PEMROGRAMAN WEB LANJUT (PWL)

"Routing, Controller, dan View"

Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyo

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

Langkah-langkah Praktikum:

a. Pada bagian ini, kita akan membuat dua buah route dengan ketentuan sebagai berikut

No	Http Verb	Url	Fungsi
1	get	/hello	Tampilkan String Hello ke browser.
2	get	/world	Tampilkan String World ke browser

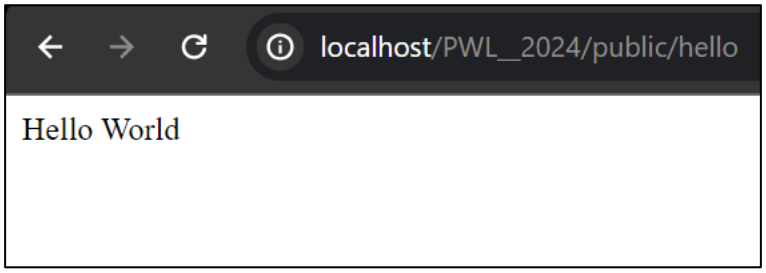
Kita akan menggunakan project minggu sebelumnya yaitu PWL_2024.

b. Buka file `routes/web.php`. Tambahkan sebuah route untuk nomor 1 seperti di bawahini:

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

Route::get('/hello', function () {
        return 'Hello World';
});
```

c. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/hello. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

Jawab:

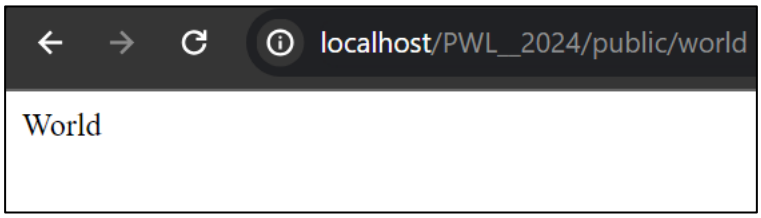


d. Untuk membuat route kedua, tambahkan route /world seperti di bawah ini:

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

Route::get('/world', function () {
        return 'World';
});
```

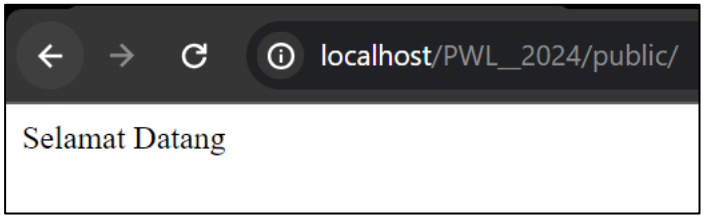
e. Bukalah pada browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/world. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

Jawab:



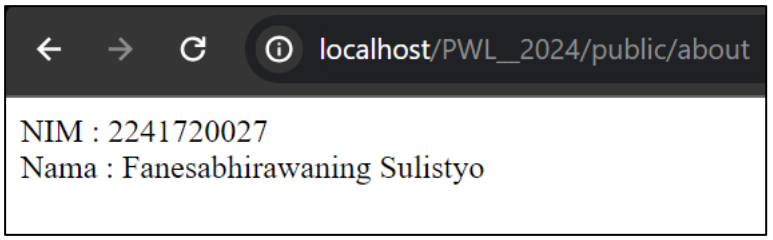
f. Selanjutnya, cobalah membuat route '/' yang menampilkan pesan 'Selamat Datang'.
Jawab:

```
Route::get('/', function () {
    return 'Selamat Datang';
});
```



g. Kemudian buatlah route '/about' yang akan menampilkan NIM dan nama Anda.
Jawab:

```
Route::get('/about', function () {
    return 'NIM : 2241720027 <br>
    Nama : Fanesabhirawaning Sulistyo';
});
```



- **Route Parameters**

Terkadang saat membuat sebuah URL, kita perlu mengambil sebuah parameter yang merupakan bagian dari segmen URL dalam route kita. Misalnya, kita membutuhkan namauser yang dikirim melalui sebuah URL.

Langkah-langkah Praktikum:

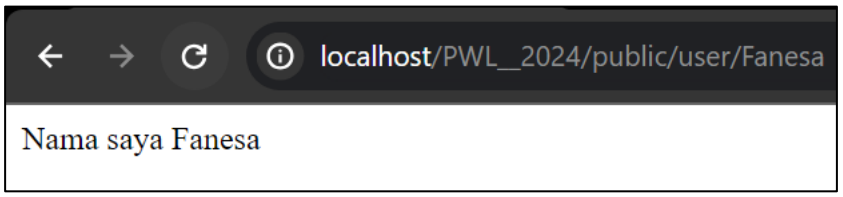
Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini

a. Kita akan memanggil route */user/{name}* sekaligus mengirimkan parameter berupanama user $name seperti kode di bawah ini.

```
Route::get('/user/{name}', function ($name) {
    return 'Nama saya '.$name;
});
```

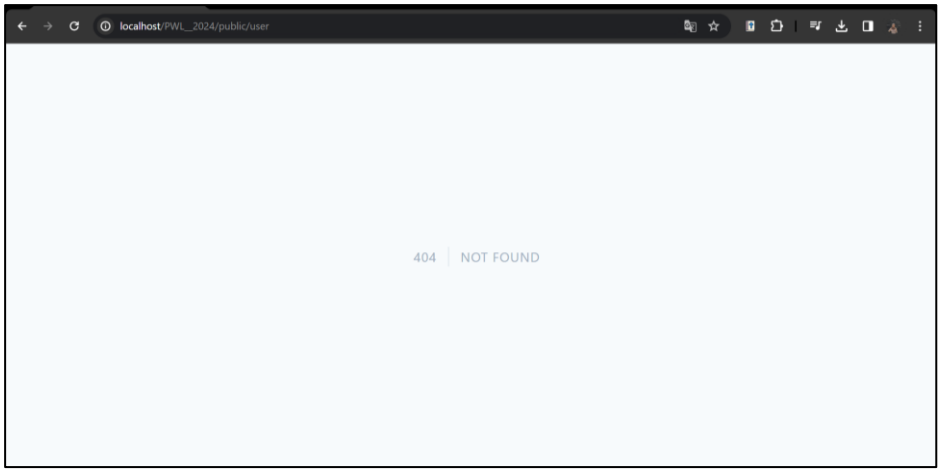
b. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/user/NamaAnda**. Perhatikan halaman yang muncul danjelaskan pengamatan Anda.

Jawab:



c. Selanjutnya, coba tuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

Jawab:

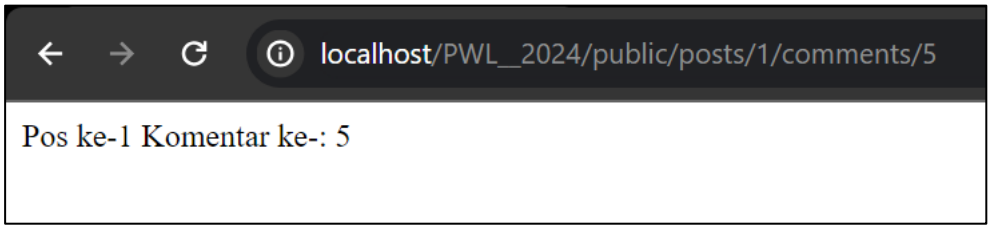


d. Suatu route, juga bisa menerima lebih dari 1 parameter seperti kode berikut ini. Route menerima parameter $postId dan juga $comment.

```
Route::get('/posts/{post}/comments/{comment}', function
($postId, $commentId) {
    return 'Pos ke-'.$postId." Komentar ke-: ".$commentId;
});
```

e. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/posts/1/comments/5**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

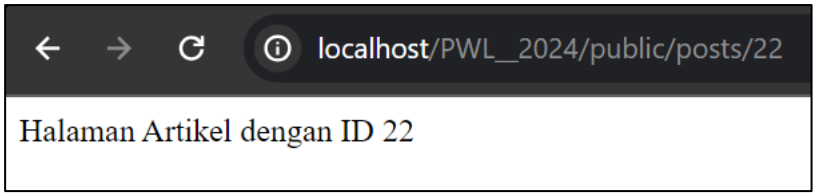
Jawab:



f. Kemudian buatlah route /articles/{id} yang akan menampilkan output “Halaman Artikeldengan ID {id}”, ganti id sesuai dengan input dari url.

Jawab:

```
Route::get('/posts/{id}', function ($id) {
    return 'Halaman Artikel dengan ID '.$id;
    });
```



- **Optional Parameters**

Kita dapat menentukan nilai parameter route, tetapi menjadikan nilai parameter route tersebut opsional. Pastikan untuk memberikan variabel yang sesuai pada route sebagai nilaidefault. Parameter opsional diberikan tanda ‘?’.

Langkah-langkah Praktikum:

Untuk membuat routing dengan optional parameter dapat dilakukan dengan cara berikut ini.

a. Kita akan memanggil route */user* sekaligus mengirimkan parameter berupa nama user $name dimana parameternya bersifat opsional.

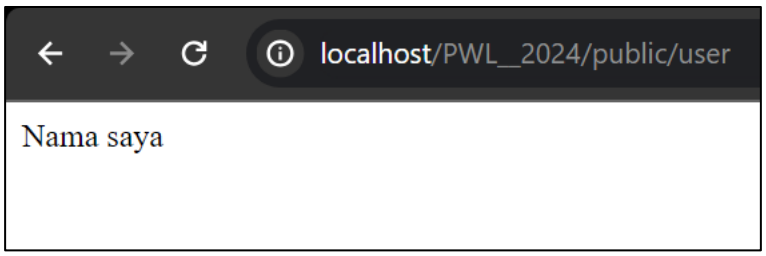
```
Route::get('/user/{name?}', function ($name=null) {
```

```
    return 'Nama saya '.$name;
});
```

b. Jalankan kode dengan menuliskan URL:

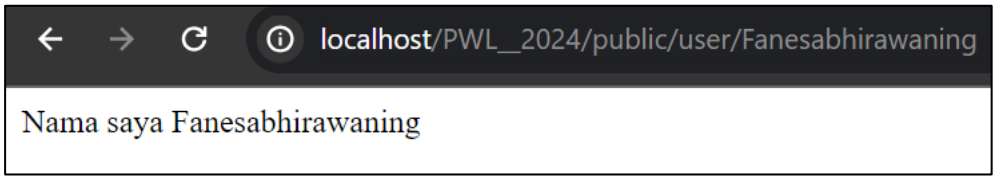
localhost/PWL_2024/public/user/.Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Jawab:



c. Selanjutnya tuliskan URL: **localhost/PWL_2024/public/user/NamaAnda**. Perhatikanhalaman yang muncul dan jelaskan pengamatan Anda

Jawab:



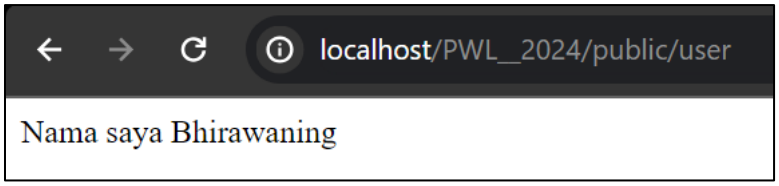
d. Ubah kode pada route /user menjadi seperti di bawah ini.

```
Route::get('/user/{name?}', function ($name='John') {
    return 'Nama saya '.$name;
});
```

e. Jalankan kode dengan menuliskan URL:

localhost/PWL_2024/public/user/.Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Jawab:



- **Route Name**

  Route name biasanya digunakan untuk mempermudah kita dalam pemanggilan route saat membangun aplikasi. Kita cukup memanggil name dari route tersebut.

```
Route::get('/user/profile', function () {
    //
})->name('profile');

Route::get(
    '/user/profile',
    [UserProfileController::class, 'show']
)->name('profile');

// Generating URLs...
$url = route('profile');

// Generating Redirects...
return redirect()->route('profile');
```

- **Route Group dan Route Prefixes**

  Beberapa route yang memiliki atribut yang sama seperti middleware yang sama dapat dikelompokkan menjadi satu kelompok untuk mempermudah penulisan route selain digunakan untuk middleware masih ada lagi penggunaan route group untuk route yang berada dibawah satu subdomain. Contoh penggunaan route group adalah sebagai berikut:

```
Route::middleware(['first', 'second'])->group(function () {
    Route::get('/', function () {
        // Uses first & second middleware...
    });

    Route::get('/user/profile', function () {
        // Uses first & second middleware...
    });
});

Route::domain('{account}.example.com')->group(function () {
    Route::get('user/{id}', function ($account, $id) {
        //
    });
});

Route::middleware('auth')->group(function () {
     Route::get('/user', [UserController::class, 'index']);
     Route::get('/post', [PostController::class, 'index']);
     Route::get('/event', [EventController::class, 'index']);

});
```

- Route Prefixes

Pengelompokan route juga dapat dilakukan untuk route yang memiliki prefix (awalan)yang sama. Untuk pembuatan route dengan prefix dapat dilihat kode seperti di bawah ini

```
Route::prefix('admin')->group(function () {
     Route::get('/user', [UserController::class, 'index']);
     Route::get('/post', [PostController::class, 'index']);
     Route::get('/event', [EventController::class, 'index']);

});
```

- **Redirect Routes**

Untuk melakukan redirect pada laravel dapat dilakukan dengan menggunakanRoute::redirect cara penggunaannya dapat dilihat pada kode program dibawah ini.

```
Route::redirect('/here', '/there');
```

Redirect ini akan sering digunakan pada kasus kasus CRUD atau kasus lain yang membutuhkan redirect.

- **View Routes**

Laravel juga menyediakan sebuah route khusus yang memudahkan dalam membuat sebuah routes tanpa menggunakan controller atau callback function. Routes ini langsung menerimainput berupa url dan mengembalikan view / tampilan. Berikut ini cara membuat view routes.

```
Route::view('/welcome', 'welcome');
Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

Pada view routes diatas /welcome akan menampilkan view welcome dan pada route kedua /welcome akan menampilkan view welcome dengan tambahan data berupa variabel name.

Simpan perubahan yang telah dilakukan pada Git.

1. Controller

Controller digunakan untuk mengorganisasi logika aplikasi menjadi lebih terstruktur. Logika action aplikasi yang masih ada kaitan dapat dikumpulkan dalam satu kelas Controller. Atau sebuah Controller dapat juga hanya berisi satu buah action. Controller pada Laravel disimpan dalam folder `app/Http/Controllers`.

- **Membuat Controller**

Langkah-langkah Praktikum:

a. Untuk membuat controller pada Laravel telah disediakan perintah untuk menggenerate struktur dasarnya. Kita dapat menggunakan perintah artisan diikuti dengan definisi nama controller yang akan dibuat.

```
php artisan make:controller WelcomeController
```

b. Buka file pada `app/Http/Controllers/WelcomeController.php`. Strukturpada controller dapat digambarkan sebagai berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
     //
}
```

c. Untuk mendefinisikan action, silahkan tambahkan function dengan access public. Sehinggacontroller di atas menjadi sebagai berikut:

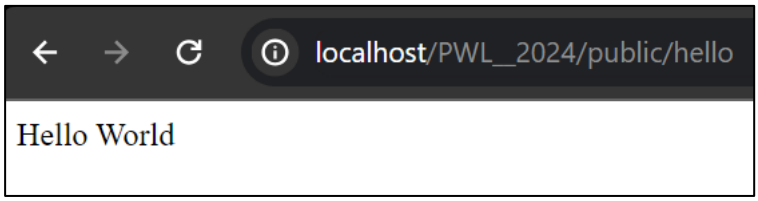
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
     public function hello() {
               return 'Hello World';
     }
}
```

d. Setelah sebuah controller telah didefinisikan action, kita dapat menambahkan controllertersebut pada route. Ubah route /hello menjadi seperti berikut:

```
Route::get('/hello', [WelcomeController::class,'hello']);
```

e. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/hello. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Jawab:



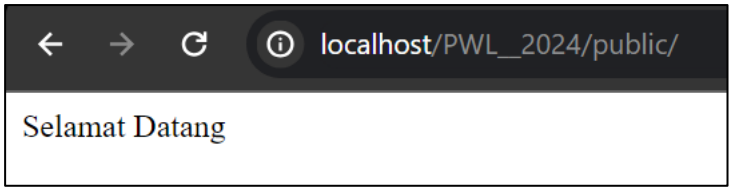
f. Modifikasi hasil pada praktikum poin 2 (Routing) dengan konsep controller. Pindahkan logika eksekusi ke dalam controller dengan nama PageController.

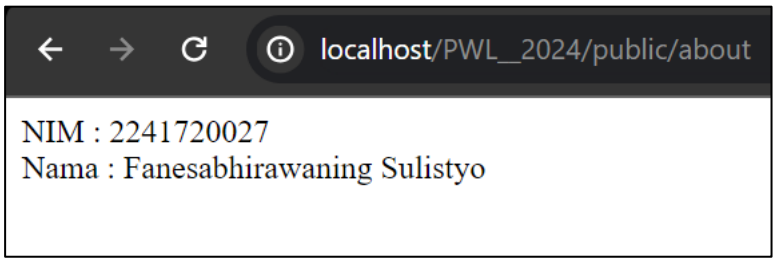
Resource	POST	GET	PUT	DELETE
/		Tampilkan Pesan 'Selamat Datang' `PageController : index`		
/about		Tampilkan Nama dan NIM `PageController : about`		
/articles/{id}		Tampilkan halaman dinamis 'Halaman Artikel dengan Id {id}' id diganti sesuai input dari url `PageController : articles`		

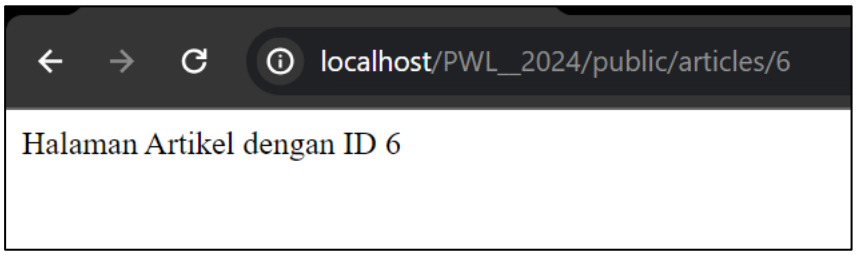
Jawab:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function index() {
        return 'Selamat Datang';
        }
    public function about() {
        return 'NIM : 2241720027 <br>
        Nama : Fanesabhirawaning Sulistyo';
        }
    public function articles($id) {
        return 'Halaman Artikel dengan ID '.$id;
        }
}
```

```php
Route::get('/', [PageController::class, 'index']);
Route::get('/about', [PageController::class, 'about']);
Route::get('/articles/{id}', [PageController::class, 'articles']);
```







g. Modifikasi kembali implementasi sebelumnya dengan konsep Single Action Controller. Sehingga untuk hasil akhir yang didapatkan akan ada `HomeController`, `AboutController` dan `ArticleController`. Modifikasi juga route yang digunakan.

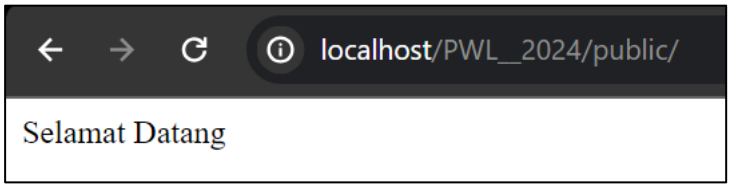
Jawab:

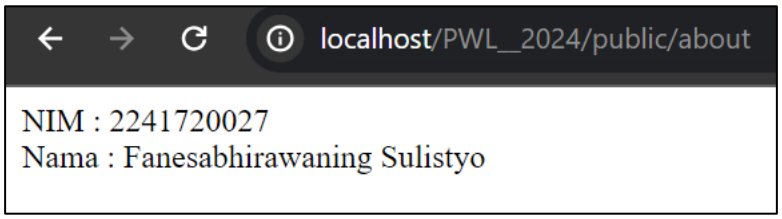
```php
class HomeController extends Controller
{
    public function index() {
        return 'Selamat Datang';
        }
}
```

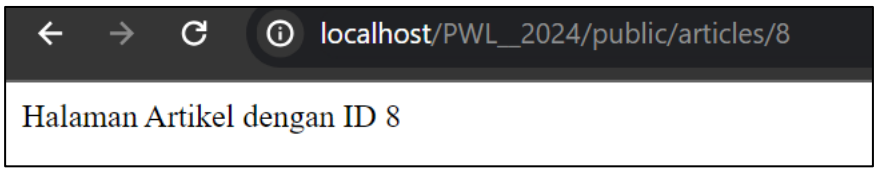
```php
class AboutController extends Controller
{
    public function about() {
        return 'NIM : 2241720027 <br>
        Nama : Fanesabhirawaning Sulistyo';
        }
}
```

```php
class ArticleController extends Controller
{
    public function articles($id) {
        return 'Halaman Artikel dengan ID '.$id;
    }
}
```

```
Route::get('/', [HomeController::class, 'index']);
Route::get('/about', [AboutController::class, 'about']);
Route::get('/articles/{id}', [ArticleController::class, 'articles']);
```







- **Resource Controller**

  Khusus untuk controller yang terhubung dengan Eloquent model dan dapat dilakukan operasi CRUD terhadap model Eloquent tersebut, kita dapat membuat sebuah controller yang bertipe Resource Controller. Dengan membuat sebuah resource controller, maka controller tersebut telah dilengkapi dengan method-method yang mendukung proses CRUD, serta terdapat sebuah route resource yang menampung route untuk controller tersebut.

Langkah-langkah Praktikum:

a. Untuk membuatnya dilakukan dengan menjalankan perintah berikut ini di terminal.

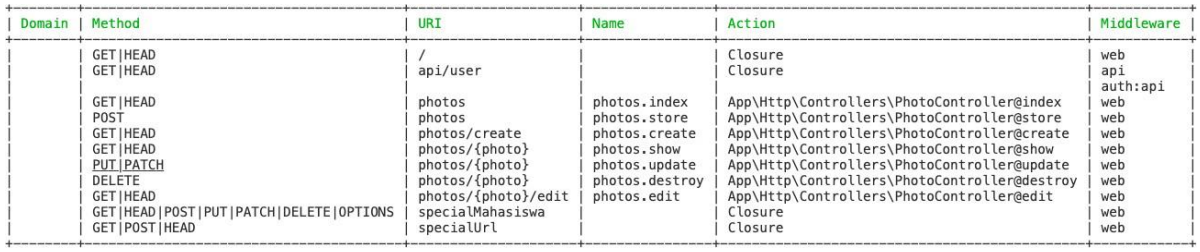
```
php artisan make:controller PhotoController --resource
```

Perintah ini akan men generate sebuah controller dengan nama PhotoController yang berisi method method standar untuk proses CRUD.

b. Setelah controller berhasil degenerate, selanjutnya harus dibuatkan route agar dapat terhubung dengan frontend. Tambahkan kode program berikut pada file web.php.

```
use App\Http\Controllers\PhotoController;

Route::resource('photos', PhotoController::class);
```

c. Jalankan cek list route (php artisan route:list) akan dihasilkan route berikut ini.



Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	api/user		Closure	api
					auth:api
	GET|HEAD	photos	photos.index	App\Http\Controllers\PhotoController@index	web
	POST	photos	photos.store	App\Http\Controllers\PhotoController@store	web
	GET|HEAD	photos/create	photos.create	App\Http\Controllers\PhotoController@create	web
	GET|HEAD	photos/{photo}	photos.show	App\Http\Controllers\PhotoController@show	web
	PUT|PATCH	photos/{photo}	photos.update	App\Http\Controllers\PhotoController@update	web
	DELETE	photos/{photo}	photos.destroy	App\Http\Controllers\PhotoController@destroy	web
	GET|HEAD	photos/{photo}/edit	photos.edit	App\Http\Controllers\PhotoController@edit	web
	GET|HEAD|POST|PUT|PATCH|DELETE|OPTIONS	specialMahasiswa		Closure	web
	GET|POST|HEAD	specialUrl		Closure	web

d. Pada route list semua route yang berhubungan untuk crud photo sudah di generate oleh laravel. Jika tidak semua route pada resource controller dibutuhkan dapat dikurangi denganmengupdate route pada web.php menjadi seperti berikut ini.

```
Route::resource('photos', PhotoController::class)->only([
    'index', 'show'
]);

Route::resource('photos', PhotoController::class)->except([
    'create', 'store', 'update', 'destroy'
]);
```

Simpan perubahan yang telah dilakukan pada Git.

2. View

Dalam kerangka kerja Laravel, View merujuk pada bagian dari aplikasi web yangbertanggung jawab untuk menampilkan antarmuka pengguna kepada pengguna akhir. View pada dasarnya adalah file template yang digunakan untuk menghasilkan HTML yang akan ditampilkan kepada pengguna.

Blade merupakan templating engine bawaan Laravel. Berguna untuk mempermudah dalam menulis kode tampilan. Dan juga memberikan fitur tambahan untuk memanipulasi data di view yang dilempar dari controller. Blade juga memungkinkan penggunaan plain PHP pada kode View. Karena Laravel menggunakan *templating engine* bawaan Blade, maka setiap *file* View diakhiri dengan .blade.php. Misal: index.blade.php, home.blade.php, product.blade.php.

- Membuat View

Langkah-langkah Praktikum:

a. Pada direktori app/resources/views, buatlah file hello.blade.php.

```
<!-- View pada resources/views/hello.blade.php -->
<html>
     <body>
     <h1>Hello, {{ $name }}</h1>
     </body>
</html>
```

b. View tersebut dapat dijalankan melalui Routing, dimana *route* akan memanggil Viewsesuai dengan nama *file* tanpa ‘blade.php’. (Catatan: Gantilah Andi dengan nama Anda)

```
Route::get('/greeting', function () {
    return view('hello', ['name' => 'Andi']);
});
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikanhalaman yang muncul dan jelaskan pengamatan Anda.

Jawab:

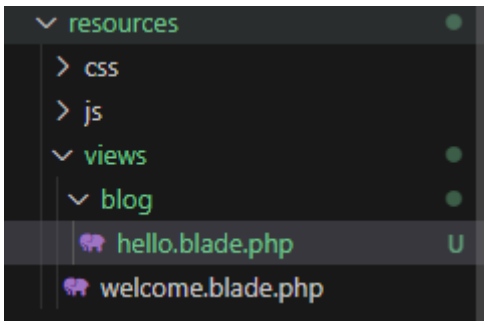
- **View dalam direktori**

  Jika di dalam direktori `resources/views` terdapat direktori lagi untuk menyimpan *file* view, sebagai contoh hello.blade.php ada di dalam direktori blog, maka kita bisa menggunakan “dot”notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

a. Buatlah direktori blog di dalam direktori views.

b. Pindahkan file hello.blade.php ke dalam direktori blog.



c. Selanjutnya lakukan perubahan pada route.

```
Route::get('/greeting', function () {
    return view('blog.hello', ['name' => 'Andi']);
});
```

d. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikanhalaman yang muncul dan jelaskan pengamatan Anda.

Jawab:

- **Menampilkan View dari Controller**

View dapat dipanggil melalui Controller. Sehingga Routing akan memanggil Controllerterlebih dahulu, dan Controller akan me-*return* view yang dimaksud.

Langkah-langkah Praktikum:

a. Buka WelcomeController.php dan tambahkan fungsi baru yaitu greeting.

```
class WelcomeController extends Controller
{
    public function hello(){
        return('Hello World');
    }

    public function greeting(){
        return view('blog.hello', ['name' => 'Andi']);
    }
}
```

b. Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting.

```
Route::get('/greeting', [WelcomeController::class,
'greeting']);
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikanhalaman yang muncul dan jelaskan pengamatan Anda.

Jawab:

- **Meneruskan data ke view**

  Pada contoh sebelumnya, kita dapat meneruskan data array ke view agar data tersebut tersediauntuk view:

```
return view('blog.hello', ['name' => 'Andi']);
```

  Saat meneruskan informasi dengan cara ini, data harus berupa array dengan pasangan kunci / nilai. Setelah memberikan data ke view, kemudian kita dapat mengakses setiap nilai dalam viewmenggunakan kunci data seperti: `<?php echo $name; ?>` atau `{{$name}}`. Sebagai alternatif untuk meneruskan array data lengkap ke fungsi view helper, kita dapat menggunakan metode **with** untuk menambahkan bagian data individual ke view. Metode **with** mengembalikan instance view objek sehingga kita dapat melanjutkan rangkaian metode sebelum mengembalikan tampilan notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

a. Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

```
class WelcomeController extends Controller
{
    public function hello(){
        return('Hello World');
    }

    public function greeting(){
        return view('blog.hello')
    ->with('name','Andi')
    -
        >with('occupation','Astronau
        t');
    }
    }
```

b. Ubah hello.blade.php agar dapat menampilkan dua parameter.

```
<html>
     <body>
     <h1>Hello, {{ $name }}</h1>
     <h1>You are {{ $occupation }}</h1>
     </body>
</html>
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikanhalaman yang muncul dan jelaskan pengamatan Anda.

Jawab:

Simpan perubahan yang telah dilakukan pada Git.

SOAL PRAKTIKUM

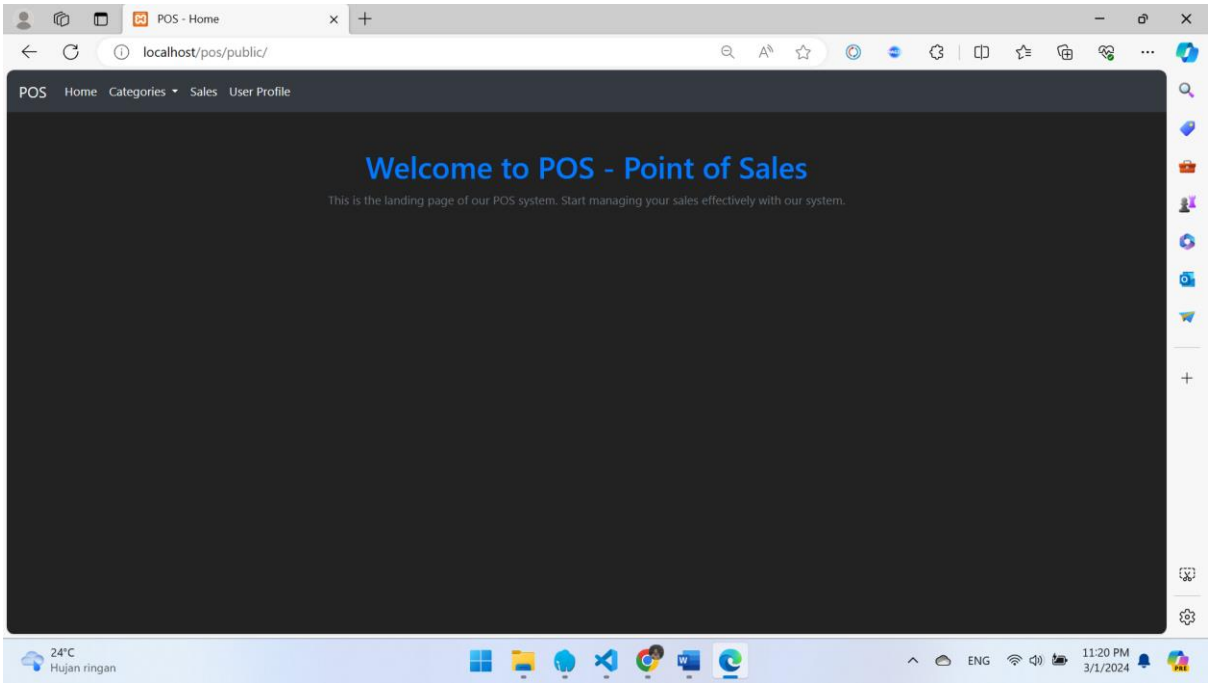
1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasiperubahan pada project PWL_2024 ke Github.
2. Buatlah project baru dengan nama POS. Project ini merupakan sebuah aplikasi Point ofSales yang digunakan untuk membantu penjualan.
3. Buatlah beberapa route, controller, dan view sesuai dengan ketentuan sebagai berikut.

1	Halaman Home Menampilkan halaman awal website
2	Halaman Products Menampilkan daftar product (route prefix) /category/food-beverage /category/beauty-health /category/home-care /category/baby-kid
3	Halaman User Menampilkan profil pengguna (route param) /user/{id}/name/{name}
4	Halaman Penjualan Menampilkan halaman transaksi POS

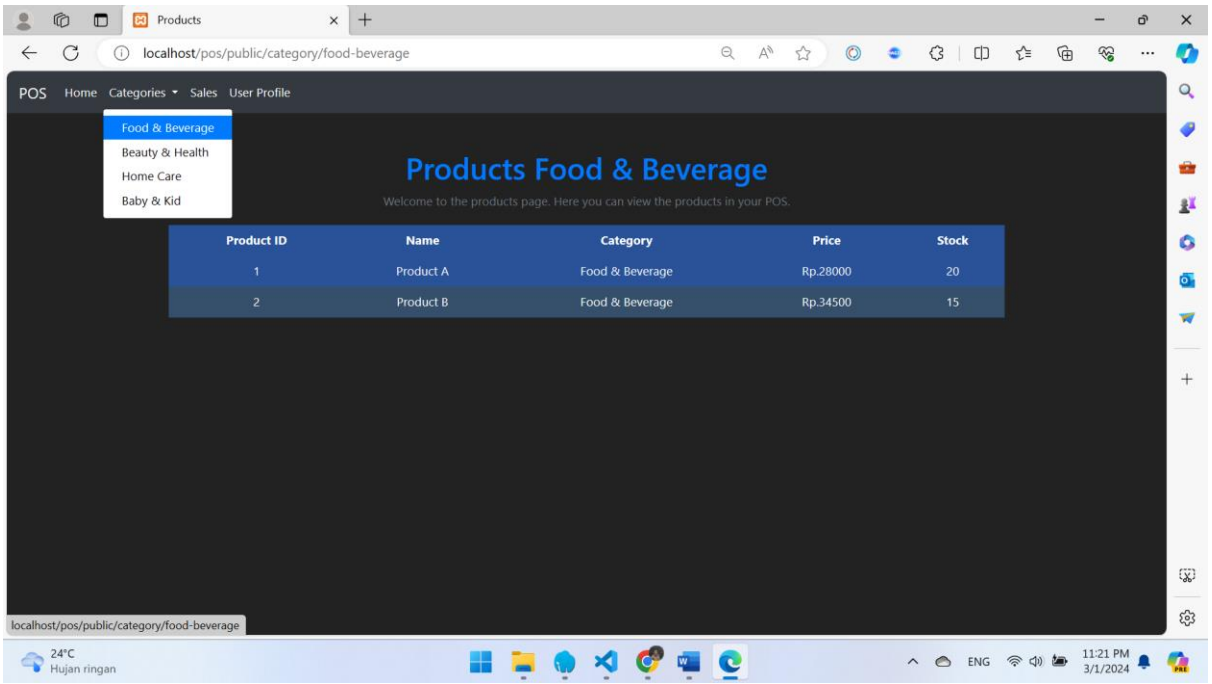
4. Route tersebut menjalankan fungsi pada Controller yang berbeda di setiap halaman.
5. Fungsi pada Controller akan memanggil view sesuai halaman yang akan ditampilkan.
6. Simpan setiap perubahan yang dilakukan pada project POS pada Git, sinkronisasi perubahan ke Github.

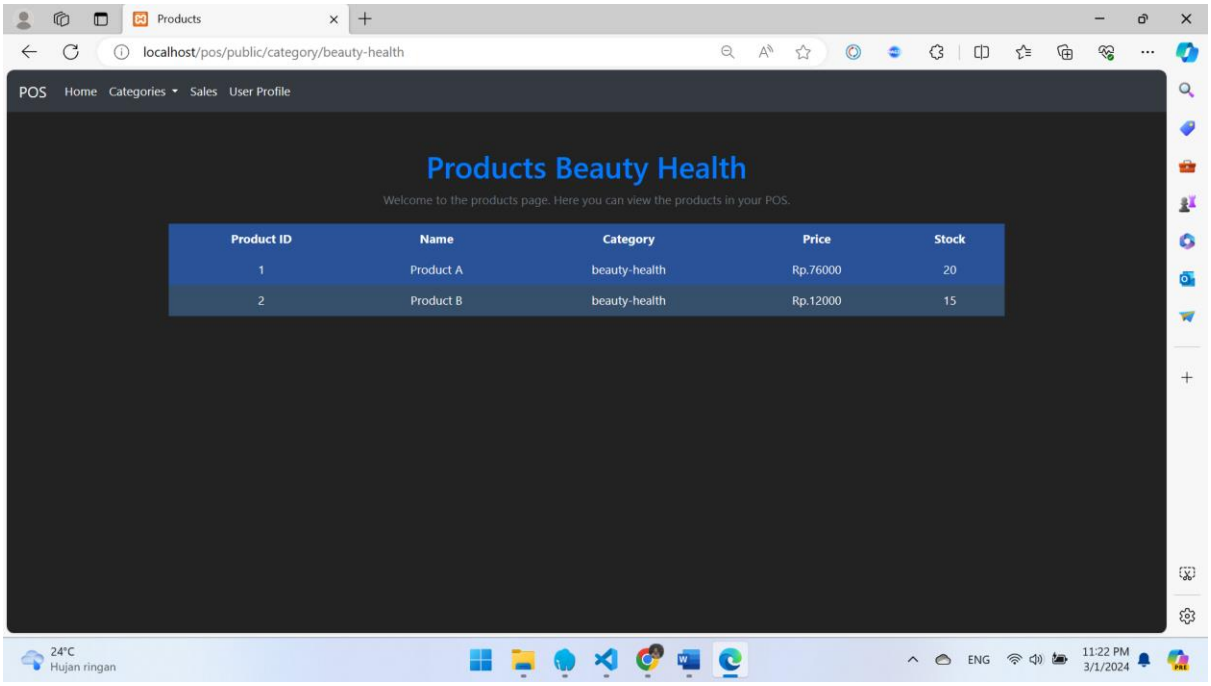
Jawab:

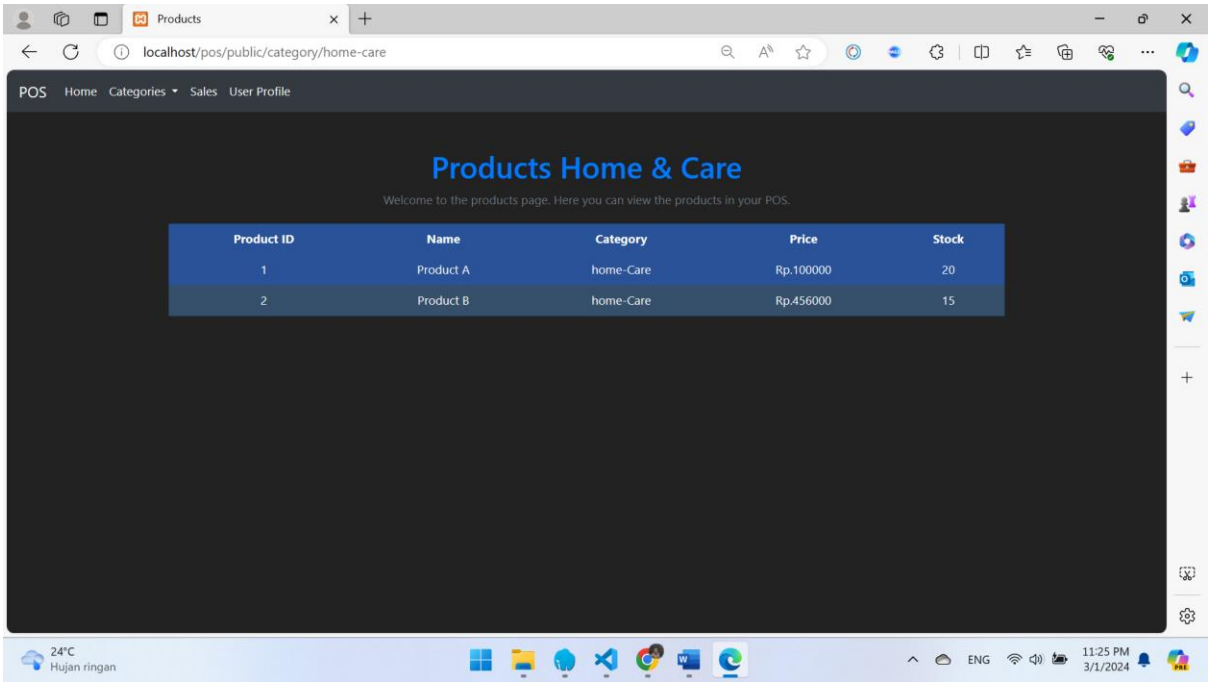
- Halaman Home

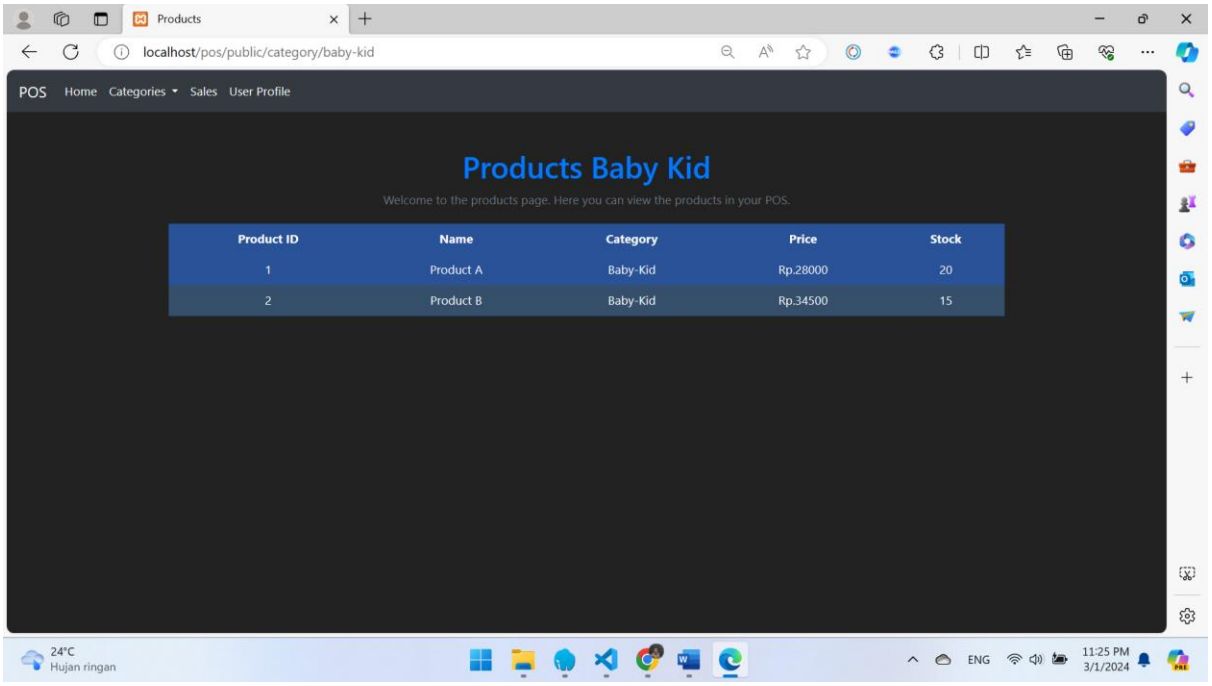


- Halaman Products

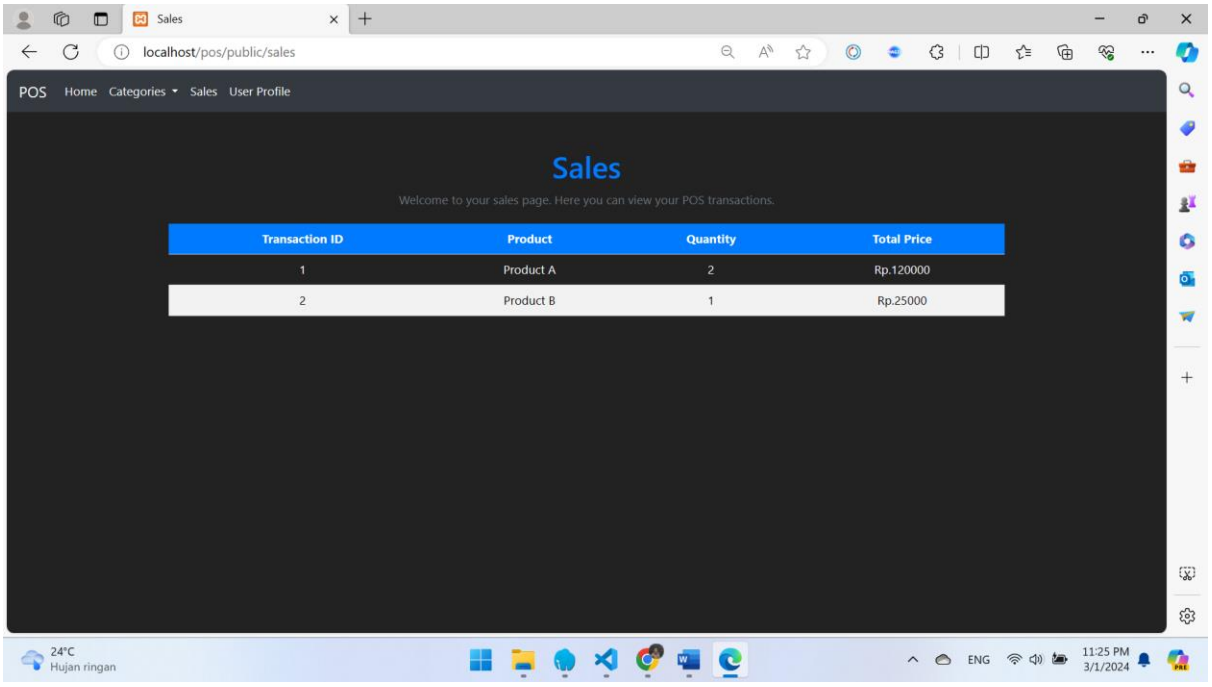



Products
localhost/pos/public/category/beauty-health
POS
Home
Categories
Sales
User Profile
Products Beauty Health
Welcome to the products page. Here you can view the products in your POS.
Product ID
Name
Category
Price
Stock
1
Product A
beauty-health
Rp.76000
20
2
Product B
beauty-health
Rp.12000
15
24°C
Hujan ringan
ENG
11:22 PM
3/1/2024


Products
localhost/pos/public/category/home-care
POS
Home
Categories
Sales
User Profile
Products Home & Care
Welcome to the products page. Here you can view the products in your POS.
Product ID
Name
Category
Price
Stock
1
Product A
home-Care
Rp.100000
20
2
Product B
home-Care
Rp.456000
15
24°C
Hujan ringan
ENG
11:25 PM
3/1/2024



- Halaman Penjualan



- Halaman User