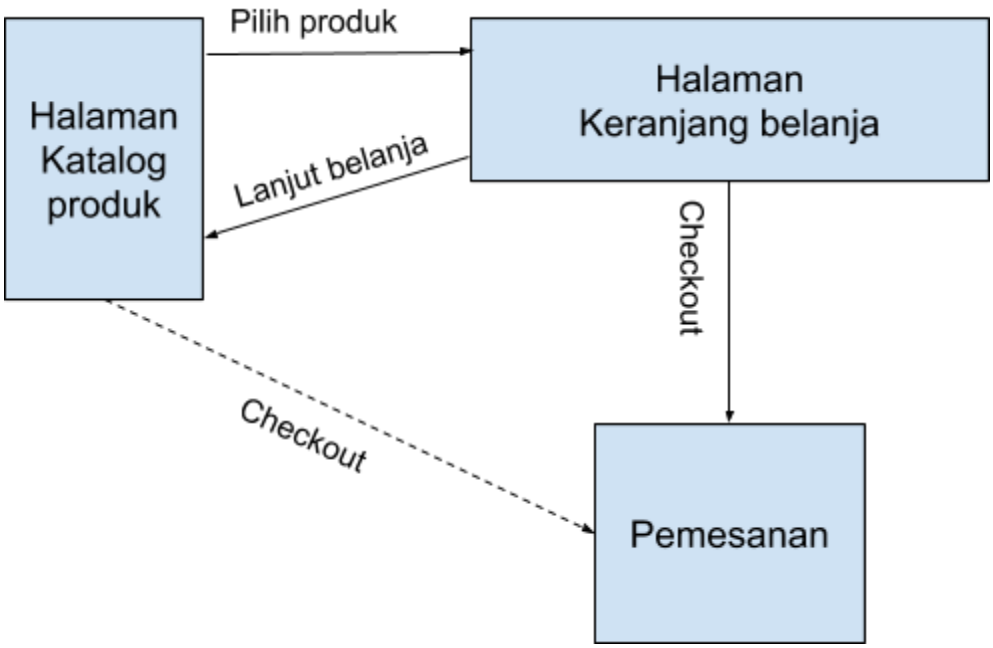
Materi Pertemuan 6 - Ecommerce Website

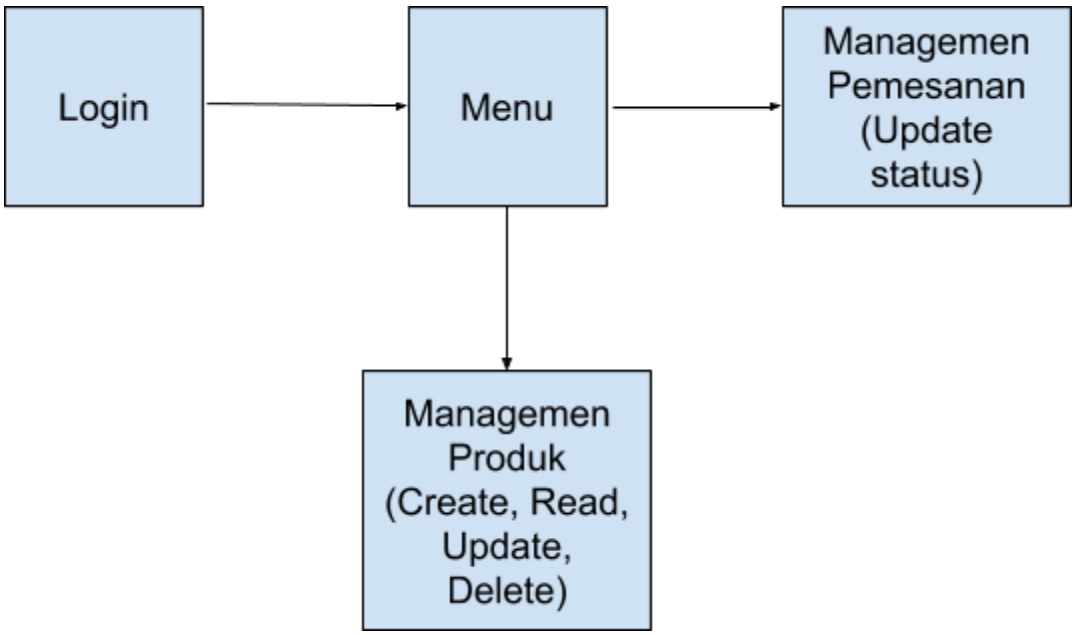
Use Case

1. Pertama kita mengidentifkasi ada dua role of actor, pembeli dan penjual
2. Pembeli bisa melihat produk, pilih beberapa produk untuk dibeli, kemudian meng-input informasi yang diperlukan untuk melakukan pemesanan
3. Penjual bisa menambah, mengedit atau mengurangi produk, menentukan apakah pemesanan telah di proses atau belum

Flow Diagram Pembeli



Flow Diagram Penjual



Database Table Design

1. users:
    - id
    - name
    - email
    - password
    - email_verification_token
    - email_verified_at
    - reset_password_token
    - created_at
    - updated_at
2. products
    - id
    - user_id
    - name

   - description
   - price
   - image_url
   - video_url
3. carts
   - id
   - user_id
   - created_at
   - updated_at
4. cart_items
   - id
   - cart_id
   - product_id
   - quantity
   - price
5. orders:
   - id
   - user_id
   - cart_id
   - total_price
   - status
   - created_at
   - updated_at
6. Next…

Laravel Project

1. Laravel yang digunakan versi 5.8.4, untuk yang berbeda versi bisa mnyesuaikan
2. Create laravel project
   composer create-project laravel/laravel ecommerce --prefer-dist

3. Create database dengan nama “ecommerce”
4. Setting up .env file
5. Untuk memastikan project berjalan, run server dan buka http://localhost:8000 di browser
   php artisan serve
6. Finish

Authentication

Bagian ini bisa kita lewati kalau kita sudah pernah bikin dan sudah mengerti.

1. Ketika membuat project laravel, secara default laravel akan men-generate controllers, models dan database migrations untuk kebutuhan authentication.
   Controller:
   - Auth/ForgotPasswordController.php
   - Auth/LoginController.php
   - Auth/RegisterController.php
   - Auth/ResetPasswordController.php
   - Auth/VerificationController.php

   Models:
   - User.php

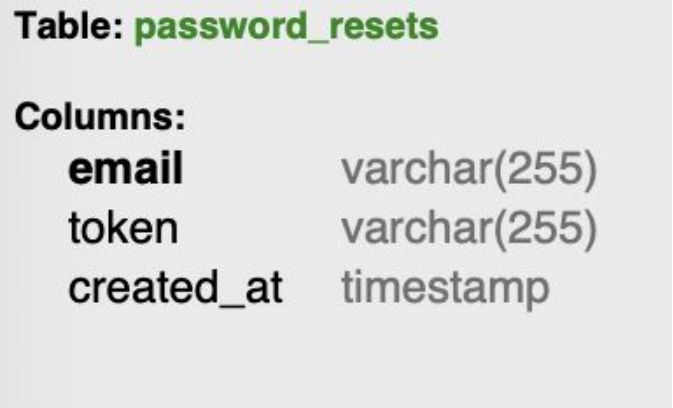
   Migrations:
   - ……_create_users_table.php
   - ….._create_password_resets_table.php

2. Jalankan migration
   php artisan migrate
3. Setelah migration kita akan melihat 2 tables dibawah ini

Table: users

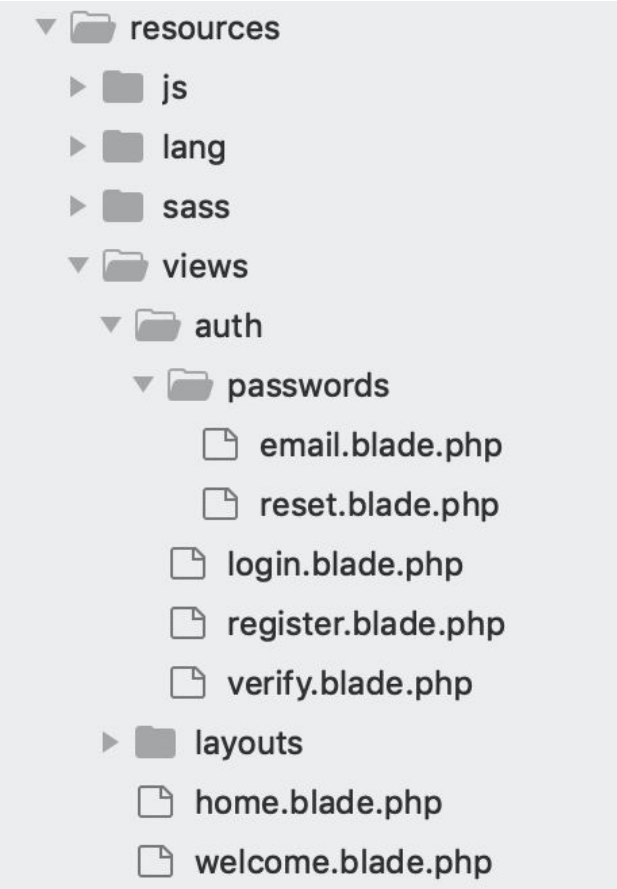
Columns:

id	bigint(20) UN AI PK
name	varchar(255)
email	varchar(255)
email_verified_at	timestamp
password	varchar(255)
remember_token	varchar(100)
created_at	timestamp
updated_at	timestamp



4. Kemudian kita akan mebuat halaman view untuk authentication, dengan menggunakan command auth scaffolding generator
   php artisan make:auth

5. Dari auth scaffolding generator tersebut, kita sudah dibuatkan views dan routes nya. Secara default laravel menggunakan bootstrap untuk framework css nya.
   Views:

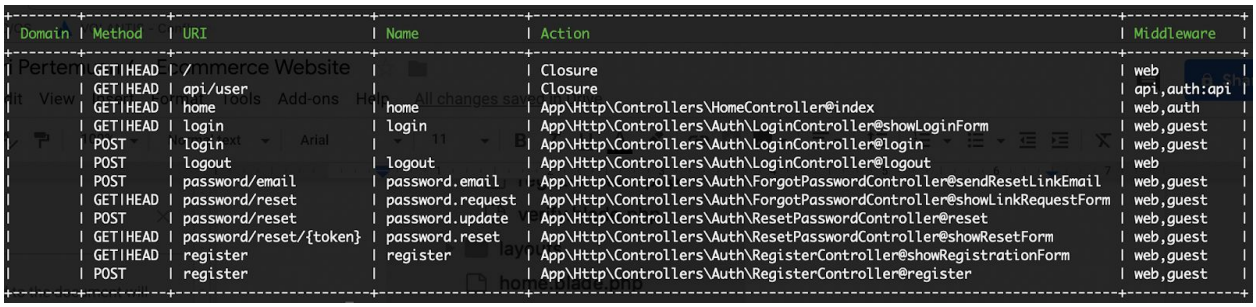


Routes:

```php
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

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

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

```

6. Untuk melihat list routing yang sudah di generate, jalankan command

php artisan route:list

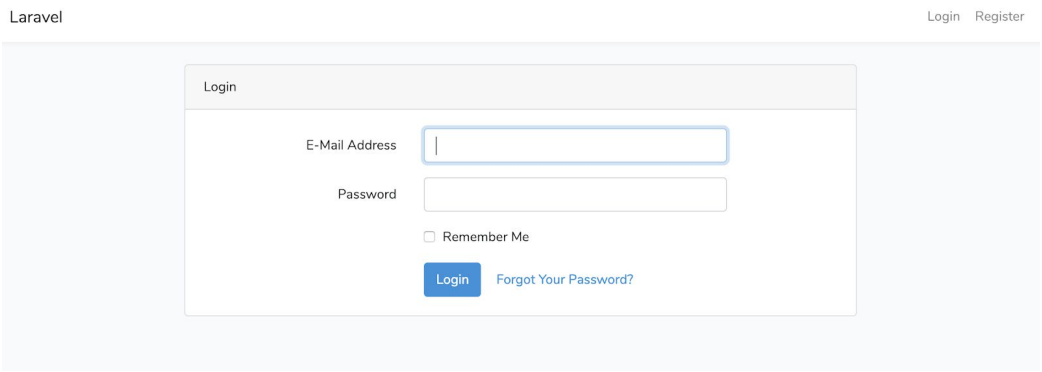


Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	api/user		Closure	api,auth:api
	GET|HEAD	home	home	App\Http\Controllers\HomeController@index	web,auth
	GET|HEAD	login	login	App\Http\Controllers\Auth\LoginController@showLoginForm	web,guest
	POST	login		App\Http\Controllers\Auth\LoginController@login	web,guest
	POST	logout	logout	App\Http\Controllers\Auth\LoginController@logout	web
	POST	password/email	password.email	App\Http\Controllers\Auth\ForgotPasswordController@sendResetLinkEmail	web,guest
	GET|HEAD	password/reset	password.request	App\Http\Controllers\Auth\ForgotPasswordController@showLinkRequestForm	web,guest
	POST	password/reset	password.update	App\Http\Controllers\Auth\ResetPasswordController@reset	web,guest
	GET|HEAD	password/reset/{token}	password.reset	App\Http\Controllers\Auth\ResetPasswordController@showResetForm	web,guest
	GET|HEAD	register	register	App\Http\Controllers\Auth\RegisterController@showRegistrationForm	web,guest
	POST	register		App\Http\Controllers\Auth\RegisterController@register	web,guest

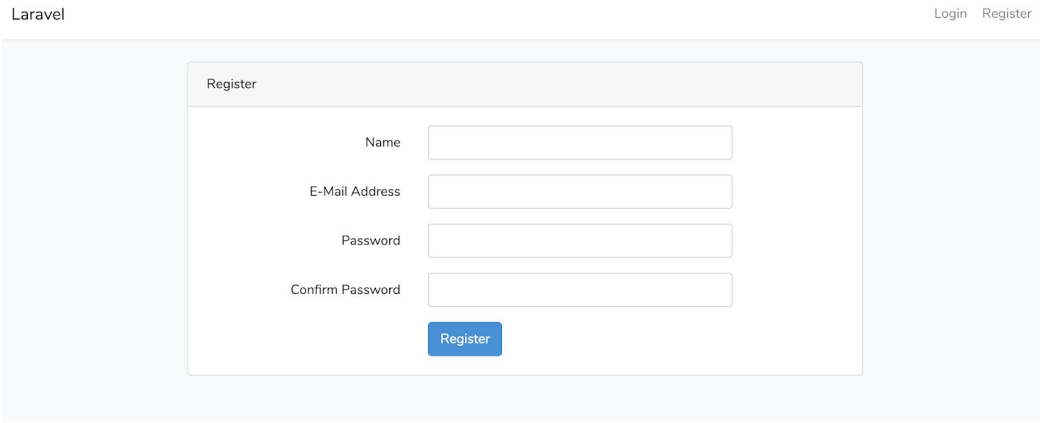
7. Sekarang silahkan jalankan server laravel, kemudian buka http://localhost:8000 di browser

Home page

Login Page



Register Page



8. Silahkan coba untuk register dan login

Membatasi halaman tertentu dari guest user

Untuk membatasi halaman tertentu bisa diakses kalau sudah login, bisa lihat di HomeController.php. Halaman http://localhost:8000/home hanya bisa diakses kalau sudah login. Perhatikan **line 16** pada gambar di bawah ini.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }
}

```

Mendapatkan current user logged in

1. Untuk mendapatkan current logged in user, kita bisa menggunakan Auth::user()

Contoh:
@auth
  Welcome back, {{ Auth::user()->name }}!
@else
        Hello, stranger! Login or Register.

```
@endauth
//-----------------------------------------------------------------------------------------------

@if (Auth::check())
        Welcome back, {{ Auth::user()->name }}!
@else
        Hello, stranger! <a href="{{ route('login') }}">Login</a> or <a href="{{
route('register') }}">Register</a>.
@endif
//-----------------------------------------------------------------------------------------------

@if (Auth::guest())
        Hello, stranger! <a href="{{ route('login') }}">Login</a> or <a href="{{
route('register') }}">Register</a>.
@else
        Welcome back, {{ Auth::user()->name }}!
@endif
```

2. Code diatas merupakan sample implementasi di views .blade.php

Memahami /admin namespace

1. Untuk melakukan management resources, misalkan CRUD products, best practice yang bisa kita ikuti adalah dengan menggunakan /admin namespace.
2. Namespace ini berguna untuk memisahkan halaman mana yang perlu login, dan halaman mana yang public.Setiap halaman yang diawali dengan path /admin, misalkan halaman http://localhost:8000/admin/products bisa diakses kalau sudah login.
3. Dibawah ini adalah sample /admin namespace yang akan kita buat.

Verb	URI	Action	Route Name
GET	`/admin/products`	index	admin/products.index
GET	`/admin/products/create`	create	admin/products.create

POST	`/admin/products`	store	admin/products.store
GET	`/admin/products/{photo}`	show	admin/products.show
GET	`/admin/products/{product_id}/edit`	edit	admin/products.edit
PUT/PATCH	`/admin/products/{product_id}`	update	admin/products.update
DELETE	`/admin/products/{product_id}`	destroy	admin/products.destroy

4. Finish

Product Management

Setting up model and database

1. Create model product
   php artisan make:model Product -m

2. Command tersebut akan men-generate 2 file:
   - app/Product.php
   - database/migration/s…._create_products_table.php
3. Untuk memudahkan pengelompokan file model, kita akan buat folder baru:
   app/Models

4. Pindahkan file app/Product.php ke app/Models/Product.php. Kemudian edit file model Product.php menjadi seperti ini

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    //
}
```

5. Kita buat scheme table databse dengan menambahkan code pada file database/migration/s…._create_products_table.php

```php
/**
 * Run the migrations.
 *
 * @return void
 */
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->integer('user_id');
        $table->string('name');
        $table->text('description');
        $table->string('price');
        $table->string('image_url')->nullable();
        $table->string('video_url')->nullable();
        $table->timestamps();
    });
}
```

6. Finish

Setting up controller

1. Generate controller product dengan menjalankan command dibawh ini
   php artisan make:controller Admin/ProductController --resource

2. Command diatas akan men-generate file app/Http/Controllers/Admin/ProductController.php dengan 7 method yang sudah tersedia:
   - index() → untuk menampilkan list produk
   - create() → untuk menambah produk baru (form view)

- store() → untuk menambah produk baru (form action)
- show() → untuk melihat detail produk
- edit() → untuk meng-edit produk (form view)
- update() → untuk meng-edit produk (form action)
- destroy() → untuk menghapus produk

3. Tambahkan code dibawah ini, untuk membatasi guest user

```php
class ProductController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }
```

4. Edit file routes/web.php, tambahkan code dibawah ini

```php
Route::name('admin.')->group(function () {
  Route::group(['prefix' => 'admin'], function () {
    Route::resource('products', 'Admin\ProductController');
  });
});
```

5. Untuk melihat hasil dari settingan routes diatas, jalan command dibawah ini
   php artisan route:list | grep product

```
| POST      | admin/products                 | admin.products.store   | App\Http\Controllers\Admin\ProductController@store
| GET|HEAD  | admin/products                 | admin.products.index   | App\Http\Controllers\Admin\ProductController@index
| GET|HEAD  | admin/products/create          | admin.products.create  | App\Http\Controllers\Admin\ProductController@create
| DELETE    | admin/products/{product}       | admin.products.destroy | App\Http\Controllers\Admin\ProductController@destroy
| PUT|PATCH | admin/products/{product}       | admin.products.update  | App\Http\Controllers\Admin\ProductController@update
| GET|HEAD  | admin/products/{product}       | admin.products.show    | App\Http\Controllers\Admin\ProductController@show
| GET|HEAD  | admin/products/{product}/edit  | admin.products.edit    | App\Http\Controllers\Admin\ProductController@edit
```

6. Finish

Setting up views

1. Untuk mempercantik tampilan, kita buat file public/css/style.css

```
/*
 * Sidebar
 */

.sidebar {
  position: fixed;
  top: 0;
  bottom: 0;
  left: 0;
  z-index: 100; /* Behind the navbar */
  padding: 0;
  box-shadow: inset -1px 0 0 rgba(0, 0, 0, .1);
}

.sidebar-sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 48px; /* Height of navbar */
  height: calc(100vh - 48px);
  padding-top: .5rem;
  overflow-x: hidden;
  overflow-y: auto; /* Scrollable contents if viewport is shorter than content. */
}

.sidebar .nav-link {
  font-weight: 500;
  color: #333;
}

.sidebar .nav-link .feather {
  margin-right: 4px;
  color: #999;
}

.sidebar .nav-link.active {
  color: #007bff;
```

```
}

.sidebar .nav-link:hover .feather,
.sidebar .nav-link.active .feather {
  color: inherit;
}

.sidebar-heading {
  font-size: .75rem;
  text-transform: uppercase;
}
```

2. Panggil file css tersbut di file resources/views/layouts/app.blade.php

```
…..
<!-- Styles -->
<link href="{{ asset('css/app.css') }}" rel="stylesheet">
<link href="{{ asset('css/style.css') }}" rel="stylesheet">
…..
```

3. Tambahkan code dibawah ini untuk membentuk sidebar menu di sebalah kiri.

```
…..
<ul class="navbar-nav mr-auto">
        @if (Auth::check())
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              Product
            </a>
            <div class="dropdown-menu" aria-labelledby="navbarDropdown">
              <a class="dropdown-item" href="{{ route('admin.products.index')
}}">List</a>
              <a class="dropdown-item" href="{{ route('admin.products.create')
}}">Tambah</a>
            </div>
          </li>
        @endif
      </ul>
…..
```

4. Code **@if (Auth::check())** akan membatasi guest user supaya tidak bisa melihat menu. Hanya user yang login yang bisa melihat menu
5. Finish

Membuat halaman list product

1. Pada app/Http/Controllers/Admin/ProductController.php tambahkan code dibawah ini
   …..
   use App\Http\Controllers\Controller;
   use App\Models\Product;

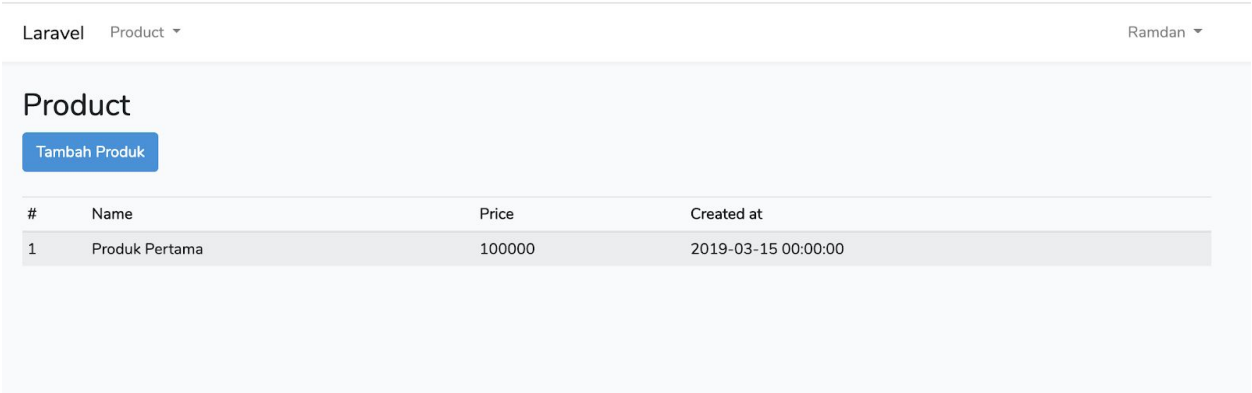
2. Pada method index() tambahkan code

```php
/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index()
{
    $products=Product::all();
    return view('admin.products.index', compact('products'));
}
```

3. Buat file resources/views/admin/products/index.blade.php

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

@section('content')
<div class="container col-md-8">
    <div class="row justify-content-center">
        <div class="col">
            <h2>Product</h2>
            <div>
                <a href="{{ route('admin.products.create') }}" class="btn btn-primary">Tambah Produk</a>
            </div>
            <br />
            <div class="table-responsive">
                <table class="table table-striped table-sm">
                  <thead>
                    <tr>
                      <th>#</th>
                      <th>Name</th>
                      <th>Price</th>
                      <th>Created at</th>
                    </tr>
                  </thead>
                  <tbody>
                        @foreach($products as $product)
                            <tr>
                                <td>{{ $product['id'] }}</td>
                                <td>{{ $product['name'] }}</td>
                                <td>{{ $product['price'] }}</td>
                                <td>{{ $product['created_at'] }}</td>
                            </tr>
                        @endforeach
                  </tbody>
                </table>
              </div>
        </div>
    </div>
</div>
@endsection

```

4. Silahkan isi table products langsung di phpmyadmin, minimal 1 row untuk melihat hasil dari halaman list produk, kemudian buka halaman http://localhost:8000/admin/products



5. Finish

Menambah fitur untuk tambah produk baru

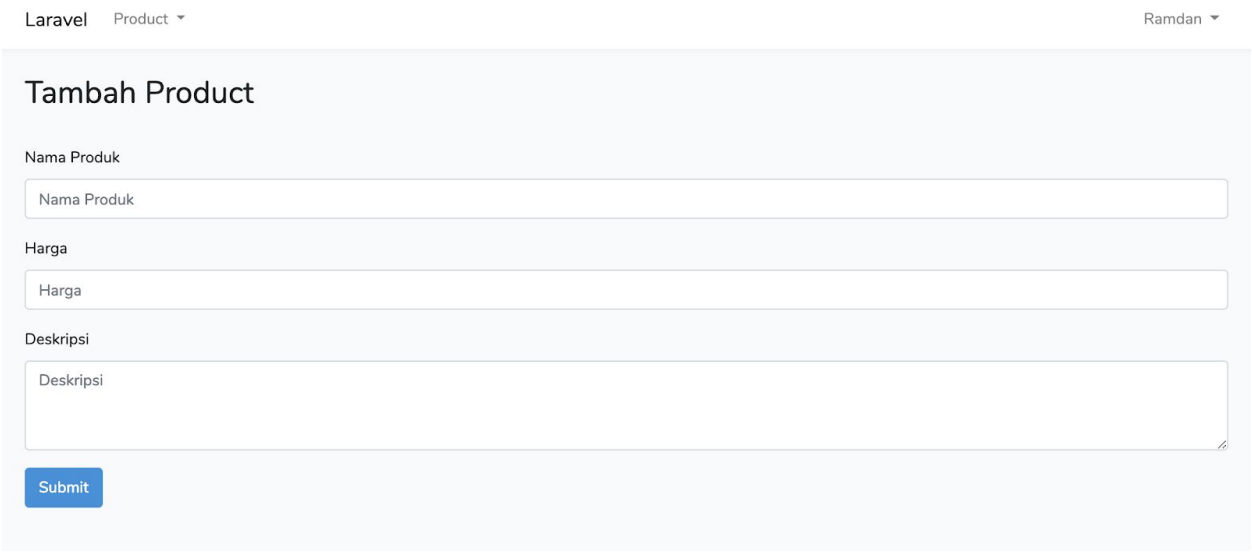
1. Pada app/Http/Controllers/Admin/ProductController.php pada method create() tambahkan code dibawah ini

```php
/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    return view('admin.products.create');
}
```

2. Buat file resources/views/admin/products/create.blade.php

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

@section('content')
  <div class="container">
    <div class="row justify-content-center">
      <div class="col">
        <h2>Tambah Product</h2>
        <br />
        <form action="{{ route('admin.products.store') }}" method="POST">
          <div class="form-group">
            <label>Nama Produk</label>
            <input type="text" name="name" class="form-control" placeholder="Nama Produk">
          </div>
          <div class="form-group">
            <label>Harga</label>
            <input type="number" name="price" class="form-control" placeholder="Harga">
          </div>
          <div class="form-group">
            <label>Deskripsi</label>
            <textarea name="description" class="form-control" rows="3" placeholder="Deskripsi"></textarea>
          </div>
          <button type="submit" class="btn btn-primary">Submit</button>
        </form>
      </div>
    </div>
  </div>
@endsection
```

3. Buka halaman ini pada browser http://localhost:8000/admin/products/create



4. Pada app/Http/Controllers/Admin/ProductController.php diatas class tambahkan code dibawah ini

   …..
   use App\Models\Product;
   use Auth;

   class ProductController extends Controller
   …..

5. Pada app/Http/Controllers/Admin/ProductController.php pada method store() tambahkan code dibawah ini

```php
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{

    $product = new Product();
    $product->user_id = Auth::user()->id;
    $product->name = $request->post('name');
    $product->price = $request->post('price');
    $product->description = $request->post('description');
    $product->save();

    return redirect('admin/products')->with('success', 'Produk berhasil di simpan');
}
```

6. Pada app/Models/Product.php, tambahkan code berikut

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'price', 'description',
    ];
}
```

7. Silahkan coba isi form dan simpan
8. Finish

Setting up form validation

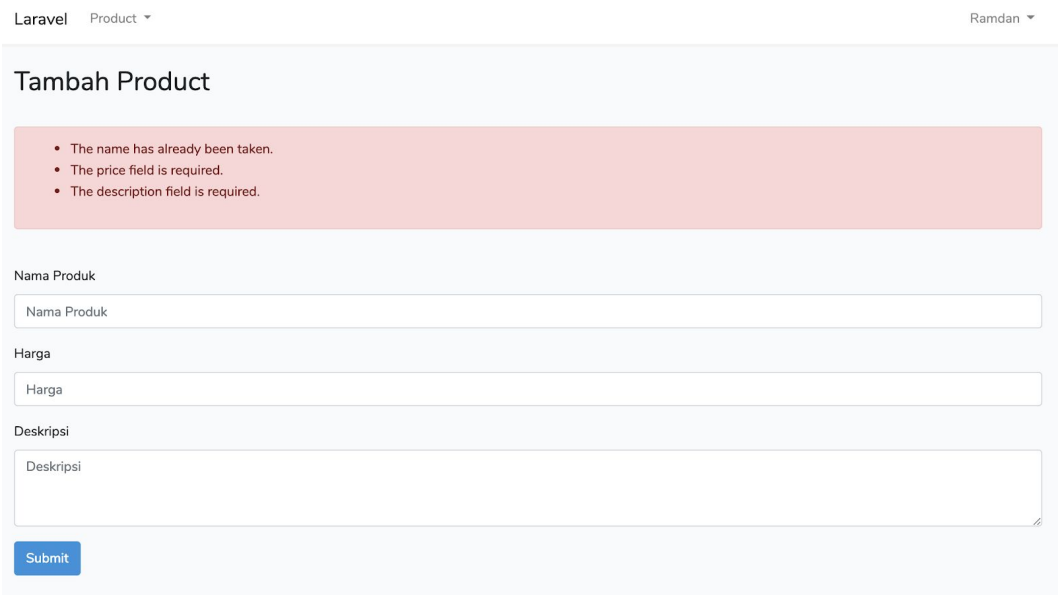
1. Pada app/Http/Controllers/Admin/ProductController.php pada method store() tambahkan code dibawah ini

```php
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{

    $this->validate(request(), [
        'name' => 'required|unique:products,name',
        'price' => 'required|numeric',
        'description' => 'required',
    ]);

    $product = new Product();
    $product->user_id = Auth::user()->id;
    $product->name = $request->post('name');
    $product->price = $request->post('price');
    $product->description = $request->post('description');
    $product->save();

    return redirect('admin/products')->with('success', 'Produk berhasil di simpan');
}
```

2. Pada file resources/views/admin/products/create.blade.php, tambahkan code ini

```php
<h2>Tambah Product</h2>

<br />
@if(count($errors))
  <div class="form-group">
    <div class="alert alert-danger">
      <ul>
        @foreach($errors->all() as $error)
            <li>{{$error}}</li>
        @endforeach
      </ul>
    </div>
  </div>
@endif
<br />

<form action="{{ route('admin.products.store') }}" method="POST">
```

3. SIlahkan coba untuk sumbit form yang tidak valid



4. Finish

Reference

1. https://laravel.com/docs/5.8/controllers
2. https://laravel.com/docs/5.8/routing#route-groups
3. https://laravel.com/docs/5.8/migrations
4. https://laravel.com/docs/5.8/validation
5. https://appdividend.com/2018/02/23/laravel-5-6-crud-tutorial/
6. https://vegibit.com/how-to-validate-form-submissions-in-laravel/