BAHAN AJAR TEORI DAN PRAKTIK

PEMROGRAMAN WEB

Menggunakan Framework Laravel

laravel

Studi Kasus:
Perancangan dan Pembangunan Aplikasi Web Retail
(Dilengkapi Praktik Back-End dan Front-End)

Disusun oleh:
Nurhasan Nugroho, ST., M.Kom.

- Konsep & Teori Pemrograman Web
- Framework Laravel dari Dasar hingga Mahir
- Studi Kasus Aplikasi Web Retail
- Praktik Lengkap Back-End dan Front-End

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
UNIVERSITAS BINA BANGSA

2026

BAHAN AJAR TEORI DAN PRAKTIK

PEMROGRAMAN WEB

Menggunakan Framework Laravel

Studi Kasus: Perancangan dan Pembangunan Aplikasi Web Retail

(Dilengkapi Praktik Back-End dan Front-End)

Disusun oleh:

Nurhasan Nugroho, ST., M.Kom.

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
UNIVERSITAS BINA BANGSA
2026

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa atas selesainya penyusunan bahan ajar "Pemrograman Web Menggunakan Framework Laravel" ini. Modul ini disusun sebagai panduan teori sekaligus praktik bagi mahasiswa Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Bina Bangsa, dalam menempuh mata kuliah Pemrograman Web.

Perkembangan teknologi web yang sangat pesat menuntut proses pembelajaran yang tidak hanya berhenti pada penguasaan konsep teoritis, tetapi juga kemampuan penerapan secara praktis pada studi kasus nyata. Oleh karena itu, modul ini dirancang dengan pendekatan studi kasus utuh, yaitu pembangunan aplikasi Web Retail, mulai dari perancangan basis data, implementasi back-end menggunakan pola Model-View-Controller, hingga pembangunan antarmuka front-end yang responsif.

Modul ini disusun secara sistematis dalam enam belas bab yang saling berkesinambungan, dilengkapi dengan contoh kode program, ilustrasi diagram alur proses, tabel rangkuman, serta latihan pada setiap akhir bab untuk menguji pemahaman mahasiswa. Diharapkan modul ini dapat menjadi rujukan yang aplikatif dan mudah diikuti, baik dalam kegiatan perkuliahan tatap muka maupun sebagai bahan belajar mandiri.

Penulis menyadari bahwa modul ini masih memiliki keterbatasan dan jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun dari pembaca, rekan sejawat dosen, maupun mahasiswa sangat penulis harapkan demi penyempurnaan pada edisi selanjutnya. Akhir kata, penulis mengucapkan terima kasih kepada seluruh pihak yang telah mendukung penyusunan bahan ajar ini.

Serang, 2026
Penulis,

Nurhasan Nugroho, ST., M.Kom.

TINJAUAN MATA KULIAH

Deskripsi Mata Kuliah

Mata kuliah Pemrograman Web merupakan mata kuliah wajib pada Program Studi Ilmu Komputer yang membekali mahasiswa dengan kemampuan merancang dan membangun aplikasi web dinamis menggunakan framework modern. Mata kuliah ini menekankan keseimbangan antara pemahaman konsep teoritis rekayasa perangkat lunak berbasis web dan keterampilan praktis dalam mengimplementasikan aplikasi nyata menggunakan Laravel, salah satu framework PHP yang paling banyak digunakan di industri saat ini.

Melalui studi kasus pembangunan aplikasi Web Retail, mahasiswa akan mengalami secara langsung siklus pengembangan perangkat lunak mulai dari analisis kebutuhan, perancangan basis data, implementasi back-end dan front-end, hingga pengujian dan persiapan deployment. Pendekatan berbasis studi kasus ini dipilih karena merepresentasikan kompleksitas nyata yang akan dihadapi mahasiswa ketika bekerja sebagai pengembang perangkat lunak profesional.

Prasyarat Mata Kuliah

- Telah menempuh mata kuliah Algoritma dan Pemrograman.
- Telah menempuh atau sedang menempuh mata kuliah Basis Data.
- Memahami dasar bahasa pemrograman PHP dan HTML/CSS.

Petunjuk Penggunaan Modul

Modul ini disusun agar dapat dipelajari secara berurutan dari Bab 1 hingga Bab 16, karena setiap bab dibangun di atas pemahaman bab sebelumnya. Setiap bab memiliki struktur baku yang terdiri atas uraian teori, contoh kode praktik, ilustrasi diagram (bila relevan), kotak rangkuman, serta latihan pada bagian akhir. Mahasiswa disarankan untuk:

- Membaca bagian teori terlebih dahulu sebelum mempraktikkan kode program.

- Mengetikkan ulang contoh kode secara mandiri, bukan hanya menyalin-tempel, agar terbentuk pemahaman struktur kode yang mendalam.
- Mengerjakan latihan pada setiap akhir bab sebagai bahan evaluasi mandiri.
- Merujuk pada Lampiran C untuk melihat kode sumber lengkap apabila mengalami kesulitan pada saat praktik.
- Menggunakan Lampiran A (Glosarium) sebagai rujukan cepat istilah teknis yang digunakan sepanjang modul.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah mendorong transformasi besar dalam cara manusia melakukan transaksi jual beli. Bisnis retail yang semula dijalankan secara konvensional di toko fisik kini bergeser ke arah digital melalui platform e-commerce dan aplikasi web retail. Kondisi ini menuntut lulusan Program Studi Ilmu Komputer untuk memiliki kompetensi dalam merancang, membangun, dan mengelola aplikasi web yang andal, aman, dan mudah dikembangkan.

Pemrograman web modern tidak lagi cukup dikerjakan dengan pendekatan prosedural sederhana menggunakan PHP native. Kompleksitas kebutuhan bisnis retail seperti manajemen produk, keranjang belanja, transaksi, pembayaran, hingga pelaporan penjualan memerlukan arsitektur perangkat lunak yang terstruktur. Oleh karena itu, framework berbasis pola desain Model-View-Controller (MVC) seperti Laravel menjadi pilihan utama di industri karena menyediakan struktur kode yang rapi, fitur bawaan yang lengkap, serta ekosistem yang matang.

Modul bahan ajar ini disusun sebagai panduan teori sekaligus praktik bagi mahasiswa Fakultas Ilmu Komputer, Program Studi Ilmu Komputer, Universitas Bina Bangsa, dalam mata kuliah Pemrograman Web. Modul ini memadukan konsep dasar pemrograman web, teori framework Laravel, serta studi kasus nyata berupa pembangunan aplikasi Web Retail secara bertahap dari sisi back-end maupun front-end.

1.2 Tujuan Pembelajaran

Setelah mempelajari modul ini secara menyeluruh, mahasiswa diharapkan mampu:

1. Memahami konsep dasar arsitektur aplikasi web, meliputi client-server, protokol HTTP, serta perbedaan front-end dan back-end.
2. Memahami konsep framework Laravel beserta pola arsitektur Model-View-Controller (MVC).
3. Melakukan instalasi dan konfigurasi lingkungan pengembangan Laravel secara mandiri.

4. Mengimplementasikan routing, controller, model, dan migration dalam Laravel.
5. Merancang dan mengimplementasikan basis data relasional menggunakan Eloquent ORM.
6. Membangun antarmuka pengguna (front-end) menggunakan Blade Templating Engine yang diintegrasikan dengan Bootstrap/Tailwind CSS.
7. Mengimplementasikan sistem autentikasi dan otorisasi pengguna (admin dan customer).
8. Membangun studi kasus utuh aplikasi Web Retail meliputi modul produk, keranjang belanja, checkout, transaksi, dan dashboard admin.
9. Menerapkan konsep keamanan dasar dan optimasi performa aplikasi web sebelum proses deployment.

1.3 Capaian Pembelajaran Mata Kuliah (CPMK)

Kode	Capaian Pembelajaran
CPMK-1	Mahasiswa mampu menjelaskan konsep dasar dan arsitektur pemrograman web.
CPMK-2	Mahasiswa mampu menerapkan pola MVC menggunakan framework Laravel.
CPMK-3	Mahasiswa mampu merancang basis data dan mengimplementasikannya dengan migration dan Eloquent ORM.
CPMK-4	Mahasiswa mampu membangun antarmuka web yang responsif menggunakan Blade dan framework CSS.
CPMK-5	Mahasiswa mampu mengimplementasikan sistem autentikasi, transaksi, dan pelaporan pada studi kasus retail.
CPMK-6	Mahasiswa mampu menerapkan prinsip keamanan dan optimasi pada aplikasi web sebelum dipublikasikan.

Tabel 7. Capaian Pembelajaran Mata Kuliah Pemrograman Web

1.4 Peta Konsep Modul

Modul ini disusun secara sistematis dari konsep dasar menuju implementasi studi kasus yang utuh. Alur pembahasan dibagi menjadi empat kelompok besar sebagaimana disajikan pada Tabel 2.

Kelompok	Cakupan Bab	Fokus Pembahasan
I. Konsep Dasar	Bab 1 - Bab 4	Pendahuluan, konsep web, pengenalan Laravel, instalasi
II. Fondasi Back-End	Bab 5 - Bab 8	Routing, controller, model/Eloquent, migration & database
III. Front-End & Integrasi	Bab 9 - Bab 10	Blade templating, integrasi Bootstrap/Tailwind
IV. Studi Kasus Web Retail	Bab 11 - Bab 16	CRUD produk, autentikasi, transaksi, dashboard, deployment, penutup

Tabel 8. Peta Konsep dan Struktur Modul Bahan Ajar

Catatan Penggunaan Modul

Setiap bab pada modul ini memuat bagian Teori, Praktik (disertai kode program dan gambar ilustrasi alur proses), Rangkuman, serta Latihan/Tugas. Mahasiswa disarankan mempraktikkan setiap contoh kode secara langsung pada lingkungan pengembangan masing-masing agar pemahaman konsep lebih mendalam.

BAB 2 KONSEP DASAR PEMROGRAMAN WEB

2.1 Arsitektur Client-Server

Aplikasi web pada dasarnya dibangun di atas arsitektur client-server, yaitu model komputasi yang membagi tugas antara penyedia layanan (server) dan pengguna layanan (client). Client, umumnya berupa peramban web (browser), mengirimkan permintaan (request) kepada server. Server kemudian memproses permintaan tersebut, menjalankan logika aplikasi, mengakses basis data bila diperlukan, dan mengembalikan tanggapan (response) berupa dokumen HTML, data JSON, atau berkas lain kepada client.

Pemisahan tugas ini memberikan sejumlah keuntungan, di antaranya kemudahan pemeliharaan karena logika bisnis terpusat di server, kemampuan melayani banyak client secara bersamaan, serta fleksibilitas dalam mengganti tampilan (client) tanpa mengubah logika inti aplikasi. Dalam konteks aplikasi Web Retail, server bertanggung jawab menyimpan data produk, memproses transaksi, dan menjaga konsistensi stok barang, sedangkan client menampilkan katalog produk dan formulir transaksi kepada pengguna.

2.2 Protokol HTTP dan HTTPS

Hypertext Transfer Protocol (HTTP) merupakan protokol standar yang digunakan untuk pertukaran data antara client dan server di web. HTTP bersifat stateless, artinya setiap permintaan diperlakukan secara independen tanpa mengingat riwayat permintaan sebelumnya. Untuk menjaga status pengguna, seperti sesi login, aplikasi web memanfaatkan mekanisme tambahan seperti cookie dan session.

HTTP Secure (HTTPS) merupakan pengembangan dari HTTP yang menambahkan lapisan enkripsi Transport Layer Security (TLS) sehingga data yang dipertukarkan antara client dan server tidak dapat dibaca oleh pihak ketiga. Pada aplikasi retail yang menangani data pembayaran dan informasi pribadi pelanggan, penggunaan HTTPS bersifat wajib demi menjaga kerahasiaan dan integritas data.

Metode HTTP	Kegunaan Umum	Contoh Penerapan pada Retail
GET	Mengambil data dari server	Menampilkan daftar produk

Metode HTTP	Kegunaan Umum	Contoh Penerapan pada Retail
POST	Mengirim data baru ke server	Menyimpan produk baru / membuat order
PUT/PATCH	Memperbarui data yang sudah ada	Mengubah data produk atau status order
DELETE	Menghapus data pada server	Menghapus produk dari katalog

Tabel 9. Metode HTTP dan Contoh Penerapannya pada Aplikasi Retail

2.3 Front-End dan Back-End

Front-end merupakan bagian aplikasi web yang berinteraksi langsung dengan pengguna, mencakup tampilan antarmuka (HTML, CSS), interaktivitas (JavaScript), serta pengalaman pengguna (user experience). Back-end merupakan bagian yang berjalan di sisi server, mencakup logika bisnis, pengelolaan basis data, autentikasi, serta penyediaan data kepada front-end.

Dalam pengembangan aplikasi Web Retail menggunakan Laravel, back-end diimplementasikan melalui Controller dan Model yang menangani logika seperti perhitungan total belanja, validasi stok, dan pemrosesan pembayaran. Front-end diimplementasikan melalui Blade Template yang dipadukan dengan framework CSS seperti Bootstrap atau Tailwind untuk menghasilkan tampilan katalog produk, keranjang belanja, dan dashboard admin yang responsif.

2.4 Web Statis dan Web Dinamis

Website statis menyajikan konten yang sama kepada setiap pengunjung karena berkas HTML tidak berubah kecuali diedit langsung oleh pengembang. Sebaliknya, website dinamis menghasilkan konten secara otomatis berdasarkan data yang tersimpan di basis data dan logika program, sehingga tampilan dapat berbeda-beda sesuai konteks, misalnya daftar produk yang berubah sesuai stok terkini atau riwayat transaksi yang unik bagi setiap pengguna.

Aplikasi Web Retail merupakan contoh nyata website dinamis, karena informasi produk, harga, stok, dan status pesanan bersifat dinamis dan harus selalu mencerminkan kondisi basis data terbaru. Framework Laravel dirancang khusus untuk mempermudah pembangunan website dinamis melalui integrasi yang erat antara logika back-end dan basis data melalui Eloquent ORM.

2.5 Arsitektur Three-Tier pada Aplikasi Web

Pengembangan lebih lanjut dari konsep client-server adalah arsitektur three-tier (tiga lapis), yang membagi aplikasi menjadi lapisan presentasi (presentation tier), lapisan logika aplikasi (application/logic tier), dan lapisan data (data tier). Pendekatan ini memisahkan tanggung jawab secara lebih granular dibandingkan arsitektur client-server dua lapis sederhana, dan menjadi dasar konseptual dari pola MVC yang diterapkan Laravel.

Lapisan (Tier)	Tanggung Jawab	Contoh Komponen pada Web Retail
Presentation Tier	Menampilkan informasi dan menerima interaksi pengguna	Blade Template, Bootstrap/Tailwind CSS
Application/Logic Tier	Memproses logika bisnis dan aturan aplikasi	Controller dan Service pada Laravel
Data Tier	Menyimpan dan mengelola data secara permanen	Basis data MySQL yang diakses melalui Eloquent ORM

Tabel 10. Pembagian Tanggung Jawab pada Arsitektur Three-Tier

Pemisahan tiga lapis ini memberikan fleksibilitas tinggi dalam pengembangan aplikasi skala besar, karena setiap lapisan dapat dikembangkan, diuji, dan diskalakan (scaling) secara independen. Sebagai contoh, apabila jumlah pengunjung toko retail meningkat drastis, lapisan data dapat ditingkatkan kapasitasnya (misalnya melalui replikasi basis data) tanpa harus mengubah logika aplikasi maupun tampilan antarmuka.

2.6 Konsep RESTful dalam Perancangan Web

Representational State Transfer (REST) merupakan gaya arsitektur yang banyak diadopsi dalam perancangan aplikasi web modern, termasuk oleh Laravel melalui konvensi Resource Controller yang telah disinggung sekilas pada bab-bab berikutnya. Prinsip REST menekankan representasi sumber daya (resource) melalui URL yang konsisten dan pemanfaatan metode HTTP secara semantik sesuai jenis operasi yang dilakukan terhadap sumber daya tersebut.

Pada aplikasi Web Retail, resource seperti produk, kategori, dan order direpresentasikan secara RESTful, misalnya /produk untuk kumpulan data dan /produk/{id} untuk satu data spesifik, dengan metode GET untuk membaca, POST

untuk membuat, PUT/PATCH untuk memperbarui, dan DELETE untuk menghapus. Konsistensi ini memudahkan pengembang lain memahami struktur API maupun rute aplikasi tanpa dokumentasi tambahan yang rumit.

Rangkuman Bab 2

Aplikasi web bekerja berdasarkan arsitektur client-server yang berkomunikasi melalui protokol HTTP/HTTPS. Front-end menangani presentasi dan interaksi pengguna, sedangkan back-end menangani logika bisnis dan pengelolaan data. Website dinamis, seperti aplikasi Web Retail, membutuhkan keterpaduan front-end dan back-end yang kuat, yang dapat dicapai secara efisien menggunakan framework seperti Laravel.

Latihan Bab 2

10. Jelaskan perbedaan mendasar antara arsitektur client-server dengan arsitektur peer-to-peer.
11. Mengapa protokol HTTPS wajib digunakan pada aplikasi yang menangani transaksi pembayaran?
12. Berikan tiga contoh elemen front-end dan tiga contoh elemen back-end pada aplikasi Web Retail.
13. Jelaskan mengapa katalog produk pada aplikasi retail harus bersifat dinamis, bukan statis.

BAB 3 PENGENALAN FRAMEWORK LARAVEL

3.1 Sejarah dan Perkembangan Laravel

Laravel merupakan framework pengembangan aplikasi web berbasis bahasa pemrograman PHP yang pertama kali dirilis oleh Taylor Otwell pada tahun 2011. Laravel hadir sebagai jawaban atas keterbatasan framework PHP sebelumnya yang dinilai kurang ekspresif dan sulit dipelajari. Sejak dirilis, Laravel berkembang pesat dan menjadi salah satu framework PHP paling populer di dunia karena filosofi desainnya yang mengutamakan sintaksis elegan (elegant syntax) dan kemudahan pengembang (developer experience).

Laravel mengadopsi banyak konsep pemrograman modern seperti dependency injection, service container, dan pengujian otomatis (automated testing) yang menjadikannya cocok untuk pengembangan aplikasi berskala kecil hingga enterprise, termasuk aplikasi e-commerce dan retail berskala besar.

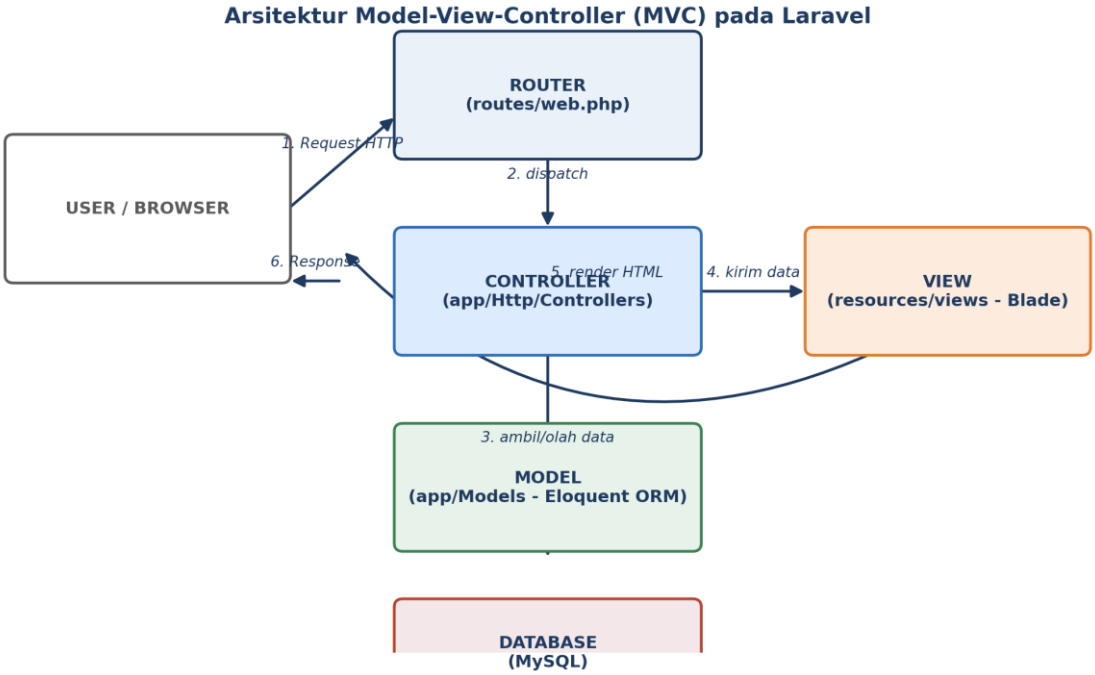
3.2 Konsep MVC (Model-View-Controller)

Model-View-Controller (MVC) adalah pola arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama yang saling berkaitan namun memiliki tanggung jawab berbeda:

- Model — bertanggung jawab atas representasi data dan logika bisnis, termasuk interaksi dengan basis data melalui Eloquent ORM.
- View — bertanggung jawab atas presentasi/tampilan data kepada pengguna, diimplementasikan menggunakan Blade Template Engine.
- Controller — bertindak sebagai penghubung antara Model dan View, menerima permintaan dari pengguna, memproses logika, lalu menentukan tampilan yang akan dikembalikan.

Pemisahan tanggung jawab pada pola MVC memberikan sejumlah manfaat penting dalam pengembangan aplikasi Web Retail, di antaranya kemudahan pemeliharaan kode karena setiap komponen dapat dikembangkan dan diuji secara terpisah, peningkatan keterbacaan kode bagi tim pengembang, serta kemudahan penggunaan

ulang (reusability) komponen, misalnya satu Model Produk dapat digunakan oleh beberapa Controller yang berbeda.

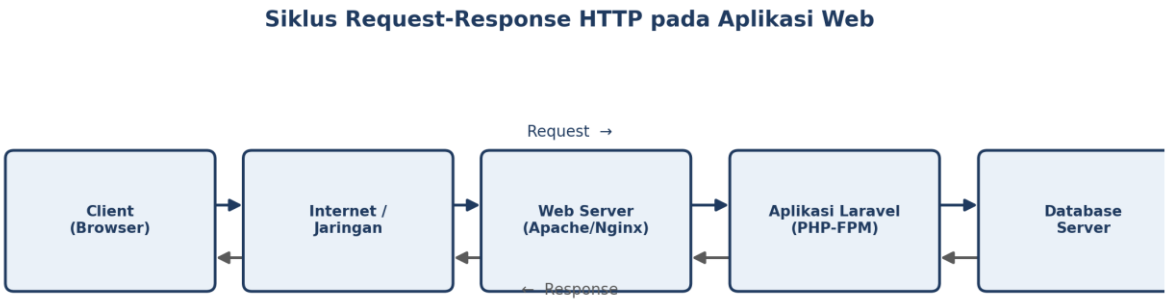


Gambar 3. Arsitektur Model-View-Controller (MVC) pada Laravel

Sebagaimana ditunjukkan pada Gambar 1, alur kerja MVC diawali ketika pengguna mengirimkan permintaan HTTP melalui browser. Permintaan tersebut diterima oleh Router yang menentukan Controller mana yang akan menanganinya. Controller kemudian berinteraksi dengan Model untuk mengambil atau mengubah data pada basis data, lalu mengirimkan data tersebut ke View untuk dirender menjadi halaman HTML yang dikembalikan sebagai response kepada pengguna.

3.3 Siklus Request-Response pada Laravel

Setiap permintaan yang masuk ke aplikasi Laravel melewati serangkaian tahapan yang disebut siklus request-response (request lifecycle). Permintaan dari client pertama kali diterima oleh web server, kemudian diteruskan ke aplikasi Laravel melalui berkas public/index.php, diproses oleh kernel HTTP, melewati middleware, dirutekan ke controller yang sesuai, hingga akhirnya menghasilkan response yang dikembalikan ke client.



Gambar 4. Siklus Request-Response HTTP pada Aplikasi Web

3.4 Fitur Unggulan Laravel

Fitur	Deskripsi Singkat
Eloquent ORM	Object Relational Mapping untuk berinteraksi dengan basis data menggunakan sintaksis PHP tanpa menulis SQL manual
Blade Templating	Mesin template ringan dengan directive seperti @if, @foreach, dan pewarisan layout
Artisan CLI	Command Line Interface untuk mengotomatisasi tugas pengembangan seperti membuat controller, model, dan migration
Migration & Seeder	Version control untuk struktur basis data serta pengisian data awal secara terprogram
Routing Ekspresif	Definisi rute aplikasi yang ringkas dan mudah dibaca, mendukung route parameter dan grup middleware
Middleware	Mekanisme penyaringan HTTP request, misalnya untuk autentikasi dan otorisasi
Autentikasi Bawaan	Paket seperti Laravel Breeze dan Fortify untuk implementasi login, register, dan reset password secara cepat
Ekosistem Luas	Didukung banyak paket resmi seperti Laravel Sanctum, Cashier, dan Livewire

Tabel 11. Fitur Unggulan Framework Laravel

3.5 Kebutuhan Sistem dan Persiapan Lingkungan Pengembangan

Sebelum melakukan instalasi Laravel, terdapat sejumlah perangkat lunak pendukung yang perlu dipersiapkan pada komputer pengembang. Kebutuhan minimum tersebut disajikan pada Tabel berikut.

Komponen	Versi Minimum yang Direkomendasikan	Fungsi
PHP	8.1 ke atas	Bahasa pemrograman inti yang menjalankan Laravel
Composer	2.x	Manajer dependensi/paket PHP
Node.js & NPM	18.x ke atas	Kompilasi aset front-end (CSS/JS) melalui Vite
MySQL / MariaDB	8.x / 10.x	Sistem manajemen basis data relasional
Web Server Lokal	Laragon / XAMPP / Herd	Lingkungan server lokal terintegrasi (Apache/Nginx, PHP, MySQL)
Code Editor	Visual Studio Code	Editor kode dengan dukungan ekstensi PHP dan Laravel

Tabel 12. Kebutuhan Perangkat Lunak untuk Pengembangan Laravel

3.6 Service Container dan Service Provider

Service Container adalah salah satu fitur inti Laravel yang berfungsi sebagai wadah untuk mengelola pembuatan objek beserta dependensinya secara otomatis (dependency injection). Dengan Service Container, pengembang tidak perlu membuat objek secara manual menggunakan kata kunci new pada setiap tempat objek tersebut dibutuhkan, melainkan cukup mendeklarasikan kebutuhan tersebut pada constructor atau method, dan Laravel akan menyediakannya secara otomatis.

Service Provider merupakan tempat terpusat untuk mengonfigurasi dan mendaftarkan berbagai layanan (service) ke dalam Service Container, termasuk binding antarmuka (interface) dengan implementasi konkretnya. Seluruh Service Provider aplikasi didaftarkan pada berkas bootstrap/providers.php (Laravel 11) atau config/app.php pada versi sebelumnya.

```
// Contoh binding interface dengan implementasi pada Service Provider
public function register(): void
{
    $this->app->bind(
        PaymentGatewayInterface::class,
```

```
        MidtransPaymentGateway::class
    );
}
```

3.7 Perbandingan Laravel dengan Framework PHP Lain

Sebelum memutuskan menggunakan Laravel sebagai framework utama dalam pengembangan aplikasi Web Retail, penting untuk memahami posisinya dibandingkan framework PHP populer lainnya, agar pemilihan teknologi didasari pertimbangan yang tepat.

Framework	Karakteristik Utama	Kurva Belajar
Laravel	Ekspresif, ekosistem lengkap (Breeze, Sanctum, Cashier), dokumentasi sangat baik	Sedang
CodeIgniter	Ringan, konfigurasi minimal, cocok untuk aplikasi kecil-menengah	Rendah
Symfony	Sangat fleksibel dan modular, sering menjadi fondasi framework lain termasuk Laravel	Tinggi
CakePHP	Konvensi kuat mirip Ruby on Rails, cepat untuk prototyping	Sedang

Tabel 13. Perbandingan Karakteristik Laravel dengan Framework PHP Lainnya

Rangkuman Bab 3

Laravel adalah framework PHP yang mengimplementasikan pola MVC untuk memisahkan logika data (Model), tampilan (View), dan pengendali alur aplikasi (Controller). Pemahaman terhadap siklus request-response dan fitur-fitur bawaan Laravel menjadi dasar penting sebelum masuk ke tahap implementasi teknis pada bab-bab selanjutnya.

Latihan Bab 3

14. Jelaskan peran masing-masing komponen Model, View, dan Controller dalam konteks aplikasi Web Retail.
15. Mengapa pemisahan logika bisnis dan tampilan penting dalam pengembangan aplikasi berskala besar?

16. Sebutkan minimal lima fitur unggulan Laravel dan jelaskan manfaatnya masing-masing.
17. Jelaskan secara singkat urutan siklus request-response ketika pengguna membuka halaman katalog produk.

BAB 4 INSTALASI DAN STRUKTUR PROYEK LARAVEL

4.1 Instalasi Laravel melalui Composer

Laravel didistribusikan sebagai paket PHP yang dikelola melalui Composer, sehingga proses instalasi dilakukan melalui command line. Terdapat dua pendekatan umum untuk membuat proyek Laravel baru, yaitu menggunakan Laravel Installer atau langsung melalui perintah composer create-project. Berikut adalah langkah-langkah instalasi proyek Laravel yang akan digunakan sebagai dasar pembangunan aplikasi Web Retail pada modul ini.

Langkah 1 — Memeriksa Versi PHP dan Composer

```
php -v
composer -V
```

Langkah 2 — Membuat Proyek Laravel Baru

```
composer create-project laravel/laravel web-retail
cd web-retail
```

Perintah di atas akan mengunduh Laravel beserta seluruh dependensinya dan membuat direktori proyek bernama web-retail yang akan digunakan sebagai studi kasus utama sepanjang modul ini.

Langkah 3 — Konfigurasi Berkas .env

Berkas .env menyimpan variabel lingkungan (environment variables) aplikasi, termasuk konfigurasi koneksi basis data. Konfigurasi ini wajib disesuaikan agar Laravel dapat terhubung dengan basis data MySQL yang akan digunakan untuk menyimpan data retail.

```
APP_NAME="Web Retail Binabangsa"
APP_ENV=local
APP_URL=http://localhost:8000

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=web_retail
DB_USERNAME=root
DB_PASSWORD=
```

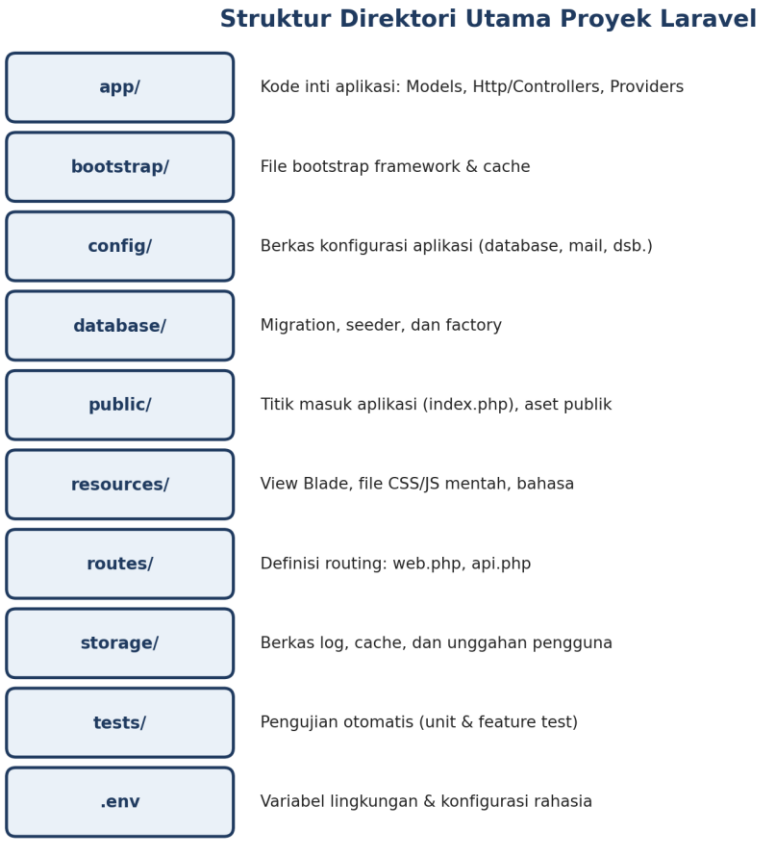
Langkah 4 — Menjalankan Server Pengembangan

Laravel menyediakan server pengembangan bawaan melalui perintah artisan serve sehingga pengembang tidak perlu mengonfigurasi Apache/Nginx secara manual pada tahap awal pengembangan.

```
php artisan serve
# Server berjalan pada http://127.0.0.1:8000
```

4.2 Struktur Direktori Proyek Laravel

Memahami struktur direktori Laravel merupakan langkah penting sebelum mulai menulis kode program, karena Laravel menerapkan konvensi penamaan dan penempatan berkas yang konsisten (convention over configuration).



Gambar 5. Struktur Direktori Utama Proyek Laravel

Direktori app/ merupakan pusat logika aplikasi yang paling sering diakses oleh pengembang, terutama subdirektori Http/Controllers untuk menyimpan seluruh Controller dan direktori Models untuk menyimpan seluruh kelas Model Eloquent. Direktori routes/ menyimpan berkas web.php yang mendefinisikan seluruh rute aplikasi berbasis browser, sedangkan direktori resources/views menyimpan seluruh berkas Blade Template yang akan menampilkan antarmuka aplikasi Web Retail.

4.3 Konfigurasi Basis Data dan Pembuatan Database

Sebelum menjalankan migration pada bab selanjutnya, basis data dengan nama web_retail perlu dibuat terlebih dahulu melalui phpMyAdmin atau command line MySQL.

```
mysql -u root -p
CREATE DATABASE web_retail CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

4.4 Perintah Artisan yang Sering Digunakan

Perintah Artisan	Fungsi
php artisan serve	Menjalankan server pengembangan lokal
php artisan make:controller NamaController	Membuat berkas Controller baru
php artisan make:model NamaModel -m	Membuat Model beserta berkas Migration terkait
php artisan migrate	Menjalankan seluruh migration ke basis data
php artisan migrate:rollback	Membatalkan migration terakhir
php artisan make:seeder NamaSeeder	Membuat berkas Seeder untuk pengisian data awal
php artisan route:list	Menampilkan seluruh rute yang terdaftar pada aplikasi
php artisan config:clear	Membersihkan cache konfigurasi aplikasi

Tabel 14. Perintah Artisan CLI yang Sering Digunakan dalam Pengembangan

Rangkuman Bab 4

Instalasi Laravel dilakukan melalui Composer dan dikonfigurasi melalui berkas .env, terutama untuk koneksi basis data. Struktur direktori Laravel mengikuti konvensi yang konsisten sehingga memudahkan navigasi kode, dengan direktori app/, routes/, dan resources/views sebagai tiga lokasi yang paling sering diakses selama pengembangan aplikasi Web Retail.

Latihan Bab 4

18. Lakukan instalasi proyek Laravel baru pada komputer masing-masing dengan nama web-retail.
19. Konfigurasikan berkas .env agar terhubung dengan basis data bernama web_retail.
20. Jalankan server pengembangan dan tangkap layar (screenshot) halaman selamat datang Laravel yang tampil di browser.
21. Jelaskan fungsi dari direktori bootstrap/ dan storage/ pada struktur proyek Laravel.

BAB 5 ROUTING PADA LARAVEL

5.1 Konsep Dasar Routing

Routing adalah mekanisme yang menghubungkan Uniform Resource Locator (URL) yang diakses oleh pengguna dengan logika program yang harus dijalankan untuk menanganinya. Pada Laravel, seluruh definisi rute untuk aplikasi berbasis browser ditulis pada berkas routes/web.php, sedangkan rute untuk keperluan Application Programming Interface (API) ditulis pada routes/api.php.

Setiap rute pada Laravel terdiri atas tiga unsur utama: metode HTTP (GET, POST, PUT, DELETE, dan sebagainya), pola URI yang akan dicocokkan, serta aksi yang akan dijalankan, baik berupa closure (fungsi anonim) maupun referensi ke method pada sebuah Controller.

5.2 Rute Dasar GET dan POST

Contoh berikut menunjukkan definisi rute dasar untuk menampilkan halaman beranda toko retail dan menangani pengiriman formulir pencarian produk.

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProductController;

// Menampilkan halaman beranda toko
Route::get('/', function () {
    return view('welcome');
});

// Menampilkan seluruh produk melalui Controller
Route::get('/produk', [ProductController::class, 'index']);

// Menangani pencarian produk (POST)
Route::post('/produk/cari', [ProductController::class, 'search']);
```

5.3 Route Parameter

Route parameter digunakan untuk menangkap segmen dinamis pada URL, misalnya identitas (ID) produk yang akan ditampilkan detailnya. Parameter dapat bersifat wajib maupun opsional, dan Laravel secara otomatis menyuntikkan nilai parameter tersebut ke dalam method Controller yang bersangkutan.

```
// Parameter wajib: menampilkan detail produk berdasarkan ID
```

```
Route::get('/produk/{id}', [ProductController::class, 'show']);

// Parameter opsional dengan nilai default
Route::get('/produk/kategori/{kategori?}', [ProductController::class,
'byCategory']);

// Constraint pada parameter, hanya menerima angka
Route::get('/produk/{id}', [ProductController::class, 'show'])-
>whereNumber('id');
```

5.4 Route Group dan Middleware

Route group digunakan untuk mengelompokkan beberapa rute yang memiliki atribut serupa, misalnya prefix URL yang sama atau middleware yang sama, sehingga kode routing menjadi lebih ringkas dan terstruktur. Middleware berfungsi sebagai lapisan penyaring yang dijalankan sebelum atau sesudah request diproses oleh Controller, misalnya untuk memastikan pengguna telah login sebelum mengakses dashboard admin.

```
// Grup rute khusus admin, dilindungi middleware auth dan role admin
Route::prefix('admin')->middleware(['auth', 'role:admin'])-
>group(function () {
    Route::get('/dashboard', [AdminController::class, 'index']);
    Route::resource('produk', ProductController::class);
    Route::resource('kategori', CategoryController::class);
});

// Grup rute untuk pelanggan yang sudah login
Route::middleware('auth')->group(function () {
    Route::get('/keranjang', [CartController::class, 'index']);
    Route::post('/checkout', [OrderController::class, 'checkout']);
});
```

5.5 Named Route

Named route memberikan nama unik pada sebuah rute sehingga dapat direferensikan pada bagian lain aplikasi, misalnya di dalam Blade Template, tanpa perlu menuliskan ulang URL secara eksplisit. Pendekatan ini mempermudah pemeliharaan aplikasi ketika struktur URL mengalami perubahan.

```
Route::get('/produk/{id}', [ProductController::class, 'show'])-
>name('produk.show');

// Penggunaan pada Blade Template
<a href="{{ route('produk.show', $produk->id) }}">Lihat Detail</a>
```

5.6 Praktik: Merancang Routing Modul Web Retail

Sebagai praktik, berikut disusun rancangan routing lengkap untuk aplikasi Web Retail yang akan diimplementasikan secara bertahap pada bab-bab selanjutnya.

Method	URI	Aksi Controller	Keterangan
GET	/	HomeController@index	Halaman beranda toko
GET	/produk	ProductController@index	Daftar seluruh produk
GET	/produk/{id}	ProductController@show	Detail satu produk
POST	/keranjang/tambah	CartController@store	Menambahkan produk ke keranjang
GET	/keranjang	CartController@index	Menampilkan isi keranjang
POST	/checkout	OrderController@checkout	Memproses checkout menjadi order
GET	/admin/produk	Admin\ProductController@index	Manajemen produk (admin)
GET	/admin/laporan	Admin\ReportController@index	Laporan penjualan (admin)

Tabel 15. Rancangan Routing Aplikasi Web Retail

Rangkuman Bab 5

Routing menghubungkan URL dengan logika aplikasi. Laravel mendukung rute dasar, route parameter, route group, middleware, dan named route yang secara bersama-sama memungkinkan penyusunan struktur navigasi aplikasi Web Retail yang rapi dan aman.

Latihan Bab 5

22. Buatlah rute untuk menampilkan halaman 'Tentang Kami' pada aplikasi Web Retail.
23. Buatlah route group untuk seluruh rute admin dengan prefix 'admin' dan middleware 'auth'.
24. Jelaskan perbedaan antara Route::get dan Route::post beserta contoh penggunaannya pada modul keranjang belanja.

25. Jelaskan manfaat named route dan berikan contoh penerapannya pada tautan 'Lihat Detail Produk'.

BAB 6 CONTROLLER PADA LARAVEL

6.1 Konsep dan Peran Controller

Controller merupakan komponen dalam pola MVC yang bertugas menerima request dari Router, memproses logika yang diperlukan (termasuk berinteraksi dengan Model), lalu menentukan View yang akan dikembalikan sebagai response. Dengan menempatkan logika pemrosesan pada Controller, kode program menjadi lebih terorganisir dibandingkan menuliskan seluruh logika langsung pada berkas routes.

6.2 Membuat Controller melalui Artisan

Laravel menyediakan perintah Artisan untuk menghasilkan berkas Controller secara otomatis beserta struktur dasarnya.

```
php artisan make:controller ProductController
php artisan make:controller CartController
php artisan make:controller OrderController
php artisan make:controller Admin/DashboardController
```

6.3 Basic Controller

Berikut adalah contoh implementasi ProductController sederhana yang menangani penampilan daftar produk dan detail produk pada aplikasi Web Retail.

```
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    // Menampilkan seluruh produk
    public function index()
    {
        $produk = Product::with('category')->latest()->paginate(12);
        return view('produk.index', compact('produk'));
    }

    // Menampilkan detail satu produk
    public function show($id)
    {
        $produk = Product::findOrFail($id);
```

```
        return view('produk.show', compact('produk'));
    }
}
```

6.4 Resource Controller

Untuk operasi Create, Read, Update, Delete (CRUD) yang bersifat berulang, Laravel menyediakan Resource Controller yang secara otomatis menghasilkan tujuh method standar (index, create, store, show, edit, update, destroy) yang sudah dipetakan pada Route::resource.

```
php artisan make:controller Admin/ProductController --resource --
model=Product
```

Method	HTTP Verb	URI	Fungsi
index	GET	/produk	Menampilkan seluruh data
create	GET	/produk/create	Menampilkan form tambah data
store	POST	/produk	Menyimpan data baru
show	GET	/produk/{id}	Menampilkan satu data
edit	GET	/produk/{id}/edit	Menampilkan form edit data
update	PUT/PATCH	/produk/{id}	Memperbarui data
destroy	DELETE	/produk/{id}	Menghapus data

Tabel 16. Tujuh Method Standar pada Resource Controller

6.5 Dependency Injection pada Controller

Laravel mendukung dependency injection melalui service container, sehingga objek seperti Request dapat langsung disuntikkan sebagai parameter method Controller tanpa perlu instansiasi manual. Pendekatan ini juga berlaku untuk Form Request yang digunakan pada validasi data.

```
public function store(Request $request)
{
    $validated = $request->validate([
        'name'        => 'required|string|max:150',
        'category_id' => 'required|exists:categories,id',
        'price'       => 'required|numeric|min:0',
```

```
        'stock'       => 'required|integer|min:0',
    ]);

    Product::create($validated);

    return redirect()->route('admin.produk.index')
                     ->with('success', 'Produk berhasil ditambahkan');
}
```

6.6 Praktik: Menyusun ProductController Lengkap untuk Web Retail

Sebagai praktik, mahasiswa diminta melengkapi ProductController agar mendukung seluruh operasi CRUD produk yang akan diintegrasikan dengan Model dan View pada bab-bab berikutnya, mengikuti struktur tujuh method standar sebagaimana dijelaskan pada Tabel 8.

6.7 Form Request untuk Validasi Terstruktur

Ketika logika validasi pada sebuah Controller menjadi kompleks, praktik terbaik yang direkomendasikan adalah memindahkannya ke kelas Form Request tersendiri. Pendekatan ini menjaga Controller tetap ringkas (thin controller) dan memungkinkan aturan validasi digunakan ulang pada beberapa method sekaligus, misalnya store dan update yang umumnya memiliki aturan validasi yang serupa.

```
class StoreProductRequest extends FormRequest
{
    public function authorize(): bool
    {
        return auth()->check() && auth()->user()->role === 'admin';
    }

    public function rules(): array
    {
        return [
            'name'        => 'required|string|max:150',
            'category_id' => 'required|exists:categories,id',
            'price'       => 'required|numeric|min:0',
            'stock'       => 'required|integer|min:0',
        ];
    }

    public function messages(): array
    {
        return [
            'name.required' => 'Nama produk wajib diisi.',
            'price.min'     => 'Harga produk tidak boleh bernilai
negatif.',
```

```
        ];
    }
}

// Penggunaan pada Controller
public function store(StoreProductRequest $request)
{
    Product::create($request->validated());
    return redirect()->route('admin.produk.index')->with('success',
'Produk tersimpan.');
}
```

6.8 Policy untuk Otorisasi Berbasis Aksi

Selain middleware berbasis peran, Laravel juga menyediakan Policy sebagai mekanisme otorisasi yang lebih granular, yaitu mengatur hak akses berdasarkan aksi spesifik terhadap sebuah model, misalnya memastikan hanya admin yang membuat produk tersebut atau admin dengan hak tertentu yang dapat menghapusnya.

```
php artisan make:policy ProductPolicy --model=Product

public function delete(User $user, Product $product): bool
{
    return $user->role === 'admin';
}

// Penggunaan pada Controller
public function destroy(Product $produk)
{
    $this->authorize('delete', $produk);
    $produk->delete();
    return back()->with('success', 'Produk dihapus.');
}
```

Rangkuman Bab 6

Controller menjembatani Router dengan Model dan View. Laravel menyediakan Basic Controller untuk kebutuhan sederhana dan Resource Controller untuk kebutuhan CRUD standar. Dependency injection memudahkan Controller mengakses objek Request dan melakukan validasi data secara ringkas.

Latihan Bab 6

26. Buatlah Resource Controller bernama CategoryController untuk mengelola data kategori produk.

27. Implementasikan method store pada CartController untuk menambahkan produk ke keranjang belanja.
28. Jelaskan perbedaan antara Basic Controller dan Resource Controller beserta kapan masing-masing sebaiknya digunakan.
29. Jelaskan konsep dependency injection dan manfaatnya pada validasi data di Controller.

BAB 7 MODEL DAN ELOQUENT ORM

7.1 Konsep Object Relational Mapping (ORM)

Object Relational Mapping (ORM) adalah teknik pemrograman yang memungkinkan pengembang berinteraksi dengan basis data relasional menggunakan objek dan sintaksis bahasa pemrograman, tanpa perlu menuliskan query SQL secara manual. Eloquent merupakan implementasi ORM bawaan Laravel yang menerapkan pola Active Record, di mana setiap Model merepresentasikan satu tabel pada basis data, dan setiap instance objek Model merepresentasikan satu baris data pada tabel tersebut.

Penggunaan Eloquent memberikan sejumlah keuntungan, antara lain kode yang lebih ringkas dan mudah dibaca, perlindungan otomatis terhadap serangan SQL Injection melalui parameter binding, serta kemudahan dalam mendefinisikan relasi antar tabel.

7.2 Membuat Model

Model dibuat menggunakan perintah Artisan dan secara konvensi dihubungkan dengan tabel pada basis data yang namanya merupakan bentuk jamak (plural) dari nama Model, misalnya Model Product terhubung dengan tabel products.

```
php artisan make:model Product -m
php artisan make:model Category -m
php artisan make:model Order -m
php artisan make:model OrderItem -m
php artisan make:model Payment -m
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Product extends Model
{
    protected $fillable = [
        'category_id', 'name', 'price', 'stock', 'image', 'description',
    ];
```

```
    public function category(): BelongsTo
    {
        return $this->belongsTo(Category::class);
    }

    public function orderItems(): HasMany
    {
        return $this->hasMany(OrderItem::class);
    }
}
```

7.3 Relasi Antar Model

Laravel Eloquent mendukung berbagai jenis relasi antar tabel yang merepresentasikan hubungan data pada dunia nyata. Pada aplikasi Web Retail, terdapat beberapa relasi penting sebagaimana dijelaskan berikut.

Jenis Relasi	Contoh pada Web Retail	Method Eloquent
One to Many	Satu Kategori memiliki banyak Produk	hasMany() / belongsTo()
One to Many	Satu Order memiliki banyak Order Item	hasMany() / belongsTo()
One to One	Satu Order memiliki satu Payment	hasOne() / belongsTo()
Many to Many	Produk dapat memiliki banyak Tag, dan Tag dapat dimiliki banyak Produk	belongsToMany()

Tabel 17. Jenis Relasi Antar Model pada Aplikasi Web Retail

```
// Relasi One to Many pada Model Category
public function products(): HasMany
{
    return $this->hasMany(Product::class);
}

// Relasi Many to Many pada Model Product
public function tags(): BelongsToMany
{
    return $this->belongsToMany(Tag::class, 'product_tag');
}
```

7.4 Query Builder dan Eloquent

Eloquent menyediakan berbagai method untuk melakukan operasi query terhadap basis data dengan sintaksis yang deklaratif dan mudah dibaca (fluent interface), sebagaimana ditunjukkan pada contoh berikut.

```
// Mengambil seluruh produk
$produk = Product::all();

// Mengambil produk dengan filter dan pengurutan
$produk = Product::where('stock', '>', 0)
                  ->where('category_id', $kategoriId)
                  ->orderBy('price', 'asc')
                  ->paginate(12);

// Eager loading relasi untuk menghindari N+1 query problem
$produk = Product::with('category')->get();

// Menyimpan data baru
Product::create([
    'category_id' => 1,
    'name'        => 'Kemeja Katun Pria',
    'price'       => 150000,
    'stock'       => 25,
]);
```

7.5 Accessor, Mutator, dan Casting

Eloquent memungkinkan transformasi data secara otomatis saat data diambil (accessor) maupun saat disimpan (mutator), serta konversi tipe data (casting) untuk memastikan konsistensi tipe data ketika berinteraksi dengan atribut Model.

```
protected function price(): Attribute
{
    return Attribute::make(
        get: fn ($value) => 'Rp ' . number_format($value, 0, ',', '.'),
    );
}
```

7.6 Eloquent Collection dan Manipulasi Data Lanjutan

Setiap hasil query Eloquent yang mengembalikan lebih dari satu baris data dikemas dalam bentuk Eloquent Collection, sebuah pembungkus (wrapper) di atas array PHP yang menyediakan puluhan method siap pakai untuk memanipulasi data tanpa perlu menulis perulangan manual. Kemampuan ini sangat berguna ketika mengolah data laporan penjualan atau merangkum isi keranjang belanja pada aplikasi Web Retail.

```
$produk = Product::all();

// Menjumlahkan total nilai stok seluruh produk
$totalNilaiStok = $produk->sum(fn($p) => $p->price * $p->stock);

// Mengelompokkan produk berdasarkan kategori
$produkPerKategori = $produk->groupBy('category_id');

// Menyaring produk dengan stok menipis
$stokMenipis = $produk->filter(fn($p) => $p->stock <= 5);

// Mengurutkan produk berdasarkan harga tertinggi
$termahal = $produk->sortByDesc('price')->take(5);
```

7.7 Soft Delete pada Model

Soft delete merupakan mekanisme penghapusan data secara logis, di mana baris data tidak benar-benar dihapus dari basis data, melainkan hanya ditandai dengan stempel waktu pada kolom deleted_at. Mekanisme ini sangat bermanfaat pada aplikasi Web Retail, misalnya ketika sebuah produk dihapus dari katalog namun tetap harus tersimpan riwayatnya karena pernah muncul pada order_items pelanggan sebelumnya.

```
use Illuminate\Database\Eloquent\SoftDeletes;

class Product extends Model
{
    use SoftDeletes;
    protected $dates = ['deleted_at'];
}

// Pada migration, tambahkan kolom deleted_at
$table->softDeletes();

// Mengambil data yang sudah dihapus (soft deleted)
$produkTerhapus = Product::onlyTrashed()->get();
// Mengembalikan data yang terhapus
Product::withTrashed()->find(3)->restore();
```

Rangkuman Bab 7

Eloquent ORM memungkinkan interaksi dengan basis data melalui objek PHP tanpa menuliskan SQL secara manual. Relasi antar Model seperti One to Many dan Many to Many merepresentasikan hubungan data retail seperti Kategori-Produk dan Order-OrderItem, sedangkan accessor dan mutator memungkinkan transformasi data secara otomatis.

Latihan Bab 7

30. Buatlah Model Category dan definisikan relasi hasMany menuju Model Product.
31. Buatlah query Eloquent untuk menampilkan lima produk dengan stok terbanyak.
32. Jelaskan perbedaan antara relasi hasOne dan belongsTo pada Eloquent.
33. Implementasikan accessor pada Model Product untuk memformat harga menjadi format Rupiah.

BAB 8 MIGRATION DAN PERANCANGAN BASIS DATA

8.1 Konsep Migration

Migration merupakan mekanisme version control untuk skema basis data yang memungkinkan pengembang mendefinisikan dan memodifikasi struktur tabel menggunakan kode PHP, bukan melalui perintah SQL manual. Dengan migration, struktur basis data dapat dibagikan antar anggota tim pengembang dan direplikasi secara konsisten pada lingkungan pengembangan, pengujian, maupun produksi.

8.2 Membuat dan Menjalankan Migration

```
php artisan make:migration create_categories_table
php artisan make:migration create_products_table
php artisan make:migration create_orders_table
php artisan make:migration create_order_items_table
php artisan make:migration create_payments_table
```

Contoh berikut menunjukkan implementasi migration untuk tabel products, lengkap dengan foreign key menuju tabel categories.

```
public function up(): void
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->foreignId('category_id')->constrained()-
>onDelete('cascade');
        $table->string('name');
        $table->string('slug')->unique();
        $table->text('description')->nullable();
        $table->decimal('price', 12, 2);
        $table->integer('stock')->default(0);
        $table->string('image')->nullable();
        $table->timestamps();
    });
}
```

```
php artisan migrate
php artisan migrate:status
php artisan migrate:rollback
```

8.3 Seeder dan Factory

Seeder digunakan untuk mengisi basis data dengan data awal (dummy/sample data), sedangkan Factory digunakan untuk menghasilkan data uji dalam jumlah besar secara otomatis, sangat berguna untuk pengujian performa aplikasi Web Retail sebelum dirilis ke lingkungan produksi.

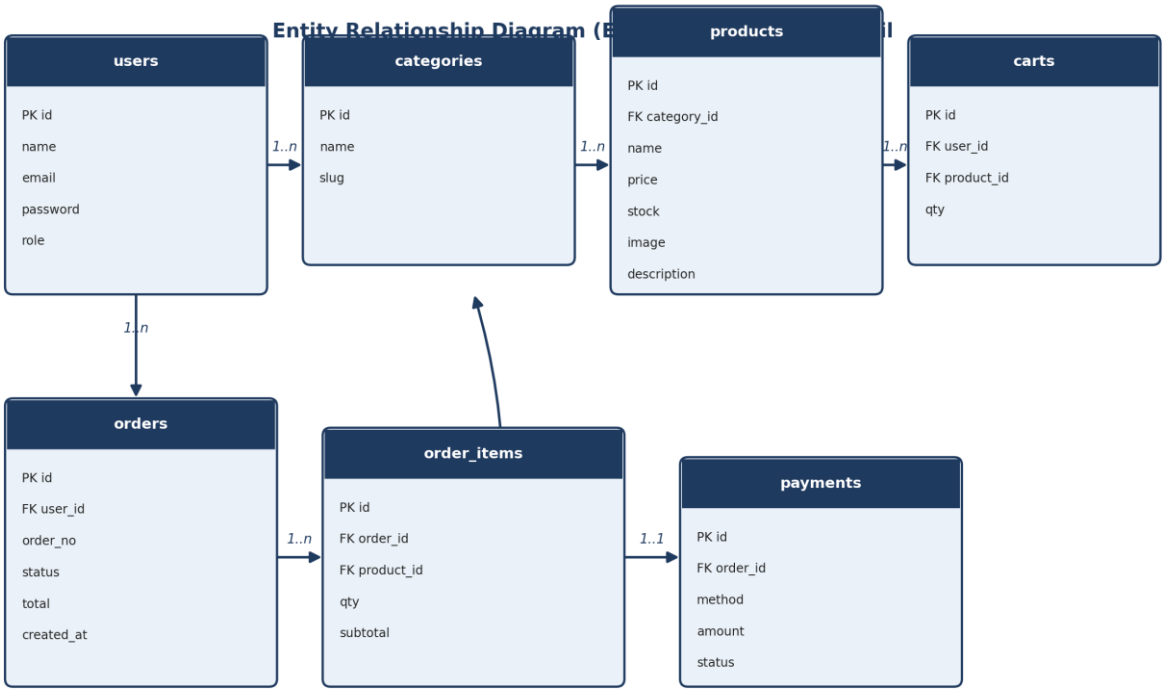
```
php artisan make:seeder ProductSeeder
php artisan make:factory ProductFactory --model=Product
```

```
// database/seeders/ProductSeeder.php
public function run(): void
{
    Product::factory()->count(50)->create();
}

// Menjalankan seluruh seeder
// php artisan db:seed
```

8.4 Perancangan Basis Data Aplikasi Web Retail (ERD)

Sebelum mengimplementasikan migration secara menyeluruh, perancangan basis data perlu dituangkan dalam bentuk Entity Relationship Diagram (ERD) agar relasi antar entitas dapat dipahami secara utuh. Rancangan ERD untuk aplikasi Web Retail pada modul ini disajikan pada Gambar berikut, mencakup tujuh entitas utama: users, categories, products, carts, orders, order_items, dan payments.



Gambar 6. Entity Relationship Diagram (ERD) Aplikasi Web Retail

Entitas users menyimpan data pengguna baik berperan sebagai admin maupun customer. Entitas products memiliki relasi many-to-one terhadap categories, artinya satu kategori dapat memiliki banyak produk. Entitas orders merepresentasikan transaksi yang dilakukan pengguna, yang kemudian memiliki relasi one-to-many terhadap order_items sebagai rincian produk yang dibeli, serta relasi one-to-one atau one-to-many terhadap payments untuk mencatat informasi pembayaran.

8.5 Praktik: Implementasi Migration Tabel Retail

Berikut disajikan struktur atribut lengkap untuk masing-masing tabel yang akan diimplementasikan pada migration.

Tabel	Atribut Utama	Keterangan
users	id, name, email, password, role	role membedakan admin dan customer
categories	id, name, slug	Kategori produk retail
products	id, category_id, name, price, stock, image	Data katalog produk
carts	id, user_id, product_id, qty	Keranjang belanja sementara

Tabel	Atribut Utama	Keterangan
orders	id, user_id, order_no, status, total	Data transaksi/pesanan
order_items	id, order_id, product_id, qty, subtotal	Rincian item pada satu order
payments	id, order_id, method, amount, status	Informasi pembayaran order

Tabel 18. Struktur Atribut Tabel Basis Data Web Retail

8.6 Normalisasi Basis Data pada Perancangan Retail

Normalisasi adalah proses perancangan struktur tabel basis data relasional untuk meminimalkan redundansi data dan menghindari anomali penyimpanan (insert, update, delete anomaly). Pada perancangan basis data Web Retail di modul ini, tabel-tabel telah dirancang mengikuti prinsip normalisasi hingga Bentuk Normal Ketiga (Third Normal Form/3NF), di mana setiap atribut non-kunci hanya bergantung penuh pada primary key tabel bersangkutan.

Bentuk Normal	Prinsip Utama	Penerapan pada Skema Retail
1NF	Setiap kolom bernilai atomik, tidak ada grup berulang	Kolom seperti name dan price bernilai tunggal, bukan array
2NF	Memenuhi 1NF dan seluruh atribut bergantung penuh pada primary key	Tabel order_items memisahkan qty dan subtotal dari data produk
3NF	Memenuhi 2NF dan tidak ada ketergantungan transitif antar atribut non-kunci	Nama kategori disimpan pada tabel categories, bukan diulang pada tabel products

Tabel 19. Penerapan Prinsip Normalisasi pada Skema Basis Data Web Retail

Meskipun normalisasi penting untuk menjaga integritas data, pada beberapa kasus tertentu diperlukan denormalisasi terbatas demi performa, misalnya menyimpan kolom subtotal pada tabel order_items alih-alih menghitungnya ulang setiap saat, karena harga produk dapat berubah sewaktu-waktu sedangkan riwayat transaksi harus tetap mencerminkan harga pada saat pembelian terjadi.

Rangkuman Bab 8

Migration memungkinkan definisi dan versioning skema basis data menggunakan kode PHP. Seeder dan Factory mempercepat pengisian data uji. Perancangan ERD

yang matang, seperti pada Gambar 4, menjadi fondasi penting sebelum implementasi migration agar relasi antar tabel pada aplikasi Web Retail konsisten dan efisien.

Latihan Bab 8

34. Implementasikan migration untuk tabel carts dan orders sesuai struktur pada Tabel 11.
35. Buatlah CategorySeeder yang mengisi minimal lima kategori produk retail.
36. Jelaskan fungsi foreign key dan constrained() pada migration Laravel.
37. Gambarkan ulang ERD pada Gambar 4 dengan menambahkan satu entitas baru, misalnya 'reviews', beserta relasinya.

BAB 9 VIEW DAN BLADE TEMPLATING ENGINE

9.1 Konsep Blade Templating Engine

Blade merupakan mesin templat (templating engine) bawaan Laravel yang digunakan untuk membangun tampilan (View) aplikasi. Berbeda dengan templat PHP native, Blade menyediakan sintaksis directive yang lebih ringkas untuk kontrol alur (percabangan dan perulangan), pewarisan tata letak (layout inheritance), serta mekanisme keamanan otomatis berupa escaping output untuk mencegah serangan Cross-Site Scripting (XSS).

Berkas Blade disimpan pada direktori resources/views dengan ekstensi .blade.php. Laravel akan mengompilasi berkas Blade menjadi kode PHP murni dan menyimpannya dalam bentuk cache agar proses rendering pada permintaan berikutnya berjalan lebih cepat.

9.2 Directive Blade

Directive	Fungsi
{{ $variabel }}	Menampilkan nilai variabel dengan escaping otomatis
@if / @elseif / @else / @endif	Struktur percabangan kondisi
@foreach / @endforeach	Perulangan terhadap koleksi data
@extends('layout')	Mewarisi tata letak (layout) utama
@section / @endsection	Mendefinisikan blok konten pada layout
@include('partial')	Menyisipkan potongan Blade lain (partial view)
@csrf	Menyisipkan token keamanan CSRF pada form
@auth / @guest	Kondisi berdasarkan status login pengguna

Tabel 20. Directive Blade yang Umum Digunakan

```
{{-- resources/views/produk/index.blade.php --}}
@extends('layouts.app')

@section('content')
<div class="row">
  @foreach ($produk as $item)
    <div class="col-md-3 mb-4">
      <div class="card h-100">
```

```
        <img src="{{ asset('storage/'.$item->image) }}" class="card-img-
top">
        <div class="card-body">
          <h6 class="card-title">{{ $item->name }}</h6>
          <p class="text-primary fw-bold">{{ $item->formatted_price
}}</p>
          <a href="{{ route('produk.show', $item->id) }}" class="btn btn-
sm btn-outline-primary">
            Lihat Detail
          </a>
        </div>
      </div>
    </div>
  @endforeach
</div>
{{ $produk->links() }}
@endsection
```

9.3 Layout Master Page

Salah satu keunggulan Blade adalah kemampuan pewarisan layout, di mana tata letak umum aplikasi seperti header, navigasi, dan footer didefinisikan sekali pada sebuah berkas layout utama, kemudian diwarisi oleh seluruh halaman lain menggunakan directive @extends dan @section. Pendekatan ini menerapkan prinsip Don't Repeat Yourself (DRY) yang menghindari duplikasi kode tampilan.

```
{{-- resources/views/layouts/app.blade.php --}}
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>@yield('title', 'Web Retail Binabangsa')</title>
  <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body>
  @include('partials.navbar')
  <main class="container my-4">
    @yield('content')
  </main>
  @include('partials.footer')
  <script src="{{ asset('js/app.js') }}"></script>
</body>
</html>
```

9.4 Komponen Blade

Selain @include, Laravel Blade juga mendukung konsep komponen (Blade Component) yang lebih terstruktur untuk elemen UI yang dapat digunakan berulang

kali dengan properti dinamis, misalnya komponen kartu produk atau tombol notifikasi.

```
php artisan make:component ProductCard

{{-- Penggunaan pada Blade lain --}}
<x-product-card :produk="$item" />
```

9.5 Praktik: Menyusun Layout Aplikasi Web Retail

Sebagai praktik, mahasiswa diminta menyusun tiga bagian utama layout aplikasi Web Retail, yaitu partials.navbar berisi menu navigasi dan ikon keranjang belanja, partials.footer berisi informasi kontak toko, serta layouts.app sebagai kerangka utama yang mewadahi kedua partial tersebut beserta blok konten dinamis.

9.6 Blade Slot dan Komponen Dinamis

Blade Component mendukung konsep slot, yaitu area konten yang dapat diisi secara dinamis oleh pemanggil komponen, mirip dengan konsep children pada komponen React atau Vue. Fitur ini sangat berguna untuk membangun komponen pembungkus (wrapper) seperti modal konfirmasi atau kartu notifikasi yang kontennya berbeda-beda pada setiap pemanggilan.

```
{{-- resources/views/components/alert-box.blade.php --}}
<div class="alert alert-{{ $type ?? 'info' }}">
  <strong>{{ $title }}</strong>
  <div>{{ $slot }}</div>
</div>

{{-- Penggunaan pada Blade lain --}}
<x-alert-box type="warning">
  <x-slot:title>Perhatian</x-slot:title>
  Stok produk ini tersisa kurang dari 5 unit.
</x-alert-box>
```

9.7 Pengelolaan Aset Statis (Gambar, CSS, JavaScript)

Laravel menyediakan helper asset() untuk menghasilkan URL menuju berkas statis yang disimpan pada direktori public, serta mekanisme symbolic link melalui perintah php artisan storage:link agar berkas yang diunggah pengguna (seperti

gambar produk) dapat diakses secara publik meski secara fisik tersimpan pada direktori storage/app/public yang tidak dapat diakses langsung oleh web server.

```
php artisan storage:link

{{-- Blade --}}
<img src="{{ asset('storage/' . $produk->image) }}" alt="{{ $produk->name
}}">
```

Rangkuman Bab 9

Blade adalah templating engine Laravel yang menyediakan directive ringkas untuk kontrol alur, pewarisan layout, dan komponen UI. Struktur layout master page dengan @extends dan @section memastikan tampilan aplikasi Web Retail konsisten pada seluruh halaman tanpa duplikasi kode.

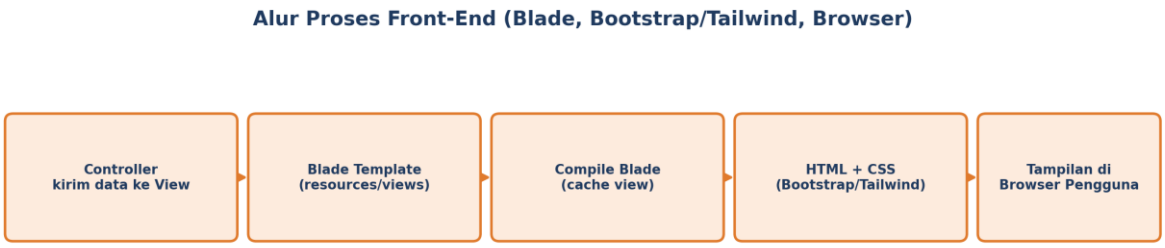
Latihan Bab 9

38. Buatlah layout master page bernama layouts.app yang memuat navbar dan footer aplikasi Web Retail.
39. Buatlah halaman produk.show yang menampilkan detail satu produk dengan mewarisi layout tersebut.
40. Jelaskan perbedaan antara @include dan Blade Component.
41. Jelaskan mengapa {{ }} pada Blade lebih aman dibandingkan <?php echo ?> biasa.

BAB 10 INTEGRASI FRONT-END (BOOTSTRAP / TAILWIND)

10.1 Peran Framework CSS dalam Pengembangan Front-End

Framework CSS seperti Bootstrap dan Tailwind CSS digunakan untuk mempercepat pengembangan antarmuka pengguna yang responsif tanpa harus menulis seluruh gaya (style) dari awal. Bootstrap menyediakan komponen UI siap pakai berbasis class utilitas dan grid system dua belas kolom, sedangkan Tailwind CSS mengusung pendekatan utility-first yang memberikan fleksibilitas desain lebih tinggi namun memerlukan proses build melalui Vite.



Gambar 7. Alur Proses Front-End: dari Blade Template hingga Tampilan Browser

10.2 Integrasi Bootstrap pada Laravel

Bootstrap dapat diintegrasikan melalui Content Delivery Network (CDN) untuk kebutuhan pengembangan cepat, atau melalui manajer paket npm dan proses build Vite untuk kebutuhan produksi yang lebih optimal.

```
<!-- Integrasi melalui CDN pada layouts/app.blade.php -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min
.css" rel="stylesheet">
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundl
e.min.js"></script>
```

10.3 Integrasi Tailwind CSS melalui Vite

```
npm install -D tailwindcss postcss autoprefixer
```

```
npx tailwindcss init -p
npm run dev     # menjalankan Vite dalam mode pengembangan
npm run build   # membangun aset untuk produksi
```

```
{{-- Menyertakan aset Vite pada layout --}}
@vite(['resources/css/app.css', 'resources/js/app.js'])
```

10.4 Merancang Komponen UI Toko Retail

Beberapa komponen antarmuka yang umum dijumpai pada aplikasi Web Retail antara lain navigasi utama dengan ikon keranjang belanja, kartu produk (product card) yang menampilkan gambar, nama, dan harga, filter kategori pada sidebar, serta tombol aksi seperti 'Tambah ke Keranjang' dan 'Checkout Sekarang'. Konsistensi visual pada komponen-komponen ini penting untuk membangun kepercayaan pengguna terhadap toko daring.

```
<!-- Contoh Navbar dengan Bootstrap 5 -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="{{ route('home') }}">Web Retail
Binabangsa</a>
    <ul class="navbar-nav ms-auto">
      <li class="nav-item">
        <a class="nav-link" href="{{ route('keranjang.index') }}">
          <i class="bi bi-cart3"></i> Keranjang ({{ $jumlahKeranjang ?? 0
}})
        </a>
      </li>
    </ul>
  </div>
</nav>
```

10.5 Responsivitas dan Pengalaman Pengguna (UX)

Desain responsif memastikan tampilan aplikasi Web Retail dapat menyesuaikan diri terhadap berbagai ukuran layar, mulai dari komputer desktop hingga perangkat mobile. Baik Bootstrap maupun Tailwind menyediakan sistem grid dan breakpoint (sm, md, lg, xl) yang memudahkan penataan elemen UI secara adaptif tanpa menulis media query CSS secara manual.

Breakpoint	Lebar Layar Minimum	Contoh Penerapan
sm	576px	Tata letak dua kolom pada tablet kecil
md	768px	Tata letak tiga kolom kartu produk
lg	992px	Tata letak empat kolom kartu produk pada desktop
xl	1200px	Lebar maksimum kontainer utama halaman

Tabel 21. Breakpoint Responsif yang Umum Digunakan

10.6 Aksesibilitas dan Konsistensi Desain Antarmuka

Selain aspek responsivitas, kualitas front-end aplikasi Web Retail juga ditentukan oleh aksesibilitas (accessibility), yaitu sejauh mana antarmuka dapat digunakan oleh pengguna dengan berbagai keterbatasan, misalnya melalui penggunaan atribut alt pada gambar, kontras warna yang memadai, serta label yang jelas pada elemen formulir. Konsistensi desain, seperti penggunaan palet warna, tipografi, dan ukuran tombol yang seragam di seluruh halaman, turut membangun kepercayaan pengguna terhadap profesionalitas toko daring.

Prinsip Desain	Penerapan pada Web Retail
Kontras Warna Memadai	Teks harga produk menggunakan warna dengan kontras tinggi terhadap latar
Label Formulir yang Jelas	Setiap input pada form checkout disertai label eksplisit, bukan hanya placeholder
Atribut Alt pada Gambar	Setiap gambar produk memiliki teks alternatif yang mendeskripsikan produk
Ukuran Tombol Konsisten	Tombol 'Tambah ke Keranjang' memiliki ukuran dan warna yang sama di seluruh halaman
Umpan Balik Visual	Notifikasi sukses/gagal ditampilkan secara konsisten menggunakan komponen alert yang sama

Tabel 22. Prinsip Desain Antarmuka yang Diterapkan pada Web Retail

Rangkuman Bab 10

Integrasi framework CSS seperti Bootstrap atau Tailwind mempercepat pembangunan antarmuka aplikasi Web Retail yang konsisten dan responsif. Vite berperan sebagai

build tool modern pada Laravel untuk mengompilasi dan mengoptimalkan aset CSS/JS sebelum digunakan di lingkungan produksi.

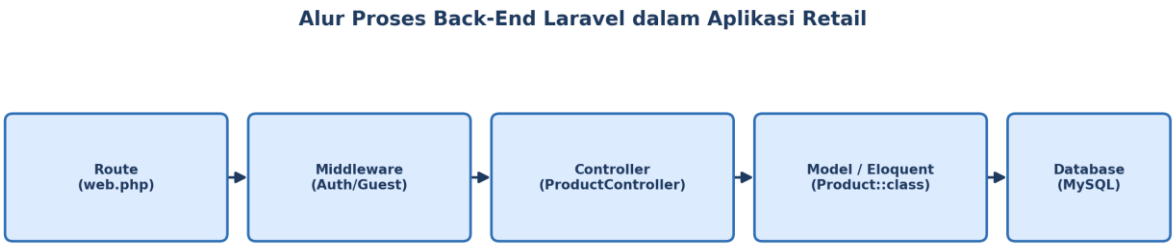
Latihan Bab 10

42. Rancang komponen kartu produk menggunakan Bootstrap 5 lengkap dengan gambar, nama, harga, dan tombol 'Tambah ke Keranjang'.
43. Jelaskan perbedaan pendekatan utility-first Tailwind dengan pendekatan komponen Bootstrap.
44. Bangun navbar responsif yang menampilkan jumlah item pada keranjang belanja.
45. Jelaskan fungsi perintah npm run dev dan npm run build pada proses pengembangan front-end Laravel.

BAB 11 CRUD OPERATIONS — STUDI KASUS MANAJEMEN PRODUK

11.1 Pendahuluan Studi Kasus

Bab ini mengimplementasikan operasi Create, Read, Update, Delete (CRUD) secara utuh pada modul manajemen produk, sebagai fondasi utama aplikasi Web Retail. Implementasi menggunakan Resource Controller yang telah dijelaskan pada Bab 6, dipadukan dengan Model Product dari Bab 7, migration dari Bab 8, serta Blade Template dari Bab 9 dan Bab 10, sehingga bab ini menjadi titik integrasi dari seluruh konsep back-end dan front-end yang telah dipelajari sebelumnya.



Gambar 8. Alur Proses Back-End Laravel dalam Menangani Permintaan Manajemen Produk

11.2 Create — Menambahkan Data Produk

Proses penambahan data produk melibatkan dua tahap utama, yaitu menampilkan form input (method create) dan memproses data yang dikirimkan (method store). Formulir tambah produk memerlukan input nama produk, kategori, harga, stok, gambar, dan deskripsi.

```
{{-- resources/views/admin/produk/create.blade.php --}}
@extends('layouts.admin')
@section('content')
<form action="{{ route('admin.produk.store') }}" method="POST"
enctype="multipart/form-data">
  @csrf
  <div class="mb-3">
    <label class="form-label">Nama Produk</label>
    <input type="text" name="name" class="form-control" required>
  </div>
  <div class="mb-3">
```

```
    <label class="form-label">Kategori</label>
    <select name="category_id" class="form-select" required>
      @foreach ($kategori as $k)
        <option value="{{ $k->id }}">{{ $k->name }}</option>
      @endforeach
    </select>
  </div>
  <div class="mb-3">
    <label class="form-label">Harga</label>
    <input type="number" name="price" class="form-control" required>
  </div>
  <div class="mb-3">
    <label class="form-label">Stok</label>
    <input type="number" name="stock" class="form-control" required>
  </div>
  <div class="mb-3">
    <label class="form-label">Gambar Produk</label>
    <input type="file" name="image" class="form-control">
  </div>
  <button type="submit" class="btn btn-primary">Simpan Produk</button>
</form>
@endsection
```

```
public function store(Request $request)
{
    $validated = $request->validate([
        'name'        => 'required|string|max:150',
        'category_id' => 'required|exists:categories,id',
        'price'       => 'required|numeric|min:0',
        'stock'       => 'required|integer|min:0',
        'image'       => 'nullable|image|max:2048',
    ]);

    if ($request->hasFile('image')) {
        $validated['image'] = $request->file('image')->store('produk',
'public');
    }
    $validated['slug'] = Str::slug($validated['name']);

    Product::create($validated);

    return redirect()->route('admin.produk.index')
                     ->with('success', 'Produk berhasil ditambahkan.');
}
```

11.3 Read — Menampilkan Data Produk

Penampilan data produk terbagi menjadi dua konteks, yaitu tampilan katalog untuk pelanggan (front-end toko) dan tampilan tabel manajemen untuk admin (back-end dashboard). Pagination digunakan agar performa aplikasi tetap optimal ketika jumlah data produk terus bertambah.

```
public function index(Request $request)
{
    $produk = Product::with('category')
        ->when($request->cari, fn($q) => $q->where('name', 'like',
"%{$request->cari}%"))
        ->latest()->paginate(10);

    return view('admin.produk.index', compact('produk'));
}
```

11.4 Update — Mengedit Data Produk

```
public function update(Request $request, Product $produk)
{
    $validated = $request->validate([
        'name'        => 'required|string|max:150',
        'category_id' => 'required|exists:categories,id',
        'price'       => 'required|numeric|min:0',
        'stock'       => 'required|integer|min:0',
        'image'       => 'nullable|image|max:2048',
    ]);

    if ($request->hasFile('image')) {
        if ($produk->image) Storage::disk('public')->delete($produk-
>image);
        $validated['image'] = $request->file('image')->store('produk',
'public');
    }

    $produk->update($validated);

    return redirect()->route('admin.produk.index')
                     ->with('success', 'Produk berhasil diperbarui.');
}
```

11.5 Delete — Menghapus Data Produk

```
public function destroy(Product $produk)
{
    if ($produk->image) {
        Storage::disk('public')->delete($produk->image);
    }
    $produk->delete();

    return back()->with('success', 'Produk berhasil dihapus.');
}
```

11.6 Validasi Formulir

Validasi merupakan langkah wajib untuk memastikan data yang masuk ke sistem konsisten dan aman. Laravel menyediakan berbagai aturan validasi bawaan

(validation rules) yang dapat dikombinasikan sesuai kebutuhan, serta mendukung pembuatan Form Request terpisah untuk validasi yang lebih kompleks dan dapat digunakan ulang.

```
php artisan make:request StoreProductRequest
```

Aturan Validasi	Keterangan
required	Kolom wajib diisi
numeric	Nilai harus berupa angka
min:0 / max:150	Batas nilai minimum atau panjang maksimum karakter
unique:products,slug	Nilai harus unik pada tabel tertentu
exists:categories,id	Nilai harus terdapat pada tabel referensi
image|max:2048	Berkas harus berupa gambar dengan ukuran maksimum 2MB

Tabel 23. Aturan Validasi yang Umum Digunakan pada Form Produk

11.7 Implementasi Modul Kategori Produk

Selain modul produk, aplikasi Web Retail memerlukan modul manajemen kategori agar admin dapat mengelompokkan produk secara fleksibel. Implementasi CategoryController mengikuti pola CRUD yang serupa dengan ProductController, namun dengan struktur data yang lebih sederhana.

```
class CategoryController extends Controller
{
    public function index()
    {
        $kategori = Category::withCount('products')->latest()-
>paginate(10);
        return view('admin.kategori.index', compact('kategori'));
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:100|unique:categories,name',
        ]);
        $validated['slug'] = Str::slug($validated['name']);

        Category::create($validated);

        return back()->with('success', 'Kategori berhasil ditambahkan.');
    }
```

```
    public function destroy(Category $kategori)
    {
        if ($kategori->products()->exists()) {
            return back()->with('error', 'Kategori tidak dapat dihapus
karena masih memiliki produk.');
        }
        $kategori->delete();
        return back()->with('success', 'Kategori berhasil dihapus.');
    }
}
```

11.8 Penanganan Kesalahan (Error Handling) pada CRUD

Penanganan kesalahan yang baik memastikan pengguna menerima pesan yang informatif ketika terjadi masalah, alih-alih melihat halaman error teknis yang membingungkan. Laravel secara otomatis menampilkan pesan validasi pada Blade melalui variabel $errors, serta mendukung penanganan exception kustom untuk kasus khusus seperti data tidak ditemukan.

```
{{-- Menampilkan pesan validasi pada Blade --}}
@if ($errors->any())
  <div class="alert alert-danger">
    <ul class="mb-0">
      @foreach ($errors->all() as $error)
        <li>{{ $error }}</li>
      @endforeach
    </ul>
  </div>
@endif

// Penanganan data tidak ditemukan pada Controller
public function show($id)
{
    $produk = Product::find($id);
    if (!$produk) {
        abort(404, 'Produk yang Anda cari tidak ditemukan.');
    }
    return view('produk.show', compact('produk'));
}
```

Rangkuman Bab 11

Modul manajemen produk mengintegrasikan Routing, Controller, Model, Migration, dan Blade Template menjadi satu alur CRUD yang utuh. Validasi data yang ketat menjadi kunci menjaga integritas data katalog produk pada aplikasi Web Retail.

Latihan Bab 11

46. Implementasikan CRUD lengkap untuk modul kategori produk mengikuti pola pada bab ini.
47. Tambahkan fitur pencarian produk berdasarkan nama pada halaman manajemen produk admin.
48. Buatlah Form Request StoreProductRequest dan pindahkan logika validasi dari Controller ke dalamnya.
49. Jelaskan risiko keamanan apabila validasi upload gambar tidak diterapkan pada form tambah produk.

BAB 12 AUTENTIKASI DAN OTORISASI

12.1 Konsep Autentikasi dan Otorisasi

Autentikasi (authentication) adalah proses verifikasi identitas pengguna, umumnya melalui kombinasi email dan kata sandi, untuk memastikan bahwa pengguna adalah pihak yang berhak mengakses sistem. Otorisasi (authorization) adalah proses penentuan hak akses pengguna yang telah terautentikasi terhadap sumber daya atau fitur tertentu dalam aplikasi. Pada aplikasi Web Retail, kedua konsep ini sangat penting untuk membedakan hak akses antara pelanggan (customer) dan pengelola toko (admin).

12.2 Instalasi Laravel Breeze

Laravel Breeze merupakan paket starter kit resmi yang menyediakan implementasi autentikasi dasar secara cepat, meliputi fitur registrasi, login, logout, verifikasi email, dan reset kata sandi, lengkap dengan tampilan antarmuka berbasis Blade dan Tailwind CSS.

```
composer require laravel/breeze --dev
php artisan breeze:install blade
npm install && npm run dev
php artisan migrate
```

12.3 Menambahkan Kolom Role pada Tabel Users

Untuk mendukung dua peran pengguna, tabel users perlu ditambahkan kolom role melalui migration tambahan.

```
php artisan make:migration add_role_to_users_table

public function up(): void
{
    Schema::table('users', function (Blueprint $table) {
        $table->enum('role', ['admin', 'customer'])->default('customer')-
>after('email');
    });
}
```

12.4 Middleware Otorisasi Berbasis Peran (Role-Based Access)

Middleware kustom dapat dibuat untuk membatasi akses halaman tertentu hanya bagi pengguna dengan peran tertentu, misalnya membatasi akses dashboard admin hanya bagi pengguna dengan role admin.

```
php artisan make:middleware EnsureUserIsAdmin

public function handle(Request $request, Closure $next): Response
{
    if ($request->user()?->role !== 'admin') {
        abort(403, 'Akses ditolak. Halaman ini khusus admin.');
    }
    return $next($request);
}
```

```
// bootstrap/app.php (Laravel 11) atau app/Http/Kernel.php (Laravel 10)
$middleware->alias([
    'admin' => \App\Http\Middleware\EnsureUserIsAdmin::class,
]);

// Penerapan pada routes/web.php
Route::middleware(['auth', 'admin'])->prefix('admin')->group(function ()
{
    Route::resource('produk', Admin\ProductController::class);
});
```

12.5 Proteksi Rute dan Tampilan Berbasis Peran

Selain membatasi akses rute melalui middleware, tampilan (View) juga perlu disesuaikan berdasarkan peran pengguna, misalnya menyembunyikan menu 'Kelola Produk' dari pelanggan biasa. Directive @auth, @guest, dan pengecekan kondisional terhadap atribut role dapat digunakan pada Blade Template.

```
@auth
  @if (auth()->user()->role === 'admin')
    <a href="{{ route('admin.dashboard') }}" class="nav-link">Dashboard
Admin</a>
  @else
    <a href="{{ route('keranjang.index') }}" class="nav-link">Keranjang
Saya</a>
  @endif
@else
  <a href="{{ route('login') }}" class="nav-link">Masuk</a>
@endauth
```

12.6 Praktik: Alur Login dan Register pada Web Retail

Sebagai praktik, mahasiswa diminta mengimplementasikan alur pendaftaran pelanggan baru yang secara otomatis diberi role customer, serta menyiapkan satu akun admin melalui Seeder untuk keperluan pengujian dashboard manajemen.

```
// database/seeders/AdminSeeder.php
public function run(): void
{
    User::create([
        'name' => 'Administrator Toko',
        'email' => 'admin@webretail.test',
        'password' => Hash::make('password123'),
        'role' => 'admin',
    ]);
}
```

12.7 Autentikasi API dengan Laravel Sanctum

Apabila aplikasi Web Retail di masa mendatang perlu diakses melalui aplikasi mobile atau Single Page Application (SPA) terpisah, mekanisme autentikasi berbasis session cookie yang digunakan Laravel Breeze tidak lagi memadai. Laravel Sanctum hadir sebagai solusi ringan untuk autentikasi API berbasis token, memungkinkan setiap pengguna memperoleh token unik yang disertakan pada setiap permintaan API sebagai bukti identitas.

```
composer require laravel/sanctum
php artisan vendor:publish --
provider="Laravel\Sanctum\SanctumServiceProvider"
php artisan migrate
```

```
// Menghasilkan token API saat login
public function apiLogin(Request $request)
{
    $user = User::where('email', $request->email)->first();

    if (!$user || !Hash::check($request->password, $user->password)) {
        return response()->json(['message' => 'Kredensial tidak valid'],
401);
    }

    $token = $user->createToken('web-retail-app')->plainTextToken;

    return response()->json(['token' => $token, 'user' => $user]);
}

// Melindungi route API dengan Sanctum
Route::middleware('auth:sanctum')->get('/api/produk',
[ApiProductController::class, 'index']);
```

12.8 Perbandingan Mekanisme Autentikasi

Mekanisme	Penyimpanan Status	Kasus Penggunaan Ideal
Session (Laravel Breeze)	Session pada server + cookie di browser	Aplikasi web tradisional berbasis Blade
Token API (Laravel Sanctum)	Token tersimpan pada client (mobile/SPA)	Aplikasi mobile atau front-end terpisah (React/Vue)
OAuth2 (Laravel Passport)	Access token dan refresh token	Aplikasi yang memerlukan integrasi pihak ketiga skala besar

Tabel 24. Perbandingan Mekanisme Autentikasi pada Ekosistem Laravel

Rangkuman Bab 12

Autentikasi memverifikasi identitas pengguna, sedangkan otorisasi menentukan hak akses. Laravel Breeze mempercepat implementasi autentikasi dasar, sementara middleware kustom berbasis peran memastikan hanya admin yang dapat mengakses fitur manajemen toko pada aplikasi Web Retail.

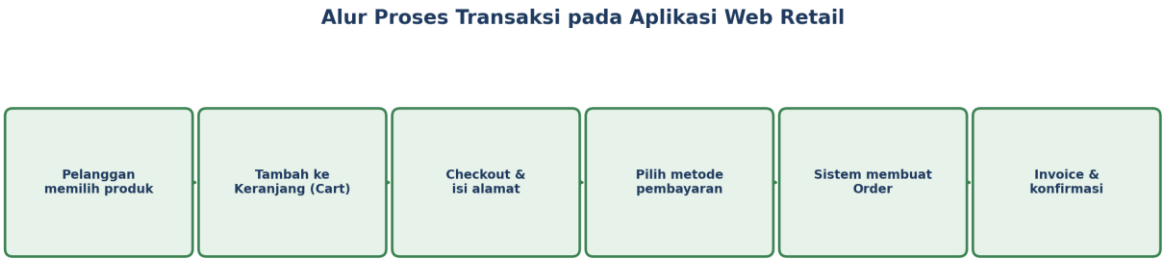
Latihan Bab 12

50. Instal Laravel Breeze pada proyek Web Retail dan uji coba fitur registrasi serta login.
51. Tambahkan kolom role pada tabel users dan buat AdminSeeder sesuai contoh pada bab ini.
52. Implementasikan middleware EnsureUserIsAdmin dan terapkan pada seluruh rute admin.
53. Jelaskan perbedaan mendasar antara konsep autentikasi dan otorisasi beserta contohnya masing-masing.

BAB 13 KERANJANG BELANJA DAN TRANSAKSI (SHOPPING CART & CHECKOUT)

13.1 Desain Alur Transaksi Retail

Proses transaksi merupakan inti dari aplikasi Web Retail. Alur transaksi yang baik harus mencerminkan pengalaman berbelanja yang alami, dimulai dari pemilihan produk, penambahan ke keranjang belanja, proses checkout, pemilihan metode pembayaran, hingga penerbitan invoice sebagai bukti transaksi.



Gambar 9. Alur Proses Transaksi pada Aplikasi Web Retail

Sebagaimana ditunjukkan pada Gambar 7, terdapat enam tahapan utama dalam alur transaksi. Setiap tahapan melibatkan interaksi antara front-end (tampilan yang dilihat pelanggan) dan back-end (logika pemrosesan data di server), yang keduanya harus bekerja secara sinkron agar data stok, harga, dan status pesanan tetap konsisten.

13.2 Implementasi Keranjang Belanja

Keranjang belanja dapat diimplementasikan dengan dua pendekatan, yaitu berbasis session (data disimpan sementara di server tanpa perlu tabel basis data terpisah) atau berbasis basis data (data disimpan pada tabel carts sehingga persisten meskipun pengguna logout). Pendekatan berbasis basis data lebih sesuai untuk aplikasi retail karena keranjang tetap tersimpan meski pengguna berpindah perangkat.

```
class CartController extends Controller
{
    public function store(Request $request)
    {
        $request->validate(['product_id' =>
'required|exists:products,id']);

        $produk = Product::findOrFail($request->product_id);
        if ($produk->stock < 1) {
            return back()->with('error', 'Stok produk habis.');
        }

        $cart = Cart::firstOrNew([
            'user_id'    => auth()->id(),
            'product_id' => $produk->id,
        ]);
        $cart->qty = ($cart->qty ?? 0) + $request->input('qty', 1);
        $cart->save();

        return back()->with('success', 'Produk ditambahkan ke
keranjang.');
    }

    public function index()
    {
        $items = Cart::with('product')->where('user_id', auth()->id())-
>get();
        $total = $items->sum(fn($i) => $i->qty * $i->product->price);

        return view('keranjang.index', compact('items', 'total'));
    }
}
```

13.3 Proses Checkout

Checkout adalah proses konversi isi keranjang belanja menjadi sebuah pesanan (order) resmi. Proses ini harus dilakukan dalam sebuah transaksi basis data (database transaction) agar konsisten — apabila terjadi kegagalan pada salah satu langkah, seluruh proses dapat dibatalkan (rollback) sehingga data tidak menjadi tidak konsisten, misalnya stok berkurang namun order gagal tercatat.

```
public function checkout(Request $request)
{
    $items = Cart::with('product')->where('user_id', auth()->id())-
>get();
    if ($items->isEmpty()) {
        return back()->with('error', 'Keranjang belanja masih kosong.');
    }

    return DB::transaction(function () use ($items, $request) {
        $order = Order::create([
            'user_id'  => auth()->id(),
```

```
            'order_no' => 'INV-' . now()->format('YmdHis') . '-' .
auth()->id(),
            'status'   => 'menunggu_pembayaran',
            'total'    => $items->sum(fn($i) => $i->qty * $i->product-
>price),
        ]);

        foreach ($items as $item) {
            if ($item->product->stock < $item->qty) {
                throw new \Exception("Stok {$item->product->name} tidak
mencukupi.");
            }
            $order->items()->create([
                'product_id' => $item->product_id,
                'qty'        => $item->qty,
                'subtotal'   => $item->qty * $item->product->price,
            ]);
            $item->product->decrement('stock', $item->qty);
        }

        Payment::create([
            'order_id' => $order->id,
            'method'   => $request->input('metode_bayar',
'transfer_bank'),
            'amount'   => $order->total,
            'status'   => 'pending',
        ]);

        Cart::where('user_id', auth()->id())->delete();

        return redirect()->route('order.invoice', $order->id)
                         ->with('success', 'Pesanan berhasil dibuat.');
    });
}
```

13.4 Manajemen Order dan Invoice

Setiap order yang berhasil dibuat memiliki status yang berubah sepanjang siklus hidupnya, misalnya dari menunggu_pembayaran, diproses, dikirim, hingga selesai. Status ini penting bagi pelanggan untuk memantau posisi pesanannya, sekaligus bagi admin untuk mengelola operasional pengiriman barang.

Status Order	Deskripsi
menunggu_pembayaran	Order dibuat namun pembayaran belum dikonfirmasi
diproses	Pembayaran telah dikonfirmasi, barang disiapkan oleh admin
dikirim	Barang telah diserahkan ke jasa pengiriman
selesai	Barang telah diterima pelanggan

Status Order	Deskripsi
dibatalkan	Order dibatalkan oleh pelanggan atau admin

Tabel 25. Status Siklus Hidup Order pada Aplikasi Web Retail

```
{{-- resources/views/order/invoice.blade.php --}}
@extends('layouts.app')
@section('content')
<div class="card p-4">
  <h4>Invoice #{{ $order->order_no }}</h4>
  <p>Status: <span class="badge bg-warning">{{ $order->status
}}</span></p>
  <table class="table">
    @foreach ($order->items as $item)
      <tr>
        <td>{{ $item->product->name }}</td>
        <td>{{ $item->qty }}</td>
        <td>Rp {{ number_format($item->subtotal, 0, ',', '.') }}</td>
      </tr>
    @endforeach
  </table>
  <h5 class="text-end">Total: Rp {{ number_format($order->total, 0, ',',
'.') }}</h5>
</div>
@endsection
```

13.5 Pengembangan Lanjutan: Diskon dan Kupon Belanja

Sebagai pengembangan lanjutan dari fitur transaksi dasar, aplikasi Web Retail dapat ditingkatkan dengan menambahkan mekanisme diskon atau kupon belanja untuk mendorong minat beli pelanggan. Implementasi ini memerlukan tabel tambahan bernama coupons yang menyimpan kode kupon, jenis potongan (persentase atau nominal tetap), serta masa berlaku.

```
Schema::create('coupons', function (Blueprint $table) {
    $table->id();
    $table->string('code')->unique();
    $table->enum('type', ['persen', 'nominal']);
    $table->decimal('value', 10, 2);
    $table->date('expired_at');
    $table->timestamps();
});

// Logika penerapan kupon pada proses checkout
$kupon = Coupon::where('code', $request->kode_kupon)
                ->where('expired_at', '>=', now())
                ->first();

if ($kupon) {
    $potongan = $kupon->type === 'persen'
        ? $total * ($kupon->value / 100)
```

```
        : $kupon->value;
    $total -= $potongan;
}
```

13.6 Pengembangan Lanjutan: Fitur Wishlist

Fitur wishlist memungkinkan pelanggan menyimpan produk yang diminati untuk dibeli di kemudian hari tanpa harus langsung memasukkannya ke keranjang belanja. Fitur ini umum dijumpai pada aplikasi retail modern dan dapat diimplementasikan dengan pola yang serupa dengan tabel carts, yaitu relasi many-to-many antara users dan products melalui tabel pivot wishlists.

```
// Model User
public function wishlist(): BelongsToMany
{
    return $this->belongsToMany(Product::class, 'wishlists');
}

// Menambahkan produk ke wishlist
auth()->user()->wishlist()->syncWithoutDetaching([$produkId]);
```

Rangkuman Bab 13

Modul keranjang belanja dan checkout merupakan jantung transaksional aplikasi Web Retail. Penggunaan database transaction (DB::transaction) memastikan proses checkout tetap konsisten meskipun terjadi kegagalan di tengah proses, sehingga data stok dan order selalu sinkron.

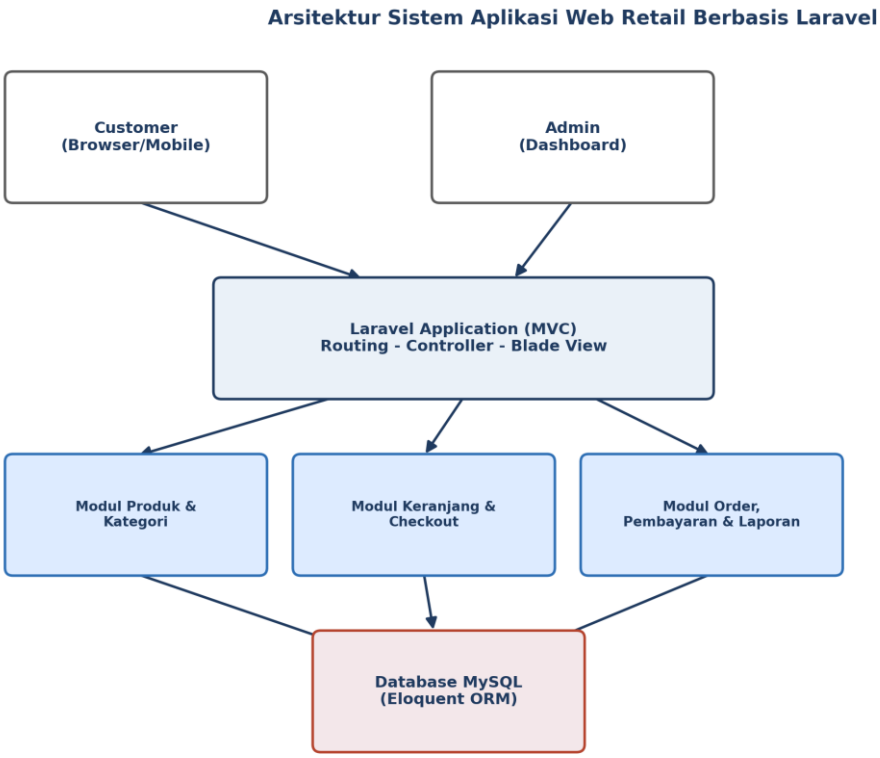
Latihan Bab 13

54. Implementasikan method update dan destroy pada CartController untuk mengubah jumlah dan menghapus item keranjang.
55. Tambahkan validasi agar pelanggan tidak dapat checkout apabila salah satu produk stoknya sudah habis.
56. Rancang halaman riwayat pesanan (order history) yang menampilkan seluruh order milik pelanggan yang sedang login.
57. Jelaskan mengapa proses checkout perlu dibungkus dalam DB::transaction().

BAB 14 DASHBOARD ADMIN DAN LAPORAN PENJUALAN

14.1 Desain Dashboard Admin

Dashboard admin berfungsi sebagai pusat kendali bagi pengelola toko untuk memantau kondisi bisnis secara ringkas, mencakup jumlah produk, jumlah order masuk, total pendapatan, serta produk dengan penjualan terbanyak. Dashboard yang baik menyajikan informasi penting secara visual (ringkasan angka dan grafik) tanpa membebani admin dengan detail yang tidak diperlukan pada pandangan pertama.



Gambar 10. Arsitektur Sistem Aplikasi Web Retail Berbasis Laravel

Gambar 8 menunjukkan bagaimana modul dashboard admin terhubung dengan modul produk, transaksi, dan laporan yang seluruhnya bermuara pada satu basis data terpusat, memastikan data yang ditampilkan pada dashboard selalu mencerminkan kondisi terkini dari seluruh aktivitas toko.

14.2 Implementasi Ringkasan Statistik Dashboard

```
class DashboardController extends Controller
{
    public function index()
    {
        $totalProduk   = Product::count();
        $totalOrder    = Order::count();
        $totalPendapatan = Order::where('status', '!=', 'dibatalkan')-
>sum('total');
        $produkTerlaris = OrderItem::selectRaw('product_id, SUM(qty) as
total_terjual')
                            ->groupBy('product_id')
                            ->orderByDesc('total_terjual')
                            ->with('product')
                            ->take(5)
                            ->get();

        return view('admin.dashboard', compact(
            'totalProduk', 'totalOrder', 'totalPendapatan',
'produkTerlaris'
        ));
    }
}
```

14.3 Manajemen Data Master

Selain statistik, dashboard admin juga menjadi pusat pengelolaan data master seperti produk, kategori, dan data pengguna. Navigasi sidebar yang jelas dan konsisten sangat penting agar admin dapat berpindah antar modul manajemen dengan efisien.

Menu Dashboard	Fungsi Utama
Ringkasan (Overview)	Menampilkan statistik utama toko
Kelola Produk	CRUD data produk sebagaimana dibahas pada Bab 11
Kelola Kategori	CRUD data kategori produk
Kelola Order	Memantau dan memperbarui status pesanan pelanggan
Kelola Pengguna	Melihat daftar pelanggan terdaftar
Laporan Penjualan	Rekapitulasi penjualan berdasarkan periode tertentu

Tabel 26. Struktur Menu Dashboard Admin Web Retail

14.4 Laporan Penjualan

Laporan penjualan menyajikan rekapitulasi transaksi dalam rentang waktu tertentu, misalnya harian, mingguan, atau bulanan, untuk mendukung pengambilan

keputusan bisnis oleh pemilik toko. Data laporan dapat difilter berdasarkan tanggal dan diekspor ke format yang mudah dibagikan.

```
public function laporan(Request $request)
{
    $mulai  = $request->input('mulai', now()->startOfMonth());
    $selesai = $request->input('selesai', now()->endOfMonth());

    $order = Order::whereBetween('created_at', [$mulai, $selesai])
        ->where('status', '!=', 'dibatalkan')
        ->with('items.product')
        ->get();

    $totalPenjualan = $order->sum('total');

    return view('admin.laporan', compact('order', 'totalPenjualan',
'mulai', 'selesai'));
}
```

14.5 Ekspor Laporan ke Excel dan PDF

Untuk mendukung kebutuhan dokumentasi dan pelaporan kepada pemilik usaha, laporan penjualan sebaiknya dapat diekspor ke format yang umum digunakan seperti Excel atau PDF. Laravel mendukung hal ini melalui paket pihak ketiga populer seperti maatwebsite/excel untuk ekspor Excel dan barryvdh/laravel-dompdf untuk ekspor PDF.

```
composer require maatwebsite/excel
composer require barryvdh/laravel-dompdf

// Controller ekspor Excel
public function exportExcel()
{
    return Excel::download(new PenjualanExport, 'laporan-
penjualan.xlsx');
}

// Controller ekspor PDF
public function exportPdf()
{
    $order = Order::whereMonth('created_at', now()->month)->get();
    $pdf = Pdf::loadView('admin.laporan-pdf', compact('order'));
    return $pdf->download('laporan-penjualan.pdf');
}
```

14.6 Notifikasi Status Pesanan

Sebagai pelengkap dashboard admin, sistem notifikasi otomatis dapat dikirimkan kepada pelanggan setiap kali status pesanannya berubah, misalnya dari 'diproses' menjadi 'dikirim'. Laravel menyediakan fitur Notification yang mendukung pengiriman melalui berbagai kanal seperti email, database, maupun layanan pihak ketiga.

```
php artisan make:notification OrderStatusUpdated

class OrderStatusUpdated extends Notification
{
    public function __construct(public Order $order) {}

    public function via($notifiable): array { return ['mail',
'database']; }

    public function toMail($notifiable): MailMessage
    {
        return (new MailMessage)
            ->subject('Status Pesanan Anda Diperbarui')
            ->line("Pesanan {$this->order->order_no} kini berstatus:
{$this->order->status}");
    }
}

// Mengirim notifikasi saat status order diperbarui
$order->user->notify(new OrderStatusUpdated($order));
```

Rangkuman Bab 14

Dashboard admin mengintegrasikan seluruh data operasional toko dalam satu tampilan ringkas, mulai dari statistik penjualan hingga navigasi ke modul manajemen data master. Laporan penjualan berbasis rentang tanggal membantu pemilik toko memantau kinerja bisnis retail secara berkala.

Latihan Bab 14

58. Implementasikan halaman dashboard admin lengkap dengan empat kartu statistik (produk, order, pendapatan, pelanggan).
59. Tambahkan grafik penjualan bulanan pada dashboard menggunakan library chart sederhana.
60. Implementasikan fitur filter tanggal pada halaman laporan penjualan.
61. Jelaskan mengapa query produk terlaris pada bab ini menggunakan groupBy dan selectRaw.

BAB 15 DEPLOYMENT, OPTIMASI, DAN KEAMANAN APLIKASI

15.1 Persiapan Sebelum Deployment

Deployment adalah proses memindahkan aplikasi dari lingkungan pengembangan (local development) menuju lingkungan produksi (production server) agar dapat diakses oleh pengguna umum melalui internet. Sebelum melakukan deployment, terdapat sejumlah langkah persiapan penting yang harus dilakukan agar aplikasi Web Retail berjalan optimal dan aman di lingkungan produksi.

Langkah Persiapan	Tujuan
Mengatur APP_ENV=production dan APP_DEBUG=false	Mencegah kebocoran informasi teknis sensitif ke pengguna
Menjalankan php artisan config:cache	Mempercepat pembacaan konfigurasi aplikasi
Menjalankan php artisan route:cache	Mempercepat proses pencocokan routing
Menjalankan npm run build	Mengompilasi dan meminifikasi aset CSS/JS untuk produksi
Mengatur izin folder storage dan bootstrap/cache	Memastikan aplikasi dapat menulis log dan cache
Menyiapkan sertifikat SSL/TLS	Mengaktifkan HTTPS untuk keamanan transaksi

Tabel 27. Checklist Persiapan Deployment Aplikasi Laravel

```
php artisan config:cache
php artisan route:cache
php artisan view:cache
php artisan optimize
npm run build
```

15.2 Optimasi Performa Laravel

Performa aplikasi Web Retail yang baik sangat memengaruhi pengalaman berbelanja pelanggan. Beberapa teknik optimasi yang dapat diterapkan antara lain eager loading untuk menghindari N+1 query problem, penggunaan indeks basis data pada kolom yang sering difilter, penerapan caching untuk data yang jarang berubah seperti daftar kategori, serta kompresi gambar produk sebelum diunggah.

```
// Contoh caching daftar kategori selama satu jam
$kategori = Cache::remember('kategori_all', 3600, function () {
    return Category::orderBy('name')->get();
```

```
});

// Menambahkan index pada migration untuk kolom yang sering difilter
$table->index('category_id');
$table->index('status');
```

15.3 Prinsip Keamanan Dasar Aplikasi Web

Keamanan aplikasi menjadi aspek krusial pada sistem retail yang menangani data pribadi dan transaksi pelanggan. Laravel menyediakan banyak mekanisme keamanan bawaan, namun pengembang tetap perlu memahami dan menerapkan praktik terbaik keamanan secara aktif.

Ancaman Keamanan	Mekanisme Perlindungan pada Laravel
SQL Injection	Query builder dan Eloquent menggunakan parameter binding secara otomatis
Cross-Site Scripting (XSS)	Blade melakukan escaping otomatis pada output {{ }}
Cross-Site Request Forgery (CSRF)	Token @csrf wajib disertakan pada setiap form
Mass Assignment	Penggunaan atribut $fillable atau $guarded pada Model
Password Lemah	Hashing password menggunakan bcrypt/argon2 melalui Hash::make()
Unauthorized Access	Middleware auth dan otorisasi berbasis peran (role-based)

Tabel 28. Ancaman Keamanan Umum dan Mekanisme Perlindungan pada Laravel

Selain mekanisme bawaan tersebut, pengembang aplikasi Web Retail perlu menerapkan validasi input yang ketat pada seluruh form, membatasi ukuran dan tipe berkas yang dapat diunggah, menerapkan rate limiting pada endpoint sensitif seperti login untuk mencegah serangan brute force, serta secara rutin memperbarui dependensi Laravel dan paket pihak ketiga untuk menutup celah keamanan yang telah diketahui.

```
// Rate limiting pada rute login (routes/web.php)
Route::post('/login', [AuthController::class, 'login'])
     ->middleware('throttle:5,1'); // maksimum 5 percobaan per menit
```

15.4 Pilihan Strategi Hosting untuk Aplikasi Laravel

Terdapat beberapa strategi hosting yang umum digunakan untuk mempublikasikan aplikasi Laravel, masing-masing dengan karakteristik biaya, fleksibilitas, dan tingkat kerumitan pengelolaan yang berbeda-beda. Pemilihan strategi hosting yang tepat perlu mempertimbangkan skala bisnis retail yang akan dilayani.

Strategi Hosting	Karakteristik	Cocok untuk
Shared Hosting	Biaya rendah, konfigurasi terbatas, cocok untuk trafik kecil	Toko retail skala kecil/UMKM
VPS (Virtual Private Server)	Kontrol penuh atas server, memerlukan pengelolaan manual	Toko retail skala menengah dengan trafik stabil
Platform as a Service (misal Laravel Forge, Heroku)	Deployment otomatis, skalabilitas mudah	Tim pengembang yang ingin fokus pada kode, bukan infrastruktur
Cloud Native (AWS, GCP, Azure)	Skalabilitas tinggi, biaya sesuai penggunaan (pay-as-you-go)	Aplikasi retail berskala besar dengan trafik fluktuatif

Tabel 29. Perbandingan Strategi Hosting untuk Aplikasi Laravel

15.5 Pengenalan Continuous Integration/Continuous Deployment (CI/CD)

Continuous Integration/Continuous Deployment (CI/CD) adalah praktik pengembangan perangkat lunak modern yang mengotomatisasi proses pengujian dan penerapan (deployment) kode setiap kali terjadi perubahan pada repositori. Penerapan CI/CD pada proyek Web Retail dapat dilakukan menggunakan layanan seperti GitHub Actions, yang secara otomatis menjalankan pengujian dan proses build setiap kali kode baru diunggah (push) ke repositori.

```
# .github/workflows/laravel.yml (contoh sederhana)
name: Laravel CI
on: [push]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Install Dependencies
        run: composer install --no-interaction
      - name: Jalankan Pengujian
        run: php artisan test
```

Rangkuman Bab 15

Deployment aplikasi Laravel memerlukan sejumlah langkah persiapan seperti caching konfigurasi dan build aset produksi. Optimasi performa dicapai melalui eager loading, indexing, dan caching data, sedangkan keamanan aplikasi bergantung pada kombinasi mekanisme bawaan Laravel dan praktik terbaik yang diterapkan secara konsisten oleh pengembang.

Latihan Bab 15

62. Jelaskan langkah-langkah yang perlu dilakukan sebelum aplikasi Web Retail dipublikasikan ke server produksi.
63. Terapkan caching pada query daftar kategori dan produk terlaris di halaman beranda.
64. Jelaskan bagaimana Blade Template melindungi aplikasi dari serangan XSS secara otomatis.
65. Rancang mekanisme rate limiting untuk membatasi percobaan login yang berulang pada aplikasi Web Retail.

BAB 16 PENUTUP

16.1 Rangkuman Modul

Modul bahan ajar ini telah membahas secara bertahap konsep pemrograman web modern menggunakan framework Laravel, dimulai dari fondasi teori arsitektur client-server dan protokol HTTP, dilanjutkan dengan pengenalan pola MVC dan instalasi lingkungan pengembangan, hingga implementasi teknis routing, controller, model/Eloquent ORM, migration, dan Blade Templating.

Pada bagian studi kasus, seluruh konsep tersebut diintegrasikan untuk membangun aplikasi Web Retail yang fungsional, mencakup manajemen produk melalui operasi CRUD, sistem autentikasi dan otorisasi berbasis peran, modul keranjang belanja dan checkout dengan penanganan transaksi basis data yang konsisten, dashboard admin dengan laporan penjualan, serta praktik deployment, optimasi performa, dan keamanan aplikasi sebelum dipublikasikan.

Melalui pendekatan yang memadukan teori mendalam dan praktik langsung disertai ilustrasi alur proses back-end dan front-end, mahasiswa diharapkan tidak hanya memahami konsep secara konseptual, tetapi juga mampu mengimplementasikannya secara mandiri dalam proyek pengembangan aplikasi web nyata, baik dalam konteks tugas akademik maupun dunia kerja.

16.2 Kompetensi yang Diharapkan Tercapai

- Mampu merancang arsitektur aplikasi web berbasis pola MVC menggunakan Laravel.
- Mampu merancang dan mengimplementasikan basis data relasional untuk kebutuhan aplikasi retail.
- Mampu membangun antarmuka pengguna yang responsif dan konsisten dengan Blade dan framework CSS.
- Mampu mengimplementasikan sistem autentikasi, otorisasi, dan transaksi yang aman.
- Mampu menerapkan prinsip keamanan dan optimasi performa sebelum aplikasi dipublikasikan.

- Mampu bekerja secara terstruktur mengikuti konvensi framework modern dalam tim pengembangan perangkat lunak.

16.3 Proyek Akhir yang Disarankan

Sebagai penerapan menyeluruh atas seluruh materi dalam modul ini, mahasiswa disarankan mengerjakan proyek akhir berupa pengembangan aplikasi Web Retail skala penuh dengan ketentuan sebagai berikut.

66. Aplikasi memiliki modul katalog produk dengan kategori dan pencarian.
67. Aplikasi memiliki sistem autentikasi dengan dua peran: admin dan customer.
68. Aplikasi memiliki modul keranjang belanja dan proses checkout yang konsisten.
69. Aplikasi memiliki dashboard admin dengan minimal empat statistik ringkasan dan laporan penjualan berbasis rentang tanggal.
70. Aplikasi menerapkan validasi form yang lengkap pada seluruh input pengguna.
71. Aplikasi telah melalui tahap optimasi dasar (caching, eager loading) dan siap untuk proses deployment.

16.4 Daftar Pustaka

Referensi berikut digunakan sebagai rujukan teori dan praktik dalam penyusunan modul ini, dan disarankan sebagai bahan bacaan lanjutan bagi mahasiswa.

- Otwell, T. (2024). Laravel Documentation. Laravel LLC. Tersedia pada: https://laravel.com/docs
- Stauffer, M. (2019). Laravel: Up and Running — A Framework for Building Modern PHP Apps (2nd ed.). O'Reilly Media.
- Nixon, R. (2021). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (6th ed.). O'Reilly Media.

- Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Disertasi Doktoral). University of California, Irvine.
- Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.
- Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education.
- Bootstrap Team. (2024). Bootstrap Documentation. Tersedia pada: https://getbootstrap.com/docs
- Tailwind Labs. (2024). Tailwind CSS Documentation. Tersedia pada: https://tailwindcss.com/docs
- OWASP Foundation. (2023). OWASP Top Ten Web Application Security Risks. Tersedia pada: https://owasp.org/www-project-top-ten
- Rosa, A. S., & Shalahuddin, M. (2018). Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Informatika Bandung.

Ucapan Penutup

Semoga modul bahan ajar ini bermanfaat bagi mahasiswa Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Bina Bangsa, dalam memahami dan menguasai pemrograman web menggunakan Laravel secara komprehensif. Kritik dan saran membangun sangat diharapkan demi penyempurnaan modul ini pada edisi berikutnya.

LAMPIRAN A — GLOSARIUM ISTILAH

Artisan — Command Line Interface (CLI) bawaan Laravel untuk mengotomatisasi tugas pengembangan.

Blade — Templating engine bawaan Laravel untuk membangun tampilan (View).

Controller — Komponen MVC yang menangani logika permintaan dan menghubungkan Model dengan View.

Eloquent ORM — Object Relational Mapping bawaan Laravel untuk berinteraksi dengan basis data.

Middleware — Lapisan penyaring HTTP request sebelum atau sesudah diproses Controller.

Migration — Mekanisme version control untuk skema basis data berbasis kode PHP.

MVC — Model-View-Controller, pola arsitektur yang memisahkan data, tampilan, dan pengendali.

ORM — Object Relational Mapping, teknik pemetaan tabel basis data ke objek pemrograman.

Resource Controller — Controller dengan tujuh method standar untuk operasi CRUD.

Routing — Mekanisme yang menghubungkan URL dengan logika aplikasi yang menanganinya.

Seeder — Mekanisme pengisian data awal pada basis data secara terprogram.

Vite — Build tool modern yang digunakan Laravel untuk mengompilasi aset front-end.

LAMPIRAN B — BIODATA PENULIS

Nurhasan Nugroho, ST., M.Kom. merupakan tenaga pengajar pada Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Bina Bangsa. Penulis mengampu sejumlah mata kuliah pada bidang rekayasa perangkat lunak dan pemrograman web, dengan fokus perhatian pada penerapan framework modern dalam pengembangan aplikasi berbasis web untuk mendukung proses pembelajaran yang aplikatif bagi mahasiswa.

LAMPIRAN C — KODE SUMBER LENGKAP APLIKASI WEB RETAIL

Lampiran ini menyajikan kode sumber lengkap aplikasi Web Retail yang telah dibangun secara bertahap pada Bab 5 hingga Bab 14. Kode sumber disusun berdasarkan struktur direktori Laravel agar mahasiswa dapat langsung menyalin, mempelajari, dan mengembangkannya lebih lanjut sebagai bahan latihan mandiri maupun proyek akhir.

C.1 Berkas Migration Lengkap

C.1.1 create_categories_table

```
public function up(): void
{
    Schema::create('categories', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('slug')->unique();
        $table->timestamps();
    });
}

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

C.1.2 create_products_table

```
public function up(): void
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->foreignId('category_id')->constrained()-
>onDelete('cascade');
        $table->string('name');
        $table->string('slug')->unique();
        $table->text('description')->nullable();
        $table->decimal('price', 12, 2);
        $table->integer('stock')->default(0);
        $table->string('image')->nullable();
        $table->timestamps();
        $table->index('category_id');
    });
}

public function down(): void
{
    Schema::dropIfExists('products');
```

```
}
```

C.1.3 create_carts_table

```
public function up(): void
{
    Schema::create('carts', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained()->onDelete('cascade');
        $table->foreignId('product_id')->constrained()-
>onDelete('cascade');
        $table->integer('qty')->default(1);
        $table->timestamps();
        $table->unique(['user_id', 'product_id']);
    });
}
```

C.1.4 create_orders_table

```
public function up(): void
{
    Schema::create('orders', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained()->onDelete('cascade');
        $table->string('order_no')->unique();
        $table->enum('status', [
            'menunggu_pembayaran', 'diproses', 'dikirim', 'selesai',
'dibatalkan',
        ])->default('menunggu_pembayaran');
        $table->decimal('total', 14, 2);
        $table->timestamps();
        $table->index('status');
    });
}
```

C.1.5 create_order_items_table

```
public function up(): void
{
    Schema::create('order_items', function (Blueprint $table) {
        $table->id();
        $table->foreignId('order_id')->constrained()-
>onDelete('cascade');
        $table->foreignId('product_id')->constrained();
        $table->integer('qty');
        $table->decimal('subtotal', 14, 2);
        $table->timestamps();
    });
}
```

C.1.6 create_payments_table

```
public function up(): void
{
    Schema::create('payments', function (Blueprint $table) {
        $table->id();
        $table->foreignId('order_id')->constrained()-
>onDelete('cascade');
        $table->string('method');
        $table->decimal('amount', 14, 2);
        $table->enum('status', ['pending', 'berhasil', 'gagal'])-
>default('pending');
        $table->timestamps();
    });
}
```

C.2 Model Eloquent Lengkap

C.2.1 Model Category

```
class Category extends Model
{
    protected $fillable = ['name', 'slug'];

    public function products(): HasMany
    {
        return $this->hasMany(Product::class);
    }
}
```

C.2.2 Model Product

```
class Product extends Model
{
    protected $fillable = [
        'category_id', 'name', 'slug', 'description', 'price', 'stock',
'image',
    ];

    public function category(): BelongsTo
    {
        return $this->belongsTo(Category::class);
    }

    public function orderItems(): HasMany
    {
        return $this->hasMany(OrderItem::class);
    }

    protected function formattedPrice(): Attribute
    {
        return Attribute::make(
            get: fn () => 'Rp ' . number_format($this->price, 0, ',',
'.'),
        );
    }
```

```
}
```

C.2.3 Model Cart, Order, OrderItem, Payment

```
class Cart extends Model
{
    protected $fillable = ['user_id', 'product_id', 'qty'];

    public function product(): BelongsTo { return $this-
>belongsTo(Product::class); }
    public function user(): BelongsTo { return $this-
>belongsTo(User::class); }
}

class Order extends Model
{
    protected $fillable = ['user_id', 'order_no', 'status', 'total'];

    public function items(): HasMany { return $this-
>hasMany(OrderItem::class); }
    public function payment(): HasOne { return $this-
>hasOne(Payment::class); }
    public function user(): BelongsTo { return $this-
>belongsTo(User::class); }
}

class OrderItem extends Model
{
    protected $fillable = ['order_id', 'product_id', 'qty', 'subtotal'];

    public function order(): BelongsTo { return $this-
>belongsTo(Order::class); }
    public function product(): BelongsTo { return $this-
>belongsTo(Product::class); }
}

class Payment extends Model
{
    protected $fillable = ['order_id', 'method', 'amount', 'status'];

    public function order(): BelongsTo { return $this-
>belongsTo(Order::class); }
}
```

C.3 Berkas Routing Lengkap (routes/web.php)

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\{HomeController, ProductController,
CartController, OrderController};
use App\Http\Controllers\Admin\{DashboardController, ProductController as
AdminProductController};
use App\Http\Controllers\Admin\{CategoryController as
AdminCategoryController, ReportController};
```

```
Route::get('/', [HomeController::class, 'index'])->name('home');
Route::get('/produk', [ProductController::class, 'index'])-
>name('produk.index');
Route::get('/produk/{produk}', [ProductController::class, 'show'])-
>name('produk.show');

Route::middleware('auth')->group(function () {
    Route::get('/keranjang', [CartController::class, 'index'])-
>name('keranjang.index');
    Route::post('/keranjang', [CartController::class, 'store'])-
>name('keranjang.store');
    Route::patch('/keranjang/{cart}', [CartController::class, 'update'])-
>name('keranjang.update');
    Route::delete('/keranjang/{cart}', [CartController::class,
'destroy'])->name('keranjang.destroy');

    Route::post('/checkout', [OrderController::class, 'checkout'])-
>name('checkout');
    Route::get('/order/{order}/invoice', [OrderController::class,
'invoice'])->name('order.invoice');
    Route::get('/order', [OrderController::class, 'index'])-
>name('order.index');
});

Route::middleware(['auth', 'admin'])->prefix('admin')->name('admin.')-
>group(function () {
    Route::get('/dashboard', [DashboardController::class, 'index'])-
>name('dashboard');
    Route::resource('produk', AdminProductController::class);
    Route::resource('kategori', AdminCategoryController::class);
    Route::get('/laporan', [ReportController::class, 'index'])-
>name('laporan');
});

require __DIR__.'/auth.php'; // Rute autentikasi dari Laravel Breeze
```

C.4 Blade View Lengkap

C.4.1 layouts/app.blade.php

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>@yield('title', 'Web Retail Binabangsa')</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min
.css" rel="stylesheet">
</head>
<body class="bg-light">
  @include('partials.navbar')
  <main class="container my-4">
    @if (session('success'))
      <div class="alert alert-success">{{ session('success') }}</div>
    @endif
```

```
    @if (session('error'))
      <div class="alert alert-danger">{{ session('error') }}</div>
    @endif
    @yield('content')
  </main>
  @include('partials.footer')
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundl
e.min.js"></script>
</body>
</html>
```

C.4.2 produk/index.blade.php

```
@extends('layouts.app')
@section('title', 'Katalog Produk')
@section('content')
<div class="row mb-3">
  <div class="col-md-6">
    <form method="GET" action="{{ route('produk.index') }}" class="d-
flex">
      <input type="text" name="cari" class="form-control me-2"
placeholder="Cari produk..."
             value="{{ request('cari') }}">
      <button class="btn btn-primary">Cari</button>
    </form>
  </div>
</div>
<div class="row">
  @foreach ($produk as $item)
    <div class="col-md-3 mb-4">
      <div class="card h-100 shadow-sm">
        <img src="{{ $item->image ? asset('storage/'.$item->image) :
asset('img/no-image.png') }}"
             class="card-img-top" style="height:180px;object-fit:cover">
        <div class="card-body d-flex flex-column">
          <h6 class="card-title">{{ $item->name }}</h6>
          <p class="text-primary fw-bold mb-2">{{ $item->formatted_price
}}</p>
          <a href="{{ route('produk.show', $item->id) }}" class="btn btn-
sm btn-outline-primary mt-auto">
            Lihat Detail
          </a>
        </div>
      </div>
    </div>
  @endforeach
</div>
{{ $produk->links() }}
@endsection
```

C.4.3 keranjang/index.blade.php

```
@extends('layouts.app')
@section('title', 'Keranjang Belanja')
```

```
@section('content')
<table class="table align-middle">

<thead><tr><th>Produk</th><th>Harga</th><th>Qty</th><th>Subtotal</th><th>
</th></tr></thead>
  <tbody>
    @foreach ($items as $item)
      <tr>
        <td>{{ $item->product->name }}</td>
        <td>{{ $item->product->formatted_price }}</td>
        <td>
          <form method="POST" action="{{ route('keranjang.update', $item-
>id) }}" class="d-flex">
            @csrf @method('PATCH')
            <input type="number" name="qty" value="{{ $item->qty }}"
min="1" class="form-control form-control-sm" style="width:70px">
            <button class="btn btn-sm btn-outline-secondary ms-
1">Ubah</button>
          </form>
        </td>
        <td>Rp {{ number_format($item->qty * $item->product->price, 0,
',', '.') }}</td>
        <td>
          <form method="POST" action="{{ route('keranjang.destroy',
$item->id) }}">
            @csrf @method('DELETE')
            <button class="btn btn-sm btn-danger">Hapus</button>
          </form>
        </td>
      </tr>
    @endforeach
  </tbody>
</table>
<div class="text-end">
  <h5>Total: Rp {{ number_format($total, 0, ',', '.') }}</h5>
  <form method="POST" action="{{ route('checkout') }}">
    @csrf
    <button class="btn btn-success">Checkout Sekarang</button>
  </form>
</div>
@endsection
```

C.4.4 admin/dashboard.blade.php

```
@extends('layouts.admin')
@section('content')
<div class="row g-3 mb-4">
  <div class="col-md-3">
    <div class="card text-bg-primary"><div class="card-body">
      <h6>Total Produk</h6><h3>{{ $totalProduk }}</h3>
    </div></div>
  </div>
  <div class="col-md-3">
    <div class="card text-bg-success"><div class="card-body">
      <h6>Total Order</h6><h3>{{ $totalOrder }}</h3>
    </div></div>
  </div>
```

```
  <div class="col-md-3">
    <div class="card text-bg-warning"><div class="card-body">
      <h6>Total Pendapatan</h6><h4>Rp {{
number_format($totalPendapatan,0,',','.') }}</h4>
    </div></div>
  </div>
</div>
<h5>Produk Terlaris</h5>
<table class="table">
  @foreach ($produkTerlaris as $p)
    <tr><td>{{ $p->product->name }}</td><td>{{ $p->total_terjual }}
terjual</td></tr>
  @endforeach
</table>
@endsection
```

C.4.5 admin/produk/create.blade.php dan edit.blade.php

Formulir tambah dan edit produk memiliki struktur yang hampir identik, sehingga dapat digabungkan menjadi satu partial view bernama _form.blade.php yang disertakan (include) oleh kedua halaman tersebut untuk menghindari duplikasi kode.

```
{{-- resources/views/admin/produk/_form.blade.php --}}
<div class="mb-3">
  <label class="form-label">Nama Produk</label>
  <input type="text" name="name" class="form-control"
         value="{{ old('name', $produk->name ?? '') }}" required>
</div>
<div class="mb-3">
  <label class="form-label">Kategori</label>
  <select name="category_id" class="form-select" required>
    @foreach ($kategori as $k)
      <option value="{{ $k->id }}"
        @selected(old('category_id', $produk->category_id ?? '') == $k-
>id)>
        {{ $k->name }}
      </option>
    @endforeach
  </select>
</div>
<div class="mb-3">
  <label class="form-label">Harga</label>
  <input type="number" name="price" class="form-control"
         value="{{ old('price', $produk->price ?? '') }}" required>
</div>
<div class="mb-3">
  <label class="form-label">Stok</label>
  <input type="number" name="stock" class="form-control"
         value="{{ old('stock', $produk->stock ?? '') }}" required>
</div>
<div class="mb-3">
  <label class="form-label">Gambar Produk</label>
  <input type="file" name="image" class="form-control">
  @if(!empty($produk?->image))
```

```
    <img src="{{ asset('storage/'.$produk->image) }}" width="80"
class="mt-2 rounded">
  @endif
</div>
<button type="submit" class="btn btn-primary">Simpan</button>

{{-- resources/views/admin/produk/create.blade.php --}}
@extends('layouts.admin')
@section('content')
<form action="{{ route('admin.produk.store') }}" method="POST"
enctype="multipart/form-data">
  @csrf
  @include('admin.produk._form')
</form>
@endsection

{{-- resources/views/admin/produk/edit.blade.php --}}
@extends('layouts.admin')
@section('content')
<form action="{{ route('admin.produk.update', $produk->id) }}"
method="POST" enctype="multipart/form-data">
  @csrf @method('PUT')
  @include('admin.produk._form')
</form>
@endsection
```

C.4.6 admin/kategori/index.blade.php

```
@extends('layouts.admin')
@section('content')
<a href="#" class="btn btn-primary mb-3" data-bs-toggle="modal" data-bs-
target="#modalTambah">
  + Tambah Kategori
</a>
<table class="table">
  <thead><tr><th>Nama</th><th>Jumlah
Produk</th><th>Aksi</th></tr></thead>
  <tbody>
    @foreach ($kategori as $k)
      <tr>
        <td>{{ $k->name }}</td>
        <td>{{ $k->products_count }}</td>
        <td>
          <form method="POST" action="{{ route('admin.kategori.destroy',
$k->id) }}" class="d-inline">
            @csrf @method('DELETE')
            <button class="btn btn-sm btn-danger" onclick="return
confirm('Hapus kategori ini?')">
              Hapus
            </button>
          </form>
        </td>
      </tr>
    @endforeach
  </tbody>
</table>
{{ $kategori->links() }}
```

```
@endsection
```

C.4.7 admin/order/index.blade.php

```
@extends('layouts.admin')
@section('content')
<table class="table">
  <thead>
    <tr><th>No.
Order</th><th>Pelanggan</th><th>Total</th><th>Status</th><th>Aksi</th></t
r>
  </thead>
  <tbody>
    @foreach ($order as $o)
      <tr>
        <td>{{ $o->order_no }}</td>
        <td>{{ $o->user->name }}</td>
        <td>Rp {{ number_format($o->total, 0, ',', '.') }}</td>
        <td>
          <form method="POST" action="{{ route('admin.order.update', $o-
>id) }}">
            @csrf @method('PATCH')
            <select name="status" class="form-select form-select-sm"
onchange="this.form.submit()">
              @foreach
(['menunggu_pembayaran','diproses','dikirim','selesai','dibatalkan'] as
$s)
                <option value="{{ $s }}" @selected($o->status == $s)>{{
$s }}</option>
              @endforeach
            </select>
          </form>
        </td>
        <td><a href="{{ route('order.invoice', $o->id) }}" class="btn
btn-sm btn-outline-secondary">Detail</a></td>
      </tr>
    @endforeach
  </tbody>
</table>
@endsection
```

C.5 Middleware EnsureUserIsAdmin Lengkap

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class EnsureUserIsAdmin
{
    public function handle(Request $request, Closure $next): Response
```

```
    {
        if (! $request->user() || $request->user()->role !== 'admin') {
            abort(403, 'Akses ditolak. Halaman ini khusus admin.');
        }

        return $next($request);
    }
}
```

C.6 Seeder Lengkap untuk Data Awal Web Retail

```
class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $this->call([
            AdminSeeder::class,
            CategorySeeder::class,
            ProductSeeder::class,
        ]);
    }
}

class CategorySeeder extends Seeder
{
    public function run(): void
    {
        $kategori = ['Pakaian Pria', 'Pakaian Wanita', 'Sepatu',
'Aksesoris', 'Elektronik'];
        foreach ($kategori as $nama) {
            Category::create(['name' => $nama, 'slug' =>
Str::slug($nama)]);
        }
    }
}
```

Catatan Penggunaan Lampiran

Seluruh kode pada lampiran ini merupakan versi ringkas untuk kebutuhan pembelajaran. Dalam pengembangan skala produksi, disarankan menambahkan penanganan error yang lebih lengkap (try-catch), pengujian otomatis (unit test dan feature test), serta dokumentasi API apabila aplikasi juga akan diakses melalui aplikasi mobile.

LAMPIRAN D — RENCANA PEMBELAJARAN SEMESTER (RPS)

Rencana Pembelajaran Semester (RPS) berikut disusun sebagai acuan pelaksanaan perkuliahan mata kuliah Pemrograman Web selama enam belas kali pertemuan (satu semester), mengacu pada struktur bab dalam modul ini.

Minggu	Bahan Kajian	Bentuk Pembelajaran
1	Pendahuluan & Konsep Dasar Pemrograman Web (Bab 1-2)	Ceramah, diskusi
2	Pengenalan Laravel & Instalasi Lingkungan (Bab 3-4)	Ceramah, demonstrasi, praktik lab
3	Routing pada Laravel (Bab 5)	Praktik lab terbimbing
4	Controller pada Laravel (Bab 6)	Praktik lab terbimbing
5	Model dan Eloquent ORM (Bab 7)	Praktik lab terbimbing
6	Migration dan Perancangan Basis Data (Bab 8)	Praktik lab, perancangan ERD
7	View dan Blade Templating (Bab 9)	Praktik lab terbimbing
8	Ujian Tengah Semester (UTS)	Evaluasi tertulis dan praktik
9	Integrasi Front-End Bootstrap/Tailwind (Bab 10)	Praktik lab terbimbing
10	CRUD Manajemen Produk (Bab 11)	Praktik lab, studi kasus
11	Autentikasi dan Otorisasi (Bab 12)	Praktik lab terbimbing
12	Keranjang Belanja dan Checkout (Bab 13)	Praktik lab, studi kasus
13	Dashboard Admin dan Laporan (Bab 14)	Praktik lab, studi kasus
14	Deployment, Optimasi, dan Keamanan (Bab 15)	Ceramah, praktik, diskusi
15	Konsultasi dan Penyempurnaan Proyek Akhir	Bimbingan proyek kelompok/individu
16	Ujian Akhir Semester (UAS) — Presentasi Proyek Web Retail	Presentasi dan demonstrasi aplikasi

Tabel 1. Rencana Pembelajaran Semester Mata Kuliah Pemrograman Web (16 Pertemuan)

D.1 Komponen dan Bobot Penilaian

Komponen Penilaian	Bobot
Kehadiran dan Keaktifan	10%
Tugas dan Latihan Praktikum per Bab	20%
Ujian Tengah Semester (UTS)	25%
Proyek Akhir Aplikasi Web Retail	25%
Ujian Akhir Semester (UAS)	20%

Tabel 2. Komponen dan Bobot Penilaian Mata Kuliah

D.2 Rubrik Penilaian Proyek Akhir

Aspek Penilaian	Indikator	Bobot
Fungsionalitas CRUD	Modul produk, kategori, dan order berjalan tanpa error	25%
Autentikasi & Otorisasi	Login/register berjalan; hak akses admin dan customer terpisah dengan benar	20%
Alur Transaksi	Keranjang belanja dan checkout menghasilkan order yang konsisten dengan stok produk	25%
Kualitas Antarmuka	Tampilan responsif, konsisten, dan mudah digunakan	15%
Struktur Kode & Dokumentasi	Kode mengikuti konvensi MVC Laravel dan disertai dokumentasi singkat	15%

Tabel 3. Rubrik Penilaian Proyek Akhir Aplikasi Web Retail

Catatan bagi Pengajar

RPS ini bersifat fleksibel dan dapat disesuaikan dengan kalender akademik serta karakteristik kelas masing-masing. Dosen pengampu disarankan menambahkan sesi review kode (code review) pada pertemuan ke-15 guna memastikan kualitas proyek akhir mahasiswa sebelum dipresentasikan pada UAS.

LAMPIRAN E — CONTOH SOAL LATIHAN UTS DAN UAS

E.1 Contoh Soal Ujian Tengah Semester (UTS)

Bagian A — Pilihan Konsep (Uraian Singkat)

72. Jelaskan perbedaan antara Model, View, dan Controller pada pola arsitektur MVC beserta contoh penerapannya masing-masing pada aplikasi Web Retail.
73. Jelaskan fungsi Route::resource dan sebutkan ketujuh method standar yang dihasilkannya.
74. Jelaskan perbedaan antara relasi hasMany dan belongsToMany pada Eloquent ORM, disertai contoh kasus penggunaannya.
75. Jelaskan tahapan yang terjadi ketika perintah php artisan migrate dijalankan pada sebuah proyek Laravel.
76. Jelaskan fungsi directive @csrf pada form Blade dan risiko keamanan apabila directive tersebut tidak disertakan.

Bagian B — Studi Kasus Praktik

77. Rancanglah migration untuk tabel reviews yang menyimpan ulasan pelanggan terhadap produk, mencakup relasi ke tabel users dan products, kolom rating (1-5), dan kolom komentar.
78. Implementasikan Controller dan Route untuk menampilkan seluruh ulasan pada sebuah halaman detail produk, diurutkan dari yang terbaru.
79. Buatlah query Eloquent untuk menghitung rata-rata rating sebuah produk berdasarkan seluruh ulasan yang dimilikinya.

E.2 Contoh Soal Ujian Akhir Semester (UAS)

Bagian A — Uraian Konsep

80. Jelaskan alur lengkap proses checkout pada aplikasi Web Retail, mulai dari pengguna menekan tombol checkout hingga invoice diterbitkan, serta

jelaskan mengapa proses tersebut perlu dibungkus dalam sebuah database transaction.

81. Jelaskan perbedaan konsep autentikasi dan otorisasi, serta bagaimana keduanya diimplementasikan pada aplikasi Web Retail menggunakan Laravel Breeze dan middleware kustom.
82. Jelaskan minimal empat langkah optimasi performa yang dapat diterapkan sebelum aplikasi Laravel dipublikasikan ke lingkungan produksi.

Bagian B — Proyek Mini (Take Home Project)

Mahasiswa diminta mengembangkan salah satu fitur berikut pada aplikasi Web Retail yang telah dibangun sepanjang perkuliahan, disertai laporan singkat mengenai rancangan basis data, potongan kode utama, dan tangkapan layar hasil implementasi:

- Fitur ulasan dan rating produk (product review).
- Fitur diskon/kupon belanja sebagaimana dibahas pada Bab 13.
- Fitur notifikasi status pesanan melalui email menggunakan Laravel Mail.
- Fitur laporan penjualan dalam format grafik interaktif pada dashboard admin.

Catatan bagi Dosen Pengampu
Contoh soal pada lampiran ini dapat dimodifikasi sesuai tingkat kesulitan kelas dan capaian pembelajaran yang ingin diukur. Soal studi kasus praktik sebaiknya dikerjakan pada lingkungan pengembangan langsung dan dinilai berdasarkan hasil eksekusi kode, bukan hanya ketepatan sintaksis.

LAMPIRAN F — PANDUAN RINCI INSTALASI LINGKUNGAN PENGEMBANGAN

F.1 Instalasi Laragon (Windows)

Laragon merupakan salah satu perangkat lunak lingkungan pengembangan lokal (local development environment) yang populer di kalangan pengembang Laravel karena kemudahan instalasi dan kelengkapan komponennya (Apache/Nginx, PHP, MySQL, phpMyAdmin) dalam satu paket instalasi.

83. Unduh Laragon versi Full pada situs resmi laragon.org.
84. Jalankan berkas installer dan ikuti proses instalasi hingga selesai.
85. Buka aplikasi Laragon, lalu klik tombol 'Start All' untuk menjalankan Apache dan MySQL secara bersamaan.
86. Verifikasi instalasi PHP dengan membuka terminal bawaan Laragon dan menjalankan perintah php -v.
87. Buat basis data web_retail melalui menu 'Database' pada Laragon, yang akan membuka phpMyAdmin secara otomatis.

F.2 Instalasi XAMPP (Multi-Platform)

88. Unduh XAMPP sesuai sistem operasi (Windows/Linux/macOS) dari situs resmi apachefriends.org.
89. Jalankan installer dan pilih komponen minimal Apache, MySQL, dan PHP.
90. Buka XAMPP Control Panel, lalu jalankan modul Apache dan MySQL.
91. Akses http://localhost/phpmyadmin untuk membuat basis data baru bernama web_retail.

F.3 Instalasi Composer

Composer merupakan manajer dependensi PHP yang wajib terpasang sebelum melakukan instalasi Laravel. Instalasi Composer dilakukan dengan mengunduh installer resmi dari getcomposer.org dan menjalankannya, kemudian memverifikasi keberhasilan instalasi melalui command line.

```
composer --version
```

F.4 Konfigurasi Visual Studio Code untuk Pengembangan Laravel

Visual Studio Code (VS Code) menjadi salah satu editor kode yang direkomendasikan untuk pengembangan Laravel karena ringan dan didukung oleh berbagai ekstensi yang mempermudah produktivitas pengembang.

Ekstensi	Manfaat
PHP Intelephense	Autocomplete dan analisis kode PHP secara real-time
Laravel Blade Snippets	Highlight sintaksis dan snippet untuk berkas Blade
Laravel Extra Intellisense	Autocomplete route, view, dan konfigurasi Laravel
GitLens	Visualisasi riwayat perubahan kode melalui Git
Prettier	Perapian format kode HTML, CSS, dan JavaScript secara otomatis

Tabel 4. Ekstensi Visual Studio Code yang Direkomendasikan untuk Pengembangan Laravel

F.5 Troubleshooting Instalasi Umum

Kendala	Kemungkinan Penyebab	Solusi
Perintah composer tidak dikenali	Composer belum ditambahkan ke PATH sistem	Instal ulang Composer dan pastikan opsi 'Add to PATH' dicentang
Error koneksi basis data	Kredensial pada .env tidak sesuai atau MySQL belum berjalan	Periksa kembali DB_HOST, DB_USERNAME, DB_PASSWORD, dan status service MySQL
Halaman blank/500 error	APP_KEY belum di-generate	Jalankan perintah php artisan key:generate
Aset CSS/JS tidak termuat	Proses build Vite belum dijalankan	Jalankan npm install lalu npm run dev atau npm run build
Permission denied pada storage	Izin folder storage tidak sesuai (khusus Linux/macOS)	Jalankan chmod -R 775 storage bootstrap/cache

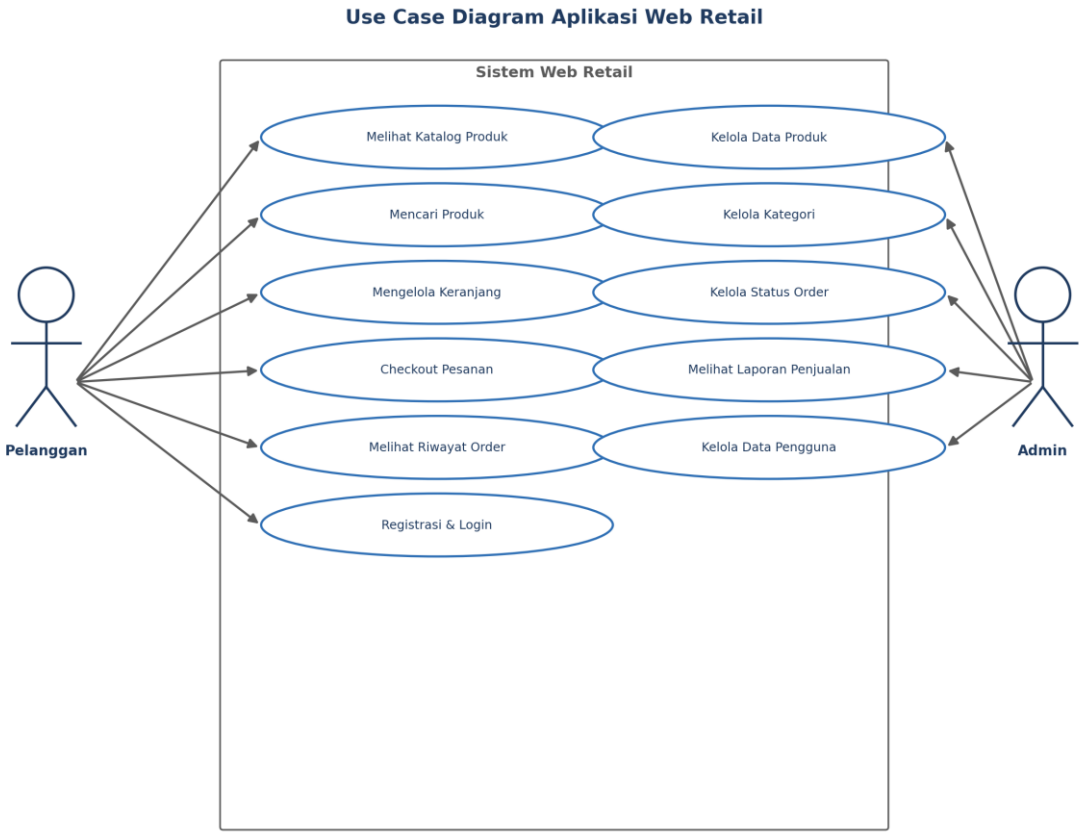
Tabel 5. Kendala Umum dan Solusi pada Instalasi Lingkungan Pengembangan Laravel

LAMPIRAN G — DIAGRAM TAMBAHAN PERANCANGAN SISTEM

Lampiran ini melengkapi diagram-diagram yang telah disajikan pada bab-bab sebelumnya dengan dua diagram perancangan sistem tambahan, yaitu Use Case Diagram yang menggambarkan interaksi aktor dengan sistem secara menyeluruh, dan Activity Diagram yang menjabarkan alur logika proses checkout secara lebih rinci dibandingkan Gambar 7 pada Bab 13.

G.1 Use Case Diagram Aplikasi Web Retail

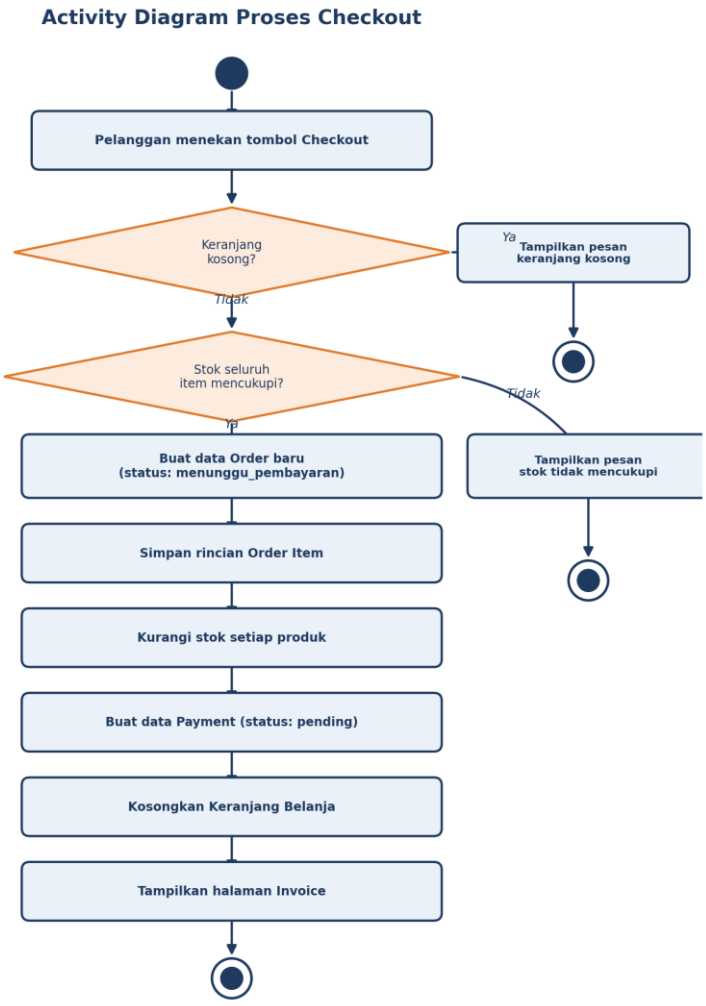
Use Case Diagram menggambarkan fungsionalitas sistem dari sudut pandang pengguna (aktor). Pada aplikasi Web Retail, terdapat dua aktor utama, yaitu Pelanggan yang berinteraksi dengan fitur belanja, dan Admin yang berinteraksi dengan fitur pengelolaan toko. Diagram ini berguna sebagai acuan awal analisis kebutuhan sistem sebelum masuk ke tahap perancangan basis data dan implementasi teknis.



Gambar 1. Use Case Diagram Aplikasi Web Retail

G.2 Activity Diagram Proses Checkout

Activity Diagram berikut menjabarkan logika keputusan (decision point) yang terjadi selama proses checkout secara lebih detail dibandingkan penjelasan naratif pada Bab 13, termasuk penanganan kondisi keranjang kosong dan stok produk yang tidak mencukupi sebelum data Order benar-benar disimpan ke basis data.



Gambar 2. Activity Diagram Proses Checkout

> **Catatan Penggunaan Diagram**
> Kedua diagram pada lampiran ini disarankan digunakan sebagai bahan diskusi kelompok sebelum mahasiswa mulai menuliskan kode program pada proyek akhir,

sebagai latihan membiasakan diri dengan tahap analisis dan perancangan sebelum implementasi (design before code).

LAMPIRAN H — STUDI KASUS PENGEMBANGAN LANJUTAN

Lampiran ini menyajikan beberapa studi kasus pengembangan lanjutan yang dapat dijadikan bahan eksplorasi mandiri oleh mahasiswa setelah menguasai seluruh materi inti pada Bab 1 hingga Bab 16, guna memperkaya kompleksitas aplikasi Web Retail menuju kebutuhan dunia nyata yang lebih menyeluruh.

H.1 Integrasi Payment Gateway

Pada implementasi dasar di Bab 13, status pembayaran dicatat secara manual oleh admin. Pada aplikasi retail produksi, integrasi dengan payment gateway pihak ketiga seperti Midtrans atau Xendit memungkinkan verifikasi pembayaran dilakukan secara otomatis dan real-time melalui mekanisme webhook.

```
composer require midtrans/midtrans-php

// Konfigurasi Midtrans pada config/services.php
'midtrans' => [
    'server_key' => env('MIDTRANS_SERVER_KEY'),
    'is_production' => env('MIDTRANS_IS_PRODUCTION', false),
],

// Membuat transaksi Snap Token
public function createSnapToken(Order $order)
{
    \Midtrans\Config::$serverKey =
config('services.midtrans.server_key');
    \Midtrans\Config::$isProduction =
config('services.midtrans.is_production');

    $params = [
        'transaction_details' => [
            'order_id'     => $order->order_no,
            'gross_amount' => (int) $order->total,
        ],
    ];

    return \Midtrans\Snap::getSnapToken($params);
}

// Endpoint webhook untuk menerima notifikasi status pembayaran
public function handleWebhook(Request $request)
{
    $notif = new \Midtrans\Notification();
    $order = Order::where('order_no', $notif->order_id)->firstOrFail();

    if ($notif->transaction_status === 'settlement') {
        $order->update(['status' => 'diproses']);
```

```
        $order->payment()->update(['status' => 'berhasil']);
    }

    return response()->json(['message' => 'OK']);
}
```

H.2 Dukungan Multi Bahasa (Localization)

Apabila aplikasi Web Retail ditargetkan untuk pasar yang lebih luas, dukungan multi bahasa menjadi pertimbangan penting. Laravel menyediakan mekanisme localization bawaan melalui berkas bahasa yang disimpan pada direktori lang, serta helper __() untuk menerjemahkan teks secara dinamis berdasarkan locale yang aktif.

```
// lang/id/retail.php
return [
    'tambah_keranjang' => 'Tambah ke Keranjang',
    'checkout'          => 'Checkout Sekarang',
];

// lang/en/retail.php
return [
    'tambah_keranjang' => 'Add to Cart',
    'checkout'          => 'Checkout Now',
];

{{-- Penggunaan pada Blade --}}
<button>{{ __('retail.tambah_keranjang') }}</button>
```

H.3 Pencarian Produk Berbasis Full-Text Search

Fitur pencarian sederhana menggunakan operator LIKE sebagaimana dicontohkan pada Bab 11 memiliki keterbatasan performa pada jumlah data yang besar. Sebagai pengembangan lanjutan, pencarian dapat ditingkatkan menggunakan mesin pencari khusus seperti Laravel Scout yang terintegrasi dengan layanan seperti Algolia atau Meilisearch untuk hasil pencarian yang lebih relevan dan responsif.

```
composer require laravel/scout
php artisan vendor:publish --
provider="Laravel\Scout\ScoutServiceProvider"

// Model Product
use Laravel\Scout\Searchable;

class Product extends Model
```

```
{
    use Searchable;

    public function toSearchableArray(): array
    {
        return [
            'name'        => $this->name,
            'description' => $this->description,
        ];
    }
}

// Penggunaan pencarian
$hasil = Product::search($request->cari)->paginate(12);
```

H.4 Perbandingan Kompleksitas Pengembangan Lanjutan

Studi Kasus	Tingkat Kompleksitas	Manfaat Bisnis Utama
Integrasi Payment Gateway	Menengah-Tinggi	Verifikasi pembayaran otomatis, mengurangi kesalahan manual
Multi Bahasa (Localization)	Rendah-Menengah	Memperluas jangkauan pasar internasional
Full-Text Search (Scout)	Menengah	Meningkatkan relevansi dan kecepatan pencarian produk
Diskon/Kupon (Bab 13.5)	Rendah	Mendorong konversi penjualan melalui promosi
Wishlist (Bab 13.6)	Rendah	Meningkatkan retensi dan keterlibatan pelanggan

Tabel 6. Perbandingan Kompleksitas dan Manfaat Studi Kasus Pengembangan Lanjutan

Latihan Lampiran H

92. Rancang alur integrasi payment gateway pada aplikasi Web Retail yang telah dibangun, mulai dari pembuatan Snap Token hingga pembaruan status order melalui webhook.
93. Tambahkan dukungan dua bahasa (Indonesia dan Inggris) pada minimal tiga halaman utama aplikasi Web Retail.
94. Bandingkan performa pencarian produk menggunakan operator LIKE dengan pendekatan Laravel Scout pada data uji berjumlah lebih dari 1000 produk.

Penutup Lampiran

Studi kasus pengembangan lanjutan pada lampiran ini bersifat opsional dan ditujukan bagi mahasiswa yang ingin memperdalam kompetensi di luar cakupan wajib mata kuliah. Dosen pengampu dapat menjadikan salah satu studi kasus ini sebagai topik tugas akhir semester bagi mahasiswa dengan minat pengembangan web tingkat lanjut.