code in the resources/views/dashboard.blade.php file:

<!DOCTYPE html>

<html>

<head>

<title>Custom Auth in Laravel</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<nav class="navbar navbar-light navbar-expand-lg mb-5" style="background-color: #e3f2fd;">

<div class="container">

<a class="navbar-brand mr-auto" href="#">PositronX</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"

aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

@guest

<li class="nav-item">

<a class="nav-link" href="{{ route('login') }}">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{{ route('register-user') }}">Register</a>

</li>

@else

<li class="nav-item">

<a class="nav-link" href="{{ route('signout') }}">Logout</a>

</li>

@endguest

</ul>

</div>

</div>

</nav>

@yield('content')

</body>

</html>

Run Laravel Development Server

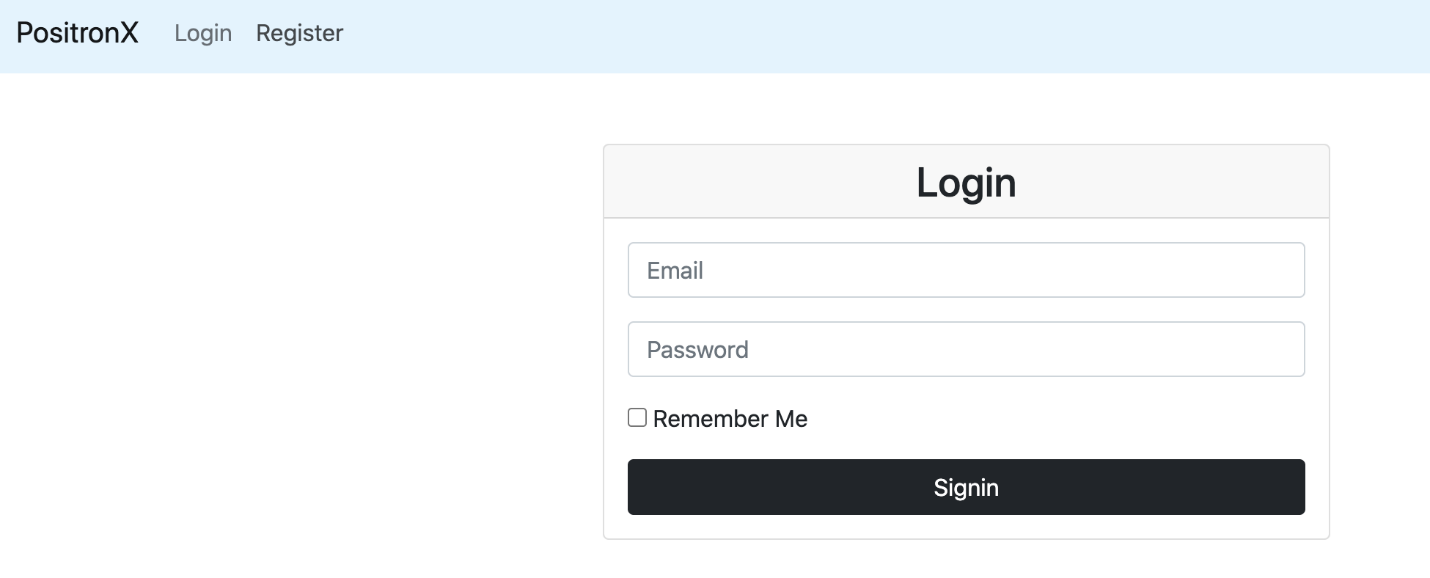
// Finally, we need to run the laravel development server, which will help us start the app on the browser. Ensure that you execute the given below command through the command prompt.

php artisan serve

Add the following url on the browser’s address bar and test the app recklessly.

<http://127.0.0.1:8000/login>

**Output**



**References:**

* <https://www.positronx.io/laravel-custom-authentication-login-and-registration-tutorial/>