Jobsheet Praktikum Kelompok: Pertemuan 3

Frontend & Templating dengan Blade, Tailwind, dan Alpine (Kerja Kelompok)

Mata Kuliah	Pemrograman Web Lanjut (SIB245007)
Pertemuan	3 (Minggu 3)
Durasi	2 sesi × 170 menit
Sub-CPMK	Sub-CPMK 2: Mahasiswa mampu menerapkan model, templating, dan operasi CRUD dalam pengembangan aplikasi web berbasis framework.
Mode Pengerjaan	Kelompok (sesuai pembagian dosen), satu repositori GitHub bersama per kelompok
Kode Awal	template repository `github.com/se-polinema/simple-pos-ch02`

A. Capaian Praktikum

Setelah menyelesaikan jobsheet kelompok ini, kamu mampu:

1. Menyiapkan repositori GitHub bersama untuk kerja kelompok, lalu berkontribusi lewat feature branch dan Pull Request yang tercatat atas nama sendiri.
2. Menyusun layout Blade (`@extends`/`@section`/`@yield`) dan component navigasi yang dipakai ulang lintas halaman.
3. Menata halaman kasir `/pos` dengan class utility Tailwind CSS yang dimuat lewat Vite, memakai `npm run dev` untuk hot reload.
4. Mengimplementasikan keranjang belanja dinamis dengan Alpine.js (`x-data`, `@click`, `x-for`, `x-text`) tanpa reload halaman.
5. Menjelaskan mengapa `{{ }}` wajib dipakai untuk data pengguna, dan mengapa subtotal yang dihitung di klien harus dihitung ulang di server.

B. Persiapan dan Prasyarat

- **Alat**: sama seperti Pertemuan 2 (PHP 8.2+, Composer, Node.js, Git), ditambah akun GitHub aktif untuk setiap anggota kelompok.
- **Pembagian kelompok**: bekerja dalam kelompok sesuai pembagian dosen. Jobsheet ini tidak menentukan siapa mengerjakan langkah yang mana, itu keputusan kelompok sendiri. Pastikan pembagian membuat Langkah 5 sampai 10, serta tantangan mandiri di Langkah 12, terbagi rata sehingga tiap anggota tercatat minimal satu commit bermakna lewat Pull Request masing-masing.
- **Identitas git per anggota**: sebelum mulai, setiap anggota memeriksa identitas git di laptopnya sendiri:

  ```
  git config --global user.name "Nama Lengkap"
  git config --global user.email "email-akun-github-kamu@..."
  ```

  Email ini sebaiknya sama dengan email akun GitHub kamu, supaya tiap commit, termasuk yang ada di dalam Pull Request-mu, terhubung ke profilmu dan kontribusimu terhitung saat dinilai.

  > ⚠ **Kalau memakai komputer lab bersama:** pakai `git config --local` (tanpa `--global`) di dalam folder proyek yang sudah kamu clone, supaya identitas git tidak tertinggal untuk pengguna berikutnya.

- **Kode awal kelompok**: kelompok memulai dari template repository `simple-pos-ch02`, bukan dari proyek pribadi salah satu anggota. Proyek individumu dari Pertemuan 1 dan 2 tetap punyamu sendiri dan tidak dipakai di jobsheet kelompok ini.

C. Langkah Kerja

Kelompok bekerja dengan feature branch: setiap langkah pembangunan (Langkah 5 sampai 10) dikerjakan di branch terpisah dari `main`, lalu digabungkan ke `main` lewat Pull Request (PR) di GitHub. Beberapa langkah saling bergantung, halaman kasir butuh layout lebih dulu, keranjang butuh halaman kasir lebih dulu, jadi selalu mulai dari `main` yang sudah ter-update (`git pull`) sebelum membuat branch baru, supaya branchmu tidak ketinggalan dari Pull Request yang sudah digabungkan anggota lain.

Langkah 1: Membuat repositori kelompok dari template

`simple-pos-ch02` sudah disiapkan sebagai template repository: kelompok tinggal membuat salinannya sendiri lewat GitHub, tanpa perlu clone dan atur remote secara manual. Salah satu anggota melakukan langkah ini lebih dulu.

1. Buka `https://github.com/se-polinema/simple-pos-ch02`.
2. Klik tombol hijau **Use this template → Create a new repository**.
3. Pilih akunmu sebagai pemilik, beri nama repositori, misalnya `simple-pos-kelompok-03`, pilih Public atau Private, lalu klik **Create repository**.
4. Di halaman repositori yang baru dibuat, buka **Settings → Collaborators**, lalu tambahkan setiap anggota kelompok dan dosen yang menilai.

> ✓ **Checkpoint:** repositori baru berisi seluruh berkas proyek Laravel (`composer.json`, `app/`, `routes/`, dst.) tapi hanya punya **satu commit** di riwayatnya, karena GitHub membuat repositori dari template dengan menyalin isi berkasnya saja, bukan riwayat commit `simple-pos-ch02`. Setiap anggota sudah menerima serta menyetujui undangan collaborator.

> ⚠ **Jika gagal:** kalau tombol **Use this template** tidak muncul, pastikan kamu membuka halaman repositori `simple-pos-ch02` itu sendiri (bukan hasil pencarian), dan sudah login ke GitHub.

Langkah 2: Semua anggota melakukan clone dan setup

Setiap anggota, di laptop masing-masing:

```
git clone https://github.com/<username-pembuat-repositori>/simple-pos-kelompok-03.git
cd simple-pos-kelompok-03
composer install
npm install
cp .env.example .env
php artisan key:generate
touch database/database.sqlite
php artisan migrate:fresh --seed
```

> ✓ **Checkpoint:** `git log --oneline` menampilkan baris yang sama persis di setiap laptop, dan `php artisan serve` berjalan tanpa error.

Langkah 3: Menyepakati alur kerja feature branch dan Pull Request

Sebelum mulai menulis kode, kelompok sepakat memakai alur berikut untuk setiap langkah kerja (Langkah 5 sampai 10):

```
git checkout main
git pull
git checkout -b short-branch-name
# ...kerjakan tugasmu...
git add .
git commit -m "pesan singkat sesuai tugas"
git push -u origin short-branch-name
```

Beri nama branch yang jelas, misalnya `base-layout` atau `alpine-cart`, supaya mudah dikenali seluruh kelompok. Setelah push, buka repositori di GitHub dan buat Pull Request dari branch itu ke `main`. Klik **Merge pull request** dengan opsi **Create a merge commit** (bukan **Squash and merge**), supaya commit asli dan namamu tetap tercatat di riwayat, lalu hapus branch-nya. Terakhir, kembali ke `main` dan ambil hasilnya:

```
git checkout main
git pull
```

Kalau sempat, minta satu anggota lain memeriksa Pull Request-mu sebelum di-merge, supaya kelompok terbiasa dengan code review.

> ⚠ **Jika gagal:** GitHub menampilkan konflik saat membuat Pull Request berarti branch-mu ketinggalan dari `main`. Jalankan `git checkout main && git pull`, lalu `git checkout your-branch-name && git merge main`, selesaikan konfliknya, dan push ulang.

Langkah 4: Semua anggota menjalankan Vite

```
npm run dev
```

Jalankan di terminal kedua, terpisah dari `php artisan serve`. Sambil menunggu, buka `resources/css/app.css` dan pastikan baris pertamanya `@import 'tailwindcss';`, lalu intip `vite.config.js` dan perhatikan plugin `laravel()` dan `tailwindcss()` yang sudah terdaftar sejak Pertemuan 1.

> ✔ **Checkpoint:** terminal menampilkan baris `VITE vX.X.X ready` diikuti alamat lokal seperti `http://localhost:5173/`.

> ⚠ **Jika gagal:** membuka `/pos` di browser dan melihat `ViteManifestNotFoundException` berarti `npm run dev` belum berjalan di terminal itu.

Kalau sudah terbiasa dengan dua terminal ini, composer run dev menjalankan php artisan serve, npm run dev, dan proses lain sekaligus dalam satu terminal, lewat script yang sudah didefinisikan di composer.json bawaan Laravel.

Langkah 5: Menyederhanakan `TransactionController`

Kode awal kelompok (template `simple-pos-ch02`) sudah berisi versi produksi `TransactionController`, yang memanggil Model `Product` dan beberapa Model lain. Model-model itu baru akan kamu buat sendiri pada pertemuan desain basis data, jadi untuk sekarang sederhanakan dulu controller-nya agar halaman kasir bisa dibangun tanpa menunggu Model. Berkas ini sudah ada sejak dibuat lewat `php artisan make:controller TransactionController` di Pertemuan 2, jadi langkah ini hanya mengedit isinya, bukan membuat berkas baru.

Buat branch baru dari `main` terbaru, misalnya `simplify-controller`:

```
git checkout main
git pull
git checkout -b simplify-controller
```

Ganti seluruh isi `app/Http/Controllers/TransactionController.php` menjadi:

```php
<?php

namespace App\Http\Controllers;

class TransactionController extends Controller
{
    public function create()
    {
        $products = collect([
            (object) ['id' => 1, 'name' => 'Kopi Sachet', 'price' => 3000, 'stock' =>
            ↪  40],
            (object) ['id' => 2, 'name' => 'Teh Celup', 'price' => 2500, 'stock' => 25],
            (object) ['id' => 3, 'name' => 'Mie Instan', 'price' => 3500, 'stock' => 8],
            (object) ['id' => 4, 'name' => 'Air Mineral 600ml', 'price' => 4000, 'stock'
            ↪  => 60],
            (object) ['id' => 5, 'name' => 'Roti Tawar', 'price' => 12000, 'stock' =>
            ↪  15],
            (object) ['id' => 6, 'name' => 'Gula Pasir 1kg', 'price' => 15000, 'stock' =>
            ↪  5],
        ]);

        return view('pos.create', ['products' => $products]);
```

```
    }

    public function store()
    {
        return 'Transaksi disimpan (belum ada logika penyimpanan)';
    }

    public function index()
    {
        return 'Daftar transaksi';
    }

    public function show(string $id)
    {
        return "Detail transaksi #{$id}";
    }
}
```

Data produk di atas sengaja ditulis langsung di controller (bukan dari basis data): cukup untuk membangun tampilan sekarang, dan akan diganti dengan Model `Product` sungguhan pada pertemuan berikutnya.

```
git add .
git commit -m "sederhanakan TransactionController untuk pertemuan 3"
git push -u origin simplify-controller
```

Buka Pull Request ke `main`, merge (ikuti alur di Langkah 3), lalu semua anggota kembali ke `main` dan pull.

> ✓ **Checkpoint:** membuka `/pos` di browser menampilkan error "view [pos.create] not found", bukan lagi error Model. Ini wajar, karena view-nya belum dibuat sampai Langkah 8.

Langkah 6: Membuat layout aplikasi

Buat branch baru dari `main` terbaru, misalnya `base-layout`. Buat berkasnya lewat artisan, lalu ganti isinya:

```
php artisan make:view layouts.app
```

Notasi titik (`layouts.app`) otomatis membuat folder `layouts/` kalau belum ada, dan menghasilkan berkas di `resources/views/layouts/app.blade.php`. Isi berkas itu menjadi:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <title>@yield('title', 'Simple POS')</title>
    @vite(['resources/css/app.css', 'resources/js/app.js'])
</head>
<body>
    <x-nav />
    <main>@yield('content')</main>
</body>
</html>
```

```
git add .
git commit -m "tambah layout dasar aplikasi"
git push -u origin base-layout
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✓ **Checkpoint:** berkas `resources/views/layouts/app.blade.php` ada di semua laptop setelah pull. Halaman `/pos` belum berubah tampilannya, layout ini baru benar-benar terpakai di Langkah 8.

Langkah 7: Membuat component navigasi

Buat branch baru, misalnya `nav-component`. Buat berkasnya lewat artisan:

```
php artisan make:component nav --view
```

Opsi `--view` membuat component tanpa class PHP terpisah, hanya berkas Blade, cocok untuk component sederhana seperti nav ini. Isi `resources/views/components/nav.blade.php` yang baru dibuat:

```
<nav class="bg-slate-900 text-white px-4 py-3 flex gap-4">
    <span class="font-semibold">Simple POS</span>
    <a href="{{ route('pos.create') }}" class="hover:underline">Kasir</a>
    <a href="{{ route('transactions.index') }}" class="hover:underline">Transaksi</a>
</nav>
```

Component ini otomatis terdeteksi Laravel dari namanya: dipanggil lewat `<x-nav />`, seperti yang sudah ditulis di layout pada Langkah 6.

```
git add .
git commit -m "tambah component nav"
git push -u origin nav-component
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✓ **Checkpoint:** berkas `resources/views/components/nav.blade.php` ada di semua laptop.

Langkah 8: Membuat halaman kasir

Buat branch baru, misalnya `cashier-page`. Buat berkasnya lewat artisan, lalu ganti isinya:

```
php artisan make:view pos.create
```

Isi `resources/views/pos/create.blade.php` menjadi:

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

@section('title', 'Kasir')

@section('content')
    <h1 class="text-lg font-semibold mb-4">Transaksi Kasir</h1>
    <div class="grid grid-cols-3 gap-4">
        @foreach ($products as $product)
            <div class="border rounded-md p-3">
                <p class="font-medium">{{ $product->name }}</p>
                <p class="text-sm text-slate-500">Rp {{ number_format($product->price)
                ↪  }}</p>
            </div>
        @endforeach
    </div>
@endsection
```

Perhatikan `{{ $product->name }}`: kurung kurawal ganda ini otomatis melakukan escaping HTML, supaya nama produk yang mengandung karakter seperti < tidak bisa dipakai untuk menyuntikkan markup asing ke halaman. Data yang berasal dari input pengguna wajib ditampilkan lewat `{{ }}`, bukan `{!! !!}`, yang melewati escaping sama sekali.

```
git add .
git commit -m "tambah halaman kasir dengan tailwind"
git push -u origin cashier-page
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✓ **Checkpoint:** membuka `http://127.0.0.1:8000/pos` menampilkan grid enam produk dengan nav di atasnya. Sambil `npm run dev` berjalan, coba ubah satu class Tailwind di berkas ini dan lihat perubahannya langsung terlihat di browser tanpa refresh manual.

Langkah 9: Menginstal Alpine.js

Buat branch baru, misalnya `install-alpine`:

```
npm install alpinejs
```

Ganti seluruh isi `resources/js/app.js`:

```
import './bootstrap';
import Alpine from 'alpinejs';

window.Alpine = Alpine;
Alpine.start();
```

```
git add .
git commit -m "instal alpine.js"
git push -u origin install-alpine
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main`, pull, **dan jalankan `npm install` lagi** karena `package.json` ikut berubah.

> ✅ **Checkpoint:** buka console browser (F12) di halaman `/pos`, ketik `Alpine.version`, dan pastikan muncul nomor versi.

> ⚠ **Jika gagal:** error `Alpine is not defined` di console setelah pull biasanya berarti kamu lupa menjalankan `npm install` ulang, atau `npm run dev` belum di-restart.

Langkah 10: Membangun keranjang dinamis

Buat branch baru, misalnya `alpine-cart`, lalu ubah `resources/views/pos/create.blade.php` menjadi:

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

@section('title', 'Kasir')

@section('content')
    <h1 class="text-lg font-semibold mb-4">Transaksi Kasir</h1>
    <div x-data="{
        cart: [],
        addToCart(id, name, price) {
            this.cart.push({ id, name, price });
        },
        subtotal() {
            return this.cart.reduce((sum, item) => sum + item.price, 0);
        }
    }">
        <div class="grid grid-cols-3 gap-4">
            @foreach ($products as $product)
                <div class="border rounded-md p-3 cursor-pointer"
                     @click="addToCart({{ $product->id }}, '{{ $product->name }}', {{
                      ↪  $product->price }})">
                    <p class="font-medium">{{ $product->name }}</p>
                    <p class="text-sm text-slate-500">Rp {{
                     ↪  number_format($product->price) }}</p>
                </div>
            @endforeach
        </div>

        <div class="mt-4 border-t pt-3">
            <template x-for="item in cart" :key="item.id">
                <p x-text="item.name + ' - Rp ' + item.price"></p>
            </template>
            <p class="font-semibold mt-2">Subtotal: Rp <span
             ↪  x-text="subtotal()"></span></p>
        </div>
    </div>
@endsection
```

```
git add .
git commit -m "tambah keranjang dinamis dengan alpine"
git push -u origin alpine-cart
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✅ **Checkpoint:** mengklik salah satu kartu produk langsung menambahkan namanya ke area ringkasan di bawah grid dan subtotal bertambah, tanpa halaman ikut memuat ulang (perhatikan address bar dan favicon tab, keduanya tidak berkedip).

> ⚠ **Jika gagal**, periksa tiga hal paling umum: (a) console browser menampilkan `Alpine is not defined`, berarti `npm run dev` tidak berjalan atau Langkah 9 belum tersimpan; (b) halaman menampilkan `ViteManifestNotFoundException`, berarti `npm run dev` belum dijalankan sama sekali; (c) pastikan seluruh grid produk dan area ringkasan benar-benar berada di dalam elemen yang membawa `x-data`, bukan jadi elemen bertetangga di luarnya.

Langkah 11: Bersama, uji integrasi dan review kode

Semua anggota `git checkout main && git pull`, jalankan kedua server, dan ulangi uji klik dari Langkah 10 di laptop masing-masing. Setelah itu, kelompok membaca kode bersama: setiap anggota menjelaskan satu berkas yang **bukan** ia tulis sendiri.

> ✅ **Checkpoint:** halaman `/pos` berperilaku sama persis di setiap laptop anggota.

Langkah 12: Tantangan mandiri kelompok dan commit `increment 3`

Bagi tugas berikut di antara anggota, supaya setiap anggota tercatat minimal satu commit bermakna lewat Pull Request masing-masing:

- Tambahkan tombol *Hapus* pada tiap baris item di keranjang, memanggil method Alpine baru yang mengeluarkan item itu dari array `cart` berdasarkan `id`-nya.
- Tambahkan badge kecil bertuliskan "Stok Menipis" memakai class Tailwind `bg-amber-100 text-amber-700`, muncul hanya ketika `stock` produk di bawah 10.
- Tambahkan highlight (misalnya class Tailwind `ring-2 ring-blue-500`) pada kartu produk yang baru saja diklik.
- Rapikan tampilan nav: beri jarak antar link, dan tandai link halaman yang sedang aktif.
- Tambahkan `@section('title', 'Riwayat Transaksi')` yang berbeda untuk method `index`, dengan view stub baru (`php artisan make:view transactions.index`) di `resources/views/transactions/index.blade.php`.

Setiap tugas: buat branch baru (misalnya `remove-cart-item`), kerjakan, commit, push, lalu buka dan merge Pull Request-nya (ikuti alur Langkah 3).

Setelah semua Pull Request masuk dan digabungkan, salah satu anggota menutup pekerjaan kelompok:

```
git checkout main
git pull
git commit --allow-empty -m "increment 3: tampilan kasir dengan blade, tailwind, dan
↪  alpine"
git push
git log --pretty="%h %an %s"
```

> ✅ **Checkpoint:** baris teratas `git log --pretty="%h %an %s"` menunjukkan `increment 3: ...`, dan baris-baris di bawahnya menunjukkan nama setiap anggota kelompok.

D. Tugas dan Deliverable

Kumpulkan hal berikut sesuai format yang diminta dosen:

- Link repositori GitHub kelompok.
- Screenshot halaman `/pos` dengan minimal 2 item di keranjang dan subtotal terisi.
- Output `git log --pretty="%h %an %s"` yang menunjukkan minimal satu commit per anggota.
- Tabel pembagian tugas: nama anggota | langkah/tugas yang dikerjakan | hash commit.
- **Tugas mandiri (dikerjakan dan dikumpulkan masing-masing anggota):** jelaskan dengan kata-katamu sendiri, dalam 3-5 kalimat: (a) kenapa `{{ }}` lebih aman dipakai untuk menampilkan nama produk dibanding `{!! !!}`, dan (b) kenapa subtotal yang dihitung Alpine.js di keranjang tidak boleh langsung dipercaya sebagai total transaksi final oleh server.

E. Kriteria Penilaian

Komponen	Bobot	Kriteria Lengkap (100%)	Kriteria Minimum
Langkah kerja tuntas (kelompok)	30%	Langkah 1-12 selesai, `/pos` berfungsi dengan keranjang dinamis	Sebagian besar langkah selesai, halaman kasir tampil
Checkpoint terverifikasi (kelompok)	20%	Screenshot, tabel pembagian tugas, dan git log lengkap dan benar	Sebagian checkpoint terbukti
Kontribusi per anggota (individu)	25%	Minimal satu commit bermakna atas nama tiap anggota, sesuai tabel pembagian tugas	Commit ada tapi kecil atau kurang jelas kaitannya
Tugas mandiri (individu)	15%	Kedua penjelasan tepat dan berdiri sendiri	Jawaban ada meski belum lengkap
Kerapian repositori dan commit	10%	Pesan `increment 3` persis, tanpa menyertakan `vendor/`/`node_modules`/`.env`, PR di-merge rapi (bukan squash)	Commit ada, pesan kurang rapi atau PR di-squash