



LARAVEL EBOOK BAHASA MELAYU

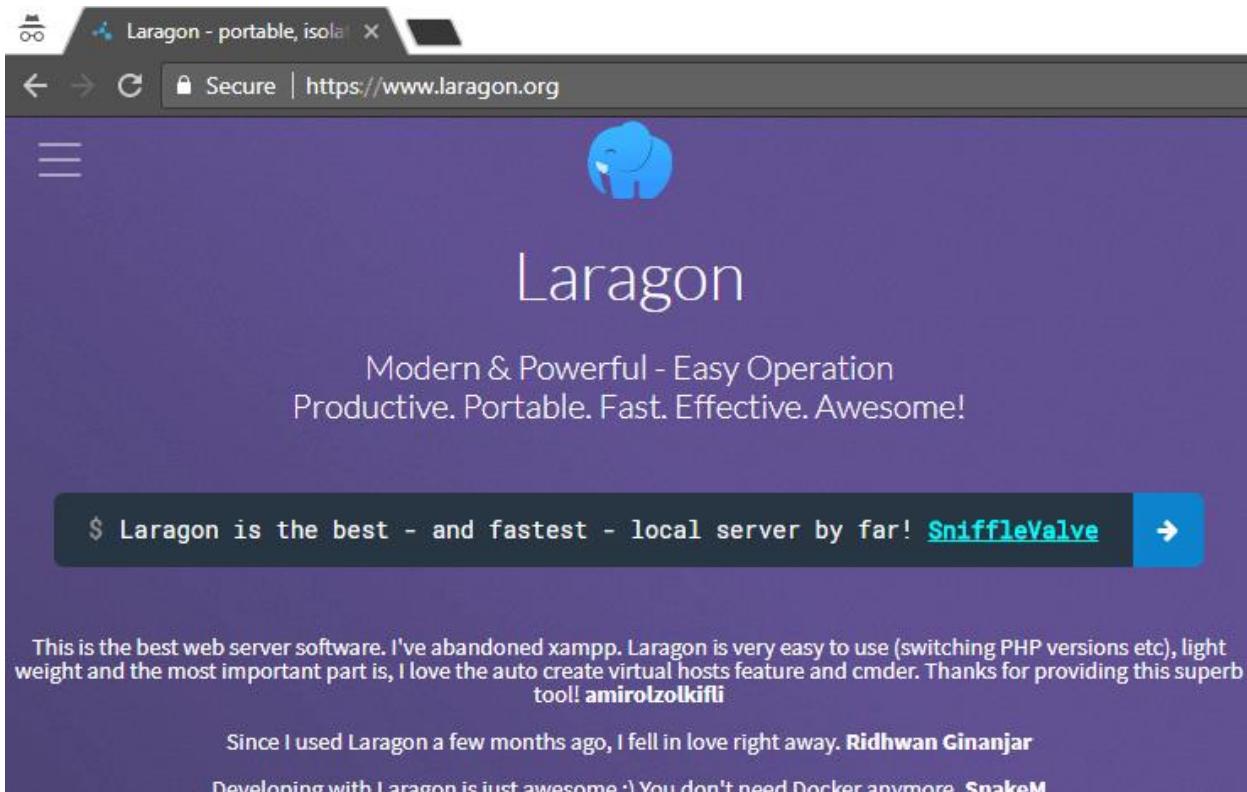
KHIRULNIZAM ABD RAHMAN

khirulnizam.com

Isi kandungan tutorial LARAVEL

Tutorial 01 Pengenalan Laravel.....	2
Tutorial 02 CRUD Pengenalan Laravel.....	18
Tutorial 03 Ubah design Antaramuka.....	35
Tutorial 04 Carian Rekod.....	44
Tutorial 05 Kemaskini dan Padam Rekod	48
Tutorial 06 Paparan carian dengan <i>Pagination</i>	60
Tutorial 07 Laravel REST API.....	63
Tutorial 08 Upload imej ke sistem Laravel	66
Tutorial 09 Peranan pengguna (user roles ACL).....	78
Tutorial 10 PHPStorm IDE Laravel	93
Tambahan: MVC dan Penulis	104

Tutorial 01 Pengenalan Laravel



This is the best web server software. I've abandoned xampp. Laragon is very easy to use (switching PHP versions etc), light weight and the most important part is, I love the auto create virtual hosts feature and cmder. Thanks for providing this superb tool! [amirozolkifli](#)

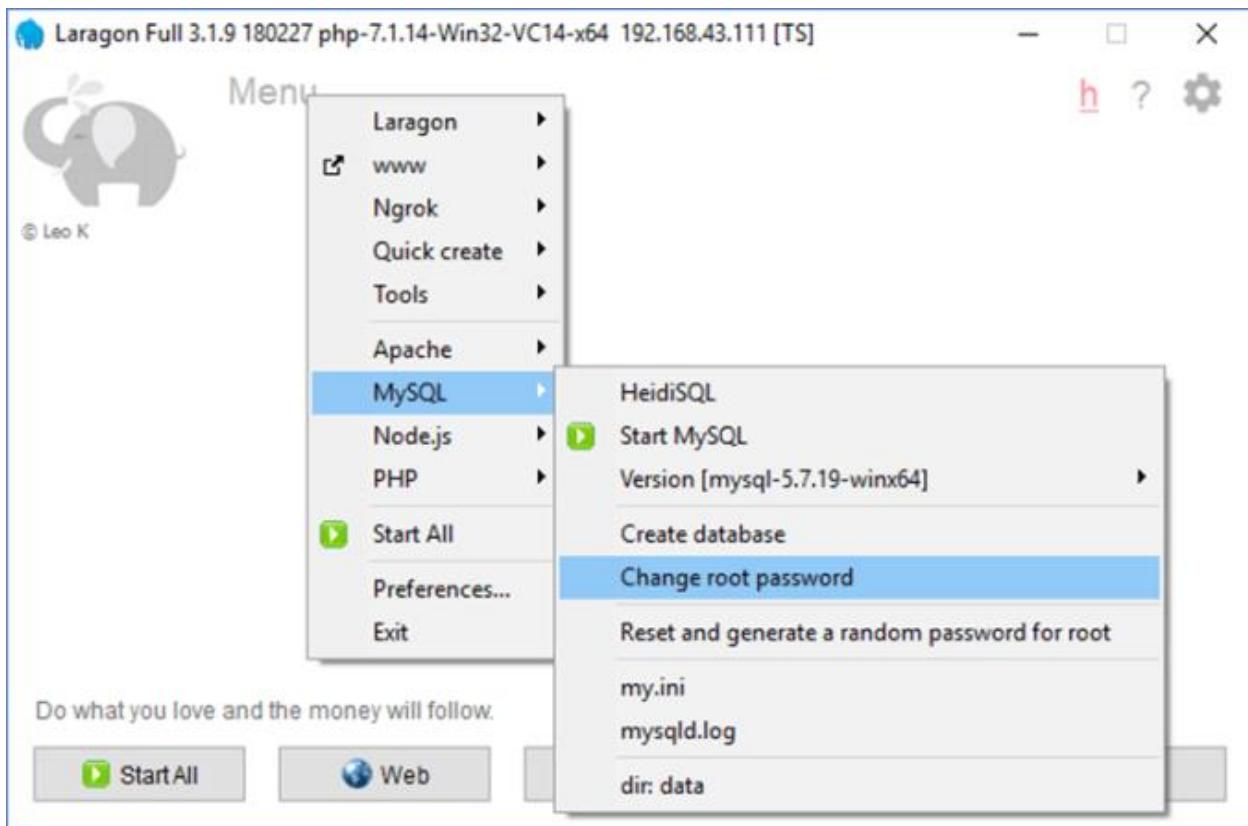
Since I used Laragon a few months ago, I fell in love right away. [Ridhwan Ginanjar](#)

Developing with Laragon is just awesome :) You don't need Docker anymore. [SnakeM](#)

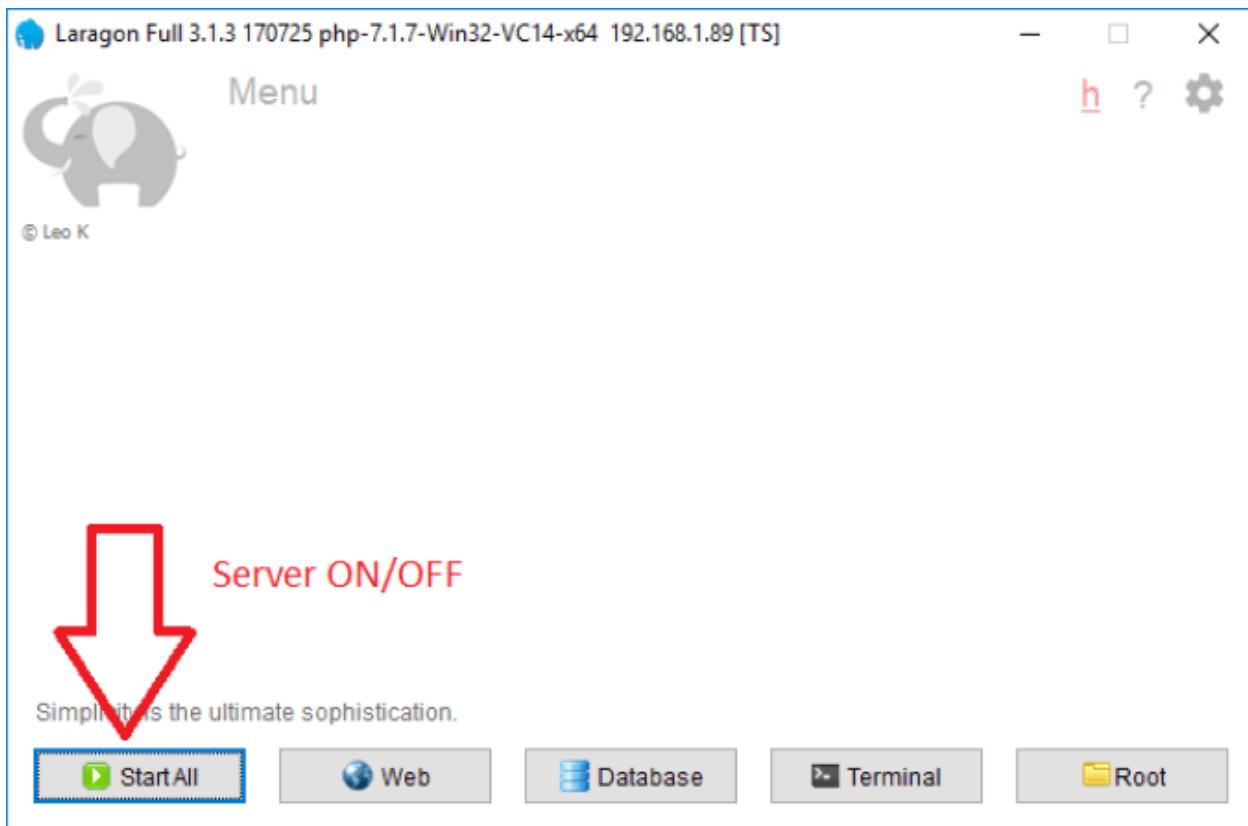
Tutorial Laravel bahasa Melayu ini dimulakan dengan pemasangan LARAGON. Laragon adalah satu perisian kompilasi yang telah disediakan semua keperluan perisian pembangunan projek Laravel.

Laragon (WAMP-Windows Apache, MySQL, PHP) terdiri daripada; Apache 2.4, Nginx 1.12, MySQL 5.7, PHP 7.1.7, Node.js 6.11, git, ... [muatturun di sini Laragon – WAMP \(112 MB\)](#)

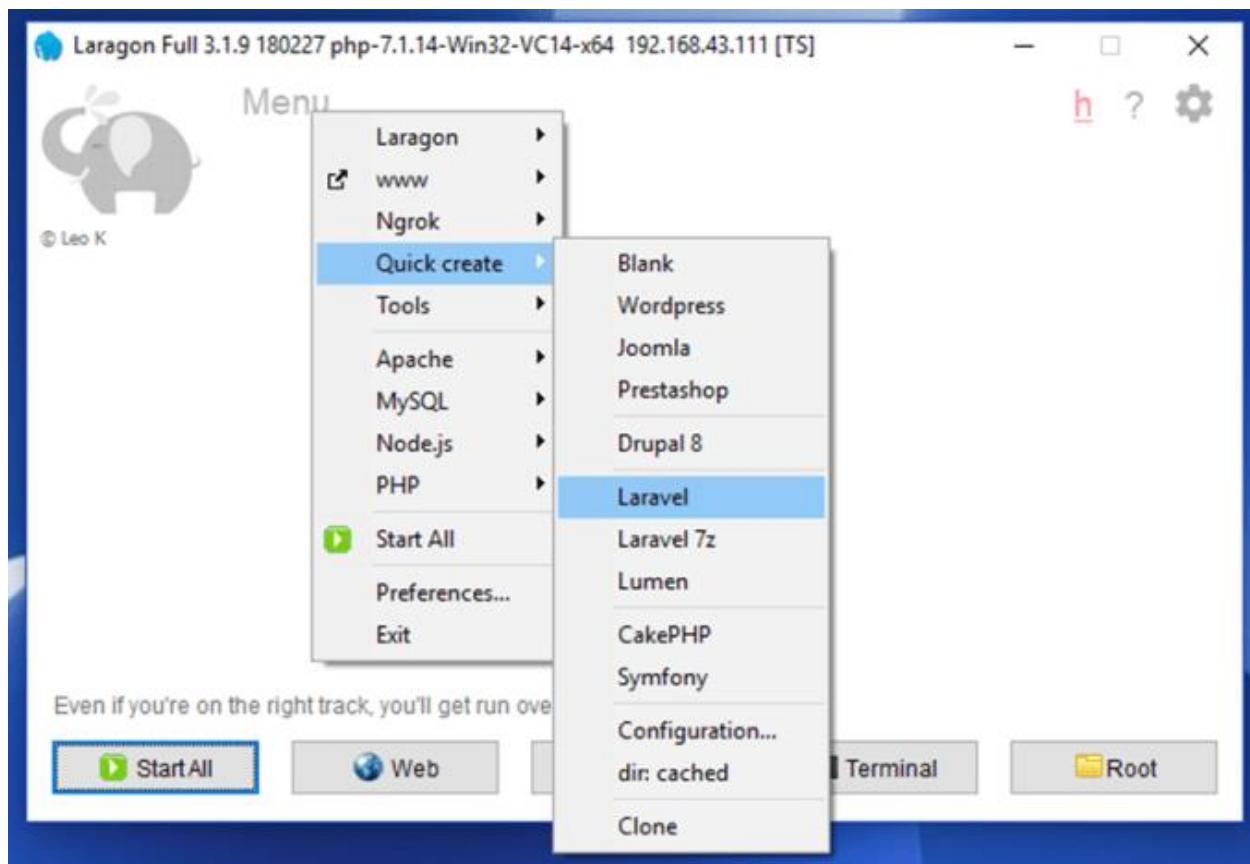
Selepas berjaya install Laragon, anda akan melihat paparan berikut. Pertamanya sila tukar kata laluan pengguna pangakalan data *root* (*root database username*) .



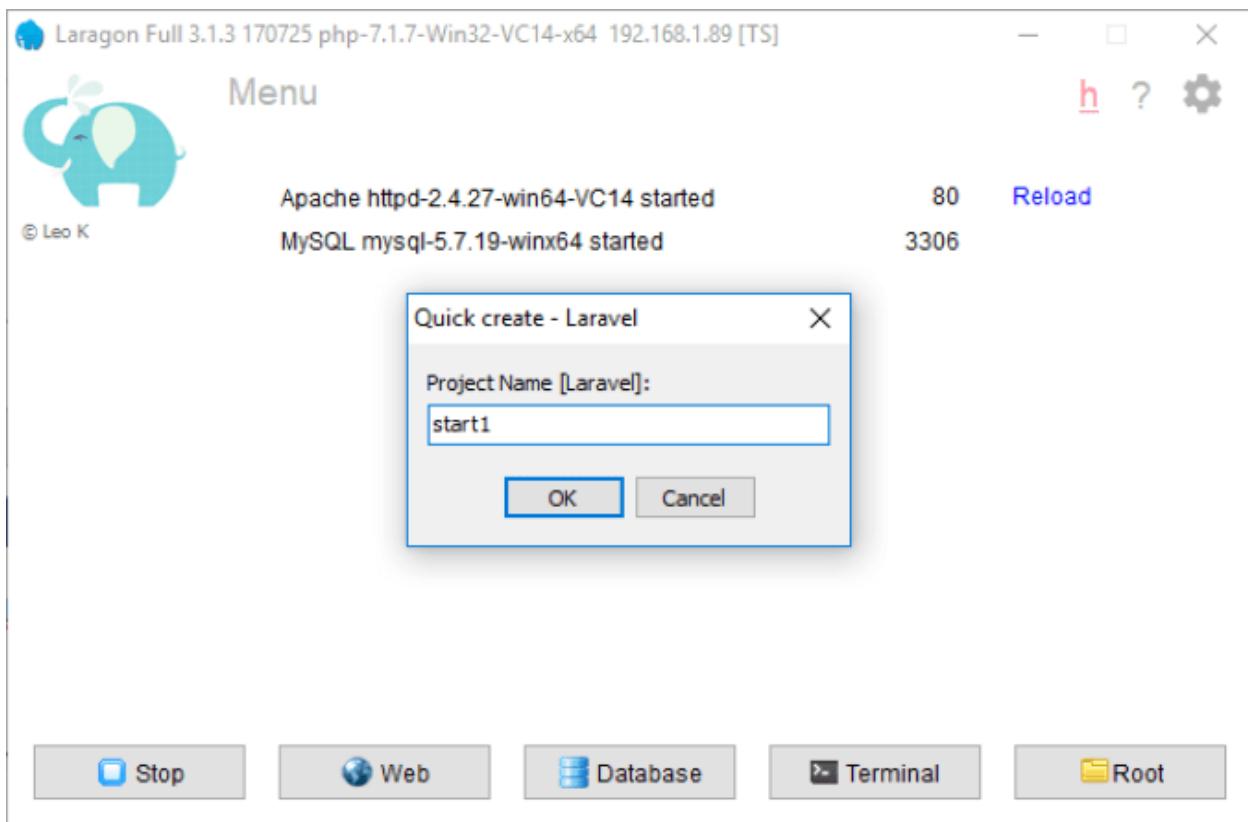
Kemudian ON web (*Apache*) dan server pangkalan data (*mysql*) dengan tekan pada butang “Start All”.



Untuk memulakan projek Laravel yang baharu, klik-kanan pada paparan Quick Create -> Laravel. Tindakan ini akan memuat-turun fail-fail template kod projek Laravel dari server repository Laravel. Sila pastikan Internet berfungsi dengan baik.



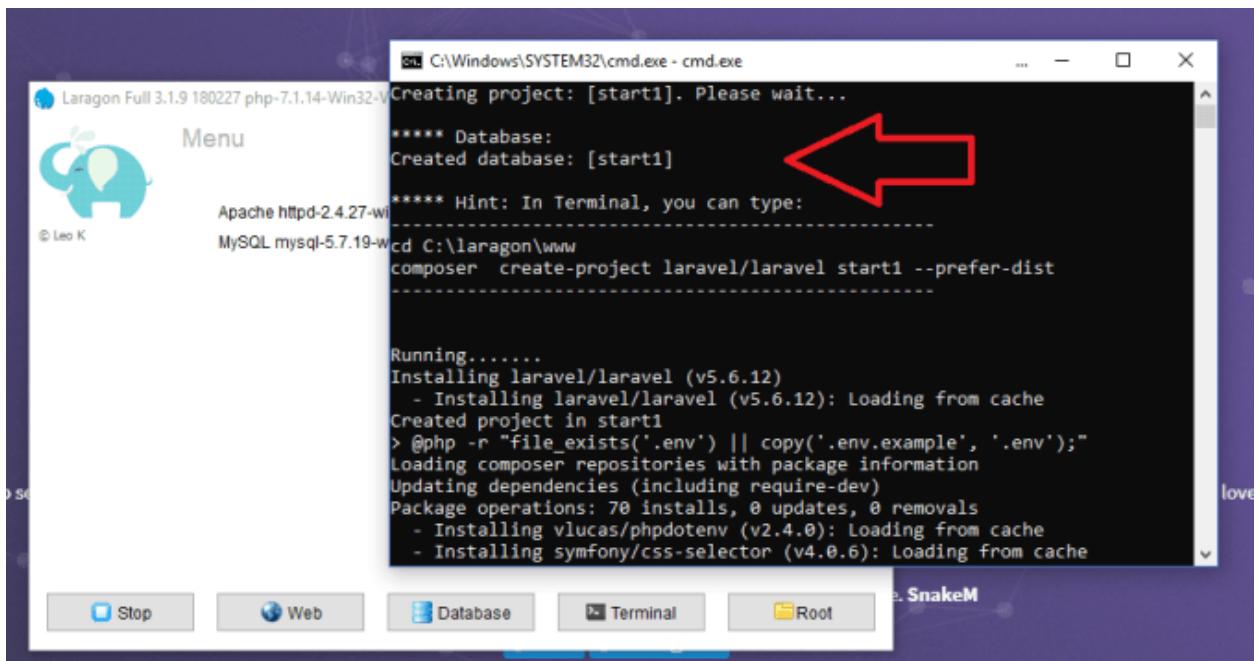
Masukkan nama projek Laravel; contoh **start1**



Paparan pada cmd.exe menunjukkan beberapa langkah untuk menjana projek baharu Laravel.

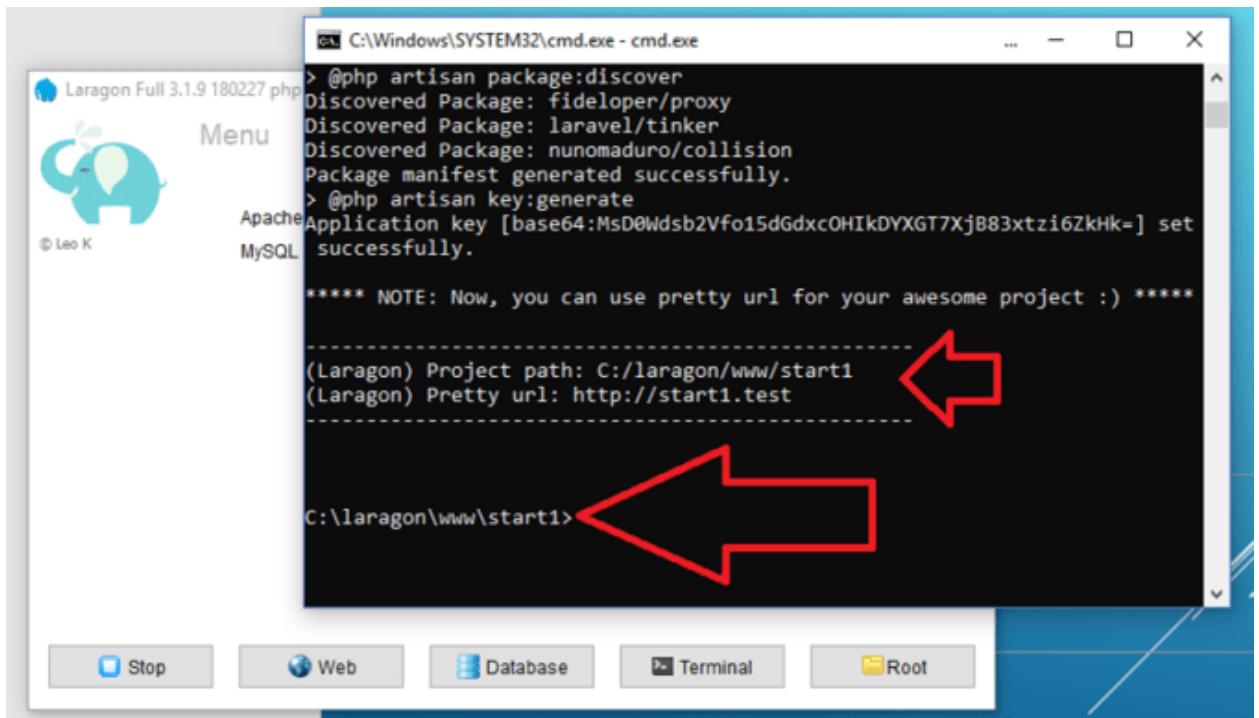
Pangkalan data projek akan dicipta

Fail-fail template projek Laravel akan dimuat turun dari server repository (sila bersabar mungkin akan mengambil masa tertakluk kepada kelajuan Internet atau rangkaian anda)



```
C:\Windows\SYSTEM32\cmd.exe - cmd.exe
Creating project: [start1]. Please wait...
***** Database:
Created database: [start1]
***** Hint: In Terminal, you can type:
cd C:\laragon\www
composer create-project laravel/laravel start1 --prefer-dist
-----
Running.....
Installing laravel/laravel (v5.6.12)
- Installing laravel/laravel (v5.6.12): Loading from cache
Created project in start1
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 70 installs, 0 updates, 0 removals
- Installing vlucas/phpdotenv (v2.4.0): Loading from cache
- Installing symfony/css-selector (v4.0.6): Loading from cache
```

...rajab bersambung...



```
C:\Windows\SYSTEM32\cmd.exe - cmd.exe
> @php artisan package:discover
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
> @php artisan key:generate
Application key [base64:MsD0Wdsb2Vfo15dGdxcOHIkDYXGT7XjB83xtzi6ZkHk=] set
successfully.

***** NOTE: Now, you can use pretty url for your awesome project :) *****

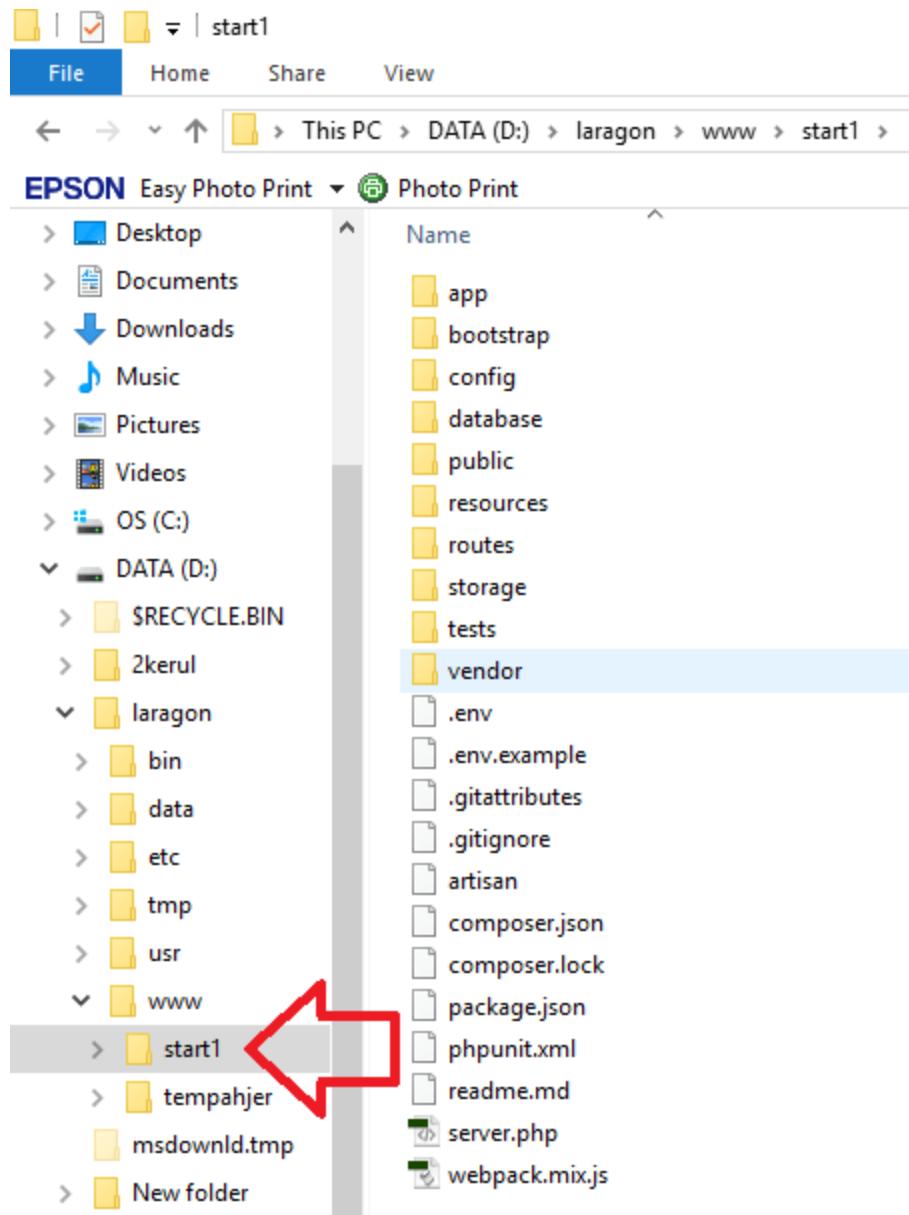
(Laragon) Project path: C:/laragon/www/start1
(Laragon) Pretty url: http://start1.test
```

Uji server anda dengan menaip start1.test

Sepatutnya paparan teks Laravel seperti dibawah akan kelihatan. Pastikan paparan seperti berikut untuk meneruskan langkah seterusnya. Sekiranya

server tidak dalam keadaan baik, atau keluar paparan laman tidak dijumpai boleh gunakan arahan php artisan serve.

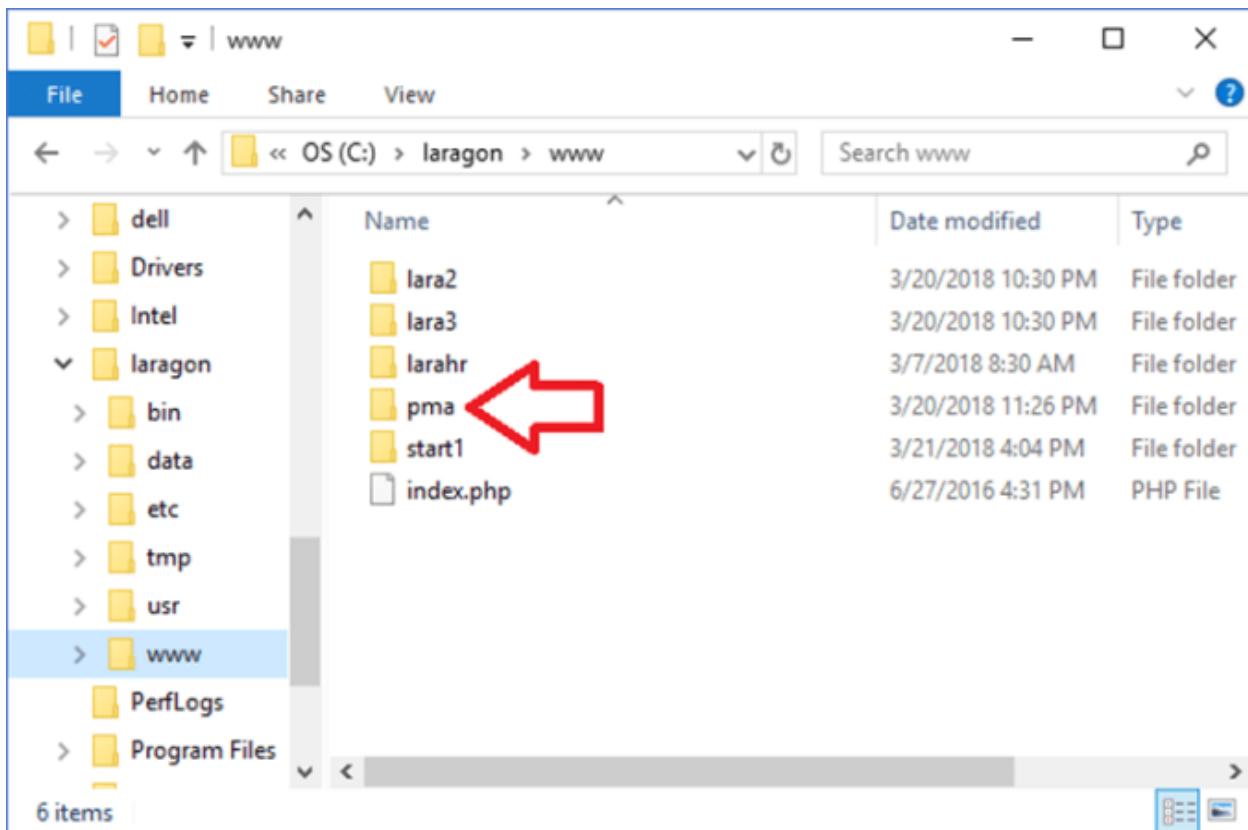
Berikut struktur fail projek Laravel yang akan diperolehi selepas selesai muatturun.



Penyelenggara Pangkalan data (*database admin*)

Proses seterusnya adalah untuk memasang database admin. Boleh pilih web phpmyadmin, atau GUI lain seperti SQLYOG atau NAVICAT dll. Di sini kami tunjukkan phpmyadmin (PMA).

Download PMA dari phpmyadmin.net dan extract file zip ke dalam folder laragon/www



PMA cuma diperlukan untuk semak pangkalan data / jadual (tables) berfungsi dengan baik. Setakat cipta projek baharu tadi, hanya pangkalan data yang disediakan. Jadual belum ada lagi. Bina jadual kita kena laksanakan proses *migrate*.

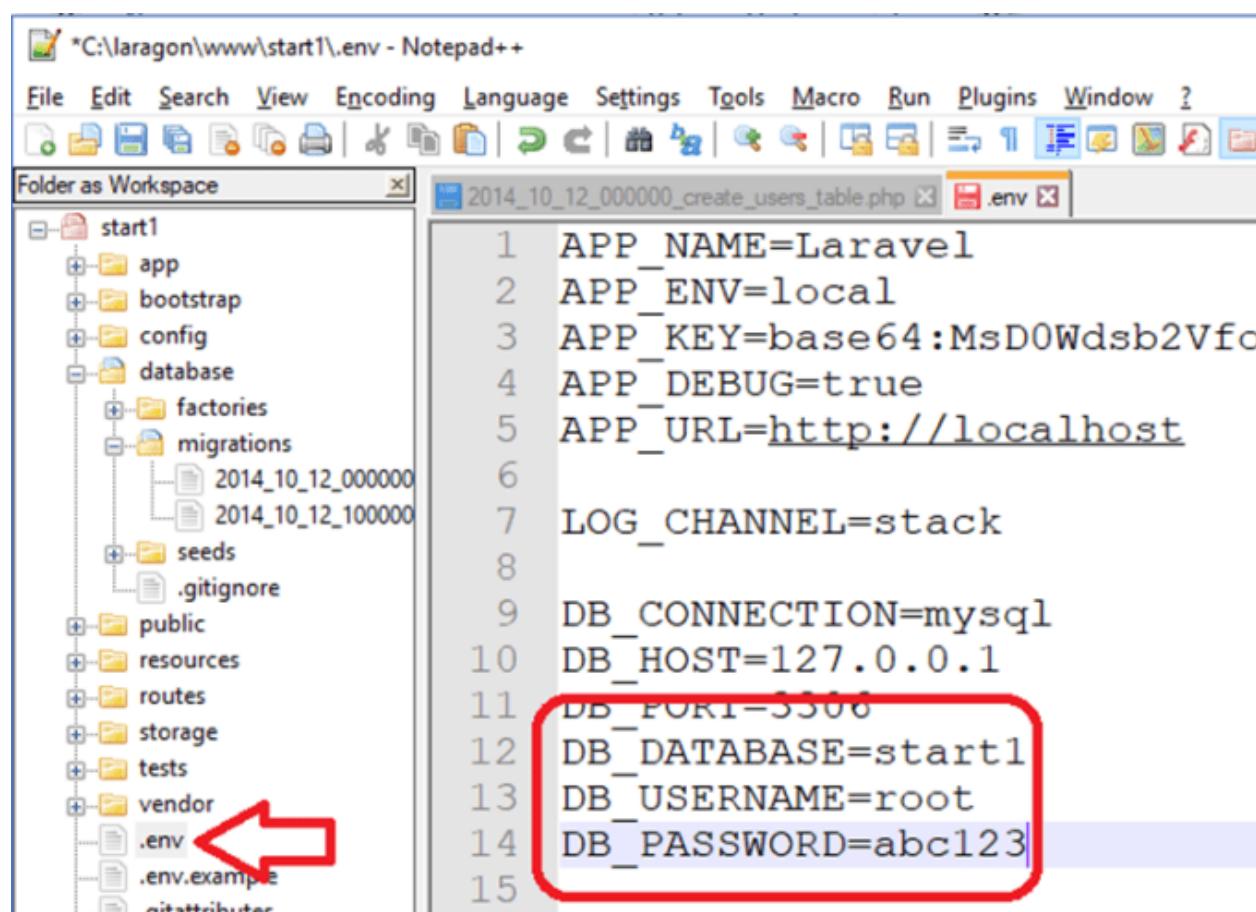
A screenshot of a web browser displaying the phpMyAdmin interface. The URL in the address bar is localhost/pma/db_structure.php?server=1&db=start1. The page title is 'localhost / localhost / start1'. The left sidebar shows a tree view of databases: New, information_schema, mysql, performance_schema, start1 (selected), sys, and tempahjer. The main content area shows the 'Structure' tab for the 'start1' database. A message 'No tables found in database.' is displayed. Below it is a 'Create table' form with fields for 'Name:' and 'Number of columns: 4'.

Latihan *User authentication*

Pangkalan data telah tersedia. Sekarang kita akan menambah modul pengurusan pengguna (*user authentication*).

Untuk menyambungkan projek Laravel kepada pangkalan data yang tersedia perlu diubah pada tetapan pangkalan data dalam fail **env**.

Gunakan pengedit kod kegemaran anda; Notepad++, Atom, Sublime atau vscode, dll. Ubah tetapan seperti dalam server database anda. Tutorial ini menggunakan Notepad++ untuk menulis kod Laravel, PHP atau blade. Editor (IDE) terbaik untuk mengedit kod Laravel yang terbaik setakat ini adalah PHPStorm (berbayar), boleh juga cuba Eclipse for PHP/web.



The screenshot shows a Notepad++ interface with two tabs: '2014_10_12_000000_create_users_table.php' and '.env'. The left pane shows a file tree for a Laravel project named 'start1'. A red arrow points to the '.env' file in the tree. The right pane displays the contents of the '.env' file:

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:MsD0Wdsb2Vfo
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=start1
13 DB_USERNAME=root
14 DB_PASSWORD=abc123
15
```

A red box highlights the database connection details (DB_PORT, DB_DATABASE, DB_USERNAME, DB_PASSWORD) in lines 11 through 14.

Untuk menambahkan modul pengguna (**auth**) dalam projek web Laravel.

Authentication Quickstart

Laravel ships with several pre-built authentication controllers, which are located in the `App\Http\Controllers\Auth` namespace. The `RegisterController` handles new user registration, the `LoginController` handles authentication, the `ForgotPasswordController` handles e-mailing links for resetting passwords, and the `ResetPasswordController` contains the logic to reset passwords. Each of these controllers uses a trait to include their necessary methods. For many applications, you will not need to modify these controllers at all.

Routing

Laravel provides a quick way to generate routes for your authentication system using one simple command:

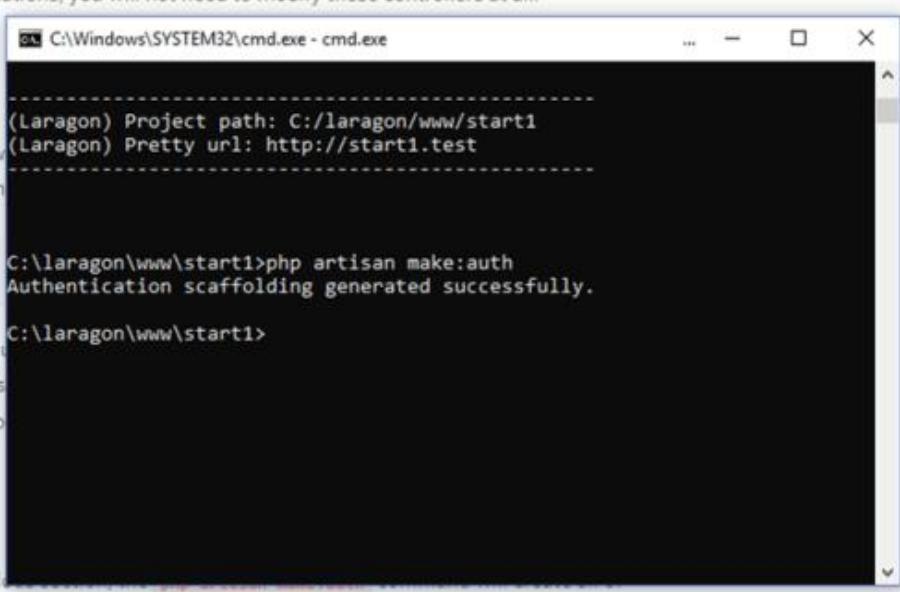
```
php artisan make:auth
```

Authentication scaffolding generated successfully.

This command should be run from within your Laravel application directory and login views, as well as controller classes, should be generated to handle post requests.

Views

As mentioned in the previous section, the `make:auth` command will generate



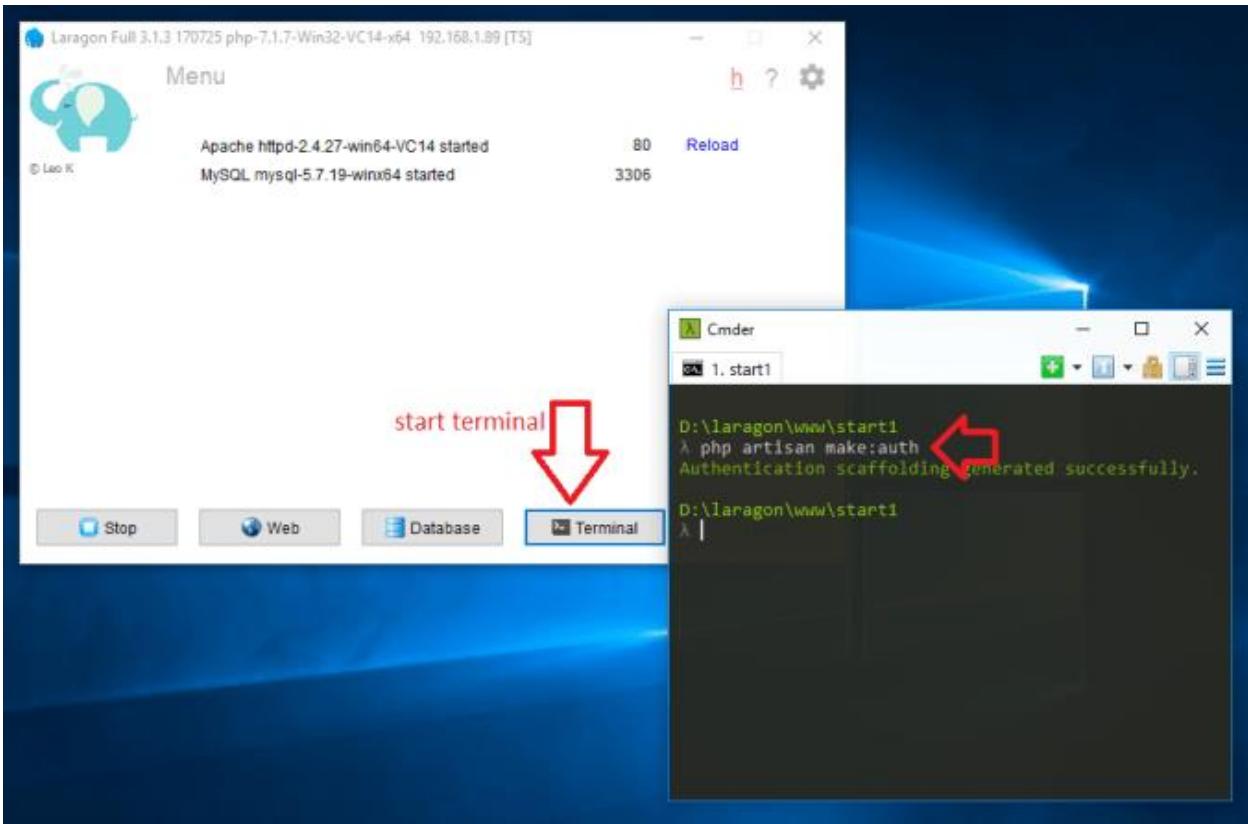
The screenshot shows a Windows Command Prompt window titled "C:\Windows\SYSTEM32\cmd.exe - cmd.exe". The window displays the following text:

```
(Laragon) Project path: C:/laragon/www/start1
(Laragon) Pretty url: http://start1.test
-----
C:\laragon\www\start1>php artisan make:auth
Authentication scaffolding generated successfully.

C:\laragon\www\start1>
```

Laksanakan arahan berikut dalam terminal untuk menambah modul auth,

```
php artisan make:auth
```



Sekiranya kita nak membuat perbandingan nama dan katalaluan pengguna () dengan maklumat disimpan dalam pangkalan data, semestinya kita ada jadual untuk menyimpan maklumat tersebut. Kaedah Laravel menggalakkan kita isytihar struktur jadual di dalam folder ***migration***. Edit fail `create_users_table.php` dalam folder `migration`. Tambah kod dalam fungsi `up()` seperti berikut.

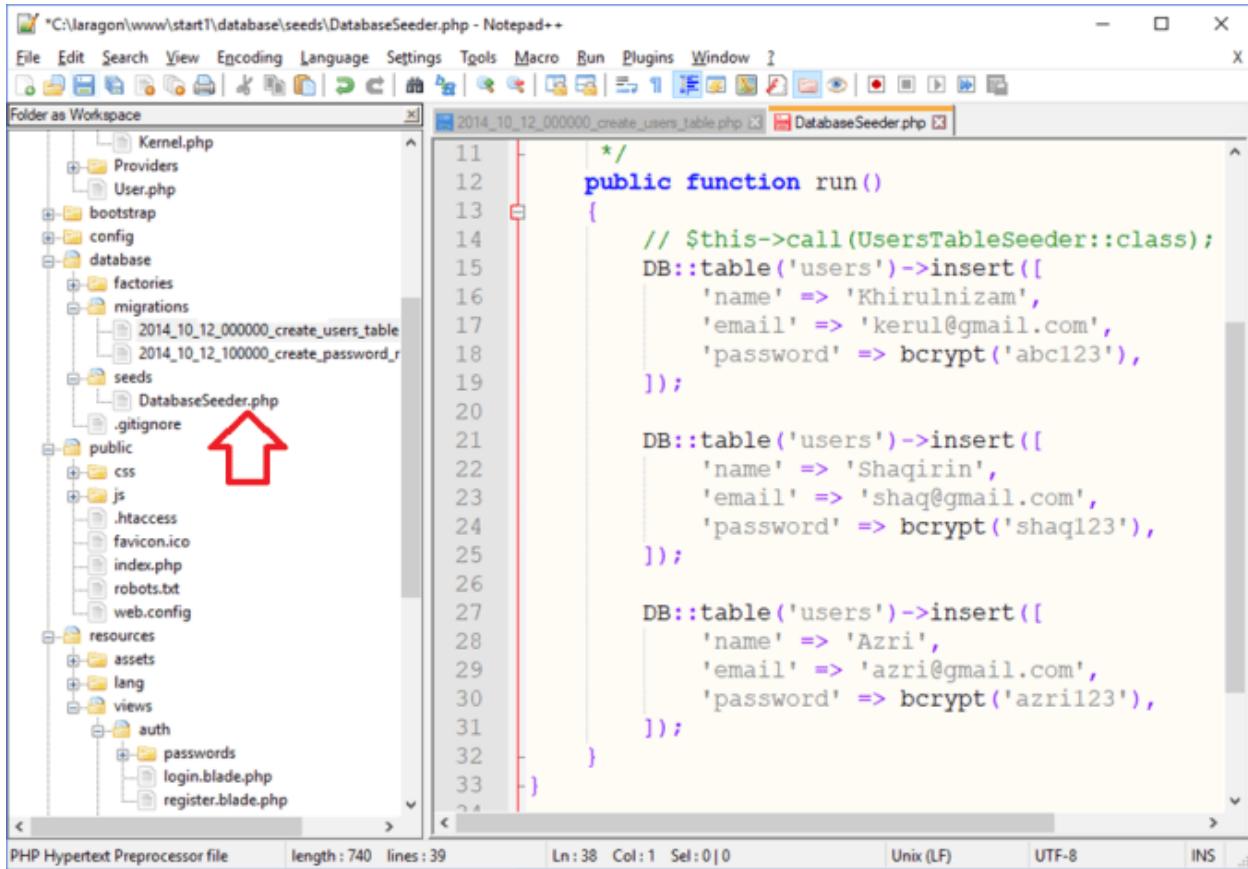
The screenshot shows a Windows desktop environment. In the top-left, there's a file explorer window titled 'Folder as Workspace' showing a project structure with a 'migrations' folder highlighted. A red arrow points to this folder with the label 'migration folder'. In the top-right, a Notepad++ window is open, displaying the contents of a file named '2014_10_12_000000_create_users_table.php'. The code defines a migration class with an 'up()' method that creates a 'users' table with columns for id, name, email (unique), password, and rememberToken. A red arrow points to the command prompt window below. In the bottom-left, a command prompt window titled 'cmd.exe - cmd.exe' is open, showing the output of the command 'php artisan migrate'. The output indicates that the migration table was created successfully, and two migrations were run: '2014_10_12_000000_create_users_table' and '2014_10_12_100000_create_password_resets_table'. A red arrow points to the word 'migrate' in the command.

Selepas selesai kod pengisytiharan jadual dimasukkan, laksanakan arahan berikut untuk mencipta jadual ke dalam pangkalan data;

`php artisan migrate`

Rujuk dalam localhost/pma untuk melihat struktur jadual telah dicipta dalam pangkalan data.

Seterusnya dalam DatabaseSeeder.php kita akan masukkan beberapa contoh rekod. Ada banyak cara untuk jana data contoh (dummy data). Laravel juga menyediakan penjana data (*data factory*). Akan dibincangkan dalam tutorial akan datang.



The screenshot shows the Notepad++ interface with the following details:

- Title Bar:** *C:\laragon\www\start1\database\seeds\DatabaseSeeder.php - Notepad++
- Menu Bar:** File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window I
- Toolbar:** Standard icons for file operations.
- File List:** Shows the project structure:
 - Kernel.php
 - Providers
 - User.php
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - 2014_10_12_000000_create_users_table
 - 2014_10_12_100000_create_password_r
 - seeds
 - DatabaseSeeder.php
 - public
 - css
 - js
 - .htaccess
 - favicon.ico
 - index.php
 - robots.txt
 - web.config
 - resources
 - assets
 - lang
 - views
 - auth
 - passwords
 - login.blade.php
 - register.blade.php

- Code Editor:** Displays the `DatabaseSeeder.php` file content:

```
11 */  
12 public function run()  
13 {  
14     // $this->call('UsersTableSeeder::class');  
15     DB::table('users')->insert([  
16         'name' => 'Khirulnizam',  
17         'email' => 'kerul@gmail.com',  
18         'password' => bcrypt('abc123'),  
19     ]);  
20  
21     DB::table('users')->insert([  
22         'name' => 'Shaqirin',  
23         'email' => 'shaq@gmail.com',  
24         'password' => bcrypt('shaql123'),  
25     ]);  
26  
27     DB::table('users')->insert([  
28         'name' => 'Azri',  
29         'email' => 'azri@gmail.com',  
30         'password' => bcrypt('azri123'),  
31     ]);  
32 }  
33 }
```
- Status Bar:** PHP Hypertext Preprocessor file length : 740 lines : 39 Ln : 38 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS ...

Data yang telah disediakan dalam DatabaseSeeder perlu dimasukkan ke dalam jadual (database). Boleh dilakukan dengan melaksana arahan berikut pada terminal;

`php artisan db:seed`

Dan hasilnya sila lihat dalam PMA jadual users telah diisi dengan data contoh.

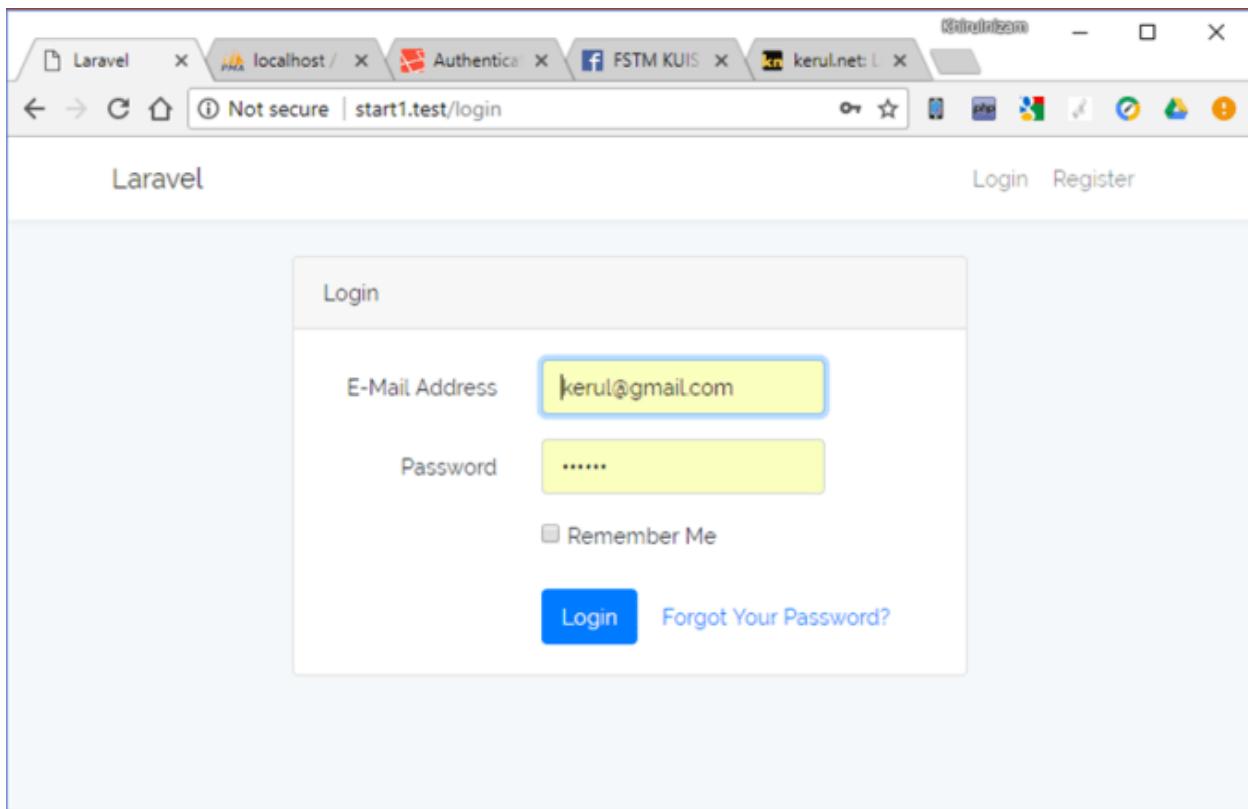
The screenshot shows a browser window with several tabs open, including 'Laravel', 'localhost /', 'Authentica', 'FSTM KUIS', and 'kerul.net: L'. Below the tabs is the 'phpMyAdmin' interface, showing the 'users' table in the 'start1' database. The table has three rows with columns: id, name, email, and password. The terminal window below shows command-line output for Laravel artisan commands:

```
C:\laragon\www\start1>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table

C:\laragon\www\start1>php artisan make:seeder DatabaseSeeder
Seeder already exists!
^C
C:\laragon\www\start1>php artisan db:seed
C:\laragon\www\start1>php artisan db:seed --class=DatabaseSeeder
```

A red arrow points from the text 'DatabaseSeeder already exists!' to the command 'db:seed'.

Arahan pada make:auth tadi juga menjana antaramuka login/logout pengguna. Cuma perlu papar dalam browser, start1.test/login dan anda akan mendapat paparan di bawah yang berfungi untuk login/logout dan daftar pengguna. Selamat mencuba dan berjaya.



Selesai tutorial pertama Laravel. Komen/cadangan/bantuan sila email khirulnizam{at}gmail.com.
Bersambung seterusnya nanti. Insya-Allah...

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>

(Slide for English version is available at <http://bit.ly/laravelfstm1>)

Tutorial 02 CRUD Pengenalan Laravel

ID	Name	Desc	Action
1	Laravel 101	Basic laravel	
2	Android1	Basic Android	
3	Android 2	Intermediate Android	
4	PHP & MySQL	PHP & MySQL . Beginner to Intermediaite. 3 days	

Tutorial Laravel bahasa Melayu ini disambung dengan pemasangan LARAGON. Laragon adalah satu perisian kompilasi yang telah disediakan semua keperluan perisian pembangunan projek Laravel.

Rangka tutorial

- Membina MODEL (jadual pangkalan data /db)
- Membina VIEW (antaramuka pengguna)
- CONTROLLER & route
- Masukkan rekod baharu
- Senaraikan semua rekod
- Kemaskini rekod

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>

Sebarang aplikasi web yang menjana laman web dinamik memerlukan pangkalan data. Kaedah penyambungan pangkalan data telah dibincangkan dalam tutorial Laravel 1. Kali ini kita cuma perlu menambah jadual (*table*) dalam DB yang sedia ada.

MEMBINA MODEL (jadual pangkalan data)

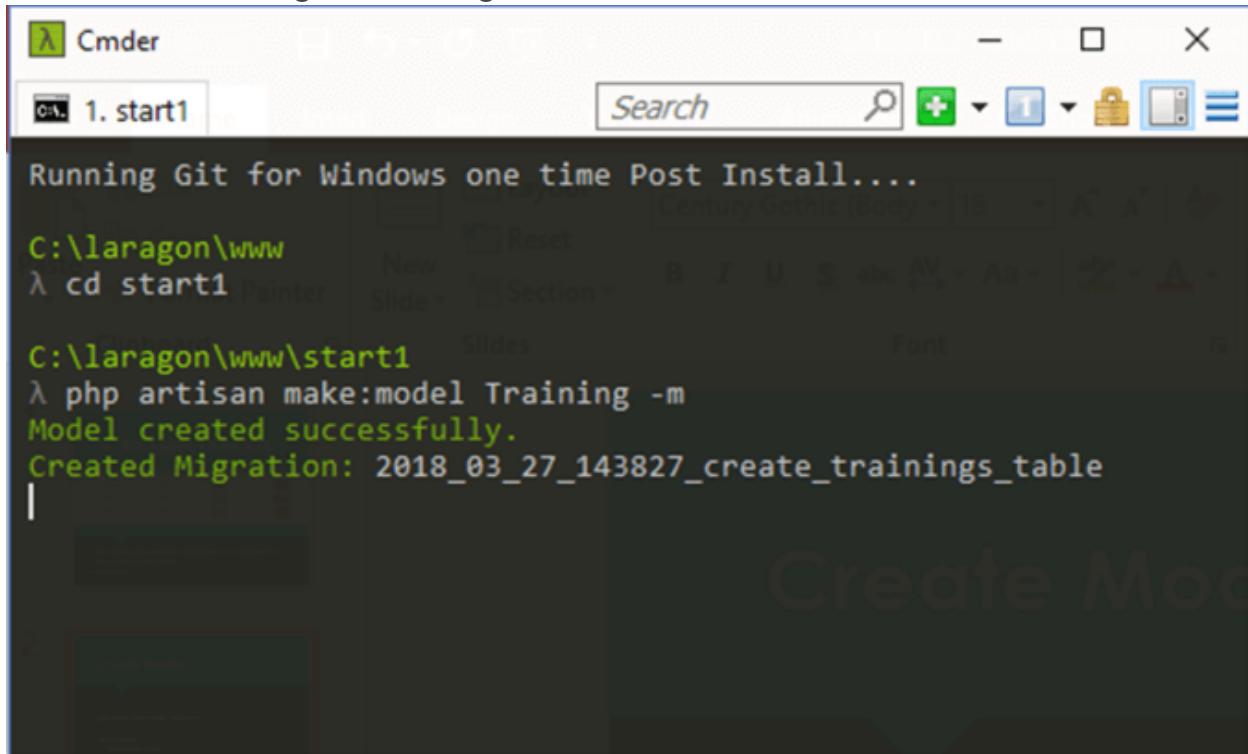
Laksanakan arahan berikut dalam *command shell* Laragon.

```
php artisan make:model Training -m
```

Arahan ini akan menyediakan dua item berikut dalam folder projek Laravel;

Fail model Training.php

Fail create_trainings_table migration



```
Running Git for Windows one time Post Install....  
C:\laragon\www  
λ cd start1  
λ php artisan make:model Training -m  
Model created successfully.  
Created Migration: 2018_03_27_143827_create_trainings_table
```

Dalam fail *create_trainings_table* , tulis kod berikut;

```

2018_03_27_143827_create_trainings_table.php 2014_10_12_000000_create_users_table.php DatabaseSeeder.php
9   /**
10    * Run the migrations.
11    *
12    * @return void
13    */
14    public function up()
15    {
16        Schema::create('trainings', function (Blueprint $table) {
17            $table->increments('id');
18            $table->string('trainingname');
19            $table->string('desc');
20            $table->string('trainer');
21            $table->timestamps();
22        });
23    }

```

Kembali ke *command shell* Laragon dan laksanakan arahan *migrate*. Arahan *migrate* akan membina jadual baharu dengan struktur seperti dalam *create_trainings_table*.

php artisan migrate

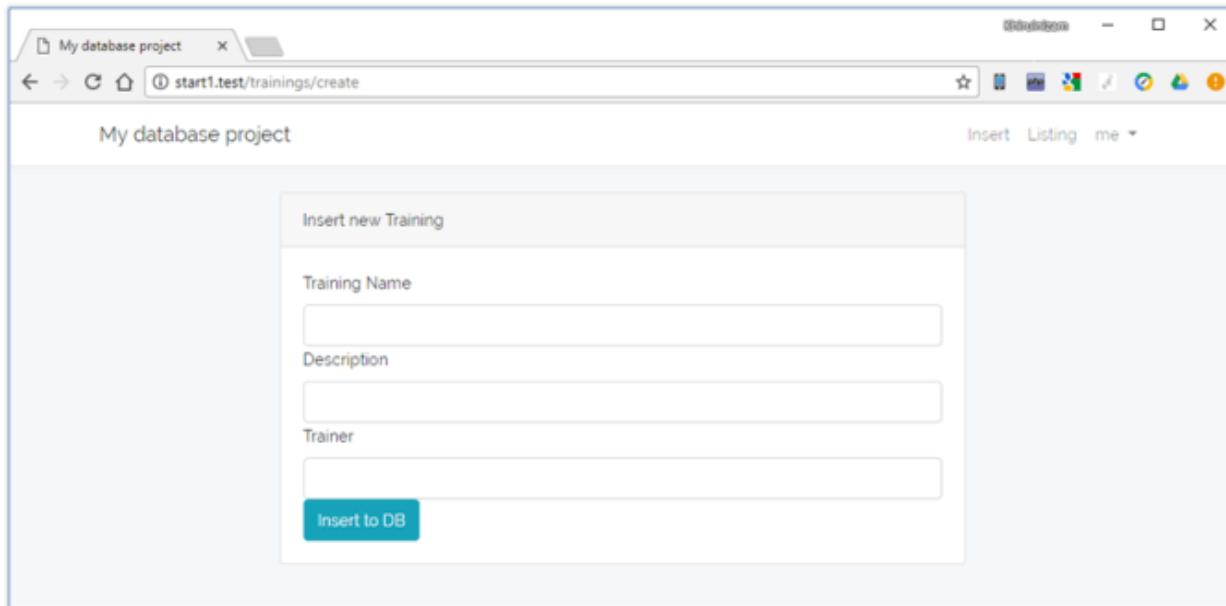
```

C:\laragon\www
λ cd start1
C:\laragon\www\start1
λ php artisan make:model Training -m
Model created successfully.
Created Migration: 2018_03_27_143827_create_trainings_table
C:\laragon\www\start1
λ php artisan migrate
Migrating: 2018_03_27_143827_create_trainings_table
Migrated: 2018_03_27_143827_create_trainings_table
C:\laragon\www\start1
λ |

```

BINA ANTARAMUKA (Views) untuk masukkan rekod baharu

Rajah di bawah adalah antaramuka pengguna yang akan dibina untuk menerima input rekod yang baharu untuk disimpan dalam jadual yang baharau iaitu *Training*.



Proses untuk membuat fail antaramuka tersebut;

Buat satu folder untuk antaramuka ***trainings*** dalam folder **resources > views**

Tambah satu fail **blade** baharu dengan nama ***create.blade.php***

```

    "C:\laragon\www\start1\resources\views\trainings\create.blade.php - Notepad+"
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 
Folder as Workspace
app.js
bootstrap.js
sass
app.scss
_variables.scss
lang
views
auth
passwords
login.blade.php
register.blade.php
layouts
app.blade.php
home.blade.php
welcome.blade.php
trainings
create.blade.php
routes
storage
tests
vendor
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package.json
phpunit.xml
1 @extends('layouts.app')
2 @section('content')
3 <div class="container">
4   <div class="row justify-content-center">
5     <div class="col-md-8">
6       <div class="card">
7         <div class="card-header">Insert new Training</div>
8
9         <div class="card-body">
10            <form method="post" action="#">
11              @csrf
12              <label for="usr">Training Name</label>
13              <input type="text" class="form-control" name="trainingname">
14
15              <label for="pwd">Description</label>
16              <input type="text" class="form-control" name="desc">
17
18              <label for="pwd">Trainer</label>
19              <input type="text" class="form-control" name="trainer">
20              <button type="submit" class="btn btn-info"> Insert to DB </button>
21            </form>
22
23          </div>
24
25        </div>
26      </div>
27    </div>
  
```

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

< div class = "container" >

< div class = "row justify-content-center" >
< div class = "col-md-8" >
< div class = "card" >
< div class = "card-header" >Insert new Training</ div >
< div class = "card-body" >
< form method = "post" action = "{{url('trainings')}}" >
@csrf
< label for = "usr" >Training Name</ label >
< input type = "text" class = "form-control" name =
"trainingname" >

< label for = "pwd" >Description</ label >
< input type = "text" class = "form-control" name =
"desc" >

< label for = "pwd" >Trainer</ label >
< input type = "text" class = "form-control" name =
"trainer" >
< button type = "submit" class = "btn btn-info" > Insert
to DB </ button >
</ form >

</ div >

</ div >

</ div >
```

```
</ div >

</ div >

@endsection
```

CONTROLLER & ROUTE

Seterusnya hubungkan antaramuka (*create.blade.php*) dengan pangkalan data, kita memerlukan Controller untuk melaksanakan aliran logik dan proses. Manakala *route* diperlukan sebagai laluan (*path*) kepada antaramuka/proses dalam *Controller* .

Untuk menambah Controller dalam modul Training ini, laksanakan arahan *make:controller* pada *Command Shell* .

```
php artisan make:controller TrainingController --resources
```

Arahan ini akan mencipta fail *TrainingController.php* . –resources bermaksud semua path yang mungkin terlibat untuk fungsi-fungsi dalam modul Training akan diwujudkan sama.

Cmder

1. start1

Model created successfully.
Created Migration: 2018_03_27_143827_create_trainings_table

```
C:\laragon\www\start1
λ php artisan migrate
Migrating: 2018_03_27_143827_create_trainings_table
Migrated: 2018_03_27_143827_create_trainings_table
```

```
C:\laragon\www\start1
λ php artisan make:controller TrainingController --resource
Controller created successfully.
```

```
C:\laragon\www\start1
λ
C:\laragon\www\start1
λ
```

Dalam fail **routes/web.php** tuliskan baris kod di bawah seperti ditunjuk anak panah.

Route::resource('trainings', 'TrainingController');

* C:\laragon\www\start1\routes\web.php - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

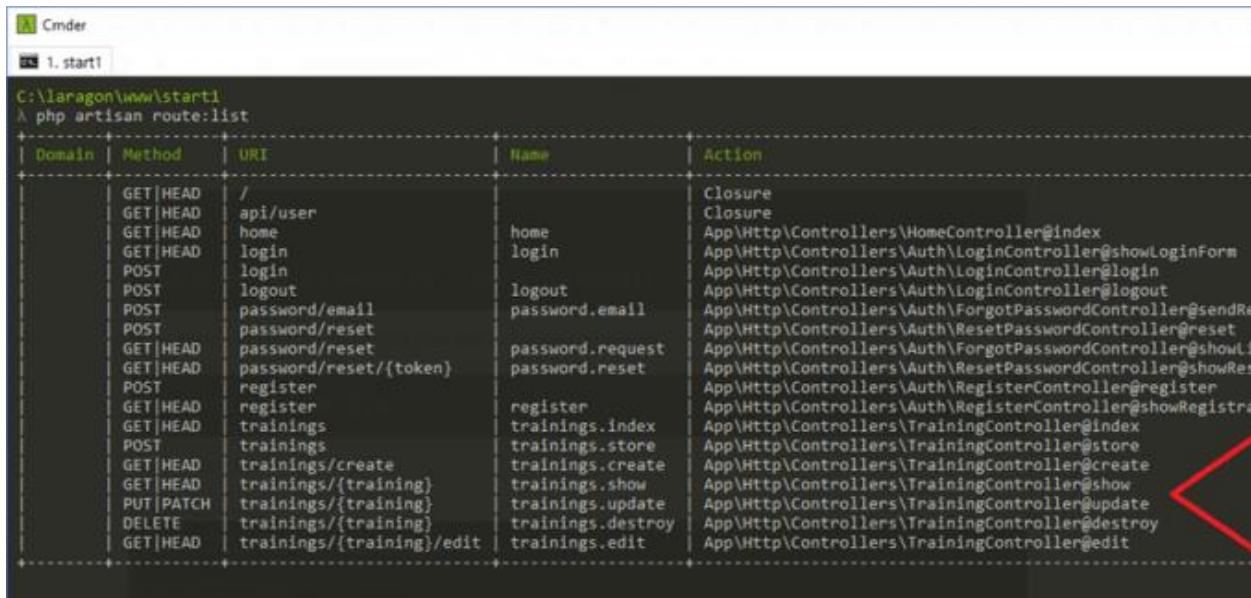
Folder as Workspace

```
3 //*
4 |
5 | Web Routes
6 |
7 |
8 | Here is where you can register web routes for your application. These
9 | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18 Auth::routes();
19
20
21
22
23
24
```

web.php

Untuk menyenaraikan path kepada fungsi yang disediakan, laksanakan arahan route:list.

php artisan route:list



Domain	Method	URI	Name	Action
	GET HEAD	/		Closure
	GET HEAD	api/user		Closure
	GET HEAD	home	home	App\Http\Controllers\HomeController@index
	GET HEAD	login	login	App\Http\Controllers\Auth\LoginController@showLoginForm
	POST	login		App\Http\Controllers\Auth\LoginController@login
	POST	logout	logout	App\Http\Controllers\Auth\LoginController@logout
	POST	password/email		App\Http\Controllers\Auth\ForgotPasswordController@sendResetLink
	POST	password/reset		App\Http\Controllers\Auth\ResetPasswordController@reset
	GET HEAD	password/reset	password.request	App\Http\Controllers\Auth\ForgotPasswordController@showResetLink
	GET HEAD	password/reset/{token}	password.reset	App\Http\Controllers\Auth\ResetPasswordController@showResetLink
	POST	register	register	App\Http\Controllers\Auth\RegisterController@register
	GET HEAD	register		App\Http\Controllers\Auth\RegisterController@showRegistrationForm
	GET HEAD	trainings	trainings.index	App\Http\Controllers\TrainingController@index
	POST	trainings	trainings.store	App\Http\Controllers\TrainingController@store
	GET HEAD	trainings/create	trainings.create	App\Http\Controllers\TrainingController@create
	GET HEAD	trainings/{training}	trainings.show	App\Http\Controllers\TrainingController@show
	PUT PATCH	trainings/{training}	trainings.update	App\Http\Controllers\TrainingController@update
	DELETE	trainings/{training}	trainings.destroy	App\Http\Controllers\TrainingController@destroy
	GET HEAD	trainings/{training}/edit	trainings.edit	App\Http\Controllers\TrainingController@edit

Perhatikan path yang disediakan disertakan ‘s’ dihujung (training s). Path yang automatik disertakan adalah;

trainings.index (fungsi senaraikan rekod),

trainings.store (fungsi simpan rekod),

trainings.create (fungsi simpan rekod baharu),

trainings.update (fungsi perbaharui rekod sedia ada),

trainings.destroy (fungsi memadam rekod),

trainings.edit (fungsi borang kemaskini rekod).

Seterusnya, akan ditunjukkan bagaimana nak paparkan borang kemasukan rekod baharu yang telah disediakan kod dalam fail *create.blade.php*.

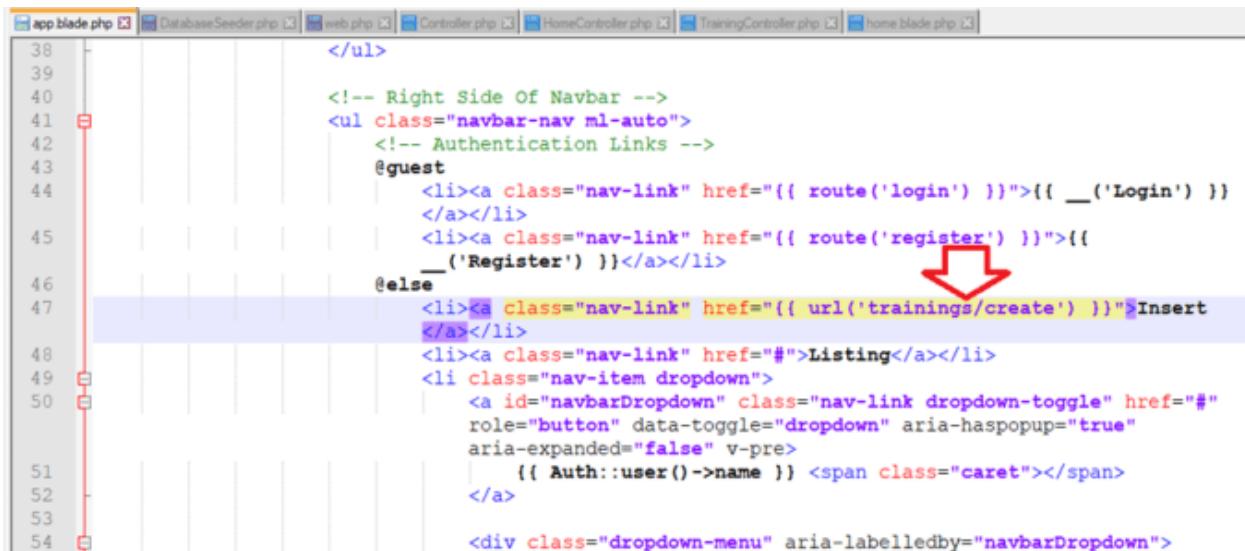
Buka fail TrainingController.php dalam folder *app/Http/Controllers/* dan tambahkan baris kod berikut ke dalam *function create()*.

```

23     \Illuminate\Http\Response
24
25     //TrainingController.php
26     public function create()
27     {
28         //display form
29         return view(
30             'trainings.create');
31     }

```

Untuk menyediakan butang/link kepada borang masuk rekod baharu, tambah kod berikut ke fail template, *resources/views/layouts/app.blade.php* .



```

38
39
40
41     <!-- Right Side Of Navbar -->
42     <ul class="navbar-nav ml-auto">
43         <!-- Authentication Links -->
44         @guest
45             <li><a class="nav-link" href="{{ route('login') }}>{{ __('Login') }}</a></li>
46             <li><a class="nav-link" href="{{ route('register') }}>{{ __('Register') }}</a></li>
47         @else
48             <li><a class="nav-link" href="{{ url('trainings/create') }}>Insert</a></li>
49             <li><a class="nav-link" href="#">Listing</a></li>
50             <li class="nav-item dropdown">
51                 <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>
52                     {{ Auth::user() -> name }} <span class="caret"></span>
53                 </a>
54                 <div class="dropdown-menu" aria-labelledby="navbarDropdown">

```

Dalam form action pula, hantar rekod baharu ke link berikut untuk proses penyimpanan data ke pangkalan data (db).



```
1 @extends('layouts.app')
2 @section('content')
3 <div class="container">
4   <div class="row justify-content-center">
5     <div class="col-md-8">
6       <div class="card">
7         <div class="card-header">Insert new Training</div>
8
9         <div class="card-body">
10            <form method="post" action="{{url('trainings')}}">
11              @csrf
```

Seterusnya dalam fail *app/Training.php* , isikan senarai lajur jadual yang terlibat. Rujuk rajah di bawah dan tambah kod berikut;



```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Training extends Model
8 {
9   //Training.php
10  //select field involved
11  protected $fillable = ['trainingname', 'desc', 'trainer'];
12
13 }
```

Sekiranya kita rujuk kepada *route:list , path ‘trainings’* akan menuju kepada fungsi *store* (simpan) sekiranya data dalam bentuk *post* .

Jadi seterusnya, tambah kod berikut dalam function *store()*, fail *app/Http/Controllers/TrainingController.php*. Kod di bawah merupakan proses simpan rekod dalam pangkalan data.

Masukkan baris berikut di bahagian atas *TrainingController.php*;

```
use App\Training; //include the namespace of Training.php
```

Baris 51: merupakan arahan *redirect* yang akan memaparkan semula borang, dengan membawa mesej rekod berjaya disimpan.

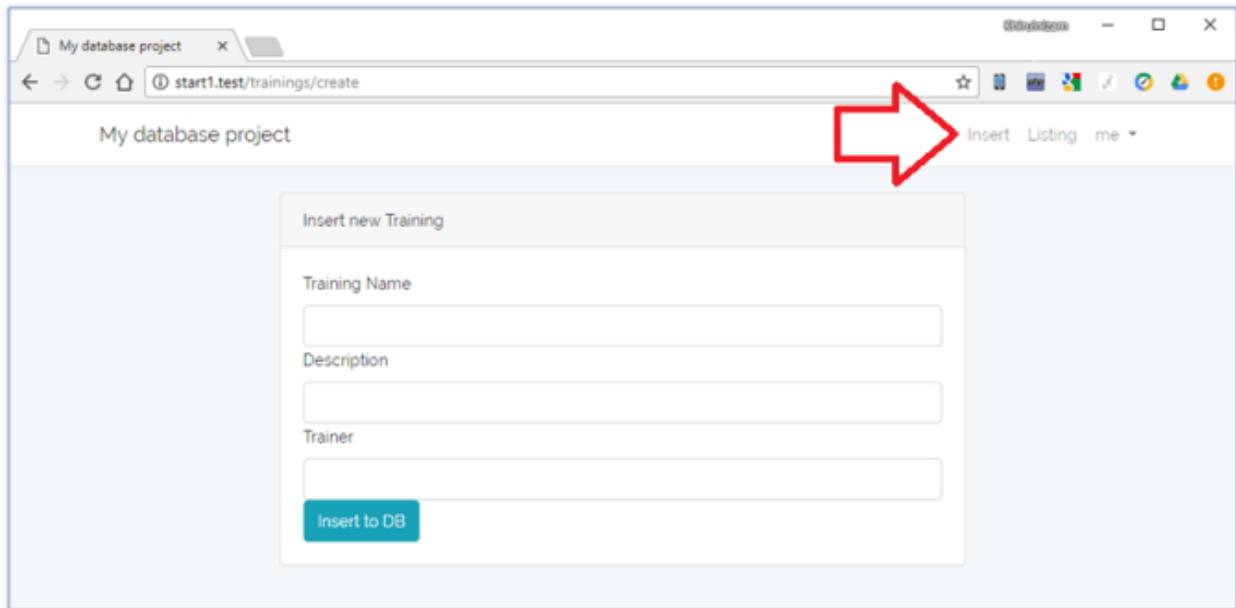
```
TrainingController.php Training.php app.blade.php DatabaseSeeder.php web.php Controller.php HomeController.php
37     */
38
39     //// TrainingController.php
40     public function store(Request $request)
41     {
42         //insert new record process is here
43         $training = $this->validate($request(), [
44             'trainingname' => 'required',
45             'desc' => 'required',
46             'trainer' => 'required'
47         ]);
48
49         Training::create($training);
50
51         return back()->with('success', 'Training has been added');
52     }
53 }
```

Seterusnya dalam fail *app/Training.php*, isikan senarai lajur jadual yang terlibat. Rujuk rajah di bawah dan tambah kod berikut;

```
Training.php app.blade.php DatabaseSeeder.php web.php Controller.php HomeController.php TrainingController.php
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Training extends Model
8 {
9     //Training.php
10    //select field involved
11    protected $fillable = ['trainingname', 'desc', 'trainer'];
12 }
13 }
```



Alhamdulillah, selesai fungsi masukkan rekod baru. Bagaimana nak uji? Klik pada link Insert ditunjukkan anak panah.



My database project

start1.test/trainings/create

Insert Listing me

Insert new Training

Training Name

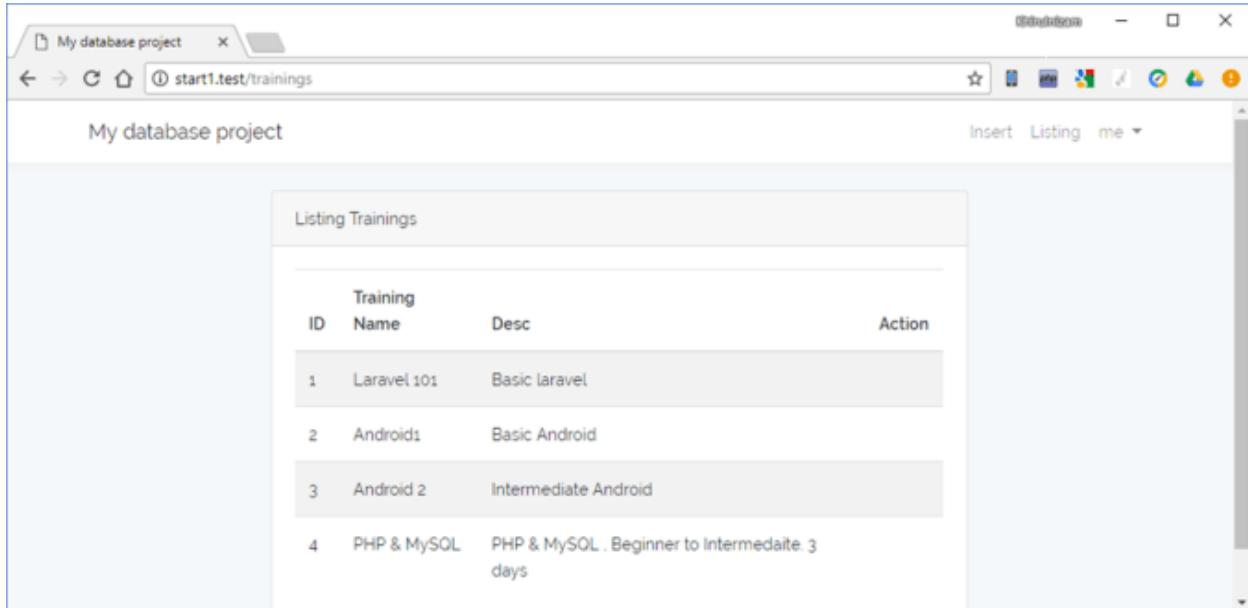
Description

Trainer

Insert to DB

SENARAIKAN REKOD

Sambungan latihan, menyenaraikan semua rekod dalam jadual. Kita akan menghasilkan antaramuka seperti di bawah.



My database project

start1.test/trainings

Insert Listing me

Listing Trainings

ID	Name	Desc	Action
1	Laravel 101	Basic laravel	
2	Android1	Basic Android	
3	Android 2	Intermediate Android	
4	PHP & MySQL	PHP & MySQL . Beginner to Intermediaite . 3 days	

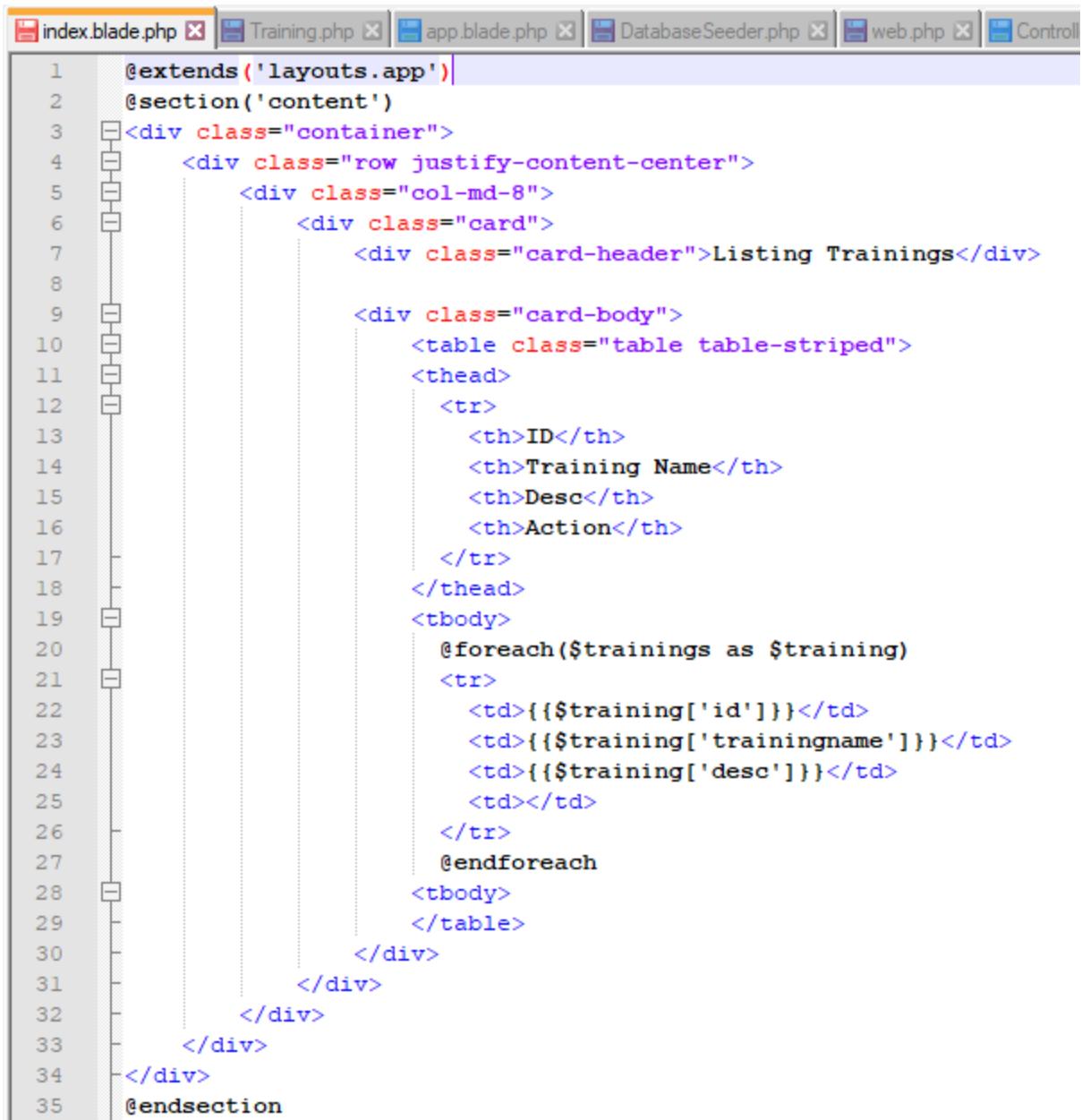
Berdasarkan `route:list`, kita boleh buat penyenaraian dalam function `index()`, fail `app/Http/Controllers/TrainingController.php`

Tulis kod berikut dalam function `index()`:



```
13  * /  
14  public function index()  
15  {  
16      // app/Http/Controllers  
17      // TrainingController.php  
18  
19      $trainings= Training::all()->toArray();  
20      return view('trainings.index', compact('trainings'));  
21  }  
22  
23  /**
```

Antaramuka yang melibatkan pembentukan jadual dinamik beserta senarai rekod boleh dibina dalam fail *resources/views/trainings/index.blade.php* . Sekiranya fail blade ini belum wujud, sila *create new file* .



```
1 @extends('layouts.app')
2 @section('content')
3 <div class="container">
4     <div class="row justify-content-center">
5         <div class="col-md-8">
6             <div class="card">
7                 <div class="card-header">Listing Trainings</div>
8
9                 <div class="card-body">
10                    <table class="table table-striped">
11                        <thead>
12                            <tr>
13                                <th>ID</th>
14                                <th>Training Name</th>
15                                <th>Desc</th>
16                                <th>Action</th>
17                            </tr>
18                        </thead>
19                        <tbody>
20                            @foreach($trainings as $training)
21                                <tr>
22                                    <td>{{$training['id']}}</td>
23                                    <td>{{$training['trainingname']}}</td>
24                                    <td>{{$training['desc']}}</td>
25                                    <td></td>
26                                </tr>
27                            @endforeach
28                        </tbody>
29                    </table>
30                </div>
31            </div>
32        </div>
33    </div>
34 @endsection
```

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

<div class = "container" >

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

<div class = "col-md-8" >
```

```
< div class = "card" >

< div class = "card-header" >Listing Trainings</ div >

< div class = "card-body" >

< table class = "table table-striped" >

< thead >

< tr >

< th >ID</ th >

< th >Training Name</ th >

< th >Desc</ th >

< th >Action</ th >

</ tr >

</ thead >

< tbody >
@foreach($trainings as $training)

< tr >

< td >{{ $training['id'] }}</ td >

< td >{{ $training['trainingname'] }}</ td >

< td >{{ $training['desc'] }}</ td >
```

```
< td ></ td >
```

```
</ tr >
```

```
@endforeach
```

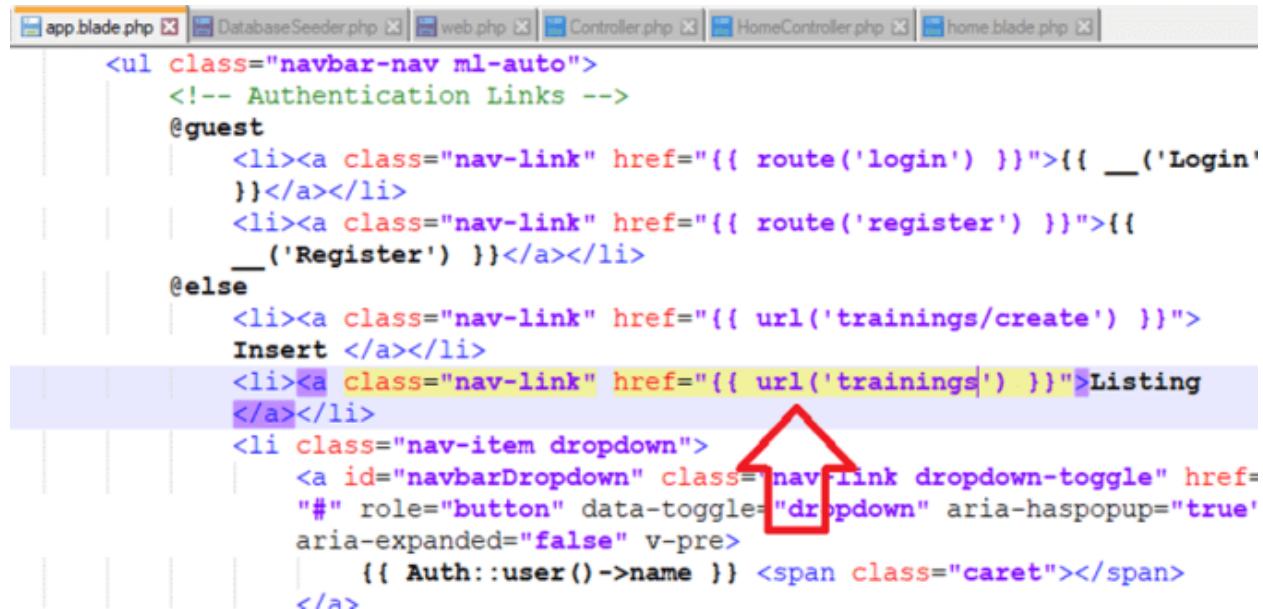
```
< tbody >
```

```
</ table >
```

```
</ div >
```

```
@endsection
```

Tambahkan link kepada antaramuka ini, buka fail
resources/views/layouts/app.blade.php .



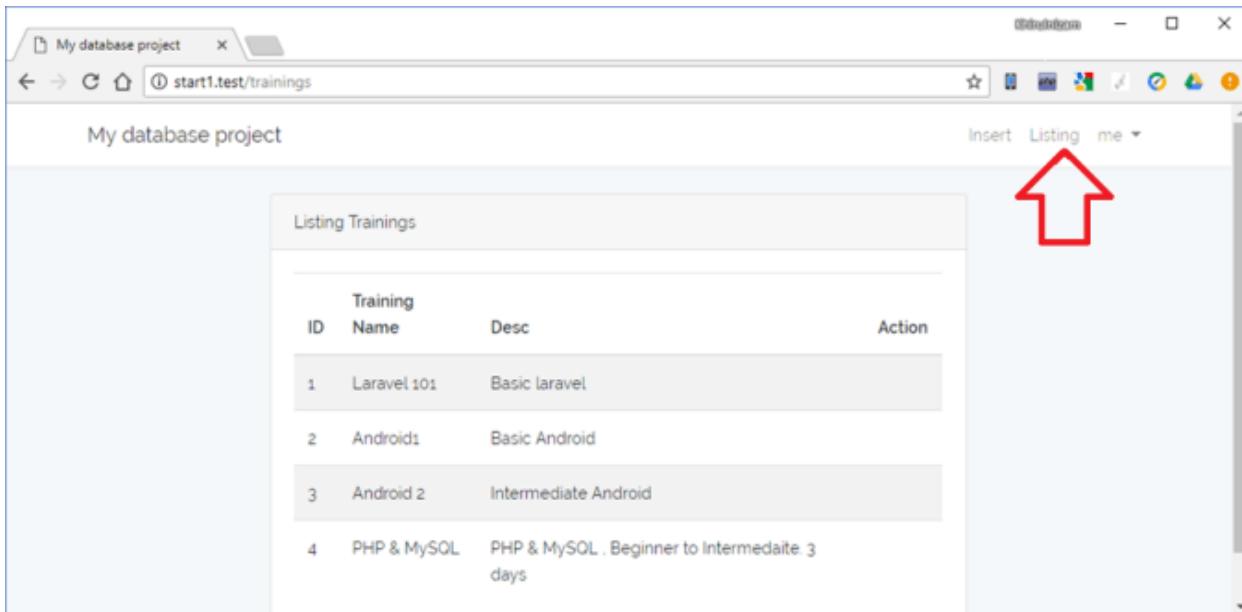
```
app.blade.php DatabaseSeeder.php web.php Controller.php HomeController.php home.blade.php



    @guest
        <li><a class="nav-link" href="{{ route('login') }}>{{ __('Login') }}</a></li>
        <li><a class="nav-link" href="{{ route('register') }}>{{ __('Register') }}</a></li>
    @else
        <li><a class="nav-link" href="{{ url('trainings/create') }}>Insert </a></li>
        <li><a class="nav-link" href="{{ url('trainings') }}>Listing </a></li>
        <li class="nav-item dropdown">
            <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>
                {{ Auth::user()->name }} <span class="caret"></span>
            </a>
        </li>
    @endif

```

Cuba fungsi senarai semua rekod di sini;



The screenshot shows a web application titled "My database project" with a URL of "start1.test/trainings". The main content is a table titled "Listing Trainings" with the following data:

ID	Name	Desc	Action
1	Laravel 101	Basic laravel	
2	Android1	Basic Android	
3	Android 2	Intermediate Android	
4	PHP & MySQL	PHP & MySQL . Beginner to Intermediaite. 3 days	

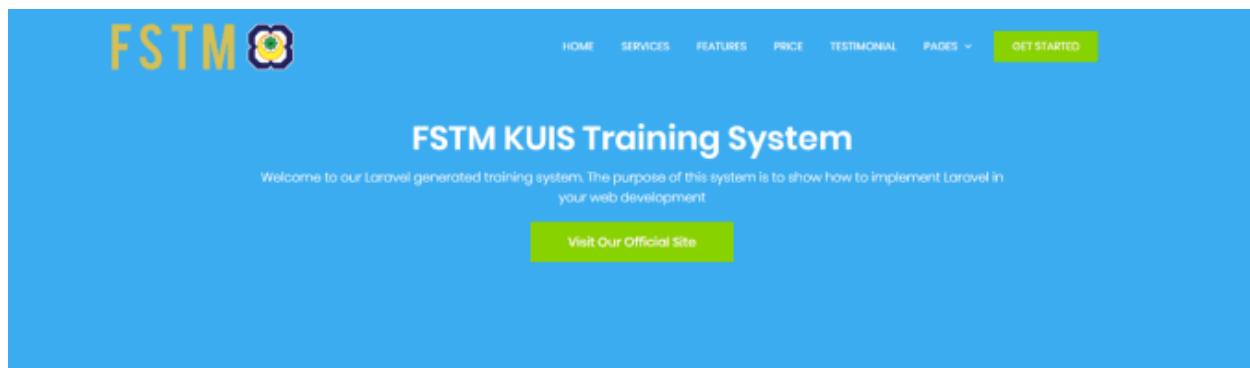
A red arrow points to the "Insert" button in the top right corner of the page.

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>

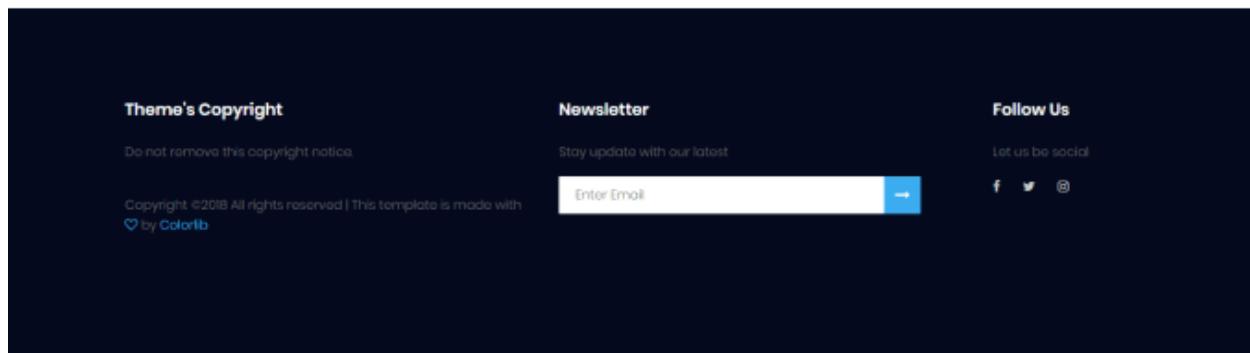
Tutorial 03 Ubah design Antaramuka



Berikut merupakan design Bootstrap yang boleh digunakan percuma. Bagaimana nak masukkan design ini dalam antaramuka sistem Laravel seperti rajah di bawah?



The system's workplace

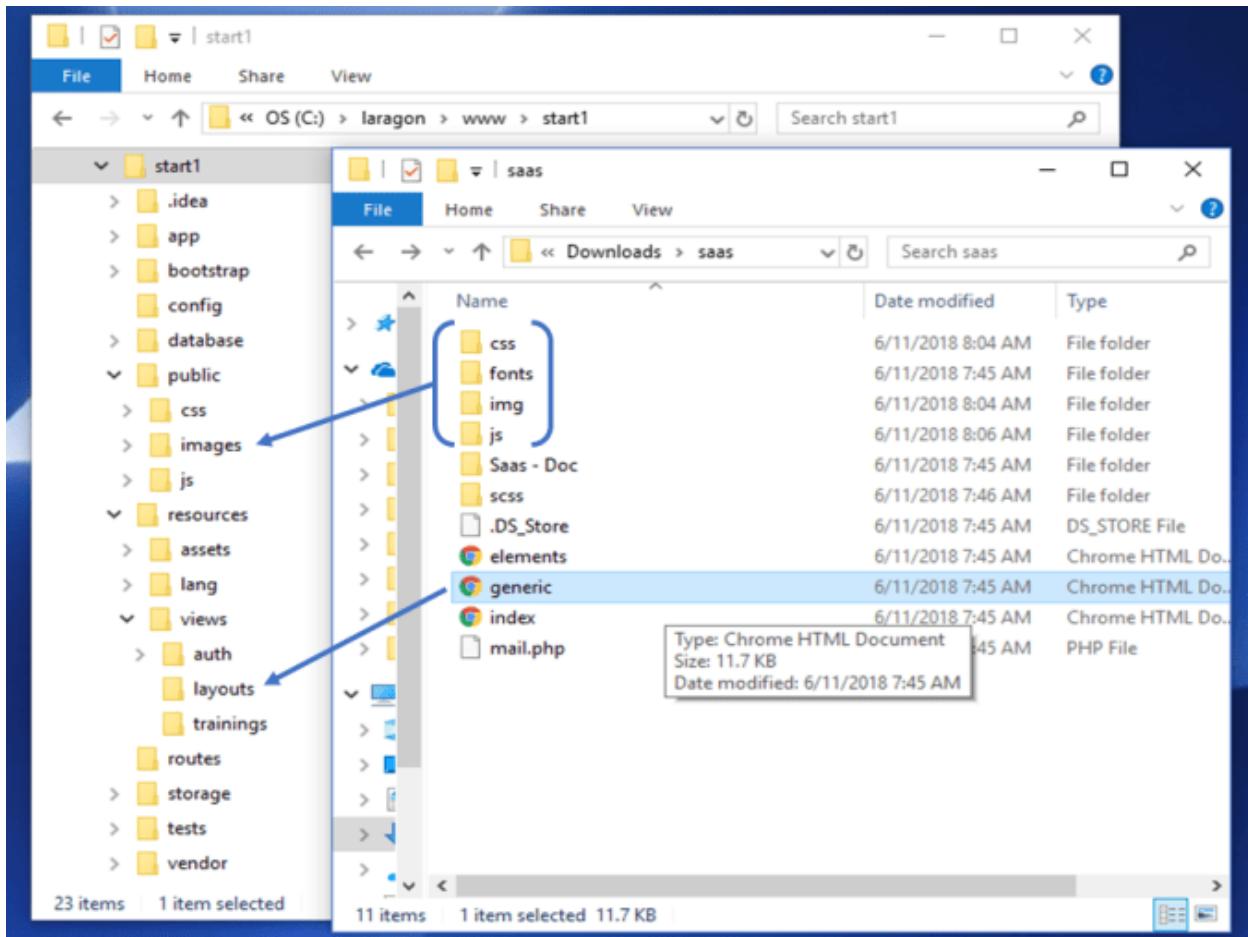


Muat-turun tema/template Bootstrap

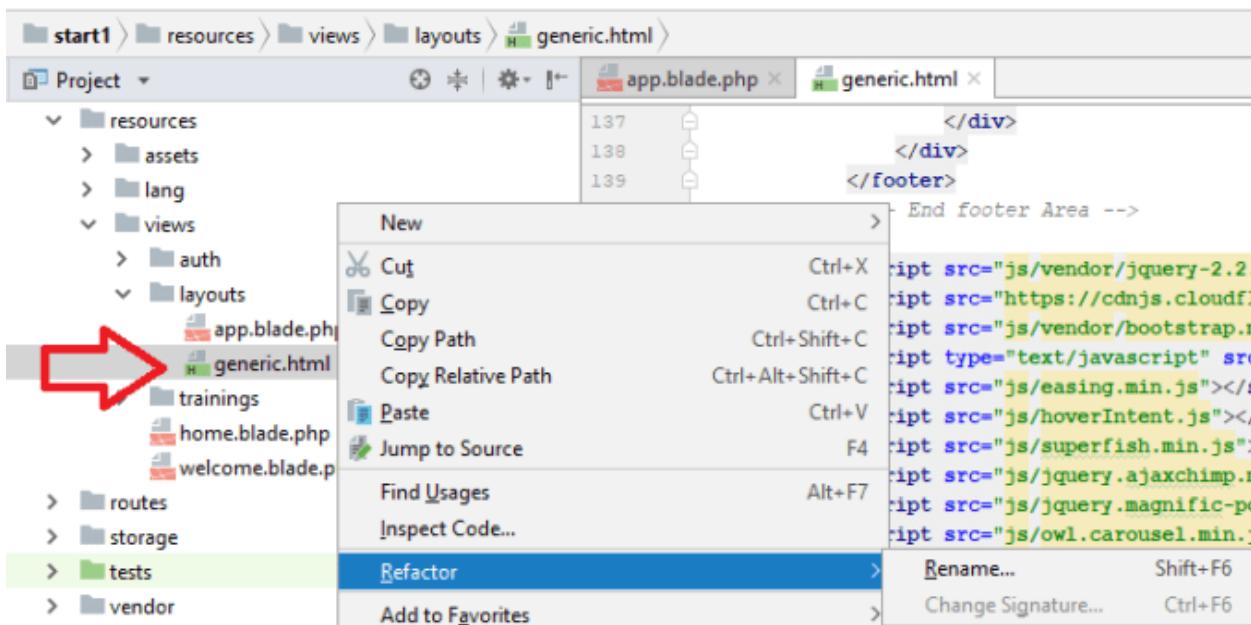
Buat carian untuk download template percuma/berbayar. Dalam tutorial ini, saya menggunakan template dari ColorLib,
<https://colorlib.com/wp/template/saas/>

Selepas berjaya muat-turun fail template ColorLib Saas, ekstrak fail zip. Ini kandungan fail template.

Pindahkan senarai fail seperti dalam arahan rajah.



Tukar nama fail HMTL, contoh generic.html kepada generic.blade.php . Ini kerana untuk mengoptimasi kebolehan Blade dalam Laravel, fail HTML mesti dinamakan *namafail.blade.html* . (Menggunakan *PHPStorm*) Klik-kanan fail HTML, *refactor* dan *rename* .



Penulis telah menukar template kepada layout yang ringkas. Boleh perolehi kod dalam pastebin.com/tgaFCjCm buat masa ini. Akan datang ada di GitHub.

Berikut beberapa penerangan tentang kod Blade yang biasa akan dijumpai. Skrip Blade biasanya disempadani oleh simbol berikut;

`{{ ... }}`

Seperti contoh di bawah, `{{ config('app.name') }}' Laravel'` merujuk kepada arahan Blade untuk paparkan nama default applikasi web projek ini. Mana nak cari setting nama default applikasi, rujuk dalam fail `.env`.

```
<!-- Meta Description -->
<meta name="description" content="">
<!-- Meta Keyword -->
<meta name="keywords" content="">
<!-- meta character set -->
<meta charset="utf-8" />
<!-- Site Title -->
<title>{{ config('app.name', 'Laravel') }} #FSTMUI</title>
```

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>

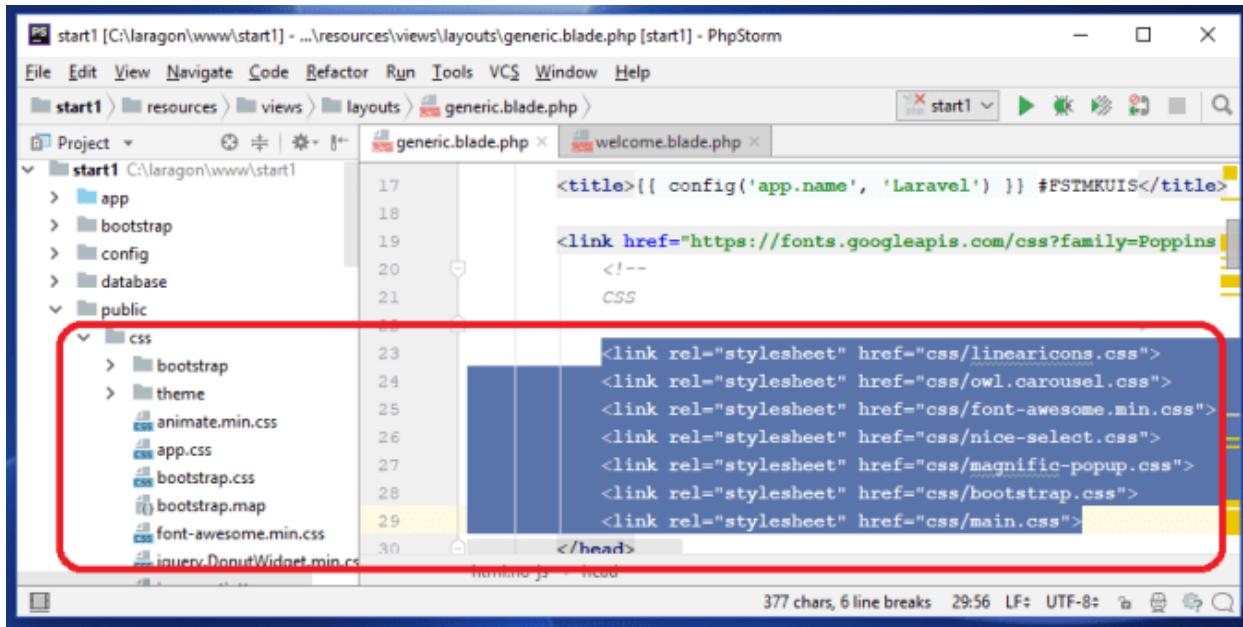
Memanggil fail kandungan page

generic.blade.php merupakan satu fail template HTML, maksudnya ia adalah fail yang akan digunakan sebagai template semua laman web dalam sistem anda. Yang akan menentukan kandungan berubah anda dalam arahan yield('content') seperti di bawah.

```
<section class="about-generic-area section-gap">
    <div class="container border-top-generic">
        <h3 class="about-title mb-30">The system's workplace</h3>
        <div class="row">
            <!-- put your content here -->
            @yield('content')
        </div>
    </div>
</section>
<!-- End Generic Start -->
```

Folder CSS, imej & JavaScript

Fail kerangka Bootstrap CSS ada dalam folder **public/css** seperti yang telah kita pindahkan peringkat awal tadi.

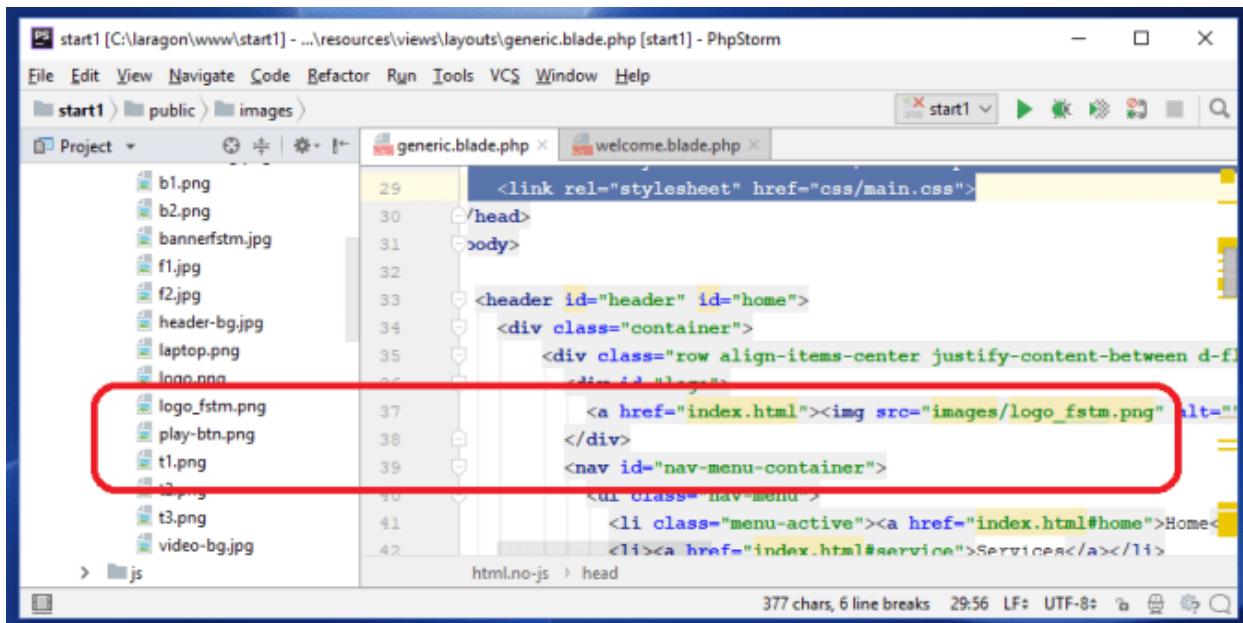


The screenshot shows the PhpStorm IDE interface with the project 'start1' open. The left sidebar displays the project structure with a red box highlighting the 'css' folder under the 'public' directory. The main editor window shows the 'generic.blade.php' file, specifically the head section where CSS links are listed:

```
<head>
    <title>{{ config('app.name', 'Laravel') }} #FSTMKUIS</title>
    <link href="https://fonts.googleapis.com/css?family=Poppins" rel="stylesheet">
    <!--
        CSS
    -->
    <link rel="stylesheet" href="css/linearicons.css">
    <link rel="stylesheet" href="css/owl.carousel.css">
    <link rel="stylesheet" href="css/font-awesome.min.css">
    <link rel="stylesheet" href="css/nice-select.css">
    <link rel="stylesheet" href="css/magnific-popup.css">
    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/main.css">
</head>
```

Below the code editor, the status bar indicates 377 chars, 6 line breaks, 29:56, LF:, UTF-8, and other standard icons.

Fail imej kita kumpulkan dalam folder **public/images** seperti yang telah kita pindahkan peringkat awal tadi.

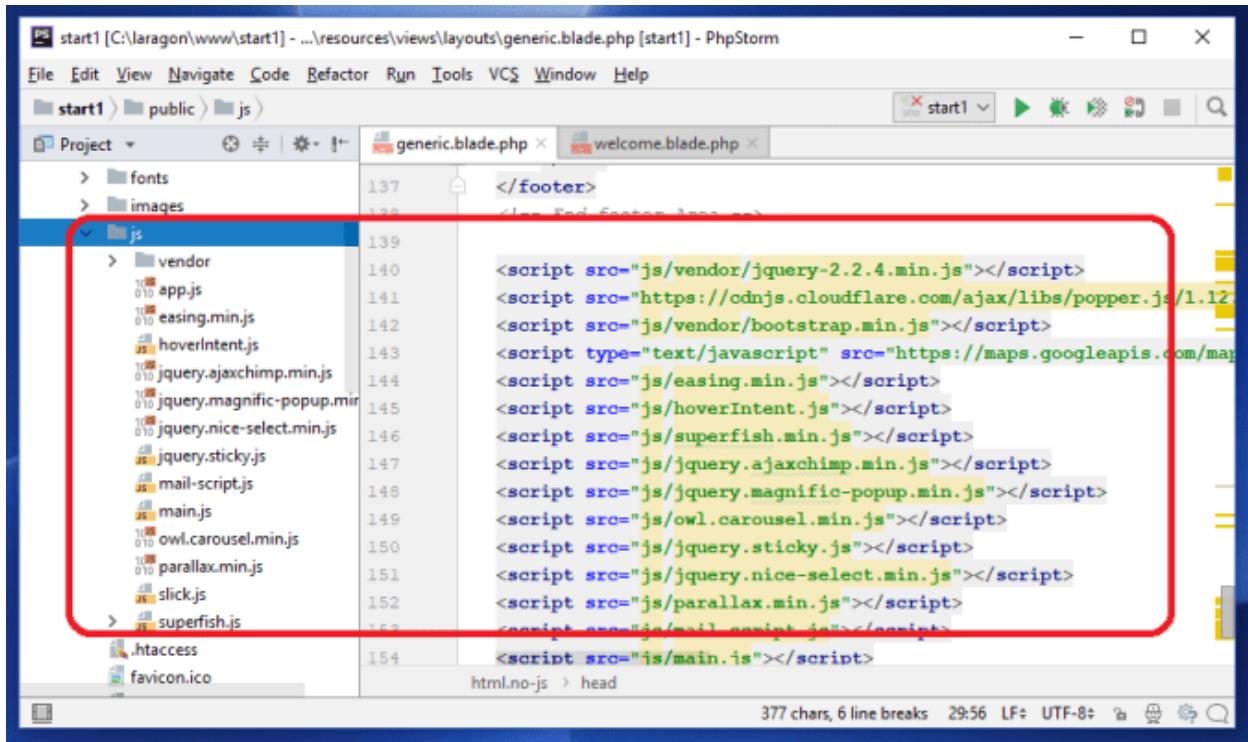


The screenshot shows the PhpStorm IDE interface with the project 'start1' open. The left sidebar displays the project structure with a red box highlighting the 'images' folder under the 'public' directory. The main editor window shows the 'generic.blade.php' file, specifically the body section where image URLs are used:

```
<body>
    <link rel="stylesheet" href="css/main.css">
    </head>
    <body>
        <header id="header" id="home">
            <div class="container">
                <div class="row align-items-center justify-content-between d-f"
                    <div id="logo">
                        <a href="index.html"></a>
                    </div>
                    <nav id="nav-menu-container">
                        <ul class="nav-menu">
                            <li class="menu-active"><a href="index.html#home">Home</a></li>
                            <li><a href="index.html#service">Services</a></li>
                            <li><a href="index.html#about">About</a></li>
                            <li><a href="index.html#contact">Contact</a></li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div class="container">
            <div class="row align-items-center justify-content-between d-f">
                <div id="play-btn">
                    
                </div>
                <div id="t1">
                    
                </div>
                <div id="t2">
                    
                </div>
                <div id="t3">
                    
                </div>
                <div id="video-bg">
                    
                </div>
            </div>
        </div>
    </body>
```

Below the code editor, the status bar indicates 377 chars, 6 line breaks, 29:56, LF:, UTF-8, and other standard icons.

Fail kerangka JavaScript ada dalam folder **public/js** seperti yang telah kita pindahkan peringkat awal tadi.



The screenshot shows the PhpStorm IDE interface with the project 'start1' open. The left sidebar displays the project structure with a red box highlighting the 'js' folder under 'public'. This folder contains several subfolders and files, including 'vendor' which holds scripts like 'app.js', 'easing.min.js', 'hoverIntent.js', 'jquery.ajaxchimp.min.js', 'jquery.magnific-popup.min.js', 'jquery.nice-select.min.js', 'jquery.sticky.js', 'mail-script.js', 'main.js', 'owl.carousel.min.js', 'parallax.min.js', 'slick.js', and 'superfish.js'. The main editor area shows the 'generic.blade.php' file with Blade template syntax. Lines 137 to 154 contain script tags for loading various JavaScript files from the 'js' folder and other sources like CDNs.

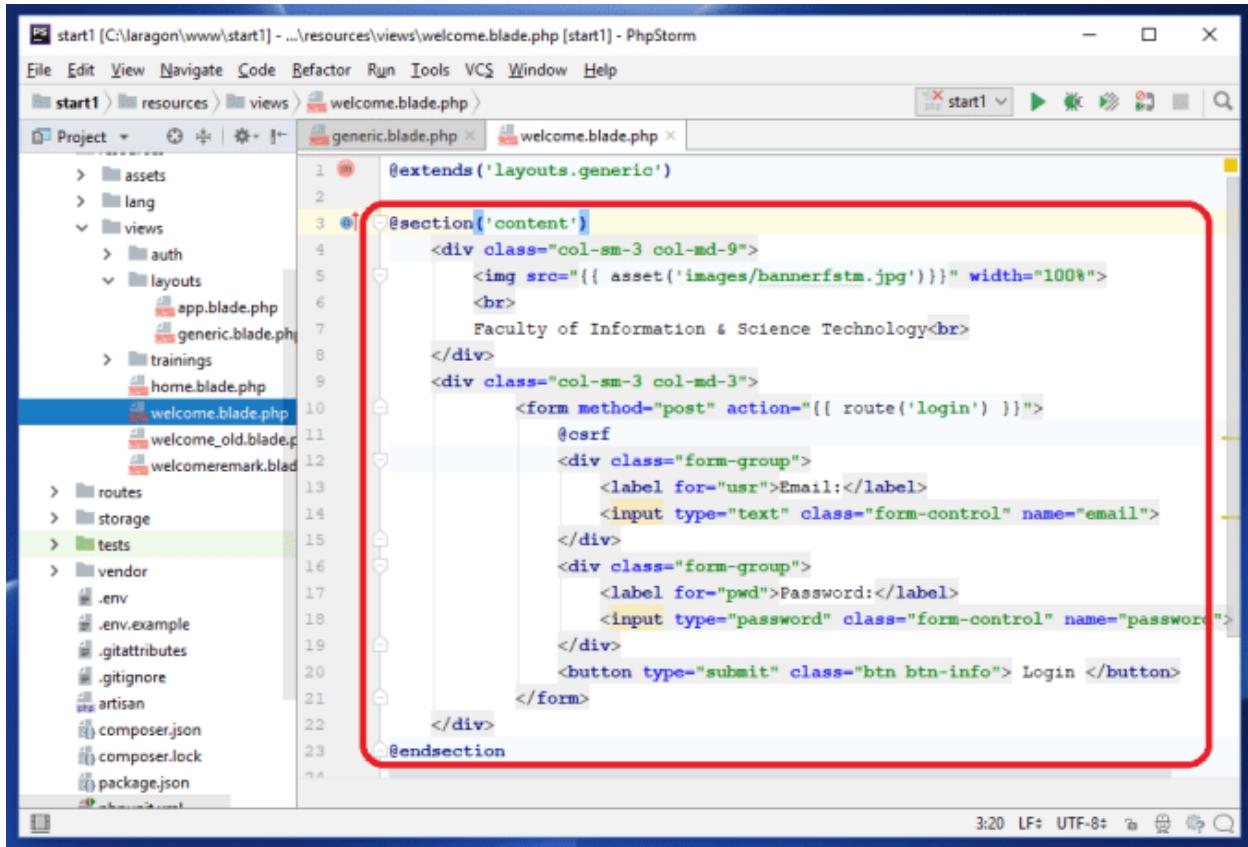
```
</footer>
<!-- End Footer Area -->
<script src="js/vendor/jquery-2.2.4.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
<script src="js/vendor/bootstrap.min.js"></script>
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?language=en"></script>
<script src="js/easing.min.js"></script>
<script src="js/hoverIntent.js"></script>
<script src="js/superfish.min.js"></script>
<script src="js/jquery.ajaxchimp.min.js"></script>
<script src="js/jquery.magnific-popup.min.js"></script>
<script src="js/owl.carousel.min.js"></script>
<script src="js/jquery.sticky.js"></script>
<script src="js/jquery.nice-select.min.js"></script>
<script src="js/parallax.min.js"></script>
<script src="js/main.js"></script>
<script src="js/html.no-js"></script>
```

Laman pertama sistem

Laman pertama yang akan dipanggil dalam sistem anda berada dalam fail **welcome.blade.php**. Jadi dalam fail ini akan diletakkan kandungan untuk page *welcome* .

Kod boleh diperolehi dari pastebin.com/W96GnW7D

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>



```
start1 [C:\laragon\www\start1] - ...resources\views\welcome.blade.php [start1] - PhpStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
start1 resources views welcome.blade.php
Project generic.blade.php welcome.blade.php
1 @extends('layouts.generic')
2
3 @section('content')
4     <div class="col-sm-3 col-md-9">
5         
6         <br>
7         Faculty of Information & Science Technology<br>
8     </div>
9     <div class="col-sm-3 col-md-3">
10        <form method="post" action="{{ route('login') }}>
11            @csrf
12            <div class="form-group">
13                <label for="usr">Email:</label>
14                <input type="text" class="form-control" name="email">
15            </div>
16            <div class="form-group">
17                <label for="pwd">Password:</label>
18                <input type="password" class="form-control" name="password">
19            </div>
20            <button type="submit" class="btn btn-info"> Login </button>
21        </form>
22    </div>
23 @endsection
```

@extends('layouts.generic') merujuk kepada fail antaramuka HTML ini akan menggunakan resources/views/layout/generic.blade.html sebagai fail layout template HTML.

Seterusnya @section('content') merujuk kepada persempadanan kandungan yang akan dipaparkan oleh @yield('content') yang terdapat dalam fail layout template *generic.blade.php*.

```
@section('content')
...
...
...
@endsection
```

Sila uji dalam browser antaramuka yang telah diubah.

The screenshot shows a web browser window with the title bar "My database project #FSTM". The address bar displays "Not secure | start1.test". The main content area features a blue header with the "FSTM" logo and navigation links for HOME, SERVICES, FEATURES, PRICE, TESTIMONIAL, PAGES, and GET STARTED. Below the header is a large yellow section containing the heading "FSTM KUIS Training System" and a welcome message: "Welcome to our Laravel generated training system. The purpose of this system is to show how to implement Laravel in your web development". A green button labeled "Visit Our Official Site" is visible. The bottom portion of the page shows a login form with fields for Email and Password, and a "Login" button. The background of the login form features a banner for "INTAKE JUN/NOV KUIS" with a graduation theme.

The system's workplace

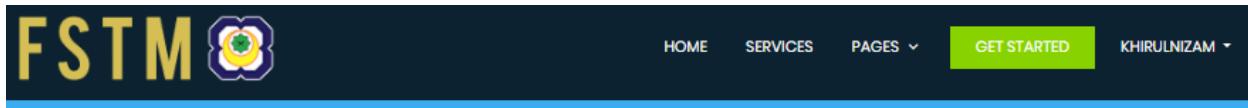
Email: kerul@gmail.com

Password:

Login

Kod Sumber dalam GITHUB – <https://github.com/khirulnizam/start1>

Tutorial 04 Carian Rekod



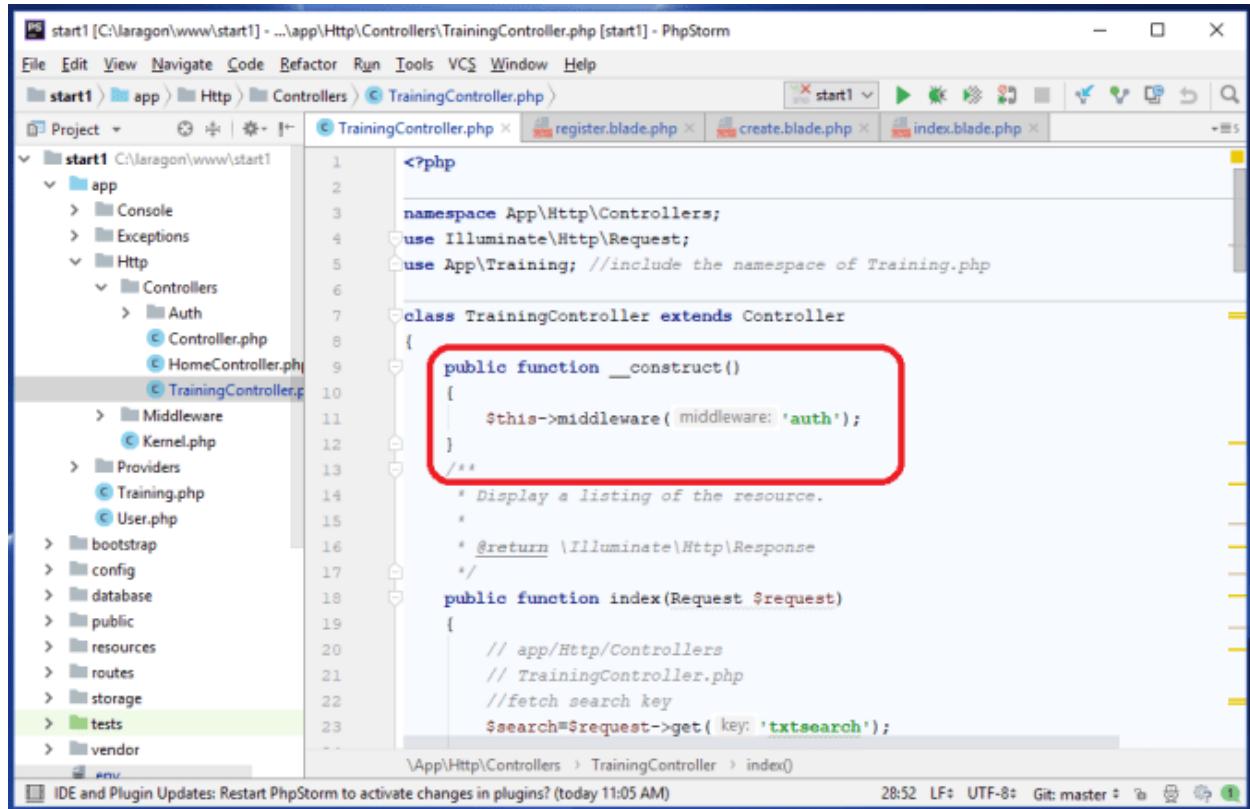
Gambar di atas menujukkan hasil tutorial yang kita akan buat menggunakan Laravel. Tutorial ini bersambung daripada modul Training dan *TrainingController* yang telah dibincangkan dalam <http://fstm.kuis.edu.my/blog/laravel2/> .

Kita akan sambung mengedit kod antaramuka dalam *resources/views/trainings/ index.blade.php* untuk menambahkan kotak dan butang carian. Perhatikan pada kod FORM action , url('training') menunjuk kepada fungsi dalam *TrainingController* yang akan menerima arahan dan data carian tersebut.

```
<form method="get" action="{{ url('trainings') }}" class="form-inline">
    @csrf
    <input type="text" id="txtsearch" name="txtsearch" class="form-control">
    <button type="submit" class="btn btn-primary">
        <i class="fa fa-search"></i>
    </button>
</form>
```

app/Http/Controllers/TrainingsController.php

Kemudian edit kod logik proses dalam app/Http/Controllers/TrainingsController.php, tambahkan kod pengesahan pengguna di bahagian atas class TrainingsController. Ini sepatutnya dilakukan pada peringkat awal lagi. Kod di bawah merupakan pengesahan pengguna, sekiranya pengguna belum login maka antaramuka ini akan terhalang. Sekiranya anda telah menulis kod berikut, maka boleh abaikan langkah ini.



```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Training; //include the namespace of Training.php

class TrainingController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // app\Http\Controllers
        // TrainingController.php
        //fetch search key
        $search=$request->get('key: 'txtsearch');
    }
}
```

Seterusnya pergi ke function index(), dan tambahkan kod berikut untuk membuat carian.

txtsearch merupakan nama/id kotak teks yang kita telah buat di atas (dalam bahagian *form*).

Segmen kod berikut akan melakukan penyenaraian biasa sahaja.

Manakala segment kod berikut akan menerima perkataan carian, dan memaparkan hasil carian yang berpadanan.

The screenshot shows the PhpStorm IDE interface with the following details:

- Title Bar:** start1 [C:\laragon\www\start1] - ...app\Http\Controllers\TrainingController.php [start1] - PhpStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** start1, register.blade.php, create.blade.php, index.blade.php, home.blade.php
- Project Tree:** start1 (C:\laragon\www\start1) expanded to show app, Http, and Controllers. Controllers contains subfolders like Auth, HomeController.php, and TrainingController.php.
- Code Editor:** The file TrainingController.php is open at line 18. The code is as follows:

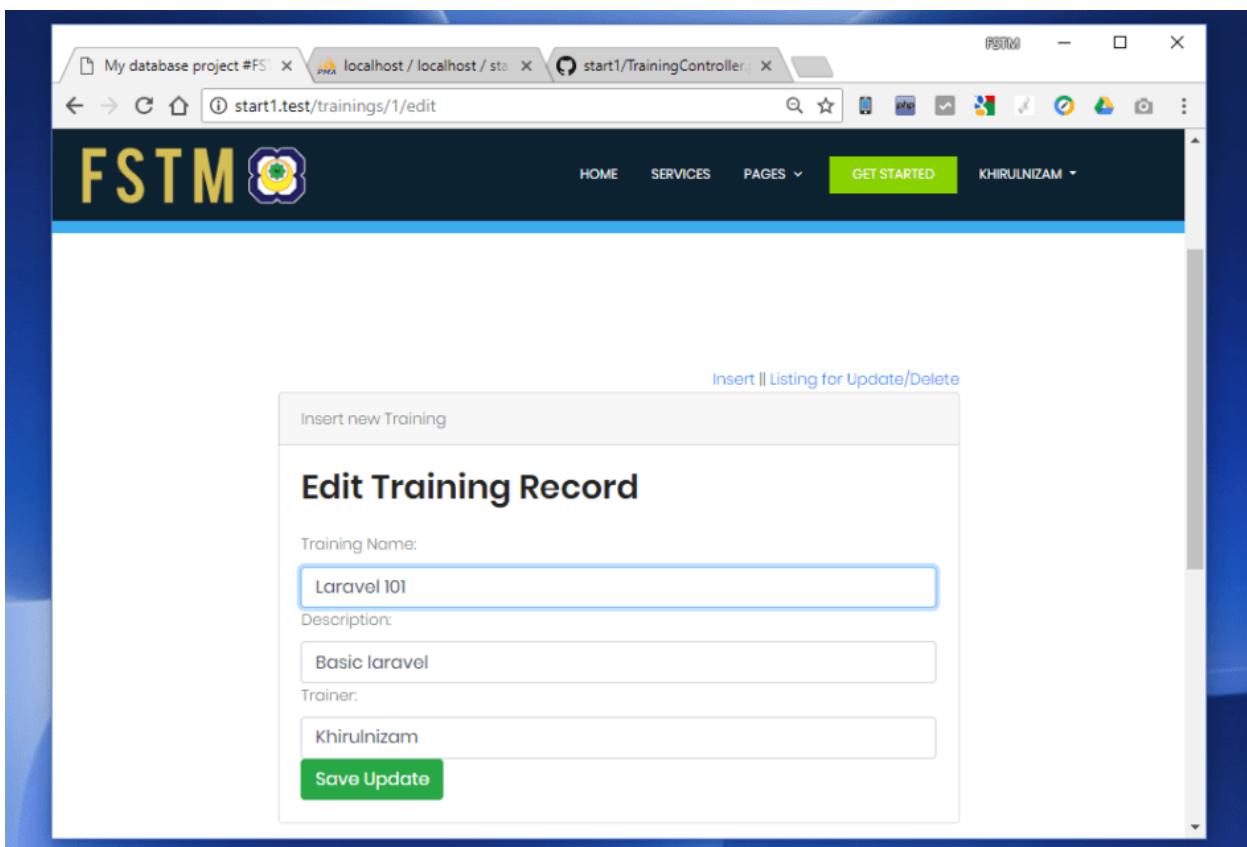
```
public function index(Request $request)
{
    // app\Http\Controllers\TrainingController.php
    // fetch search key
    $search=$request->get('txtsearch');

    //Trainings= new Training();
    if ($search==null){
        //display all record
        $trainings= Training::all()->toArray();
        return view('trainings.index', compact('varname: 'trainings'));
    }
    else{
        //display record based on search key
        $trainings= Training::where('trainingname', 'like', '%'.$search.'%');
        $trainings=$trainings->get();
        return view('trainings.index', compact('varname: 'trainings'));
    }
}
```
- Bottom Status Bar:** IDE and Plugin Updates: Restart PhpStorm to activate changes in plugins? (today 11:05 AM)
- Bottom Right:** 19:6 LF: UTF-8: Git: master :

Butang edit/padam

Untuk menghasilkan butang edit dan pada, kita perlu mengubahsuai kod memaparkan rekod, mohon rujuk tutorial seterusnya dalam Tutorial 05.

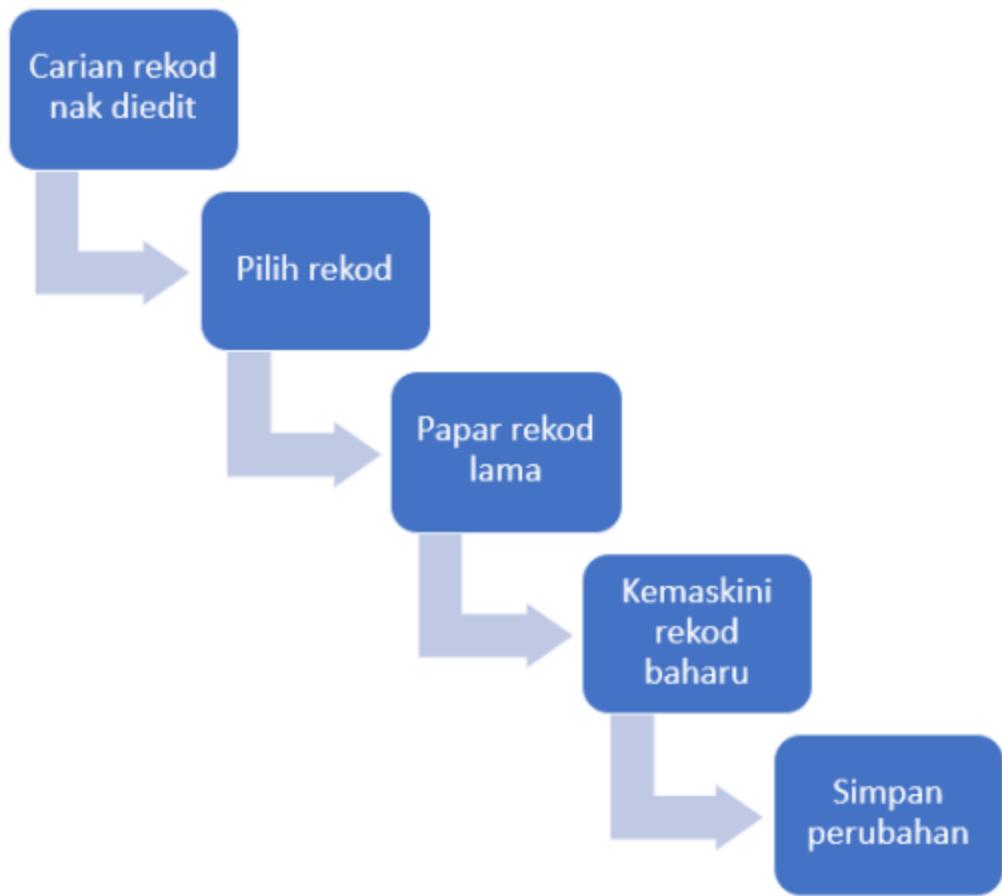
Tutorial 05 Kemaskini dan Padam Rekod



Bersambung dari asas [Laravel tutorial ke 4; Carian rekod dalam Laravel](#). Dalam tutorial sebelum ini, kita terhenti pada penjanaan butang edit dan padam rekod seperti rajah di bawah.

ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel	

Untuk menghasilkan butang edit dan padam, kita perlu mengubahsuai kod memaparkan rekod dalam fail `resources/views/trainings/ index.blade.php`. Secara logiknya, pengguna akan mencari rekod yang hendak dikemaskini, kemudian paparan data lama dalam borang, maklumat lama diedit, dan simpan maklumat baru.



Antaramuka borang papar data lama

Jadi sekarang ini kita hendak sediakan butang untuk pengguna pilih mana rekod yang nak dikemaskini. Kod seperti di bawah boleh diperolehi terus dari GitHub –

<https://github.com/khirulnizam/start1/blob/master/resources/views/trainings/index.blade.php>

Butang kemaskini dan padam rekod berada pada hujung paparan rekod, rekod berada dalam jadual/baris (table row).

The screenshot shows the PhpStorm IDE interface with the file 'index.blade.php' open. The code displays a table structure with a header and a body containing a foreach loop. Annotations with red boxes highlight specific parts of the code:

- A red box surrounds the table header section, labeled "table header".
- A red box surrounds the first column of the table body, which contains an [update button](#).
- A red box surrounds the last column of the table body, which contains a [delete button inside a form](#).

```
<table class="table table-striped">
<thead>
    <tr>
        <th>ID</th>
        <th>Training Name</th>
        <th>Desc</th>
        <th>Action</th>
        <th></th>
    </tr>
</thead>
<tbody>
    @foreach($trainings as $training)
        <tr>
            <td>{{ $training['id'] }}</td>
            <td>{{ $training['trainingname'] }}</td>
            <td>{{ $training['desc'] }}</td>
            <td><a href="{{ action('TrainingController@edit', $training['id']) }}" class="btn btn-warning">
                <i class="fa fa-edit"></i>
            </a></td>
            <td>
                <form action="{{ action('TrainingController@destroy', $training['id']) }}" method="post">
                    @csrf
                    <input name="_method" type="hidden" value="DELETE">
                    <button class="btn btn-danger" type="submit">
                        <i class="fa fa-remove"></i>
                    </button>
                </form>
            </td>
        </tr>
    @endforeach
</tbody>
```

Kod tadi akan menghasilkan paparan seperti di bawah.

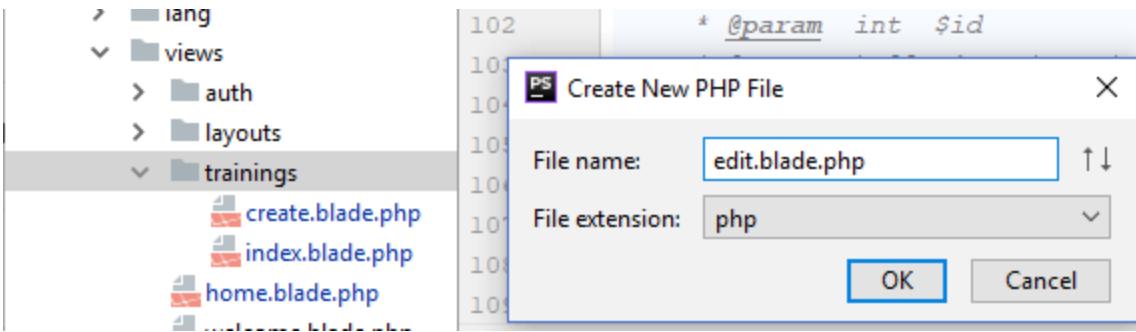
Listing Trainings for Update/Delete			
Laravel		<input type="button" value="Search"/>	
ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
6	Laravel Intermediate	Laravel Intermediate level to handle join tables	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
7	Laravel Advanced	To develop own boiler-plate	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Pengguna akan klik pada butang edit (jingga) untuk memilih rekod mana yang mahu dikemaskini. Apa yang berlaku apabila pengguna klik butang edit diprogramkan dalam fail `TrainingController.php`. Buka fail tersebut dalam `app/Http/Controller`, fokus kepada function `edit()`.

(GitHub –
<https://github.com/khirulnizam/start1/blob/master/app/Http/Controllers/TrainingController.php>).

Arahan `return view('trainings/edit')` akan membuka borang kemaskini dalam `resources/views/trainings/ edit.blade.php`. Jadi anda perlu sediakan borang yang boleh kemaskini rekod `Training`.

Untuk membuat borang HTML Form ini, klik-kanan pada folder `resources/views/trainings` dan pilih *Create New PHP file* dari menu.



Apa yang penting adalah borang untuk paparkan data lama dan boleh diedit. Lihat pada kod berikut;

```
<input type="text" class="form-control" name="trainingname" value="  
{{$training->trainingname}} ">
```

Itu adalah arahan memaparkan data lama dalam value textbox.

```
<div class="card-body">
    <h2>Edit Training Record</h2>
    <br>
    <form method="post" action="{{action('TrainingController@update', $id)}}>
        @csrf
        <input name="_method" type="hidden" value="PATCH">

        <label for="trainingname">Training Name:</label>
        <input type="text" class="form-control" name="trainingname"
               value="{{{$training->trainingname}}}">

        <label for="desc">Description:</label>
        <input type="text" class="form-control" name="desc"
               value="{{{$training->desc}}}">

        <label for="trainer">Trainer:</label>
        <input type="text" class="form-control" name="trainer"
               value="{{{$training->trainer}}}">

        <button type="submit" class="btn btn-success">Save Update</button>
    </form>
```

Output kod di atas dari antaramuka *resources/views/trainings/ edit.blade.php*

Insert || Listing for Update/Delete

Insert new Training

Edit Training Record

Training Name:

Description:

Trainer:

 ← update to new data and save

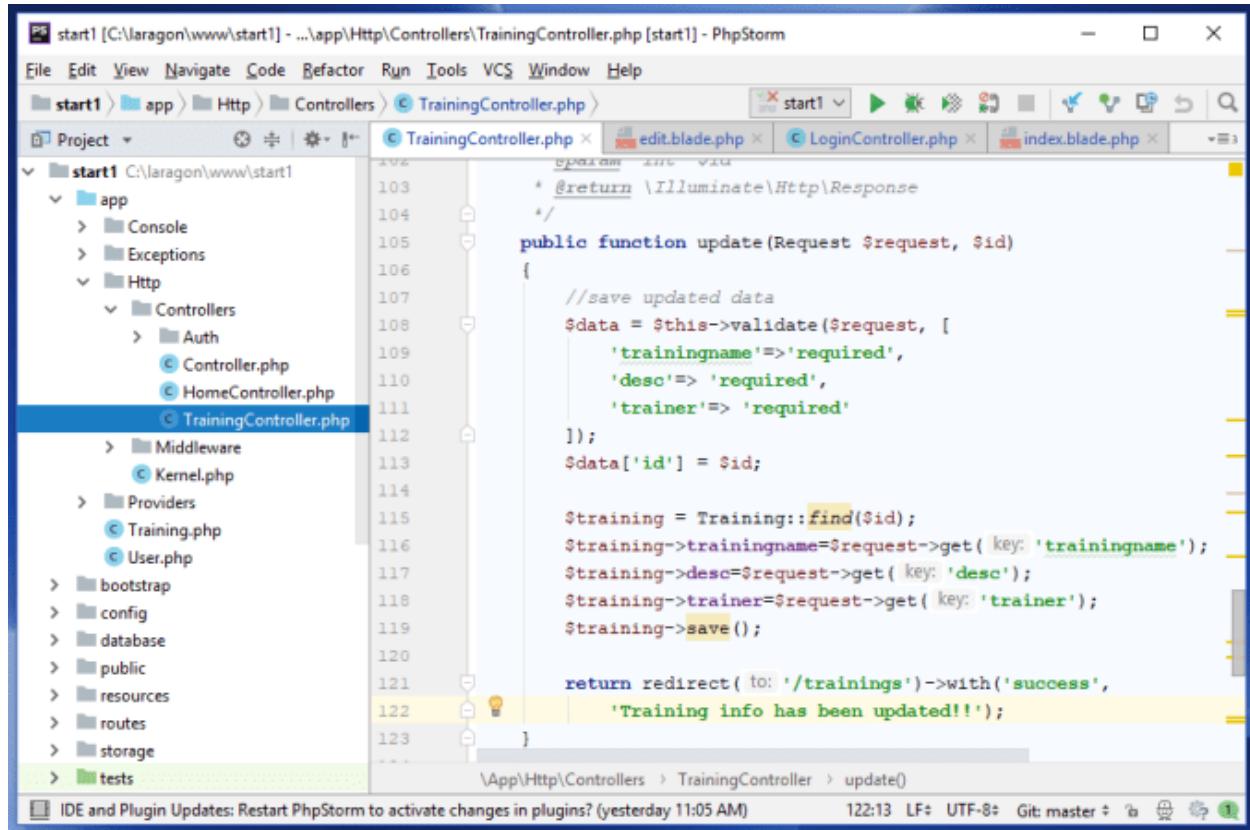
Save Update

Ke mana data dari form ini akan dihantar? Perhatikan pada baris kod di bawah , data akan dihantar ke function update() dalam fail *TrainingController* dengan membawa id rekod.

```
<form method="post" action="{{action('TrainingController@update', $id)}}>
```

Proses menyimpan data dikemaskini

Seterusnya rujuk fail *TrainingController.php* (dalam *app/Http/Controllers/*), di bahagian *function update()* . Tambah baris kod berikut;



```
start1 [C:\laragon\www\start1] - ...app\Http\Controllers\TrainingController.php [start1] - PhpStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
start1 app Http Controllers TrainingController.php
Project start1 app Http Controllers edit.blade.php LoginController.php index.blade.php
TrainingController.php
103 * @return \Illuminate\Http\Response
104 */
105 public function update(Request $request, $id)
106 {
107     //save updated data
108     $data = $this->validate($request, [
109         'trainingname'=>'required',
110         'desc'=> 'required',
111         'trainer'=> 'required'
112     ]);
113     $data['id'] = $id;
114
115     $training = Training::find($id);
116     $training->trainingname=$request->get( key: 'trainingname' );
117     $training->desc=$request->get( key: 'desc' );
118     $training->trainer=$request->get( key: 'trainer' );
119     $training->save();
120
121     return redirect( to: '/trainings')->with('success',
122         'Training info has been updated!!');
123 }
```

IDE and Plugin Updates: Restart PhpStorm to activate changes in plugins? (yesterday 11:05 AM) 12:13 LF: UTF-8: Git: master

Kod lengkap dalam GITHUB –

<https://github.com/khirulnizam/start1/blob/master/app/Http/Controllers/TrainingController.php>

Selepas data baru disimpan, sistem akan hantar paparan ke trainings/index.blade.php dan mesej berjaya dipaparkan.

success message

ID	Name	Desc	Action
1	Laravel 101	Basic laravel using Laragon	
2	Android 101	Basic Android	

Bagaimana memaparkan mesej kejayaan tersebut, tambahkan **div** berikut pada trainings/index.blade.php ([kod lengkap dalam GITHUB](#)).

Padam satu rekod

Terakhir untuk tutorial ini pilihan memadam rekod. Butang padam rekod berwarna merah berikut telah disediakan. Bersama borang tersebut ada borang (HTML form) untuk menghantar data dalam **method=post** .

Listing Trainings for Update/Delete			
Laravel		<input type="button" value="Search"/>	
ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
6	Laravel Intermediate	Laravel Intermediate level to handle join tables	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
7	Laravel Advanced	To develop own boiler-plate	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Kita perhati kod HTML untuk hasilkan butang padam berikut;

Pada *form action* sistem akan menghantar data id rekod yang pengguna pilih ke TrainingController/function destroy() method post adalah yang ditetapkan dalam route:list.

Rujuk fail app/Http/Controllers/TrainingController.php untuk menambah kod seterusnya dalam function destroy().

The screenshot shows the PhpStorm IDE interface with the following details:

- Title Bar:** start1 [C:\laragon\www\start1] - ...app\Http\Controllers\TrainingController.php [start1] - PhpStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Includes icons for Run, Stop, Refresh, and others.
- Project Explorer:** Shows the project structure under "start1". The "app" folder is expanded, showing "Http" which contains "Controllers". Inside "Controllers", "TrainingController.php" is selected and highlighted in blue.
- Code Editor:** Displays the PHP code for the "destroy" method:

```
124 /**
125 * Remove the specified resource from storage.
126 *
127 * @param int $id
128 * @return \Illuminate\Http\Response
129 */
130
131 public function destroy($id)
132 {
133     //delete selected record
134     $training = \App\Training::find($id);
135     $training->delete();
136     return redirect( to: 'trainings')->with('successdelete',
137         'Information has been deleted');
138 }
139 }
```

The code editor has a red box drawn around the entire "destroy" method block.
- Status Bar:** Shows "136:58 LF: UTF-8 Git: master 3b 57 1".

Paparan output selepas padam rekod.

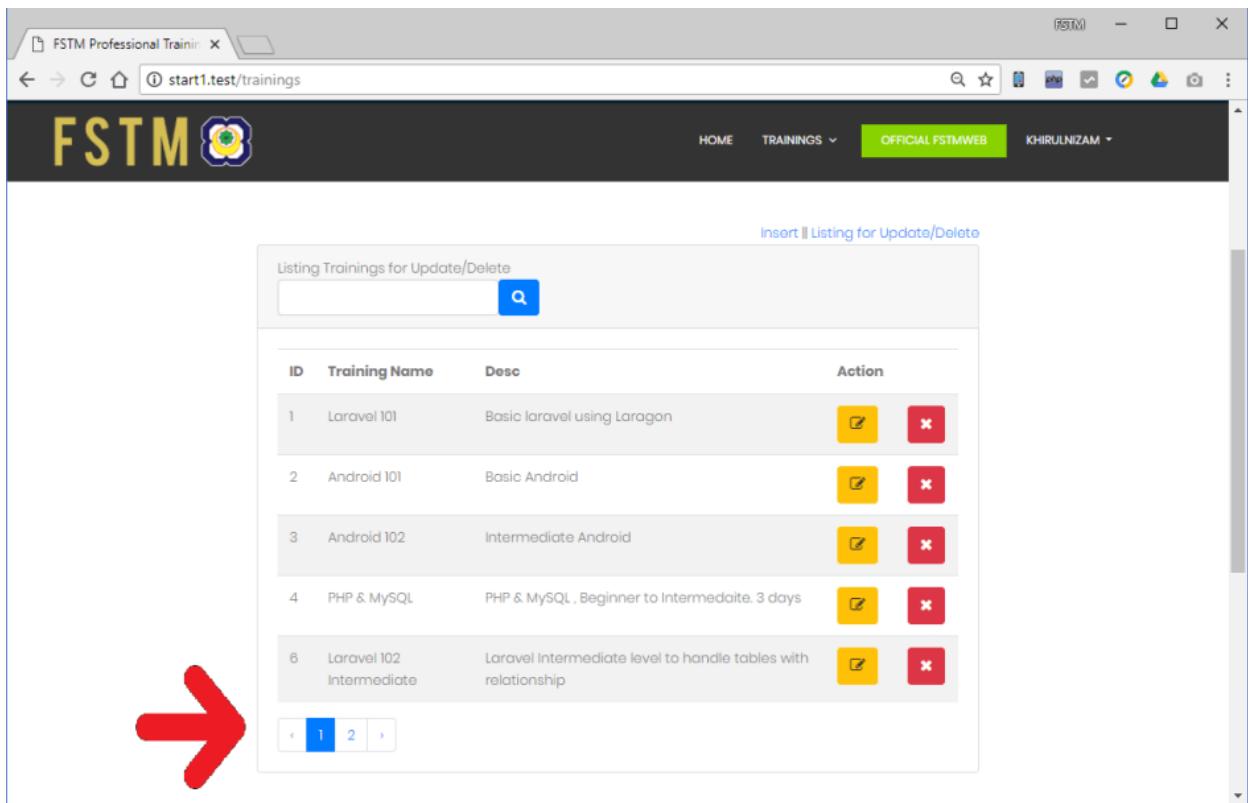
A screenshot of a web browser window titled "start1/test/trainings". The page header includes the FSTM logo, navigation links for HOME, SERVICES, PAGES, GET STARTED, and KHIRULNIZAM. A red arrow points from the text "success message record deleted" to a yellow success message box containing the text "Information has been deleted". Below the message box is a table listing two training entries:

ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel using Laragon	<input checked="" type="checkbox"/> <input type="button" value="x"/>
2	Android 101	Basic Android	<input checked="" type="checkbox"/> <input type="button" value="x"/>

Bagaimana memaparkan mesej kejayaan tersebut, tambahkan **div** berikut pada trainings/index.blade.php ([kod lengkap dalam GITHUB](#)).

[Selamat mencuba. Sila LIKE [fb.com/kuis.fstm](#) untuk menerima update tutorial seterusnya]

Tutorial 06 Paparan carian dengan *Pagination*



The screenshot shows a web browser window for 'FSTM Professional Trainer' at the URL 'start1.test/trainings'. The page displays a table of training programs with columns for ID, Training Name, Desc, and Action. Each row has a yellow edit icon and a red delete icon. A red arrow points to the pagination links at the bottom left of the table, which show pages 1 and 2.

ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel using Laragon	 
2	Android 101	Basic Android	 
3	Android 102	Intermediate Android	 
4	PHP & MySQL	PHP & MySQL . Beginner to Intermediate. 3 days	 
6	Laravel 102 Intermediate	Laravel Intermediate level to handle tables with relationship	 

Rekod setakat sepuluh data tak perlu kot *pagination* . Tapi makin lama, data pastinya akan berkembang. Tak patut kita paparkan seribu rekod dalam satu page. *Defeat the purpose of searching!*

Pagination merupakan kaedah memaparkan sebahagian hasil carian dan pengguna boleh pilih untuk paparkan hasil seterusnya dengan klik pada butang page seterusnya. Lihat contoh seperti ditunjukkan oleh anak panah pada rajah di atas.

Paginate() merupakan tambahan kepada *resultset* kita untuk implementasi kemudahan pagination.

Dalam function `index()` , fail `TrainingControllers.php` (lokasi `app/Http/Controllers/`) sila tambah kod seperti rajah di bawah.

```

start1 [C:\laragon\www\start1] - ...\\app\\Http\\Controllers\\TrainingController.php [start1] - PhpStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
start1 > app > Http > Controllers > TrainingController.php
Project app
> Console
> Exceptions
> Http
> Controllers
> Auth
Controller.php
HomeController.php
TrainingController.php
Middleware
Kernel.php
Providers
Training.php
User.php
bootstrap
config
database
public
resources
> assets
> lang
views
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
public function index(Request $request)
{
    // app/Http/Controllers
    // TrainingController.php
    //fetch search key
    $search=$request->get('key', 'txtsearch');

    //Strainings= new Training();
    if ($search==null){
        //display all record
        //Strainings= Training::all()->toArray();
        $strainings=Training::paginate(5);
        return view('trainings.index', compact('strainings'));
    }
    else{
        //display record based on search key
        $strainings= Training::where('trainingname', 'like', "%{$search}%")->paginate(5);
        //strainings=$strainings->get();
        //strainings=$strainings->paginate(5);
        return view('trainings.index', compact('strainings'));
    }
}
//end function index

```

Seterusnya, untuk memaparkan resultset pilihan page ke-2, ke-3 dan seterusnya, tambah di bawah jadual paparan data/rekod.

```

start1 [C:\laragon\www\start1] - ...\\resources\\views\\trainings\\index.blade.php [start1] - PhpStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
start1 > resources > views > trainings > index.blade.php
Project resources
> assets
> lang
views
> auth
> layouts
> trainings
create.blade.php
edit.blade.php
index.blade.php
flash-message.blade.php
home.blade.php
welcome.blade.php
welcome_old.blade.php
68
69
70
71
72
73
74
75
76
77
78
79
80
81
<button class="btn h
    <i class="fa fa-
    </button>
</td>
</tr>
@endforeach
<tbody>
</table>
{{ $strainings->links() }}
</div>
</div>
</div>

```

Dan akhirnya kita uji perubahan kod dalam browser, sepatutnya dapat paparan berikut.

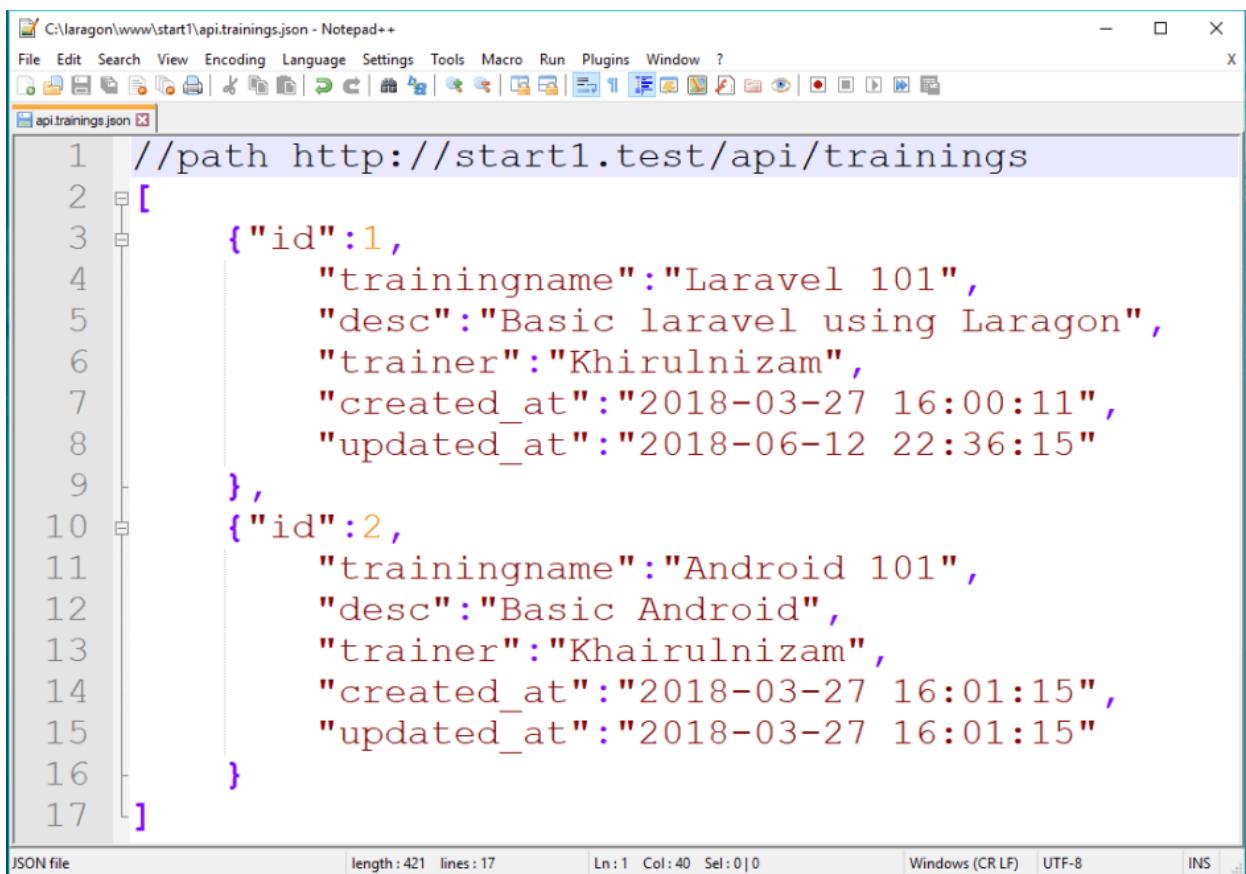
The screenshot shows a web application interface for managing training programs. At the top, there's a header with the FSTM logo and navigation links for HOME, TRAININGS, OFFICIAL FSTMWEB, and KHIRULNIZAM. Below the header is a search bar and a link to 'Insert || Listing for Update/Delete'. The main content area is titled 'Listing Trainings for Update/Delete' and contains a table with the following data:

ID	Training Name	Desc	Action
1	Laravel 101	Basic laravel using Laragon	
2	Android 101	Basic Android	
3	Android 102	Intermediate Android	
4	PHP & MySQL	PHP & MySQL , Beginner to Intermediate. 3 days	
6	Laravel 102 Intermediate	Laravel Intermediate level to handle tables with relationship	

At the bottom of the table, there are navigation buttons for page 1, 2, and 3.

[Selamat mencuba. Sila LIKE fb.com/kuis.fstm untuk menerima update tutorial seterusnya]

Tutorial 07 Laravel REST API



The screenshot shows a Notepad++ window displaying a JSON file named 'api.trainings.json'. The file contains two training records:

```
//path http://start1.test/api/trainings
[{"id":1, "trainingname":"Laravel 101", "desc":"Basic laravel using Laragon", "trainer":"Khirulnizam", "created_at":"2018-03-27 16:00:11", "updated_at":"2018-06-12 22:36:15"}, {"id":2, "trainingname":"Android 101", "desc":"Basic Android", "trainer":"Khairulnizam", "created_at":"2018-03-27 16:01:15", "updated_at":"2018-03-27 16:01:15"}]
```

The Notepad++ interface includes a toolbar, menu bar, status bar at the bottom, and a vertical scroll bar on the right.

Tutorial seterusnya akan menyenarai rekod melalui format data JSON.

Penyediaan persekitaran Laravel menggunakan LARAGON mungkin anda perlu lalui dua tutorial ini sekiranya masih baru dalam Laravel –

T1: Kaedah install & guna Laragon (<http://fstm.kuis.edu.my/blog/laravel1>)

T2: Asas CRUD dalam Laravel (<http://fstm.kuis.edu.my/blog/laravel2>)

REST API

Dengan keperluan mendesak capaian pangkalan data melalui app mobile dan framework JavaScript, Sistem web pangkalan data sekarang ini perlu menyokong penghantaran data universal. RESTful API merupakan pilihan yang terbaik untuk membenarkan capaian kepada data tanpa perlu berkongsi katalaluan pangkalan data kepada semua orang.

Format penghantaran data yang biasa diguna adalah format JSON (atau XML). Dalam tutorial ini kita akan membincangkan bagaimana Laravel menjana data API dalam format JSON.

Contoh JSON format data

Di akhir tutorial ini kita akan menjana format data JSON seperti dalam gambar di atas.

HTTP Verbs

Dalam RESTful API, kita akan menggunakan HTTP verbs sebagai perlakuan operasi ke atas data.

GET: dapatkan data

POST: masukkan data baharu

PUT: kemaskini data

DELETE: padam data

routes/api.php

Ini merupakan fail yang penting untuk menjana API data. Tambah beberapa baris kod berikut berdasarkan *HTTP verbs*.

```
Use App\Training; //use Trainings model

// trainings API
// Let's create the basic endpoints for our application:
// create, retrieve the list, retrieve a single one,
update, and delete.
// On the routes/api.php file, we can simply do these:

Route::get('trainings', function() {
// If the Content-Type and Accept headers are set to
'application/json',
// this will return a JSON structure. This will be
cleaned up later.
return Training::all();
//this is to list all records
//available from http://start1.test/api/trainings
```

```

}) ;

Route::get('trainings/{id}', function($id) {
return Training::find($id);
//this is to list one specific record by id
//available from http://start1.test/api/trainings/3
}) ;

Route::post('trainings', function(Request $request) {
return Trainings::create($request->all());
//to insert a new record
}) ;

Route::put('trainings/{id}', function(Request $request,
$id) {
$training = Trainings::findOrFail($id);
$training->update($request->all()) ;

return $training;
//to search record
}) ;

Route::delete('trainings/{id}', function($id) {
Training::find($id)->delete();

return 204;
//to delete one record with specific id
}) ;

```

Dah selesai tambah kod berikut dalam routes/api.php, sila test melalui path yang ditambah /api/

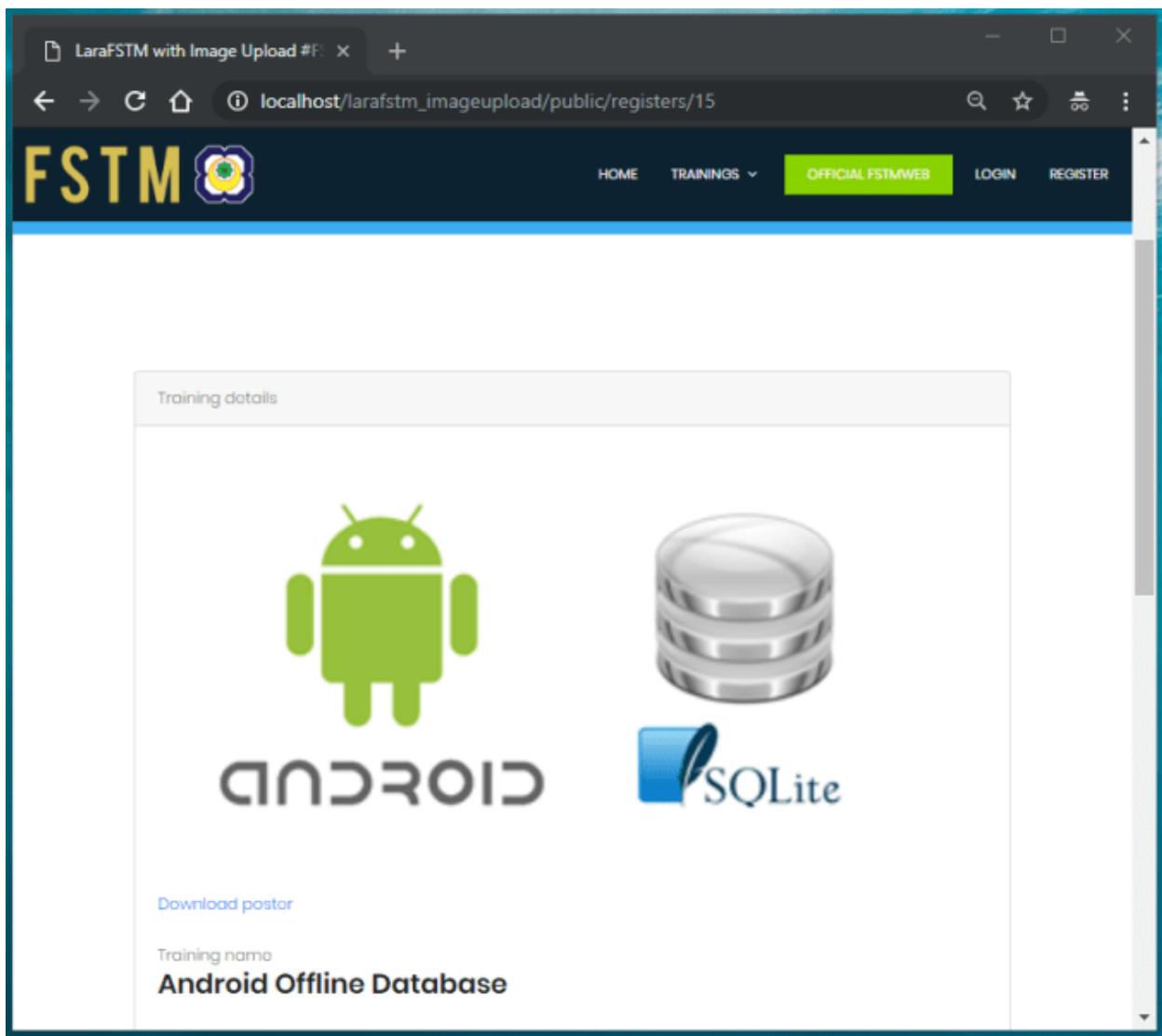
Contoh: *http://start1.test/ api /trainings*

Tutorial 08 Upload imej ke sistem Laravel

Gigih kawan-kawan kita dari UMP, dari KUIS yang buat projek akhir datang cuti Deepavali sedut ilmu Laravel. Semangat peserta berjangkit ke trainer untuk menggigihkan diri menambah koleksi modul Laravel seterusnya. Macam-macam kisah cabaran programmer2 nak sampai ke KUIS – kami kagum dengan anda semua. Kali ini nak sambung dengan cara-cara upload imej dalam sistem web berasaskan Laravel.

Cuma sebelum itu admin harap tuan-puan dah pernah buat tutorial yang terdahulu sebelum harung tutorial ini. Kalau belum sila ke [Tutorial CRUD Pengenalan Laravel](#)

Kod lengkap tutorial ini ada di
github.com/khirulnizam/larafstm_imageupload



Contoh page yang memaparkan imej yang telah dimuat-naik ke server
Kerangka tutorial ini;

Menambahkan medan dalam jadual (table) trainings untuk menyimpan nama fail imej yang dimuat-naik ke server.

Mengubahsuai setting storan dalam *config/filesystem.php*

Menyediakan link (virtual folder dalam public) yang akan mencapai fail imej dalam storan.

Menyediakan borang (*form*) memilih dan memuat-naik fail imej.

Menambah proses simpan fail imej dalam *TrainingController.php* function *store()*.

Membuat paparan imej yang telah dimuat-naik.

Menambahkan medan dalam jadual (table) *trainings*

Medan tambahan ini diperlukan untuk menyimpan nama fail imej yang dimuat-naik ke server. Kita telah menyediakan jadual ***trainings*** dalam tutorial lepas. Pengubah-suaian struktur jadual dalam projek Laravel mesti dilakukan dalam *migration file*. Buat satu migration-file tambahan dengan arahan berikut;

```
php artisan make:migration add_file_to_trainings
```

Ini akan menghasilkan migration-file dalam folder database/migrations.

Masukkan kod berikut untuk menambah lajur pada table ***trainings*** yang sedia ada.

The screenshot shows a Notepad++ window with the file '2018_11_08_005356_add_file_to_trainings.php' open. The code adds three nullable string columns to the 'trainings' table: 'filename', 'mime', and 'original_filename'. The file is located in the 'migrations' directory of a Laravel project named 'larafstm_imageupload'. The Notepad++ interface includes a file browser on the left showing the project structure, and various status indicators at the bottom.

```
13 *+
14
15 public function up()
16 {
17     //adding columns to existing table trainings
18
19     //first, to create this additional migration file
20     //php artisan make:migration add_file_to_trainings
21
22     Schema::table('trainings', function (Blueprint $table) {
23         $table->string('filename')->nullable();
24         $table->string('mime')->nullable();
25         $table->string('original_filename')->nullable();
26     });
27
28     //after adding some columns definition to store filename,
29     //run
30     //php artisan migrate
31 }
32 }
```

Klik untuk besarkan kod menambah lajur

Medan filename menyimpan nama fail imej, mime menyimpan jenis file , original_filename menyimpan nama fail imej yang asal.

```
Schema::table( 'trainings' , function (Blueprint $table)
{
    $table ->string( 'filename' )->nullable();
    $table ->string( 'mime' )->nullable();
```

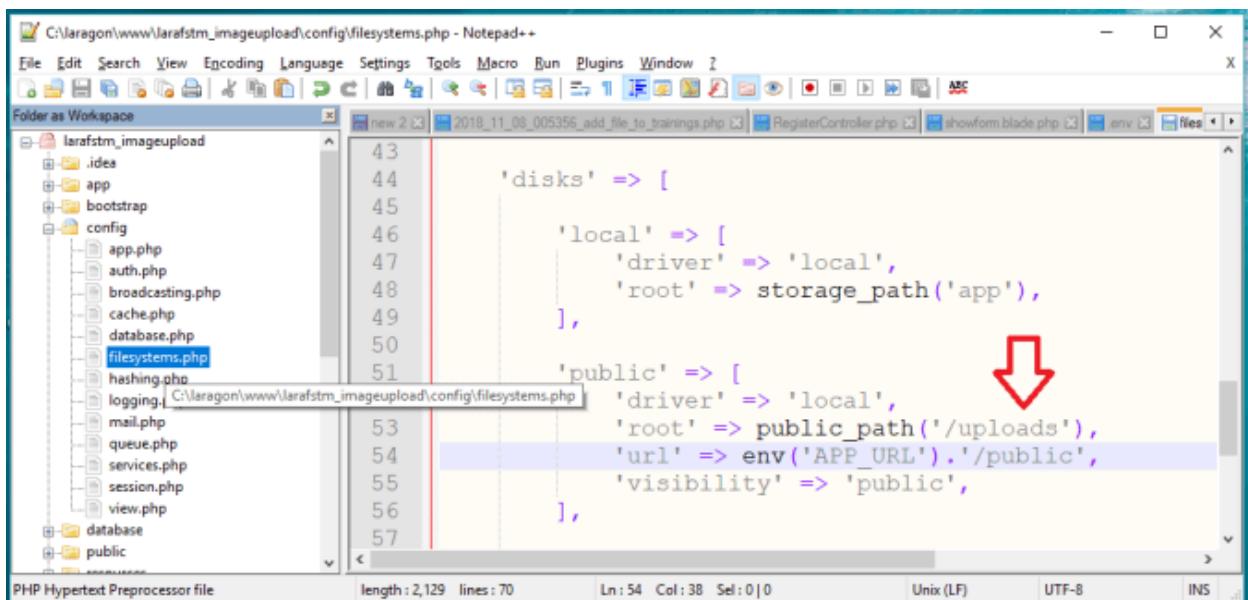
```
$table ->string( 'original_filename' )->nullable();  
});
```

Seterusnya laksanakan migration untuk menambah medan-medan (field) tambahan ke pangkalan data;

php artisan migrate

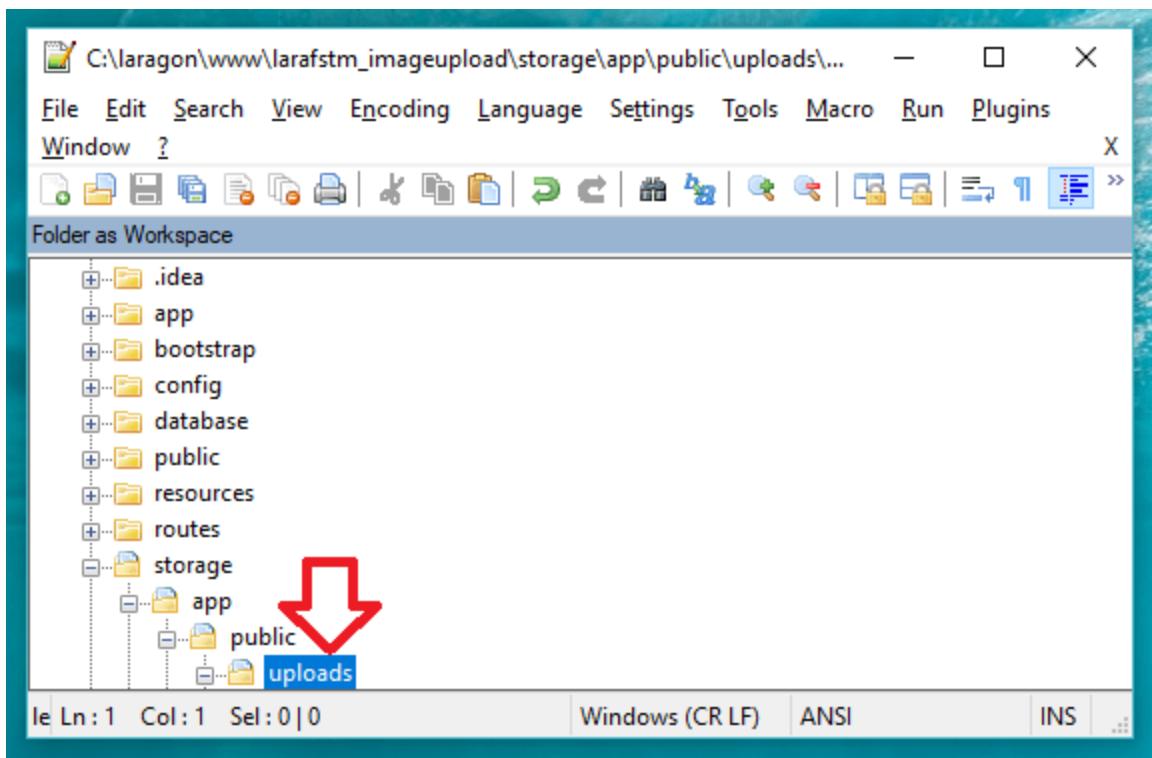
Mengubahsuai setting storan dalam config/filesystem.php

Sekiranya ada keperluan ubah setting dalam fail ini. Settings ini merujuk kepada di mana fail imej yang user muat-naik akan disimpan.



```
C:\laragon\www\larafstm_imageupload\config\filesystems.php - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Folder as Workspace  
C:\larafstm_imageupload  
.idea  
app  
bootstrap  
config  
app.php  
auth.php  
broadcasting.php  
cache.php  
database.php  
filesystems.php  
hashing.php  
logging.php  
mail.php  
queue.php  
services.php  
session.php  
view.php  
database  
public  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
'disks' => [  
    'local' => [  
        'driver' => 'local',  
        'root' => storage_path('app'),  
    ],  
    'public' => [  
        'driver' => 'local',  
        'root' => public_path('/uploads'),  
        'url' => env('APP_URL').'/public',  
        'visibility' => 'public',  
    ],  
];
```

Dalam tutorial ini kita akan menambah folder *uploads* kepada folder **storage/app/public**. Jadi local folder untuk simpan fail imej berada dalam **storage/app/public/uploads**. Secara manual cipta folder *uploads* dalam lokasi yang ditetapkan.



Lokasi folder uploads tempat simpan fail imej yang user muat-naik

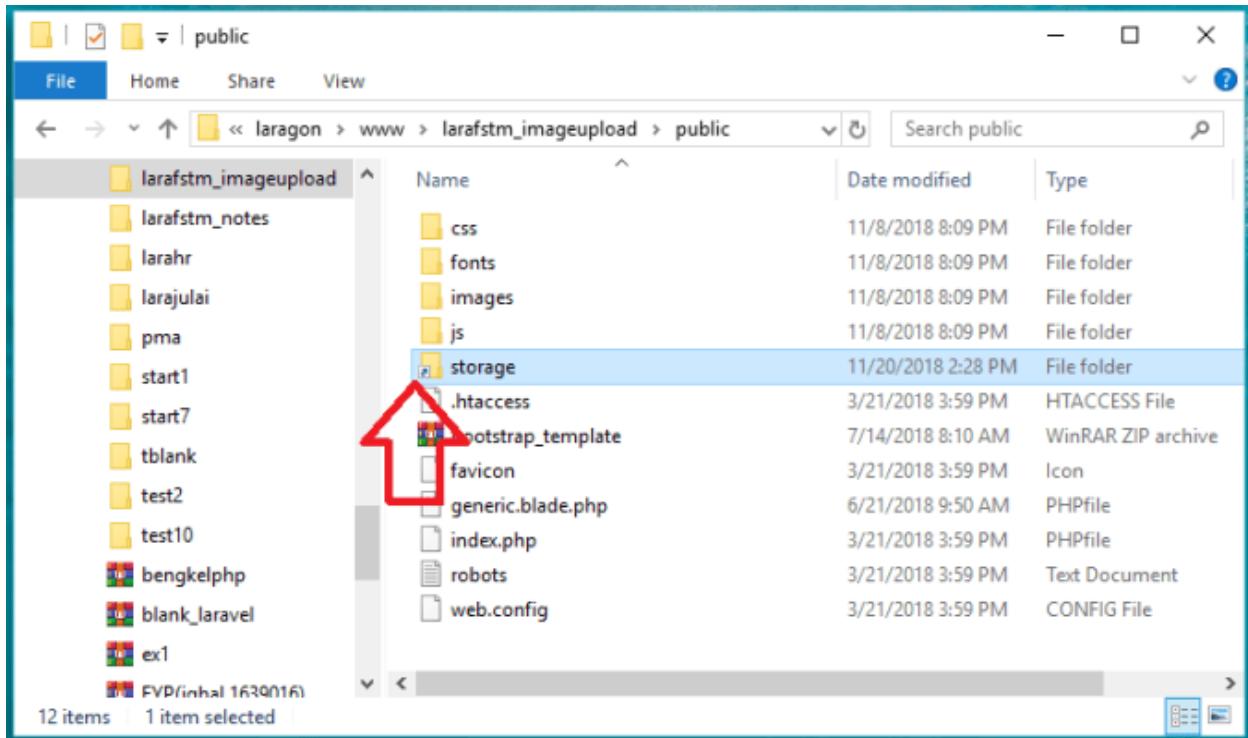
Menyediakan link (virtual folder dalam public)

Proses seterusnya yang akan memberikan capaian public kepada fail imej dalam ***storage/app/public/uploads***. Proses ini perlu dilakukan sebab storan fail imej yang dimuat-naik user berada dalam folder bukan-public, makanya *virtual folder* (link) perlu diwujudkan supaya imej boleh dipaparkan pada paparan public.

Arahan berikut akan menghasilkan virtual link tersebut;

`php artisan storage:link`

Hasil daripada arahan `php artisan storage:link` akan menghasilkan virtual-link yang akan membolehkan storan fail-fail image melalui folder *public* projek Laravel. Perhatikan pada icon folder terdapat anak panah kecil biru menandakan ia bukan folder fizikal, cuma sekadar virtual link.



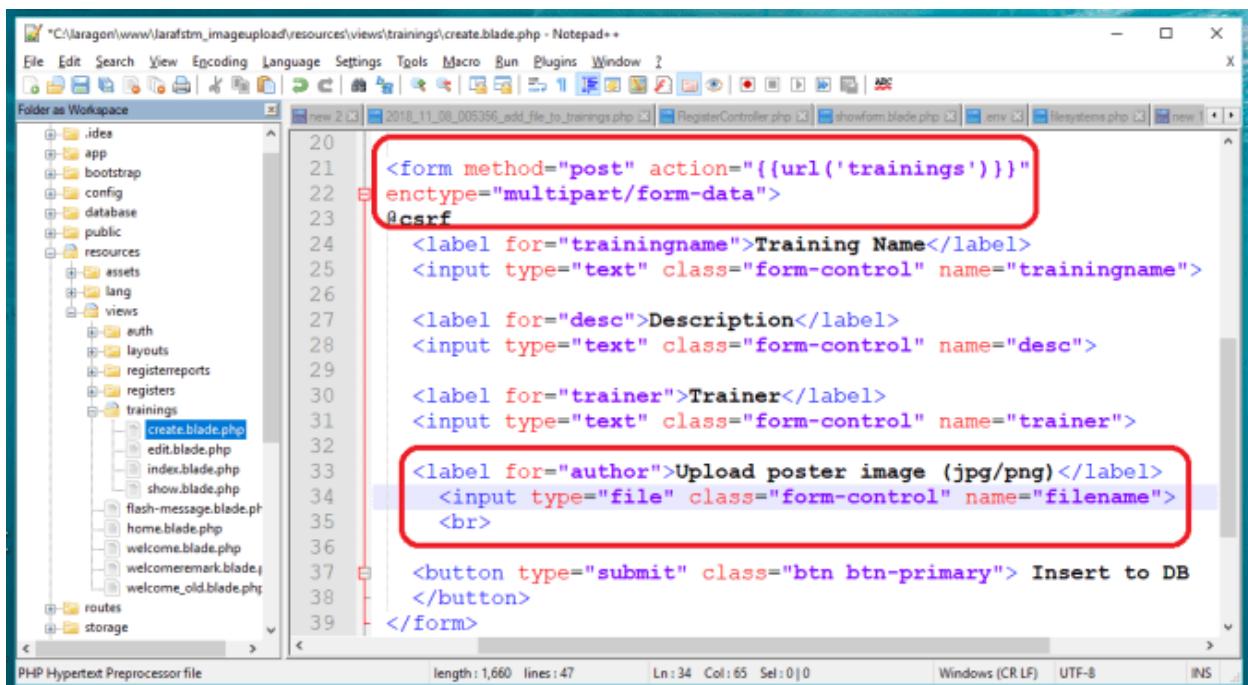
Virtual-link folder storage untuk simpan fail-fail imej dimuat-naik pengguna

Menyediakan borang (*form*) memilih dan memuat-naik fail imej.

Sedikit perubahan pada borang cipta rekod perlu ditambah untuk membenarkan pengguna memilih fail imej dan memuatnaik ke server. Jadi buka fail blade di resources/views/trainings/create.blade.php, dan tambah input-control file-upload.

Tambah juga pada form atribut berikut;

enctype="multipart/form-data"

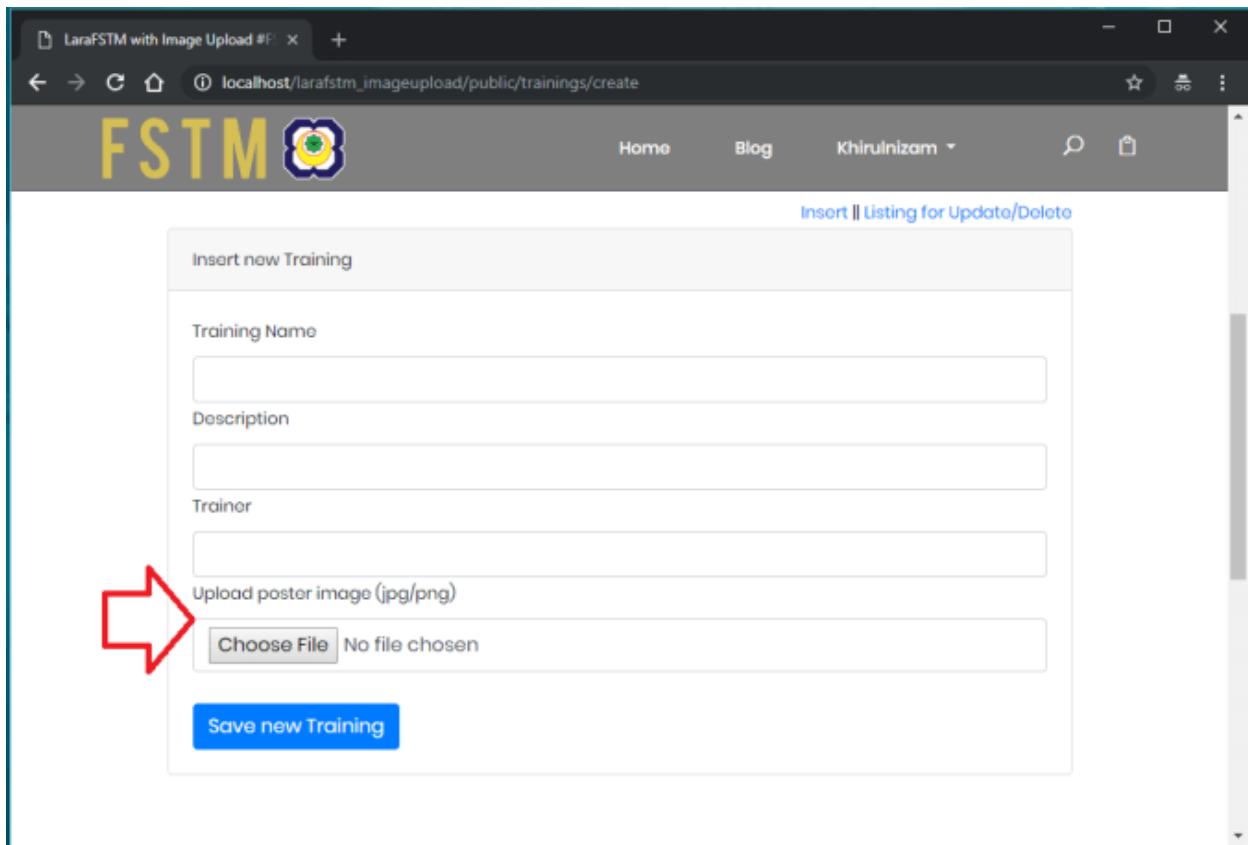


```

20
21 <form method="post" action="{{url('trainings')}}">
22   @csrf
23
24   <label for="trainingname">Training Name</label>
25   <input type="text" class="form-control" name="trainingname">
26
27   <label for="desc">Description</label>
28   <input type="text" class="form-control" name="desc">
29
30   <label for="trainer">Trainer</label>
31   <input type="text" class="form-control" name="trainer">
32
33   <label for="author">Upload poster image (jpg/png)</label>
34   <input type="file" class="form-control" name="filename">
35   <br>
36
37   <button type="submit" class="btn btn-primary"> Insert to DB
38   </button>
39 </form>

```

Tambahan enctype dan file-upload input control
 Borang yang terhasil dengan muat-naik fail.



LaraFSTM with Image Upload #FSTM

localhost/larafstm_imageupload/public/trainings/create

FSTM

Home Blog Khirulnizam ▾

Insert || Listing for Update/Delete

Insert new Training

Training Name

Description

Trainer

Upload poster image (jpg/png)

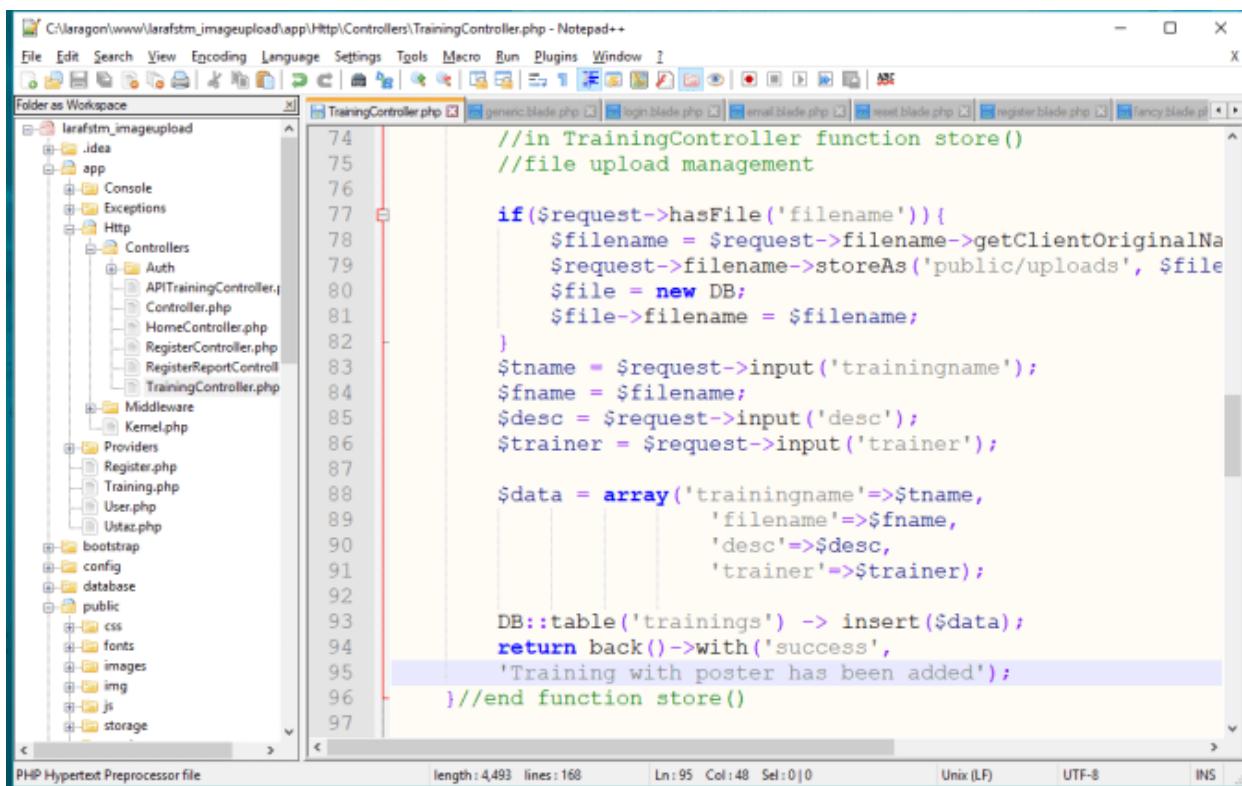
Choose File No file chosen

Save new Training

Borang yang terhasil dengan muat-naik fail

Menambah proses simpan fail imej

Seterusnya dalam TrainingController.php function store(), tambah kod untuk simpan fail imej yang user muat-naik ke server.



The screenshot shows the Notepad++ interface with the file 'TrainingController.php' open. The code adds logic to handle file uploads:

```
//in TrainingController function store()
//file upload management

if($request->hasFile('filename')){
    $filename = $request->filename->getClientOriginalName();
    $request->filename->storeAs('public/uploads', $file);
    $file = new DB;
    $file->filename = $filename;
}

$fname = $request->input('trainingname');
$desc = $request->input('desc');
$trainer = $request->input('trainer');

$data = array('trainingname'=>$fname,
              'filename'=>$filename,
              'desc'=>$desc,
              'trainer'=>$trainer);

DB::table('trainings') -> insert($data);
return back()->with('success',
                     'Training with poster has been added');
}//end function store()
```

Fungsi store() dalam TrainingController untuk simpan fail imej dimuat-naik

```
//// TrainingController.php
public function store(Request $request )
{
//validate insert new record process is here
$training = $this ->validate(request(), [
'trainingname' => 'required' ,
'desc' => 'required' ,
'trainer' => 'required'

]);
```

```

//in TrainingController function store()
//file upload management

if ( $request ->hasFile( 'filename' ) ){
$filename = $request ->filename->getClientOriginalName();
$request ->filename->storeAs( 'public/uploads' ,
$filename );
$file = new DB;
$file ->filename = $filename ;
}
$tname = $request ->input( 'trainingname' );
$fname = $filename ;
$desc = $request ->input( 'desc' );
$trainer = $request ->input( 'trainer' );

$data = array ( 'trainingname' => $tname ,
'filename' => $fname ,
'desc' => $desc ,
'trainer' => $trainer );

DB::table( 'trainings' ) -> insert( $data );
return back()->with( 'success' ,
'Training with poster has been added' );
} //end function store()

Kod penuh di GITHUB
https://github.com/khirulnizam/larafstm\_imageupload/blob/master/app/Http/Controllers/TrainingController.php

```

Paparan imej yang telah dimuat-naik

Page seterusnya akan memilih rekod yang telah disimpan bersama fail imej – show().

Tambahan butang show-all boleh dibuat dengan menambah lajur pada senarai rekod seperti di bawah. Untuk menambahkan lajur dan butang-butang tersebut gunakan kod berikut;

```

<td>
<a href=
"{{ action('TrainingController@show', $training['id']) }}"
class = "btn btn-info btn-sm" >
<i class = "fa fa-file-text-o" ></i>
</a>
</td>

```

The screenshot shows a web browser displaying a Laravel application titled 'FSTM'. The page is titled 'Listing Trainings for Update/Delete'. It features a search bar and a table with four rows of training data. Each row includes an 'Action' column with three icons: a blue square with a white document, a yellow square with a white checkmark, and a red square with a white 'X'. A red box highlights this 'Action' column.

ID	Training Name	Desc	Action
17	Fillmora Video Editing	Professional Video editing with Fillmora	
18	Crowds	Picture of crowds in lab	
19	Kamal	Kamal	
20	Network	CCNA	

Butang papar semua maklumat dalam rekod
 Seterusnya pada TrainingController@show function – tambah kod untuk
 mengaut semua data dari rekod yang dipilih.

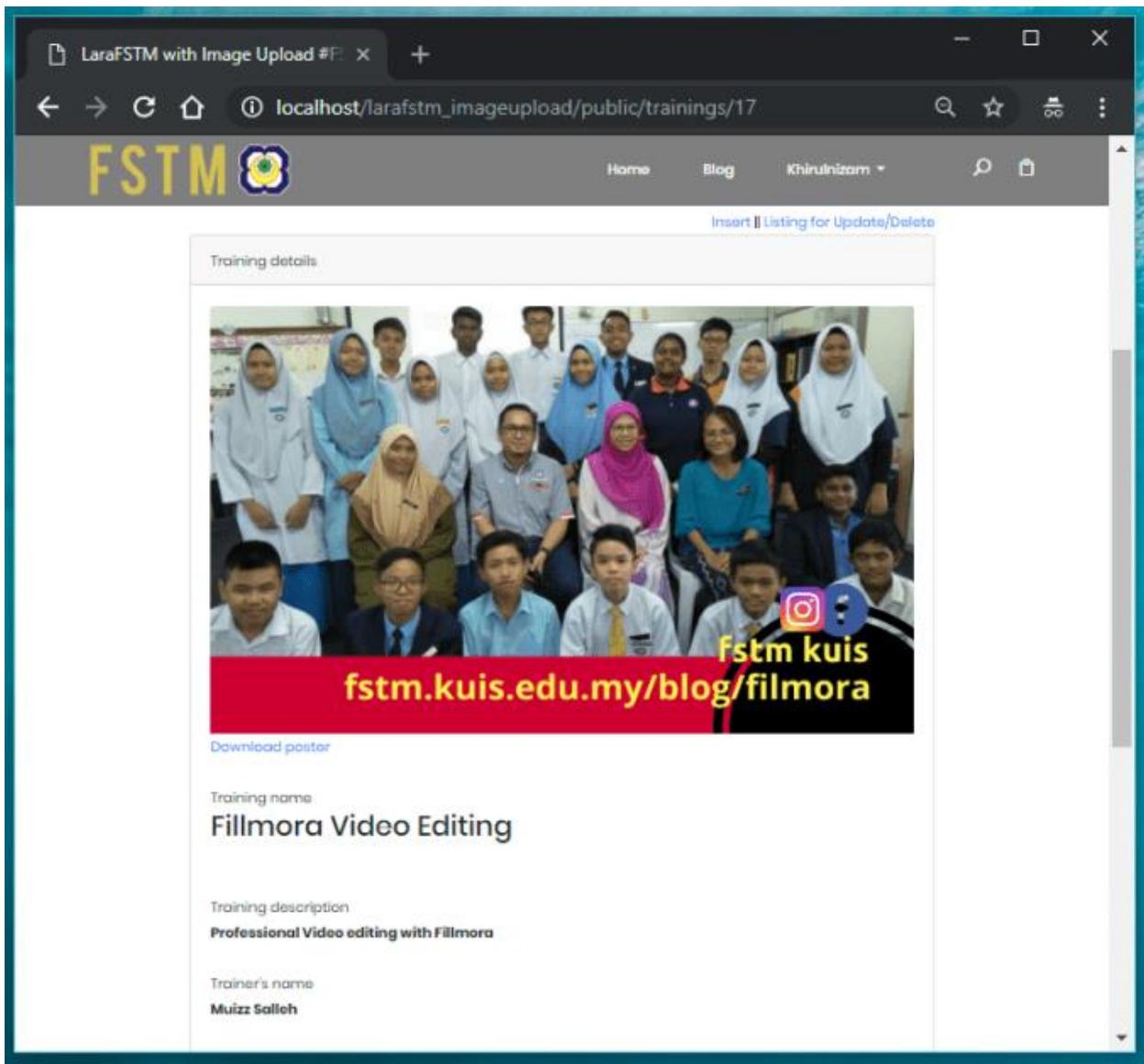
```

public function show($id)
{
//TrainingController@show
//display complete record
$training = Training::find($id);
return view('trainings.show',compact('training','id'));

```

}

Function show() pada akhir kod akan redirect kepada page *trainings/show.blade.php* seperti di bawah.



Paparan imej yang telah dimuat-naik

Perhatikan paparan imej terhasil daripada kod segmen berikut;

```
<!-- Sebahagian kod HTML dalam  
trainings/show.blade.php segment ini memaparkan imej dari  
virtual folder public/storage/uploads -->
```

```
< img class = "card-img-top"
src="{!!url('storage/uploads/'.$training->filename) !!}"
alt="{!!$training->filename!!}>
< a href="{!!url('storage/uploads/'.$training-
>filename) !!}"> Download poster </ a >
(Kod lengkap cara memaparkan imej tersebut boleh diperolehi dari
https://github.com/khirulnizam/larafstm\_imageupload/blob/master/resources/vi
ews/trainings/show.blade.php )
```

Selamat mencuba dan berjaya!

Tutorial 09 Peranan pengguna (user roles ACL)

Example:
Azroy hasRole as 'participant'
Khirulnizam hasRole as 'admin'



Table: roles

id	name	description
1	admin	Admin User
2	participant	Participant User

Table: role-user

id	role_id	user_id
1	1	1
2	1	2
3	2	3
4	2	4
5	2	5

Table: user

id	name	email
1	Admin Name	admin@example.com
2	Abd Rahman	kerul@gmail.com
3	Participant Name	participant@example.com
4	Shaqirin	shaq@gmail.com
5	Azroy	azroy@gmail.com

fstm.kuis.edu.my/blog/laravel

User hasRole relationship in Laravel visualised

Kod lengkap projek Laravel kali ini InsyaAllah dah siap dan boleh muat-turun dari [GITHUB – https://github.com/khirulnizam/lara_acl](https://github.com/khirulnizam/lara_acl) , cuma tutorial masih dalam proses pembangunan. Tuan/puan sila bookmark dulu, kami akan umumkan bila sudah siap. Templet Bootstrap untuk admin dasboard/panel diubahsuai dari [SB-ADMIN](#) . Telah disediakan templet SB-ADMIN dalam layouts dan juga bootstrap framework dalam *public/sbadmin*

Rangka tutorial

- Controller menambah admin baru
- Antaramuka tambah admin baru
- Database migration
- Controller store new admin
- Antaramuka kemaskini maklumat admin (update/delete)

Intro projek tutorial ACL Laravel

Tutorial kali ini cuba menerangkan dalam bentuk proof-of-concept bagaimana nak implement beberapa peranan pengguna (user-role) dalam Laravel. Dalam contoh sebelum ini, pengguna ada dua kategori; guest & admin. Guest tidak perlu login untuk melihat maklumat awam (public). Tetapi untuk menambah rekod, kemaskini atau padam rekod perlu pengguna yang sah yang mesti melalui proses login.

Bagaimana kalau kita ada peranan pengguna yang lain contohnya peserta boleh mendaftar kursus, mengedit user-profile atau membatalkan kursus yang telah ditdaftar?

Senarai peranan pengguna (user-roles)

Tetamu (guest) – boleh memaparkan maklumat umum contohnya carian semua kursus.

Peserta (participant) – perlu daftar, login dan boleh daftar kursus. Boleh juga melihat kembali senarai kursus yang telah didaftar, batal penyertaan, memperbaharui maklumat profile.

Penyelenggara (admin) – hanya didaftarkan oleh admin yang lain, perlu login untuk menyelenggara senarai kursus dan penyertaan. Boleh buat carian, kemaskini rekod training, padam rekod training, senaraikan penyertaan peserta.

Model Training dan migrations

Model Training beserta *migration file* boleh dicipta dengan arahan berikut;

`php artisan make:model Training -m`

Selepas laksanakan arahan di atas, buka dalam folder *database/migrations/create_trainings_table* dan tambah arahan berikut dalam *function up()* ;

```
public function up()
{
Schema::create( 'trainings' , function (Blueprint $table)
) {
$table ->increments( 'id' );
$table ->string( 'trainingname' );
$table ->string( 'desc' );
$table ->string( 'trainer' );
$table ->timestamps();
} );
}
```

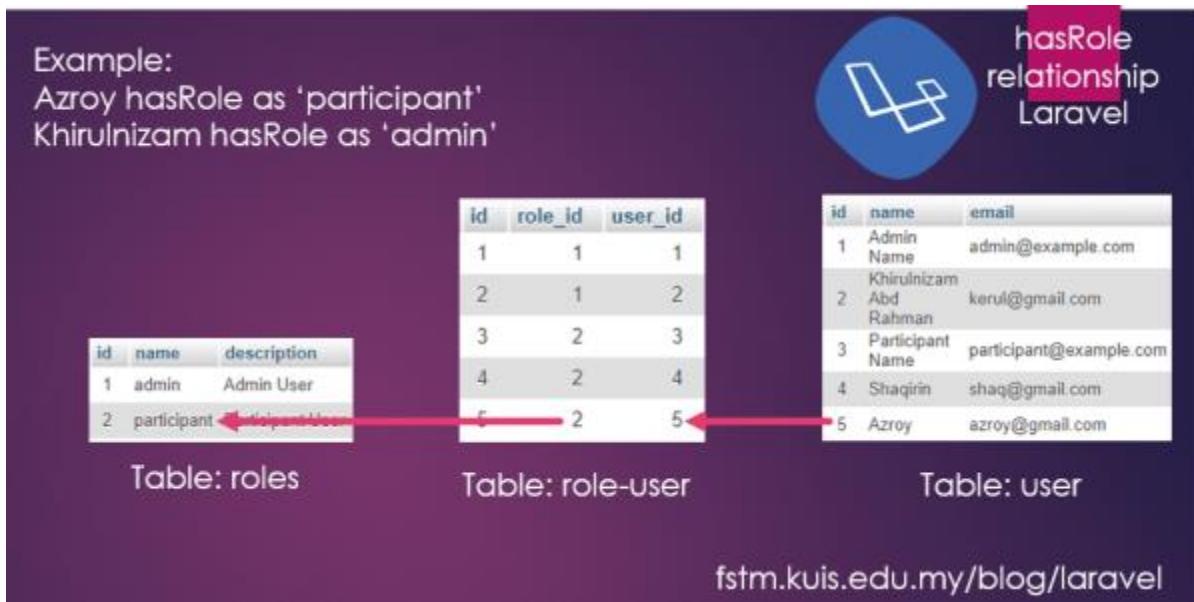
Model User dan migrations

Model beserta *migration file* telah tersedia untuk table *users* . Biasanya cuma perlu *make:auth* untuk implement **Auth::user** facilities.

```
php artisan make:auth
```

Model Role dan migrations

Konsep/carta rekabentuk database Role seperti di bawah. Kita cuma akan hasilkan dua roles, iaitu ‘admin’ dan ‘participant’. Sekiranya anda perlukan multiple user-roles sila google ‘permission’.



User hasRole relationship in Laravel visualised

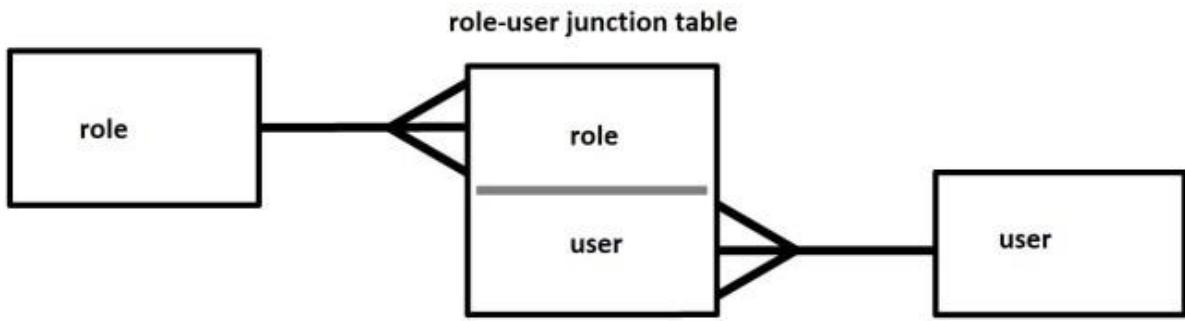
`php artisan make:model Role -m`

Selepas laksanakan arahan di atas, buka dalam folder

`database/migrations/create_roles_table` dan tambah arahan berikut dalam `function up()` :

```
public function up()
{
    Schema::create('roles', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('description');
        $table->timestamps();
    });
}
```

Oleh kerana hubungan (*relationship*) antara pengguna dan peranan adalah *many-to-many*, makanya kita perlukan satu table tambahan penghubung roles dengan users.



Untuk mencipta migration file laksanakan arahan di bawah;

`php artisan make:migration create_role_user_table`

Buka fail yang terhasil dalam database/migration/create_role_user_table dan tambah kod baris berikut;

```

public function up()
{
Schema::create('role_user', function (Blueprint $table) {
$table ->increments('id');
$table ->integer('role_id')->unsigned();
$table ->integer('user_id')->unsigned();
$table ->timestamps();
});
}

```

Untuk menetapkan hubungan many-to-many antara User dan Role , dalam User model tambahkan *function roles()* dan sediakan kod seperti di bawah;

```

public function roles()
{
return $this ->belongsToMany(Role::class);
//file location app/User.php
}

```

Manakala dalam Role model, tambahkan *function user()* pula;

```

public function users()

```

```

{
return $this->belongsToMany(User::class);
//file location app/Role.php
}

```

Seeder untuk sediakan data cubaan table users & roles

Untuk menghasilkan dummy data, tambah dalam fail *database/seeds/UserTableSeeder.php*:

```

public function run()
{
//
$role_admin = Role::where('name', 'admin')->first();
$role_participant = Role::where('name', 'participant')
->first();
//add admin user

$admin = new User();
$admin->name = 'Admin Name';
$admin->email = 'admin@example.com';
$admin->password = bcrypt('secret');
$admin->save();
$admin->roles()->attach($role_admin);

$admin = new User();
$admin->name = 'Khirulnizam Abd Rahman';
$admin->email = 'kerul@gmail.com';
$admin->password = bcrypt('abc123');
$admin->save();
$admin->roles()->attach($role_admin);

```

```

//add participant users
$participant = new User();
$participant ->name = 'Participant Name' ;
$participant ->email = 'participant@example.com' ;
$participant ->password = bcrypt( 'secret' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

$participant = new User();
$participant ->name = 'Shaqirin' ;
$participant ->email = 'shaq@gmail.com' ;
$participant ->password = bcrypt( 'shaq123' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

$participant = new User();
$participant ->name = 'Azroy' ;
$participant ->email = 'azroy@gmail.com' ;
$participant ->password = bcrypt( 'azroy123' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

}

Dalam fail database/seeds/ RoleTableSeeder.php;

```

```

public function run()
{
//
$role_employee = new Role();
$role_employee ->name = 'admin' ;
$role_employee ->description = 'Admin User' ;
$role_employee ->save();

$role_manager = new Role();
$role_manager ->name = 'participant' ;
$role_manager ->description = 'Participant User' ;
$role_manager ->save();
}

```

Dalam fail database/seeds/ **TrainingTableSeeder**.php;

```

public function run()
{
// $this->call(UserTableSeeder::class);
DB::table( 'trainings' )->insert([
'trainingname' => 'Laravel 101' ,
'desc' => 'Web development using PHP framework Laravel' ,
'trainer' => 'Khirulnizam Abd Rahman' ,
'filename' => 'sig-fstm.png' ,
]) ;

DB::table( 'trainings' )->insert([
'trainingname' => 'Android SQLite' ,
'desc' => 'Android Studio apps development with local database using SQLite' ,
'trainer' => 'Khirulnizam Abd Rahman' ,
'filename' => 'android-sqlite.png' ,
]) ;

DB::table( 'trainings' )->insert([
'trainingname' => 'Video Editing Filmora' ,
'desc' => 'An easy video editing using Filmora editor' ,
'trainer' => 'Muizz Salleh' ,
'filename' => 'muizz-filmora.png' ,
]) ;
}

```

Dalam fail *database/seeds/ DatabaseSeeder.php*

```

public function run()
{
// Role comes before User seeder here.
$this ->call(RoleTableSeeder:: class );
// User seeder will use the roles above created.
$this ->call(UserTableSeeder:: class );
// Training seeder dummy data
$this ->call(TrainingTableSeeder:: class );

}

```

Seterusnya dalam model app/User.php , tambahkan tiga fungsi-fungsi berikut;

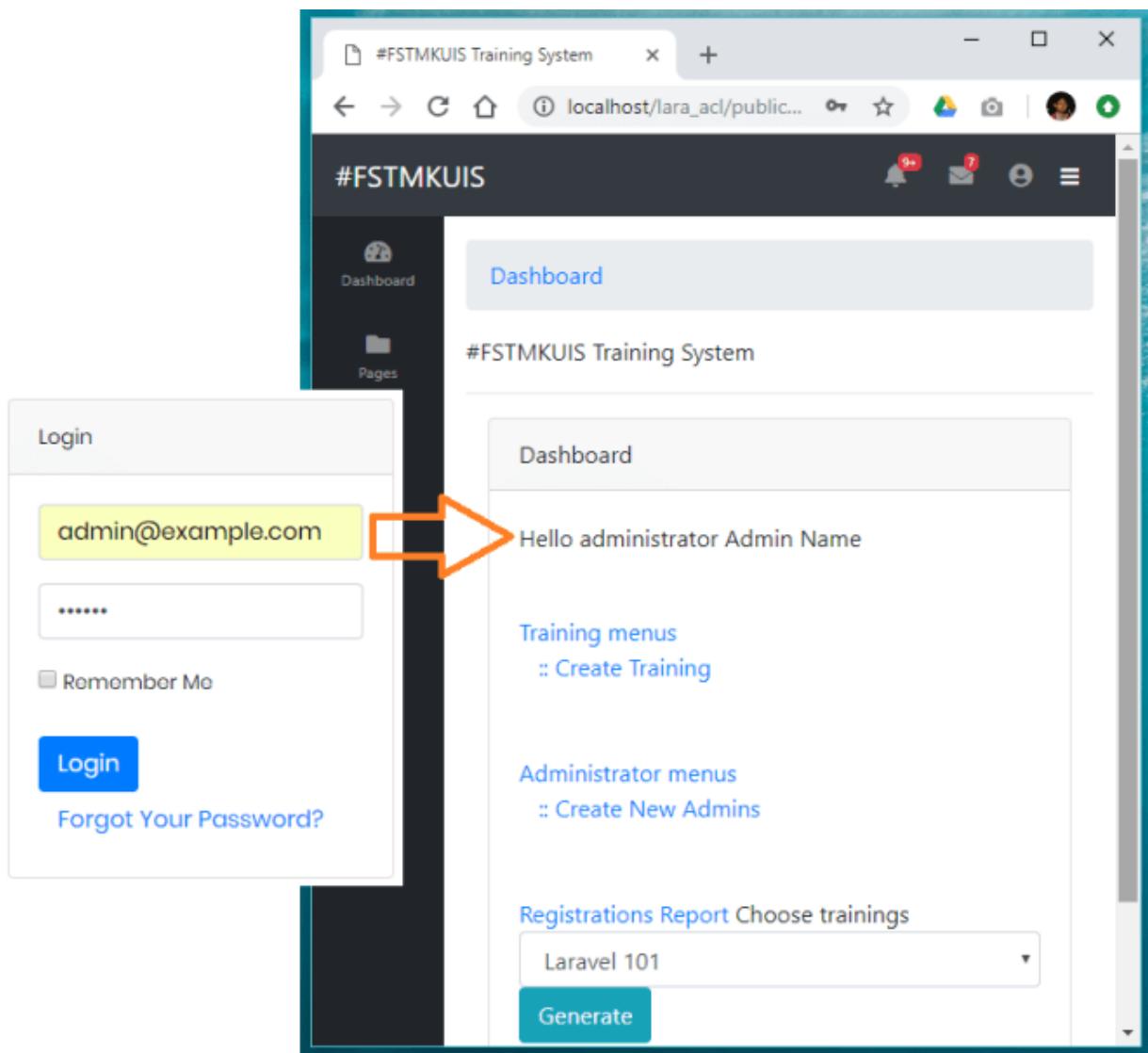
```
public function authorizeRoles( $roles )
{
    if ( is_array ( $roles ) ) {
        return $this ->hasAnyRole( $roles ) || 
        abort(401, 'This action is unauthorized.' );
    }
    return $this ->hasRole( $roles ) || 
        abort(401, 'This action is unauthorized.' );
}
/** 
 * Check multiple roles
 * @param array $roles
 */
public function hasAnyRole( $roles )
{
    return null !== $this ->roles()->whereIn( 'name' , $roles
)->first();
}
/** 
 * Check one role
 * @param string $role
 */
public function hasRole( $role )
{
    return null !== $this ->roles()->where( 'name' , $role )-
>first();
}
```

Percubaan login sebagai admin / participant

Berdasarkan UserTableSeeder di atas, terdapat pengguna yang diberikan role ‘admin’ dan ‘participant’. Cuba paparkan skrin login dan cuba kedua-dua

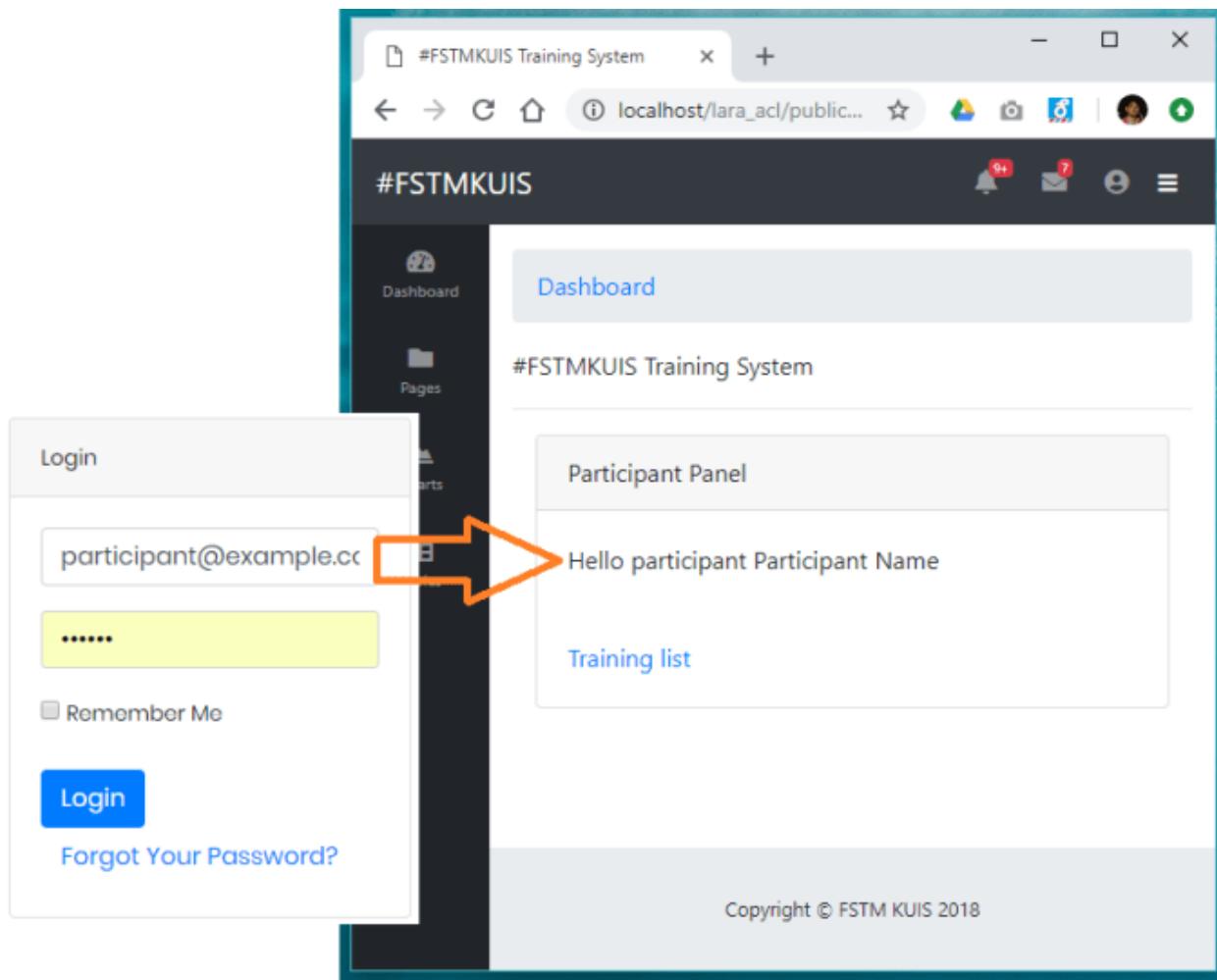
peranan penngguna berikut. Perhatikan bahawa kedua-dua peranan user mendapat skrin panel pengguna yang berbeza.

Admin username: *admin@example.com* ; dengan password *secret*.



User admin login, paparan skrin admin panel

Participant username: *participant@example.com* ; dengan password *secret* .



User participant login, paparan skrin participant panel

Perbezaan paparan ini boleh dilakukan dengan menujukan kepada skrin yang berbeza apabila jenis peranan berbeza login masuk sistem. Ini boleh dilakukan dalam **app/HomeController.php**

```
public function index(Request $request)
{
    if (Auth::user()->hasRole( 'admin' )) {
        $trainings = Training::all()->toArray();
        return view( 'home' , compact( 'trainings' ));
    }
    else if (Auth::user()->hasRole( 'participant' )) {
        return view( 'homeparticipant' );
    }
}
```

Apabila peranan pengguna adalah ‘admin’ (hasRole), HomeController akan redirect ke ***resources/views/home.blade.php***, manakala sebaliknya apabila peranan pengguna (hasRole) adalah ‘participant’- maka akan dipaparkan pula ***resources/views/homeparticipant.blade.php***

resources/views/homeparticipant.blade.php

```
@extends('layouts.sbadmin')
@section('content')

< div class = "container" >

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

< div class = "col-md-10" >

< div class = "card" >

< div class = "card-header" >Participant Panel</ div >

< div class = "card-body" >
@if (session('status'))

< div class = "alert alert-success" >
{{ session('status') }}
</ div >

@endif

Hello participant {{ Auth::user()->name } }

< a href = "{{ url('trainings') }}" > Training list</ a >

</ div >
```

```

</ div >

</ div >

</ div >

</ div >

@endsection
resources/views/home.blade.php

@extends('layouts.sadmin')
@section('content')

< div class = "container" >

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

< div class = "col-md-10" >

< div class = "card" >

< div class = "card-header" >Dashboard</ div >

< div class = "card-body" >
@if (session('status'))

< div class = "alert alert-success" >
{{ session('status') }}
</ div >

@endif

Hello administrator {{ Auth::user()->name }}
```

```

< a href = "{{ url('trainings') }}" > Training menus</ a >

< a href = "{{ url('trainings/create') }}" >&nbsp;&nbsp;
:: Create Training</ a >

< a href = "{{ url('admins') }}" > Administrator menus</ a >

< a href = "{{ url('admins/create') }}" >&nbsp;&nbsp; ::>
Create New Admins</ a >

< a href = "#" > Registrations Report</ a >
Choose trainings

< form method = "GET" action =
"{{url('registerreports')}}" class = "form-inline" >
@csrf
< select name = "id" class = "form-control" >
@foreach($trainings as $training)
< option value = "{{ $training['id'] }}" >
{{ $training['trainingname'] }} </ option >
@endforeach
</ select >

< button type = "submit" class = "btn btn-info" >
Generate </ button >
</ form >

</ div >

</ div >

</ div >

```

```
</ div >
```

```
</ div >
```

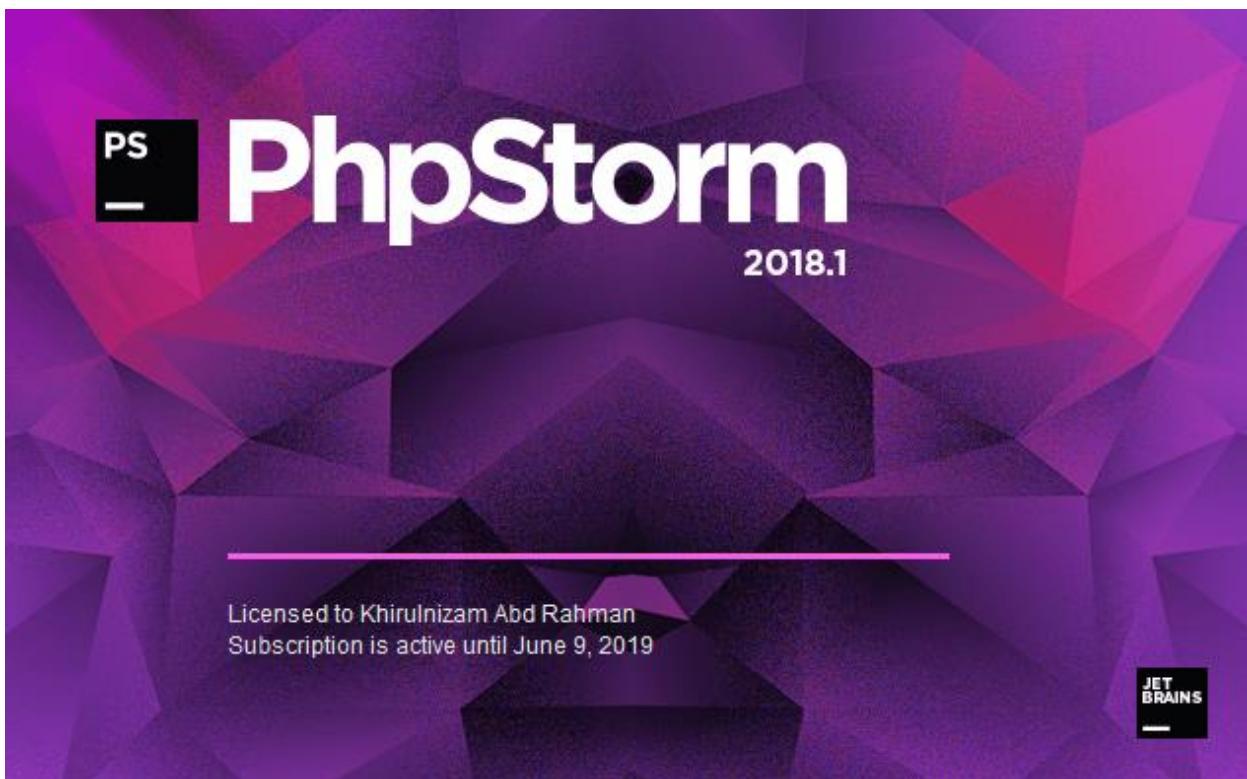
```
@endsection
```

Kod lengkap projek Laravel kali ini InsyaAllah dah siap dan boleh muat-turun dari **GITHUB – https://github.com/khirulnizam/lara_acl**

Selamat mencuba dan berjaya!

Rujukan: <https://medium.com/@ezp127/laravel-5-4-native-user-authentication-role-authorization-3dbae4049c8a>

Tutorial 10 PHPStorm IDE Laravel



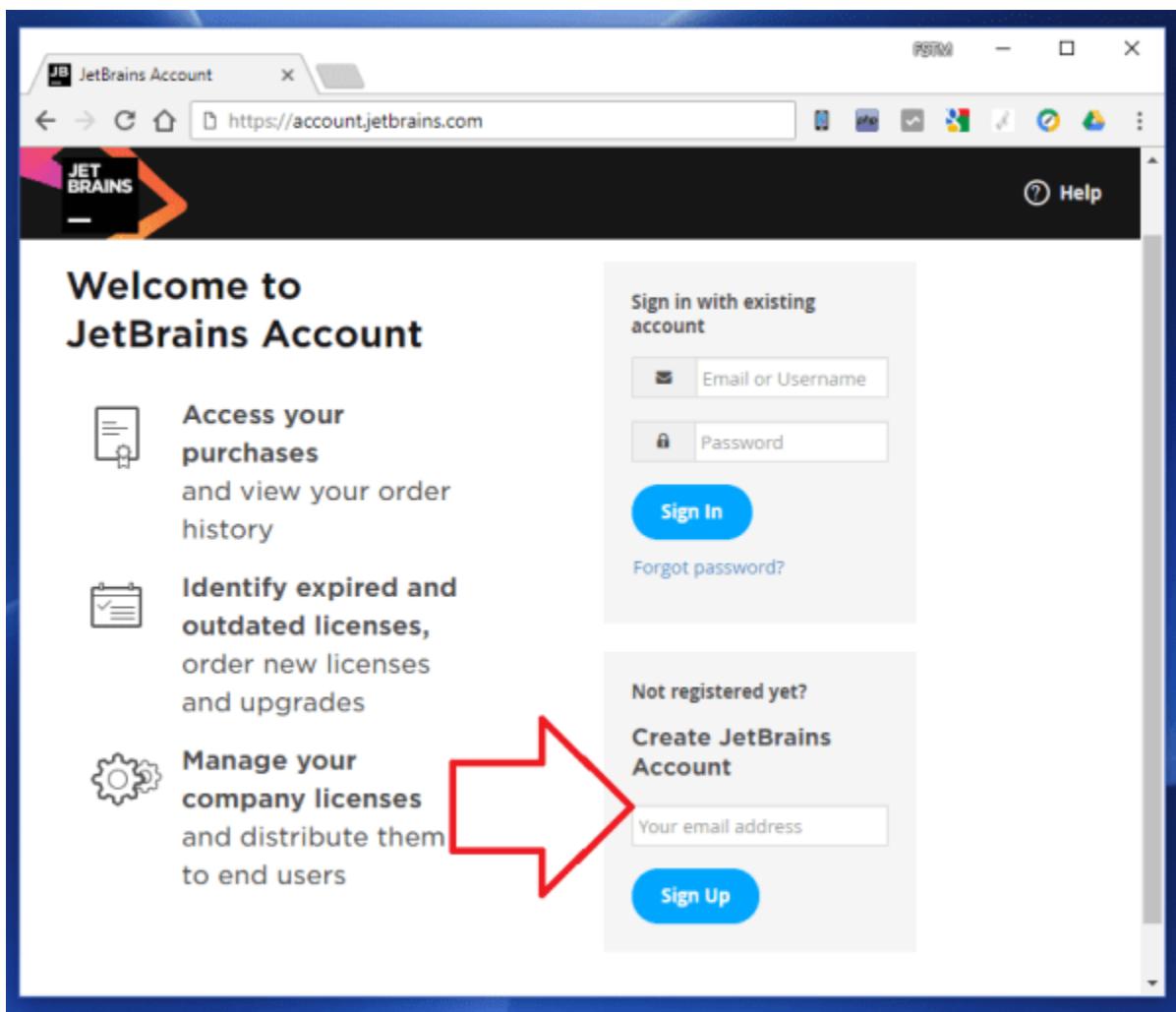
IDE (Integrated Development Environment) yang dimaksudkan bukan sahaja berupaya menulis kod HTML, CSS, JavaScript, PHP & Laravel, ia juga mampu membantu mengesan latar sintax, cadangan autocomplete. IDE seperti Eclipse for PHP, Visual Studio dan PHPStorm antara yang berkebolehan sebagai IDE yang terbaik, manakala Atom, VSCode & Sublime bertindak hanya sebagai editor kod sahaja. Penulis memilih PHPStorm sebagai IDE kegemaran (artikel ini cuma pendapat peribadi penulis berdasarkan pengalaman).

Pemasangan Perisian PHPStorm

PHPStorm boleh dimuat-turun dari laman syarikat pembangunnya iaitu JetBrains (yang terkenal dengan IDE Java IntelliJ – IDE terbaik Java). Namun ianya bukan perisian percuma (sekiranya memerlukan sumber terbuka/pilihan percuma rujuk *Eclipse for PHP*). Masa percubaan hanya 30 hari, selepas itu sekiranya syarikat anda ada bajet boleh beli lesen.

Lesen percuma pendidikan

Pastikan anda berdaftar dulu dalam website JetBrains – <https://account.jetbrains.com> .



Sekiranya anda pelajar atau pensyarah IPTA/IPTS boleh dapatkan lesen penggunaan percuma dengan daftar menggunakan email *edu* yang disediakan institusi. Penulis akan menunjukkan cara permohonan lesen percuma menggunakan email ***@kuis.edu.my .

Boleh mohon lesen untuk pendidikan/kelas/makmal komputer [di sini](#) , atau [lesen individu di sini](#) . Untuk permohonan lesen individu pelajar/pensyarah, sila sediakan email yang ada **edu** pada alamat email anda atau boleh sertakan kad matrix/pelajar bersama permohonan.

The screenshot shows a web browser window for the JetBrains Account. The URL is https://account.jetbrains.com/licenses. The page displays a '1 License' section for a 'JetBrains Product Pack for Students'. The license details are as follows:

Licensed to:	Khirulnizam Abd Rahman
License restriction:	For educational use only
Valid through:	June 09, 2019

A 'Download' button is available, and a 'License ID' field is shown with a redacted value. A link to 'Download activation code for offline usage' is also present.

Selepas anda berjaya mendapatkan lesen pendidikan, login akaun anda melalui perisian PHPStorm yang telah anda download dan install. Perisian PHPStorm desktop boleh diperolehi di sini;

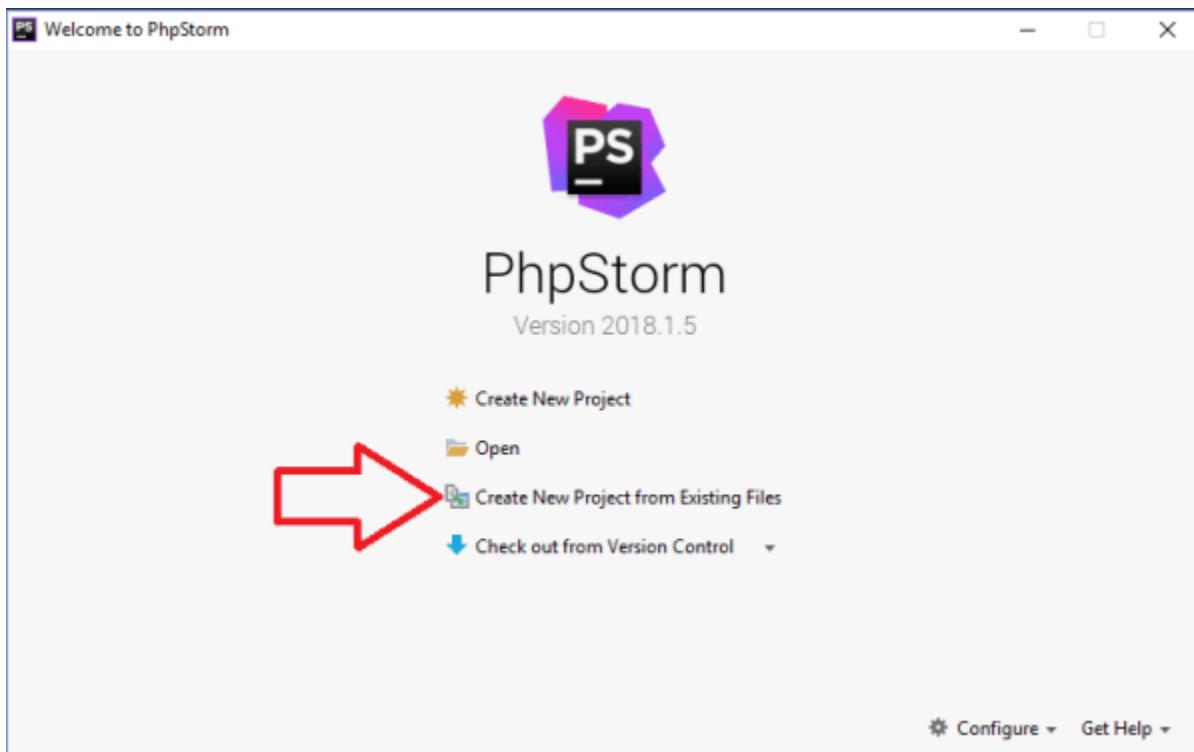


<https://www.jetbrains.com/phpstorm/>

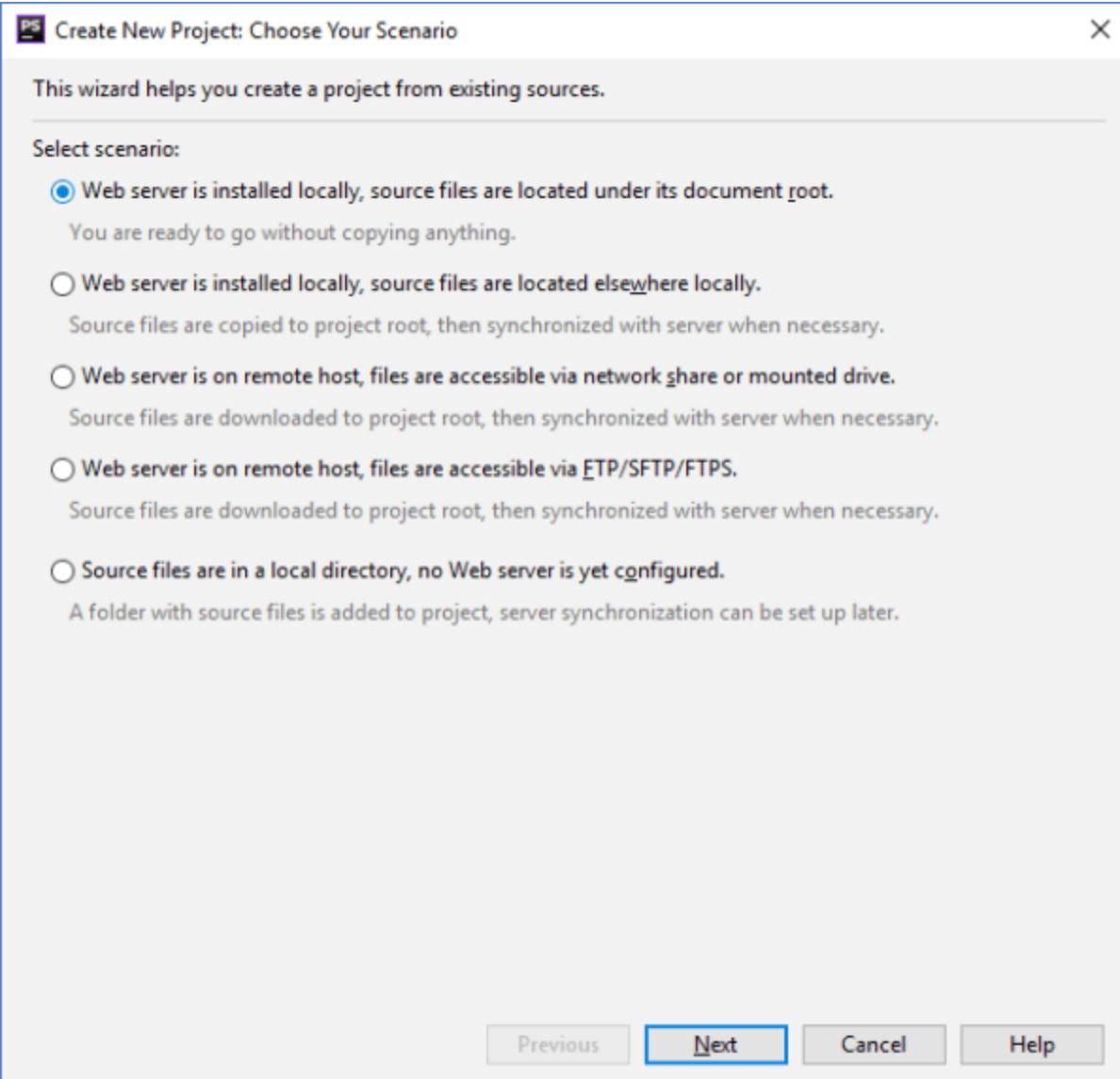
Projek Laravel

Pengujian ini dibuat melalui PC dengan Windows10 dan RAM 8GB (4GB kurang disyorkan memandangkan perisian agak berat). Projek yang akan dibangunkan menggunakan framework Laravel. Memandangkan BLADE juga disokong dalam PHPStorm.

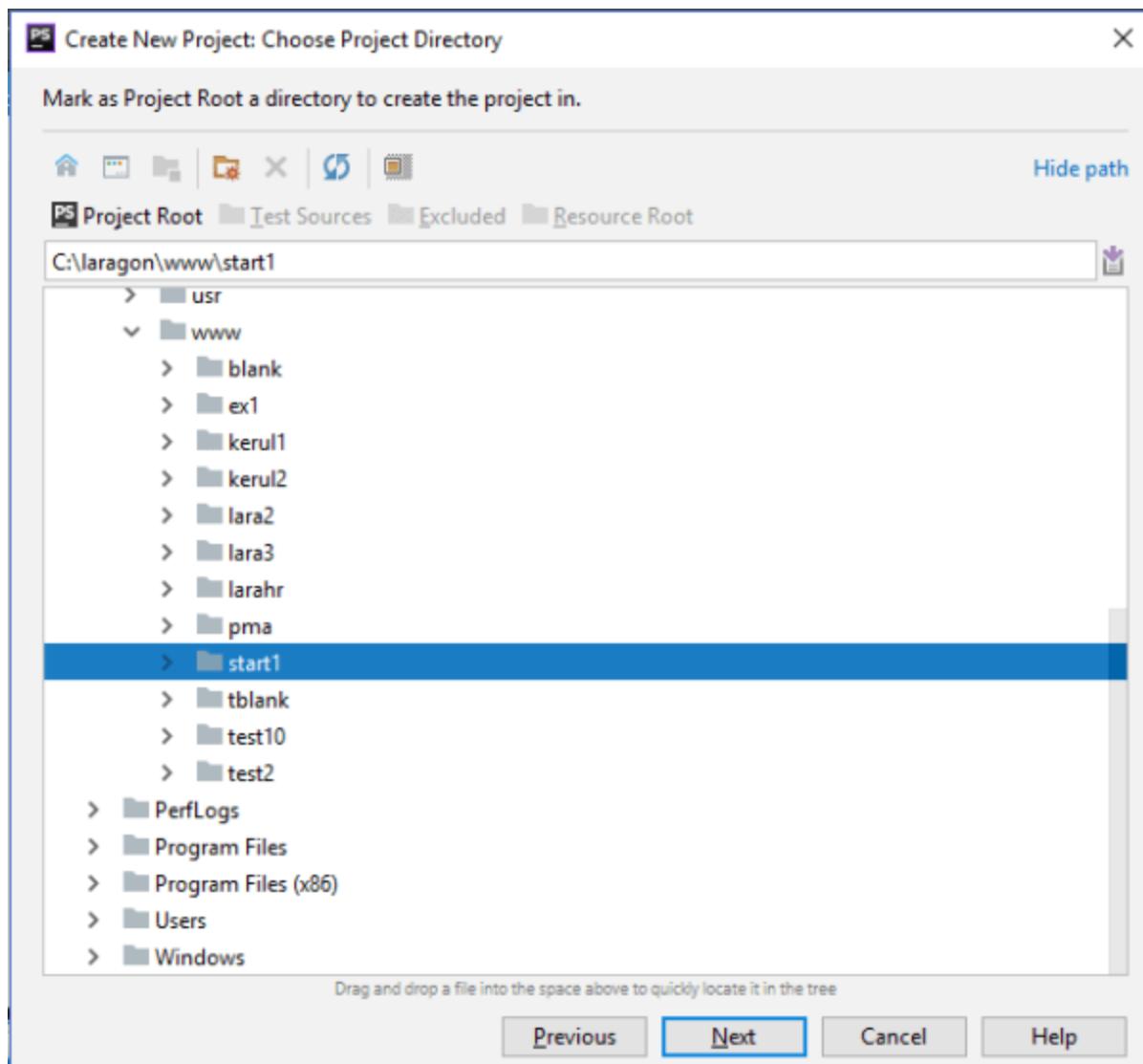
Bina projek daripada folder yang dah diwujudkan dan *point* kepada folder projek PHP/Laravel. (sekiranya anda ingin tahu cara membuat projek baru Laravel sila ke link ini <http://fstm.kuis.edu.my/blog/laravel1>)



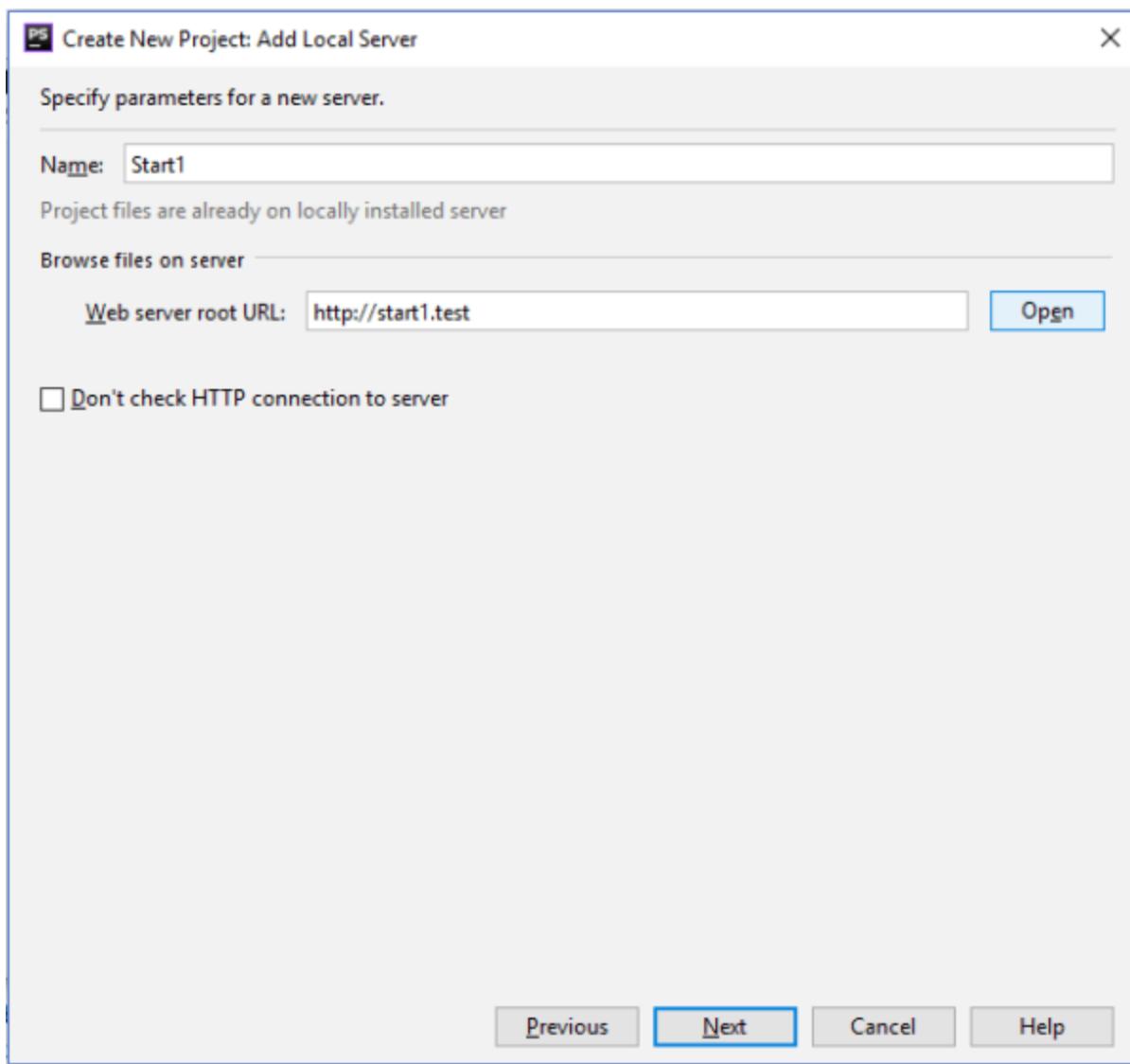
Set lokasi fail dan server (persekitaran *localhost*).



Tentukan lokasi fail projek Laravel anda.

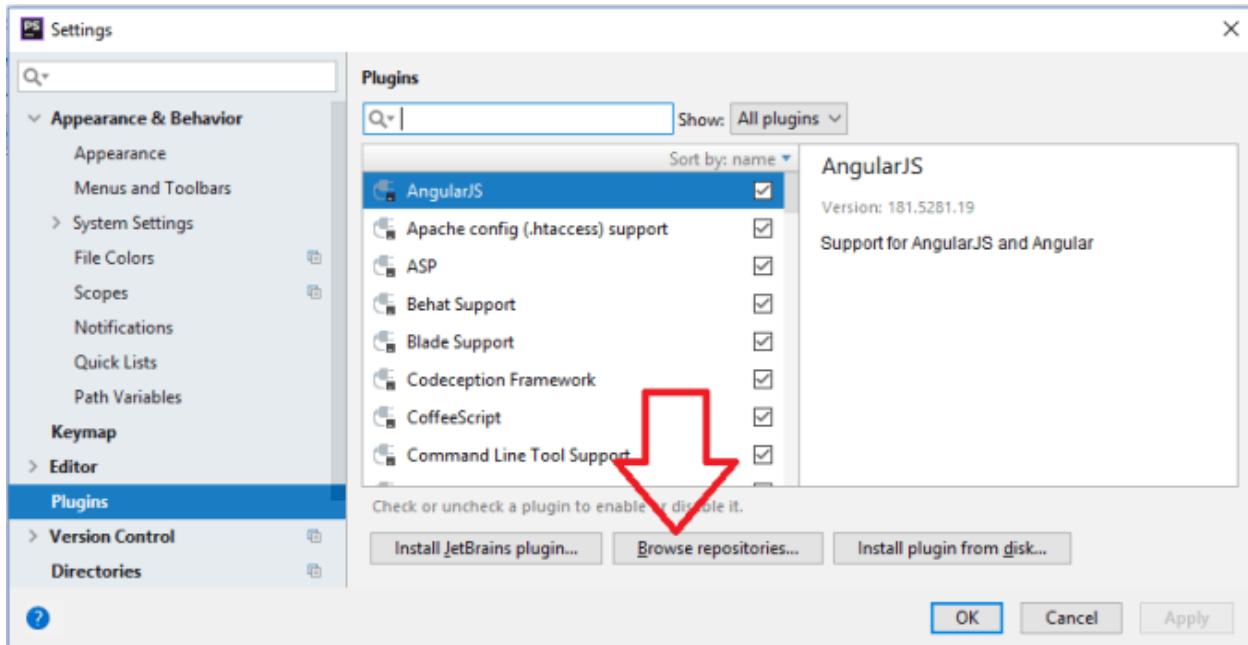


Tentukan maklumat web server (seperti yang disediakan oleh Laragon). Klik NEXT dan kemudian FINISH.

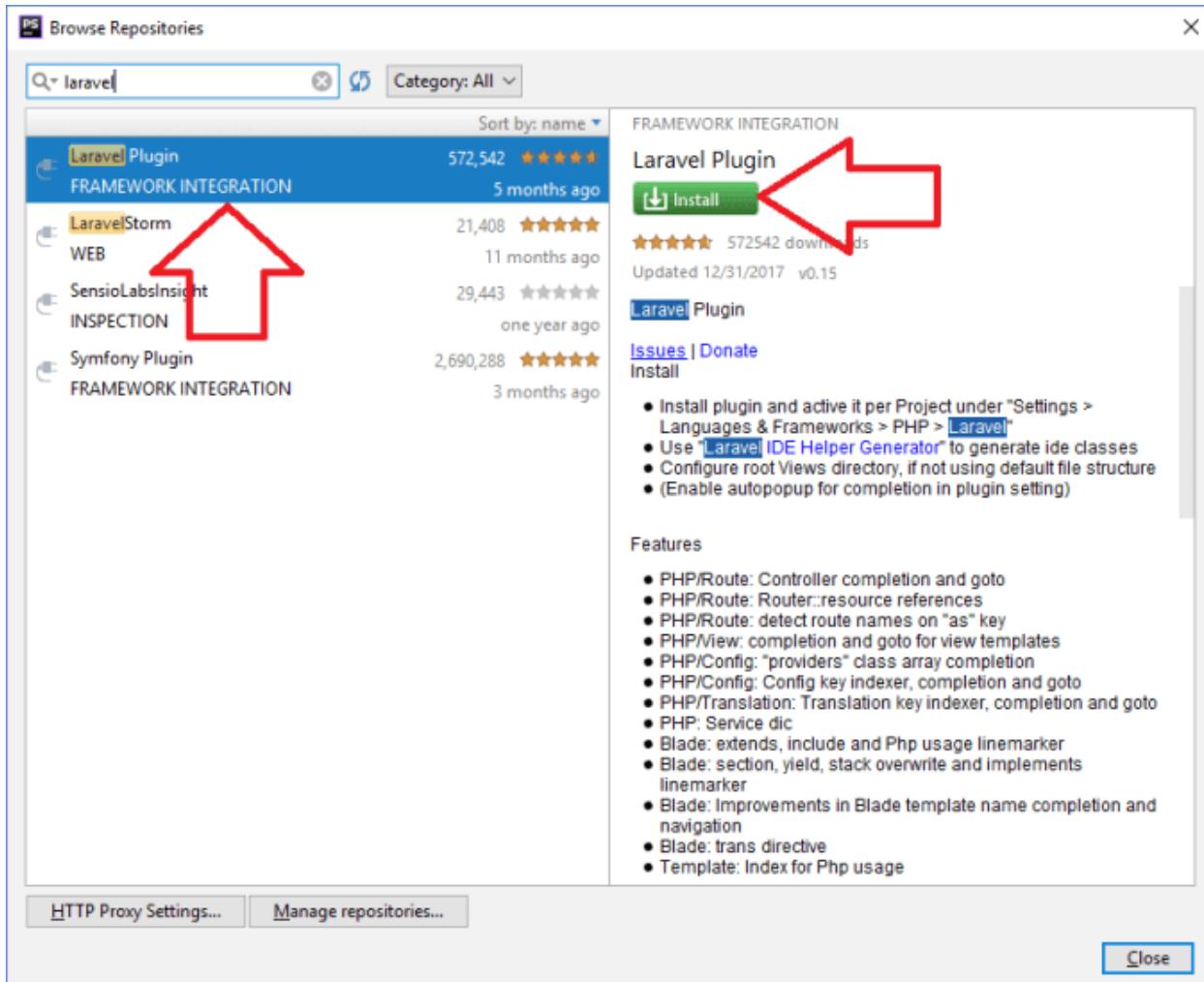


Pengurusan kod Laravel dalam PHPStorm

Untuk install plug-ins Laravel, pergi ke menu *File->Settings->Plugins* (rujuk gambar di bawah).

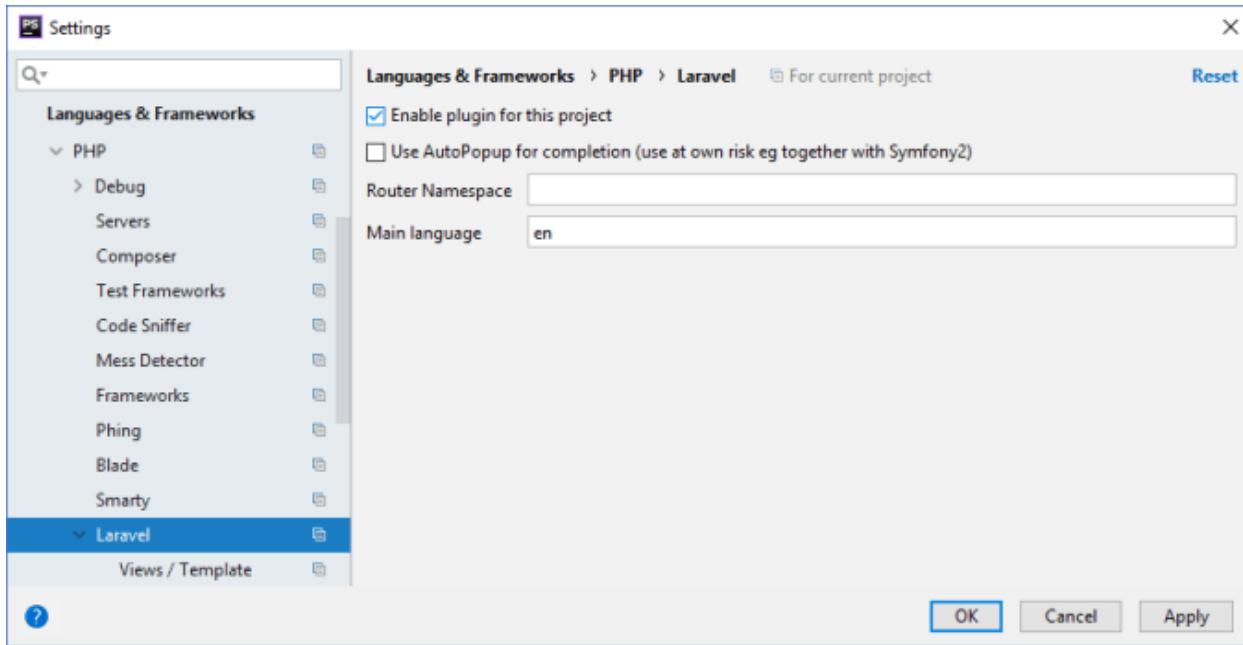


Seterusnya carian Laravel, dan klik butang install seperti di bawah.



Pastikan restart PHPStorm untuk mengaktifkan plug-ins Laravel yang baru kita install.

Kemudian aktifkan plug-ins Laravel dalam setiap projek yang guna Laravel di *File->Settings->Languages&Frameworks->Laravel*



(Install plug-ins untuk Laravel boleh juga muat-turun dari sini untuk install secara manual <https://plugins.jetbrains.com/plugin/7532-laravel-plugin>)

Penulisan kod HTML/PHP/Laravel/Blade

Contoh *suggestion* dalam kod HTML/Bootstrap/PHP

```

72
73     <main class="py-4">
74         @yield('content')
75     </main>
76     </div>
77
78     <a href="Hello" class="btn btn-|">
79     </body>
80     </html>
81
82
83
84
85
86
87
88
89
90
91
92

```

Class	File Path
.btn-block	public\css\app.css
.btn-danger	public\css\app.css
.btn-dark	public\css\app.css
.btn-group	public\css\app.css
.btn-group-lg	public\css\app.css
.btn-group-sm	public\css\app.css
.btn-group-toggle	public\css\app.css
.btn-group-vertical	public\css\app.css
.btn-info	public\css\app.css
.btn-lg	public\css\app.css
.btn-light	public\css\app.css

Press Ctrl+. to choose the selected (or first) suggestion and insert a dot afterwards [»](#)

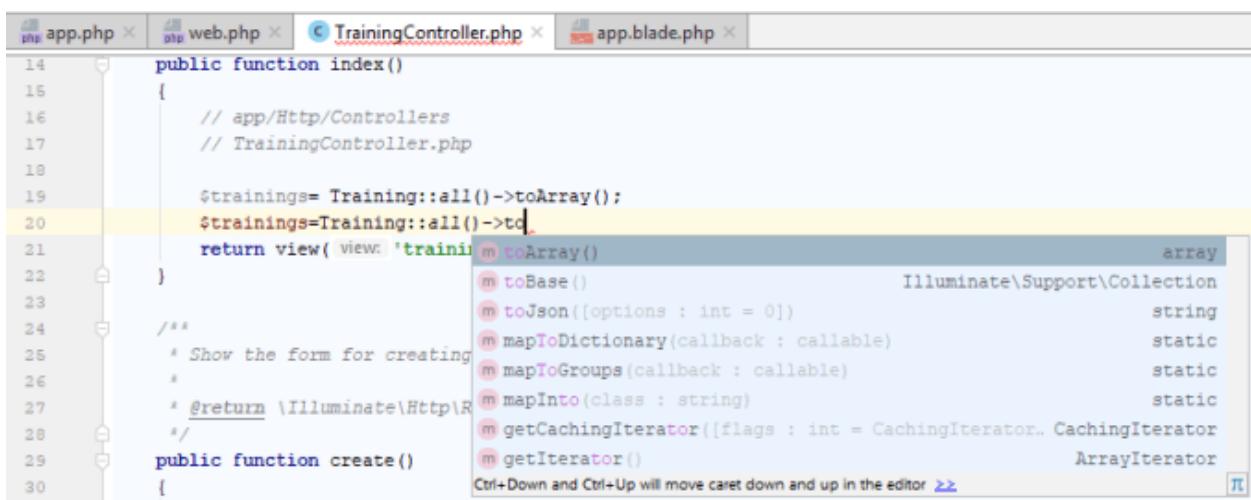
Contoh suggestion dalam kod Blade



A screenshot of a code editor showing a Blade template file. The cursor is positioned at the end of a line of code. A tooltip suggests several Blade directives: @else, @elseauth, @elsecan, @elsecannot, @elseguest, @elseif, @endforelse, and @forelse. The code itself includes navigation links and a logout button.

```
42     <!-- Authentication Links -->
43     @guest
44         <li><a class="nav-link" href="{{ route('login') }}>{{ __('Login') }}</a></li>
45         <li><a class="nav-link" href="{{ route('register') }}>{{ __('Register') }}</a>
46     @else
47         @else
48             @elseauth
49             @elsecan
50             @elsecannot
51             @elseguest
52             @elseif
53             @endforelse
54             @forelse
55
56             Press Ctrl+ to choose the selected (or first) suggestion and insert a dot afterwards ↗
57             document.getElementById('logout-form').submit();>
58             {{ __('Logout') }}
59         </a>
60     
```

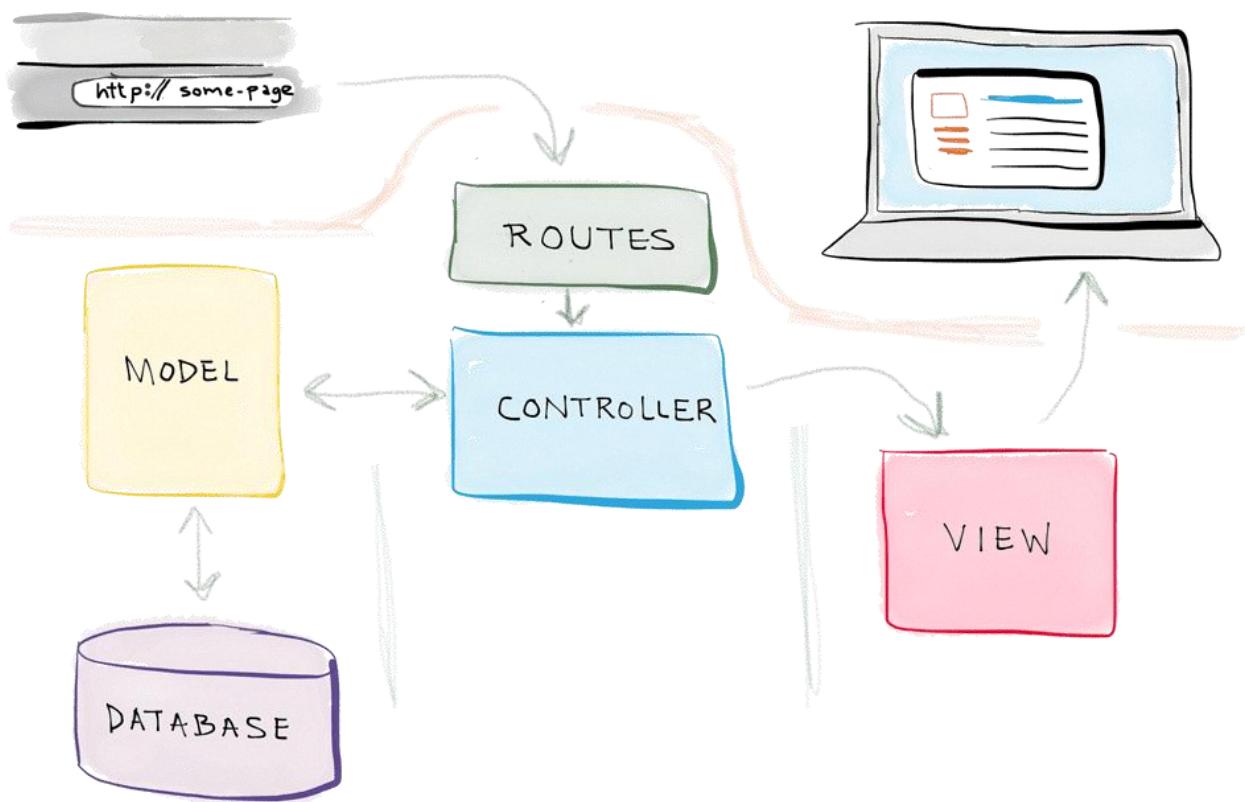
Contoh suggestion dalam kod Laravel



A screenshot of a code editor showing a Laravel controller file. The cursor is positioned at the end of a line of code. A tooltip suggests methods for a collection: toBase(), toJson(), mapToDictionary(), mapToGroups(), mapInto(), getCachingIterator(), and getIterator(). The code includes database queries and controller logic.

```
14     public function index()
15     {
16         // app/Http/Controllers
17         // TrainingController.php
18
19         $trainings= Training::all()->toArray();
20         $trainings=Training::all()->t
21         return view( view: 'trainin m toArray()
22     }
23
24     /**
25      * Show the form for creating
26      *
27      * @return \Illuminate\Http\R
28     */
29     public function create()
30     {
```

Tambahan: MVC dan Penulis



Laravel mengaplikasi konsep rekabentuk MVC.

Model – merujuk kepada capaian/query kepada pangkalan data dan mengembalikan data yang diminta.

Views – laman yang memaparkan data atau antaramuka.

Controllers – mengendalikan permintaan pengguna, capaian data kepada Model dan menghantar data kepada View.

(Sumber: Self Taught Coders)



Penulis tutorial:

KHIRULNIZAM ABD RAHMAN, Pensyarah Jabatan Sains Komputer, FSTM KUIS.

Beliau merupakan seorang trainer dalam bidang pengaturcaraan server dan antaramuka web (web front-end & backend) semenjak tahun 2000. Disamping itu juga amat berminat dalam pembangunan aplikasi mobile Android, JSON, LARAVEL dan PHP-MySQL.

Antara kursus yang beliau kendalikan;

- [Android Studio fstm.kuis.edu.my/blog/android](http://fstm.kuis.edu.my/blog/android)
- [Pembangunan Web dengan PHP&MySQL fstm.kuis.edu.my/blog/php](http://fstm.kuis.edu.my/blog/php)
- [Pembangunan Sistem Web menggunakan kerangka Laravel
Khirulnizam.com](http://khirulnizam.com)

Blog peribadi beliau di khirulnizam.com . Beliau boleh dihubungi melalui email khirulnizam@gmail.com , atau Whatsapp: <http://wasap.my/60129034614>