Dokumen R&D Lengkap - Sistem Inventori Toko Kelontong

# 1. Deskripsi Proyek

Sistem ini digunakan oleh toko kelontong untuk mencatat inventaris barang secara digital, menggantikan pencatatan manual. Sistem mencakup fitur pengelolaan data barang, kategori barang, stok masuk dan keluar, serta laporan barang.  
  
Framework: Laravel 12  
Database: MySQLi (MySQL)  
Frontend: Blade Template (Bootstrap)  
Metodologi: Agile (Scrum) - sprint mingguan

# 2. Struktur dan Relasi Database

Diagram Entitas Relasi (ERD):  
- users (1) --- (n) stock\_in / stock\_out  
- items (1) --- (n) stock\_in / stock\_out  
- categories (1) --- (n) items  
  
Tabel:  
- users(id, name, email, password, created\_at, updated\_at)  
- categories(id, name, created\_at, updated\_at)  
- items(id, name, category\_id, stock, unit, price, created\_at, updated\_at)  
- stock\_in(id, item\_id, quantity, date, user\_id)  
- stock\_out(id, item\_id, quantity, date, user\_id)

# 3. Dokumentasi API

Autentikasi:  
POST /api/login  
  
Barang:  
GET /api/items  
POST /api/items  
PUT /api/items/{id}  
DELETE /api/items/{id}  
  
Kategori:  
GET /api/categories  
POST /api/categories  
  
Stok Masuk:  
GET /api/stock-in  
POST /api/stock-in  
  
Stok Keluar:  
GET /api/stock-out  
POST /api/stock-out  
  
Laporan:  
GET /api/reports/stock

# 4. Fungsi Menu dan Modul

- Dashboard: Statistik barang dan grafik stok.  
- Data Barang: CRUD barang toko.  
- Kategori: Pengelolaan kategori.  
- Stok Masuk: Pencatatan barang masuk.  
- Stok Keluar: Pencatatan barang keluar.  
- Laporan: Laporan rekap bulanan/tahunan.  
- Pengguna: Manajemen akun admin.

# 5. Rancangan Tampilan Antarmuka

Sidebar Navigasi:  
- Dashboard  
- Barang  
- Kategori  
- Stok Masuk / Keluar  
- Laporan  
  
Halaman Barang:  
- Tabel daftar barang  
- Tombol Tambah/Edit/Hapus  
- Form input barang  
  
Halaman Laporan:  
- Filter berdasarkan tanggal/kategori  
- Export ke PDF/Excel

# 6. Konvensi Penamaan Variabel

- Nama variabel: camelCase (example: itemName, totalStock)  
- Nama API endpoint: kebab-case (example: /api/stock-in)  
- Nama model: PascalCase (User, Item, StockIn)  
- Penamaan tabel: snake\_case (stock\_out, stock\_in)

# 7. Rencana Pengembangan Agile (Sprint)

Sprint 1:  
- Setup project Laravel  
- Desain database  
- Fitur login dan autentikasi  
  
Sprint 2:  
- CRUD kategori & barang  
- Tampilan dashboard awal  
  
Sprint 3:  
- Fitur stok masuk/keluar  
- Validasi & notifikasi  
  
Sprint 4:  
- Fitur laporan dan export  
- Pengujian & deployment

# 8. Strategi Pengujian

- Unit testing: Controller, Model  
- API testing: Postman  
- UI testing: Manual klik dan uji input form  
- Blackbox testing: Simulasi penggunaan sistem oleh user  
  
Pengujian dilakukan pada akhir setiap sprint.

1. Sprint1
2. Instalasi: composer create-project laravel/laravel tokoKu

Composer create-project laravel/laravel tokoKu “10.\*”

1. Buat database:

php artisan make:migration create\_users\_table:

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

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

1. Buat Login
   1. Modifikasiapp/Model/User.php:

<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

use Notifiable;

protected $fillable = [

'name', 'email', 'password',

];

protected $hidden = [

'password', 'remember\_token',

];

// Relasi dengan stock\_in dan stock\_out

public function stockIns()

{

return $this->hasMany(StockIn::class);

}

public function stockOuts()

{

return $this->hasMany(StockOut::class);

}

}

b. Buat controller : php artisan make:controller Auth/AuthController

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class AuthController extends Controller

{

public function showLoginForm()

{

return view('auth.login');

}

public function login(Request $request)

{

$credentials = $request->validate([

'email' => 'required|email',

'password' => 'required',

]);

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

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

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

}

return back()->withErrors([

'email' => 'Email atau password salah.',

]);

}

public function logout(Request $request)

{

Auth::logout();

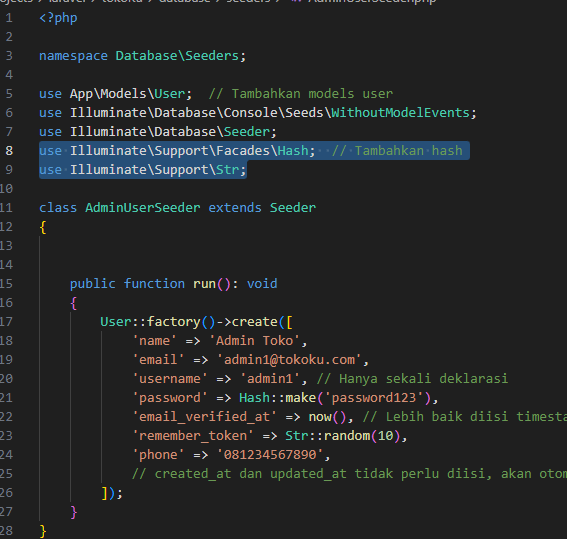
$request->session()->invalidate();

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

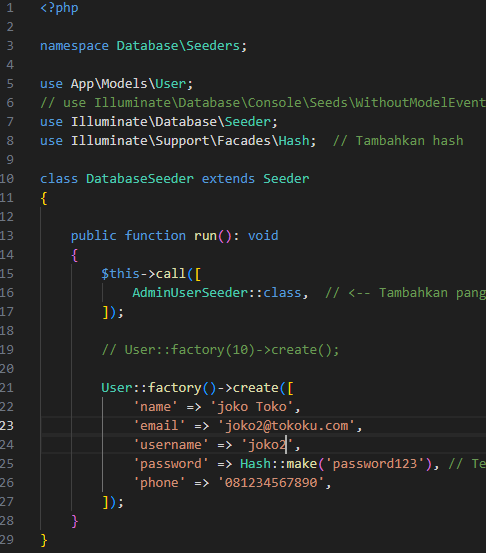
return redirect('/');

}

}

Admin seeder databse: 

Defaul seeder:



Route:



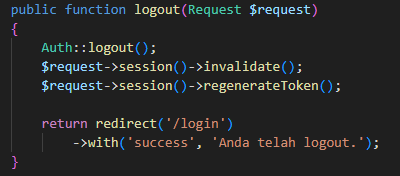
Controller login:



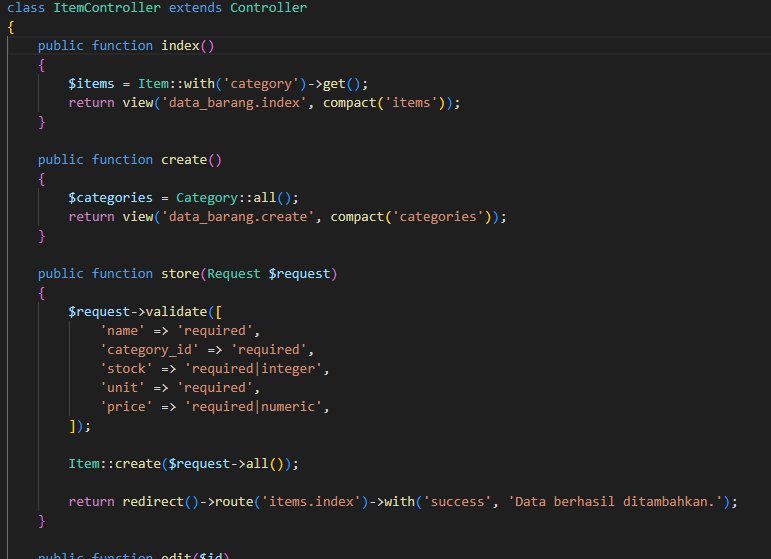
View login:

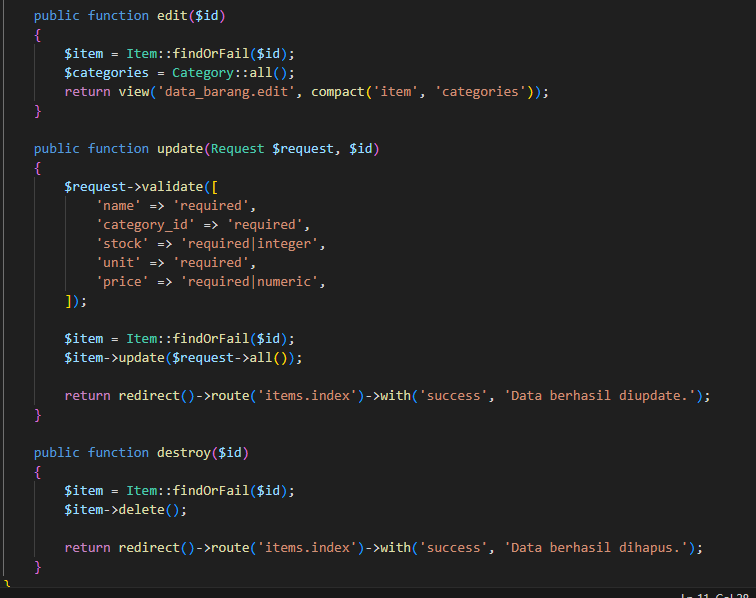


Logout:



Controller item:





View item:

@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header d-flex justify-content-between align-items-center">

        <h5>Data Barang</h5>

        <a href="{{ route('items.create') }}" class="btn btn-primary btn-sm">+ Tambah Data</a>

    </div>

    <div class="card-body">

        <div class="table-responsive">

            <table id="myTable" class="table table-striped">

                <thead>

                    <tr>

                        <th>Nama Barang</th>

                        <th>Kategori</th>

                        <th>Stok</th>

                        <th>Satuan</th>

                        <th>Harga</th>

                        <th>Aksi</th> <!-- Tambahkan kolom Aksi -->

                    </tr>

                </thead>

                <tbody>

                    @foreach($items as $item)

                    <tr>

                        <td>{{ $item->name }}</td>

                        <td>{{ $item->category->name }}</td>

                        <td>{{ $item->stock }}</td>

                        <td>{{ $item->unit }}</td>

                        <td>Rp {{ number\_format($item->price, 2, ',', '.') }}</td>

                        <td>

                            <a href="{{ route('items.edit', $item->id) }}" class="btn btn-warning btn-sm">Edit</a>

                            <form action="{{ route('items.destroy', $item->id) }}" method="POST" style="display:inline-block;"  class="delete-form">

                                @csrf

                                @method('DELETE')

                                <button class="btn btn-danger btn-sm" type="submit">Hapus</button>

                            </form>

                        </td>

                    </tr>

                    @endforeach

                </tbody>

            </table>

        </div>

    </div>

</div>

@endsection

@push('scripts')

<script>

$(document).ready(function() {

    $('#myTable').DataTable({

        responsive: true

    });

    // Auto hide alert success

    setTimeout(function() {

        $("#success-alert").fadeOut('slow');

    }, 1500);

    // SweetAlert untuk delete

    $('.delete-form').on('submit', function(e) {

        e.preventDefault();

        const form = this;

        Swal.fire({

            title: 'Yakin mau hapus?',

            text: "Data yang dihapus tidak bisa dikembalikan!",

            icon: 'warning',

            showCancelButton: true,

            confirmButtonColor: '#3085d6',

            cancelButtonColor: '#d33',

            confirmButtonText: 'Ya, hapus!',

            cancelButtonText: 'Batal'

        }).then((result) => {

            if (result.isConfirmed) {

                form.submit();

            }

        });

    });

    // SweetAlert setelah berhasil tambah/edit/delete

    @if (session('success'))

        Swal.fire({

            icon: 'success',

            title: '{{ session('success') }}',

            showConfirmButton: false,

            timer: 1500

        });

    @endif

});

</script>

@endpush

Create itm:@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header">

        Tambah Data Barang

    </div>

    <div class="card-body">

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

            @csrf

            <div class="mb-3">

                <label>Nama Barang</label>

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

            </div>

            <div class="mb-3">

                <label>Kategori</label>

                <select name="category\_id" class="form-control" required>

                    <option value="">-- Pilih Kategori --</option>

                    @foreach($categories as $category)

                    <option value="{{ $category->id }}">{{ $category->name }}</option>

                    @endforeach

                </select>

            </div>

            <div class="mb-3">

                <label>Stok</label>

                <input type="number" name="stock" class="form-control" required>

            </div>

            <div class="mb-3">

                <label>Satuan</label>

                <input type="text" name="unit" class="form-control" required>

            </div>

            <div class="mb-3">

                <label>Harga</label>

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

            </div>

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

            <a href="{{ route('items.index') }}" class="btn btn-secondary">Batal</a>

        </form>

    </div>

</div>

@endsection

Edit item:@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header">

        Edit Data Barang

    </div>

    <div class="card-body">

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

            @csrf

            @method('PUT')

            <div class="mb-3">

                <label>Nama Barang</label>

                <input type="text" name="name" class="form-control" value="{{ $item->name }}" required>

            </div>

            <div class="mb-3">

                <label>Kategori</label>

                <select name="category\_id" class="form-control" required>

                    @foreach($categories as $category)

                    <option value="{{ $category->id }}" {{ $item->category\_id == $category->id ? 'selected' : '' }}>

                        {{ $category->name }}

                    </option>

                    @endforeach

                </select>

            </div>

            <div class="mb-3">

                <label>Stok</label>

                <input type="number" name="stock" class="form-control" value="{{ $item->stock }}" required>

            </div>

            <div class="mb-3">

                <label>Satuan</label>

                <input type="text" name="unit" class="form-control" value="{{ $item->unit }}" required>

            </div>

            <div class="mb-3">

                <label>Harga</label>

                <input type="number" name="price" class="form-control" value="{{ $item->price }}" required>

            </div>

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

            <a href="{{ route('items.index') }}" class="btn btn-secondary">Batal</a>

        </form>

    </div>

</div>

@endsection

End

Dokumentasi Sistem Inventory Toko Kelontong - Laravel

# 1. Migration Files

Migration digunakan untuk mendefinisikan struktur tabel di database.   
Beberapa file migration yang penting di dalam proyek ini meliputi:  
- create\_users\_table  
- create\_categories\_table  
- create\_items\_table  
- create\_stock\_ins\_table  
- create\_stock\_outs\_table  
  
Setiap file migration memiliki fungsi up() untuk membuat tabel dan fungsi down() untuk rollback (menghapus tabel).

# 2. Seeder Files

Seeder digunakan untuk mengisi data awal pada database.  
Contoh seeder dalam sistem ini antara lain:  
- AdminUserSeeder: membuat akun admin default.  
- ItemSeeder: mengisi barang-barang contoh.  
- StockInSeeder: mengisi contoh data stok masuk.  
  
Seeder dijalankan dengan perintah php artisan db:seed.

# 3. Model Files

Model dalam Laravel menggunakan Eloquent ORM untuk mewakili tabel.  
Contoh model:  
- Category: memiliki relasi one-to-many ke Item.  
- Item: relasi ke Category, StockIn, StockOut.  
- StockIn & StockOut: relasi belongsTo ke Item.  
  
Model ini digunakan di controller untuk query dan manipulasi data.

# 4. Controller Files

Controller menerima request dari user dan mengatur logika bisnis, lalu melemparkan data ke view.  
Contoh controller:  
- DashboardController: Menampilkan data ringkasan seperti total barang dan transaksi.  
- ItemController: Mengatur CRUD barang.  
- StockInController: Menangani transaksi masuk barang.  
- StockOutController: Menangani transaksi keluar barang.

# 5. View Files (Blade Template)

View adalah file Blade (.blade.php) yang menjadi antarmuka pengguna.  
Contoh folder view:  
- dashboard/index.blade.php: Menampilkan halaman dashboard.  
- barang/index.blade.php: Tabel data barang.  
- stokmasuk/index.blade.php: Tabel data stok masuk.  
- stokkeluar/index.blade.php: Tabel data stok keluar.  
  
Semua view menggunakan template utama layout app.blade.php.

# 6. Routing (web.php)

Routing menghubungkan URL ke controller yang sesuai.  
Contoh route:  
- Route::get('/dashboard', [DashboardController::class, 'index']);  
- Route::resource('barang', ItemController::class);  
- Route::resource('stokmasuk', StockInController::class);  
- Route::resource('stokkeluar', StockOutController::class);  
  
Route::resource otomatis membuat route untuk index, create, store, edit, update, destroy.

# 7. Penjelasan Alur Sistem (Flowchart)

1. Pengguna login → diarahkan ke DashboardController@index  
2. Dashboard → Menampilkan ringkasan data (jumlah barang, stok masuk, stok keluar)  
3. Menu Barang → CRUD oleh ItemController  
4. Menu Stok Masuk → form input → disimpan oleh StockInController@store  
5. Menu Stok Keluar → form input → disimpan oleh StockOutController@store  
6. Data ditampilkan via Blade View  
7. Semua proses routing dikontrol di web.php

Dokumentasi Migration: create\_users\_table.php

# 1. Deskripsi Umum

Migration ini berfungsi untuk membuat tabel users pada database Laravel. Struktur tabel ini mencakup informasi dasar pengguna seperti nama, email, username, password, dan lainnya.

# 2. Penjelasan Setiap Bagian Kode

Bagian Import:

<?php  
  
use Illuminate\Database\Migrations\Migration;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Support\Facades\Schema;

* Mengimpor class-class penting:
* Migration: class dasar untuk membuat migration
* Blueprint: digunakan untuk mendefinisikan struktur tabel
* Schema: facade untuk manipulasi database

Class Anonymous:

return new class extends Migration {

## 2.1 Fungsi up()

public function up(): void {  
 Schema::create('users', function (Blueprint $table) {  
 $table->id();  
 $table->string('name');  
 $table->string('username')->nullable()->unique();  
 $table->string('email')->unique();  
 $table->timestamp('email\_verified\_at')->nullable();  
 $table->string('password');  
 $table->string('phone')->nullable();

$table->string('role')->default('admin');  
 $table->rememberToken();  
 $table->timestamps();  
 });  
}

### Penjelasan:

* id: kolom primary key (auto increment, tipe bigint)
* name: kolom nama pengguna
* username: opsional, harus unik jika diisi
* email: wajib dan unik
* email\_verified\_at: waktu verifikasi email, nullable
* password: menyimpan kata sandi (hashed)
* phone: opsional
* 'role:admin secara default
* rememberToken: untuk fitur 'ingat saya'
* timestamps: Laravel menambahkan kolom created\_at dan updated\_at

## 2.2 Fungsi down()

public function down(): void {  
 Schema::dropIfExists('users');  
}

Fungsi ini menghapus tabel users saat dilakukan rollback migration.

# 3. Ringkasan Kolom Tabel users

|  |  |  |
| --- | --- | --- |
| Kolom | Tipe | Keterangan |
| id | bigint | primary key |
| name | string | wajib |
| username | string | unik, boleh kosong |
| email | string | wajib, unik |
| email\_verified\_at | timestamp | nullable |
| password | string | wajib |
| phone | string | nullable |
| role | string | Admin secara default |
| remember\_token | string | nullable, digunakan Laravel |
| created\_at | timestamp | otomatis oleh Laravel |
| updated\_at | timestamp | otomatis oleh Laravel |

# 4. Perintah Terkait

* Menjalankan migration:
* php artisan migrate
* Membatalkan migration:
* php artisan migrate:rollback
* Category:

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

            $table->id();

            $table->string('name');

            $table->timestamps();

        });

* item:

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

            $table->id();

            $table->string('nama');

            $table->foreignId('category\_id')->constrained();

            $table->decimal('harga', 12, 2);      // Harga per satuan\_harga

            $table->string('satuan\_harga');       // Satuan untuk harga (contoh: biji, liter)

            $table->integer('stok');              // Jumlah stok

            $table->string('satuan\_stok');        // Satuan untuk stok fisik (contoh: lusin, dus)

            $table->text('keterangan')->nullable();

            $table->timestamps();

        });

* Stockin:

Schema::create('stock\_in', function (Blueprint $table) {

            $table->id();

            $table->foreignId('item\_id')->constrained();

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

            $table->decimal('harga\_per\_satuan', 12, 2); // Harga beli per satuan\_jumlah

            $table->integer('jumlah');                   // Jumlah yang dibeli

            $table->string('satuan\_jumlah');            // Satuan transaksi (contoh: lusin, dus)

            $table->text('keterangan')->nullable();     // Keterangan tambahan

            $table->date('tanggal');

            $table->timestamps();

        });

* Stockout:

Schema::create('stock\_out', function (Blueprint $table) {

            $table->id();

            $table->foreignId('item\_id')->constrained();

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

            $table->decimal('harga\_jual\_per\_satuan', 12, 2); // Harga jual per satuan\_harga

            $table->integer('jumlah');                        // Jumlah yang dijual

            $table->string('satuan\_jumlah');                 // Satuan transaksi (contoh: biji, batang)

            $table->text('keterangan')->nullable();          // Keterangan penjualan

            $table->date('tanggal');

            $table->timestamps();

        });

* Session:

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

            $table->string('id')->primary();

            $table->foreignId('user\_id')->nullable()->index();

            $table->string('ip\_address', 45)->nullable();

            $table->text('user\_agent')->nullable();

            $table->longText('payload');

            $table->integer('last\_activity')->index();

        });

* Password reset:

 Schema::create('password\_reset\_tokens', function (Blueprint $table) {

            $table->string('email')->index();

            $table->string('token');

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

        });

* Seeder:
* php artisan db:seed

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

use Carbon\Carbon;

use Illuminate\Support\Str;

class DatabaseSeeder extends Seeder

{

    public function run()

    {

        // Seed users first (no dependencies)

        DB::table('users')->insert([

            [

                'name' => 'Admin Toko',

                'username' => 'admin1',

                'email' => 'admin1@tokoku.com',

                'password' => Hash::make('password123'),

                'phone' => '081234567890',

                'role'  => 'admin',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ],

            [

                'name' => 'Kasir Toko',

                'username' => 'kasir1',

                'email' => 'kasir1@tokoku.com',

                'password' => Hash::make('password123'),

                'phone' => '082345678901',

                'role'  => 'admin',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ]

        ]);

        // Seed categories (no dependencies)

        $categories = [

            ['name' => 'Sembako', 'created\_at' => Carbon::now(), 'updated\_at' => Carbon::now()],

            ['name' => 'Minuman', 'created\_at' => Carbon::now(), 'updated\_at' => Carbon::now()],

            ['name' => 'Snack', 'created\_at' => Carbon::now(), 'updated\_at' => Carbon::now()],

            ['name' => 'Perlengkapan Mandi', 'created\_at' => Carbon::now(), 'updated\_at' => Carbon::now()],

            ['name' => 'Rokok', 'created\_at' => Carbon::now(), 'updated\_at' => Carbon::now()]

        ];

        DB::table('categories')->insert($categories);

        // Seed items (depends on categories)

        $items = [

            [

                'nama' => 'Beras Premium',

                'category\_id' => 1,

                'harga' => 12500,

                'satuan\_harga' => 'kg',

                'stok' => 100,

                'satuan\_stok' => 'kg',

                'keterangan' => 'Beras kualitas premium',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ],

            [

                'nama' => 'Minyak Goreng 1L',

                'category\_id' => 1,

                'harga' => 18000,

                'satuan\_harga' => 'botol',

                'stok' => 50,

                'satuan\_stok' => 'botol',

                'keterangan' => 'Minyak goreng kemasan botol',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ],

            [

                'nama' => 'Aqua Botol 600ml',

                'category\_id' => 2,

                'harga' => 5000,

                'satuan\_harga' => 'botol',

                'stok' => 200,

                'satuan\_stok' => 'botol',

                'keterangan' => 'Air mineral kemasan botol',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ],

            [

                'nama' => 'Indomie Goreng',

                'category\_id' => 3,

                'harga' => 3500,

                'satuan\_harga' => 'bungkus',

                'stok' => 150,

                'satuan\_stok' => 'bungkus',

                'keterangan' => 'Mi instan rasa goreng',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ],

            [

                'nama' => 'Lifebuoy Sabun',

                'category\_id' => 4,

                'harga' => 4500,

                'satuan\_harga' => 'batang',

                'stok' => 80,

                'satuan\_stok' => 'batang',

                'keterangan' => 'Sabun mandi Lifebuoy',

                'created\_at' => Carbon::now(),

                'updated\_at' => Carbon::now()

            ]

        ];

        DB::table('items')->insert($items);

// Seed stock\_in dan stock\_out

$items = DB::table('items')->pluck('id')->toArray();

$users = DB::table('users')->pluck('id')->toArray();

$satuan = ['kg', 'botol', 'bungkus', 'batang', 'lusin', 'dus'];

for ($i = 0; $i < 50; $i++) {

    DB::table('stock\_in')->insert([

        'item\_id' => $items[array\_rand($items)],

        'user\_id' => $users[array\_rand($users)],

        'harga\_per\_satuan' => rand(1000, 30000),

        'jumlah' => rand(1, 100),

        'satuan\_jumlah' => $satuan[array\_rand($satuan)],

        'keterangan' => 'Stok masuk otomatis ke-' . ($i+1),

        'tanggal' => Carbon::now()->subDays(rand(0, 90)),

        'created\_at' => Carbon::now(),

        'updated\_at' => Carbon::now(),

    ]);

    DB::table('stock\_out')->insert([

        'item\_id' => $items[array\_rand($items)],

        'user\_id' => $users[array\_rand($users)],

        'harga\_jual\_per\_satuan' => rand(2000, 50000),

        'jumlah' => rand(1, 80),

        'satuan\_jumlah' => $satuan[array\_rand($satuan)],

        'keterangan' => 'Stok keluar otomatis ke-' . ($i+1),

        'tanggal' => Carbon::now()->subDays(rand(0, 90)),

        'created\_at' => Carbon::now(),

        'updated\_at' => Carbon::now(),

    ]);

}

    }

}

Berikut adalah penjelasan tentang arti dan fungsi dari skrip PHP yang Anda berikan:

### 1. ****Seeding Pengguna (Users)****

Bagian pertama dari kode ini memasukkan dua pengguna ke dalam tabel users:

**Admin Toko** dengan peran admin

**Kasir Toko** dengan peran admin

Password pengguna di-hash menggunakan Hash::make(), dan waktu saat data dimasukkan (created\_at dan updated\_at) menggunakan Carbon::now() untuk mendapatkan waktu saat ini.

### 2. ****Seeding Kategori (Categories)****

Pada bagian ini, lima kategori produk dimasukkan ke dalam tabel categories:

**Sembako** (Bahan Pokok)

**Minuman** (Drinks)

**Snack** (Makanan Ringan)

**Perlengkapan Mandi** (Bathroom Supplies)

**Rokok** (Cigarettes)

Waktu pembuatan dan pembaruan data juga diset menggunakan Carbon::now().

### 3. ****Seeding Produk (Items)****

Selanjutnya, lima produk dimasukkan ke dalam tabel items, masing-masing dikaitkan dengan kategori yang sudah dibuat sebelumnya:

**Beras Premium**

**Minyak Goreng 1L**

**Aqua Botol 600ml**

**Indomie Goreng**

**Lifebuoy Sabun**

Setiap produk memiliki nama, kategori (category\_id), harga, stok, satuan, dan deskripsi produk.

### 4. ****Seeding Stok Masuk dan Stok Keluar (Stock In and Stock Out)****

Bagian ini mensimulasikan 50 entri transaksi stok secara acak ke dalam tabel stock\_in dan stock\_out. Setiap entri berisi:

item\_id dan user\_id yang dipilih secara acak dari daftar produk dan pengguna yang ada

Harga satuan stok masuk (harga\_per\_satuan) dan harga satuan stok keluar (harga\_jual\_per\_satuan) yang dipilih secara acak dalam rentang tertentu

Jumlah produk yang masuk dan keluar (jumlah acak antara 1 dan 100 untuk stok masuk dan 1 dan 80 untuk stok keluar)

Satuan produk yang digunakan (misalnya, kg, botol, bungkus, batang, dll.)

Tanggal transaksi stok (tanggal), yang diset menggunakan Carbon::now()->subDays(rand(0, 90)) untuk memilih tanggal secara acak dalam rentang 90 hari terakhir

### Struktur Database yang Digunakan:

**Tabel Users**: Menyimpan data pengguna, termasuk peran (admin atau kasir).

**Tabel Categories**: Menyimpan kategori produk.

**Tabel Items**: Menyimpan informasi produk, termasuk nama, harga, stok, dan kategori.

**Tabel Stock In**: Mencatat transaksi barang masuk ke dalam stok, termasuk harga dan jumlah.

**Tabel Stock Out**: Mencatat transaksi barang keluar dari stok, termasuk harga dan jumlah.

### Kata yang Tepat untuk Seeder Ini:

Jika dijelaskan dalam satu kata atau frasa, skrip ini adalah **"Database Seeder untuk Sistem Inventaris"**. Skrip ini mengisi tabel-tabel yang diperlukan dalam sistem manajemen toko atau inventaris dengan data contoh, termasuk pengguna, kategori produk, barang, dan transaksi stok.

Berikut adalah penjelasan dari kode SuperAdminSeeder.php :

php artisan db:seed --class=SuperAdminSeeder

User::create([

            'name' => 'Super Admin',

            'email' => 'superadmin@example.com',

            'password' => bcrypt('password123'),

            'role' => User::ROLE\_SUPER\_ADMIN,

        ]);

### 🔍 ****Tujuan Umum Kode Ini****

Kode ini digunakan untuk **menambahkan satu akun "Super Admin" ke dalam database** secara otomatis saat proses seeding dijalankan di Laravel.

### 🧩 ****Penjelasan Baris per Baris****

<?php

namespace Database\Seeders;

Baris ini menyatakan bahwa file ini berada dalam namespace Database\Seeders, yang merupakan folder standar untuk seeder di Laravel.

use Illuminate\Database\Console\Seeds\WithoutModelEvents;

use Illuminate\Database\Seeder;

use App\Models\User;

Seeder: Kelas dasar Laravel untuk membuat seeder.

User: Model User yang mewakili tabel users di database.

WithoutModelEvents tidak digunakan di dalam fungsi, jadi ini bisa dihapus jika tidak diperlukan.

class SuperAdminSeeder extends Seeder

Mendefinisikan class bernama SuperAdminSeeder yang **turunan dari class** Seeder, digunakan untuk menjalankan proses seeding ke database.

public function run(): void

Method run() adalah fungsi yang dipanggil saat seeder dijalankan (php artisan db:seed), dan berisi logika untuk menyisipkan data ke database.

User::create([

'name' => 'Super Admin',

'email' => 'superadmin@example.com',

'password' => bcrypt('password123'),

'role' => User::ROLE\_SUPER\_ADMIN,

]);

Baris ini membuat entri baru pada tabel users:

name: Nama pengguna

email: Email pengguna

password: Password yang di-hash menggunakan fungsi bcrypt()

role: Peran user, diambil dari konstanta User::ROLE\_SUPER\_ADMIN (diasumsikan sudah didefinisikan dalam model User, misalnya sebagai const ROLE\_SUPER\_ADMIN = 'superadmin';)

### ✅ ****Kesimpulan****

Seeder ini berguna untuk **menambahkan akun awal** dengan peran **Super Admin** ke tabel users. Ini biasanya digunakan saat pertama kali mengatur sistem agar ada akun dengan hak akses penuh.

Ingin saya bantu buatkan isi model User.php untuk mendukung konstanta ROLE\_SUPER\_ADMIN?

MODEL.

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
use Illuminate\Notifications\Notifiable;  
use Laravel\Sanctum\HasApiTokens; // Wajib ada

class User extends Authenticatable  
{  
/\*\* @use HasFactory<\Database\Factories\UserFactory> \*/  
use HasFactory, Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var list<string>

\*/

protected $fillable = [

'name',

'email',

'username',

'password',

'phone',

'role'

];

const ROLE\_ADMIN = 'admin';

const ROLE\_SUPER\_ADMIN = 'super\_admin';

public function isAdmin(): bool

{

return $this->role === self::ROLE\_ADMIN;

}

public function isSuperAdmin(): bool

{

return $this->role === self::ROLE\_SUPER\_ADMIN;

}

/\*\*

\* The attributes that should be hidden for serialization.

\*

\* @var list<string>

\*/

protected $hidden = [

'password',

];

// Relasi dengan stock\_in dan stock\_out

public function stockIns()

{

return $this->hasMany(StockIn::class);

}

public function stockOuts()

{

return $this->hasMany(StockOut::class);

}

/\*\*

\* Get the attributes that should be cast.

\*

\* @return array<string, string>

\*/

protected function casts(): array

{

return [

'email\_verified\_at' => 'datetime',

'password' => 'hashed',

];

}

}# END

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;

class StockOut extends Model  
{  
use HasFactory;

protected $table = 'stock\_out';

protected $fillable = [

'item\_id',

'harga\_jual\_per\_satuan',

'jumlah',

'satuan\_jumlah',

'keterangan',

'tanggal',

'user\_id'

];

protected $dates = ['tanggal'];

protected $casts = [

'tanggal' => 'date',

];

public function item()

{

return $this->belongsTo(Item::class);

}

public function user()

{

return $this->belongsTo(User::class);

}

// StockOut.php

public function scopeLast30Days($query)  
{  
return $query->where('tanggal', '>=', now()->subDays(30));  
}  
}# END

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;

class StockIn extends Model  
{  
use HasFactory;

protected $table = 'stock\_in';

protected $fillable = [

'item\_id',

'harga\_per\_satuan',

'jumlah',

'satuan\_jumlah',

'keterangan',

'tanggal',

'user\_id'

];

protected $dates = ['tanggal'];

// app\Models\StockIn.php

protected $casts = [

'tanggal' => 'date',

];

public function item()

{

return $this->belongsTo(Item::class);

}

public function user()

{

return $this->belongsTo(User::class);

}

// StockIn.php

public function scopeLast30Days($query)  
{  
return $query->where('tanggal', '>=', now()->subDays(30));  
}

}# END

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
// use Illuminate\Database\Eloquent\Relations\HasMany;

class Item extends Model  
{  
use HasFactory;

protected $fillable = [

'nama',

'category\_id',

'harga',

'satuan\_harga',

'stok',

'satuan\_stok',

'keterangan'

];

public function category(): BelongsTo

{

return $this->belongsTo(Category::class);

}

public function stockIns()

{

return $this->hasMany(StockIn::class);

}

public function stockOuts()

{

return $this->hasMany(StockOut::class);

}

// Accessor for formatted price

public function getFormattedHargaAttribute()

{

return 'Rp ' . number\_format($this->harga, 0, ',', '.');

}

}# END

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;

class Category extends Model  
{  
use HasFactory;

protected $fillable = ['name'];

public function items()

{

return $this->hasMany(Item::class);

}

}

Berikut adalah penjelasan menyeluruh dari kode-kode model Laravel yang kamu lampirkan:

## 1. ****Model**** User

Model ini mewakili pengguna/admin dalam sistem.

### Fitur-fitur penting:

Menggunakan **Authenticatable**: agar bisa login dan terautentikasi.

Ada dua **konstanta peran**:

const ROLE\_ADMIN = 'admin';

const ROLE\_SUPER\_ADMIN = 'super\_admin';

Method isAdmin() dan isSuperAdmin() digunakan untuk mengecek peran pengguna.

### Relasi:

hasMany(StockIn::class) → user bisa melakukan banyak pemasukan stok.

hasMany(StockOut::class) → user juga bisa melakukan banyak pengeluaran stok.

### fillable:

Properti yang bisa diisi massal seperti saat create/update:

'name', 'email', 'username', 'password', 'phone', 'role'

## 2. ****Model**** StockOut

Mewakili **pengeluaran stok barang**.

### Relasi:

belongsTo(Item::class) → stok keluar terkait satu item.

belongsTo(User::class) → stok keluar dicatat oleh satu user.

### Scope:

scopeLast30Days() → untuk mengambil data 30 hari terakhir:

$data = StockOut::last30Days()->get();

## 3. ****Model**** StockIn

Mewakili **pemasukan stok barang**.

### Relasi:

Sama seperti StockOut, relasi dengan Item dan User.

Ada juga scopeLast30Days() untuk filter 30 hari terakhir.

## 4. ****Model**** Item

Mewakili **barang** yang tersedia di sistem.

### Relasi:

belongsTo(Category::class) → satu item memiliki satu kategori.

hasMany(StockIn::class) dan hasMany(StockOut::class) → item punya banyak catatan pemasukan & pengeluaran.

### Aksesori:

getFormattedHargaAttribute() → membuat properti formatted\_harga dengan format rupiah:

echo $item->formatted\_harga; // Rp 10.000

## 5. ****Model**** Category

Mewakili **kategori dari item/barang**.

### Relasi:

hasMany(Item::class) → satu kategori bisa punya banyak item.

## Kesimpulan Hubungan Antar Model

User

├── hasMany StockIn

└── hasMany StockOut

Item

├── belongsTo Category

├── hasMany StockIn

└── hasMany StockOut

StockIn

├── belongsTo User

└── belongsTo Item

StockOut

├── belongsTo User

└── belongsTo Item

Category

└── hasMany Item

Ingin saya buatkan diagram ERD (Entity Relationship Diagram) berdasarkan model ini?

CONTROLLER:

LOGIN.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

class LoginController extends Controller

{

    /\*\*

     \* Show the login form

     \*/

    public function showLoginForm()

    {

        return view('auth.login');

    }

    /\*\*

     \* Handle login request

     \*/

    protected function rememberDuration()

{

    return 24 \* 60; // 1 days in minutes

}

    public function login(Request $request)

    {

        // Validate input

        $credentials = $request->validate([

            'email' => 'required|email',

            'password' => 'required'

        ]);

        $remember = $request->has('remember');

        // Attempt to authenticate user

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

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

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

                ->with('success', 'Login berhasil!');

        }

        // If authentication fails

        return back()

            ->withErrors([

                'email' => 'Email atau password salah.',

            ])

            ->onlyInput('email');

    }

    public function showRegisterForm()

{

    return view('auth.register');

}

// app/Http/Controllers/LoginController.php

public function register(Request $request)

{

    $validated = $request->validate([

        'name' => 'required|string|max:255',

        'username' => 'nullable|string|unique:users,username',

        'email' => 'required|string|email|max:255|unique:users',

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

    ]);

    $user = User::create([

        'name' => $validated['name'],

        'username' => $validated['username'],

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

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

        'role' => User::ROLE\_ADMIN, // Default role admin

    ]);

    // Auth::login($user);

    return redirect('/login')->with('success', 'Registrasi berhasil!');

}

public function showForgotPasswordForm()

{

    return view('auth.forgot\_password');

}

public function sendResetLink(Request $request)

{

    $request->validate(['email' => 'required|email']);

    $status = \Password::sendResetLink(

        $request->only('email')

    );

    return $status === \Password::RESET\_LINK\_SENT

        ? back()->with(['status' => \_\_($status)])

        : back()->withErrors(['email' => \_\_($status)]);

}

    /\*\*

     \* Handle logout request

     \*/

    public function logout(Request $request)

    {

        Auth::logout();

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

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

        return redirect('/login')

            ->with('success', 'Anda telah logout.');

    }

}

Berikut penjelasan kode pada LoginController di Laravel:

### 📄 ****Namespace & Import****

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

LoginController berada di namespace App\Http\Controllers.

Mengimpor model User, Request, dan helper seperti Auth (untuk autentikasi) serta Hash (untuk hashing password).

### ✅ ****Menampilkan Form Login****

public function showLoginForm()

{

return view('auth.login');

}

Menampilkan halaman login (resources/views/auth/login.blade.php).

### 🔐 ****Proses Login****

public function login(Request $request)

{

// Validasi input dari form login

$credentials = $request->validate([

'email' => 'required|email',

'password' => 'required'

]);

$remember = $request->has('remember'); // Cek apakah opsi "remember me" dicentang

// Mencoba login

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

$request->session()->regenerate(); // Mencegah session fixation

return redirect()->intended('/dashboard')->with('success', 'Login berhasil!');

}

// Jika gagal

return back()->withErrors(['email' => 'Email atau password salah.'])->onlyInput('email');

}

Validasi form login.

Gunakan Auth::attempt() untuk login.

Jika berhasil, arahkan ke dashboard.

Jika gagal, tampilkan pesan error.

### 🕒 ****Durasi Remember Me****

protected function rememberDuration()

{

return 24 \* 60; // 1 hari dalam menit

}

Fungsi opsional yang bisa digunakan untuk mengatur durasi fitur "remember me" (meskipun dalam kode ini belum dipakai aktif).

### 📝 ****Menampilkan Form Registrasi****

public function showRegisterForm()

{

return view('auth.register');

}

Menampilkan form registrasi.

### 👤 ****Proses Registrasi****

public function register(Request $request)

{

$validated = $request->validate([

'name' => 'required|string|max:255',

'username' => 'nullable|string|unique:users,username',

'email' => 'required|string|email|max:255|unique:users',

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

]);

$user = User::create([

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

'username' => $validated['username'],

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

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

'role' => User::ROLE\_ADMIN, // defaultnya role admin

]);

return redirect('/login')->with('success', 'Registrasi berhasil!');

}

Validasi data registrasi.

Simpan user baru ke database, termasuk enkripsi password.

Redirect ke login setelah berhasil.

### 🔁 ****Form & Proses Lupa Password****

public function showForgotPasswordForm()

{

return view('auth.forgot\_password');

}

public function sendResetLink(Request $request)

{

$request->validate(['email' => 'required|email']);

$status = \Password::sendResetLink(

$request->only('email')

);

return $status === \Password::RESET\_LINK\_SENT

? back()->with(['status' => \_\_($status)])

: back()->withErrors(['email' => \_\_($status)]);

}

Menampilkan form lupa password.

Mengirim link reset password melalui email menggunakan helper Password.

### 🔓 ****Logout****

public function logout(Request $request)

{

Auth::logout(); // Logout user

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

$request->session()->regenerateToken(); // Regenerasi token CSRF

return redirect('/login')->with('success', 'Anda telah logout.');

}

Mengakhiri sesi user dan kembali ke halaman login.

DASBOARD.

<?php

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\Category;

use App\Models\StockIn;

use App\Models\StockOut;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Carbon;

class DashboardController extends Controller

{

    public function index()

    {

        // Data untuk 12 bulan terakhir

        $months = collect(range(0, 11))->map(function ($i) {

            return Carbon::now()->subMonths($i);

        })->reverse();

        // Data stok masuk/keluar

        $stockData = $months->map(function ($month) {

            $stockIn = StockIn::whereMonth('tanggal', $month->month)

                ->whereYear('tanggal', $month->year)

                ->sum('jumlah');

            $stockOut = StockOut::whereMonth('tanggal', $month->month)

                ->whereYear('tanggal', $month->year)

                ->sum('jumlah');

            return [

                'month' => $month->format('M Y'),

                'stock\_in' => $stockIn,

                'stock\_out' => $stockOut,

                'net\_stock' => $stockIn - $stockOut,

                'percentage\_change' => $this->calculatePercentageChange($month)

            ];

        });

        // Data untuk pie chart kategori

        $categoriesData = Category::withCount('items')

            ->orderByDesc('items\_count')

            ->limit(5)

            ->get();

        // Hitung persentase kategori

        $totalItems = Item::count();

        $categoriesData->each(function ($category) use ($totalItems) {

            $category->percentage = $totalItems > 0

                ? round(($category->items\_count / $totalItems) \* 100, 2)

                : 0;

        });

        // Data item dengan stok rendah

        $lowStockItems = Item::where('stok', '<', 10)

            ->orderBy('stok')

            ->limit(5)

            ->get()

            ->each(function ($item) {

                $item->percentage = ($item->stok / 10) \* 100; // Persentase dari ambang batas 10

            });

        // Top 5 item paling aktif

        $activeItems = Item::withSum('stockIns as total\_in', 'jumlah')

            ->withSum('stockOuts as total\_out', 'jumlah')

            ->orderByDesc(DB::raw('total\_in + total\_out'))

            ->limit(5)

            ->get()

            ->each(function ($item) {

                $item->total\_activity = $item->total\_in + $item->total\_out;

            });

        // Hitung total stok global

        $totalStockIn = StockIn::sum('jumlah');

        $totalStockOut = StockOut::sum('jumlah');

        $currentStock = Item::sum('stok');

        $totalTransactions = $totalStockIn + $totalStockOut;

        // Statistik

        $stats = [

            'total\_items' => Item::count(),

            'stockin\_30' => StockIn::last30Days()->sum('jumlah'),

            'stockout\_30' => StockOut::last30Days()->sum('jumlah'),

            'low\_stock' => Item::where('stok', '<', 10)->count(),

            'total\_categories' => Category::count(),

            'total\_stock\_in' => $totalStockIn,

            'total\_stock\_out' => $totalStockOut,

            'current\_stock' => $currentStock,

            'stock\_in\_percentage' => $totalTransactions > 0 ? round(($totalStockIn / $totalTransactions) \* 100, 2) : 0,

            'stock\_out\_percentage' => $totalTransactions > 0 ? round(($totalStockOut / $totalTransactions) \* 100, 2) : 0,

            'avg\_daily\_stockin' => StockIn::count() > 0

                ? round($totalStockIn / StockIn::distinct('tanggal')->count(), 2)

                : 0,

            'avg\_daily\_stockout' => StockOut::count() > 0

                ? round($totalStockOut / StockOut::distinct('tanggal')->count(), 2)

                : 0,

        ];

        return view('dashboard.index', compact(

            'stockData',

            'stats',

            'categoriesData',

            'lowStockItems',

            'activeItems'

        ));

    }

    private function calculatePercentageChange($month)

    {

        $prevMonth = $month->copy()->subMonth();

        $currentIn = StockIn::whereMonth('tanggal', $month->month)

            ->whereYear('tanggal', $month->year)

            ->sum('jumlah');

        $prevIn = StockIn::whereMonth('tanggal', $prevMonth->month)

            ->whereYear('tanggal', $prevMonth->year)

            ->sum('jumlah');

        $currentOut = StockOut::whereMonth('tanggal', $month->month)

            ->whereYear('tanggal', $month->year)

            ->sum('jumlah');

        $prevOut = StockOut::whereMonth('tanggal', $prevMonth->month)

            ->whereYear('tanggal', $prevMonth->year)

            ->sum('jumlah');

        return [

            'in' => $prevIn > 0 ? round((($currentIn - $prevIn) / $prevIn) \* 100, 2) : 0,

            'out' => $prevOut > 0 ? round((($currentOut - $prevOut) / $prevOut) \* 100, 2) : 0

        ];

    }

}

Kode DashboardController ini merupakan bagian dari backend aplikasi Laravel yang bertugas menyajikan **data untuk dashboard** sistem manajemen stok. Berikut adalah penjelasan bagian per bagian:

### 🧠 FUNGSI UTAMA: index()

#### 1. ****Data untuk 12 Bulan Terakhir****

$months = collect(range(0, 11))->map(function ($i) {

return Carbon::now()->subMonths($i);

})->reverse();

Mengambil 12 bulan terakhir dari bulan sekarang.

Digunakan untuk grafik tren stok masuk dan keluar.

#### 2. ****Data Stok Masuk/Keluar per Bulan****

$stockData = $months->map(function ($month) {

...

});

Mengambil total jumlah dari StockIn dan StockOut berdasarkan bulan dan tahun.

net\_stock: selisih antara masuk dan keluar.

percentage\_change: memanggil fungsi calculatePercentageChange() untuk menghitung perubahan stok dibanding bulan sebelumnya.

### 📊 GRAFIK PIE KATEGORI

$categoriesData = Category::withCount('items')->orderByDesc('items\_count')->limit(5)->get();

Mengambil 5 kategori teratas berdasarkan jumlah item.

Menghitung persentase item dari setiap kategori dibanding total semua item.

### ⚠️ ITEM DENGAN STOK RENDAH

$lowStockItems = Item::where('stok', '<', 10)->orderBy('stok')->limit(5)->get();

Mengambil 5 item dengan stok terendah (kurang dari 10).

percentage: persentase dibanding ambang batas 10.

### 🔁 ITEM PALING AKTIF

$activeItems = Item::withSum('stockIns as total\_in', 'jumlah')

->withSum('stockOuts as total\_out', 'jumlah')

->orderByDesc(DB::raw('total\_in + total\_out'))

->limit(5)

->get();

Mengambil 5 item dengan total aktivitas stok (masuk + keluar) tertinggi.

total\_activity: penjumlahan dari total masuk dan keluar.

### 📈 STATISTIK UMUM

$stats = [

'total\_items' => Item::count(),

'stockin\_30' => StockIn::last30Days()->sum('jumlah'),

...

];

Statistik dasar seperti:

Jumlah item, kategori, dan item dengan stok rendah.

Total stok masuk dan keluar.

Rata-rata stok masuk/keluar per hari (berdasarkan jumlah tanggal unik).

Persentase stok masuk dan keluar dari total transaksi.

### 📐 FUNGSI BANTUAN: calculatePercentageChange($month)

Menghitung **persentase perubahan** stok masuk dan keluar bulan ini dibandingkan bulan sebelumnya.

Formula:

(current - previous) / previous \* 100

Jika tidak ada data bulan sebelumnya (prev = 0), maka dikembalikan 0 untuk menghindari pembagian oleh nol.

### 📄 VIEW

return view('dashboard.index', compact(...));

Mengirim semua data ke tampilan dashboard.index untuk ditampilkan dalam UI.

### ✅ INTI LOGIKA DASHBOARD

Menyediakan **analitik lengkap** mengenai:

Aktivitas stok masuk/keluar

Item dengan stok rendah

Kategori populer

Item paling aktif

Statistik dan tren dalam periode waktu

ITEM:

<?php

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\Category;

use Illuminate\Http\Request;

class ItemController extends Controller

{

    public function index()

    {

        $items = Item::with('category')->get();

        return view('data\_barang.index', compact('items'));

    }

    public function create()

    {

        $categories = Category::all();

        return view('data\_barang.create', compact('categories'));

    }

    public function store(Request $request)

    {

        $request->validate([

            'nama' => 'required|string|max:255',

            'category\_id' => 'required|exists:categories,id',

            'harga' => 'required|numeric|min:0',

            'satuan\_harga' => 'required|string|max:50',

            'stok' => 'required|integer|min:0',

            'satuan\_stok' => 'required|string|max:50',

            'keterangan' => 'nullable|string'

        ]);

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

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

            ->with('success', 'Data barang berhasil ditambahkan');

    }

    public function edit($id)

    {

        $item = Item::findOrFail($id);

        $categories = Category::all();

        return view('data\_barang.edit', compact('item', 'categories'));

    }

    public function update(Request $request, $id)

    {

        $request->validate([

            'nama' => 'required|string|max:255',

            'category\_id' => 'required|exists:categories,id',

            'harga' => 'required|numeric|min:0',

            'satuan\_harga' => 'required|string|max:50',

            'stok' => 'required|integer|min:0',

            'satuan\_stok' => 'required|string|max:50',

            'keterangan' => 'nullable|string'

        ]);

        $item = Item::findOrFail($id);

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

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

            ->with('success', 'Data barang berhasil diperbarui');

    }

    public function destroy($id)

    {

        $item = Item::findOrFail($id);

        $item->delete();

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

            ->with('success', 'Data barang berhasil dihapus');

    }

}

Berikut adalah penjelasan kode pada ItemController di Laravel, yang digunakan untuk **mengelola data barang** (CRUD) dan relasinya dengan kategori:

## 📂 ****Namespace dan Import****

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\Category;

use Illuminate\Http\Request;

ItemController berada di namespace App\Http\Controllers.

Menggunakan model Item dan Category, serta class Request dari Laravel.

## 🧾 ****Fungsi-fungsi CRUD****

### 1. 📋 index(): Menampilkan Semua Barang

public function index()

{

$items = Item::with('category')->get();

return view('data\_barang.index', compact('items'));

}

Mengambil semua data barang dengan relasi ke tabel kategori (with('category')).

Menampilkan ke view data\_barang/index.blade.php.

### 2. ➕ create(): Menampilkan Form Tambah Barang

public function create()

{

$categories = Category::all();

return view('data\_barang.create', compact('categories'));

}

Mengambil semua kategori untuk pilihan dropdown.

Menampilkan form tambah barang di view data\_barang/create.blade.php.

### 3. ✅ store(Request $request): Menyimpan Barang Baru

public function store(Request $request)

{

$request->validate([

'nama' => 'required|string|max:255',

'category\_id' => 'required|exists:categories,id',

'harga' => 'required|numeric|min:0',

'satuan\_harga' => 'required|string|max:50',

'stok' => 'required|integer|min:0',

'satuan\_stok' => 'required|string|max:50',

'keterangan' => 'nullable|string'

]);

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

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

->with('success', 'Data barang berhasil ditambahkan');

}

Validasi input form.

Menyimpan data baru ke database menggunakan Item::create().

Redirect ke halaman daftar barang dengan pesan sukses.

### 4. ✏️ edit($id): Menampilkan Form Edit Barang

public function edit($id)

{

$item = Item::findOrFail($id);

$categories = Category::all();

return view('data\_barang.edit', compact('item', 'categories'));

}

Mengambil data barang berdasarkan ID.

Mengambil semua kategori untuk dropdown.

Menampilkan form edit pada data\_barang/edit.blade.php.

### 5. 🔄 update(Request $request, $id): Memperbarui Data Barang

public function update(Request $request, $id)

{

$request->validate([

'nama' => 'required|string|max:255',

'category\_id' => 'required|exists:categories,id',

'harga' => 'required|numeric|min:0',

'satuan\_harga' => 'required|string|max:50',

'stok' => 'required|integer|min:0',

'satuan\_stok' => 'required|string|max:50',

'keterangan' => 'nullable|string'

]);

$item = Item::findOrFail($id);

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

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

->with('success', 'Data barang berhasil diperbarui');

}

Validasi data form.

Mengupdate data barang berdasarkan ID.

### 6. 🗑️ destroy($id): Menghapus Barang

public function destroy($id)

{

$item = Item::findOrFail($id);

$item->delete();

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

->with('success', 'Data barang berhasil dihapus');

}

Menghapus data barang berdasarkan ID.

## 💡 Catatan Tambahan

Pastikan relasi Item dengan Category sudah didefinisikan di model Item:

public function category()

{

return $this->belongsTo(Category::class);

}

File view berada di folder resources/views/data\_barang.

STOKIN:

<?php

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\StockIn;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class StockInController extends Controller

{

    public function index()

    {

        $stockIns = StockIn::with(['item', 'user'])->latest()->get();

        return view('stock\_in.index', compact('stockIns'));

    }

    public function create()

    {

        $items = Item::all();

        return view('stock\_in.create', compact('items'));

    }

    public function store(Request $request)

    {

        $validated = $request->validate([

            'item\_id' => 'required|exists:items,id',

            'harga\_per\_satuan' => 'required|numeric|min:0',

            'jumlah' => 'required|integer|min:1',

            'satuan\_jumlah' => 'required|string',

            'tanggal' => 'required|date',

            'keterangan' => 'nullable|string'

        ]);

        $stockIn = StockIn::create([

            'item\_id' => $validated['item\_id'],

            'harga\_per\_satuan' => $validated['harga\_per\_satuan'],

            'jumlah' => $validated['jumlah'],

            'satuan\_jumlah' => $validated['satuan\_jumlah'],

            'tanggal' => $validated['tanggal'],

            'keterangan' => $validated['keterangan'],

            'user\_id' => Auth::id()

        ]);

        // Update stok item

        $item = Item::find($validated['item\_id']);

        $item->increment('stok', $validated['jumlah']);

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

            ->with('success', 'Stok masuk berhasil dicatat');

    }

    public function edit(StockIn $stockIn)

    {

        $items = Item::all();

        return view('stock\_in.edit', compact('stockIn', 'items'));

    }

    public function update(Request $request, StockIn $stockIn)

    {

        $validated = $request->validate([

            'item\_id' => 'required|exists:items,id',

            'harga\_per\_satuan' => 'required|numeric|min:0',

            'jumlah' => 'required|integer|min:1',

            'satuan\_jumlah' => 'required|string',

            'tanggal' => 'required|date',

            'keterangan' => 'nullable|string'

        ]);

        $oldItemId = $stockIn->item\_id;

        $oldJumlah = $stockIn->jumlah;

        // Update data stok masuk

        $stockIn->update([

            'item\_id' => $validated['item\_id'],

            'harga\_per\_satuan' => $validated['harga\_per\_satuan'],

            'jumlah' => $validated['jumlah'],

            'satuan\_jumlah' => $validated['satuan\_jumlah'],

            'tanggal' => $validated['tanggal'],

            'keterangan' => $validated['keterangan'],

            'user\_id' => Auth::id()

        ]);

        // Jika item berubah, kembalikan stok lama lalu tambahkan stok baru

        if ($oldItemId != $validated['item\_id']) {

            Item::find($oldItemId)->decrement('stok', $oldJumlah);

            Item::find($validated['item\_id'])->increment('stok', $validated['jumlah']);

        } else {

            // Jika item sama, hanya update selisih jumlah

            $difference = $validated['jumlah'] - $oldJumlah;

            Item::find($validated['item\_id'])->increment('stok', $difference);

        }

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

            ->with('success', 'Stok masuk berhasil diperbarui');

    }

    public function destroy(StockIn $stockIn)

    {

        // Kurangi stok item terkait

        $item = $stockIn->item;

        if ($item && $stockIn->jumlah <= $item->stok) {

            $item->decrement('stok', $stockIn->jumlah);

        }

        $stockIn->delete();

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

            ->with('success', 'Stok masuk berhasil dihapus');

    }

}

Berikut adalah penjelasan kode dari StockInController di Laravel, yang digunakan untuk **mengelola proses** stok masuk **(barang masuk)** ke dalam sistem inventori:

## 📂 ****Namespace dan Import****

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\StockIn;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

Mengatur controller dalam namespace App\Http\Controllers.

Menggunakan model Item dan StockIn.

Menggunakan Request untuk menangani input pengguna.

Menggunakan Auth untuk mengambil ID user yang sedang login.

## 🧾 ****Fungsi-fungsi CRUD untuk Stok Masuk****

### 1. 📋 index(): Menampilkan Semua Data Stok Masuk

public function index()

{

$stockIns = StockIn::with(['item', 'user'])->latest()->get();

return view('stock\_in.index', compact('stockIns'));

}

Mengambil seluruh data stok masuk dan relasi dengan item dan user (yang mencatat).

Disusun dari yang terbaru (latest()).

Ditampilkan pada view stock\_in/index.blade.php.

### 2. ➕ create(): Menampilkan Form Input Stok Masuk

public function create()

{

$items = Item::all();

return view('stock\_in.create', compact('items'));

}

Mengambil semua data barang untuk ditampilkan dalam form pemilihan barang.

View: stock\_in/create.blade.php.

### 3. ✅ store(Request $request): Menyimpan Data Stok Masuk Baru

public function store(Request $request)

{

$validated = $request->validate([

'item\_id' => 'required|exists:items,id',

'harga\_per\_satuan' => 'required|numeric|min:0',

'jumlah' => 'required|integer|min:1',

'satuan\_jumlah' => 'required|string',

'tanggal' => 'required|date',

'keterangan' => 'nullable|string'

]);

$stockIn = StockIn::create([

... // (data validasi)

'user\_id' => Auth::id()

]);

// Tambahkan jumlah ke stok barang

$item = Item::find($validated['item\_id']);

$item->increment('stok', $validated['jumlah']);

}

**Validasi input**.

Menyimpan data stok masuk ke DB dengan user yang login (Auth::id()).

**Update stok barang** (increment).

### 4. ✏️ edit(StockIn $stockIn): Form Edit Stok Masuk

public function edit(StockIn $stockIn)

{

$items = Item::all();

return view('stock\_in.edit', compact('stockIn', 'items'));

}

Mengambil data stok masuk yang akan diedit dan semua barang yang tersedia.

### 5. 🔄 update(Request $request, StockIn $stockIn): Memperbarui Data Stok Masuk

public function update(Request $request, StockIn $stockIn)

{

$validated = $request->validate([...]);

$oldItemId = $stockIn->item\_id;

$oldJumlah = $stockIn->jumlah;

$stockIn->update([...]);

if ($oldItemId != $validated['item\_id']) {

// Jika barang berubah, kembalikan stok lama, lalu tambah stok baru

Item::find($oldItemId)->decrement('stok', $oldJumlah);

Item::find($validated['item\_id'])->increment('stok', $validated['jumlah']);

} else {

// Jika barang sama, hanya selisih jumlah yang diperhitungkan

$difference = $validated['jumlah'] - $oldJumlah;

Item::find($validated['item\_id'])->increment('stok', $difference);

}

}

Validasi input.

Jika **barang berubah**, maka:

Kurangi stok dari barang lama.

Tambah stok ke barang baru.

Jika **barang sama**, hanya update selisih jumlah.

### 6. 🗑️ destroy(StockIn $stockIn): Menghapus Data Stok Masuk

public function destroy(StockIn $stockIn)

{

// Kurangi stok item jika mencukupi

$item = $stockIn->item;

if ($item && $stockIn->jumlah <= $item->stok) {

$item->decrement('stok', $stockIn->jumlah);

}

$stockIn->delete();

}

Kurangi stok barang sesuai jumlah yang akan dihapus.

Hanya dikurangi jika **stok cukup** (<= $item->stok).

Hapus data stok masuk dari database.

## 💡 Ringkasan

**StockInController** mengelola proses **pencatatan stok masuk**.

Setiap penambahan/ubah/hapus stok masuk akan **mengubah stok pada tabel** items.

Data stok masuk dikaitkan dengan **barang (**item**) dan user (**Auth::id()**)**.

STOKOUT:

<?php

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\StockOut;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class StockOutController extends Controller

{

    public function index()

    {

        $stockOuts = StockOut::with(['item', 'user'])->latest()->get();

        return view('stock\_out.index', compact('stockOuts'));

    }

    public function create()

    {

        $items = Item::all();

        return view('stock\_out.create', compact('items'));

    }

    public function store(Request $request)

    {

        $validated = $request->validate([

            'item\_id' => 'required|exists:items,id',

            'harga\_jual\_per\_satuan' => 'required|numeric|min:0',

            'jumlah' => 'required|integer|min:1',

            'satuan\_jumlah' => 'required|string',

            'tanggal' => 'required|date',

            'keterangan' => 'nullable|string'

        ]);

        $stockOut = StockOut::create([

            'item\_id' => $validated['item\_id'],

            'harga\_jual\_per\_satuan' => $validated['harga\_jual\_per\_satuan'],

            'jumlah' => $validated['jumlah'],

            'satuan\_jumlah' => $validated['satuan\_jumlah'],

            'tanggal' => $validated['tanggal'],

            'keterangan' => $validated['keterangan'],

            'user\_id' => Auth::id()

        ]);

        // Update stok item (dikurangi)

        $item = Item::find($validated['item\_id']);

        $item->decrement('stok', $validated['jumlah']);

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

            ->with('success', 'Stok keluar berhasil dicatat');

    }

    public function edit(StockOut $stockOut)

    {

        $items = Item::all();

        return view('stock\_out.edit', compact('stockOut', 'items'));

    }

    public function update(Request $request, StockOut $stockOut)

    {

        $validated = $request->validate([

            'item\_id' => 'required|exists:items,id',

            'harga\_jual\_per\_satuan' => 'required|numeric|min:0',

            'jumlah' => 'required|integer|min:1',

            'satuan\_jumlah' => 'required|string',

            'tanggal' => 'required|date',

            'keterangan' => 'nullable|string'

        ]);

        $oldItemId = $stockOut->item\_id;

        $oldJumlah = $stockOut->jumlah;

        // Update data stok keluar

        $stockOut->update([

            'item\_id' => $validated['item\_id'],

            'harga\_jual\_per\_satuan' => $validated['harga\_jual\_per\_satuan'],

            'jumlah' => $validated['jumlah'],

            'satuan\_jumlah' => $validated['satuan\_jumlah'],

            'tanggal' => $validated['tanggal'],

            'keterangan' => $validated['keterangan'],

            'user\_id' => Auth::id()

        ]);

        // Jika item berubah

        if ($oldItemId != $validated['item\_id']) {

            // Kembalikan stok item lama

            Item::find($oldItemId)->increment('stok', $oldJumlah);

            // Kurangi stok item baru

            Item::find($validated['item\_id'])->decrement('stok', $validated['jumlah']);

        } else {

            // Jika item sama, update selisih jumlah

            $difference = $validated['jumlah'] - $oldJumlah;

            Item::find($validated['item\_id'])->decrement('stok', $difference);

        }

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

            ->with('success', 'Stok keluar berhasil diperbarui');

    }

    public function destroy(StockOut $stockOut)

    {

        // Kembalikan stok item

        $item = $stockOut->item;

        $item->increment('stok', $stockOut->jumlah);

        $stockOut->delete();

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

            ->with('success', 'Stok keluar berhasil dihapus');

    }

}

Penjelasan berikut hanya mencakup bagian-bagian yang **belum dijelaskan pada** StockInController, karena fungsionalitas dasarnya mirip (CRUD dengan penyesuaian stok). Yang berbeda dalam StockOutController adalah **proses pengurangan stok** dan penanganan kondisi stok keluar. Berikut penjelasan khususnya:

### Yang Berbeda & Belum Dijelaskan di StockInController:

#### 1. harga\_jual\_per\_satuan

Digunakan pada StockOutController, sedangkan di StockInController menggunakan harga\_per\_satuan.

Fungsinya untuk mencatat **harga jual** satuan dari item yang keluar, biasanya digunakan untuk analisis penjualan atau laporan keuntungan.

#### 2. store() - Pengurangan Stok

$item->decrement('stok', $validated['jumlah']);

**Stok barang dikurangi** saat terjadi stock out, artinya barang keluar dari gudang.

Pastikan logika ini tidak menyebabkan nilai stok negatif (tidak dicek di sini, sebaiknya ditambahkan validasi stok cukup).

#### 3. update() - Penyesuaian Stok Jika Item Sama

$difference = $validated['jumlah'] - $oldJumlah;

Item::find($validated['item\_id'])->decrement('stok', $difference);

Bila barang **sama**, maka cukup kurangi stok dengan selisih jumlah lama dan baru.

Misalnya: sebelumnya keluar 3 unit, lalu diubah jadi 5 unit → stok dikurangi 2 lagi.

#### 4. update() - Jika Item Berubah

Item::find($oldItemId)->increment('stok', $oldJumlah);

Item::find($validated['item\_id'])->decrement('stok', $validated['jumlah']);

Jika jenis item berubah, maka:

**Item lama dikembalikan** stoknya.

**Item baru dikurangi** stoknya.

Ini menjaga integritas stok meski data diedit.

#### 5. destroy() - Mengembalikan Stok Saat Data Dihapus

$item->increment('stok', $stockOut->jumlah);

Saat data **stok keluar dihapus**, stok dikembalikan karena dianggap pengeluaran tersebut batal.

Ini kebalikan dari StockIn, yang justru mengurangi stok jika data pemasukan dihapus

### Rangkuman Perbedaan Inti StockOutController

| **Fitur** | **StockInController** | **StockOutController** |
| --- | --- | --- |
| Harga | harga\_per\_satuan (beli) | harga\_jual\_per\_satuan (jual) |
| Perubahan Stok | Menambah stok | Mengurangi stok |
| Hapus Data | Kurangi stok | Kembalikan stok |
| Update Item | Kembalikan stok lama lalu tambah baru | Kembalikan stok lama lalu kurangi baru |

USERCONTROLLER:

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Validation\Rules;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Gate;

class UserController extends Controller

{

    public function index()

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        $users = User::orderBy('name')->get();

        return view('users.index', [

            'users' => $users,

            'roles' => [

                User::ROLE\_ADMIN => 'Admin',

                User::ROLE\_SUPER\_ADMIN => 'Super Admin'

            ]

        ]);

    }

    public function create()

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        return view('users.create', [

            'roles' => [

                User::ROLE\_ADMIN => 'Admin',

                User::ROLE\_SUPER\_ADMIN => 'Super Admin'

            ]

        ]);

    }

    public function store(Request $request)

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        $request->validate([

            'name' => 'required|string|max:255',

            'email' => 'required|string|email|max:255|unique:users',

            'username' => 'nullable|string|max:255|unique:users',

            'phone' => 'nullable|string|max:20',

            'password' => ['required', 'confirmed', Rules\Password::defaults()],

            'role' => 'required|in:' . User::ROLE\_ADMIN . ',' . User::ROLE\_SUPER\_ADMIN,

        ]);

        User::create([

            'name' => $request->name,

            'email' => $request->email,

            'username' => $request->username,

            'phone' => $request->phone,

            'password' => Hash::make($request->password),

            'role' => $request->role,

        ]);

        return redirect()->route('users.index')->with('success', 'User berhasil dibuat');

    }

    public function edit(User $user)

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        return view('users.edit', [

            'user' => $user,

            'roles' => [

                User::ROLE\_ADMIN => 'Admin',

                User::ROLE\_SUPER\_ADMIN => 'Super Admin'

            ]

        ]);

    }

    public function update(Request $request, User $user)

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        $request->validate([

            'name' => 'required|string|max:255',

            'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,

            'username' => 'nullable|string|max:255|unique:users,username,' . $user->id,

            'phone' => 'nullable|string|max:20',

            'password' => ['nullable', 'confirmed', Rules\Password::defaults()],

    'role' => 'required|in:'.User::ROLE\_ADMIN.','.User::ROLE\_SUPER\_ADMIN,

        ]);

        $data = [

            'name' => $request->name,

            'email' => $request->email,

            'username' => $request->username,

            'phone' => $request->phone,

            'role' => $request->role,

        ];

        if ($request->password) {

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

        }

        $user->update($data);

        return redirect()->route('users.index')->with('success', 'User berhasil diupdate');

    }

    public function destroy(User $user)

    {

        if (!Gate::allows('manage-users')) {

            abort(403);

        }

        if (auth()->id() === $user->id) {

            return back()->with('error', 'Tidak bisa menghapus akun sendiri');

        }

        $user->delete();

        return redirect()->route('users.index')->with('success', 'User berhasil dihapus');

    }

    public function profile()

    {

        return view('users.profile', [

            'user' => auth()->user()

        ]);

    }

    public function updateProfile(Request $request)

    {

        $user = auth()->user();

        $request->validate([

            'name' => 'required|string|max:255',

            'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,

            'username' => 'nullable|string|max:255|unique:users,username,' . $user->id,

            'phone' => 'nullable|string|max:20',

    'password' => ['nullable', 'confirmed', Rules\Password::defaults()],

        ]);

        $data = [

            'name' => $request->name,

            'email' => $request->email,

            'username' => $request->username,

            'phone' => $request->phone,

        ];

        if ($request->password) {

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

        }

        $user->update($data);

        return back()->with('success', 'Profile berhasil diupdate');

    }

}

Berikut adalah penjelasan dari **kode** UserController.php yang **belum dijelaskan dalam penjelasan** StockInController sebelumnya:

### ****1.**** use Illuminate\Support\Facades\Gate;

Digunakan untuk **mengontrol otorisasi** berdasarkan kebijakan tertentu (authorization gate). Dalam hal ini, digunakan untuk memastikan hanya user yang memiliki hak akses (manage-users) yang bisa mengelola data user.

### ****2.**** Gate::allows('manage-users')

Baris ini digunakan untuk **mengecek apakah user saat ini memiliki izin untuk mengelola user**. Jika tidak, maka proses akan dihentikan dengan abort(403) yang berarti akses ditolak (forbidden).  
Contoh penggunaannya ada di semua method penting seperti index, create, store, edit, update, dan destroy.

### ****3.**** Rules\Password::defaults()

Digunakan untuk menerapkan aturan password **default dari Laravel**, biasanya termasuk panjang minimal, kombinasi huruf besar dan angka, dsb. Ini dipakai saat membuat atau mengubah password (store, update, updateProfile).

### ****4. Penanganan Role User****

'roles' => [

User::ROLE\_ADMIN => 'Admin',

User::ROLE\_SUPER\_ADMIN => 'Super Admin'

]

Menentukan bahwa hanya dua jenis peran (Admin dan Super Admin) yang tersedia untuk user. Konstanta ini biasanya didefinisikan di model User.

### ****5. Validasi Unik pada Update****

'unique:users,email,' . $user->id

Berarti saat mengupdate user, sistem akan mengecek apakah email sudah digunakan **oleh user lain**, bukan oleh user itu sendiri. Ini untuk mencegah konflik validasi saat user tidak mengubah email/username-nya.

### ****6. Cek agar tidak bisa menghapus akun sendiri****

if (auth()->id() === $user->id) {

return back()->with('error', 'Tidak bisa menghapus akun sendiri');

}

Baris ini **melindungi user agar tidak menghapus akunnya sendiri**, yang bisa menyebabkan kehilangan akses.

### ****7. Fungsi Profil (****profile() ****dan**** updateProfile()****)****

Metode profile() menampilkan profil user yang sedang login.  
updateProfile() digunakan untuk memperbarui informasi pribadi user sendiri, seperti nama, email, username, telepon, dan password.

### Rangkuman Fitur Tambahan yang Tidak Ada di StockInController:

| **Fitur** | **Penjelasan Singkat** |
| --- | --- |
| Gate authorization | Mengatur hak akses |
| Manajemen Role | Admin & Super Admin |
| Password rule default Laravel | Validasi keamanan kata sandi |
| Validasi unik saat update | Hindari konflik data user |
| Tidak bisa hapus akun sendiri | Perlindungan pengguna |
| Halaman profil & update data user sendiri | Fitur untuk user aktif |

FORGET PASSWORD:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Password;

use Illuminate\Support\Facades\Hash;

use App\Models\User;

class ResetPasswordController extends Controller

{

    public function showResetForm(Request $request, $token)

    {

        return view('auth.reset\_password', [

            'token' => $token,

            'email' => $request->email

        ]);

    }

    public function reset(Request $request)

    {

        $request->validate([

            'token' => 'required',

            'email' => 'required|email',

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

        ]);

        $status = Password::reset(

            $request->only('email', 'password', 'password\_confirmation', 'token'),

            function (User $user, $password) {

                $user->forceFill([

                    'password' => Hash::make($password)

                ])->save();

            }

        );

        return $status == Password::PASSWORD\_RESET

                    ? redirect('/login')->with('status', \_\_($status))

                    : back()->withErrors(['email' => [\_\_($status)]]);

    }

}

Berikut adalah **penjelasan lengkap kode** ResetPasswordController yang digunakan untuk **mengatur proses reset password pengguna** di Laravel:

## 📄 File: ResetPasswordController.php

Digunakan untuk menangani proses **reset password melalui email**, biasanya setelah user mengklik tautan reset password dari email yang dikirim oleh sistem.

### ✅ 1. showResetForm(Request $request, $token)

**Fungsi ini menampilkan halaman form reset password.**

return view('auth.reset\_password', [

'token' => $token,

'email' => $request->email

]);

$token adalah token reset yang dikirim via email.

$request->email digunakan agar form tetap menyertakan email user.

View yang ditampilkan: resources/views/auth/reset\_password.blade.php

### ✅ 2. reset(Request $request)

**Fungsi ini menjalankan proses reset password.**

#### a. Validasi Input:

$request->validate([

'token' => 'required',

'email' => 'required|email',

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

]);

Token reset harus ada.

Email harus valid.

Password minimal 6 karakter dan harus dikonfirmasi (password\_confirmation).

#### b. Proses Reset Password:

$status = Password::reset(

$request->only('email', 'password', 'password\_confirmation', 'token'),

function (User $user, $password) {

$user->forceFill([

'password' => Hash::make($password)

])->save();

}

);

Password::reset() adalah fitur bawaan Laravel untuk mengatur logika reset.

Jika berhasil, password user akan di-hash dan disimpan langsung.

forceFill() digunakan untuk langsung mengisi atribut yang dilindungi (fillable) tanpa masalah.

#### c. Redirect:

return $status == Password::PASSWORD\_RESET

? redirect('/login')->with('status', \_\_($status))

: back()->withErrors(['email' => [\_\_($status)]]);

Jika sukses: redirect ke /login dengan pesan sukses.

Jika gagal: kembali ke form dengan error pada email.

### 🧠 Rangkuman Fungsi

| **Fungsi** | **Tujuan** |
| --- | --- |
| showResetForm() | Menampilkan form pengisian password baru |
| reset() | Memvalidasi dan menyimpan password baru user melalui token |
| Password::reset() | Fungsi bawaan Laravel untuk reset password yang aman dan standar |

### 🔒 Keamanan:

**Token**: Mencegah reset password sembarangan.

**Hash**: Password disimpan dalam bentuk aman.

**Validasi**: Mencegah input sembarangan atau manipulasi form.

REPORT:

<?php

namespace App\Http\Controllers;

use App\Models\Item;

use App\Models\Category;

use App\Models\StockIn;

use App\Models\StockOut;

use Illuminate\Http\Request;

use Carbon\Carbon;

use Barryvdh\DomPDF\Facade\Pdf;

use PhpOffice\PhpWord\PhpWord;

use PhpOffice\PhpWord\IOFactory;

use PhpOffice\PhpWord\Style\Font;

class ReportController extends Controller

{

    public function index()

    {

        return view('reports.index');

    }

    public function profit\_pdf()

{

    $startDate = Carbon::now()->startOfMonth()->format('Y-m-d');

    $endDate = Carbon::now()->format('Y-m-d');

    $data = [

        'startDate' => $startDate,

        'endDate' => $endDate,

        'items' => Item::with(['category', 'stockOuts' => function($q) use ($startDate, $endDate) {

            $q->whereBetween('tanggal', [$startDate, $endDate]);

        }])

        ->whereHas('stockOuts', function($q) use ($startDate, $endDate) {

            $q->whereBetween('tanggal', [$startDate, $endDate]);

        })

        ->get()

    ];

    return view('reports.exports.profit\_pdf', $data);

}

    // Laporan Stok Barang

    public function stockReport()

    {

        $items = Item::with('category')

            ->orderBy('stok', 'desc')

            ->get();

        $categories = Category::all();

        return view('reports.stock', compact('items', 'categories'));

    }

    // Laporan Transaksi Masuk/Keluar

    public function transactionReport(Request $request)

    {

        $startDate = $request->input('start\_date', Carbon::now()->subMonth()->format('Y-m-d'));

        $endDate = $request->input('end\_date', Carbon::now()->format('Y-m-d'));

        $stockIns = StockIn::with(['item', 'user'])

            ->whereBetween('tanggal', [$startDate, $endDate])

            ->orderBy('tanggal', 'desc')

            ->get();

        $stockOuts = StockOut::with(['item', 'user'])

            ->whereBetween('tanggal', [$startDate, $endDate])

            ->orderBy('tanggal', 'desc')

            ->get();

        return view('reports.transaction', compact('stockIns', 'stockOuts', 'startDate', 'endDate'));

    }

    // Laporan Profit/Laba

    public function profitReport(Request $request)

    {

        $startDate = $request->input('start\_date', Carbon::now()->subMonth()->format('Y-m-d'));

        $endDate = $request->input('end\_date', Carbon::now()->format('Y-m-d'));

        $items = Item::with(['stockIns', 'stockOuts' => function($q) use ($startDate, $endDate) {

            $q->whereBetween('tanggal', [$startDate, $endDate]);

        }])->get();

        return view('reports.profit', compact('items', 'startDate', 'endDate'));

    }

    // Laporan Kategori Barang

    public function categoryReport()

    {

        $categories = Category::with(['items' => function($query) {

            $query->orderBy('stok', 'desc');

        }])->get();

        return view('reports.category', compact('categories'));

    }

    #EXPORT

    public function exportProfitReport(Request $request, $type)

    {

        $startDate = $request->input('start\_date', Carbon::now()->startOfMonth()->format('Y-m-d'));

        $endDate = $request->input('end\_date', Carbon::now()->format('Y-m-d'));

        $data = [

            'startDate' => $startDate,

            'endDate' => $endDate,

            'items' => Item::with(['category', 'stockOuts' => function($q) use ($startDate, $endDate) {

                $q->whereBetween('tanggal', [$startDate, $endDate]);

            }])

            ->whereHas('stockOuts', function($q) use ($startDate, $endDate) {

                $q->whereBetween('tanggal', [$startDate, $endDate]);

            })

            ->get()

        ];

        $filename = 'Laporan\_Profit\_'.Carbon::parse($startDate)->format('d\_m\_Y').'\_'.Carbon::parse($endDate)->format('d\_m\_Y');

        if ($type === 'pdf') {

            $pdf = Pdf::loadView('reports.exports.profit\_pdf', $data);

            return $pdf->download($filename.'.pdf');

        }

        if ($type === 'word') {

            $phpWord = new PhpWord();

            $section = $phpWord->addSection();

            // Judul Laporan

            $section->addText(

                'LAPORAN PROFIT/LABA',

                ['bold' => true, 'size' => 16],

                ['alignment' => 'center']

            );

            // Periode Laporan

            $section->addText(

                'Periode: ' . Carbon::parse($startDate)->translatedFormat('d F Y') . ' - ' .

                Carbon::parse($endDate)->translatedFormat('d F Y'),

                ['size' => 12],

                ['alignment' => 'center', 'spaceAfter' => 500]

            );

            // Buat Tabel

            $table = $section->addTable([

                'borderSize' => 6,

                'borderColor' => '000000',

                'cellMargin' => 80

            ]);

            // Header Tabel

            $headers = ['No', 'Nama Barang', 'Kategori', 'Harga Beli', 'Terjual', 'Total Penjualan', 'Total Pembelian', 'Laba'];

            $table->addRow();

            foreach ($headers as $header) {

                $table->addCell(2000)->addText($header, ['bold' => true]);

            }

            // Isi Tabel

            foreach ($data['items'] as $key => $item) {

                $sold = $item->stockOuts->sum('jumlah');

                $totalSale = $item->stockOuts->sum(function($trx) {

                    return $trx->harga\_jual\_per\_satuan \* $trx->jumlah;

                });

                $totalPurchase = $item->harga \* $sold;

                $profit = $totalSale - $totalPurchase;

                $table->addRow();

                $table->addCell()->addText($key + 1, null, ['alignment' => 'center']);

                $table->addCell()->addText($item->nama);

                $table->addCell()->addText($item->category->name ?? '-');

                $table->addCell()->addText('Rp ' . number\_format($item->harga, 0, ',', '.'), null, ['alignment' => 'right']);

                $table->addCell()->addText($sold, null, ['alignment' => 'center']);

                $table->addCell()->addText('Rp ' . number\_format($totalSale, 0, ',', '.'), null, ['alignment' => 'right']);

                $table->addCell()->addText('Rp ' . number\_format($totalPurchase, 0, ',', '.'), null, ['alignment' => 'right']);

                // PERBAIKAN: Warna laba (hijau jika profit, merah jika rugi)

                $textRun = $table->addCell()->addTextRun(['alignment' => 'right']);

                if ($profit >= 0) {

                    $textRun->addText('Rp ' . number\_format($profit, 0, ',', '.'), ['color' => '2d862d']);

                } else {

                    $textRun->addText('Rp ' . number\_format($profit, 0, ',', '.'), ['color' => 'cc0000']);

                }

            }

            // Footer (Tanggal Cetak)

            $section->addTextBreak(2);

            $section->addText(

                'Dicetak pada: ' . Carbon::now()->translatedFormat('d F Y H:i'),

                ['italic' => true],

                ['alignment' => 'right']

            );

            // Simpan ke File Sementara

            $tempFile = tempnam(sys\_get\_temp\_dir(), 'PHPWord');

            $writer = IOFactory::createWriter($phpWord, 'Word2007');

            $writer->save($tempFile);

            // Download dan Hapus File Sementara

            return response()->download($tempFile, $filename . '.docx')->deleteFileAfterSend(true);

        }

    }

}

**Penjelasan Kode:** ReportController

Kode di atas adalah **Controller di Laravel** yang mengelola berbagai laporan seperti laporan **stok barang**, **transaksi masuk/keluar**, dan **profit/laba**. Selain itu, controller ini juga menyediakan fitur **eksport laporan** dalam format **PDF** dan **Word**.

### 1. ****Method**** index()

Menampilkan halaman utama laporan.

**View** yang dipanggil: reports.index

public function index()

{

return view('reports.index');

}

### 2. ****Method**** profit\_pdf()

Mengambil data laporan profit berdasarkan tanggal saat ini (dari awal bulan hingga tanggal hari ini).

Data yang diambil meliputi **item** dan **stockOuts** (stok keluar) dalam rentang tanggal tersebut.

**View** yang dipanggil: reports.exports.profit\_pdf

public function profit\_pdf()

{

$startDate = Carbon::now()->startOfMonth()->format('Y-m-d');

$endDate = Carbon::now()->format('Y-m-d');

$data = [

'startDate' => $startDate,

'endDate' => $endDate,

'items' => Item::with(['category', 'stockOuts' => function($q) use ($startDate, $endDate) {

$q->whereBetween('tanggal', [$startDate, $endDate]);

}])

->whereHas('stockOuts', function($q) use ($startDate, $endDate) {

$q->whereBetween('tanggal', [$startDate, $endDate]);

})

->get()

];

return view('reports.exports.profit\_pdf', $data);

}

### 3. ****Method**** stockReport()

Menampilkan laporan stok barang yang diurutkan berdasarkan jumlah stok dari yang terbesar.

Mengambil data dari **Item** dan **Category**.

**View** yang dipanggil: reports.stock

public function stockReport()

{

$items = Item::with('category')

->orderBy('stok', 'desc')

->get();

$categories = Category::all();

return view('reports.stock', compact('items', 'categories'));

}

### 4. ****Method**** transactionReport(Request $request)

Menampilkan laporan transaksi masuk (stock in) dan keluar (stock out) dalam rentang tanggal yang ditentukan oleh pengguna.

**View** yang dipanggil: reports.transaction

public function transactionReport(Request $request)

{

$startDate = $request->input('start\_date', Carbon::now()->subMonth()->format('Y-m-d'));

$endDate = $request->input('end\_date', Carbon::now()->format('Y-m-d'));

$stockIns = StockIn::with(['item', 'user'])

->whereBetween('tanggal', [$startDate, $endDate])

->orderBy('tanggal', 'desc')

->get();

$stockOuts = StockOut::with(['item', 'user'])

->whereBetween('tanggal', [$startDate, $endDate])

->orderBy('tanggal', 'desc')

->get();

return view('reports.transaction', compact('stockIns', 'stockOuts', 'startDate', 'endDate'));

}

### 5. ****Method**** profitReport(Request $request)

Menampilkan laporan profit/laba berdasarkan transaksi **stock in** dan **stock out** dalam rentang tanggal yang ditentukan oleh pengguna.

**View** yang dipanggil: reports.profit

public function profitReport(Request $request)

{

$startDate = $request->input('start\_date', Carbon::now()->subMonth()->format('Y-m-d'));

$endDate = $request->input('end\_date', Carbon::now()->format('Y-m-d'));

$items = Item::with(['stockIns', 'stockOuts' => function($q) use ($startDate, $endDate) {

$q->whereBetween('tanggal', [$startDate, $endDate]);

}])->get();

return view('reports.profit', compact('items', 'startDate', 'endDate'));

}

### 6. ****Method**** categoryReport()

Menampilkan laporan kategori barang dengan item yang diurutkan berdasarkan jumlah stok terbesar.

**View** yang dipanggil: reports.category

public function categoryReport()

{

$categories = Category::with(['items' => function($query) {

$query->orderBy('stok', 'desc');

}])->get();

return view('reports.category', compact('categories'));

}

### 7. ****Method**** exportProfitReport(Request $request, $type)

Menghasilkan laporan profit dalam format **PDF** atau **Word** berdasarkan rentang tanggal yang ditentukan oleh pengguna.

Tipe yang dapat dipilih: pdf atau word.

Untuk **PDF**, menggunakan **DomPDF**.

Untuk **Word**, menggunakan **PhpWord**.

#### PDF:

if ($type === 'pdf') {

$pdf = Pdf::loadView('reports.exports.profit\_pdf', $data);

return $pdf->download($filename.'.pdf');

}

#### Word:

if ($type === 'word') {

// Membuat laporan dalam format Word menggunakan PhpWord

}

**Proses Word**:

Membuat laporan dalam format Word dengan **tabel** yang berisi informasi tentang item, kategori, harga beli, terjual, total penjualan, total pembelian, dan laba.

Laba yang negatif diberi warna merah, sedangkan laba positif diberi warna hijau.

**Download** file Word setelah selesai.

### Penjelasan Tambahan:

**Penggunaan** Carbon: Untuk menangani tanggal secara mudah, seperti mengambil tanggal hari ini atau tanggal awal bulan.

**Model yang digunakan**:

Item, Category, StockIn, StockOut adalah model yang merepresentasikan entitas dalam sistem.

StockIn dan StockOut digunakan untuk transaksi barang masuk dan keluar.

### Kesimpulan:

Controller ini mengelola berbagai laporan untuk sistem inventaris, menyediakan tampilan laporan berdasarkan kriteria tertentu (seperti tanggal), serta memungkinkan ekspor laporan dalam format **PDF** dan **Word**.

VIEW:

<!DOCTYPE html>

<html lang="en" data-bs-theme="light">

<head>

    <meta charset="UTF-8">

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

    <title>Login</title>

    <link rel="stylesheet" href="{{ asset('/css/login.css') }}">

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Font Awesome for icons -->

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

</head>

<body>

<div class="container">

    <div class="row justify-content-center align-items-center vh-100">

        <div class="col-md-4 col-sm-8 col-10">

            <div class="card shadow-lg p-4">

                <div class="text-center mb-4">

                    <i class="fas fa-user-circle fa-4x text-primary mb-3"></i>

                    <h3 class="fw-bold">Welcome Back</h3>

                    <p class="text-muted">Please login to continue</p>

                </div>

                @if ($errors->any())

                    <div class="alert alert-danger alert-dismissible fade show">

                        {{ $errors->first() }}

                        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>

                    </div>

                @endif

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

                    @csrf

                    <div class="mb-3">

                        <label for="email" class="form-label">Email address</label>

                        <div class="input-group">

                            <span class="input-group-text"><i class="fas fa-envelope"></i></span>

                            <input type="email" class="form-control" id="email" name="email" required autofocus placeholder="Enter your email">

                        </div>

                    </div>

                    <div class="mb-3">

                        <label for="password" class="form-label">Password</label>

                        <div class="input-group">

                            <span class="input-group-text"><i class="fas fa-lock"></i></span>

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

                        </div>

                    </div>

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

                        <input type="checkbox" class="form-check-input" id="remember" name="remember">

                        <label class="form-check-label" for="remember">Remember me</label>

                    </div>

                    <div class="d-grid gap-2 mb-3">

                        <button type="submit" class="btn btn-primary py-2">

                            <i class="fas fa-sign-in-alt me-2"></i> Login

                        </button>

                    </div>

                </form>

                <div class="text-center">

    <a href="{{ route('password.request') }}" class="text-decoration-none">Forgot password?</a><br>

    <a href="{{ route('register') }}" class="text-decoration-none">Belum punya akun? Register</a>

</div>

            </div>

        </div>

    </div>

</div>

<!-- Theme toggle button -->

<div class="theme-toggle" onclick="toggleTheme()" title="Toggle dark mode">

    <i class="fas fa-moon"></i>

</div>

<!-- Bootstrap JS -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<script>

    function toggleTheme() {

        const html = document.querySelector('html');

        const theme = html.dataset.bsTheme === 'light' ? 'dark' : 'light';

        html.dataset.bsTheme = theme;

        // Update icon

        const icon = document.querySelector('.theme-toggle i');

        icon.className = theme === 'light' ? 'fas fa-moon' : 'fas fa-sun';

        // Update title

        document.querySelector('.theme-toggle').title = theme === 'light'

            ? 'Toggle dark mode'

            : 'Toggle light mode';

    }

    // Initialize theme icon

    document.addEventListener('DOMContentLoaded', function() {

        const html = document.querySelector('html');

        const icon = document.querySelector('.theme-toggle i');

        if (html.dataset.bsTheme === 'dark') {

            icon.className = 'fas fa-sun';

            document.querySelector('.theme-toggle').title = 'Toggle light mode';

        }

    });

</script>

</body>

</html>

Ini adalah sebuah dokumen HTML yang membuat halaman login dengan desain modern dan responsif. Berikut adalah penjelasan tentang setiap bagian dan fungsionalitasnya:

### 1. ****Struktur Dokumen:****

<!DOCTYPE html>: Menyatakan bahwa ini adalah dokumen HTML5.

Elemen HTML utama menggunakan atribut lang="en" untuk bahasa Inggris dan data-bs-theme="light" yang mengatur tema awal halaman ke mode terang (light mode).

### 2. ****Bagian Head:****

**Meta Tags**:

charset="UTF-8" menentukan encoding karakter yang digunakan dalam dokumen.

Tag viewport memastikan halaman responsif dan skala dengan baik di berbagai perangkat.

**Title**: Mengatur judul halaman menjadi "Login."

**Stylesheets**:

File CSS eksternal login.css disertakan untuk gaya kustom.

**Bootstrap CSS**: Dikirim dari CDN untuk layout dan desain responsif.

**Font Awesome**: Digunakan untuk ikon, seperti ikon pengguna dan kunci.

### 3. ****Bagian Body:****

**Container Layout**:

Konten utama dibungkus dalam elemen div dengan kelas container, yang memastikan formulir login berada di tengah layar secara vertikal dan horizontal menggunakan kelas grid Bootstrap (row justify-content-center align-items-center vh-100).

Sebuah komponen card digunakan untuk menampung formulir login dengan padding (p-4) dan efek bayangan.

**Konten Kartu**:

**Ikon**: Ikon lingkaran pengguna (fa-user-circle) ditampilkan di bagian atas, diikuti dengan judul "Welcome Back" dan pesan untuk login.

**Tampilan Error**:

Jika ada kesalahan (seperti kredensial login yang salah), pesan error akan ditampilkan di dalam kotak peringatan berwarna merah (alert-danger).

**Formulir Login**:

**Input Email**: Kolom untuk alamat email dengan ikon amplop dari Font Awesome.

**Input Password**: Kolom untuk password dengan ikon kunci.

**Checkbox Remember Me**: Opsi untuk mengingat login pengguna.

**Tombol Kirim**: Tombol besar untuk mengirim formulir login, dengan ikon masuk di samping teks.

### 4. ****Tautan:****

Di bawah formulir, terdapat dua tautan:

**Forgot password**: Mengarahkan ke halaman pemulihan kata sandi.

**Register**: Tautan menuju halaman pendaftaran untuk pengguna baru.

### 5. ****Tombol Pengubah Tema:****

Sebuah tombol kecil dengan ikon disediakan untuk beralih antara mode terang dan gelap.

Tombol ini akan berganti antara ikon bulan (fa-moon) untuk mode terang dan ikon matahari (fa-sun) untuk mode gelap.

Fungsi pengubah tema (toggleTheme()) mengubah tema halaman secara dinamis dengan memperbarui atribut data-bs-theme pada elemen <html>.

Fungsi toggleTheme() juga mengubah ikon dan judul tombol pengubah tema berdasarkan tema yang sedang digunakan.

### 6. ****JavaScript:****

Fungsi pengubah tema (toggleTheme) bekerja dengan memeriksa tema saat ini (data-bs-theme) dan mengganti antara 'light' dan 'dark' saat tombol diklik.

Event listener DOMContentLoaded memastikan ikon tema diatur dengan benar saat halaman dimuat berdasarkan tema awal.

### 7. ****Bootstrap & JavaScript:****

Bundle JavaScript dari Bootstrap disertakan untuk komponen dinamis seperti menu yang dapat dilipat, modal, atau tooltip yang mungkin dibutuhkan.

JavaScript di bagian bawah halaman menangani perilaku pengubahan tema.

### Kesimpulan:

Halaman login ini menggunakan kombinasi Bootstrap untuk layout dan responsivitas, Font Awesome untuk ikon, dan CSS kustom (login.css) untuk penataan visual. Halaman ini memungkinkan pengguna untuk login dengan email dan password, menampilkan pesan kesalahan jika login tidak valid, serta menyediakan fungsionalitas untuk mengganti tema halaman antara mode terang dan gelap.

REGISTER:

<!DOCTYPE html>

<html lang="en" data-bs-theme="light">

<head>

    <meta charset="UTF-8">

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

    <title>Login</title>

    <link rel="stylesheet" href="{{ asset('/css/login.css') }}">

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Font Awesome for icons -->

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

</head>

<body>

<div class="container">

    <div class="row justify-content-center align-items-center vh-100">

        <div class="col-md-4 col-sm-8 col-10">

            <div class="card shadow-lg p-4">

                <div class="text-center mb-4">

                    <i class="fas fa-user-circle fa-4x text-primary mb-3"></i>

                    <h3 class="fw-bold">Welcome Back</h3>

                    <p class="text-muted">Please login to continue</p>

                </div>

                @if ($errors->any())

                    <div class="alert alert-danger alert-dismissible fade show">

                        {{ $errors->first() }}

                        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>

                    </div>

                @endif

                <!-- Sama struktur seperti login, ganti form dan judul -->

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

    @csrf

    <!-- Name -->

    <div class="mb-3">

        <label>Name</label>

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

    </div>

    <!-- Username (optional) -->

    <div class="mb-3">

        <label>Username</label>

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

    </div>

    <!-- Email -->

    <div class="mb-3">

        <label>Email</label>

        <input type="email" name="email" class="form-control" required>

    </div>

    <!-- Password -->

    <div class="mb-3">

        <label>Password</label>

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

    </div>

    <!-- Confirm Password -->

    <div class="mb-3">

        <label>Confirm Password</label>

        <input type="password" name="password\_confirmation" class="form-control" required>

    </div>

    <button class="btn btn-primary w-100">Register</button>

</form>

<p class="mt-3 text-center">

    Sudah punya akun? <a href="{{ route('login') }}">Login di sini</a>

</p>

            </div>

        </div>

    </div>

</div>

<!-- Theme toggle button -->

<div class="theme-toggle" onclick="toggleTheme()" title="Toggle dark mode">

    <i class="fas fa-moon"></i>

</div>

<!-- Bootstrap JS -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<script>

    function toggleTheme() {

        const html = document.querySelector('html');

        const theme = html.dataset.bsTheme === 'light' ? 'dark' : 'light';

        html.dataset.bsTheme = theme;

        // Update icon

        const icon = document.querySelector('.theme-toggle i');

        icon.className = theme === 'light' ? 'fas fa-moon' : 'fas fa-sun';

        // Update title

        document.querySelector('.theme-toggle').title = theme === 'light'

            ? 'Toggle dark mode'

            : 'Toggle light mode';

    }

    // Initialize theme icon

    document.addEventListener('DOMContentLoaded', function() {

        const html = document.querySelector('html');

        const icon = document.querySelector('.theme-toggle i');

        if (html.dataset.bsTheme === 'dark') {

            icon.className = 'fas fa-sun';

            document.querySelector('.theme-toggle').title = 'Toggle light mode';

        }

    });

</script>

</body>

</html>

APP:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>{{ config('app.name', 'TokoKu') }}</title>

    <!-- Bootstrap 5 CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- DataTables & Font Awesome -->

    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">

    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" rel="stylesheet">

    @stack('styles')

</head>

<body class="d-flex flex-column">

    <!-- Topbar Fixed -->

    @include('layouts.topbar')

    <div class="d-flex flex-grow-1">

        <!-- Sidebar Fixed -->

        @include('layouts.sidebar')

        <!-- Konten Utama -->

        <main class="flex-grow-1 overflow-auto" style="margin-top: 56px; margin-left: 250px;">

            <div class="container-fluid p-4">

                @yield('content')

            </div>

        </main>

    </div>

    <!-- Scripts -->

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>

    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

    @if(session('success'))

    <script>

    Swal.fire({

            icon: 'success',

            title: 'Sukses',

            text: '{{ session('success') }}',

            timer: 1500,

            showConfirmButton: false

        });

    </script>

    @endif

    @stack('scripts')

</body>

</html>

Ini adalah sebuah dokumen HTML yang mendefinisikan struktur dasar halaman web dengan beberapa fitur tambahan menggunakan Bootstrap, DataTables, Font Awesome, dan beberapa plugin JavaScript lainnya. Berikut adalah penjelasan rinci tentang setiap bagian dan fungsionalitasnya:

### 1. ****Bagian**** <head>****:****

**Meta Tags**:

charset="UTF-8": Menentukan encoding karakter untuk halaman ini agar bisa menampilkan berbagai karakter internasional.

viewport: Memastikan halaman responsif di perangkat mobile dengan pengaturan lebar viewport dan skala yang sesuai.

**Judul Halaman (**<title>**)**:

{{ config('app.name', 'TokoKu') }}: Dinamis, mengambil nama aplikasi dari konfigurasi Laravel (config('app.name')) atau jika tidak ada, menggunakan nilai default 'TokoKu'.

**Stylesheets**:

**Bootstrap 5**: Menyertakan CSS dari Bootstrap versi 5.3.2 untuk tata letak dan komponen responsif.

**DataTables CSS**: Menggunakan CSS dari DataTables (versi 1.13.6) untuk menata tabel interaktif yang menggunakan fitur pencarian, pengurutan, dan paginasi.

**Font Awesome**: Menyediakan akses ke ikon-ikon dari Font Awesome (versi 6.4.2), yang sering digunakan untuk ikon seperti tombol, menu, dan lainnya.

**@stack('styles')**: Menyertakan stylesheet tambahan jika ada bagian yang membutuhkan penyesuaian lebih lanjut dari tampilan.

### 2. ****Bagian**** <body>****:****

body **dengan kelas** d-flex flex-column: Ini memungkinkan halaman menggunakan Flexbox untuk penataan elemen secara vertikal. d-flex memastikan elemen anak akan ditata menggunakan Flexbox, dan flex-column menata elemen-elemen anak secara vertikal.

**Topbar**:

@include('layouts.topbar'): Mengimpor bagian topbar dari file layout Laravel. Biasanya ini berisi elemen-elemen navigasi seperti logo, menu pengguna, atau notifikasi.

**Sidebar**:

@include('layouts.sidebar'): Mengimpor bagian sidebar dari layout. Sidebar ini mungkin berisi menu navigasi untuk akses ke berbagai bagian aplikasi.

**Konten Utama**:

Konten utama dari halaman web terletak di dalam <main>. Di dalamnya ada kelas container-fluid untuk memastikan konten mengisi seluruh lebar layar. Ada juga kelas p-4 yang memberi padding di sekitar konten utama.

@yield('content'): Menandakan tempat di mana konten spesifik halaman akan dimasukkan. Setiap halaman turunan dari layout ini dapat menyertakan konten yang berbeda dengan cara mendefinisikan bagian content.

### 3. ****Bagian**** <script>****:****

**Script External**:

**jQuery**: Library JavaScript yang sangat populer untuk manipulasi DOM dan Ajax.

**Bootstrap 5**: Menyertakan file JavaScript dari Bootstrap untuk komponen interaktif seperti modal, dropdown, dan tombol navigasi.

**DataTables**: Menyediakan skrip untuk menambahkan fungsionalitas interaktif pada tabel, seperti pencarian, penyaringan, dan pengurutan.

**Chart.js**: Digunakan untuk membuat grafik dan chart interaktif.

**SweetAlert2**: Digunakan untuk menampilkan pesan notifikasi yang lebih menarik (seperti alert dengan tampilan lebih baik).

**Pesan Sukses dengan SweetAlert2**:

Jika ada pesan sukses yang disimpan dalam sesi (session('success')), maka akan menampilkan notifikasi menggunakan SweetAlert2 yang menunjukkan pesan sukses tersebut. Pesan ini akan muncul selama 1,5 detik dan tidak memerlukan konfirmasi dari pengguna.

**Script SweetAlert2**:

Swal.fire({

icon: 'success',

title: 'Sukses',

text: '{{ session('success') }}',

timer: 1500,

showConfirmButton: false

});

Ini akan menampilkan notifikasi "Sukses" dengan teks yang didapat dari sesi Laravel.

**@stack('scripts')**: Menyertakan skrip tambahan yang dapat ditambahkan pada halaman tertentu.

### Kesimpulan:

Halaman ini adalah layout dasar yang dirancang menggunakan Laravel. Halaman ini menyertakan beberapa fitur utama:

**Topbar** dan **Sidebar**: Bagian navigasi yang dimasukkan menggunakan @include().

**Konten Utama**: Tempat untuk menyisipkan konten spesifik halaman dengan @yield('content').

**Integrasi dengan Bootstrap**: Untuk tata letak responsif dan komponen interaktif.

**DataTables**: Untuk menampilkan dan mengelola tabel interaktif.

**SweetAlert2**: Untuk menampilkan pesan sukses secara interaktif.

**Dukungan untuk penambahan CSS dan JS**: Menggunakan @stack() untuk memungkinkan penambahan gaya dan skrip khusus per halaman.

Ini adalah struktur dasar yang fleksibel dan dapat digunakan untuk membuat aplikasi web dinamis dengan fitur-fitur tambahan seperti navigasi, notifikasi, dan tabel interaktif.

SIDEBAR:

<!-- resources/views/layouts/sidebar.blade.php -->

<nav class="d-flex flex-column bg-primary text-white position-fixed start-0 top-0 h-100" style="width: 250px; z-index: 1030; margin-top: 60px;">

    <div class="p-3">

        <a class="d-flex align-items-center text-white text-decoration-none mb-4">

            <i class="fas fa-store fs-4 me-2"></i>

            <span class="fs-4">Inventaris Toko</span>

        </a>

        <ul class="nav nav-pills flex-column">

            <!-- Dashboard -->

            <li class="nav-item">

                <a class="nav-link text-dark {{ request()->routeIs('dashboard') ? 'active bg-light' : '' }}"

                   href="{{ route('dashboard') }}">

                    <i class="fas fa-tachometer-alt me-2"></i> Dashboard

                </a>

            </li>

            <!-- Inventory -->

            <li>

                <a class="nav-link text-dark {{ request()->routeIs('items.\*') ? 'active bg-light' : '' }}"

                   href="{{ route('items.index') }}">

                    <i class="fas fa-boxes me-2"></i>

                    Data Barang

                </a>

            </li>

            <li>

                <a class="nav-link text-dark {{ request()->routeIs('stockin.\*') ? 'active bg-light' : '' }}"

                   href="{{ route('stockin.index') }}">

                    <i class="fas fa-arrow-circle-down me-2"></i>

                    Barang Masuk

                </a>

            </li>

            <li>

                <a class="nav-link text-dark {{ request()->routeIs('stockout.\*') ? 'active bg-light' : '' }}"

                   href="{{ route('stockout.index') }}">

                    <i class="fas fa-arrow-circle-up me-2"></i>

                    Barang Keluar

                </a>

            </li>

            <!-- Reports (Super Admin Only) -->

            @can('access-reports')

            <li class="nav-item dropdown">

                <a class="nav-link text-dark dropdown-toggle {{ request()->routeIs('reports.\*') ? 'active bg-light' : '' }}"

                   href="#" id="reportsDropdown" role="button" data-bs-toggle="dropdown">

                    <i class="fas fa-file-alt me-2"></i>

                    Laporan

                </a>

                <ul class="dropdown-menu bg-primary w-100">

                    <li>

                        <a class="dropdown-item text-dark {{ request()->routeIs('reports.index') ? 'active bg-light' : '' }}"

                           href="{{ route('reports.index') }}">

                            Overview

                        </a>

                    </li>

                    <li>

                        <a class="dropdown-item text-dark {{ request()->routeIs('reports.stock') ? 'active bg-light' : '' }}"

                           href="{{ route('reports.stock') }}">

                            Stok

                        </a>

                    </li>

                    <li>

                        <a class="dropdown-item text-dark {{ request()->routeIs('reports.transaction') ? 'active bg-light' : '' }}"

                           href="{{ route('reports.transaction') }}">

                            Transaksi

                        </a>

                    </li>

                    <li>

                        <a class="dropdown-item text-dark {{ request()->routeIs('reports.profit') ? 'active bg-light' : '' }}"

                           href="{{ route('reports.profit') }}">

                            Profit

                        </a>

                    </li>

                    <li>

                        <a class="dropdown-item text-dark {{ request()->routeIs('reports.category') ? 'active bg-light' : '' }}"

                           href="{{ route('reports.category') }}">

                            Kategori

                        </a>

                    </li>

                </ul>

            </li>

            @endcan

            <!-- User Management (Super Admin Only) -->

            @can('manage-users')

            <li>

                <a class="nav-link text-dark {{ request()->routeIs('users.\*') ? 'active bg-light' : '' }}"

                   href="{{ route('users.index') }}">

                    <i class="fas fa-users me-2"></i>

                    Manajemen User

                </a>

            </li>

            @endcan

        </ul>

        <!-- Current User Info -->

        <div class="mt-auto pt-3 border-top">

            <div class="d-flex align-items-center">

                <div class="flex-shrink-0">

                    <i class="fas fa-user-circle fs-3"></i>

                </div>

                <div class="flex-grow-1 ms-3">

                    <div class="fw-bold">{{ auth()->user()->name }}</div>

                    <small class="text-white-50 text-capitalize">{{ auth()->user()->role }}</small>

                </div>

            </div>

        </div>

    </div>

</nav>

Berikut penjelasan untuk kode yang terdapat dalam file sidebar.blade.php:

### 1. ****Struktur Sidebar:****

**Tag** <nav>: Digunakan untuk membuat navigasi sidebar. Sidebar ini menggunakan kelas Flexbox (d-flex flex-column) untuk mengatur elemen-elemen secara vertikal. Sidebar ini juga memiliki warna latar belakang biru (bg-primary), teks berwarna putih (text-white), dan diposisikan secara tetap di sisi kiri halaman dengan menggunakan position-fixed, agar tetap berada di tempat meskipun halaman digulir.

style="width: 250px; z-index: 1030; margin-top: 60px;": Sidebar memiliki lebar tetap 250px, z-index tinggi untuk menempatkannya di atas elemen lain, dan margin-top 60px untuk memberi ruang di atasnya.

### 2. ****Header Sidebar:****

**Logo dan Nama**:

Terdapat sebuah elemen a yang berfungsi sebagai logo dan nama aplikasi atau toko. Di dalamnya terdapat ikon fas fa-store (ikon toko) dan teks "Inventaris Toko" yang ditampilkan dengan ukuran font fs-4.

Menggunakan kelas text-decoration-none untuk menghilangkan garis bawah pada teks dan text-white untuk warna teks putih.

### 3. ****Menu Navigasi:****

Menu navigasi sidebar diatur dalam sebuah <ul> dengan kelas nav dan nav-pills, yang menampilkan menu dalam bentuk daftar vertikal. Setiap item menu dikategorikan dalam elemen <li>, dan menggunakan kelas nav-item dan nav-link untuk mengatur tampilannya.

**Menu Dashboard**:

**Link ke Dashboard**: Menu pertama adalah link ke dashboard dengan ikon fas fa-tachometer-alt.

Kondisi {{ request()->routeIs('dashboard') ? 'active bg-light' : '' }} menambahkan kelas active bg-light pada item ini jika rute saat ini adalah dashboard, menandakan bahwa menu ini sedang aktif.

**Menu Data Barang**:

Link menuju data barang dengan ikon fas fa-boxes.

Kondisi {{ request()->routeIs('items.\*') ? 'active bg-light' : '' }} menambahkan kelas aktif jika halaman yang sedang dibuka terkait dengan data barang.

**Menu Barang Masuk dan Barang Keluar**:

Masing-masing menu ini berfungsi untuk mengarahkan pengguna ke halaman manajemen barang masuk dan barang keluar, dengan ikon yang sesuai.

### 4. ****Laporan (Hanya untuk Super Admin):****

**Menu Laporan**:

Hanya dapat diakses oleh pengguna yang memiliki izin access-reports. Dapat ditandai dengan kondisi @can('access-reports') untuk memeriksa apakah pengguna memiliki izin tersebut.

Terdapat menu dropdown dengan beberapa pilihan laporan:

**Overview**, **Stok**, **Transaksi**, **Profit**, dan **Kategori**.

Setiap submenu ini juga memiliki kelas aktif yang ditentukan oleh rute yang sedang aktif, menggunakan kondisi {{ request()->routeIs('reports.\*') ? 'active bg-light' : '' }}.

### 5. ****Manajemen Pengguna (Hanya untuk Super Admin):****

**Menu Manajemen User**:

Tersedia hanya untuk pengguna dengan izin manage-users. Kondisi @can('manage-users') memeriksa apakah pengguna memiliki izin untuk mengakses bagian ini.

Memungkinkan manajemen pengguna dengan tampilan ikon fas fa-users.

### 6. ****Informasi Pengguna Saat Ini****:

Di bagian bawah sidebar, ada informasi tentang pengguna yang sedang login:

**Nama Pengguna**: Ditampilkan menggunakan auth()->user()->name.

**Peran Pengguna**: Ditampilkan menggunakan auth()->user()->role, yang menunjukkan peran pengguna, misalnya "Admin" atau "Super Admin".

**Ikon Pengguna**: Menggunakan ikon fas fa-user-circle untuk menunjukkan gambar profil atau status pengguna.

### 7. ****Fitur dan Kustomisasi:****

**Kelas Dinamis**: Pada setiap menu, kondisi dinamis ditambahkan menggunakan request()->routeIs() untuk memastikan item menu yang sesuai dengan rute saat ini mendapatkan kelas active bg-light, yang memberi indikasi bahwa menu tersebut aktif.

**Kontrol Akses dengan** @can: Menambahkan kontrol akses untuk memastikan hanya pengguna dengan izin tertentu yang bisa melihat dan mengakses menu-menu seperti Laporan dan Manajemen Pengguna.

### Kesimpulan:

Sidebar ini adalah bagian penting dari antarmuka pengguna untuk aplikasi manajemen inventaris toko.

Menggunakan **kontrol akses berbasis peran** dengan direktif @can untuk menampilkan menu tertentu hanya kepada pengguna yang berwenang (Super Admin).

Menyediakan **navigasi dinamis** dengan kelas aktif yang berubah sesuai dengan halaman yang sedang diakses.

**Menampilkan informasi pengguna** di bagian bawah untuk memberikan konfirmasi tentang siapa yang sedang login.

Ini memungkinkan sidebar menjadi alat navigasi yang interaktif, aman, dan mudah disesuaikan dengan kebutuhan aplikasi.

TOPBAR:

<!-- Topbar -->

<nav class="navbar navbar-expand navbar-light shadow-sm fixed-top" style="

    z-index: 1040;

    background: url('{{ asset('images/bgtopbar.png') }}');

    background-size: 200% auto;

    background-position: 20% center;

    background-repeat: no-repeat;

    height: 60px;

">

    <div class="container-fluid">

        <!-- Logo Toko -->

        <a class="navbar-brand me-3" href="#" style="padding-left: 40px;">

            <img src="{{ asset('images/iconTokoKu.png') }}" alt="Logo TokoKu" style="height: 70px;">

        </a>

        <!-- Tombol Toggle Sidebar -->

        <button class="btn btn-link d-lg-none" data-bs-toggle="offcanvas" data-bs-target="#sidebarMobile">

            <i class="fas fa-bars"></i>

        </button>

        <!-- Menu User -->

        <div class="navbar-nav ms-auto">

            <div class="nav-item dropdown">

                <a class="nav-link dropdown-toggle text-white" href="#" data-bs-toggle="dropdown" style="text-shadow: 1px 1px 2px rgba(0,0,0,0.5);">

                    <i class="fas fa-user-circle me-1"></i>

                    {{ Auth::user()->name ?? 'Guest' }}

                </a>

                <ul class="dropdown-menu dropdown-menu-end">

                    <li><a class="dropdown-item" href="#"><i class="fas fa-user me-2"></i> Profil</a></li>

                    <li><hr class="dropdown-divider"></li>

                    <li>

                        <form action="{{ route('logout') }}" method="POST">

                            @csrf

                            <button type="submit" class="dropdown-item">

                                <i class="fas fa-sign-out-alt me-2"></i> Logout

                            </button>

                        </form>

                    </li>

                </ul>

            </div>

        </div>

    </div>

</nav>

Berikut adalah penjelasan mengenai kode yang terdapat pada bagian **Topbar** (navbar) dari tampilan aplikasi:

### 1. ****Struktur Navbar:****

**Tag** <nav>: Digunakan untuk membungkus elemen navbar. Navbar ini menggunakan kelas navbar, navbar-expand, dan navbar-light dari Bootstrap untuk membuat navbar yang responsif, terang, dan bisa mengembang.

fixed-top: Kelas ini membuat navbar tetap berada di bagian atas halaman saat digulir.

shadow-sm: Menambahkan efek bayangan ringan pada navbar agar tampilannya lebih menarik.

z-index: 1040: Memastikan navbar selalu berada di atas elemen lain dengan memberi nilai z-index yang lebih tinggi.

background: Diatur menggunakan gambar latar belakang yang diambil dari file bgtopbar.png pada folder images. Gambar ini diatur dengan ukuran **200%** dari ukuran asli dan posisinya diatur agar lebih fokus di bagian kiri tengah.

### 2. ****Logo Toko:****

<a class="navbar-brand me-3">: Elemen ini berfungsi sebagai logo toko yang bisa diklik. Kelas navbar-brand adalah kelas Bootstrap untuk menandakan logo atau brand di navbar.

<img src="{{ asset('images/iconTokoKu.png') }}" alt="Logo TokoKu">: Menampilkan logo toko yang diambil dari file iconTokoKu.png. Gambar ini diatur tingginya menjadi **70px**.

### 3. ****Tombol Toggle Sidebar (Untuk Tampilan Mobile):****

<button class="btn btn-link d-lg-none">: Tombol ini muncul hanya pada tampilan layar kecil (menggunakan kelas d-lg-none yang artinya tombol hanya akan ditampilkan pada perangkat dengan ukuran layar kecil). Tombol ini berfungsi untuk menampilkan atau menyembunyikan sidebar ketika ikon **hamburger** (tiga garis horizontal) diklik.

data-bs-toggle="offcanvas" data-bs-target="#sidebarMobile": Menggunakan Bootstrap 5 untuk menampilkan sidebar di perangkat mobile dengan efek offcanvas. Ketika tombol ini diklik, sidebar dengan ID sidebarMobile akan ditampilkan.

### 4. ****Menu Dropdown Pengguna:****

<div class="navbar-nav ms-auto">: Bagian ini membuat menu user berada di sebelah kanan navbar dengan menggunakan kelas ms-auto yang mengatur margin kiri otomatis (untuk mendorong elemen ke kanan).

<div class="nav-item dropdown">: Menyediakan menu dropdown untuk menampilkan informasi pengguna yang sedang login.

<a class="nav-link dropdown-toggle text-white">: Tautan untuk membuka dropdown, yang menampilkan nama pengguna saat ini. Jika pengguna tidak login, teks "Guest" akan ditampilkan.

{{ Auth::user()->name ?? 'Guest' }}: Menampilkan nama pengguna yang sedang login dengan menggunakan **Laravel's authentication**. Jika pengguna tidak login, maka akan menampilkan teks "Guest".

<i class="fas fa-user-circle me-1"></i>: Ikon **user-circle** untuk menunjukkan bahwa ini adalah menu pengguna.

### 5. ****Dropdown Menu:****

**Menu Profil**: Di dalam dropdown, terdapat opsi **Profil**, yang dapat digunakan untuk mengakses halaman profil pengguna (meskipun URL-nya tidak diatur di sini, biasanya mengarah ke halaman profil pengguna).

**Tombol Logout**: Menyediakan tombol **Logout** yang memungkinkan pengguna keluar dari aplikasi.

Tombol ini berisi formulir **POST** dengan rute route('logout') yang menangani proses logout.

@csrf: Menambahkan token CSRF untuk memastikan keamanan saat pengiriman formulir logout.

<button type="submit" class="dropdown-item">: Tombol untuk logout dengan ikon **sign-out-alt**.

### 6. ****Penggunaan Kelas Bootstrap dan Ikon Font Awesome:****

**Kelas Bootstrap** seperti navbar, navbar-expand, navbar-light, dan dropdown-menu digunakan untuk mempermudah pembuatan desain responsif dan interaktif.

**Ikon Font Awesome** seperti fas fa-user-circle dan fas fa-sign-out-alt digunakan untuk mempercantik tampilan dengan ikon yang relevan (ikon profil dan logout).

### Kesimpulan:

**Navbar ini dirancang responsif** dan menyesuaikan tampilannya untuk perangkat desktop dan mobile.

Menggunakan gambar latar belakang dan logo untuk memperindah tampilan navbar.

**Menu pengguna** memungkinkan pengguna untuk melihat nama mereka dan menyediakan opsi untuk melihat profil atau keluar dari aplikasi.

**Tombol hamburger** yang hanya muncul pada layar kecil memungkinkan pengguna untuk mengakses sidebar di perangkat mobile.

**Penggunaan dropdown menu** memberikan opsi tambahan, seperti **profil** dan **logout**.

Ini memberikan navigasi yang jelas dan aksesibilitas bagi pengguna baik di desktop maupun perangkat mobile.

DATA\_BARANG.INDEX:

@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header">

        Tambah Data Barang

    </div>

    <div class="card-body">

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

            @csrf

            <div class="mb-3">

                <label>Nama Barang</label>

                <input type="text" name="nama" class="form-control" required>

                @error('nama')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Kategori</label>

                <select name="category\_id" class="form-control" required>

                    <option value="">-- Pilih Kategori --</option>

                    @foreach($categories as $category)

                    <option value="{{ $category->id }}">{{ $category->name }}</option>

                    @endforeach

                </select>

                @error('category\_id')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Harga</label>

                <input type="number" name="harga" class="form-control" min="0" required>

                @error('harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Harga</label>

                <input type="text" name="satuan\_harga" class="form-control" required>

                @error('satuan\_harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Stok</label>

                <input type="number" name="stok" class="form-control" min="0" required>

                @error('stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Stok</label>

                <input type="text" name="satuan\_stok" class="form-control" required>

                @error('satuan\_stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Keterangan</label>

                <textarea name="keterangan" class="form-control" rows="3"></textarea>

                @error('keterangan')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="d-flex gap-2">

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

                <a href="{{ route('items.index') }}" class="btn btn-secondary">Batal</a>

            </div>

        </form>

    </div>

</div>

@endsection

Berikut adalah penjelasan mengenai kode pada halaman **Tambah Data Barang**:

### 1. @extends('layouts.app')

@extends digunakan untuk mewarisi layout dasar yang ada di layouts.app. Artinya, halaman ini akan menggunakan struktur dasar yang sudah didefinisikan di dalam layouts.app, seperti header, footer, dan navigasi.

### 2. @section('content')

Bagian ini mendefinisikan konten halaman yang akan di-inject ke dalam bagian @yield('content') yang ada pada layout utama. Konten yang ada di dalam @section('content') ini akan menggantikan bagian @yield('content') dalam layout tersebut.

### 3. ****Struktur Formulir (Form)****

Di dalam <div class="card-body">, terdapat formulir (**form**) untuk menambah data barang baru. Berikut penjelasan setiap bagian:

**Action Formulir**:

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

Formulir ini akan dikirim ke rute items.store menggunakan metode **POST**. Rute items.store biasanya mengarah ke metode **store** dalam controller **ItemsController** yang bertugas untuk menyimpan data barang baru ke dalam database.

@csrf:

Token **CSRF (Cross-Site Request Forgery)** digunakan untuk melindungi aplikasi dari serangan CSRF. Laravel secara otomatis menambahkan token ini di setiap formulir yang menggunakan metode **POST**.

### 4. ****Input untuk Nama Barang****

**Nama Barang** diinput menggunakan elemen <input> dengan tipe **text**.

required: Atribut ini memastikan bahwa kolom **Nama Barang** wajib diisi.

@error('nama'): Menampilkan pesan kesalahan jika terdapat kesalahan validasi pada input nama.

### 5. ****Input untuk Kategori****

**Kategori** diinput menggunakan elemen <select> yang memungkinkan pengguna memilih kategori dari daftar yang ada.

@foreach($categories as $category): Menampilkan daftar kategori yang tersedia dari variabel $categories yang berasal dari controller.

<option value="{{ $category->id }}">{{ $category->name }}</option>: Menampilkan setiap kategori dalam dropdown. Nilai $category->id akan dikirim ke server saat pengguna memilih kategori tersebut.

### 6. ****Input untuk Harga dan Satuan Harga****

**Harga** diinput dengan tipe **number** dan memiliki atribut min="0" yang memastikan harga yang dimasukkan tidak bernilai negatif.

**Satuan Harga** diinput dengan tipe **text**.

### 7. ****Input untuk Stok dan Satuan Stok****

**Stok** diinput menggunakan elemen <input> dengan tipe **number**. Atribut min="0" memastikan jumlah stok yang dimasukkan tidak bernilai negatif.

**Satuan Stok** diinput dengan tipe **text**.

### 8. ****Input untuk Keterangan****

**Keterangan** diinput menggunakan elemen <textarea>, yang memungkinkan pengguna memasukkan deskripsi atau informasi tambahan tentang barang.

### 9. ****Tombol Submit dan Batal****

**Tombol Simpan** digunakan untuk mengirimkan formulir dan menyimpan data barang ke database. Tombol ini menggunakan kelas btn btn-primary untuk memberi tampilan tombol utama.

**Tombol Batal** mengarahkan pengguna kembali ke halaman daftar barang dengan menggunakan **route** items.index. Tombol ini menggunakan kelas btn btn-secondary untuk memberi tampilan tombol sekunder.

### 10. ****Penggunaan**** @error

Untuk setiap input, Laravel menyediakan @error untuk menangani validasi dan menampilkan pesan kesalahan jika input yang dimasukkan tidak sesuai dengan aturan yang ditentukan di controller atau request.

### Kesimpulan:

**Formulir ini memungkinkan pengguna untuk menambah data barang baru** ke dalam sistem dengan memasukkan nama, kategori, harga, stok, satuan harga, satuan stok, dan keterangan barang.

Formulir ini memanfaatkan validasi untuk memastikan bahwa semua kolom yang dibutuhkan diisi dengan benar.

Pengguna dapat membatalkan proses penambahan barang dengan menekan tombol **Batal**, yang akan membawa mereka kembali ke halaman daftar barang (items.index).

EDIT:

@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header">

        Edit Data Barang

    </div>

    <div class="card-body">

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

            @csrf

            @method('PUT')

            <div class="mb-3">

                <label>Nama Barang</label>

                <input type="text" name="nama" class="form-control" value="{{ old('nama', $item->nama) }}" required>

                @error('nama')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Kategori</label>

                <select name="category\_id" class="form-control" required>

                    @foreach($categories as $category)

                    <option value="{{ $category->id }}" {{ old('category\_id', $item->category\_id) == $category->id ? 'selected' : '' }}>

                        {{ $category->name }}

                    </option>

                    @endforeach

                </select>

                @error('category\_id')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Harga</label>

                <input type="number" name="harga" class="form-control" value="{{ old('harga', $item->harga) }}" min="0" required>

                @error('harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Harga</label>

                <input type="text" name="satuan\_harga" class="form-control" value="{{ old('satuan\_harga', $item->satuan\_harga) }}" required>

                @error('satuan\_harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Stok</label>

                <input type="number" name="stok" class="form-control" value="{{ old('stok', $item->stok) }}" min="0" required>

                @error('stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Stok</label>

                <input type="text" name="satuan\_stok" class="form-control" value="{{ old('satuan\_stok', $item->satuan\_stok) }}" required>

                @error('satuan\_stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Keterangan</label>

                <textarea name="keterangan" class="form-control" rows="3">{{ old('keterangan', $item->keterangan) }}</textarea>

                @error('keterangan')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="d-flex gap-2">

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

                <a href="{{ route('items.index') }}" class="btn btn-secondary">Batal</a>

            </div>

        </form>

    </div>

</div>

@endsection

Berikut penjelasan mengenai kode yang terdapat pada halaman untuk **Edit Data Barang**:

### 1. @extends('layouts.app')

Perintah ini memberitahu Laravel untuk menggunakan **layout** dasar yang ada pada file layouts.app. Ini memungkinkan halaman yang sedang dibuat untuk mewarisi struktur HTML dasar, seperti header, footer, dan navigasi yang ada di dalam layout tersebut.

### 2. @section('content')

Bagian ini mendefinisikan konten yang akan dimasukkan ke dalam @yield('content') yang ada di layout layouts.app. Konten yang berada dalam @section('content') ini akan digantikan pada bagian @yield('content') di layout tersebut.

### 3. ****Struktur Formulir****

Di dalam <div class="card-body">, terdapat sebuah formulir (**form**) yang digunakan untuk memperbarui data barang yang sudah ada. Penjelasan untuk masing-masing bagian formulir:

**Action Formulir**:

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

Aksi formulir menggunakan metode **POST** yang diarahkan ke route items.update, yang mengarah ke metode **update** pada controller **ItemsController**. ID barang yang akan diperbarui di-passing melalui $item->id.

@csrf:

Token CSRF digunakan untuk melindungi formulir ini dari serangan CSRF (Cross-Site Request Forgery). Laravel secara otomatis menyertakan token CSRF di setiap formulir yang menggunakan **POST**.

@method('PUT'):

Laravel menganggap formulir ini menggunakan **POST**, namun untuk memperbarui data (mengedit), kita menggunakan metode **PUT**. Laravel menangani ini dengan @method('PUT') untuk menyatakan bahwa ini adalah permintaan **PUT** meskipun formulir secara teknis menggunakan **POST**.

### 4. ****Input untuk Nama Barang****

**Nama Barang** diinput menggunakan <input> dengan tipe teks.

value="{{ old('nama', $item->nama) }}": Nilai input diisi dengan data lama (dari request sebelumnya) atau dengan nama barang yang sudah ada pada objek $item->nama.

@error('nama'):

Menampilkan pesan kesalahan validasi jika input nama tidak valid (misalnya, kosong atau tidak sesuai aturan).

### 5. ****Input untuk Kategori****

**Kategori** diinput menggunakan elemen <select> yang memungkinkan pengguna memilih kategori dari daftar kategori yang ada di database.

@foreach($categories as $category): Loop untuk menampilkan semua kategori dalam elemen dropdown.

Nilai selected pada elemen <option> akan dipilih berdasarkan data sebelumnya atau nilai lama yang disimpan di sesi.

### 6. ****Input untuk Harga dan Satuan Harga****

**Harga** diinput dengan tipe **number** dan memiliki atribut min="0" agar pengguna tidak bisa memasukkan nilai negatif.

**Satuan Harga** diinput dengan tipe **text**.

### 7. ****Input untuk Stok dan Satuan Stok****

**Stok** diinput menggunakan number, dengan nilai minimal **0** untuk menghindari stok negatif.

**Satuan Stok** diinput menggunakan text.

### 8. ****Input untuk Keterangan****

**Keterangan** diinput menggunakan elemen <textarea>, yang memungkinkan pengguna memasukkan deskripsi atau informasi tambahan tentang barang.

### 9. ****Tombol Submit dan Batal****

**Tombol Submit** digunakan untuk mengirimkan formulir setelah pengguna melakukan perubahan. Ini menggunakan kelas btn btn-primary.

**Tombol Batal** mengarahkan pengguna kembali ke halaman daftar barang dengan menggunakan **route** items.index. Tombol ini menggunakan kelas btn btn-secondary untuk memberikan tampilan sekunder (lebih ringan).

### 10. ****Penggunaan**** @error

Untuk setiap input, Laravel menyediakan @error untuk menangani validasi dan menampilkan pesan kesalahan jika input yang dimasukkan tidak sesuai dengan aturan yang ditentukan di controller atau request.

### Kesimpulan:

**Formulir ini memungkinkan pengguna untuk mengedit informasi barang** yang sudah ada, dengan memanfaatkan data yang sudah ada dalam objek **$item**.

Menggunakan validasi @error untuk menampilkan pesan kesalahan jika input tidak valid.

Data yang dimasukkan akan dikirim ke server menggunakan metode **PUT** untuk memperbarui data barang di database.

**Dropdown kategori** menampilkan semua kategori yang ada, dan harga, stok, satuan harga, serta satuan stok bisa dimasukkan langsung

TAMBAH::

@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header">

        Tambah Data Barang

    </div>

    <div class="card-body">

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

            @csrf

            <div class="mb-3">

                <label>Nama Barang</label>

                <input type="text" name="nama" class="form-control" required>

                @error('nama')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Kategori</label>

                <select name="category\_id" class="form-control" required>

                    <option value="">-- Pilih Kategori --</option>

                    @foreach($categories as $category)

                    <option value="{{ $category->id }}">{{ $category->name }}</option>

                    @endforeach

                </select>

                @error('category\_id')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Harga</label>

                <input type="number" name="harga" class="form-control" min="0" required>

                @error('harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Harga</label>

                <input type="text" name="satuan\_harga" class="form-control" required>

                @error('satuan\_harga')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Stok</label>

                <input type="number" name="stok" class="form-control" min="0" required>

                @error('stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Satuan Stok</label>

                <input type="text" name="satuan\_stok" class="form-control" required>

                @error('satuan\_stok')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="mb-3">

                <label>Keterangan</label>

                <textarea name="keterangan" class="form-control" rows="3"></textarea>

                @error('keterangan')

                    <div class="text-danger">{{ $message }}</div>

                @enderror

            </div>

            <div class="d-flex gap-2">

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

                <a href="{{ route('items.index') }}" class="btn btn-secondary">Batal</a>

            </div>

        </form>

    </div>

</div>

@endsection

Berikut adalah penjelasan mengenai kode **Tambah Data Barang** pada Laravel:

### 1. @extends('layouts.app')

@extends digunakan untuk mewarisi layout dasar yang sudah ada di dalam layouts.app. Layout ini biasanya berisi elemen-elemen umum yang digunakan di seluruh halaman, seperti header, footer, dan navigasi.

### 2. @section('content')

@section('content') digunakan untuk mendefinisikan bagian konten yang akan disisipkan ke dalam layout utama. Bagian ini akan mengisi tempat kosong yang disediakan oleh @yield('content') di dalam layout layouts.app.

### 3. <div class="card">

**Card** adalah elemen UI dari Bootstrap yang digunakan untuk membungkus konten seperti formulir atau informasi terkait, memberikan struktur dan desain yang lebih baik pada tampilan.

### 4. ****Formulir untuk Menambah Data Barang****

**Formulir ini digunakan untuk menambah data barang baru**. Formulir dikirimkan ke rute items.store dengan metode **POST** yang akan mengarah ke aksi **store()** pada controller **ItemsController** untuk menyimpan data ke dalam database.

@csrf:

Menambahkan token CSRF (Cross-Site Request Forgery) untuk melindungi formulir ini dari serangan CSRF. Token ini secara otomatis disertakan oleh Laravel pada setiap formulir yang menggunakan metode **POST**.

### 5. ****Input untuk Nama Barang****

**Nama Barang** diinput menggunakan elemen <input> dengan tipe **text**. Pengguna diharuskan mengisi kolom ini karena ada atribut required.

@error('nama') digunakan untuk menampilkan pesan kesalahan jika ada masalah dalam validasi input nama.

### 6. ****Input untuk Kategori****

**Kategori** menggunakan elemen <select> untuk memilih kategori barang.

@foreach($categories as $category) digunakan untuk menampilkan daftar kategori yang diambil dari controller. Setiap kategori ditampilkan dalam elemen <option>.

<option value="{{ $category->id }}">{{ $category->name }}</option>: Menampilkan setiap kategori dengan $category->name sebagai nama yang ditampilkan dan $category->id sebagai nilai yang akan dipilih.

### 7. ****Input untuk Harga****

**Harga** diinput menggunakan elemen <input> dengan tipe **number** dan nilai minimum **0** melalui atribut min="0", memastikan harga tidak bisa bernilai negatif.

@error('harga') untuk menampilkan pesan kesalahan jika ada masalah validasi pada kolom harga.

### 8. ****Input untuk Satuan Harga****

**Satuan Harga** diinput menggunakan elemen <input> dengan tipe **text** dan wajib diisi karena atribut required.

### 9. ****Input untuk Stok****

**Stok** diinput dengan elemen <input> bertipe **number**, dengan batasan minimal nilai **0** untuk memastikan stok tidak bisa bernilai negatif.

@error('stok') menampilkan pesan kesalahan jika validasi stok gagal.

### 10. ****Input untuk Satuan Stok****

**Satuan Stok** diinput menggunakan elemen <input> bertipe **text** dan wajib diisi.

### 11. ****Input untuk Keterangan****

**Keterangan** diinput menggunakan elemen <textarea>, yang memungkinkan pengguna untuk memberikan penjelasan atau informasi tambahan tentang barang.

### 12. ****Tombol Simpan dan Batal****

**Tombol Simpan** untuk mengirimkan data formulir dan menyimpan barang baru ke database.

Tombol ini menggunakan kelas btn btn-primary untuk memberikan tampilan tombol utama.

**Tombol Batal** mengarah kembali ke halaman daftar barang (items.index) dengan menggunakan route('items.index'). Tombol ini menggunakan kelas btn btn-secondary untuk memberikan tampilan tombol sekunder.

### 13. ****Penggunaan**** @error

Untuk setiap input, terdapat penggunaan @error untuk menangani dan menampilkan pesan kesalahan jika validasi gagal pada setiap kolom. Misalnya, jika kolom nama tidak diisi atau inputnya tidak valid, maka pesan kesalahan akan ditampilkan di bawah kolom tersebut.

### Kesimpulan

Halaman ini menyediakan formulir untuk menambah data barang baru dengan beberapa inputan seperti nama barang, kategori, harga, stok, dan keterangan.

Setiap kolom dalam formulir dilengkapi dengan validasi yang akan menampilkan pesan kesalahan jika ada kesalahan dalam pengisian.

Tombol **Simpan** untuk menyimpan data barang baru ke dalam database, dan tombol **Batal** untuk kembali ke halaman sebelumnya tanpa menyimpan data..

USER.INDEX:

@extends('layouts.app')

@section('content')

<div class="container-fluid">

    <div class="row mb-4">

        <div class="col">

            <h2>Manajemen User</h2>

        </div>

        <div class="col text-end">

            @can('manage-users')

            <a href="{{ route('users.create') }}" class="btn btn-primary">

                <i class="fas fa-plus"></i> Tambah User

            </a>

            @endcan

        </div>

    </div>

    <div class="card shadow">

        <div class="card-body">

            <div class="table-responsive">

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

                    <thead class="table-dark">

                        <tr>

                            <th>#</th>

                            <th>Nama</th>

                            <th>Email</th>

                            <th>Username</th>

                            <th>Role</th>

                            <th>Aksi</th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach($users as $key => $user)

                        <tr>

                            <td>{{ $key + 1 }}</td>

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

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

                            <td>{{ $user->username ?? '-' }}</td>

                            <td>

                                <span class="badge bg-{{ $user->role === 'super\_admin' ? 'danger' : 'primary' }}">

                                    {{ $roles[$user->role] }}

                                </span>

                            </td>

                            <td>

                                <div class="d-flex gap-2">

                                    @if(auth()->user()->can('manage-users') || auth()->id() === $user->id)

                                    <a href="{{ route('users.edit', $user->id) }}" class="btn btn-sm btn-warning">

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

                                    </a>

                                    @endif

                                    @can('manage-users')

                                    @if(auth()->id() !== $user->id)

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

                                        @csrf

                                        @method('DELETE')

                                        <button type="submit" class="btn btn-sm btn-danger"

                                            onclick="return confirm('Yakin ingin menghapus user ini?')">

                                            <i class="fas fa-trash"></i>

                                        </button>

                                    </form>

                                    @endif

                                    @endcan

                                </div>

                            </td>

                        </tr>

                        @endforeach

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@endsection

Berikut adalah penjelasan mengenai kode **Manajemen User** pada Laravel:

### 1. @extends('layouts.app')

@extends digunakan untuk mewarisi layout dasar yang ada di layouts.app. Layout ini biasanya berisi elemen umum seperti header, footer, dan navigasi yang digunakan di seluruh halaman.

### 2. @section('content')

@section('content') digunakan untuk mendefinisikan bagian konten halaman yang akan dimasukkan ke dalam layout utama. Bagian ini akan mengisi tempat kosong yang disediakan oleh @yield('content') di dalam layout layouts.app.

### 3. ****Kontainer dan Struktur Halaman****

<div class="container-fluid"> adalah elemen untuk membuat sebuah kontainer responsif yang akan menampung semua konten pada halaman ini. container-fluid membuat kontainer lebar penuh, menyesuaikan ukuran layar.

<div class="row mb-4"> adalah baris untuk mengatur layout dengan margin bawah (**mb-4**), yang mengandung dua kolom:

Kolom pertama (<div class="col">) berisi judul **Manajemen User**.

Kolom kedua (<div class="col text-end">) berisi tombol **Tambah User** yang hanya dapat diakses oleh pengguna dengan izin manage-users menggunakan @can.

### 4. ****Tombol Tambah User:****

@can('manage-users') memeriksa apakah pengguna saat ini memiliki izin untuk mengelola pengguna (dengan izin manage-users). Jika ya, maka tombol **Tambah User** akan ditampilkan.

Tombol **Tambah User** mengarah ke rute users.create yang memungkinkan admin untuk menambah pengguna baru.

### 5. ****Tabel Daftar Pengguna****

**Tabel** ini berisi daftar pengguna yang ada, dengan kolom-kolom berikut:

**#**: Nomor urut pengguna dalam daftar.

**Nama**: Nama pengguna.

**Email**: Email pengguna.

**Username**: Nama pengguna (jika ada).

**Role**: Menampilkan badge yang menunjukkan peran pengguna, apakah **super\_admin** atau peran lainnya.

**Aksi**: Berisi tombol untuk mengedit dan menghapus pengguna.

@foreach($users as $key => $user): Melakukan iterasi pada setiap pengguna di dalam variabel $users dan menampilkan informasi masing-masing dalam baris tabel.

### 6. ****Menampilkan Role Pengguna****

$user->role memeriksa peran pengguna dan menampilkan **badge** dengan warna yang berbeda berdasarkan peran:

**Super Admin**: Badge berwarna merah (**danger**).

Peran lainnya: Badge berwarna biru (**primary**).

$roles[$user->role] digunakan untuk menampilkan nama peran berdasarkan data peran yang disimpan di dalam array $roles.

### 7. ****Tombol Aksi (Edit dan Hapus)****

**Edit**: Pengguna dapat mengedit data pengguna jika mereka memiliki izin manage-users atau jika pengguna yang sedang login adalah pengguna yang sedang ditampilkan (auth()->id() === $user->id).

Tombol **Edit** mengarah ke rute users.edit yang membuka halaman untuk mengedit pengguna.

**Hapus**: Pengguna dapat menghapus pengguna lain jika mereka memiliki izin manage-users dan bukan pengguna yang sedang login (auth()->id() !== $user->id).

Proses penghapusan dilakukan dengan formulir **POST** dengan metode **DELETE** untuk menghapus pengguna melalui rute users.destroy. Setelah tombol di klik, pengguna akan diminta konfirmasi untuk menghapus data.

### 8. ****Formulir Penghapusan Pengguna****

Formulir **DELETE** yang digunakan untuk menghapus pengguna dari database, menggunakan metode **POST** dengan @method('DELETE') agar formulir tersebut bisa mengirimkan permintaan **DELETE** meskipun formulir HTML hanya mendukung **GET** dan **POST**.

onclick="return confirm('Yakin ingin menghapus user ini?')": Menampilkan konfirmasi sebelum melanjutkan penghapusan, untuk mencegah penghapusan yang tidak sengaja.

### 9. ****Penggunaan Badge untuk Role****

<span class="badge bg-{{ $user->role === 'super\_admin' ? 'danger' : 'primary' }}">: Membuat badge dengan warna yang dinamis, berdasarkan role pengguna. Badge akan berwarna **red (danger)** jika role pengguna adalah **super\_admin**, dan **blue (primary)** untuk peran lainnya.

### Kesimpulan

**Manajemen User** adalah halaman yang digunakan untuk melihat, mengedit, dan menghapus pengguna dalam aplikasi.

Hanya pengguna yang memiliki izin manage-users yang dapat menambah, mengedit, atau menghapus pengguna.

Tabel menampilkan informasi pengguna termasuk nama, email, username, dan role, dengan aksi edit dan hapus yang bisa dilakukan tergantung pada izin pengguna yang sedang login.

REPORT.TRANSACTION:

@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header d-flex justify-content-between align-items-center">

        <h5>Laporan Transaksi</h5>

        <div>

            <form action="{{ route('reports.transaction') }}" method="GET" class="form-inline">

                <input type="date" name="start\_date" value="{{ $startDate }}" class="form-control form-control-sm mr-2">

                <input type="date" name="end\_date" value="{{ $endDate }}" class="form-control form-control-sm mr-2">

                <button type="submit" class="btn btn-sm btn-primary mr-2">

                    <i class="fas fa-filter"></i> Filter

                </button>

                <a href="#" onclick="window.print()" class="btn btn-sm btn-success">

                    <i class="fas fa-print"></i> Cetak

                </a>

            </form>

        </div>

    </div>

    <div class="card-body">

        <div class="row mb-4">

            <div class="col-md-6">

                <h5>Stok Masuk</h5>

                <div class="table-responsive">

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

                        <thead class="bg-light">

                            <tr>

                                <th>Tanggal</th>

                                <th>Barang</th>

                                <th>Jumlah</th>

                                <th>Harga Beli</th>

                                <th>Total</th>

                            </tr>

                        </thead>

                        <tbody>

                            @foreach($stockIns as $trx)

                            <tr>

                                <td>{{ $trx->tanggal->format('d/m/Y') }}</td>

                                <td>{{ $trx->item->nama }}</td>

                                <td>{{ $trx->jumlah }} {{ $trx->satuan\_jumlah }}</td>

                                <td>Rp {{ number\_format($trx->harga\_per\_satuan, 0, ',', '.') }}</td>

                                <td>Rp {{ number\_format($trx->harga\_per\_satuan \* $trx->jumlah, 0, ',', '.') }}</td>

                            </tr>

                            @endforeach

                        </tbody>

                    </table>

                </div>

            </div>

            <div class="col-md-6">

                <h5>Stok Keluar</h5>

                <div class="table-responsive">

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

                        <thead class="bg-light">

                            <tr>

                                <th>Tanggal</th>

                                <th>Barang</th>

                                <th>Jumlah</th>

                                <th>Harga Jual</th>

                                <th>Total</th>

                            </tr>

                        </thead>

                        <tbody>

                            @foreach($stockOuts as $trx)

                            <tr>

                                <td>{{ $trx->tanggal->format('d/m/Y') }}</td>

                                <td>{{ $trx->item->nama }}</td>

                                <td>{{ $trx->jumlah }} {{ $trx->satuan\_jumlah }}</td>

                                <td>Rp {{ number\_format($trx->harga\_jual\_per\_satuan, 0, ',', '.') }}</td>

                                <td>Rp {{ number\_format($trx->harga\_jual\_per\_satuan \* $trx->jumlah, 0, ',', '.') }}</td>

                            </tr>

                            @endforeach

                        </tbody>

                    </table>

                </div>

            </div>

        </div>

        <div class="row">

            <div class="col-md-12">

                <h5>Ringkasan Transaksi</h5>

                <div class="table-responsive">

                    <table class="table table-bordered">

                        <thead class="bg-light">

                            <tr>

                                <th>Jenis</th>

                                <th>Jumlah Transaksi</th>

                                <th>Total Kuantitas</th>

                                <th>Total Nilai</th>

                            </tr>

                        </thead>

                        <tbody>

                            <tr>

                                <td>Stok Masuk</td>

                                <td>{{ $stockIns->count() }}</td>

                                <td>{{ $stockIns->sum('jumlah') }}</td>

                                <td>Rp {{ number\_format($stockIns->sum(function($trx) {

                                    return $trx->harga\_per\_satuan \* $trx->jumlah;

                                }), 0, ',', '.') }}</td>

                            </tr>

                            <tr>

                                <td>Stok Keluar</td>

                                <td>{{ $stockOuts->count() }}</td>

                                <td>{{ $stockOuts->sum('jumlah') }}</td>

                                <td>Rp {{ number\_format($stockOuts->sum(function($trx) {

                                    return $trx->harga\_jual\_per\_satuan \* $trx->jumlah;

                                }), 0, ',', '.') }}</td>

                            </tr>

                        </tbody>

                    </table>

                </div>

            </div>

        </div>

    </div>

</div>

@endsection

Berikut adalah penjelasan tentang kode **Laporan Transaksi** di Laravel:

### 1. @extends('layouts.app')

@extends('layouts.app') digunakan untuk mewarisi layout dasar yang ada di layouts.app. Layout ini berisi elemen-elemen umum seperti header, footer, dan navigasi yang digunakan di seluruh halaman.

### 2. @section('content')

@section('content') mendefinisikan bagian konten untuk halaman ini. Konten ini akan dimasukkan ke dalam tempat kosong yang disediakan oleh @yield('content') dalam layout layouts.app.

### 3. <div class="card">

<div class="card"> adalah elemen untuk membuat tampilan berbentuk **card** dengan header dan body, memberikan kesan yang rapi dan terstruktur pada halaman.

### 4. ****Filter Tanggal (Form)****

<form action="{{ route('reports.transaction') }}" method="GET" class="form-inline">: Formulir ini digunakan untuk melakukan filter laporan berdasarkan tanggal, dengan **method GET** agar bisa mengirimkan parameter **start\_date** dan **end\_date** ke URL untuk memfilter data laporan transaksi.

<input type="date" name="start\_date" value="{{ $startDate }}" class="form-control form-control-sm mr-2">: Input untuk memilih tanggal mulai laporan transaksi.

<input type="date" name="end\_date" value="{{ $endDate }}" class="form-control form-control-sm mr-2">: Input untuk memilih tanggal akhir laporan transaksi.

**Tombol Filter**: Tombol untuk mengirimkan formulir dan memfilter laporan berdasarkan rentang tanggal yang dipilih.

**Tombol Cetak**: Tombol untuk mencetak laporan yang ditampilkan. Menggunakan window.print() untuk memunculkan dialog print pada browser.

### 5. ****Tabel Laporan Stok Masuk****

**Stok Masuk**: Bagian ini menampilkan daftar transaksi stok masuk, yang berisi informasi tentang tanggal, barang, jumlah, harga beli per satuan, dan total harga beli.

@foreach($stockIns as $trx): Melakukan iterasi melalui **$stockIns**, yang berisi data transaksi stok masuk. Setiap transaksi ditampilkan dalam baris tabel.

**Format Tanggal**: $trx->tanggal->format('d/m/Y') digunakan untuk memformat tanggal transaksi menjadi format dd/mm/yyyy.

**Total Harga**: Menggunakan number\_format() untuk menampilkan harga dalam format mata uang Indonesia (misalnya, **Rp 10,000**).

### 6. ****Tabel Laporan Stok Keluar****

**Stok Keluar**: Bagian ini menampilkan daftar transaksi stok keluar, yang berisi informasi serupa dengan stok masuk, tetapi menggunakan harga jual per satuan dan total harga jual.

@foreach($stockOuts as $trx): Melakukan iterasi melalui **$stockOuts**, yang berisi data transaksi stok keluar. Setiap transaksi ditampilkan dalam baris tabel.

### 7. ****Ringkasan Transaksi****

**Ringkasan Transaksi**: Menampilkan ringkasan dari total transaksi stok masuk dan stok keluar, dengan informasi sebagai berikut:

**Jumlah Transaksi**: Jumlah transaksi untuk masing-masing jenis (stok masuk dan keluar).

**Total Kuantitas**: Jumlah total barang yang masuk atau keluar.

**Total Nilai**: Nilai total transaksi, dihitung dengan mengalikan jumlah barang dengan harga per satuan.

$stockIns->sum('jumlah'): Menghitung total kuantitas stok masuk.

$stockIns->sum(function($trx) { ... }): Menghitung total nilai stok masuk berdasarkan harga per satuan dan jumlah transaksi.

**Format Mata Uang**: Menggunakan number\_format() untuk menampilkan total nilai dalam format mata uang Indonesia.

### 8. ****Bagian Tabel****

<table class="table table-sm table-bordered">: Membuat tabel dengan kelas table-sm untuk ukuran tabel yang lebih kecil dan table-bordered untuk menambahkan garis batas pada tabel.

thead **dan** tbody: Digunakan untuk mendefinisikan bagian header dan tubuh tabel, dengan bg-light pada header untuk memberi warna latar belakang yang lebih terang.

### 9. @endsection

@endsection menutup bagian konten yang dibuka dengan @section('content'), mengakhiri penulisan konten untuk halaman ini.

### Kesimpulan:

Halaman ini berfungsi untuk menampilkan laporan transaksi stok, baik **stok masuk** maupun **stok keluar**, dengan fitur untuk memfilter data berdasarkan rentang tanggal yang dipilih oleh pengguna. Setiap transaksi ditampilkan dalam bentuk tabel, termasuk rincian barang, jumlah, harga, dan total. Selain itu, terdapat ringkasan transaksi yang menampilkan jumlah transaksi, total kuantitas, dan total nilai transaksi untuk stok masuk dan keluar. Ada juga tombol untuk mencetak laporan tersebut.

PROFIT:  
@extends('layouts.app')

@section('content')

<div class="card">

    <div class="card-header d-flex justify-content-between align-items-center">

        <h5>Laporan Profit/Laba</h5>

        <div class="d-flex">

            <form id="filterForm" action="{{ route('reports.profit') }}" method="GET" class="form-inline mr-3">

                <div class="input-group input-group-sm">

                    <input type="date" name="start\_date" value="{{ $startDate }}" class="form-control form-control-sm">

                    <span class="input-group-text">s/d</span>

                    <input type="date" name="end\_date" value="{{ $endDate }}" class="form-control form-control-sm">

                    <button type="submit" class="btn btn-primary btn-sm">

                        <i class="fas fa-filter"></i> Filter

                    </button>

                </div>

            </form>

            <div class="btn-group">

                <button type="button" class="btn btn-success btn-sm dropdown-toggle" data-bs-toggle="dropdown">

                    <i class="fas fa-file-export"></i> Export

                </button>

                <ul class="dropdown-menu">

                    <li><a class="dropdown-item" href="#" onclick="exportReport('pdf')"><i class="fas fa-file-pdf text-danger"></i> PDF</a></li>

                    <li><a class="dropdown-item" href="#" onclick="exportReport('excel')"><i class="fas fa-file-excel text-success"></i> Excel</a></li>

                    <li><a class="dropdown-item" href="#" onclick="exportReport('word')"><i class="fas fa-file-word text-primary"></i> Word</a></li>

                </ul>

            </div>

        </div>

    </div>

    <div class="card-body">

        <div class="table-responsive" id="reportContent">

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

                <thead class="thead-dark">

                    <tr>

                        <th>No</th>

                        <th>Nama Barang</th>

                        <th>Kategori</th>

                        <th>Harga Beli</th>

                        <th>Terjual</th>

                        <th>Total Penjualan</th>

                        <th>Total Pembelian</th>

                        <th>Laba</th>

                    </tr>

                </thead>

                <tbody>

                    @foreach($items as $key => $item)

                    @php

                        $sold = $item->stockOuts->sum('jumlah');

                        $totalSale = $item->stockOuts->sum(function($trx) {

                            return $trx->harga\_jual\_per\_satuan \* $trx->jumlah;

                        });

                        $totalPurchase = $item->harga \* $sold;

                        $profit = $totalSale - $totalPurchase;

                    @endphp

                    <tr>

                        <td>{{ $key+1 }}</td>

                        <td>{{ $item->nama }}</td>

                        <td>{{ $item->category->name ?? '-' }}</td>

                        <td>Rp {{ number\_format($item->harga, 0, ',', '.') }}</td>

                        <td>{{ $sold }}</td>

                        <td>Rp {{ number\_format($totalSale, 0, ',', '.') }}</td>

                        <td>Rp {{ number\_format($totalPurchase, 0, ',', '.') }}</td>

                        <td class="{{ $profit >= 0 ? 'text-success' : 'text-danger' }}">

                            Rp {{ number\_format($profit, 0, ',', '.') }}

                        </td>

                    </tr>

                    @endforeach

                </tbody>

            </table>

        </div>

    </div>

</div>

<script>

function exportReport(type) {

    const form = document.getElementById('filterForm');

    const action = "{{ route('export', ':type') }}".replace(':type', type);

    const exportForm = document.createElement('form');

    exportForm.method = 'GET';

    exportForm.action = action;

    const startDate = document.createElement('input');

    startDate.type = 'hidden';

    startDate.name = 'start\_date';

    startDate.value = form.elements['start\_date'].value;

    exportForm.appendChild(startDate);

    const endDate = document.createElement('input');

    endDate.type = 'hidden';

    endDate.name = 'end\_date';

    endDate.value = form.elements['end\_date'].value;

    exportForm.appendChild(endDate);

    document.body.appendChild(exportForm);

    exportForm.submit();

    document.body.removeChild(exportForm);

}

</script>

@endsection

Berikut adalah penjelasan **detail** dari kode Blade Laravel yang menampilkan **laporan profit/laba** berdasarkan penjualan:

### 💡 1. @extends('layouts.app')

Mengindikasikan bahwa tampilan ini mewarisi layout dari file layouts/app.blade.php. Layout ini biasanya berisi struktur dasar HTML seperti <head>, navbar, dan footer.

### 💡 2. @section('content') ****hingga**** @endsection

Bagian ini mengisi section content dari layout utama. Semua elemen dalamnya akan ditampilkan dalam area @yield('content') di file layout.

### 📄 ****Card Header (Filter & Export)****

<div class="card-header d-flex justify-content-between align-items-center">

Menampilkan:

Judul laporan: Laporan Profit/Laba.

Form pencarian berdasarkan tanggal (start\_date, end\_date).

Tombol **filter** untuk submit form ke route reports.profit.

Tombol dropdown **Export** untuk cetak PDF, Excel, atau Word.

#### 📆 Filter Form:

<form id="filterForm" action="{{ route('reports.profit') }}" method="GET">

Mengirim tanggal ke controller agar laporan bisa difilter.

#### ⬇️ Dropdown Export:

<ul class="dropdown-menu">

<li><a class="dropdown-item" onclick="exportReport('pdf')">PDF</a></li>

...

</ul>

Akan menjalankan JavaScript function exportReport(type) untuk mengirim filter ke route export.

### 📊 ****Card Body: Tabel Profit****

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

Menampilkan data per barang dengan informasi:

| **No** | **Nama Barang** | **Kategori** | **Harga Beli** | **Terjual** | **Total Penjualan** | **Total Pembelian** | **Laba** |
| --- | --- | --- | --- | --- | --- | --- | --- |

#### 🧠 Perhitungan dalam @php:

@php

$sold = $item->stockOuts->sum('jumlah');

$totalSale = $item->stockOuts->sum(function($trx) {

return $trx->harga\_jual\_per\_satuan \* $trx->jumlah;

});

$totalPurchase = $item->harga \* $sold;

$profit = $totalSale - $totalPurchase;

@endphp

$sold: total jumlah barang yang **keluar** (dijual).

$totalSale: total **pendapatan** dari penjualan.

$totalPurchase: total **biaya pembelian** (harga beli x jumlah).

$profit: laba (selisih penjualan - pembelian).

#### 🔢 Format Angka:

number\_format($profit, 0, ',', '.')

Contoh output: Rp 12.500.000

#### 🎨 Warna Teks Laba:

<td class="{{ $profit >= 0 ? 'text-success' : 'text-danger' }}">

Hijau (text-success) jika profit ≥ 0.

Merah (text-danger) jika profit negatif (rugi).

### 🧾 ****JavaScript: exportReport(type)****

Fungsi JavaScript untuk ekspor PDF/Excel/Word:

function exportReport(type) {

const action = "{{ route('export', ':type') }}".replace(':type', type);

...

exportForm.submit();

}

Mengambil nilai start\_date dan end\_date dari form.

Membuat form dinamis dan mengirim ke URL export sesuai tipe (contoh: /export/pdf).

Data tetap difilter sesuai tanggal.

### 🔁 Alur Keseluruhan:

User memilih tanggal filter → klik **Filter** → hasil ditampilkan ulang.

User klik **Export** → laporan yang difilter dikirim ke endpoint export/{type}.

Laporan dapat disimpan sebagai PDF, Excel, atau Word.