

Sistem penjualan motor sport

1. Deskripsi mengenai project tugas akhir

Latar Belakang:

Proyek Tugas Akhir ini bertujuan untuk mengatasi beberapa tantangan dalam industri penjualan motor sport, yang semakin berkembang dengan pesat. Perkembangan teknologi dan tren belanja online telah mengubah perilaku konsumen, membuat pentingnya adopsi platform e-commerce khusus untuk penjualan motor sport. Saat ini, pelanggan mengharapkan kemudahan dalam menelusuri, membandingkan, dan membeli produk sepeda motor sport tanpa harus mengunjungi secara fisik toko. Oleh karena itu, pembuatan website penjualan motor sport menjadi solusi yang relevan dan efektif.

Studi Kasus:

Proyek ini akan fokus pada pengembangan sebuah website e-commerce yang didedikasikan untuk penjualan motor sport. Studi kasus yang menjadi dasar proyek ini dapat melibatkan toko offline, yang menghadapi beberapa masalah seperti keterbatasan ruang pameran, kurangnya integrasi dengan sistem pembayaran modern, atau kesulitan dalam menyajikan informasi produk dengan jelas.

Deskripsi Proyek:

1. Desain dan Pengembangan Website: Proyek ini akan mencakup perancangan antarmuka pengguna yang responsif dan menarik untuk website e-commerce. Desain tersebut harus memudahkan pengguna dalam menemukan dan memilih produk motor sport yang mereka cari.
2. Manajemen Inventaris: Untuk meningkatkan efisiensi, proyek ini akan mencakup sistem manajemen inventaris yang terotomatisasi.
3. Pencarian Produk: Pengguna dapat dengan mudah menelusuri produk dan akan memastikan pengalaman belanja yang lebih baik.
4. Keamanan dan Privasi Pengguna: Keamanan data pengguna dan informasi transaksi adalah prioritas utama. Proyek ini akan memastikan pengguna merasa aman dan nyaman dalam berbelanja online.

Melalui proyek ini, diharapkan dapat memberikan solusi yang inovatif dan efektif untuk meningkatkan pengalaman belanja motor sport secara online serta mendukung pertumbuhan bisnis pada industri ini.

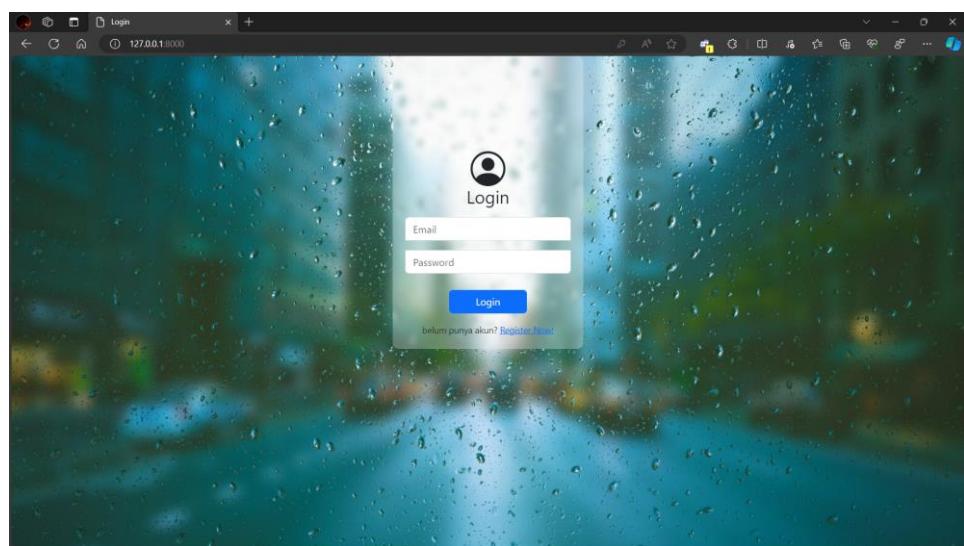
Link youtube presentasi :

<https://youtu.be/s-DJiPbfevE>

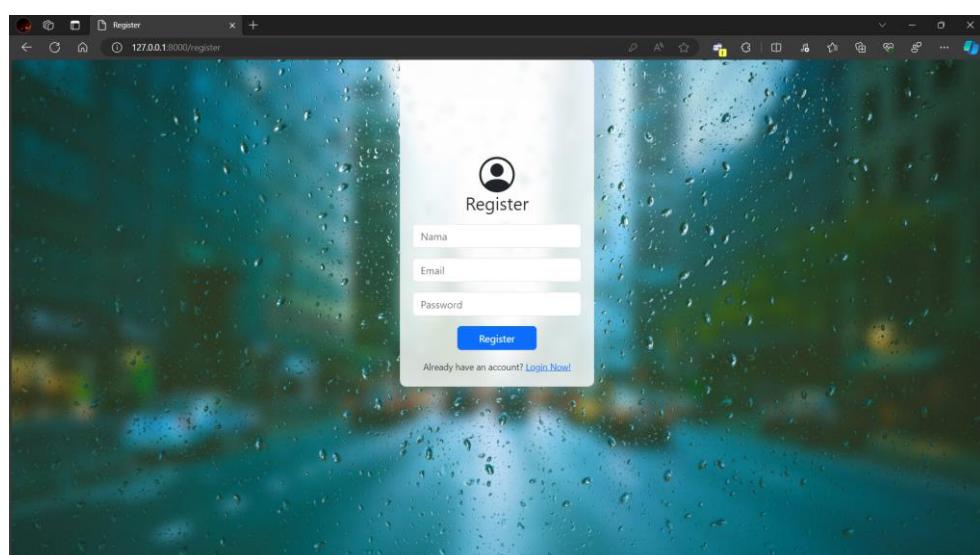
Project ini sudah saya coba hosting untuk link hosting :

<http://motor-sport-ajil.lovestoblog.com/>

2. Menjelaskan perancangan sistem. Pada bagian ini bisa ditampilkan dan dijelaskan lewat gambar dari relasi tabel yang digunakan.

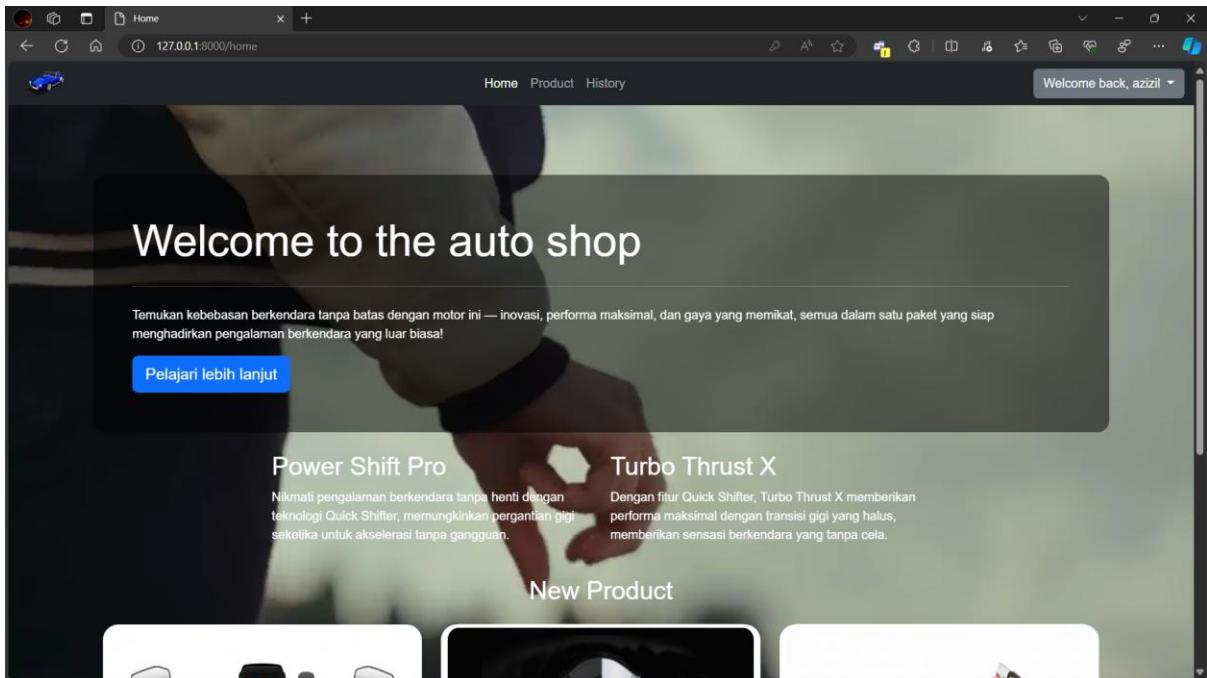


(Halaman Login)

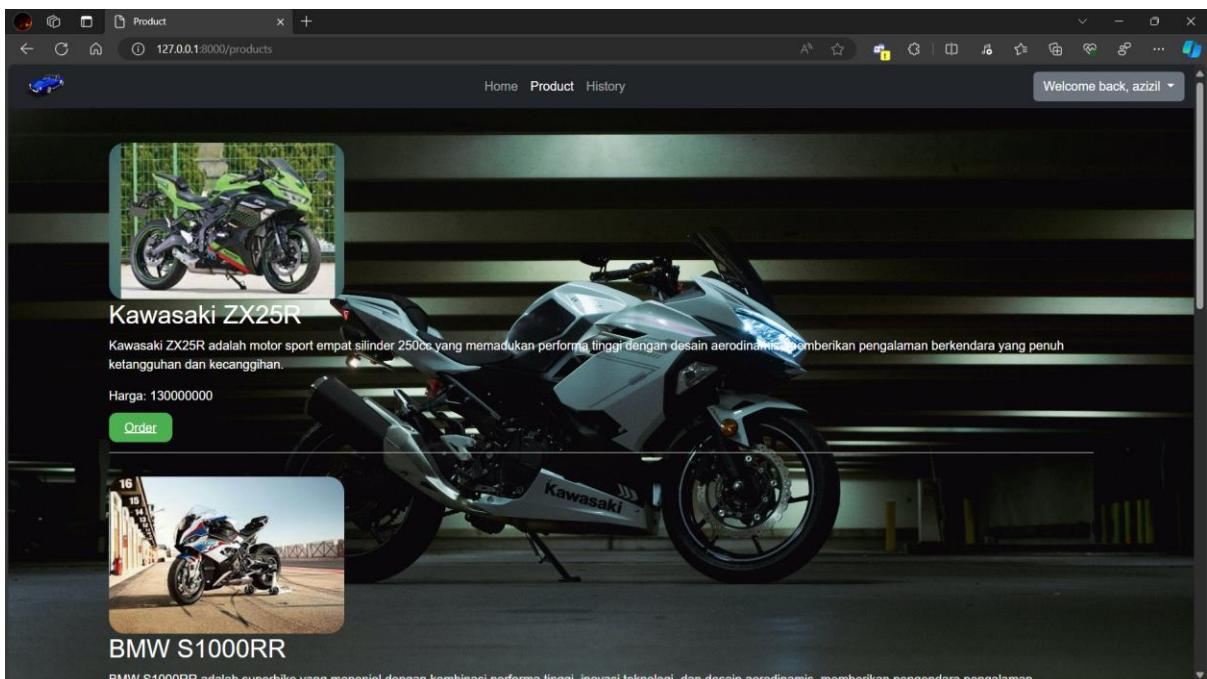


(Halaman Register)

- a. Halaman Utama yang dapat diakses oleh user dan admin



(Halaman Home)



(Halaman Product)

A screenshot of a web browser window titled "Your Purchased Products". The table has two columns: "Product Name" and "Price". One row is present, showing "Aprilia RS" and "60000000". Below the table, there are sections for "Informasi Kontak" (Address: Jalan purwosari, surakarta, Email: azizilputra@gmail.com, no : 0991873) and "Tautan" (links to "Kebijakan Privasi" and "Syarat dan Ketentuan"). A "Welcome back, azizil" message is visible in the top right corner.

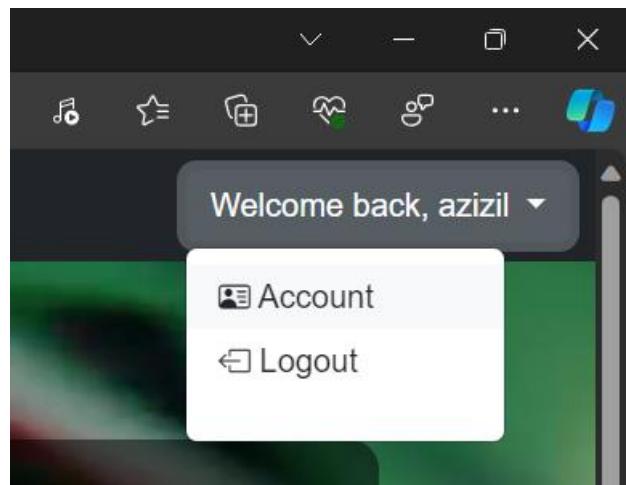
Product Name	Price
Aprilia RS	60000000

Informasi Kontak
Alamat : Jalan purwosari, surakarta
Email : azizilputra@gmail.com
no : 0991873

Tautan
[Kebijakan Privasi](#)
[Syarat dan Ketentuan](#)

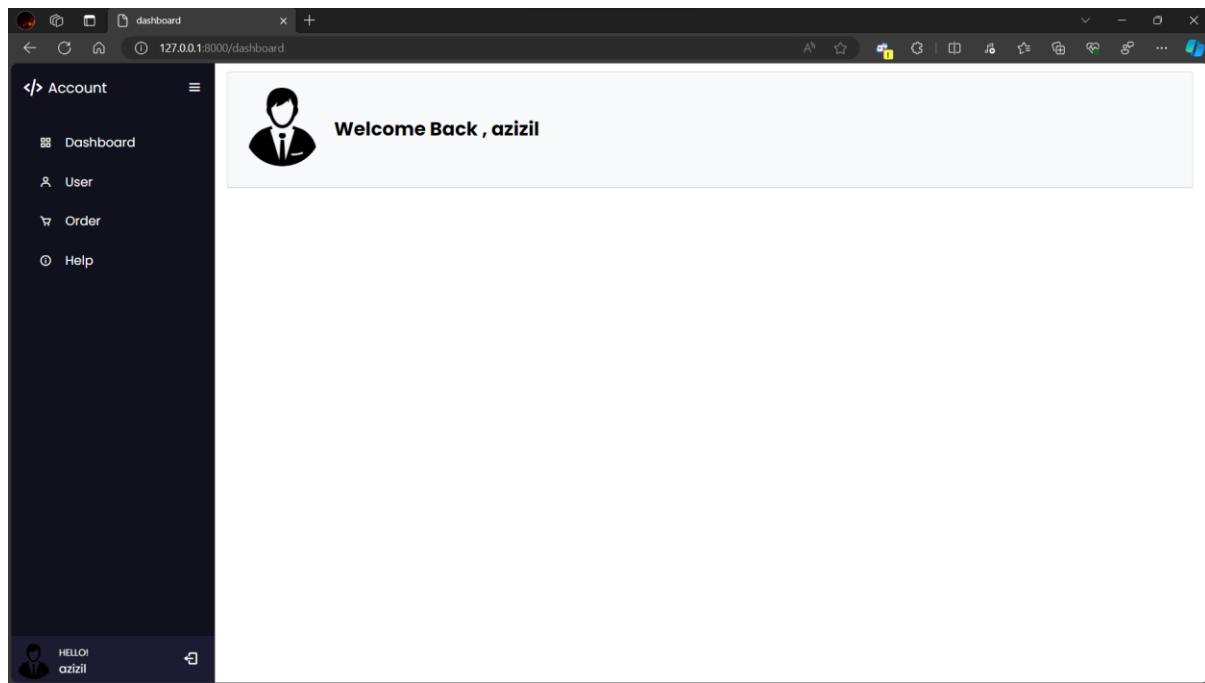
Welcome back, azizil

(Halaman History)

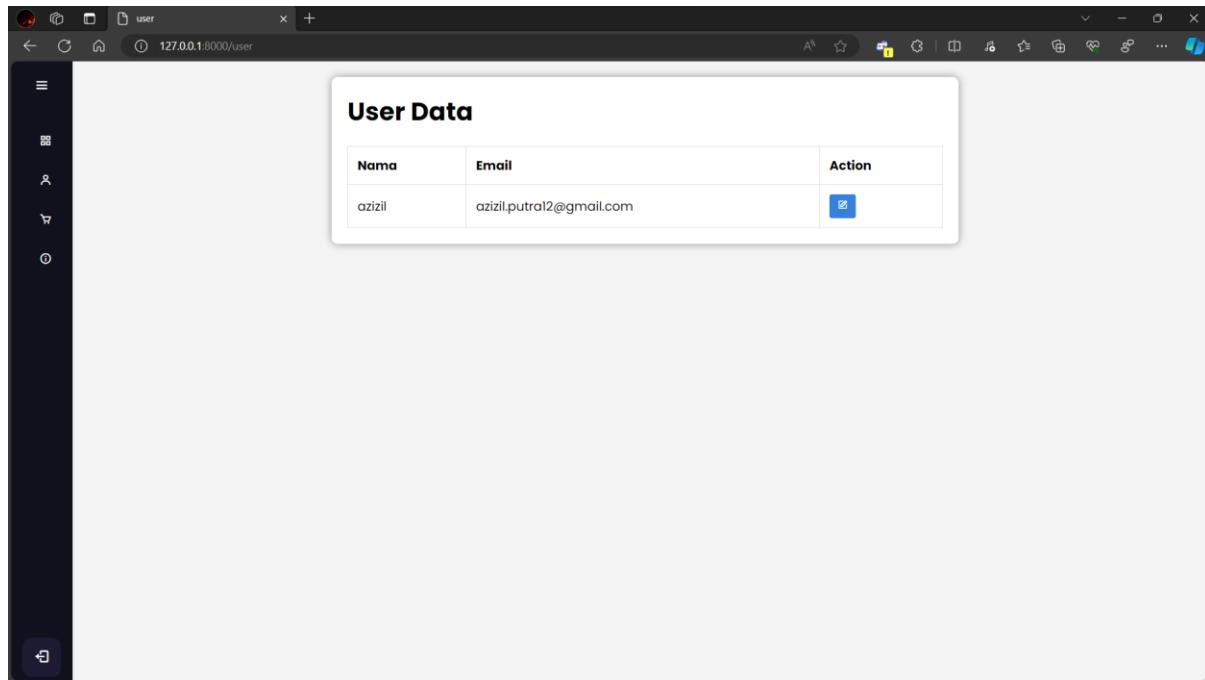


(Dropdown yang berisi Account dan logout)

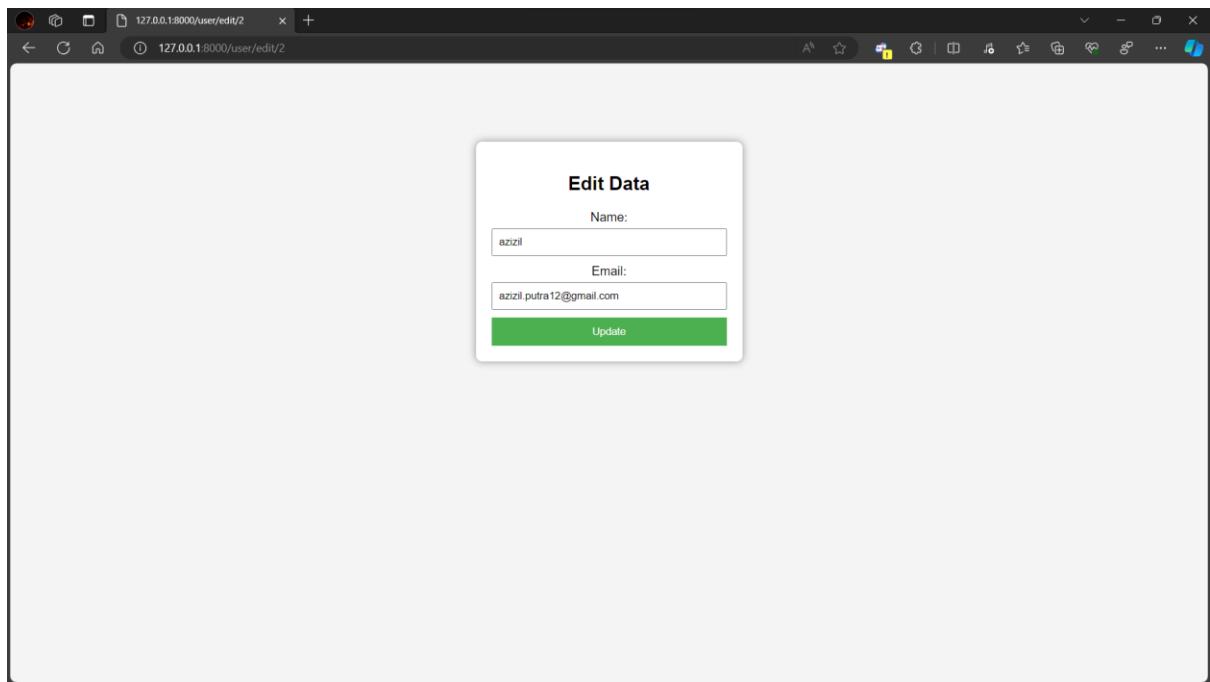
- b. Halaman yang hanya diakses oleh user



(Halaman dashboard user dan tampilan sidebar user)



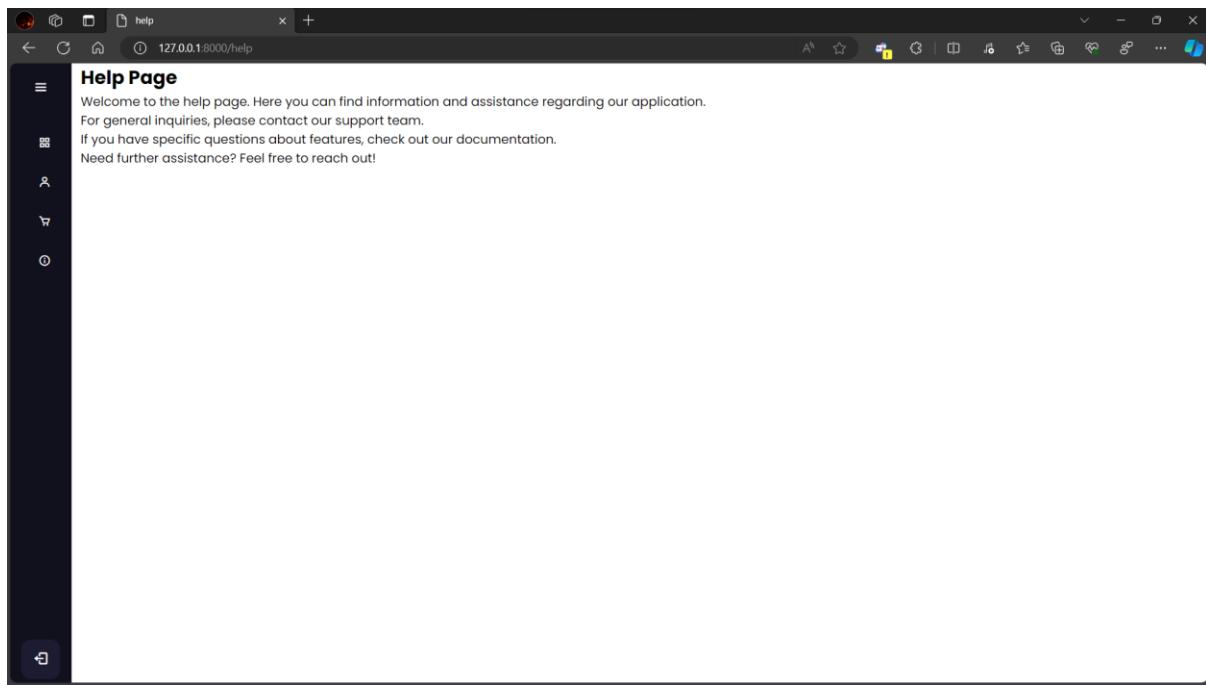
(Halaman data user)



(Halaman edit data user)

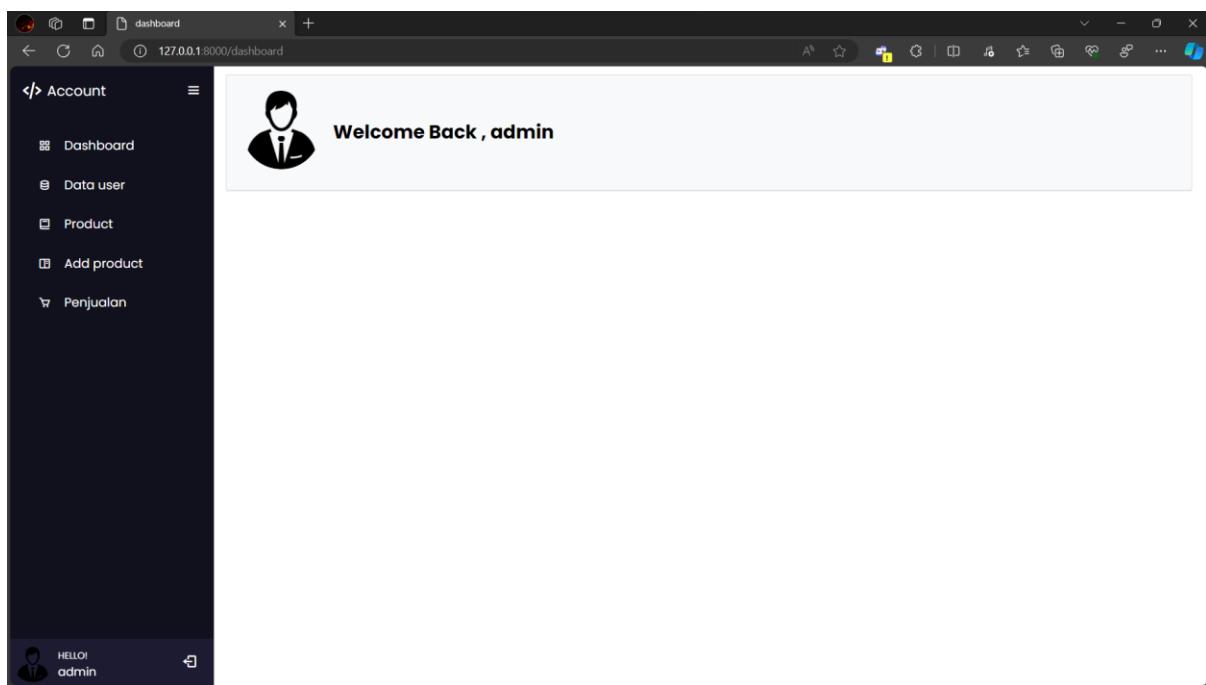
Your Purchased Products	
Product Name	Price
Aprilia RS	60000000

(Halaman Product yang dibeli oleh user)



(Halaman Help berisi tentang informasi yang diberikan untuk user)

c. Halaman yang hanya bisa diakses oleh Admin



(Tampilan Sidebar admin dan halaman dashboard admin)

Screenshot of a web browser showing the 'User Data' page. The URL is 127.0.0.1:8000/admin.

User Data

Nama	Email	Action
azizil	azizil.putra12@gmail.com	
makpur	makpur@gmail.com	

(Halaman admin yang menampilkan data user dan dapat dihapus)

Screenshot of a web browser showing the 'Product List' page. The URL is 127.0.0.1:8000/adminProducts.

Product List

Gambar	Nama	Harga	Deskripsi	Action
	Kawasaki ZX25R	130000000	Kawasaki ZX25R adalah motor sport empat silinder 250cc yang memadukan performa tinggi dengan desain aerodinamis, memberikan pengalaman berkendara yang penuh ketangguhan dan kecanggihan.	 
	BMW S1000RR	150000000	BMW S1000RR adalah superbike yang menonjol dengan kombinasi performa tinggi, inovasi teknologi, dan desain aerodinamis, memberikan pengendara pengalaman penuh kecanggihan di lintasan maupun jalan raya.	 
	Kawasaki Ninja H2	110000000	Kawasaki Ninja H2 adalah motor supercharged yang terkenal dengan desain agresif dan dilengkapi dengan teknologi supercharger, memberikan kinerja luar biasa dan kecepatan tinggi yang memukau para pecinta motor sport.	 
	Aprilia RS	600000000	Aprilia RS adalah motor sport ber tenaga tinggi yang dikenal karena kombinasi desain agresif, performa kelas atas, dan teknologi canggih, menyajikan pengalaman berkendara yang memikat para pecinta motor.	 

(Halaman menampilkan data product serta adanya button edit masuk ke halaman edit dan button delete untuk menghapus data product)

A screenshot of a web browser window titled "Edit Data". The URL is 127.0.0.1:8000/product/1/edit. The form contains the following fields:

- Name: Kawasaki ZX25R
- Deskripsi:

Kawasaki ZX25R adalah motor sport empat silinder 250cc yang memadukan performa tinggi dengan desain aerodinamis, memberikan pengalaman berkendara yang penuh ketangguhan dan kecanggihan.
- Harga: 13000000

At the bottom right of the form is a green "Update" button.

(Halaman edit data product)

A screenshot of a web browser window titled "Add Product". The URL is 127.0.0.1:8000/createProducts. The form contains the following fields:

- Name: (empty input field)
- Description: (empty input field)
- Price: (empty input field)

Below the input fields is a section labeled "Tambahkan Gambar model" (Add Model Image) with a "Pilih File" (Select File) button and a message "Tidak ada file yang dipilih" (No file selected). At the bottom right of the form is a green "Submit" button.

(Halaman tambah product)

User	Product Name	Price	Action
azizil	Aprilia RS	600000000	
makpur	Kawasaki ZX25RR	130000000	

(Halaman yang menampilkan data user yang membeli product)

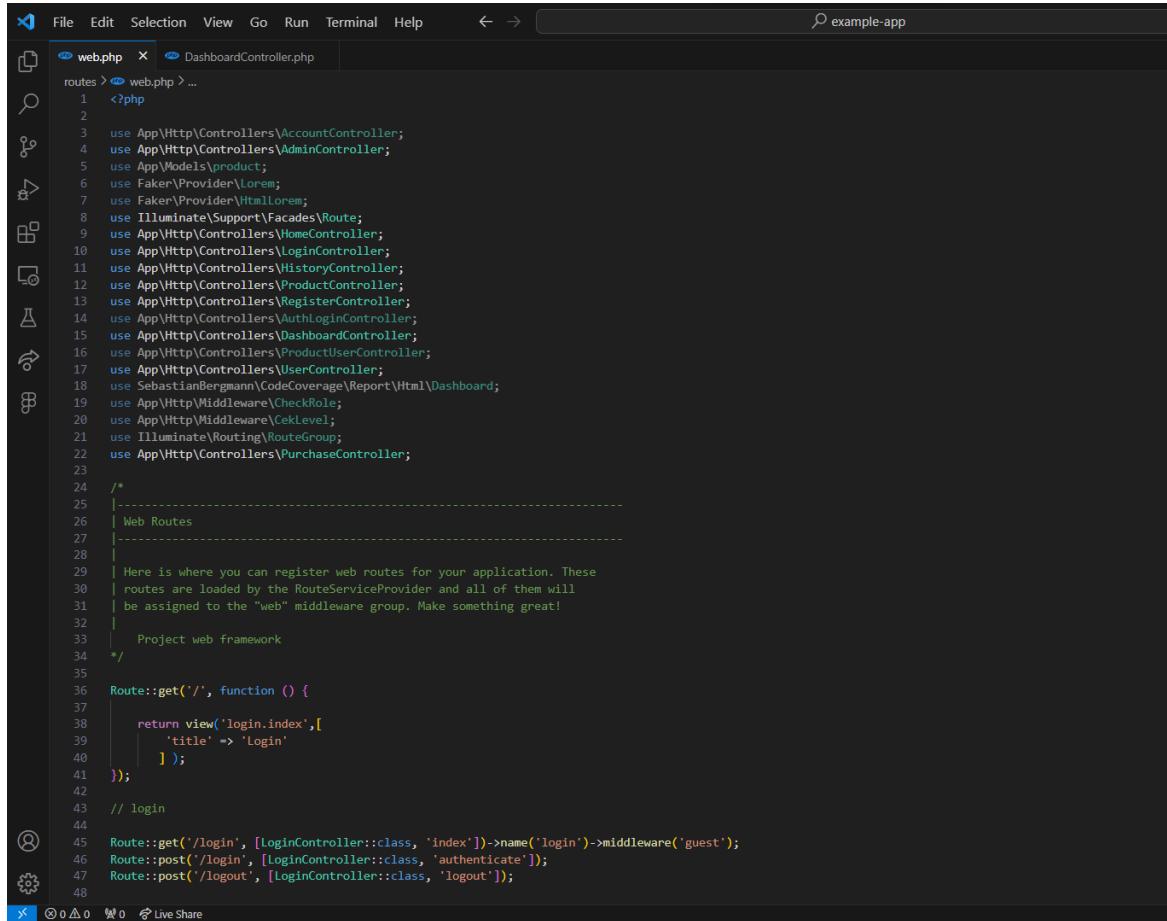
3. Menjelaskan fitur-fitur apa saja yang terdapat pada project. Adakah fitur CRUD? Adakah fitur authentication? Dan tambahan fitur yang lainnya jika ada.

Fitur-fitur yang terdapat pada sistem penjualan motor sport adanya

1. fitur CRUD dimana bisa menampilkan product, menambahkan sebuah product, edit product dan hapus sebuah product. Selain itu ada juga untuk membuat sebuah data user, update user, dan hapus data user yang dilakukan oleh admin.
2. Fitur authentication disini pada halaman login ketika user melakukan login akan masuk kedalam halaman selanjutnya ketika sudah melakukan login
3. Fitur middleware disini saya membuat middleware ceklevel dimana untuk melakukan authentication pada user dan admin, jika user melakukan login maka akan masuk kedalam halaman yang dapat diakses oleh user, dan jika admin melakukan login dengan data admin maka akan masuk ke dalam halaman yang dapat diakses oleh admin saja.
4. Fitur eloquent relationship dimana menggabungkan 2 database untuk memberikan data ke tabel baru, pada project saya untuk melakukan pada setiap user dapat melakukan pembelian product dan hanya user itu sendiri yang dapat mengakses data yang dibeli tersebut.

4. Penjelasan source code yang dituliskan. Pada bagian ini yang perlu dijelaskan diantaranya adalah : bagian **route**, **controller**, **model**, bagian **view**, file **.env** dan juga file **migration** yang ditambahkan.

1) Route



The screenshot shows the routes/web.php file in a code editor. The file contains PHP code defining web routes for a Laravel application. The code includes imports for various controllers and models, and defines routes for login, logout, and dashboard pages. A comment indicates that routes are loaded by the RouteServiceProvider and assigned to the 'web' middleware group. The code editor interface includes tabs for 'routes' and 'DashboardController.php', and various status icons at the bottom.

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\AccountController;
4  use App\Http\Controllers\AdminController;
5  use App\Models\product;
6  use Faker\Provider\Lorem;
7  use Faker\Provider\HtmlLorem;
8  use Illuminate\Support\Facades\Route;
9  use App\Http\Controllers\HomeController;
10 use App\Http\Controllers\LoginController;
11 use App\Http\Controllers\HistoryController;
12 use App\Http\Controllers\ProductController;
13 use App\Http\Controllers\RegisterController;
14 use App\Http\Controllers\AuthLoginController;
15 use App\Http\Controllers\DashboardController;
16 use App\Http\Controllers\ProductUserController;
17 use App\Http\Controllers\UserController;
18 use SebastianBergmann\CodeCoverage\Report\Html\Dashboard;
19 use App\Http\Middleware\CheckRole;
20 use App\Http\Middleware\CekLevel;
21 use Illuminate\Routing\RouteGroup;
22 use App\Http\Controllers\PurchaseController;
23
24 /**
25 | -----
26 | Web Routes
27 | -----
28 |
29 | Here is where you can register web routes for your application. These
30 | routes are loaded by the RouteServiceProvider and all of them will
31 | be assigned to the "web" middleware group. Make something great!
32 |
33 | Project web framework
34 */
35
36 Route::get('/', function () {
37     return view('login.index',[
38         'title' => 'Login'
39     ]);
40 });
41
42 // login
43
44 Route::get('/login', [LoginController::class, 'index'])->name('login')->middleware('guest');
45 Route::post('/login', [LoginController::class, 'authenticate']);
46 Route::post('/logout', [LoginController::class, 'logout']);
```

```

routes > web.php > ...
49 // register
50 Route::get('/register', [RegisterController::class, 'index'])->middleware('guest');
51 Route::post('/register', [RegisterController::class, 'store']);
52 |
53 // dashboard admin
54 Route::group(['middleware' => ['auth','ceklevel:admin']], function(){
55     // User data
56     Route::get('/admin', [AdminController::class, 'admin'])->name('admin');
57     Route::delete('/delete/user{id}', [AdminController::class, 'destroy'])->name('delete.user');
58 |
59     // Penjualan - admin
60     Route::get('/adminPenjualan', [AdminController::class, 'adminPenjualan'])->name('admin');
61     Route::delete('/delete/penjualan{id}', [AdminController::class, 'destroyPenjualan'])->name('delete.penjualan');
62 |
63     // product - admin
64     Route::get('/adminProducts', [AdminController::class, 'adminProduct'])->name('products.index');
65     Route::get('/createProducts', [AdminController::class, 'create']);
66     Route::post('/adminProducts', [AdminController::class, 'store'])->name('products.store');
67     // update product
68     Route::get('/product/{id}/edit', [AdminController::class, 'edit']);
69     Route::put('/product/{id}', [AdminController::class, 'update'])->name('product.update');
70     // delete product
71     Route::delete('/delete/product/{id}', [AdminController::class, 'destroyProduct'])->name('delete.product');
72 });
73 |
74 // dashboard user
75 Route::group(['middleware' => ['auth','ceklevel:user']], function(){
76     // dashboard user
77     Route::get('/user', [UserController::class, 'user']);
78 |
79     Route::get('/user/edit{id}', [UserController::class, 'edit']);
80     Route::put('/user/{id}', [UserController::class, 'update']);
81 |
82     Route::get('/order', [UserController::class, 'order']);
83     Route::get('/help', [UserController::class, 'help']);
84 });
85 |
86 |
87 // HOMEPAGE
88 Route::group(['middleware' => ['auth','ceklevel:admin,user']], function(){
89     // home
90     Route::get('/home', [HomeController::class, 'index'])->middleware('auth');
91     Route::get('/history', [HistoryController::class, 'index']);
92 |
93     // product
94     Route::get('/products', [ProductController::class, 'index'])->middleware('auth');
95 });
96 |
97 |
98 // pembelian
99 Route::get('/products', [ProductController::class, 'index'])->name('product.index');
100 Route::get('/purchased-products', [PurchaseController::class, 'viewPurchasedProducts'])->name('purchased-products.index');
101 Route::post('/products/{productId}/purchase', [PurchaseController::class, 'purchaseProduct'])->name('product.purchase');
102 |
103 // dashboard
104 Route::get('/dashboard', [DashboardController::class, 'index']);
105 });
106 |
107 |
108 |
109

```

Kode di atas adalah file definisi rute web pada framework Laravel

1. Penggunaan Namespace:

```

```php
use App\Http\Controllers\...
use App\Models\...
use Faker\Provider\...
use Illuminate\Support\Facades\Route;
use App\Http\Middleware\...
```

```

2. Definisi Rute Utama:

- Rute utama mengarah ke halaman login.
- Penggunaan fungsi `view` untuk menampilkan halaman login dengan data judul.

3. Rute Login dan Logout:

- Rute untuk menampilkan halaman login, mengotentikasi pengguna, dan keluar (logout).

4. Rute Registrasi:

- Rute untuk menampilkan halaman registrasi dan menyimpan data registrasi.

5. Rute Dashboard Admin:

- Mengelompokkan rute untuk admin dengan middleware autentikasi dan pengecekan level admin.
- Rute untuk manajemen pengguna, penjualan, dan produk.
- Fungsi CRUD untuk produk.

6. Rute Dashboard User:

- Mengelompokkan rute untuk pengguna dengan middleware autentikasi dan pengecekan level pengguna.
- Rute untuk dasbor pengguna, pengaturan profil, pesanan, dan bantuan.

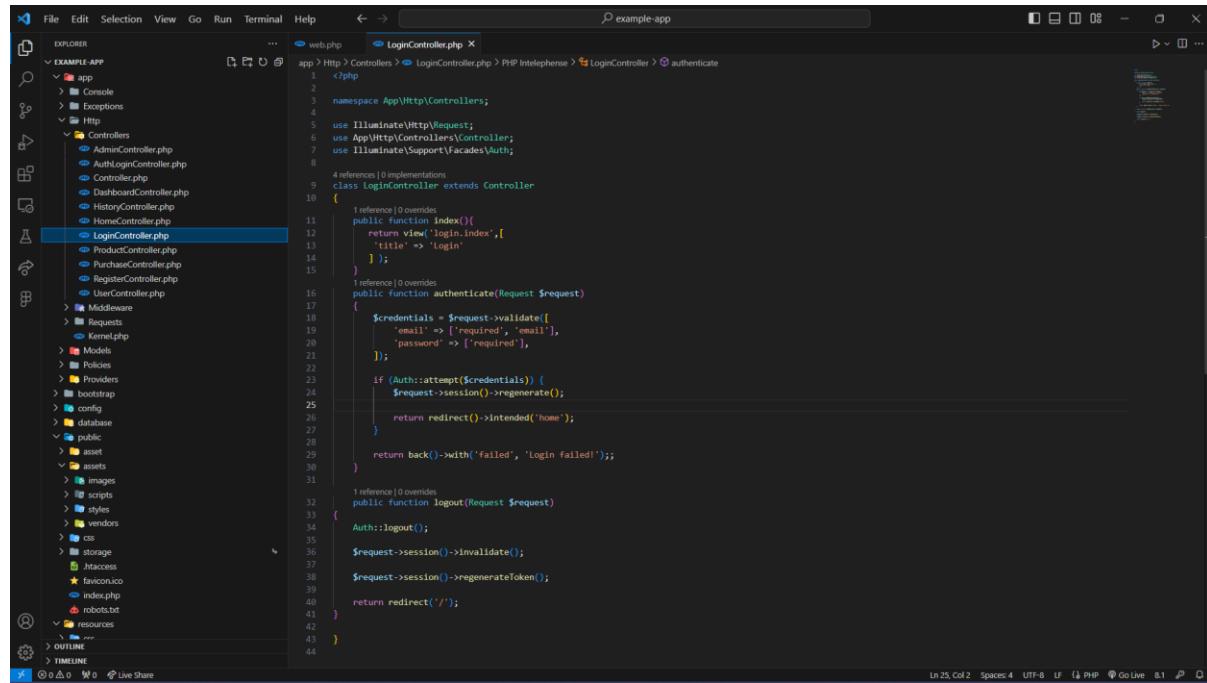
7. Rute Homepage:

- Mengelompokkan rute untuk pengguna dan admin dengan middleware autentikasi dan pengecekan level.
- Rute untuk beranda, riwayat, produk, pembelian, dan dasbor.

Dalam keseluruhan, kode ini memanfaatkan middleware untuk memastikan bahwa pengguna yang mengakses rute tertentu telah terautentikasi dan memiliki level akses yang sesuai. Selain itu, rute-rute tersebut juga memberikan akses ke fungsi-fungsi kontrol yang berbeda sesuai dengan peran pengguna (admin atau user) dan menangani operasi CRUD untuk produk serta fungsi lainnya.

2) Controller

1. LoginController



```
class LoginController extends Controller
{
    public function index()
    {
        return view('login.index', [
            'title' => 'Login'
        ]);
    }

    public function authenticate(Request $request)
    {
        $credentials = $request->validate([
            'email' => ['required', 'email'],
            'password' => ['required'],
        ]);

        if (Auth::attempt($credentials)) {
            $request->session()->regenerate();

            return redirect()->intended('home');
        }

        return back()->with('failed', 'Login failed');
    }

    public function logout(Request $request)
    {
        Auth::logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

Kode di atas adalah bagian dari sebuah kontroler (controller) pada aplikasi Laravel yang mengelola operasi terkait login dan logout pengguna.

1. index Method:

- Menampilkan halaman login.
- Mengembalikan tampilan `login.index` dengan data judul ('Login').

2. authenticate Method:

- Menerima permintaan (Request) dari formulir login.
- Validasi input email dan password menggunakan Method `validate`.
- Jika validasi berhasil, mencoba untuk mengautentikasi pengguna dengan menggunakan Method `Auth::attempt` dari Laravel.
- Jika autentikasi berhasil, mengarahkan pengguna ke halaman yang dimaksud (intended), dalam hal ini 'home'.
- Jika autentikasi gagal, kembali ke halaman login dengan pesan kesalahan.

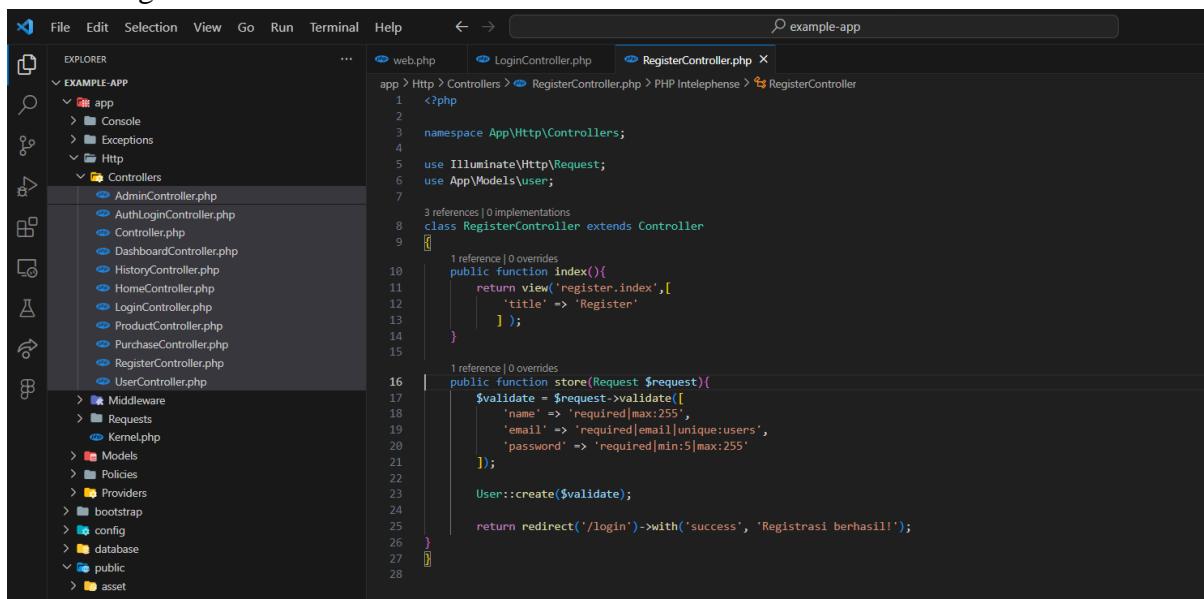
3. logout Method:

- Menangani permintaan logout.
- Memanggil Method `Auth::logout` untuk logout pengguna.
- Mematikan sesi (invalidate session), meregenerasi token, dan mengarahkan pengguna ke halaman utama.
- Tujuannya adalah untuk membersihkan status login pengguna dan mengarahkannya ke halaman awal.

Catatan:

- Kode ini menggunakan fasilitas otentikasi bawaan Laravel, seperti `Auth::attempt` untuk mencoba mengautentikasi pengguna berdasarkan kredensial yang diberikan.
- Fungsi `redirect()->intended('home')` digunakan untuk mengarahkan pengguna ke URL yang dimaksudkan sebelumnya atau ke 'home' jika tidak ada URL sebelumnya yang disimpan.
- Pesan kesalahan ('failed') dikirim kembali ke halaman login jika autentikasi gagal, dan pesan ini dapat digunakan untuk memberikan umpan balik kepada pengguna.

2. RegisterController



The screenshot shows a code editor with the file `RegisterController.php` open. The file is located in the `Http\Controllers` directory of a Laravel application named `example-app`. The code defines a `RegisterController` class that extends `Controller`. It contains two methods: `index()` and `store(Request $request)`. The `index()` method returns a view named `'register.index'` with a title of 'Register'. The `store` method validates the request input, creates a new `User` instance using the `create` method, and then redirects the user to the login page with a success message.

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
use App\Models\User;  
  
class RegisterController extends Controller  
{  
    public function index()  
    {  
        return view('register.index',[  
            'title' => 'Register'  
        ]);  
    }  
  
    public function store(Request $request)  
    {  
        $validate = $request->validate([  
            'name' => 'required|max:255',  
            'email' => 'required|email|unique:users',  
            'password' => 'required|min:5|max:255'  
        ]);  
  
        User::create($validate);  
  
        return redirect('/login')->with('success', 'Registrasi berhasil!');  
    }  
}
```

Kode di atas adalah bagian dari sebuah kontroler (controller) pada aplikasi Laravel yang menangani operasi registrasi pengguna.

1. `index` Method:`

- Menampilkan halaman registrasi.
- Mengembalikan tampilan `'register.index'` dengan data judul ('register').

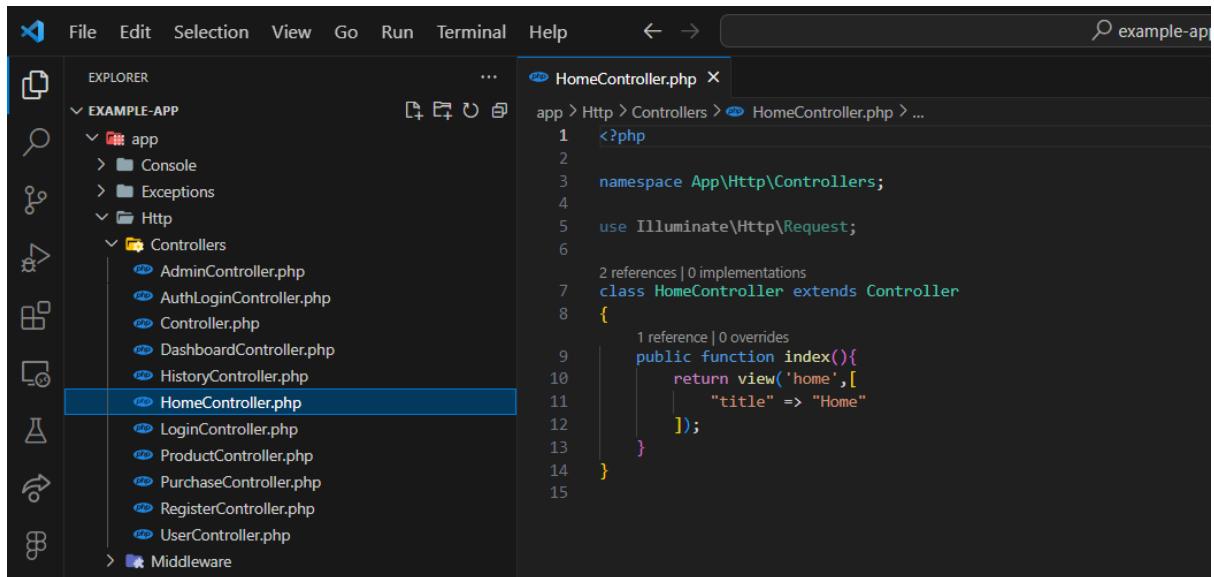
2. ``store` Method:`

- Menerima permintaan (Request) dari formulir registrasi.
- Melakukan validasi input dengan menggunakan Method `'validate'` pada objek permintaan.
- Validasi melibatkan:
 - Nama harus wajib diisi dan maksimal 255 karakter.
 - Email harus wajib diisi, harus dalam format email, dan harus unik di dalam tabel 'users'.
 - Password harus wajib diisi, minimal 5 karakter, dan maksimal 255 karakter.
- Jika validasi berhasil, membuat instansi baru dari model `'User'` menggunakan Method `'create'` dan menyimpannya ke dalam tabel 'users'.
- Mengarahkan pengguna ke halaman login dengan pesan sukses ('success').

Catatan:

- Kode ini menggunakan fasilitas validasi bawaan Laravel (`\$request->validate`) untuk memastikan bahwa data yang diterima dari formulir sesuai dengan aturan yang ditentukan.
- Fungsi `User::create(\$validate)` digunakan untuk membuat dan menyimpan instansi baru dari model `User` ke dalam database, dengan data yang sudah divalidasi.
- Pesan sukses ('success') dikirim kembali ke halaman login dan dapat digunakan untuk memberikan umpan balik kepada pengguna setelah berhasil mendaftar.

3. HomeController



The screenshot shows a code editor interface with a dark theme. On the left is an 'EXPLORER' sidebar showing the project structure of 'EXAMPLE-APP'. Under 'Http\Controllers', the 'HomeController.php' file is selected and highlighted with a blue bar at the bottom. The main panel displays the contents of the HomeController.php file:

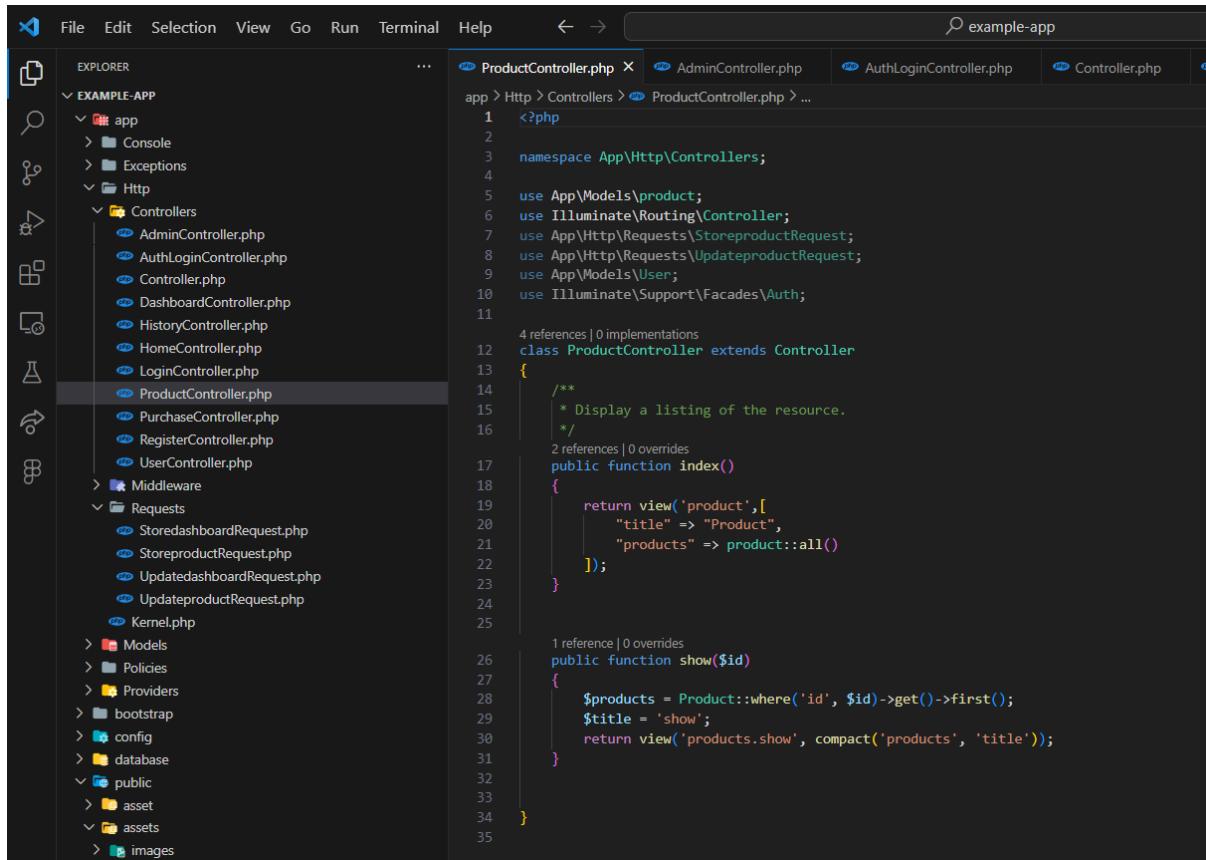
```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
  
class HomeController extends Controller  
{  
    public function index()  
    {  
        return view('home',[  
            "title" => "Home"  
        ]);  
    }  
}
```

Kode di atas adalah bagian dari sebuah kontroler (controller) pada aplikasi Laravel yang menangani operasi terkait dengan halaman utama (home).

1. `index` Method:

- Menangani permintaan untuk menampilkan halaman utama.
- Mengembalikan tampilan dengan nama 'home' dan menyertakan data judul ('title') sebagai parameter.
- Data judul akan digunakan dalam tampilan untuk menampilkan judul halaman.

4. ProductController



The screenshot shows a code editor interface with the following details:

- File Explorer (Left):** Shows the project structure under "EXAMPLE-APP". Key folders include "app", "Http", and "Controllers". The "Controllers" folder contains files like "AdminController.php", "AuthLoginController.php", "Controller.php", "DashboardController.php", "HistoryController.php", "HomeController.php", "LoginController.php", "ProductController.php" (which is selected), "PurchaseController.php", "RegisterController.php", and "UserController.php".
- Code Editor (Right):** Displays the content of "ProductController.php". The code defines two methods: "index()" and "show(\$id)".
- Search Bar:** At the top right, it says "example-app".

```
<?php  
namespace App\Http\Controllers;  
  
use App\Models\product;  
use Illuminate\Routing\Controller;  
use App\Http\Requests\StoreproductRequest;  
use App\Http\Requests\UpdateproductRequest;  
use App\Models\User;  
use Illuminate\Support\Facades\Auth;  
  
class ProductController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     */  
    public function index()  
    {  
        return view('product',[  
            "title" => "Product",  
            "products" => product::all()  
        ]);  
    }  
  
    public function show($id)  
    {  
        $products = Product::where('id', $id)->get()->first();  
        $title = 'show';  
        return view('products.show', compact('products', 'title'));  
    }  
}
```

Kode di atas adalah bagian dari sebuah kontroler (controller) pada aplikasi Laravel yang menangani operasi terkait produk.

1. `index` Method:

- Menampilkan daftar produk.
- Mengembalikan tampilan dengan nama 'product' dan menyertakan data judul ('title') dan daftar produk ('products') sebagai parameter.
- Data produk diperoleh dengan menggunakan Method `product::all()`.

2. `show` Method:

- Menampilkan detail produk berdasarkan ID.
- Menerima ID produk sebagai parameter.
- Menggunakan model `Product` untuk mencari produk dengan ID yang sesuai.
- Mengembalikan tampilan dengan nama 'products.show', dan menyertakan data produk ('products') dan judul ('title') sebagai parameter.
- Method `compact` digunakan untuk membuat array asosiatif dari variabel yang ingin disertakan dalam tampilan.

5. HistoryController

```

<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class HistoryController extends Controller
{
    public function index(){
        return view('history',[ "title" => "History"]);
    }
}

```

Kode di atas adalah bagian dari sebuah kontroler (controller) pada aplikasi Laravel yang menangani operasi terkait dengan riwayat (history).

1. `index` Method:

- Menampilkan halaman riwayat.
- Mengembalikan tampilan dengan nama 'history' dan menyertakan data judul ('title') sebagai parameter.

6. AdminController

```

<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Product;
use App\Models\Purchase;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
class AdminController extends Controller
{
    // bagian view admin
    public function admin(){
        $users = User::where('level', '!=', 'admin')->get();
        return view('account.admin.admin',[ "users" => $users,
        "title" => 'admin']);
    }

    // bagian view adminProduct
    public function adminProduct(){
        return view('account.admin.adminProduct',[ "title" => 'admin',
        "products" => product::all()]);
    }

    // bagian create
    public function create()
    {
        return view('account.admin.createProduct',[ "title" => 'admin']);
    }
}

```

```

File Edit Selection View Go Run Terminal Help < > example-app
EXPLORER app Http Controllers AdminController.php DashboardController.php
EXAMPLE-APP
    app
        Console
        Exceptions
    Http
        Controllers
            AdminController.php
            AuthLoginController.php
            Controller.php
            DashboardController.php
            HistoryController.php
            HomeController.php
            LoginController.php
            ProductController.php
            PurchaseController.php
            RegisterController.php
            UserController.php
        Middleware
    Requests
        StoredashboardRequest.php
        StoreproductRequest.php
        UpdatedashboardRequest.php
        UpdateproductRequest.php
        Kernel.php
            Models
            Policies
            Providers
                bootstrap
                config
                database
            public
                asset
                assets
                images
                scripts
                styles
                vendors
                css
                storage
                .htaccess
            .outline
            .timeline
    OUTLINE
    TIMELINE
    Live Share
File Edit Selection View Go Run Terminal Help < > example-app
EXPLORER AdminController.php DashboardController.php
EXAMPLE-APP
    app
        Console
        Source Control (Ctrl+Shift+G)
            Http
                Controllers
                    AdminController.php
                    AuthLoginController.php
                    Controller.php
                    DashboardController.php
                    HistoryController.php
                    HomeController.php
                    LoginController.php
                    ProductController.php
                    PurchaseController.php
                    RegisterController.php
                    UserController.php
                Middleware
            Requests
                StoredashboardRequest.php
                StoreproductRequest.php

```

AdminController.php (Top Editor)

```

45     $validate = $request->validate([
46         'nama' => 'required',
47         'deskripsi' => 'required',
48         'harga' => 'required|numeric',
49         'gambar' => 'image|file',
50     ]);
51
52     if($request->file('gambar')){
53         $validate['gambar'] = $request->file('gambar')->store('post-image');
54     }
55
56     Product::create($validate);
57
58     return redirect()->route('products.index')->with('success', 'Product created successfully.');
59
60 }
61
62 }

1 reference | 0 overrides
public function edit($id){
    $product = Product::find($id);
    return view('account.admin.editProduct', [
        'product' => $product,
        'title' => 'user edit'
    ]);
}

1 reference | 0 overrides
public function update(Request $request, $id)
{
    $request->validate([
        'nama' => 'required|string|max:255',
        'deskripsi' => 'required|string',
        'harga' => 'required|numeric',
    ]);

    $product = Product::find($id);
    $product->update([
        'nama' => $request->input('nama'),
        'deskripsi' => $request->input('deskripsi'),
        'harga' => $request->input('harga'),
    ]);

    return redirect('/adminProducts')->with('success', 'Product updated successfully');
}

```

AdminController.php (Bottom Editor)

```

91 // bagian view adminPenjualan
92
93     public function adminPenjualan()
94     {
95         // Get all purchases with related user and product information
96         $purchases = Purchase::with(['user', 'product'])->get();
97         $title = 'show';
98
99         return view('account.admin.adminPenjualan', compact('purchases', 'title'));
100    }

1 reference | 0 overrides
101    public function destroyPenjualan($id){
102        $purchases = Purchase::find($id);
103        $purchases->delete();
104
105        return redirect('/adminPenjualan')->with('success', 'image deleted successfully');
106    }

107
108
109
110 }

111

```

Kode di atas adalah bagian dari kontroler (controller) pada aplikasi Laravel yang mengelola operasi terkait dengan admin. Berikut adalah penjelasan untuk masing-masing Method dalam kontroler tersebut:

1. Method `admin`:

- Menampilkan daftar pengguna yang bukan admin.
- Mengembalikan tampilan 'account.admin.admin' dengan data pengguna ('users') dan judul ('title') sebagai parameter.
- Fungsi ini digunakan untuk menampilkan halaman admin.

2. Method `destroy`:

- Menghapus pengguna berdasarkan ID.
- Menggunakan model `User` untuk mencari dan menghapus pengguna.
- Mengarahkan pengguna kembali ke halaman admin setelah pengguna dihapus, dengan memberikan pesan sukses ('success').

3. Method `adminProduct`:

- Menampilkan daftar produk.
- Mengembalikan tampilan 'account.admin.adminProduct' dengan data judul ('title') dan daftar produk ('products') sebagai parameter.
- Fungsi ini digunakan untuk menampilkan halaman admin produk.

4. Method `create`:

- Menampilkan formulir pembuatan produk.
- Mengembalikan tampilan 'account.admin.createProduct' dengan data judul ('title').

5. Method `store`:

- Menyimpan data produk yang baru dibuat.
- Melakukan validasi input produk.
- Mengelola gambar produk jika ada.
- Membuat dan menyimpan produk baru menggunakan model `Product`.
- Mengarahkan pengguna ke indeks produk dengan memberikan pesan sukses ('success').

6. Method `edit`:

- Menampilkan formulir edit untuk produk tertentu.
- Mengembalikan tampilan 'account.admin.editProduct' dengan data produk ('product') dan judul ('title') sebagai parameter.

7. Method `update`:

- Mengupdate data produk yang telah diubah.
- Melakukan validasi input produk.
- Mengupdate produk yang sesuai berdasarkan ID menggunakan model `Product`.
- Mengarahkan pengguna ke halaman daftar produk admin dengan memberikan pesan sukses ('success').

8. Method `adminPenjualan`:

- Menampilkan daftar pembelian dengan informasi pengguna dan produk terkait.
- Mengembalikan tampilan 'account.admin.adminPenjualan' dengan data pembelian ('purchases') dan judul ('title') sebagai parameter.

9. Method `destroyPenjualan`:

- Menghapus pembelian berdasarkan ID.

- Menggunakan model `Purchase` untuk mencari dan menghapus pembelian.
- Mengarahkan pengguna kembali ke halaman admin penjualan setelah pembelian dihapus, dengan memberikan pesan sukses ('success').

Catatan:

- Kode ini menggunakan beberapa model seperti `User`, `Product`, dan `Purchase` yang terhubung dengan tabel-tabel terkait dalam basis data.
- Fungsi-fungsi tersebut digunakan untuk mengelola operasi CRUD pada pengguna, produk, dan pembelian.

7. UserController

```

File Edit Selection View Go Run Terminal Help ← →
EXPLORER DashboardController.php UserController.php
EXAMPLE-APP Controllers
  - AdminController.php
  - AuthLoginController.php
  - Controller.php
  - DashboardController.php
  - HistoryController.php
  - HomeController.php
  - LoginController.php
  - ProductController.php
  - PurchaseController.php
  - RegisterController.php
  - UserController.php
  - Middleware
    - StoredashboardRequest.php
    - StoreproductRequest.php
    - UpdatedashboardRequest.php
    - UpdatetproductRequest.php
  - Kernel.php
  - Models
  - Policies
  - Providers
  - bootstrap
  - config
  - database
  - public
    - asset
    - assets
      - images
      - scripts
      - styles
    - vendors
  - storage
  - .htaccess
  - favicon.ico
  - index.php
  - robots.txt
  - resources
    - css
    - js
    - views
      - account
        - admin
          - admin.blade.php
          - adminRegular.blade.php
          - adminProduct.blade.php
          - createProduct.blade.php
          - editProduct.blade.php
        - layouts
          - main.blade.php
        - partials
          - sidebar.blade.php
      - user
        - help.blade.php
        - order.blade.php
        - user.blade.php
        - useredit.blade.php
        - dashboard.blade.php
      - layouts
        - main.blade.php
        - sign.blade.php
      - login
        - index.blade.php
      - partials
        - footer.blade.php
        - navbar.blade.php
      - products
        - _product.blade.php
OUTLINE
TIMELINE
Live Share

```

```

DashboardController.php UserController.php
app/Http/Controllers/UserController.php > PHP Intelephense > UserController
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Routing\Controller;
7 use Illuminate\Foundation\Auth\User;
8 use Illuminate\Support\Facades\Auth;
9
10
11 // references | 0 implementations
12 class UserController extends Controller
13 {
14     // help
15     public function help(){
16         return view('account.user.help',[

17             'title' => 'help'
18         ]);
19     }
20
21     // order
22     public function order(){
23         $user = auth()->user();
24         $purchasedProducts = $user->purchases()->with('product')->get();
25
26         return view('account.user.order',[

27             'purchasedProducts' => $purchasedProducts,
28             'title' => 'order'
29         ]);
30     }
31
32
33     // user
34     public function user(){
35         $loggedInUser = Auth::user();
36         $users = User::where('id', $loggedInUser->id)->get();
37
38         return view('account.user.user',[

39             'users' => $users,
40             'title' => 'user'
41         ]);
42     }
43
44     // edit
45     public function edit($id){
46         $user = User::find($id);
47         return view('account.user.useredit',[

48             'user' => $user,
49             'title' => 'user edit'
50         ]);
51     }
52
53     // references | 0 overrides
54     public function update(Request $request, $id)
55     {
56         // Validate the request data
57         $request->validate([
58             'name' => 'required|string|max:255',
59             'email' => 'required|email|unique:users,email,' . $id,
60         ]);
61
62         // Find the user by ID
63         $user = User::findOrFail($id);
64
65         // Update user data
66         $user->name = $request->input('name');
67         $user->email = $request->input('email');
68         // You can add more fields to update as needed
69
70         // Save the changes
71         $user->save();
72
73         // Redirect to the edit view with the updated user data
74         return redirect('/user')->with('success', 'Data updated successfully.');
75     }
76

```

Kode di atas adalah bagian dari kontroler (controller) pada aplikasi Laravel yang mengelola operasi terkait pengguna.

1. Metode `help`:

- Menampilkan halaman bantuan.
- Mengembalikan tampilan 'account.user.help' dengan data judul ('title') sebagai parameter.

2. Metode `order`:

- Menampilkan daftar produk yang telah dibeli oleh pengguna.
- Mendapatkan pengguna yang terautentikasi.
- Mengambil produk yang telah dibeli oleh pengguna dengan bantuan relasi Eloquent `purchases()` pada model `User`.
- Mengembalikan tampilan 'account.user.order' dengan data produk yang telah dibeli ('purchasedProducts') dan judul ('title') sebagai parameter.

3. Metode `user`:

- Menampilkan informasi pengguna yang terautentikasi.
- Mendapatkan pengguna yang terautentikasi dan mengembalikan tampilan 'account.user.user' dengan data pengguna ('users') dan judul ('title') sebagai parameter.

4. Metode `edit`:

- Menampilkan formulir edit untuk pengguna tertentu.
- Mengembalikan tampilan 'account.user.useredit' dengan data pengguna ('user') dan judul ('title') sebagai parameter.

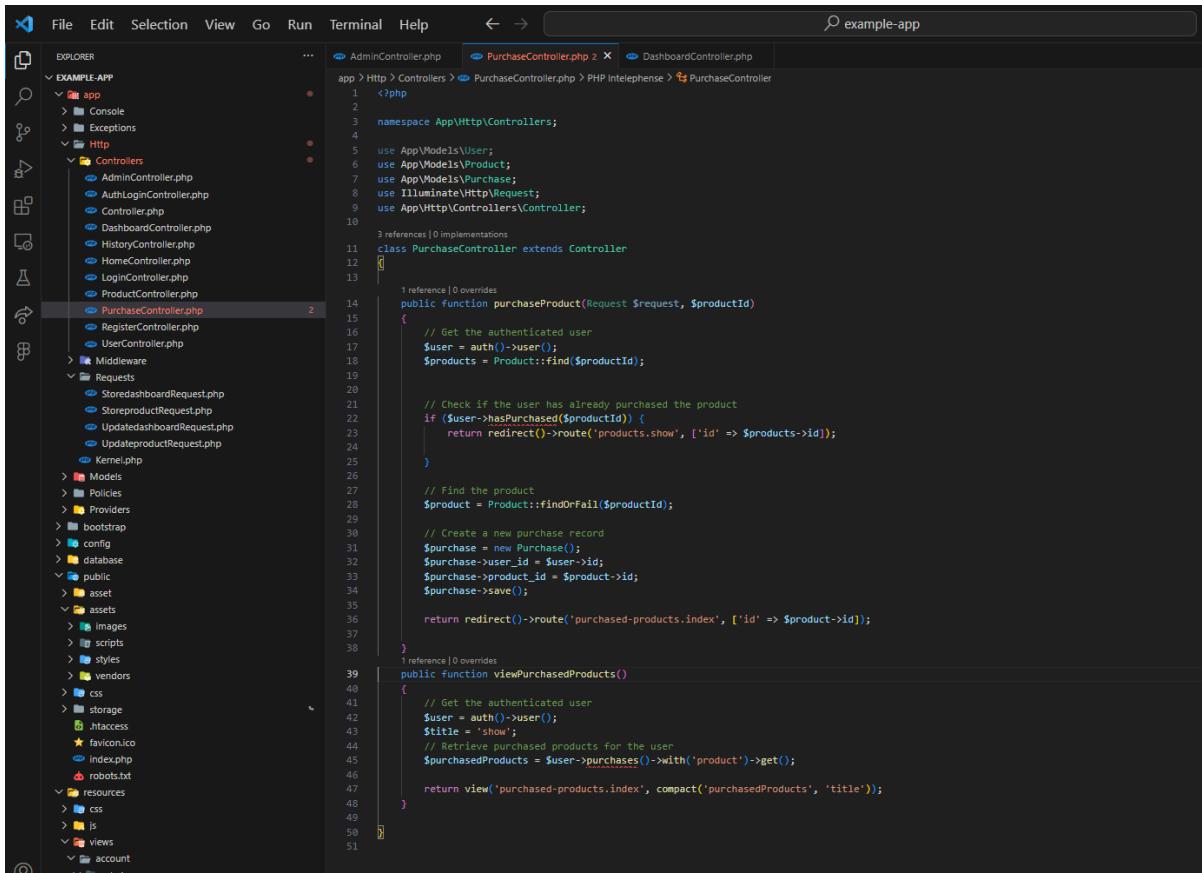
5. Metode `update`:

- Mengupdate data pengguna yang telah diubah.
- Melakukan validasi input pengguna.
- Mengupdate pengguna yang sesuai berdasarkan ID menggunakan model `User`.
- Mengarahkan pengguna ke halaman tampilan pengguna setelah pengguna diubah, dengan memberikan pesan sukses ('success').

Catatan:

- Kode ini menggunakan model `User` yang terhubung dengan tabel 'users' dalam basis data.
- Fungsi-fungsi tersebut digunakan untuk mengelola operasi terkait informasi pengguna, seperti menampilkan bantuan, daftar produk yang telah dibeli, dan mengelola data pengguna.

8. PurchaseController



The screenshot shows a code editor with the PurchaseController.php file open in the central pane. The left sidebar displays the project structure of an 'EXAMPLE-APP' Laravel application, including the app, Http, Controllers, Requests, and other standard Laravel directories. The PurchaseController.php file contains PHP code for managing purchases.

```
<?php  
namespace App\\Http\\Controllers;  
  
use App\\Models\\User;  
use App\\Models\\Product;  
use App\\Models\\Purchase;  
use Illuminate\\Http\\Request;  
use App\\Http\\Controllers\\Controller;  
  
class PurchaseController extends Controller  
{  
  
    public function purchaseProduct(Request $request, $productId)  
    {  
        // Get the authenticated user  
        $user = auth()>>user();  
        $products = Product::find($productId);  
  
        // Check if the user has already purchased the product  
        if ($user->hasPurchased($productId)) {  
            return redirect()->route('products.show', ['id' => $products->id]);  
        }  
  
        // Find the product  
        $product = Product::findOrFail($productId);  
  
        // Create a new purchase record  
        $purchase = new Purchase();  
        $purchase->user_id = $user->id;  
        $purchase->product_id = $product->id;  
        $purchase->save();  
  
        return redirect()->route('purchased-products.index', ['id' => $product->id]);  
    }  
  
    public function viewPurchasedProducts()  
    {  
        // Get the authenticated user  
        $user = auth()>>user();  
        $title = 'show';  
        // Retrieve purchased products for the user  
        $purchasedProducts = $user->purchases()->with('product')->get();  
  
        return view('purchased-products.index', compact('purchasedProducts', 'title'));  
    }  
}
```

Kode di atas adalah bagian dari kontroler (controller) pada aplikasi Laravel yang menangani operasi terkait pembelian produk.

1. Metode `purchaseProduct`:

- Menerima permintaan pembelian produk dengan ID tertentu.
- Mendapatkan pengguna yang terautentikasi.
- Memeriksa apakah pengguna telah membeli produk tersebut sebelumnya.
- Jika sudah, pengguna diarahkan kembali ke halaman tampilan produk.
- Jika belum, mencari produk berdasarkan ID.
- Membuat catatan pembelian baru dan menyimpannya.
- Mengarahkan pengguna ke halaman daftar produk yang telah dibeli.

2. Metode `viewPurchasedProducts`:

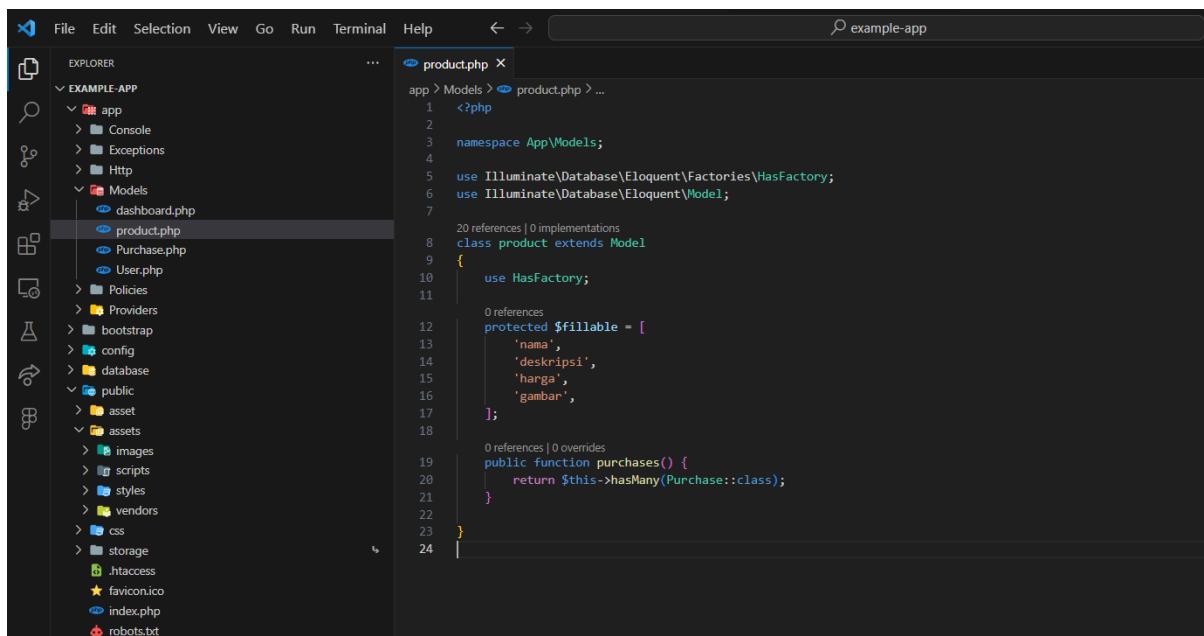
- Menampilkan daftar produk yang telah dibeli oleh pengguna.
- Mendapatkan pengguna yang terautentikasi.
- Mengambil produk yang telah dibeli oleh pengguna dengan bantuan relasi Eloquent `purchases()` pada model `User`.
- Mengembalikan tampilan 'purchased-products.index' dengan data produk yang telah dibeli ('purchasedProducts') dan judul ('title') sebagai parameter.

Catatan:

- Kode ini menggunakan model `User`, `Product`, dan `Purchase`, yang terhubung dengan tabel-tabel terkait dalam basis data.
- Fungsi `auth()->user()` digunakan untuk mendapatkan pengguna yang sedang terautentikasi.
- Fungsi `with('product')` digunakan untuk mengambil data produk terkait dengan setiap pembelian.
- Pada metode `purchaseProduct`, ada pengecekan apakah pengguna sudah membeli produk tersebut sebelumnya atau belum sebelum membuat catatan pembelian baru.

3) Model

1. Model Product



The screenshot shows a code editor with the file `product.php` open in the `App\Models` directory of a Laravel application named `example-app`. The code defines a `Product` model that extends `Model` and implements `HasFactory`. It has attributes `nama`, `deskripsi`, `harga`, and `gambar`. A relationship `purchases` is defined as a many-to-many association with the `Purchase` model.

```
<?php  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Product extends Model  
{  
    use HasFactory;  
  
    protected $fillable = [  
        'nama',  
        'deskripsi',  
        'harga',  
        'gambar',  
    ];  
  
    public function purchases()  
    {  
        return $this->hasMany(Purchase::class);  
    }  
}
```

Kode di atas merupakan definisi model Eloquent pada aplikasi Laravel yang disebut `Product`.

1. Pewarisan dan Penggunaan Fitur:

- Model `Product` menggunakan fitur-fitur dari trait `HasFactory`.
- Trait `HasFactory` digunakan untuk menyediakan metode-fungsi pembuat (factory) yang memudahkan pembuatan data uji.

2. Atribut Fillable:

- Atribut `fillable` (`$fillable`) menentukan kolom-kolom yang dapat diisi pada model ini secara massal.
- Dalam hal ini, atribut `fillable` mencakup `'nama'`, `'deskripsi'`, `'harga'`, dan `'gambar'`.

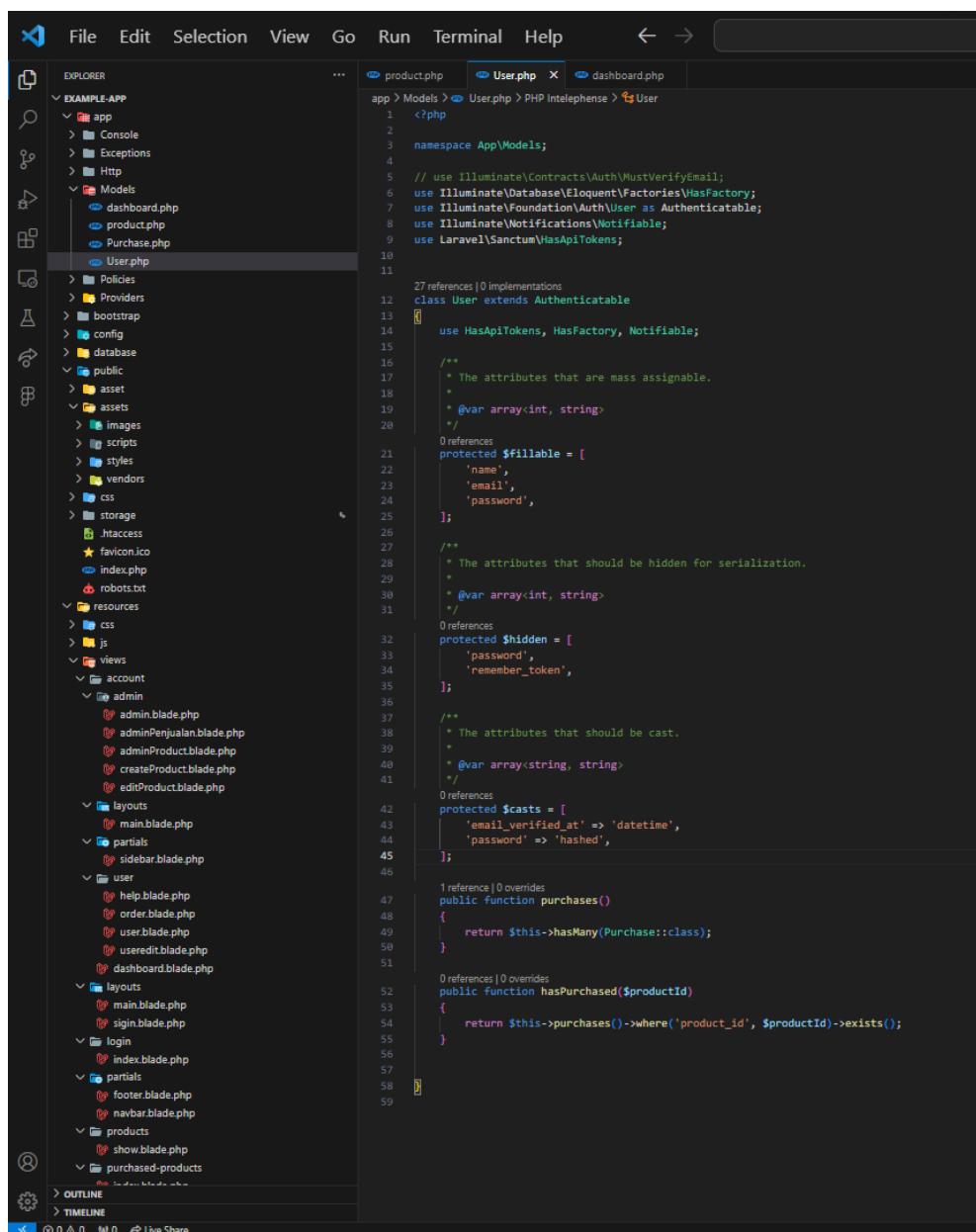
3. Relasi Pembelian:

- Model `Product` memiliki relasi dengan model `Purchase` melalui metode `purchases()`.
- Metode ini mendefinisikan relasi 'hasMany', yang menunjukkan bahwa satu produk dapat memiliki banyak pembelian.

Catatan:

- Kode ini mencerminkan struktur dasar model Eloquent pada Laravel.
- Model ini dapat digunakan untuk mengakses dan mengelola data pada tabel 'products' dalam basis data.

2. Model User



The screenshot shows a code editor interface with the User.php file open. The file is located in the app/Models directory of a Laravel application named EXAMPLE-APP. The code defines a User model that extends the Authenticatable trait. It includes mass assignable attributes (name, email, password), hidden attributes (password, remember_token), and casted attributes (email_verified_at, password). The purchases() method returns a hasMany relationship to the Purchase model. The hasPurchased(\$productId) method checks if there is a purchase for a given product ID.

```
File: User.php
app > Models > User.php > PHP Intelphense > User
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 // use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9 use Laravel\Sanctum\HasApiTokens;
10
11
12 27 references | 0 implementations
13 class User extends Authenticatable
14 {
15     use HasApiTokens, HasFactory, Notifiable;
16
17     /**
18      * The attributes that are mass assignable.
19      *
20      * @var array<int, string>
21     */
22     protected $fillable = [
23         'name',
24         'email',
25         'password',
26     ];
27
28     /**
29      * The attributes that should be hidden for serialization.
30      *
31      * @var array<int, string>
32     */
33     protected $hidden = [
34         'password',
35         'remember_token',
36     ];
37
38     /**
39      * The attributes that should be cast.
40      *
41      * @var array<string, string>
42     */
43     protected $casts = [
44         'email_verified_at' => 'datetime',
45         'password' => 'hashed',
46     ];
47
48     1 reference | 0 overrides
49     public function purchases()
50     {
51         return $this->hasMany(Purchase::class);
52     }
53
54     0 references | 0 overrides
55     public function hasPurchased($productId)
56     {
57         return $this->purchases()->where('product_id', $productId)->exists();
58     }
59 }
```

Kode di atas merupakan definisi model Eloquent pada aplikasi Laravel yang disebut `User`.

1. Pewarisan dan Penggunaan Fitur:

- Model `User` menggunakan fitur-fitur dari trait `HasApiTokens`, `HasFactory`, dan `Notifiable`.
- `HasApiTokens` memungkinkan pengguna menggunakan Sanctum untuk otentikasi API.
- `HasFactory` memberikan fungsi-fungsi factory yang membantu pembuatan data uji.
- `Notifiable` memberikan notifikasi melalui berbagai saluran komunikasi.

2. Atribut Fillable:

- Atribut fillable (`\$fillable`) menentukan kolom-kolom yang dapat diisi secara massal.
- Dalam hal ini, atribut fillable mencakup 'name', 'email', dan 'password'.

3. Atribut Hidden:

- Atribut hidden (`\$hidden`) menentukan kolom-kolom yang tidak akan ditampilkan saat data di-serialize.
- Dalam hal ini, 'password' dan 'remember_token' tidak akan ditampilkan.

4. Atribut Casts:

- Atribut casts (`\$casts`) menentukan tipe data untuk atribut tertentu.
- Dalam hal ini, 'email_verified_at' di-cast sebagai tipe data 'datetime', dan 'password' di-cast sebagai tipe data 'hashed'.

5. Relasi Pembelian:

- Model `User` memiliki relasi dengan model `Purchase` melalui metode `purchases()`.
- Metode ini mendefinisikan relasi 'hasMany', menunjukkan bahwa satu pengguna dapat memiliki banyak pembelian.

6. Metode `hasPurchased`:

- Metode ini digunakan untuk memeriksa apakah pengguna telah membeli produk tertentu.
- Menggunakan metode `exists()` untuk menentukan apakah ada pembelian dengan `product_id` yang sesuai.

Catatan:

- Model ini mencerminkan struktur dasar model Eloquent pada Laravel.
- Model `User` umumnya digunakan untuk mengelola data pengguna, termasuk autentikasi dan relasi dengan entitas lain seperti pembelian.

3. Model Purchase

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Purchase extends Model
{
    use HasFactory;
    protected $fillable = ['user_id', 'product_id'];
    public function product() {
        return $this->belongsTo(Product::class);
    }
    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

Kode di atas adalah definisi model Eloquent pada aplikasi Laravel yang disebut `Purchase`.

1. Pewarisan dan Penggunaan Fitur:

- Model `Purchase` menggunakan fitur-fitur dari trait `HasFactory`.
- Trait `HasFactory` menyediakan fungsi-fungsi factory yang membantu pembuatan data uji.

2. Atribut Fillable:

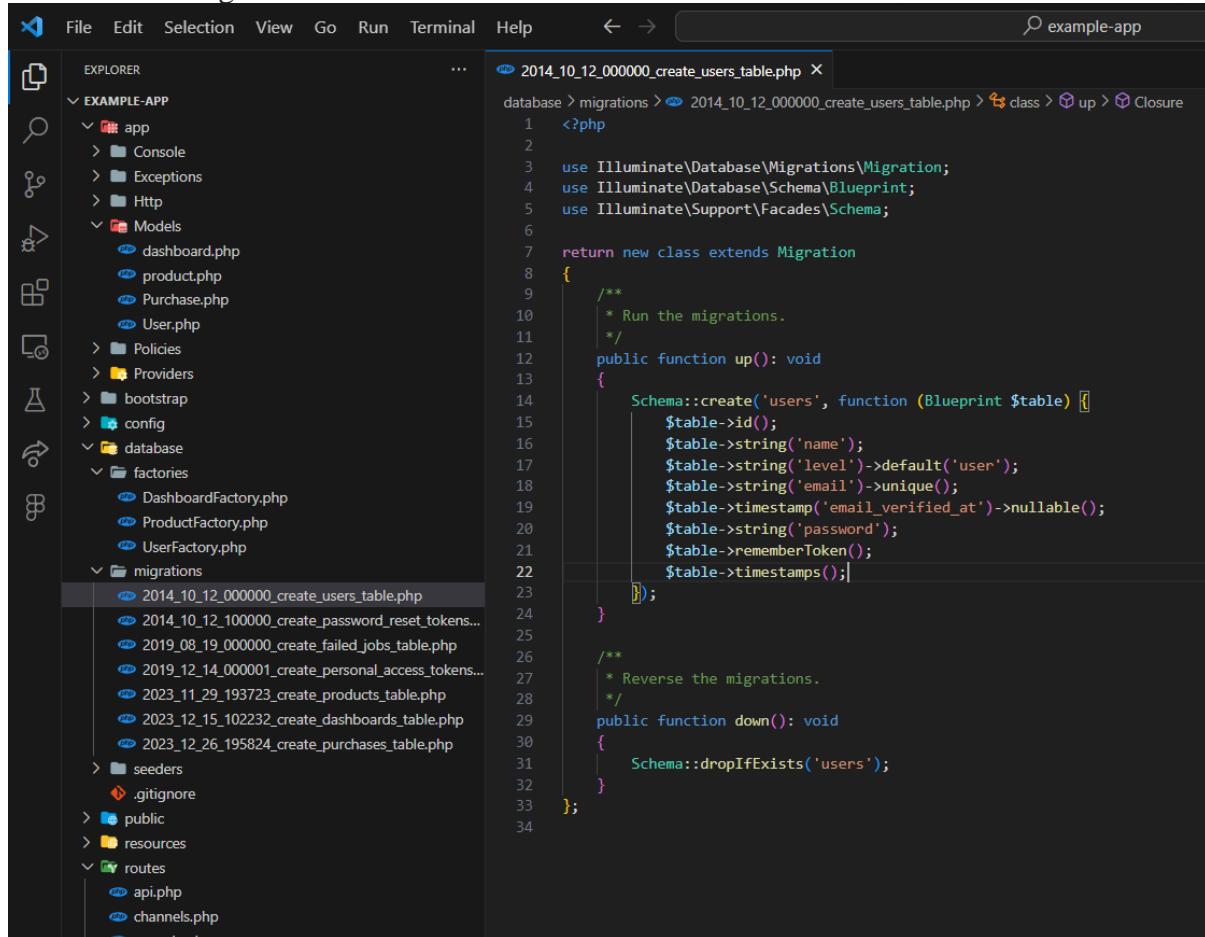
- Atribut fillable (`\$fillable`) menentukan kolom-kolom yang dapat diisi secara massal.
- Dalam hal ini, atribut fillable mencakup 'user_id' dan 'product_id'.

3. Relasi dengan Produk dan Pengguna:

- Model `Purchase` memiliki dua metode yang mendefinisikan relasi Eloquent:
- `product()`: Menunjukkan bahwa suatu pembelian dimiliki oleh satu produk (menggunakan relasi 'belongsTo' ke model `Product`).
- `user()`: Menunjukkan bahwa suatu pembelian dimiliki oleh satu pengguna (menggunakan relasi 'belongsTo' ke model `User`).

4) Migrations

1. Users migrations



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('users', function (Blueprint $table) [
15             $table->id();
16             $table->string('name');
17             $table->string('level')->default('user');
18             $table->string('email')->unique();
19             $table->timestamp('email_verified_at')->nullable();
20             $table->string('password');
21             $table->rememberToken();
22             $table->timestamps();
23         ]);
24     }
25
26     /**
27      * Reverse the migrations.
28      */
29     public function down(): void
30     {
31         Schema::dropIfExists('users');
32     }
33 };
34
```

Kode di atas adalah sebuah migrasi (migration) pada aplikasi Laravel yang bertanggung jawab untuk membuat dan menghapus tabel 'users' dalam basis data.

1. Metode `up`:

- Membuat tabel 'users' dengan menggunakan `Schema::create`.
- Tabel ini memiliki beberapa kolom seperti 'id', 'name', 'level', 'email', 'email_verified_at', 'password', 'rememberToken', dan dua kolom timestamp ('created_at' dan 'updated_at').

2. Metode `down`:

- Menghapus tabel 'users' menggunakan `Schema::dropIfExists`.

3. Kolom 'level':

- Kolom 'level' memiliki nilai default 'user', yang berarti jika tidak ada nilai yang disediakan, nilai default ini akan digunakan.

4. Kolom 'email_verified_at':

- Kolom 'email_verified_at' bersifat nullable, artinya dapat memiliki nilai null, dan biasanya digunakan untuk menyimpan waktu verifikasi email jika diperlukan.

5. Remember Token:

- Kolom 'rememberToken' digunakan untuk mengimplementasikan fitur "Remember Me" dalam autentikasi.

Catatan:

- Kode migrasi ini menciptakan tabel 'users' yang dapat digunakan untuk menyimpan informasi pengguna.
 - Migration digunakan untuk mengatur struktur basis data dan dapat dijalankan atau di-rollback untuk memodifikasinya sesuai kebutuhan aplikasi.

2. Products Migrations

The screenshot shows a Visual Studio Code interface with a Laravel application named 'EXAMPLE-APP'. The left sidebar displays the project structure, including 'Console', 'Exceptions', 'Http', 'Models' (containing 'dashboard.php', 'product.php', 'Purchase.php', 'User.php'), 'Policies', 'Providers', 'bootstrap', 'config', 'database' (containing 'factories' with 'DashboardFactory.php', 'ProductFactory.php', 'UserFactory.php'), 'migrations' (containing several migration files like '2014_10_12_000000_create_users_table.php', '2014_10_12_100000_create_password_reset_tokens...', '2019_08_19_000000_create_failed_jobs_table.php', '2019_12_14_000001_create_personal_access_tokens...', '2023_11_29_193723_create_products_table.php'), 'seeders', '.gitignore', 'public', 'resources', and 'routes'. The right pane shows the content of the selected migration file, '2023_11_29_193723_create_products_table.php'. The code defines a migration class that creates a 'products' table with columns for id, gambar (nullable string), nama (text), deskripsi (text), and harga (big integer). It also includes a down() method to drop the table if it exists.

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('products', function (Blueprint $table) {
16             $table->id();
17             $table->string('gambar')->nullable();
18             $table->string('nama');
19             $table->text('deskripsi');
20             $table->bigInteger('harga');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30         Schema::dropIfExists('products');
31     }
32 };
```

Kode di atas adalah migrasi (migration) pada aplikasi Laravel yang bertanggung jawab untuk membuat dan menghapus tabel 'products' dalam basis data.

1. Metode `up`:

- Membuat tabel 'products' dengan menggunakan `Schema::create`.
 - Tabel ini memiliki beberapa kolom seperti 'id', 'gambar', 'nama', 'deskripsi', 'harga', dan dua kolom timestamp ('created_at' dan 'updated_at').

2. Metode `down`:

- Menghapus tabel 'products' menggunakan `Schema::dropIfExists`.

3. Kolom 'gambar':

- Kolom 'gambar' bersifat nullable, artinya bisa memiliki nilai null.
 - Kolom ini digunakan untuk menyimpan nama berkas gambar terkait produk.

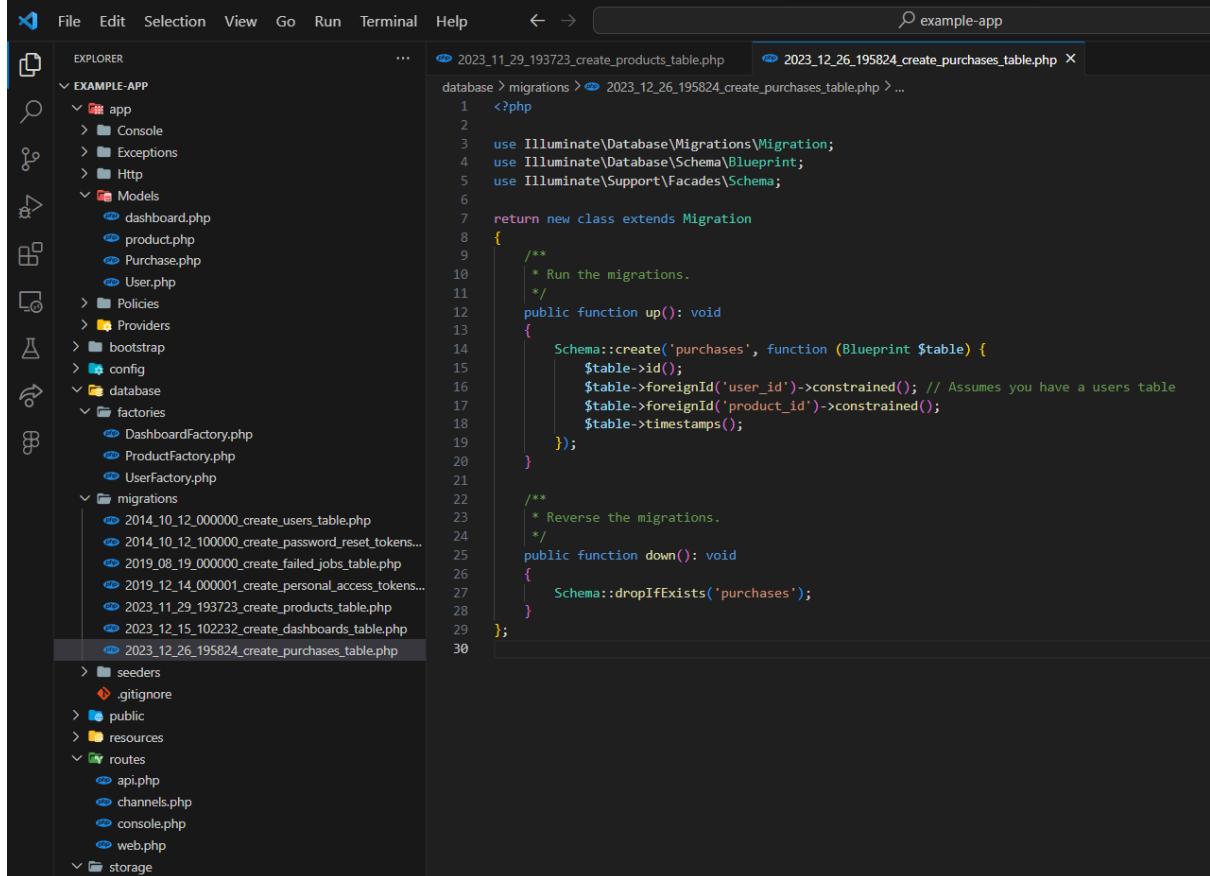
4. Kolom 'harga':

- Kolom 'harga' memiliki tipe data 'bigInteger', yang biasanya digunakan untuk menyimpan nilai harga yang besar.

Catatan:

- Kode migrasi ini menciptakan tabel 'products' yang dapat digunakan untuk menyimpan informasi tentang produk.

3. Purchases Migrations



```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('purchases', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained(); // Assumes you have a users table
            $table->foreignId('product_id')->constrained();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('purchases');
    }
};

```

Kode di atas adalah migrasi (migration) pada aplikasi Laravel yang bertanggung jawab untuk membuat dan menghapus tabel 'purchases' dalam basis data.

1. Metode `up`:

- Membuat tabel 'purchases' dengan menggunakan `Schema::create`.
- Tabel ini memiliki beberapa kolom seperti 'id', 'user_id', 'product_id', dan dua kolom timestamp ('created_at' dan 'updated_at').
- Kolom 'user_id' dan 'product_id' adalah kunci asing (foreign key) yang terhubung ke tabel 'users' dan 'products', masing-masing.

2. Metode `down`:

- Menghapus tabel 'purchases' menggunakan `Schema::dropIfExists`.

3. Kolom 'user_id' dan 'product_id':

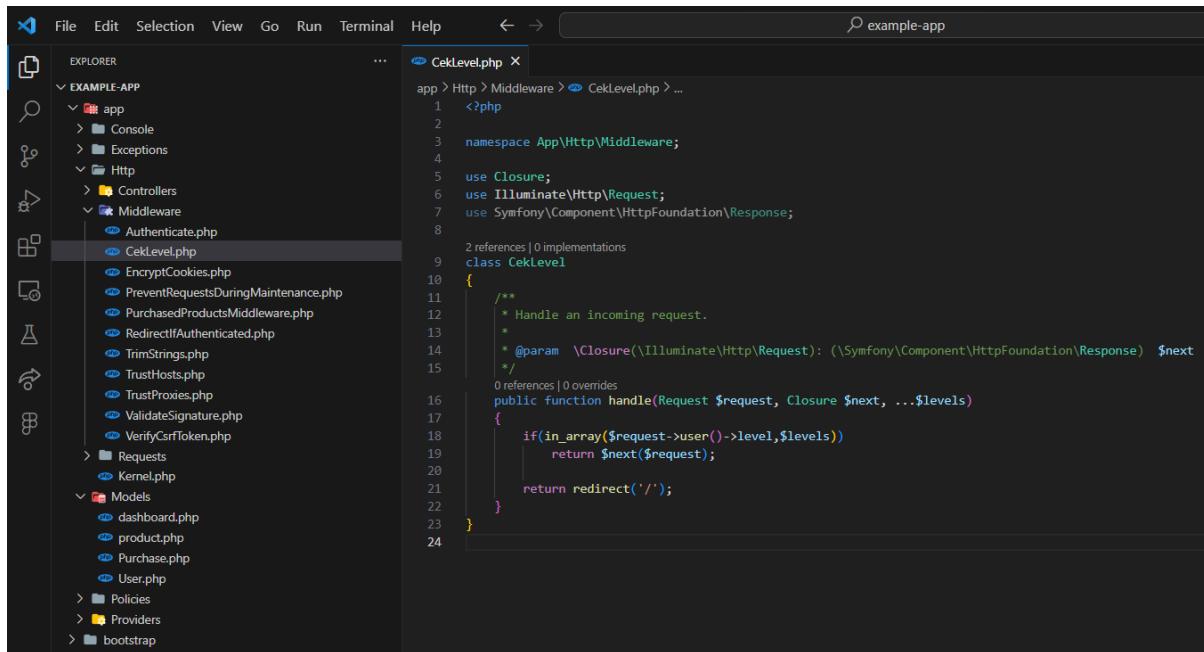
- Kedua kolom ini merupakan kunci asing (foreign key) yang menggunakan metode `constrained()` untuk mengaitkan tabel 'purchases' dengan tabel 'users' dan 'products'.
- `constrained()` memastikan bahwa nilai di dalam kolom 'user_id' dan 'product_id' harus ada di kolom 'id' masing-masing tabel terkait.

Catatan:

- Tabel 'purchases' ini umumnya digunakan untuk merepresentasikan pembelian atau transaksi antara pengguna dan produk.

5) Middleware

1. CekLevel Middleware



The screenshot shows a code editor with the file `CekLevel.php` open. The file is located in the `app/Http/Middleware` directory. The code defines a middleware class `CekLevel` that checks if a user's level is in a specified array before proceeding to the next middleware or controller action.

```
<?php  
namespace App\Http\Middleware;  
  
use Closure;  
use Illuminate\Http\Request;  
use Symfony\Component\HttpFoundation\Response;  
  
class CekLevel  
{  
    /**  
     * Handle an incoming request.  
     *  
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next  
     */  
    public function handle(Request $request, Closure $next, ...$levels)  
    {  
        if(in_array($request->user()->level,$levels))  
            return $next($request);  
  
        return redirect('/');  
    }  
}
```

Kode di atas adalah contoh middleware pada aplikasi Laravel yang disebut `CekLevel`. Middleware digunakan untuk memproses permintaan HTTP sebelum mencapai penanganan oleh kontroler.

1. Metode `handle`:

- Metode ini menerima permintaan (`Request`), penanganan berikutnya (`Closure \$next`), dan parameter `...\$levels` yang merupakan tingkat level yang diizinkan.
- Memeriksa apakah tingkat level pengguna saat ini ada dalam daftar tingkat level yang diizinkan.
- Jika iya, membiarkan permintaan melanjutkan ke penanganan berikutnya dengan menggunakan `'\$next(\$request)`.
- Jika tidak, melakukan pengalihan (redirect) ke halaman utama ('/').

Catatan:

- Middleware ini digunakan untuk membatasi akses berdasarkan tingkat level pengguna.
- Middleware dapat digunakan di rute atau grup rute tertentu untuk memberlakukan pembatasan tingkat level pada akses ke halaman-halaman tertentu.

6) ENV

The screenshot shows a code editor interface with the title bar "example-app". The left sidebar is the "EXPLORER" view, showing the project structure of "EXAMPLE-APP" which includes "Requests", "Models", "Policies", "Providers", "bootstrap", "config", "database", "public", "resources", "routes", "storage", "app", "framework", "logs", "tests", "vendor", ".editorconfig", ".env.example", ".gitattributes", ".gitignore", "artisan", "composer.json", "composer.lock", "package.json", "phpunit.xml", "README.md", and "vite.config.js". The right pane is the "CekLevel.php" file, with the ".env" tab selected. The code in the .env file is as follows:

```
1 # konfigurasi publik pada image
2 FILESYSTEM_DISK = public
3 |
4 APP_NAME=Laravel
5 APP_ENV=local
6 APP_KEY=base64:LvCVXhFxJAyEHgiwpud+65vjcYjTHm2ec8IDJ3EfgXc=
7 APP_DEBUG=true
8 APP_URL=http://localhost
9
10 LOG_CHANNEL=stack
11 LOG_DEPRECATIONS_CHANNEL=null
12 LOG_LEVEL=debug
13
14 DB_CONNECTION=mysql
15 DB_HOST=127.0.0.1
16 DB_PORT=3306
17 DB_DATABASE=motor_sport
18 DB_USERNAME=root
19 DB_PASSWORD=
20
21 BROADCAST_DRIVER=log
22 CACHE_DRIVER=file
23 FILESYSTEM_DISK=local
24 QUEUE_CONNECTION=sync
25 SESSION_DRIVER=file
26 SESSION_LIFETIME=120
27
28 MEMCACHED_HOST=127.0.0.1
29
30 REDIS_HOST=127.0.0.1
31 REDIS_PASSWORD=null
32 REDIS_PORT=6379
33
34 MAIL_MAILER=smtp
35 MAIL_HOST=mailpit
36 MAIL_PORT=1025
37 MAIL_USERNAME=null
38 MAIL_PASSWORD=null
39 MAIL_ENCRYPTION=null
40 MAIL_FROM_ADDRESS="hello@example.com"
41 MAIL_FROM_NAME="${APP_NAME}"
42
43 AWS_ACCESS_KEY_ID=
44 AWS_SECRET_ACCESS_KEY=
45 AWS_DEFAULT_REGION=us-east-1
46 AWS_BUCKET=
47 AWS_USE_PATH_STYLE_ENDPOINT=false
48
```

File yang Anda berikan adalah berkas konfigurasi lingkungan (`.env`) pada aplikasi Laravel.

1. `FILESYSTEM_DISK`

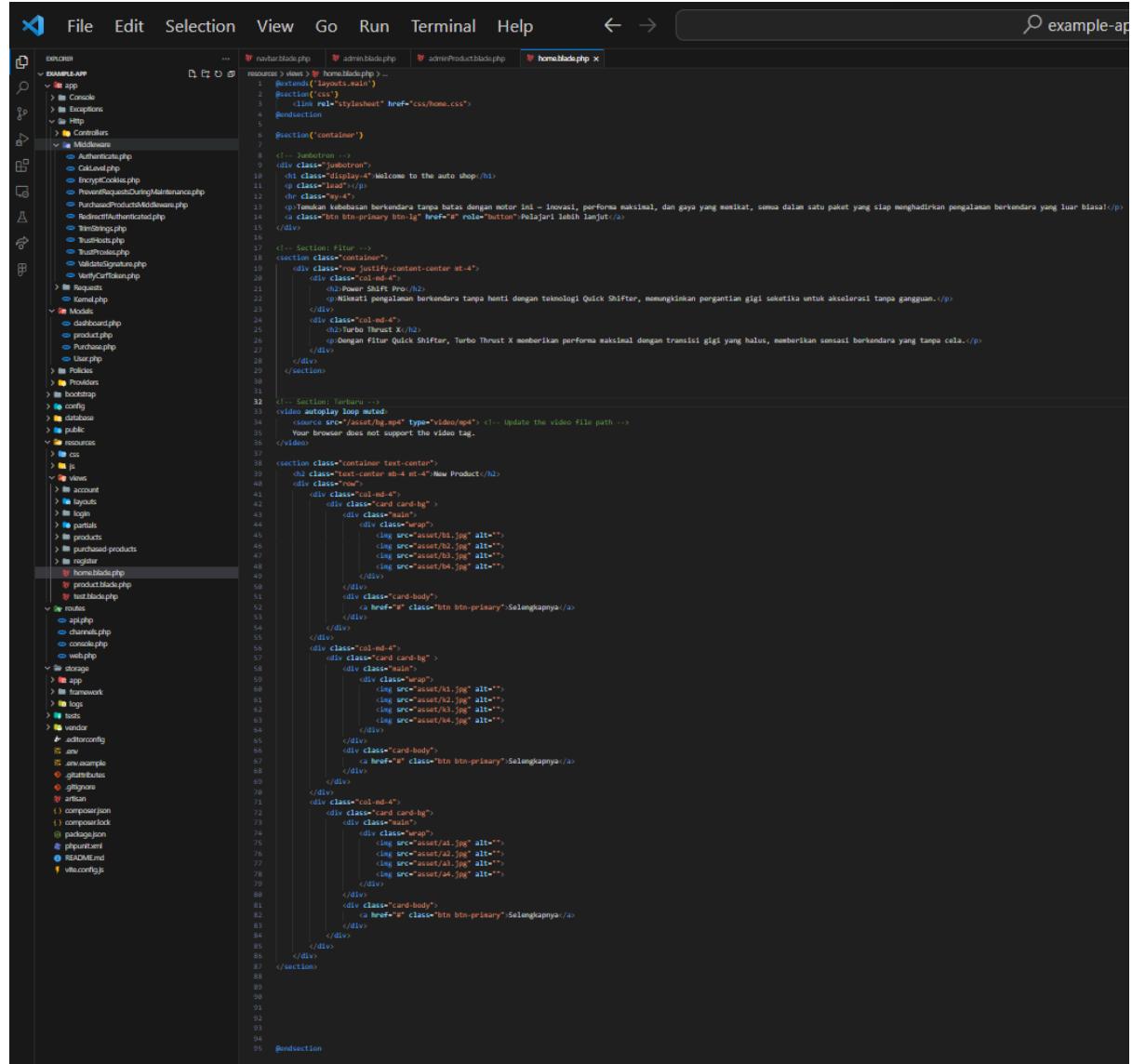
- Kunci ini menentukan driver sistem file yang digunakan oleh Laravel.
- Nilai `public` menunjukkan bahwa sistem file yang digunakan adalah public disk, yang biasanya digunakan untuk menyimpan file-file statis yang dapat diakses publik.

2. Pengaturan Database (`DB_CONNECTION`, `DB_HOST`, `DB_PORT`, `DB_DATABASE`, `DB_USERNAME`, `DB_PASSWORD`)

- Menentukan parameter koneksi basis data seperti host, port, nama basis data, nama pengguna, dan kata sandi.

7) VIEW

1. Home



```
resources/views/home.blade.php
1 <extends('layouts.main')>
2 <section>('css')
3   <link rel="stylesheet" href="css/home.css">
4 </section>
5 <section>('container')>
6   <div>('jumbotron')>
7     <div>('jumbottron')>
8       <h1>Welcome to the auto shop</h1>
9       <p>lead</p>
10      <br>
11      <p>Temukan kebutuhan berkendara tanpa batas dengan motor ini - inovasi, performa maksimal, dan gaya yang manis, semua dalam satu paket yang siap menghadirkan pengalaman berkendara yang luar biasa!</p>
12      <a href="#" class="btn btn-primary btn-lg" role="button">Pelajari lebih lanjut</a>
13    </div>
14  </div>
15 </div>
16 <!-- Section: Filter -->
17 <div>('row justify-content-center')>
18   <div>('col-md-4')>
19     <h2>Power Shift Pro</h2>
20     <p>Miliki pengalaman berkendara tanpa henti dengan teknologi Quick Shifter, memungkinkan pergantian gigi seketika untuk akselerasi tanpa gangguan.</p>
21   </div>
22 </div>
23 <div>('col-md-4')>
24   <h2>Turbo Thrust X</h2>
25   <p>Dengan fitur Quick Shifter, Turbo Thrust X memberikan performa maksimal dengan transisi gigi yang halus, memberikan sensasi berkendara yang tanpa cela.</p>
26 </div>
27 </div>
28 </div>
29 </div>
30 </div>
31 <!-- Section: Return -->
32 <div>('autoplay loop muted')>
33   <video src="asset/bg.mp4" type="video/mp4"> <!-- Update the video file path -->
34   Your browser does not support the video tag.
35 </video>
36 </div>
37 <div>('text-center')>
38   <h2>New Product</h2>
39   <div>('row')>
40     <div>('col-md-4')>
41       <div>('card card-bg')>
42         <div>('card-body')>
43           
44           
45           
46           
47         </div>
48       </div>
49     </div>
50     <div>('col-md-4')>
51       <div>('card card-bg')>
52         <div>('card-body')>
53           <a href="#" class="btn btn-primary">Selengkapnya</a>
54         </div>
55       </div>
56     </div>
57     <div>('col-md-4')>
58       <div>('card card-bg')>
59         <div>('main')>
60           <div>('wrap')>
61             
62             
63             
64             
65           </div>
66         </div>
67         <div>('card-body')>
68           <a href="#" class="btn btn-primary">Selengkapnya</a>
69         </div>
70       </div>
71     </div>
72     <div>('col-md-4')>
73       <div>('card card-bg')>
74         <div>('main')>
75           <div>('wrap')>
76             
77             
78             
79             
80           </div>
81         </div>
82       </div>
83     </div>
84   </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
```

Kode yang Anda berikan adalah tampilan Blade pada aplikasi Laravel yang mencakup halaman utama dengan jumbotron, sejumlah fitur, dan bagian terbaru.

1. Pemanggilan CSS Tambahan:

- Menggunakan `@section('css')` untuk memanggil file CSS tambahan ('home.css') yang diletakkan di folder 'css'.

2. **Jumbotron:**

- Bagian ini menampilkan jumbotron dengan pesan selamat datang dan tombol "Pelajari lebih lanjut".

3. **Section Fitur:**

- Menampilkan sejumlah fitur dengan judul dan deskripsi, seperti "Power Shift Pro" dan "Turbo Thrust X".

4. **Section Terbaru:**

- Menampilkan video latar belakang (background video) dengan sumber dari file MP4.
- Berisi bagian "New Product" yang menampilkan produk baru dalam bentuk kartu (card). Setiap kartu memiliki gambar dan tombol "Selengkapnya".

5. Video Latar Belakang:

- Menggunakan elemen `<video>` untuk memasukkan video latar belakang dengan file MP4 sebagai sumber. Jika browser tidak mendukung video tag, pesan "Your browser does not support the video tag." akan ditampilkan.

6. Container dan Row:

- Menggunakan Bootstrap untuk mengatur struktur grid dengan `container` dan `row`.

7. Card Component:

- Menerapkan komponen kartu Bootstrap untuk menampilkan produk baru dengan gambar dan tombol "Selengkapnya".

2. Product

```

File Edit Selection View Go Run Terminal Help ← → 🔍 example-app
EXPLORER resources > views > product.blade.php ...
Authenticat...
CekLevel...
EncryptCookies...
PreventRequestsDuringMaintenance...
PurchasedProductsMiddleware...
RedirectIfAuthenticated...
TrimStrings...
TrustHosts...
TrustProxies...
ValidateSignature...
VerifyCsrfToken...
Requests Kernel.php
Models
  - dashboard.php
  - product.php
  - Purchase.php
  - User.php
Policies
Providers
bootstrap
config
product.blade.php
  1 @extends('layouts.main')
  2 @section('css')
  3 <link rel="stylesheet" href="css/product.css">
  4 @endsection
  5
  6 @section('content')
  7 <section class="content">
  8 @foreach ($products as $product)
  9   <div class="product">
  10     @if ($product->gambar)
  11       
  14     @endif
  15     <h2>{{ $product['nama'] }}</h2>
  16     <p>{{ $product['deskripsi'] }}</p>
  17     <p>Harga: {{ $product['harga'] }}</p>
  18     <a href="{{ url('/products/' . $product->id) }}" class="beli-btn">Order</a>
  19   </div>
  20 @endforeach
  21 </section>
  22
  23 @endsection

```

Kode Blade yang Anda berikan adalah tampilan produk

1. Pemanggilan CSS Tambahan:

- Menggunakan `@section('css')` untuk memanggil file CSS tambahan (`product.css`) yang diletakkan di folder 'css'.

2. Loop Produk:

- Menggunakan `@foreach (\$products as \$product)` untuk melakukan loop melalui array produk yang diterima dari controller.

3. Div Product:

- Setiap produk dibungkus dalam elemen `<div class="product">`.

4. Gambar Produk:

- Memeriksa apakah produk memiliki gambar (`\$product->gambar`) menggunakan kondisi `@if`. Jika ya, menampilkan gambar menggunakan asset Laravel (`asset('storage/' . \$product->gambar)`). Jika tidak, menampilkan gambar dari URL yang diberikan (`\$product['gambar']`).

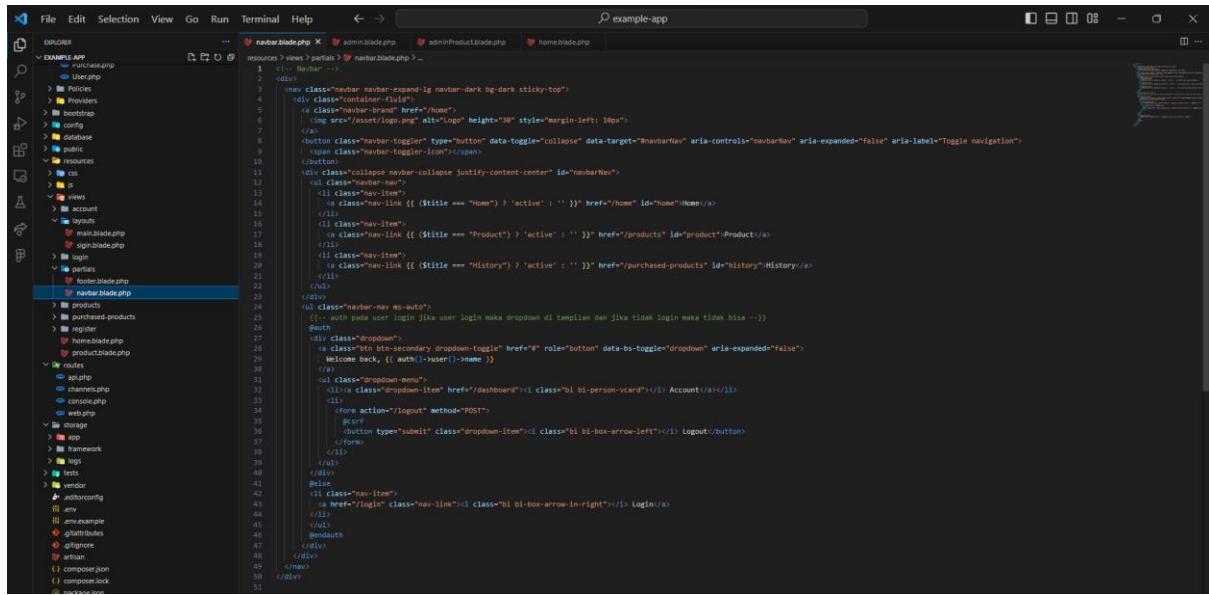
5. Informasi Produk:

- Menampilkan informasi produk seperti nama, deskripsi, dan harga menggunakan tag HTML sesuai

6. Link Order:

- Menyertakan tombol "Order" yang mengarahkan pengguna ke halaman spesifik untuk memesan produk tertentu. Link diatur dengan `href="{{ url('/products/' . \$product->id) }}`.

3. Navbar



```

<nav class="navbar navbar-expand-lg navbar-dark bg-dark sticky-top">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">
      
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse justify-content-center" id="navbarSupportedContent">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" {{ ($title === "Home") ? 'active' : '' }} href="/home" id="Home">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" {{ ($title === "Product") ? 'active' : '' }} href="/products" id="product">Product</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" {{ ($title === "History") ? 'active' : '' }} href="/purchased-products" id="history">History</a>
        </li>
      </ul>
    </div>
    <div class="dropdown">
      <!-- auto pada user login jika user login maka dropdown di tampilkan dan jika tidak login maka tidak bisa -->
      <!-- auth-->
      <div class="dropdown">
        <a class="btn btn-secondary dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
          Welcome back, {{ auth()->user()?-name() }}>
        </a>
        <ul class="dropdown-menu">
          <li><a href="#">Account</a></li>
          <li><a href="#">Logout</a></li>
        </ul>
      </div>
    </div>
  </div>
</nav>

```

Kode di atas merupakan bagian dari tampilan navbar

1. Logo dan Toggle Button:

- Menggunakan tag `<nav>` untuk mengelompokkan elemen navbar.

- Menampilkan logo pada sisi kiri navbar dengan menggunakan tag ``.

2. Toggle Button untuk Responsiveness:

- Menambahkan tombol toggler (`<button class="navbar-toggler" ...>`) untuk responsivitas pada tampilan layar kecil.

3. Menu Navigasi:

- Menggunakan `<div class="collapse navbar-collapse justify-content-center" ...>` untuk mengelompokkan menu navigasi.
- Menampilkan menu navigasi dengan menggunakan tag `<ul class="navbar-nav">` dan masing-masing `<li class="nav-item">`.

4. Link Navigasi:

- Setiap link navigasi (``) memiliki kelas yang disesuaikan dengan kondisi yang ditentukan oleh variabel `\$title`. Jika `\$title` sesuai dengan halaman yang bersangkutan, maka link tersebut akan memiliki kelas `active`.

5. Dropdown Menu (Jika User Login):

- Menggunakan dropdown untuk menyambut pengguna yang sudah login.
- Menampilkan nama pengguna dan dropdown menu dengan opsi "Account" dan "Logout". Tombol "Logout" berupa formulir POST untuk keamanan.

6. Link Login (Jika User Belum Login):

- Menampilkan tombol "Login" jika pengguna belum login.

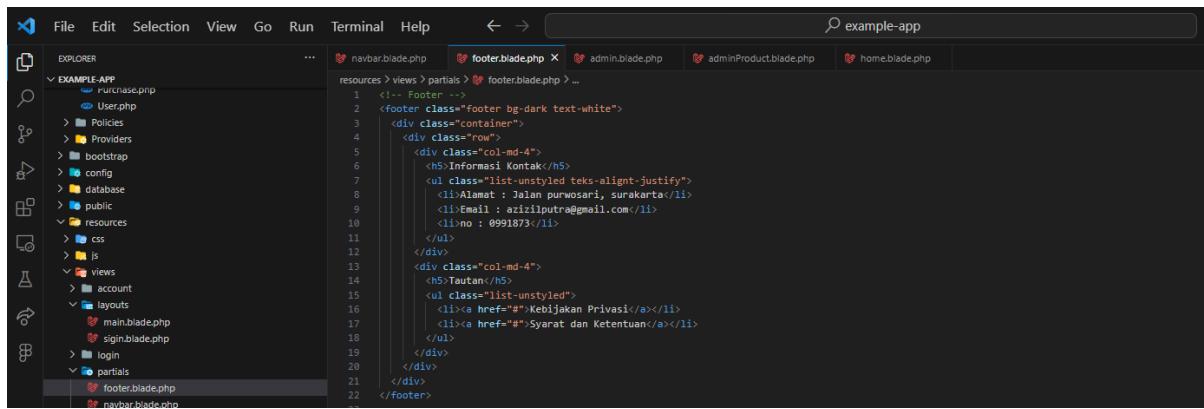
7. Auth Check (`@auth` dan `@endauth`):

- Menggunakan `@auth` dan `@endauth` untuk mengecek status autentikasi. Jika pengguna telah login, maka tampilkan dropdown; jika belum, tampilkan tombol "Login".

8. Icon Menggunakan Bootstrap Icons:

- Menggunakan ikon dari Bootstrap Icons (contoh: `<i class="bi bi-person-vcard"></i>`).

4. Footer



The screenshot shows the VS Code interface with the 'example-app' project open. The 'EXPLORER' sidebar on the left lists files and folders, including 'Purchase.php', 'User.php', 'Policies', 'Providers', 'bootstrap', 'config', 'database', 'public', 'resources', 'css', 'js', 'views', 'layouts', 'main.blade.php', 'signin.blade.php', 'login', 'partials', 'footer.blade.php', and 'navbar.blade.php'. The 'FOOTER' tab in the top bar is active, and the code editor displays the 'footer.blade.php' file:

```
<!-- Footer -->
<footer class="footer bg-dark text-white">
    <div class="container">
        <div class="row">
            <div class="col-md-4">
                <h5>Informasi Kontak</h5>
                <ul class="list-unstyled teks-align-justify">
                    <li>Alamat : Jalan purwosari, surakarta</li>
                    <li>Email : azizilputra@gmail.com</li>
                    <li>No : 0991873</li>
                </ul>
            </div>
            <div class="col-md-4">
                <h5>Tautan</h5>
                <ul class="list-unstyled">
                    <li><a href="#">Kebijakan Privasi</a></li>
                    <li><a href="#">Syarat dan Ketentuan</a></li>
                </ul>
            </div>
        </div>
    </div>
</footer>
```

Kode di atas merupakan bagian dari tampilan footer pada aplikasi Laravel.

1. Footer Container:

- Menggunakan tag `<footer class="footer bg-dark text-white">` untuk membuat bagian footer dengan latar belakang gelap dan teks berwarna putih.
- Diberikan kelas `container` untuk mengatur lebar dan penempatan kontennya.

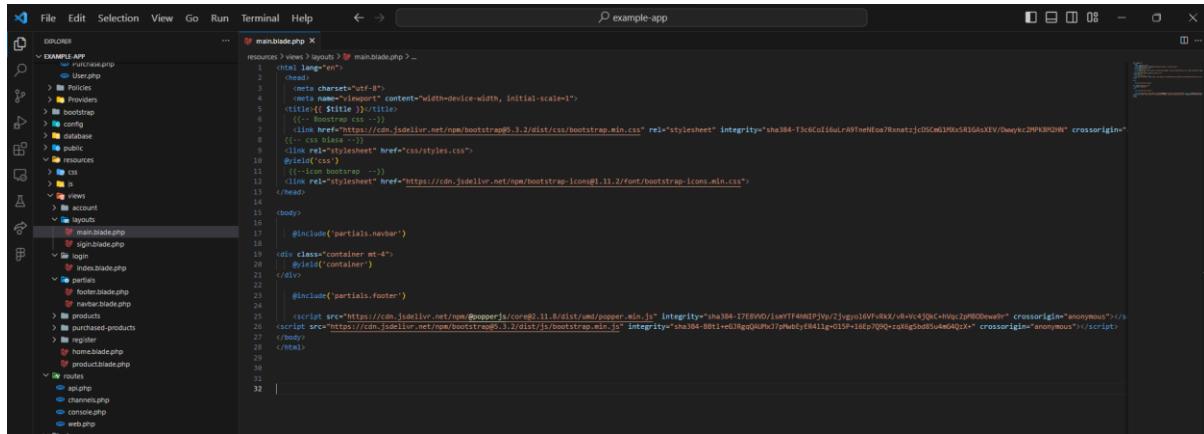
2. Kolom Informasi Kontak:

- Bagian ini memiliki dua kolom (`<div class="col-md-4">`) untuk menampilkan informasi kontak dan tautan.
- Kolom pertama menampilkan informasi kontak seperti alamat, email, dan nomor telepon.

3. Kolom Tautan:

- Kolom kedua menampilkan tautan dengan judul "Tautan". Tautan ini dapat berupa kebijakan privasi, syarat, dan ketentuan.
- Menggunakan tag `<ul class="list-unstyled">` untuk membuat daftar tautan tanpa gaya bawaan dari browser.

5. Main



The screenshot shows the VS Code interface with the 'example-app' project open. The 'EXPLORER' sidebar on the left lists files and folders, including 'Purchase.php', 'User.php', 'Policies', 'Providers', 'bootstrap', 'config', 'database', 'public', 'resources', 'css', 'js', 'views', 'layouts', 'main.blade.php', 'signin.blade.php', 'login', 'partials', 'footer.blade.php', 'navbar.blade.php', and 'PurchaseController.php'. The 'MAIN' tab in the top bar is active, and the code editor displays the 'main.blade.php' file:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>{{ $title }}</title>
        {{-- Bootstrap CSS --}}
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3cD7sWu6kH3+XgQ0B8eHNTvQZl0OjZJLZPfjtH+oRqzJGnVYUOyKzHPP3M2H" crossorigin="anonymous">
        {{-- CSS --}}
        <link rel="stylesheet" href="css/styles.css" style="background-color: #f0f0f0; color: #333; font-size: 1em; font-weight: bold; margin-bottom: 10px;">
        {{-- Bootstrap Icons --}}
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.2/font/bootstrap-icons.min.css" style="background-color: #f0f0f0; color: #333; font-size: 1em; font-weight: bold; margin-bottom: 10px;">
    </head>
    <body>
        @include('partials.navbar')
        <div class="container pt-4">
            @yield('content')
        </div>
        @include('partials.footer')
        {{-- Scripts --}}
        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-T7eBQ2TDnuHqKLw08A51P2CXeu4qJ7e71o2d4t2zI5Etc/4eBtk4by4UMZo" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js" integrity="sha384-EEtuQ100zu4wF4Hgq6Ckxh/Z/0Q+z0igSbdSu4d4QzX=</script>
    </body>
</html>
```

Kode di atas merupakan struktur dasar HTML dari halaman web dalam framework Laravel.

1. Tag `<html>`:

- Attribut `lang="en"` menunjukkan bahwa bahasa yang digunakan adalah bahasa Inggris.

2. Tag `<head>`:

- Menyertakan tag `<meta>` untuk mengatur karakter set dan tampilan responsif pada perangkat seluler.
- Tag `<title>` digunakan untuk menampilkan judul halaman sesuai dengan variabel `'\$title'`.
- Menyertakan link CSS dari Bootstrap dan file lokal "styles.css" untuk mengatur tata letak dan tampilan halaman.
- Menggunakan ikon Bootstrap dengan link ke CDN Bootstrap Icons.

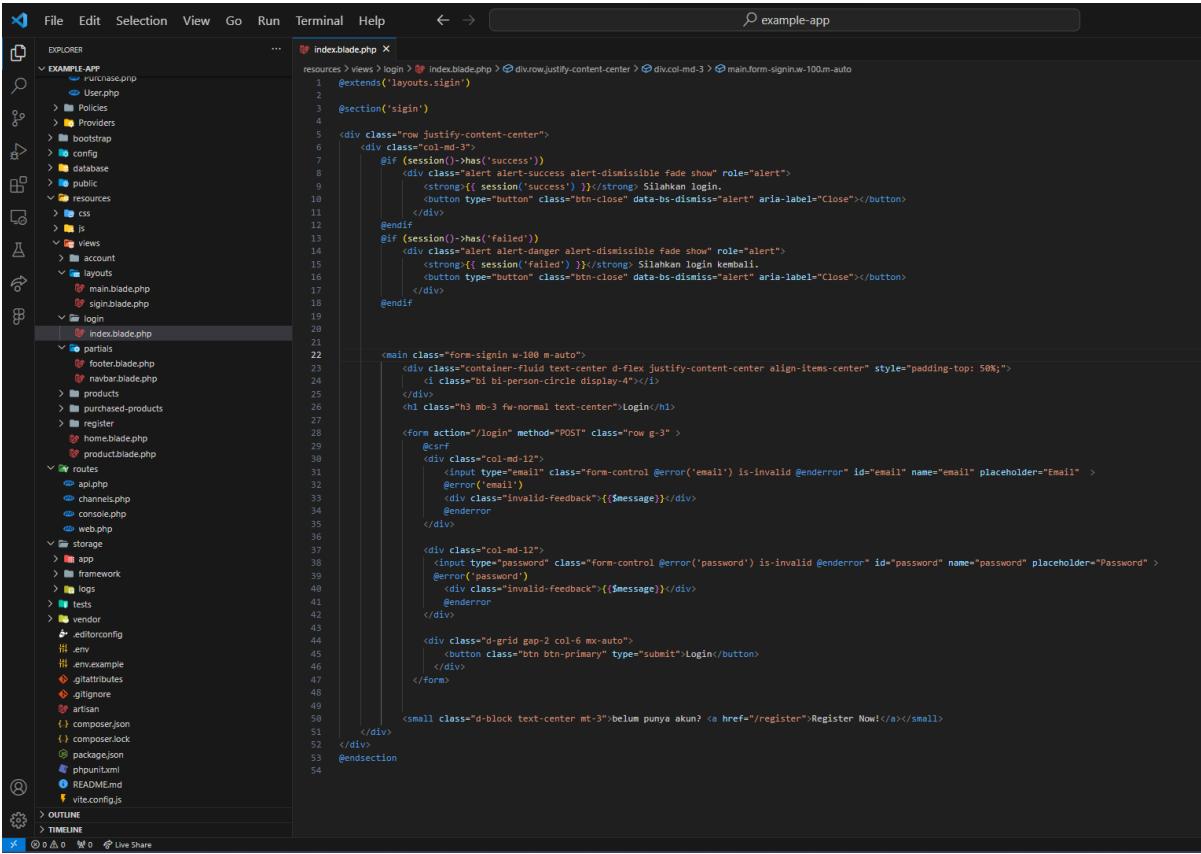
3. Tag `<body>`:

- Menggunakan directive Blade `@include` untuk menyertakan navbar dan footer yang mungkin merupakan bagian dari halaman template yang lebih besar.
- Tag `<div class="container mt-4">` digunakan untuk memberikan padding dan margin atas (top margin) pada konten halaman.

4. Tag `<script>`:

- Menyertakan script dari Bootstrap dan Popper.js yang diperlukan untuk fitur Bootstrap seperti dropdown dan modal.

6. Login



```
File Edit Selection View Go Run Terminal Help ... index.blade.php x resources > views > login > index.blade.php > div.row.justify-content-center > div.col-md-3 > main.form-signin.w-100.m-auto
1 @extends('layouts.sigin')
2
3 @section('signin')
4
5 <div class="row justify-content-center">
6   <div class="col-md-3">
7     @if ($session->has('success'))
8       <div class="alert alert-success alert-dismissible fade show" role="alert">
9         <strong>{{ $session['success'] }}</strong> Silahkan login.
10        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
11      </div>
12    @endif
13    @if ($session->has('failed'))
14      <div class="alert alert-danger alert-dismissible fade show" role="alert">
15        <strong>{{ $session['failed'] }}</strong> Silahkan login kembali.
16        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
17      </div>
18    @endif
19
20
21
22 <main class="form-signin w-100 m-auto">
23   <div class="container-fluid text-center d-flex justify-content-center align-items-center" style="padding-top: 50%;">
24     <i class="bi bi-person-circle display-4"></i>
25   <h1 class="h3 mb-3 fw-normal text-center">Login</h1>
26
27   <form action="/login" method="POST" class="row g-3">
28     @csrf
29     <div class="col-md-12">
30       <input type="email" class="form-control @error('email') is-invalid @enderror" id="email" name="email" placeholder="Email" >
31       @error('email')
32         <div class="invalid-feedback">{{ $message }}</div>
33       @enderror
34     </div>
35
36     <div class="col-md-12">
37       <input type="password" class="form-control @error('password') is-invalid @enderror" id="password" name="password" placeholder="Password" >
38       @error('password')
39         <div class="invalid-feedback">{{ $message }}</div>
40       @enderror
41     </div>
42
43     <div class="d-grid gap-2 col-6 mx-auto">
44       <button class="btn btn-primary" type="submit">Login</button>
45     </div>
46   </form>
47
48   <small class="d-block text-center mt-3">Belum punya akun? <a href="/register">Register Now!</a></small>
49
50 </div>
51 </div>
52 </div>
53 @endsession
54
```

Kode di atas merupakan halaman Blade untuk menampilkan formulir login.

1. Directive Blade `@extends` dan `@section`:

- Mendefinisikan bahwa halaman ini menggunakan layout "sigin" yang mungkin berisi struktur umum untuk halaman login.
- Mendefinisikan bagian "sigin" dengan menggunakan `@section('sigin')`.

2. **Alerts:**

- Menampilkan pesan sukses atau pesan gagal jika sesi memiliki kunci 'success' atau 'failed'. Menggunakan Bootstrap untuk tampilan alert.

3. Formulir Login:

- Membuat formulir login dengan menggunakan tag `<form>` yang mengarah ke rute "/login".
- Menggunakan metode POST dan menyertakan direktif Blade `@csrf` untuk melindungi dari serangan CSRF.
- Menampilkan input untuk email dan password, dengan memberikan class `is-invalid` pada input yang tidak valid dan menampilkan pesan error dengan menggunakan `@error`.
- Menggunakan Bootstrap untuk styling dan tata letak.

- Menampilkan tombol "Login" dengan class Bootstrap `btn btn-primary`.

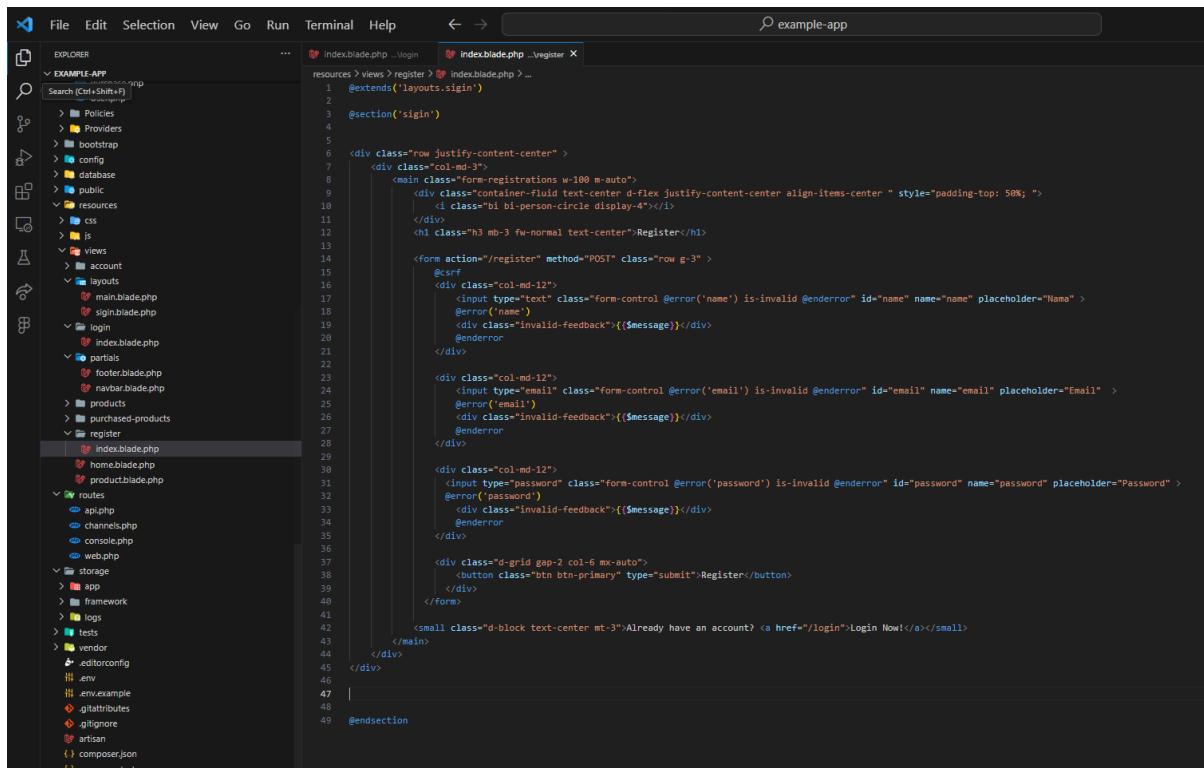
4. Tautan Register:

- Menampilkan tautan ke halaman register ("register") jika pengguna belum memiliki akun.

5. Styling:

- Menggunakan ikon Bootstrap (`bi bi-person-circle`) sebagai ikon pengguna di tengah halaman.
- Mengatur tata letak dengan menggunakan class-class Bootstrap seperti `container`, `row`, `col-md-3`, dan `d-grid`.

7. Register



```

<?php

@extends('layouts.sigin')
@section('sigin')



<?php
    <div class="col-md-12">
        <input type="text" class="form-control @error('name') is-invalid @enderror" id="name" name="name" placeholder="Name" value="<?php echo old('name'); >">
        <error('name')>
            <div class="invalid-feedback">{{ $message }}</div>
        </error>
    </div>

    <div class="col-md-12">
        <input type="email" class="form-control @error('email') is-invalid @enderror" id="email" name="email" placeholder="Email" value="<?php echo old('email'); >">
        <error('email')>
            <div class="invalid-feedback">{{ $message }}</div>
        </error>
    </div>

    <div class="d-grid gap-2 col-6 mx-auto">
        <button class="btn btn-primary" type="submit">Register</button>
    </div>


```

Kode di atas merupakan halaman Blade untuk menampilkan formulir registrasi. Berikut

1. Directive Blade `@extends` dan `@section`:

- Mendefinisikan bahwa halaman ini menggunakan layout "sigin" yang mungkin berisi struktur umum untuk halaman registrasi.
- Mendefinisikan bagian "sigin" dengan menggunakan `@section('sigin')`.

2. Formulir Registrasi:

- Membuat formulir registrasi dengan menggunakan tag `<form>` yang mengarah ke rute "/register".

- Menggunakan metode POST dan menyertakan direktif Blade `@csrf` untuk melindungi dari serangan CSRF.
 - Menampilkan input untuk nama, email, dan password, dengan memberikan class `is-invalid` pada input yang tidak valid dan menampilkan pesan error dengan menggunakan `@error`.
 - Menggunakan Bootstrap untuk styling dan tata letak.
 - Menampilkan tombol "Register" dengan class Bootstrap `btn btn-primary`.

3. Tautan Login:

- Menampilkan tautan ke halaman login ("login") jika pengguna sudah memiliki akun.

4. Styling:

- Menggunakan ikon Bootstrap (`bi bi-person-circle`) sebagai ikon pengguna di tengah halaman.
 - Mengatur tata letak dengan menggunakan class-class Bootstrap seperti `container`, `row`, `col-md-3`, dan `d-grid`.

8. Signin

Kode di atas merupakan template dasar halaman HTML yang digunakan untuk halaman sign-in.

1. Elemen `<head>`:

- Menyertakan meta tag untuk pengaturan karakter dan tampilan responsif.
 - Menyertakan judul halaman yang diambil dari variabel `\$title`.
 - Menyertakan Bootstrap CSS dari CDN untuk mengaplikasikan gaya dan tata letak ke halaman.

2. Ikon Bootstrap dan CSS Tambahan:

- Menyertakan ikon Bootstrap dari CDN (`bootstrap-icons`) untuk digunakan dalam halaman.

- Menyertakan CSS tambahan yang disematkan langsung melalui tag `<link>` untuk memberikan gaya tambahan, seperti yang mungkin didefinisikan dalam file "signin.css".

3. Elemen `<body>`:

- Membuka tag `<body>` dan memberikan class "container" untuk memberikan margin dan padding.
- Memanggil bagian konten halaman sign-in dengan menggunakan `@yield('signin')`.
- Menyertakan Bootstrap JS dari CDN untuk mendukung fungsi interaktif jika diperlukan.

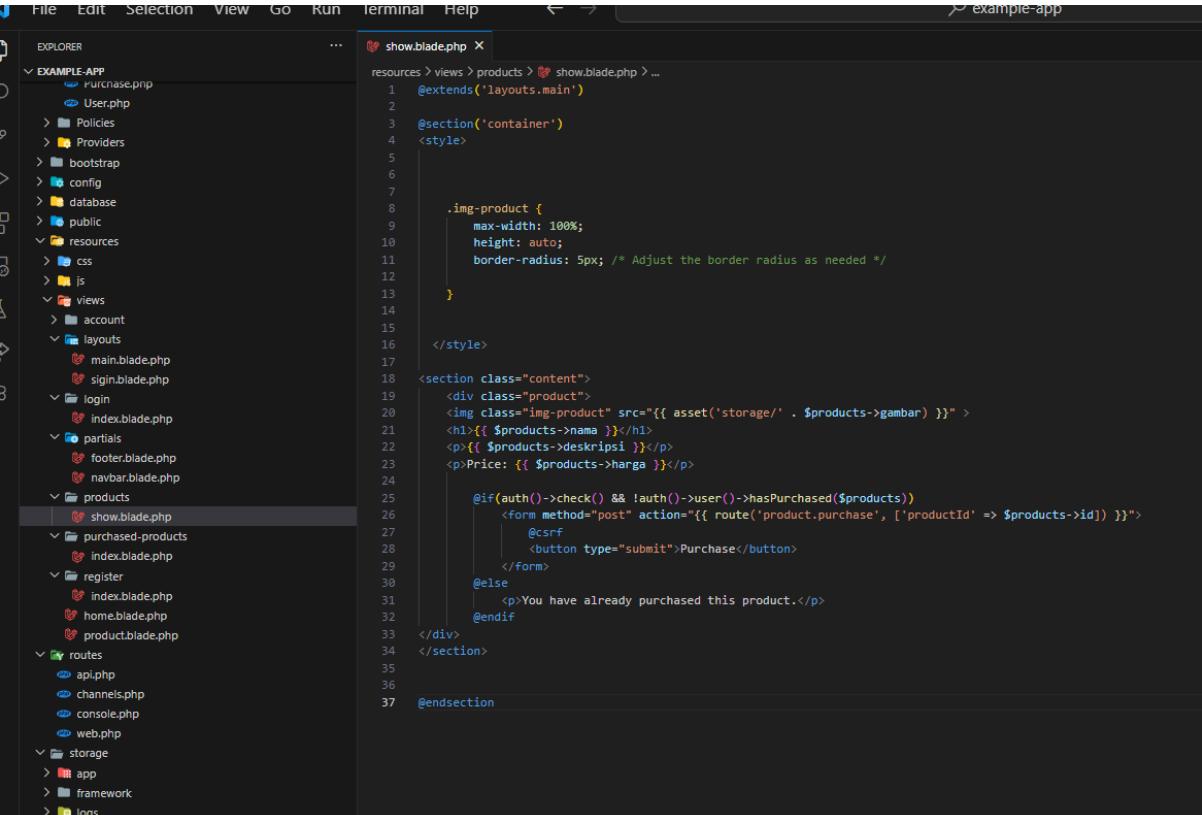
4. Script Bootstrap:

- Menyertakan script Bootstrap JS dari CDN untuk mendukung interaksi dan komponen Bootstrap.

5. Styling Tambahan:

- Menggunakan class "container" pada tag `<body>` untuk memberikan tata letak dan margin sesuai dengan konvensi Bootstrap.
- Menggunakan CSS untuk menentukan gaya halaman sign-in, yang kemungkinan didefinisikan dalam file "signin.css".

9. Product show



```
resources > views > products > show.blade.php > ...
1 @extends('layouts.main')
2
3 @section('container')
4 <style>
5
6
7     .img-product {
8         max-width: 100%;
9         height: auto;
10        border-radius: 5px; /* Adjust the border radius as needed */
11    }
12
13 </style>
14
15 <section class="content">
16     <div class="product">
17         
18         <h2>{{ $products->nama }}</h2>
19         <p>{{ $products->deskripsi }}</p>
20         <p>Price: {{ $products->harga }}</p>
21
22         @if(auth()->check() && !auth()->user()->hasPurchased($products))
23             <form method="post" action="{{ route('product.purchase', ['productId' => $products->id]) }}">
24                 @csrf
25                 <button type="submit">Purchase</button>
26             </form>
27         @else
28             <p>You have already purchased this product.</p>
29         @endif
30     </div>
31
32 </section>
33
34 @endsection
```

Kode di atas adalah bagian dari halaman yang menampilkan detail produk.

1. Directive `@extends` dan `@section`:

- Meng-extend layout dari `layouts.main` yang mungkin memiliki struktur umum untuk seluruh halaman.
- Memiliki sebuah section yang disebut `container` yang akan diisi dengan konten spesifik untuk halaman ini.

2. CSS Styling:

- Membuka tag `

- Jika pengguna sudah membeli produk ini sebelumnya, menampilkan pesan bahwa pengguna telah membeli produk tersebut.

4. Directive `@endsection`:

- Mengakhiri bagian konten halaman dengan directive `@endsection`.

10. History product

```
resources > views > purchased-products > index.blade.php > @contains > @table > @thead > @tr
  4   <tr>
  5     <td>{{ $purchase->product->name }}</td>
  6     <td>{{ $purchase->product->harga }}</td>
  7   </tr>
  8 </tbody>
  9 </table>
10 </div>
11 
12 <@else>
13   <p>No purchased products yet.</p>
14 </@else>
15 </@endif>
16 </div>
17 
18 <@endsection>
```

Kode di atas adalah bagian dari halaman yang menampilkan daftar produk yang telah dibeli oleh pengguna.

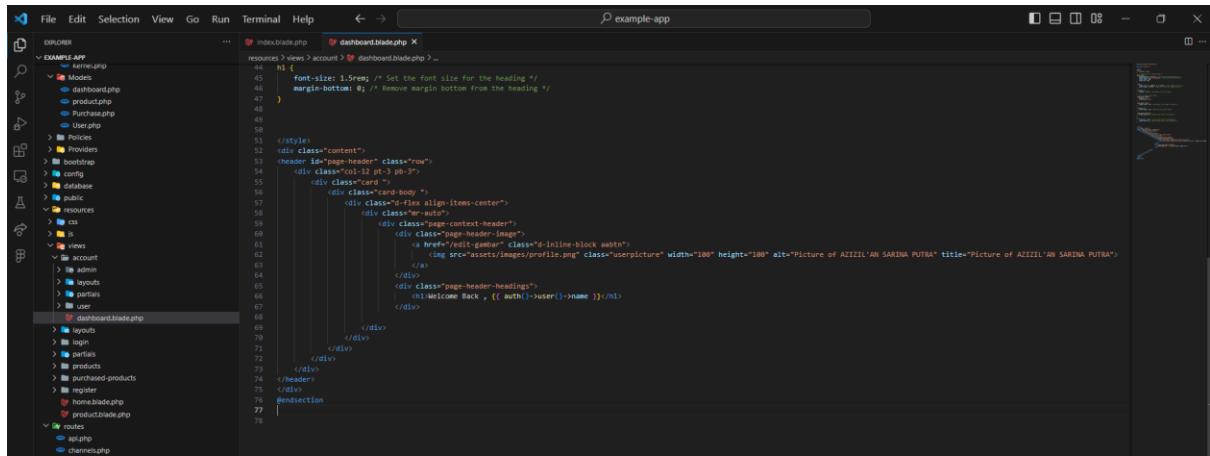
1. Directive `@extends` dan `@section`:

- Meng-extend layout dari `layouts.main` yang mungkin memiliki struktur umum untuk seluruh halaman.
 - Memiliki sebuah section yang disebut `container` yang akan diisi dengan konten spesifik untuk halaman ini.

2. Bagian Konten Halaman:

- Membuka tag `<div>` dengan class "container" untuk mengelompokkan konten spesifik halaman ini.
 - Menampilkan judul "Your Purchased Products".
 - Menampilkan tabel dengan kolom "Product Name" dan "Price".
 - Menggunakan struktur kontrol aliran untuk menampilkan entri tabel jika ada produk yang telah dibeli atau pesan "No purchased products yet." jika tidak ada produk yang dibeli.

11. Dashboard user dan admin



The screenshot shows a code editor with the file `dashboard.blade.php` open. The file contains the following code:

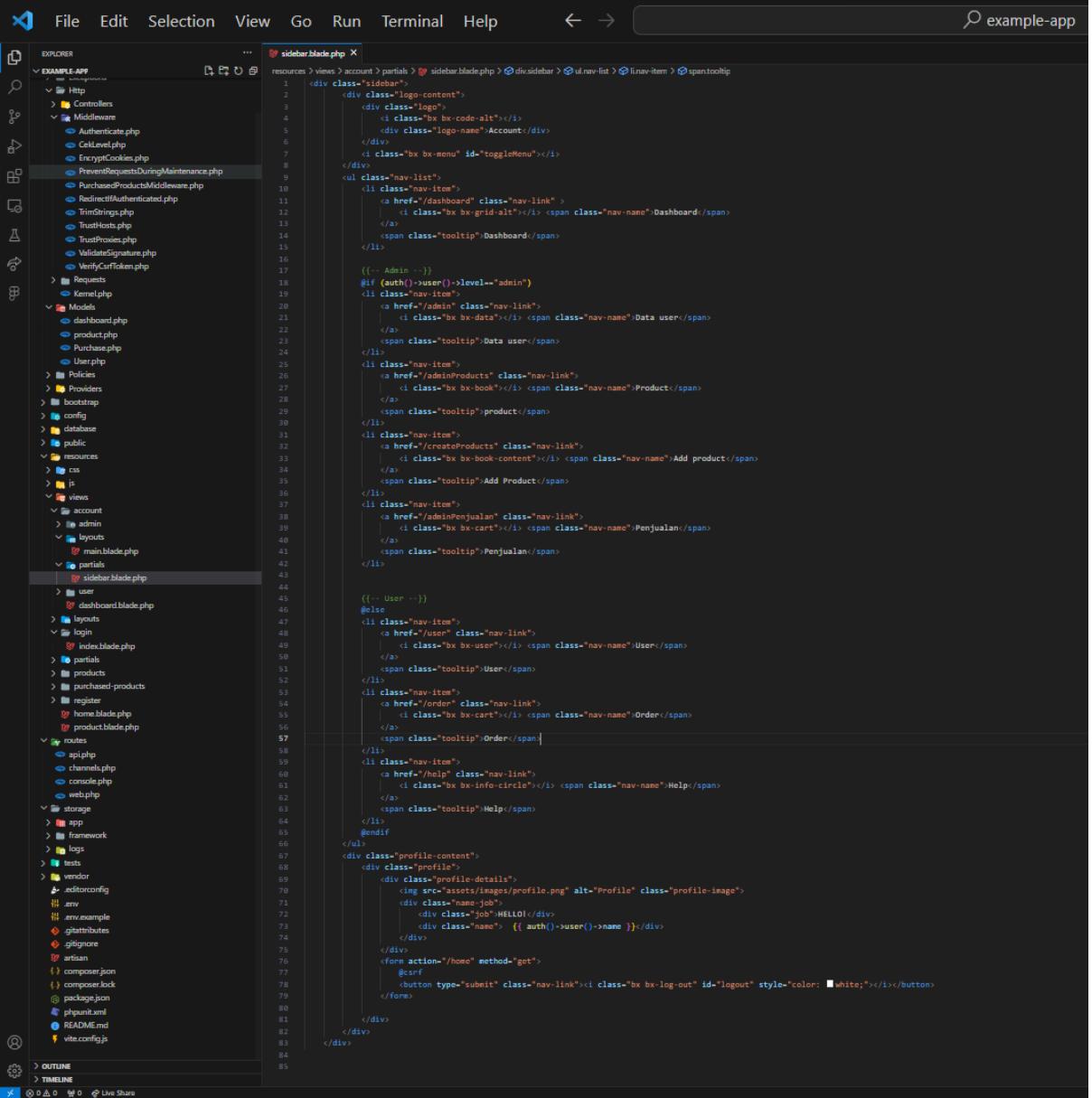
```
resources > views > account > dashboard.blade.php > ...
1<!DOCTYPE html>
2<html>
3    <head>
4        <meta charset="UTF-8">
5        <title>Dashboard</title>
6    </head>
7    <body>
8        <div class="content">
9            <header id="page-header" class="row">
10                <div class="col-12 pt-3 pb-3">
11                    <div class="card">
12                        <div class="card-body">
13                            <div class="d-flex align-items-center">
14                                <div class="mr-2">
15                                    <div class="content">
16                                        <div class="page-header-image">
17                                            <div class="d-inline-block w-100">
18                                                
19                                            <a href="/edit-profile">Edit profile</a>
20                                        </div>
21                                    </div>
22                                </div>
23                                <div class="page-header-headings">
24                                    <h1>Welcome Back, {{ auth()->user()->name }}</h1>
25                                </div>
26                            </div>
27                        </div>
28                    </div>
29                </div>
30            </header>
31            <div>
32                <h2>Dashboard</h2>
33                <p>This is the dashboard for the application. It displays the user's profile picture and name, and provides a link to edit the profile.</p>
34            </div>
35        </div>
36    </body>
37</html>
```

Kode di atas adalah bagian dari halaman akun yang menampilkan header halaman untuk pengguna yang sudah login.

1. Bagian Konten Halaman:

- Menggunakan tag `<div>` dengan class "content" untuk mengelompokkan konten spesifik halaman ini.
- Menampilkan header halaman dengan menggunakan tag `<header>` dan ID "page-header".
- Menggunakan kartu (`<div class="card">`) untuk mengelompokkan elemen-elemen dalam header.
- Menampilkan gambar profil pengguna dengan nama kelas "userpicture" dan memberikan tautan untuk mengedit gambar.
- Menampilkan pesan selamat datang dan nama pengguna.

12. SideBar



```
resources > views > account > partials > sidebar.blade.php > div.sidebar > ul.nav-list > li.nav-item > span.tooltip
1  <div class="sidebar">
2    <div class="logo-content">
3      <i class="bx bx-code-alt"></i>
4      <div class="logo-name">Account</div>
5    </div>
6    <div class="bx-menu" id="toggleMenu"></div>
7  </div>
8  <ul class="nav-list">
9    <li class="nav-item">
10      <a href="/dashboard" class="nav-link" >
11        <i class="bx bx-grid-alt"></i> <span class="nav-name">Dashboard</span>
12      </a>
13      <span class="tooltip">Dashboard</span>
14    </li>
15  </ul>
16  {{-- Admin --}}
17  @if (auth()->user()->level=="admin")
18    <li class="nav-item">
19      <a href="/admin" class="nav-link" >
20        <i class="bx bx-data"></i> <span class="nav-name">Data user</span>
21      </a>
22      <span class="tooltip">Data user</span>
23    </li>
24    <li class="nav-item">
25      <a href="/admin/products" class="nav-link" >
26        <i class="bx bx-book"></i> <span class="nav-name">Product</span>
27      </a>
28      <span class="tooltip">Product</span>
29    </li>
30    <li class="nav-item">
31      <a href="/createProducts" class="nav-link" >
32        <i class="bx bx-book-content"></i> <span class="nav-name">Add product</span>
33      </a>
34      <span class="tooltip">Add product</span>
35    </li>
36    <li class="nav-item">
37      <a href="/admin/penjualan" class="nav-link" >
38        <i class="bx bx-cart"></i> <span class="nav-name">Penjualan</span>
39      </a>
40      <span class="tooltip">Penjualan</span>
41    </li>
42  {{-- User --}}
43  @else
44    <li class="nav-item">
45      <a href="/user" class="nav-link" >
46        <i class="bx bx-user"></i> <span class="nav-name">User</span>
47      </a>
48      <span class="tooltip">User</span>
49    </li>
50    <li class="nav-item">
51      <a href="/order" class="nav-link" >
52        <i class="bx bx-cart"></i> <span class="nav-name">Order</span>
53      </a>
54      <span class="tooltip">Order</span>
55    </li>
56  </ul>
57  <div class="profile-content">
58    <div class="profile">
59      <div class="profile-details">
60        
61        <div class="name-job">
62          <div class="name">HELLO!</div>
63          <div class="name">{{ auth() -> user() -> name }}</div>
64        </div>
65      </div>
66      <form action="/home" method="get" >
67        @csrf
68        <button type="submit" class="nav-link" ><i class="bx bx-log-out" id="logout" style="color: white;"></i></button>
69      </form>
70    </div>
71  </div>
72  </div>
73  </div>
74  </div>
75  </div>
76  </div>
77  </div>
78  </div>
79  </div>
80  </div>
81  </div>
82  </div>
83  </div>
84  </div>
85  </div>
```

Kode di atas adalah bagian dari tata letak sidebar untuk halaman akun.

1. Elemen Sidebar:

- Menggunakan `<div>` dengan class "sidebar" untuk menyusun elemen-elemen dalam sidebar.
- Terdapat elemen dengan class "logo-content" yang menampilkan logo dan ikon menu.
- Elemen dengan class "nav-list" berisi daftar navigasi.

2. Daftar Navigasi:

- Setiap item navigasi (`<li class="nav-item">`) memiliki tautan (`<a>`) yang mengarah ke rute tertentu.

- Menggunakan ikon dari Bootstrap Icons (`<i class="bx ...">`) untuk menunjukkan jenis navigasi.
- Beberapa item navigasi memiliki tooltip (``) untuk memberikan informasi tambahan saat dihover.

3. Kondisional Navigasi:

- Menggunakan kondisi untuk menentukan tampilan navigasi berdasarkan level pengguna (admin atau user).
- Jika pengguna adalah admin, menampilkan navigasi untuk mengelola data pengguna, produk, dan penjualan.
- Jika pengguna adalah user, menampilkan navigasi untuk melihat data pengguna, memesan produk, dan bantuan.

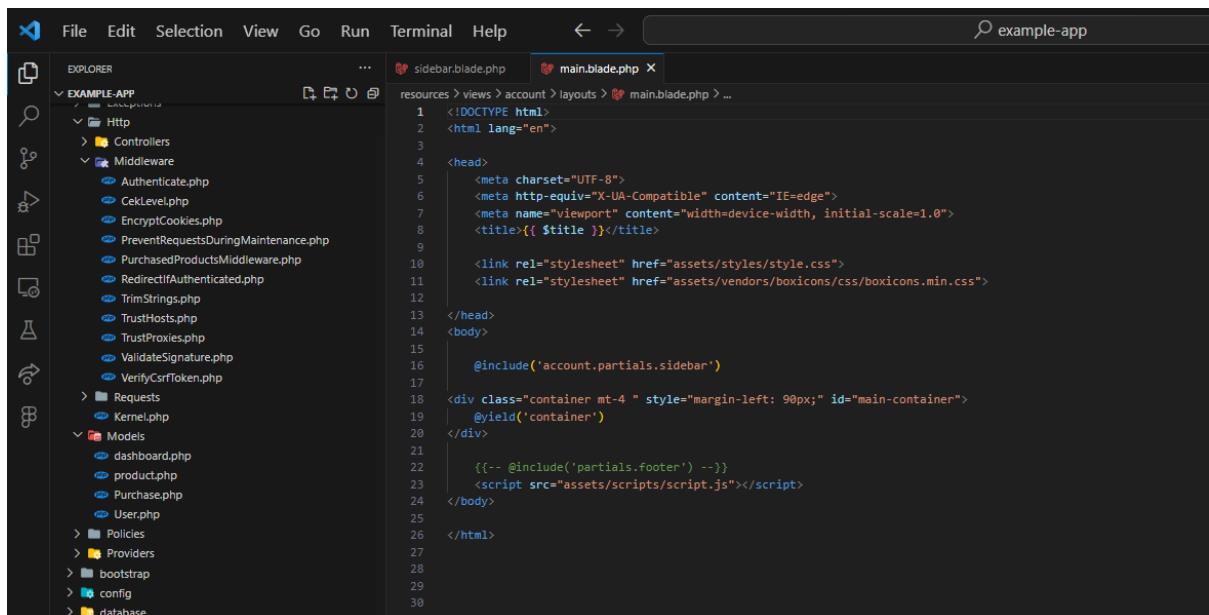
4. Profil Pengguna:

- Menampilkan profil pengguna di bagian bawah sidebar.
- Memperlihatkan gambar profil, nama pengguna, dan tombol untuk keluar (logout).

5. Tombol Logout:

- Menggunakan formulir untuk membuat tombol logout menggunakan metode HTTP GET.
- Tombol logout berisi ikon untuk keluar dari sesi pengguna saat ini.

13. Main pada dashboard account



```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>{{ $title }}</title>
<link rel="stylesheet" href="assets/styles/style.css">
<link rel="stylesheet" href="assets/vendors/boxicons/css/boxicons.min.css">
</head>
<body>
@include('account.partials.sidebar')
<div class="container mt-4 " style="margin-left: 90px;" id="main-container">
@yield('container')
</div>
{{-- @include('partials.footer') --}}
<script src="assets/scripts/script.js"></script>
</body>
</html>

```

1. Kode HTML di atas adalah templat dasar untuk halaman Main pada dashboard account

1. Elemen `<head>`:

- Mendefinisikan karakter set (`<meta charset="UTF-8">`) dan tag `<meta>` lainnya.
- Menentukan versi IE (`<meta http-equiv="X-UA-Compatible" content="IE=edge">`).
- Menentukan viewport (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`) untuk responsifitas di perangkat seluler.
- Memberikan judul halaman dengan variabel judul (`<title>{{ \$title }}</title>`).
- Menambahkan tautan ke file CSS eksternal (`<link rel="stylesheet" href="assets/styles/style.css">`).

2. Elemen `<body>`:

- Menyisipkan sidebar dan konten utama menggunakan `@include`.
- Memuat kontainer utama (`<div class="container mt-4" style="margin-left: 90px;" id="main-container">`) yang berisi konten halaman utama. Style `margin-left` diatur untuk memberikan ruang bagi sidebar.

3. Skrip JavaScript:

- Menambahkan skrip JavaScript eksternal (`<script src="assets/scripts/script.js"></script>`).

4. Link ke Boxicons:

- Menambahkan tautan ke file CSS eksternal Boxicons (`<link rel="stylesheet" href="assets/vendors/boxicons/css/boxicons.min.css">`).

5. Baris Kode Blade Templating:

- Beberapa bagian menggunakan sintaks Blade templating Laravel, seperti `{{ \$title }}`.

14. Admin

```
File Edit Selection View Go Run Terminal Help < > example-app
EXPLORER sidebar.blade.php main.blade.php admin.blade.php
resources > views > account > admin > admin.blade.php > div.content > table.edit > tr
81     .edit-link {
82         margin-left: 0;
83         margin-top: 10px; /* Add some spacing below the text */
84     }
85 
```

```
</style>
86
87 <div class="content">
88     <h1 class="edit">User Data</h1>
89     <table class="edit">
90         <thead>
91             <tr>
92                 <th class="edit">Name</th>
93                 <th class="edit">Email</th>
94                 <th class="edit">Action</th>
95             </tr>
96         </thead>
97         <tbody>
98             @foreach ($users as $user)
99                 <tr>
100                     <td class="edit">{{ $user->name }}
101                     <td class="edit">
102                         {{ $user->email }}
103                     </td>
104                     <td class="edit">
105                         <form onsubmit="return confirm('Tekan untuk menghapus data'); action='/delete/user/{{ $user->id }}' method='POST'>
106                             @csrf
107                             @method('DELETE')
108                             <button type='submit'><i class="bx bx-trash delete-icon"></i></button>
109                         </form>
110                     </td>
111                 </tr>
112             @endforeach
113         </tbody>
114     </table>
115     @endsection
116 
```

Kode Blade di atas (`account.admin.admin.blade.php`) adalah tampilan untuk menampilkan data pengguna (user) dengan beberapa tindakan seperti menghapus data pengguna.

1. Ekstensi Layout:

- Blade mencakup `@extends('account.layouts.main')`, yang mengindikasikan bahwa tampilan ini memperluas tata letak utama yang didefinisikan dalam `account.layouts.main`.

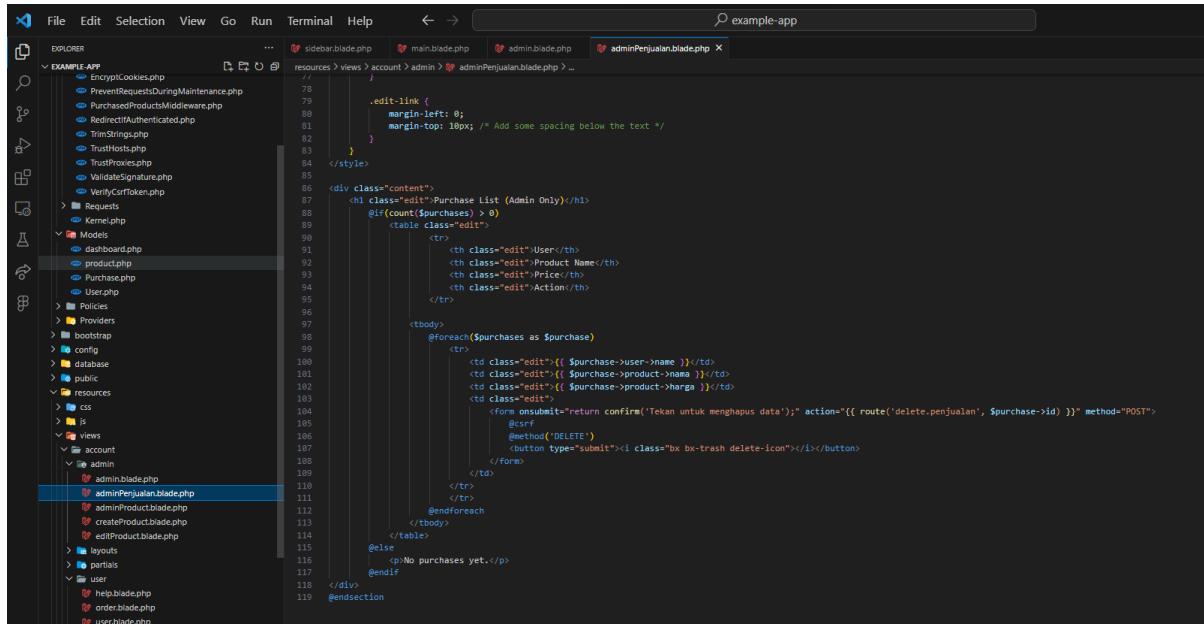
2. Section 'container':

- Dalam tag `

5. CSS Responsif:

- Menambahkan aturan CSS untuk responsifitas, khususnya saat lebar layar kurang dari 600px. Dalam hal ini, tabel akan ditampilkan dalam blok dan sel-sel akan ditata ulang menjadi blok dengan lebar 100%.

15. AdminPenjualan



The screenshot shows a code editor with the file 'adminPenjualan.blade.php' selected in the sidebar. The code is a Blade template for displaying a list of purchases. It includes a header section with a title and a table for listing purchases. The table has columns for User, Product Name, Price, and Action. The 'Action' column contains a form with a delete button. If there are no purchases, it displays a message saying 'No purchases yet.' The code uses Bootstrap classes like 'edit-link' and 'edit' for styling.

```
// sidebar.blade.php main.blade.php admin.blade.php adminPenjualan.blade.php
...
78     .edit-link {
79         margin-left: 0;
80         margin-top: 10px; /* Add some spacing below the text */
81     }
82 }
83 }
84 </style>
85
86 <div class="content">
87     <h1 class="edit">Purchase List (Admin Only)</h1>
88     @if(count($purchases) > 0)
89         <table class="edit">
90             <thead>
91                 <tr>
92                     <th class="edit"><User/></th>
93                     <th class="edit"><Product Name/></th>
94                     <th class="edit"><Price/></th>
95                     <th class="edit"><Action/></th>
96                 </tr>
97             </thead>
98             <tbody>
99                 @foreach($purchases as $purchase)
100                     <tr>
101                         <td class="edit">{{ $purchase->user->name }}</td>
102                         <td class="edit">{{ $purchase->product->nama }}</td>
103                         <td class="edit">{{ $purchase->product->harga }}</td>
104                         <td class="edit">
105                             <form onsubmit="return confirm('Tekan untuk menghapus data');" action="{{ route('delete.penjualan', $purchase->id) }}" method="POST">
106                                 @csrf
107                                 @method('DELETE')
108                                 <button type="submit">i class="bx bx-trash delete-icon"</i></button>
109                             </form>
110                         </td>
111                     </tr>
112                 @endforeach
113             </tbody>
114         </table>
115     @else
116         <p>No purchases yet.</p>
117     @endif
118     </div>
119     @endsection
```

Kode Blade di atas (`purchase-admin.blade.php`) adalah tampilan untuk menampilkan daftar pembelian produk oleh pengguna dengan peran admin.

1. Ekstensi Layout:

- Blade mencakup `@extends('account.layouts.main')`, yang mengindikasikan bahwa tampilan ini memperluas tata letak utama yang didefinisikan dalam `account.layouts.main`.

2. Section 'container':

- Dalam tag `<style>...</style>`, terdapat aturan CSS yang diterapkan untuk mengatur tata letak dan gaya halaman.
- Sebuah `<div class="content">` berisi tabel untuk menampilkan daftar pembelian produk oleh pengguna dengan peran admin.

3. Aturan CSS:

- Aturan CSS mencakup styling untuk elemen-elemen seperti tabel, tombol, dan responsifitas.

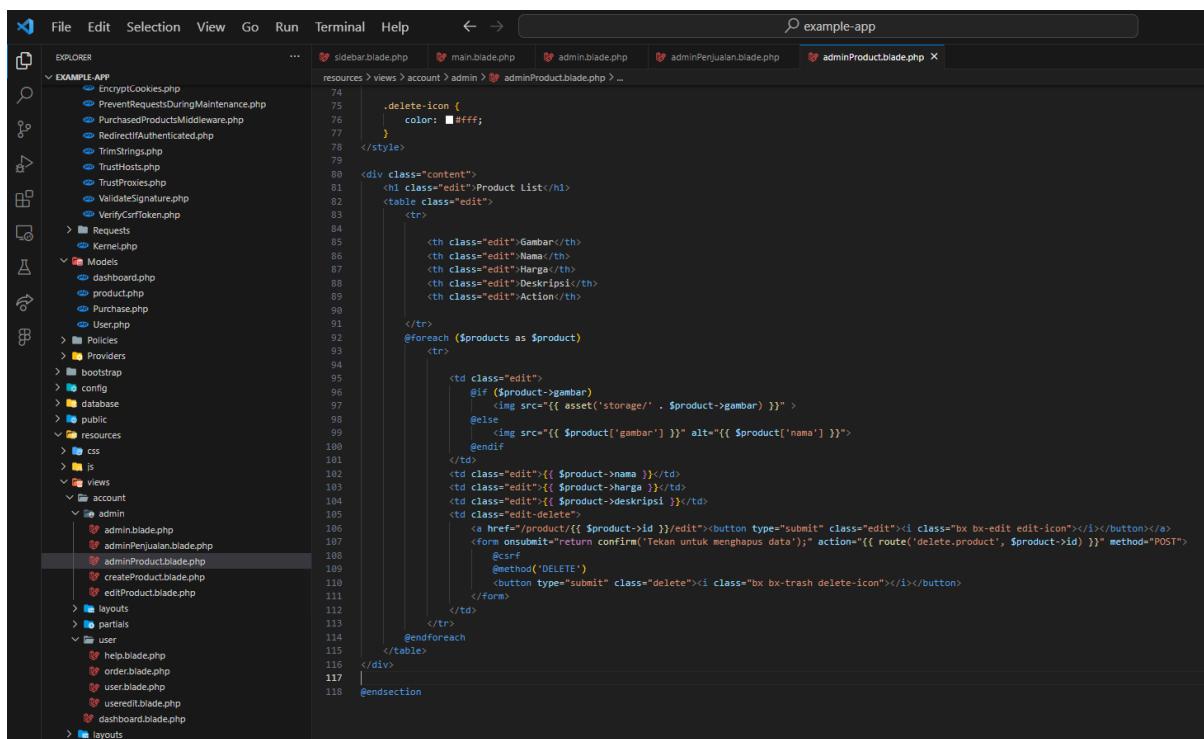
4. Penggunaan Blade Templating:

- Dalam tabel, menggunakan `@foreach` untuk mengulang setiap pembelian dan menampilkan informasi seperti nama pengguna, nama produk, harga, dan tombol hapus.
- Dalam form, menggunakan `@csrf` untuk melindungi formulir dari serangan csrf.
- Tombol hapus dalam formulir dengan `@method('DELETE')` dan konfirmasi penghapusan menggunakan JavaScript (`onsubmit`).

5. CSS Responsif:

- Menambahkan aturan CSS untuk responsifitas, khususnya saat lebar layar kurang dari 600px. Dalam hal ini, tabel akan ditampilkan dalam blok dan sel-sel akan ditata ulang menjadi blok dengan lebar 100%.

16. AdminProduct



The screenshot shows a code editor interface with the file 'adminProduct.blade.php' open. The left sidebar displays a project structure for an 'EXAMPLE_APP' with various PHP files and a 'resources' folder containing Blade templates. The main editor area shows the following Blade template code:

```


<h1>Product List</h1>
    <table class="edit">
        <thead>
            <tr>
                <th>Gambar</th>
                <th>Nama</th>
                <th>Harga</th>
                <th>Deskripsi</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($products as $product)
            <tr>
                <td class="edit">
                    @if ($product->gambar)
                        
                    @endif
                </td>
                <td class="edit">{{ $product->nama }}</td>
                <td class="edit">{{ $product->harga }}</td>
                <td class="edit">{{ $product->deskripsi }}</td>
                <td class="edit-delete">
                    <a href="/product/{{ $product->id }}/edit"><button type="submit" class="edit"><i class="bx bx-edit edit-icon"></i></button></a>
                    @csrf
                    <form onsubmit="return confirm('Tekan untuk menghapus data');" action="{{ route('delete.product', $product->id) }}" method="POST">
                        <button type="submit" class="delete"><i class="bx bx-trash delete-icon"></i></button>
                    </form>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>


```

Kode Blade di atas ('product-list.blade.php') adalah tampilan untuk menampilkan daftar produk.

1. Ekstensi Layout:

- Blade mencakup (`@extends('account.layouts.main')`), yang mengindikasikan bahwa tampilan ini memperluas tata letak utama yang didefinisikan dalam `account.layouts.main`.

2. Section 'container':

- Didalam tag `<style>...</style>`, terdapat aturan CSS yang diterapkan untuk mengatur tata letak dan gaya halaman.
- Sebuah `<div class="content">` berisi tabel untuk menampilkan daftar produk.

3. Aturan CSS:

- Aturan CSS mencakup styling untuk elemen-elemen seperti tabel, tombol, gambar, dan responsifitas.
- Tombol "Edit" dan "Delete" memiliki warna latar belakang yang berbeda saat dihover.

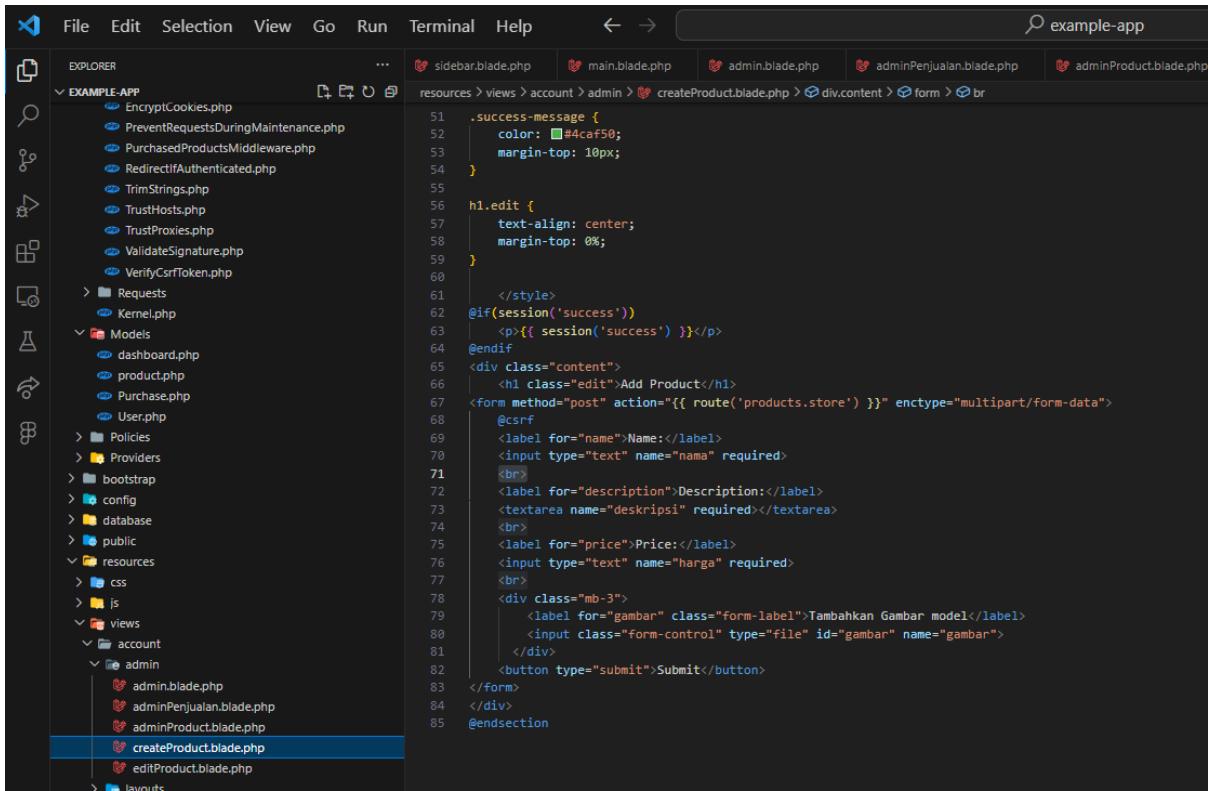
4. Penggunaan Blade Templating:

- Dalam tabel, menggunakan `@foreach` untuk mengulang setiap produk dan menampilkan informasi seperti gambar, nama, harga, deskripsi, dan tombol "Edit" dan "Delete".
- Dalam form, menggunakan `@csrf` untuk melindungi formulir dari serangan csrf.
- Tombol "Delete" dalam formulir dengan `@method('DELETE')` dan konfirmasi penghapusan menggunakan JavaScript (`onsubmit`).

5. CSS Responsif:

- Menambahkan aturan CSS untuk responsifitas, khususnya pada gambar dan tombol, untuk memastikan tampilan yang baik pada layar yang lebih kecil.

17. CreateProduct



The screenshot shows a code editor with the following details:

- File Explorer:** Shows the project structure under "EXAMPLE-APP".
 - Kernel.php
 - Requests
 - Models
 - dashboard.php
 - product.php
 - Purchase.php
 - User.php
 - Policies
 - Providers
 - bootstrap
 - config
 - database
 - public
 - resources
 - css
 - js
 - views
 - account
 - admin
 - admin.blade.php
 - adminPenjualan.blade.php
 - adminProduct.blade.php
 - createProduct.blade.php
 - editProduct.blade.php
 - layouts

- Code Editor:** Displays the content of "createProduct.blade.php".

```
resources > views > account > admin > createProduct.blade.php > div.content > form > br

51 .success-message {
52   color: #4CAF50;
53   margin-top: 10px;
54 }
55
56 h1.edit {
57   text-align: center;
58   margin-top: 0%;
59 }
60
61 </style>
62 @if(session('success'))
63   <p>{{ session('success') }}</p>
64 @endif
65 <div class="content">
66   <h1 class="edit">Add Product</h1>
67   <form method="post" action="{{ route('products.store') }}" enctype="multipart/form-data">
68     @csrf
69     <label for="name">Name:</label>
70     <input type="text" name="nama" required>
71     <br>
72     <label for="description">Description:</label>
73     <textarea name="deskripsi" required></textarea>
74     <br>
75     <label for="price">Price:</label>
76     <input type="text" name="harga" required>
77     <br>
78     <div class="mb-3">
79       <label for="gambar" class="form-label">Tambahkan Gambar model</label>
80       <input class="form-control" type="file" id="gambar" name="gambar">
81     </div>
82     <button type="submit">Submit</button>
83   </form>
84 </div>
85 @endsection
```

Kode Blade di atas (`add-product.blade.php`) adalah tampilan untuk menambahkan produk baru.

1. Ekstensi Layout:

- Blade mencakup `@extends('account.layouts.main')`, yang mengindikasikan bahwa tampilan ini memperluas tata letak utama yang didefinisikan dalam `account.layouts.main`.

2. Section 'container':

- Dalam tag `

- Aturan CSS digunakan untuk mengatur desain formulir, termasuk warna latar belakang, padding, dan bayangan.
 - Tombol "Submit" memiliki warna latar belakang yang berbeda saat dihover.

5. Pesan Sukses:

- Jika ada pesan sukses (`session('success')`), itu akan ditampilkan di bagian atas formulir.

6. CSS Responsif:

- Body memiliki properti `justify-content: center;` untuk menengahkan formulir di tengah halaman.
 - Formulir memiliki lebar maksimum dan terletak di tengah halaman menggunakan `width: 500px;` dan `margin: 0 auto;`.

18. EditProduct

Kode HTML di atas ('edit-product.blade.php') adalah tampilan untuk mengedit data produk.

1. Struktur Dokumen HTML:

- Dokumen HTML dimulai dengan tag `<!DOCTYPE html>` dan elemen `<html>` dengan atribut `lang="en"`.

2. Tag `<head>`:

- Tag `<head>` mengandung elemen-elemen `<meta>` untuk pengaturan karakter dan viewport.
 - Terdapat tag `<style>` yang berisi aturan CSS untuk mendesain tampilan halaman.

3. CSS:

- CSS digunakan untuk merancang tata letak dan gaya formulir.

- Penggunaan `@media` query untuk membuat formulir responsif ketika lebar layar kurang dari 600px.

4. Tag `<body>`:

- Body memiliki beberapa gaya CSS seperti `font-family`, `margin`, `padding`, dan `display: flex` untuk menengahkan kontennya.

5. Formulir Edit Produk:

- Formulir ini menggunakan metode `POST` dan `PUT` untuk mengirim data ke rute yang ditentukan dalam atribut `action`. Rutenya adalah `{{ route('product.update', ['id' => \$product->id]) }}`.
 - Dalam form, menggunakan `@csrf` dan `@method('PUT')` untuk melindungi formulir dari serangan csrf dan menentukan metode HTTP yang benar.
 - Terdapat input untuk Nama (`nama`), Deskripsi (`deskripsi`), Harga (`harga`).

6. CSS Responsif:

- Mengatur lebar input menjadi 100% ketika lebar layar kurang dari 600px.

19. User

Tampilan Blade ini (`user-data.blade.php`) digunakan untuk menampilkan data pengguna dan menyediakan tombol untuk mengedit data pengguna.

1. Struktur Dokumen Blade:

- Blade template dimulai dengan `@extends('account.layouts.main')`, yang mengacu pada layout utama.
- Kemudian, kita menggunakan `@section('container')` untuk menentukan bagian konten dari layout utama yang akan diisi dengan konten halaman ini.
- Kode CSS dan HTML Blade ditempatkan di dalam `@section('container')` dan diakhiri dengan `@endsection` untuk menutup bagian konten.

2. CSS:

- Didefinisikan aturan CSS untuk desain tampilan halaman, termasuk pengaturan font, warna latar belakang, dan beberapa properti gaya.

3. Alerts:

- Ditempatkan di dalam `

` untuk memberikan umpan balik ke pengguna menggunakan Bootstrap alerts (`alert-success` dan `alert-danger`).
- Menggunakan Blade `session()` untuk menampilkan pesan berhasil atau pesan kesalahan jika ada.

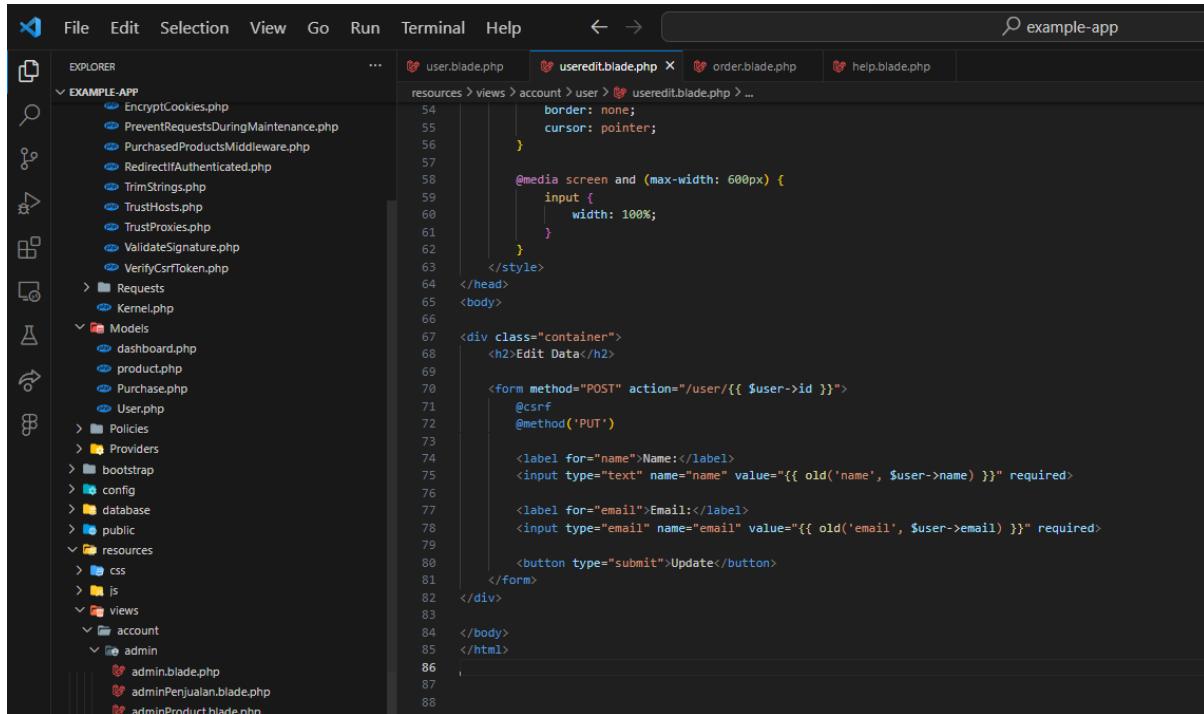
4. Daftar Pengguna:

- Menggunakan tabel untuk menampilkan data pengguna dengan kolom Nama, Email, dan tombol Edit.
- Setiap pengguna memiliki baris dalam tabel, dan tombol Edit diintegrasikan dengan link yang merujuk ke rute edit pengguna.

5. Tombol Edit:

- Menggunakan ikon edit dari Bootstrap Icons (`bx bx-edit`) di dalam tombol untuk memberikan tautan ke halaman pengeditan pengguna (`/user/edit/{id}`).

20. UserEdit



The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** example-app
- Explorer:** Shows the project structure under EXAMPLE-APP, including Models, Requests, Policies, Providers, bootstrap, config, database, public, resources (css, js), and views (account, admin). Specific files like EncryptCookies.php, PreventRequestsDuringMaintenance.php, PurchasedProductsMiddleware.php, RedirectIfAuthenticated.php, TrimStrings.php, TrustHosts.php, TrustProxies.php, ValidateSignature.php, VerifyCsrfToken.php, Kernel.php, dashboard.php, product.php, Purchase.php, User.php, admin.blade.php, adminPenjualan.blade.php, and adminProduct.blade.php are visible.
- Code Editor:** The current file is useredit.blade.php, which contains the following Blade template code:

```
<head>
    <style>
        input {
            border: none;
            cursor: pointer;
        }

        @media screen and (max-width: 600px) {
            input {
                width: 100%;
            }
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Edit Data</h2>

    <form method="POST" action="/user/{{ $user->id }}">
        @csrf
        @method('PUT')

        <label for="name">Name:</label>
        <input type="text" name="name" value="{{ old('name', $user->name) }}" required>

        <label for="email">Email:</label>
        <input type="email" name="email" value="{{ old('email', $user->email) }}" required>

        <button type="submit">Update</button>
    </form>
</div>

</body>
</html>
```

Tampilan Blade ini (`edit-user.blade.php`) digunakan untuk mengedit data pengguna.

1. Struktur Dokumen HTML:

- HTML template dimulai dengan tag `<!DOCTYPE html>` dan elemen `<html>` yang menyertakan bahasa ("en" untuk bahasa Inggris).
- Dalam elemen `<head>`, terdapat informasi meta dan stylesheet (meskipun tidak ada stylesheet eksternal dalam contoh ini).

2. CSS:

- Terdapat aturan CSS yang ditempatkan di dalam tag `<style>`. Aturan tersebut mendefinisikan properti gaya untuk elemen-elemen pada halaman, seperti font, warna latar belakang, dan tata letak.

3. Kontainer:

- Terdapat sebuah kontainer (`<div class="container">`) yang menentukan tampilan form untuk mengedit data pengguna. Kontainer ini memiliki lebar maksimum (`max-width: 800px`), bayangan kotak, dan latar belakang putih dengan border-radius.

4. Formulir Edit:

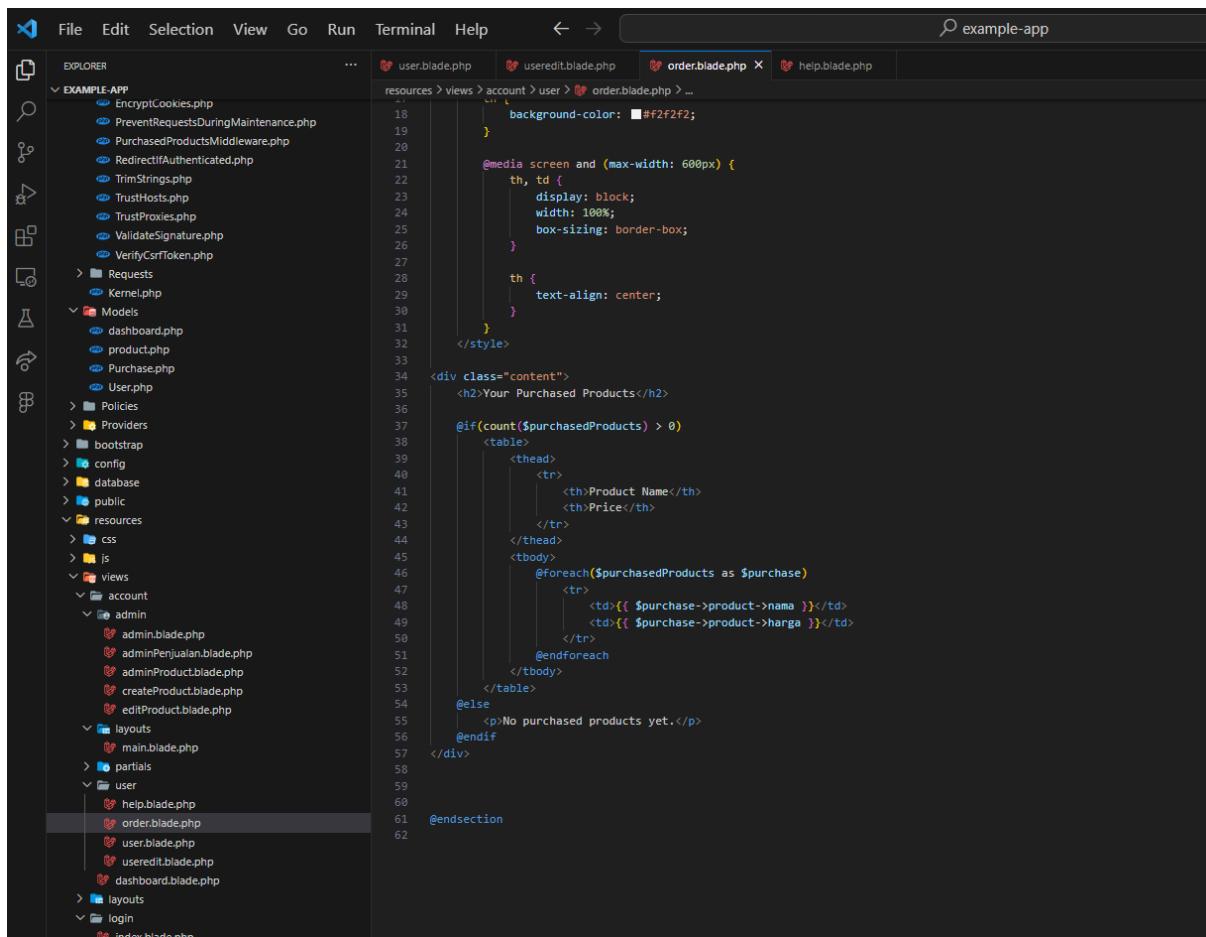
- Formulir menggunakan metode POST dan mengarah ke rute `/user/{id}` dengan metode PUT untuk mengupdate data pengguna.
- Menggunakan Blade directive `@csrf` untuk menyertakan token CSRF.

- Setiap input memiliki label dan nilai default yang berasal dari data pengguna saat ini.
- Terdapat input untuk nama (`name`) dan email (`email`).
- Tombol "Update" digunakan untuk mengirimkan formulir.

5. Responsif:

- Menggunakan aturan media query untuk menyesuaikan lebar input jika layar memiliki lebar maksimum 600px.

21. Order



```

<!-- order.blade.php -->
<div class="content">
    <h2>Your Purchased Products</h2>
    @if(count($purchasedProducts) > 0)
        <table>
            <thead>
                <tr>
                    <th>Product Name</th>
                    <th>Price</th>
                </tr>
            </thead>
            <tbody>
                @foreach($purchasedProducts as $purchase)
                    <tr>
                        <td>{{ $purchase->product->nama }}</td>
                        <td>{{ $purchase->product->harga }}</td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    @else
        <p>No purchased products yet.</p>
    @endif
</div>

```

Kode Blade yang Anda berikan (`purchased-products.blade.php`) adalah tampilan yang menampilkan produk yang telah dibeli oleh pengguna.

1. Tampilan Produk yang Dibeli:

- Tampilan dimulai dengan menentukan struktur HTML menggunakan Blade directive `@extends` untuk menggabungkan layout utama.
- Konten tampilan ditempatkan di dalam blok konten menggunakan `@section('container')`.

2. CSS untuk Tabel:

- Aturan CSS di dalam tag `<style>` digunakan untuk mendesain tampilan tabel.
- Tabel memiliki lebar 100%, batasan kolapsi border, dan margin atas sebesar 20px.
- Gaya `th` dan `td` menetapkan batas dan padding untuk sel-sel tabel.

3. Tabel Produk yang Dibeli:

- Jika ada produk yang telah dibeli (`count(\$purchasedProducts) > 0`), tampilkan tabel dengan informasi produk.
- Tabel memiliki header (`thead`) dengan kolom "Product Name" dan "Price".
- Menggunakan perulangan `@foreach` untuk menampilkan setiap produk yang telah dibeli.
- Informasi produk diambil dari objek `\$purchase->product` yang memiliki properti `nama` dan `harga`.

4. Responsif pada Layar Kecil:

- Menggunakan aturan media query untuk mengubah tata letak tabel jika lebar layar kurang dari atau sama dengan 600px.

5. Kondisi Jika Tidak Ada Produk yang Dibeli:

- Jika tidak ada produk yang dibeli (`else`), tampilkan pesan "No purchased products yet."

22. Help

```

File Edit Selection View Go Run Terminal Help ← → 🔍 example-app
EXPLORER resources > views > account > user > help.blade.php ...
EXAMPLE-APP
  EncryptCookies.php
  PreventRequestsDuringMaintenance.php
  PurchasedProductsMiddleware.php
  RedirectAuthenticated.php
  TrimStrings.php
  TrustHosts.php
  TrustProxies.php
  ValidateSignature.php
  VerifyCsrfToken.php
  Requests
  Kernel.php
  Models
    dashboard.php
    product.php
    Purchase.php
    User.php
  Policies
  Providers
  bootstrap
  config
  database
  events
  ...
  @extends('account.layouts.main')
  @section('content')
    <div class="content">
      <h2>Help Page</h2>
      <p>Welcome to the help page. Here you can find information and assistance regarding our application.</p>
      <ul>
        <li>For general inquiries, please contact our support team.</li>
        <li>If you have specific questions about features, check out our documentation.</li>
        <li>Need further assistance? Feel free to reach out!</li>
      </ul>
    </div>
  @endsection

```

1. `@extends('account.layouts.main')`: Mendeklarasikan bahwa halaman ini akan menggunakan layout yang telah ditentukan sebelumnya, yakni 'account.layouts.main'. Layout ini mungkin sudah mencakup elemen-elemen umum seperti header, footer, dan struktur tata letak halaman.

2. `@section('container')`: Memulai bagian konten khusus halaman, yang akan ditempatkan dalam "container" di layout utama.
3. `<div class="content">`: Mendefinisikan div dengan kelas "content" yang biasanya digunakan untuk mengatur tampilan dan styling konten halaman.
4. `<h2>Halaman Bantuan</h2>`: Menampilkan judul halaman bantuan.
5. `<p>...</p>`: Menyediakan paragraf yang memberikan sambutan dan tujuan dari halaman bantuan.
6. `...`: Membuat daftar tak terurut (unordered list) untuk menyajikan informasi berguna. Setiap elemen `` dalam daftar memberikan poin-poin penting tentang cara menghubungi dukungan dan mengecek dokumentasi.
7. `@endsection`: Menutup bagian konten halaman.

8) Kelebihan

Kelebihan sistem ini menurut saya adalah pada sistem ini dapat melakukan login dengan page yang sama tetapi jika memasukan data user maka hanya menampilkan halaman yang dapat diakses oleh user sedangkan jika memasukan data admin maka akan menampilkan data yang hanya diakses oleh admin. Kelebihan lain juga sistem ini sangat simple dan mudah dipahami oleh user maupun admin