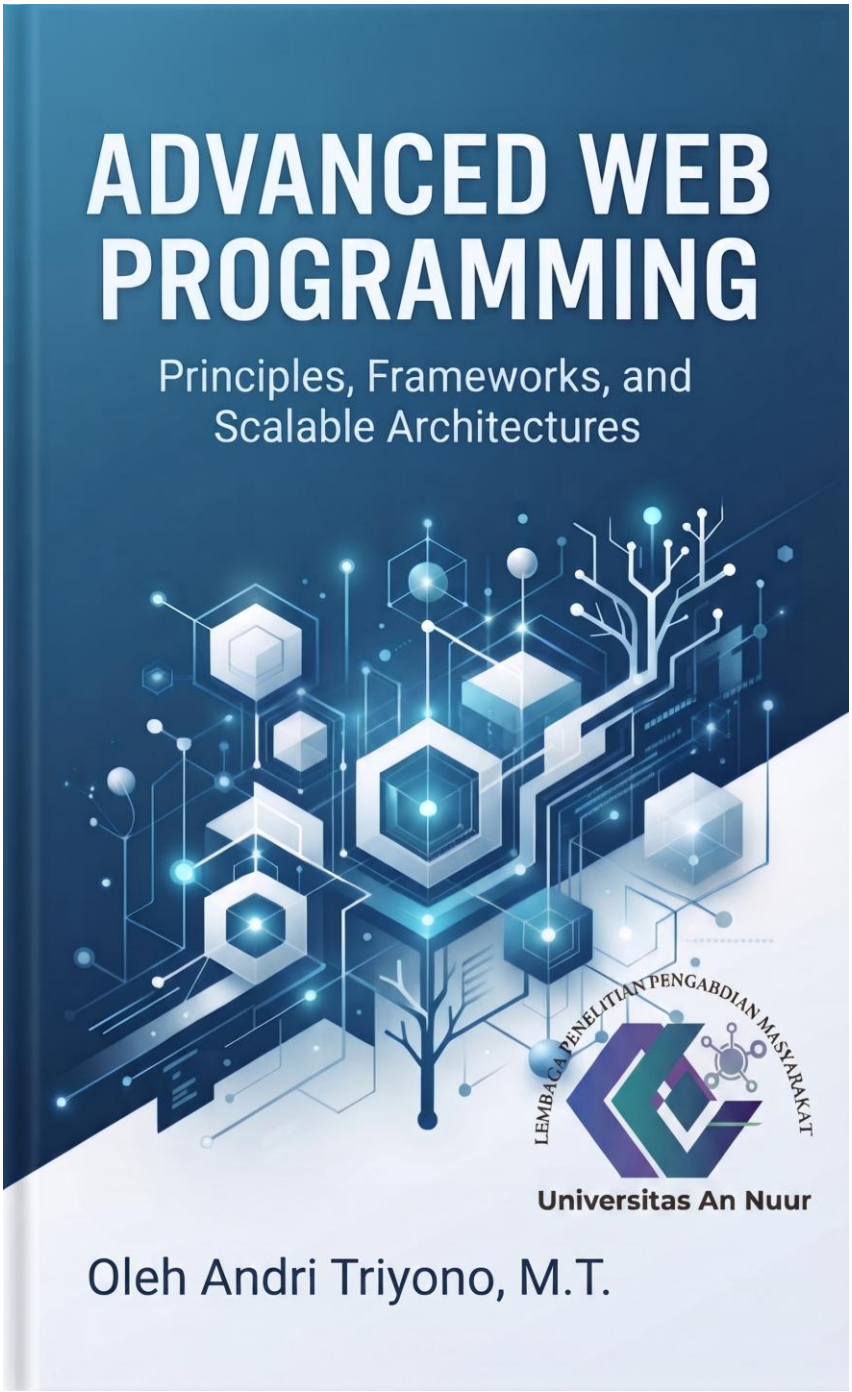

ADVANCED WEB PROGRAMMING
Principles, Frameworks, and Scalable Architectures
LEMBAGA PENELITIAN PENGABDIAN MASYARAKAT
Universitas An Nuur
Oleh Andri Triyono, M.T.

PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS SAINS DAN KESEHATAN
UNIVERSITAS AN NUUR

ADVANCED WEB PROGRAMMING

Principles, Frameworks, and Scalable Architectures

Buku Ajar Pemrograman Web Lanjut Berbasis Outcome-Based Education (OBE), Project-Based Learning (PjBL), dan Integrasi Riset Mutakhir

Penulis:
Andri Triyono, M.T.
Eko Supriyadi, S.Kom., M.Tr.Kom.
Kartika Imam Santoso, M.Kom.
Rohman Hadi Al Haq, M.Kom.
Dhika Malita Puspita A., M.Kom.
Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan
Universitas An Nuur (UNAN)

KETERANGAN IDENTITAS BUKU & HAK CIPTA

Judul Buku : Advanced Web Programming: Principles, Frameworks, and Scalable Architectures
Penulis : Andri Triyono, M.T., Eko Supriyadi, S.Kom., M.Tr.Kom., Kartika Imam Santoso, M.Kom., Rohman Hadi Al Haq, M.Kom., Dhika Malita Puspita A., M.Kom.
Editor / Penelaah : Tim Kurikulum OBE Program Studi S1 Ilmu Komputer Universitas An Nuur
Penerbit : UNAN Press (Unit Penerbitan & Publikasi Ilmiah Universitas An Nuur)
Alamat : Jl. Gajah Mada No. 10, Purwodadi, Grobogan, Jawa Tengah 58111
Edisi / Cetakan : Cetakan Pertama, Mei 2026

PURWODADI / GROBOGAN — 2026

KATA PENGANTAR

Peningkatan mutu pendidikan tinggi di era transformasi digital menuntut adanya inovasi pedagogis yang radikal, adaptif, dan berorientasi pada masa depan. Perguruan tinggi tidak lagi hanya bertugas mentransfer pengetahuan teoritis konseptual, melainkan wajib membekali mahasiswa dengan kemampuan pemecahan masalah riil di industri, ketajaman analisis, serta adaptabilitas terhadap disrupsi teknologi yang berjalan sangat masif. Selaras dengan semangat Outcome-Based Education (OBE) dan kebijakan Merdeka Belajar, proses pembelajaran harus dirancang agar berpusat pada mahasiswa melalui pengalaman praktis yang terukur.

Dalam konteks Ilmu Komputer, rekayasa perangkat lunak berbasis web merupakan salah satu pilar fundamental yang perkembangannya terjadi hampir setiap hari. Sayangnya, tantangan terbesar dalam pengajaran pemrograman web selama ini adalah adanya keterputusan linieritas antara pendalaman teori komputasi dengan eksekusi praktik rekayasa. Mahasiswa sering kali cakap dalam menulis baris kode secara terisolasi, namun gagap ketika dihadapkan pada arsitektur sistem berskala besar, optimasi performa infrastruktur server, pengamanan siber, hingga efisiensi energi komputasi.

Kehadiran buku "Pemrograman Web Lanjut Berbasis Outcome-Based Education (OBE) dan Project-Based Learning (PjBL)" yang disusun oleh Saudara Andri Triyono, M.T. ini merupakan sebuah jawaban tepat, segar, dan solutif atas tantangan tersebut. Selaku Ketua Program Studi, saya menyambut baik dan memberikan apresiasi yang setinggi-tingginya atas penerbitan buku ini.

Buku ini memiliki keunggulan akademis dan praktis yang sangat langka ditemukan pada buku-buku sejenis di pasaran. Melalui integrasi harmonis antara mata kuliah Pemrograman Web Lanjut Teori (GA07206) dan Pemrograman Web Lanjut Praktik (GA07208), penulis secara cerdas menerapkan metodologi Project-Based Learning (PjBL) melalui satu studi kasus komprehensif yang berjalan secara akumulatif, yaitu proyek EcoTask. Mahasiswa dituntun secara metodologis untuk membangun produk dari tingkat dasar, memahami mekanika internal runtime engine, bermigrasi ke framework modern, mengelola basis data, melakukan kontainerisasi, hingga menyentuh ranah teknologi masa depan seperti Edge Computing, On-Device Artificial Intelligence, dan Green Computing.

Penulis buku ini, Saudara Andri Triyono, M.T., merupakan dosen tetap pada Program Studi S1 Ilmu Komputer sekaligus Kepala UPT Sistem Informasi Universitas An Nuur. Kombinasi kepakaran akademis di ruang kelas serta pengalaman praktis beliau dalam mengelola dan mengorkestrasi infrastruktur server riil institusi, membuat kedalaman materi dalam buku ini terasa sangat bernyawa, berbasis industri, namun tetap kukuh secara fondasi sains komputasi.

Buku ini tidak hanya menjadi instrumen krusial dalam pemenuhan Indikator Kinerja Utama (IKU) universitas dalam hal pembelajaran berbasis proyek yang relevan dengan kebutuhan zaman, tetapi juga menjadi modal berharga bagi mahasiswa untuk membangun portofolio rekayasa perangkat lunak yang kompetitif di tingkat global.

Saya merekomendasikan buku ini sebagai pegangan wajib bagi mahasiswa, referensi utama bagi dosen pengampu, serta inspirasi bagi praktisi yang ingin mendalami rekayasa web modern secara struktural. Semoga karya ilmiah yang berbobot ini dapat memberikan kontribusi signifikan bagi kemajuan kualitas

lulusan Ilmu Komputer, khususnya di lingkungan Universitas An Nuur, serta memicu lahirnya inovator-inovator digital baru di tanah air.

Purwodadi, Mei 2026

Eko Supriyadi, M.Kom.
Ketua Program Studi S1 Ilmu Komputer Universitas An Nuur

PRAKATA PENULIS

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas rahmat, hidayah, dan karunia-Nya, buku yang berjudul "Pemrograman Web Lanjut Berbasis Outcome-Based Education (OBE) dan Project-Based Learning (PjBL)" ini dapat diselesaikan dengan baik. Buku ini disusun khusus untuk menjadi panduan utama sekaligus kompas akademis bagi mahasiswa program studi S1 Ilmu Komputer, Universitas An Nuur, dalam menempuh mata kuliah Pemrograman Web Lanjut Teori (GA07206) dan Pemrograman Web Lanjut Praktik (GA07208).

Lanskap rekayasa perangkat lunak berbasis web telah mengalami evolusi yang luar biasa masif dalam beberapa tahun terakhir. Pergeseran dari aplikasi monolitik statis menuju arsitektur terdistribusi, reaktif, cerdas, hingga komputasi hijau (green computing) menuntut adanya rekonstruksi radikal dalam cara kita mengajarkan pemrograman web di perguruan tinggi. Seringkali, terdapat celah pemisah (gap) yang cukup lebar antara pemahaman teoritis ilmu komputasi di ruang kelas dengan keterampilan teknis rekayasa riil yang dibutuhkan oleh industri global. Mahasiswa sering kali terjebak dalam pola "meniru kode" (code-soup) tanpa benar-benar memahami apa yang terjadi di balik layar pada alokasi memori biner, siklus event loop, maupun efisiensi infrastruktur server.

Buku ini hadir dengan membawa misi tunggal: mengeliminasi celah tersebut. Melalui integrasi kurikulum berbasis capaian (Outcome-Based Education) dan pembelajaran berbasis proyek (Project-Based Learning), buku ini mengadopsi pendekatan incremental development. Sepanjang membaca dan mempraktikkan isi buku ini, mahasiswa tidak akan mempelajari potongan kode yang terisolasi. Sebaliknya, pembaca ditantang untuk membangun satu produk perangkat lunak utuh berskala luas yang diberi nama EcoTask—sebuah aplikasi manajemen tugas produktivitas modern yang dirancang adaptif, cerdas, dan efisien. Proyek ini akan bertumbuh secara bertahap dari Bab 1 hingga Bab 14, selaras dengan kompetensi baru yang dipelajari.

Struktur materi dalam buku ini memandu pembaca melintasi empat fase evolusi teknologi:
1. Fondasi Arsitektur Klien: Membedah mekanika tingkat rendah (low-level mechanics) dari Browser Runtime Engine, manipulasi DOM yang optimal, serta desain responsif tingkat lanjut.
2. Transformasi Framework & Integrasi Data: Bermigrasi secara deklaratif menuju Single Page Application (SPA) menggunakan framework modern, membangun RESTful API yang kukuh dengan Node.js/Express, serta mengoptimalkan transaksi basis data SQL dan NoSQL.
3. Optimasi, Keamanan, & Skalabilitas Cloud: Mengaudit performa melalui Core Web Vitals, memitigasi celah keamanan siber berdasarkan standar OWASP Top 10, hingga melakukan kontainerisasi menggunakan Docker dan mengotomatisasi pipa rilis CI/CD.
4. Integrasi Teknologi Cerdas & Berkelanjutan: Membawa kapabilitas masa depan ke dalam aplikasi melalui Progressive Web Apps (PWA), Edge Computing, On-Device Web Machine Learning (Agentic Workflows), hingga tanggung jawab moral ekologis melalui tata cara kalkulasi emisi karbon digital (Green Computing).

Keberhasilan penyusunan buku ini tidak lepas dari dukungan, bimbingan, dan kerja sama dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan apresiasi dan terima kasih yang sebesar-besarnya kepada:

1. Rektorat dan seluruh jajaran pimpinan Universitas An Nuur (UNAN) atas atmosfer akademik yang kondusif serta dukungan fasilitas yang diberikan kepada penulis.
2. Sejawat dosen di program studi S1 Ilmu Komputer Universitas An Nuur, khususnya kepada Bapak Eko Supriyadi, M.Kom., selaku Ketua Program Studi, atas diskusi, masukan, dan sinergi orkestrasi kurikulum yang tiada henti.
3. Rekan-rekan di Unit Pelaksana Teknis Sistem Informasi (UPT SI) Universitas An Nuur, yang selalu menjadi laboratorium hidup dan rekan bertukar pikiran dalam mengimplementasikan teknologi server riil di lingkungan kampus.
4. Istri tercinta, Mega Maelani, serta keluarga besar yang senantiasa memberikan doa, kesabaran, pengorbanan, dan dukungan moral yang luar biasa di setiap malam-malam panjang penulisan draf buku ini.
5. Seluruh mahasiswa S1 Ilmu Komputer Universitas An Nuur, yang menjadi inspirasi dan energi utama penulis untuk terus menyederhanakan konsep-konsep komputasi yang rumit agar menjadi materi yang hidup, aplikatif, dan menyenangkan.

Penulis menyadari bahwa buku ini masih jauh dari kesempurnaan, mengingat dinamisnya perkembangan teknologi web yang terjadi setiap harinya. Oleh karena itu, segala kritik, saran, dan masukan yang membangun dari rekan sejawat dosen, praktisi industri, maupun mahasiswa pembaca sangat penulis harapkan demi penyempurnaan buku ini di edisi masa depan.

Akhir kata, semoga buku ini tidak hanya menjadi sekadar referensi teks, melainkan mampu menjadi pemantik api kreativitas, ketajaman berpikir logis, dan semangat rekayasa bagi para calon software engineer dalam membangun masa depan digital yang cerdas, aman, dan berkelanjutan.

Purwodadi, Mei 2026

Andri Triyono, M.T.
NIDN. 0620068802
Dosen S1 Ilmu Komputer & Kepala UPT Sistem Informasi Universitas An Nuur

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAGIAN I: FONDASI ARSITEKTUR & EKOSISTEM FULL-STACK MODERN

BAB 1
PARADIGMA & EVOLUSI ARSITEKTUR WEB MODERN: DARI MONOLITH, MICROSERVICES, HINGGA SERVERLESS

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab pembuka ini dirancang sebagai landasan arsitektural komprehensif bagi pembaca lintas jenjang kompetensi—mulai dari pembaca umum, mahasiswa strata satu, hingga praktisi industri perangkat lunak. Pembelajaran disusun mengikuti standar Outcome-Based Education (OBE) Kurikulum 2026 Program Studi S1 Ilmu Komputer Universitas An Nuur.

Tabel 1.1: *Matriks Keselarasan Konstruktif OBE (Constructive Alignment) Capaian Pembelajaran Bab 1.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software Quality Assurance Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi kebutuhan computing yang efisien dan arsitektur solusi sistem perangkat lunak modern.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu merancang, menganalisis performa, dan memilih pola arsitektur web yang skalabel dan hemat sumber daya.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.1: Mampu menganalisis komparatif arsitektur Monolith, Microservices, dan Serverless/FaaS serta merumuskan strategi dekomposisi sistem (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

1. Mahasiswa dan pembaca umum mampu mengidentifikasi karakteristik distingtif, kelebihan, serta kelemahan struktural dari arsitektur Monolith, Microservices, dan Serverless.

2. Mampu mengevaluasi metrik kinerja sistem (latensi jaringan, throughput transaksi, konsumsi memori/energi, dan cold start delay) pada berbagai skala beban komputasi.

3. Mampu merancang strategi migrasi dekomposisi bertahap (Strangler Fig Pattern) dan Domain-Driven Design (DDD) untuk aplikasi berskala enterprise.

2. Pengantar & Konsep Teoretis: Evolusi Ekosistem Web

2.1 Paradigma Fondasi: Arsitektur Monolitik (Monolith)

Dalam fase awal rekayasa perangkat lunak web, arsitektur monolitik merupakan standar utama industri. Seluruh logika bisnis (business logic), antarmuka pengguna (user interface), autentikasi, manajemen sesi, hingga lapisan akses basis data (data access layer) dikemas menjadi satu kesatuan unit kompilasi dan dieksekusi dalam satu runtime process tunggal.

Analogi sederhana untuk memahami sistem monolitik adalah seperti sebuah 'Restoran All-in-One Konvensional' berskala kecil, di mana koki, kasir, pelayan, dan manajer operasional berada di dalam satu ruangan dapur yang sama. Ketika jumlah pesanan meningkat drastis di kasir, koki dan pelayan ikut terganggu karena ruangan menjadi sesak dan antrean bertumpuk pada satu pintu keluar.

Keunggulan Utama: Kelebihan utama monolitik terletak pada kesederhanaan pengembangan awal (simplicity), nihilnya overhead jaringan internal (in-memory function call dengan latensi nanodetik), dan jaminan integritas transaksional ACID lokal yang sangat mudah diimplementasikan.

Kelemahan Kritis: Namun, monolitik menghadapi tantangan bottleneck berat saat aplikasi membesar: keterikatan kode tinggi (tight coupling), skalabilitas tidak efisien (all-or-nothing scaling), serta risiko Single Point of Failure (SPoF) di mana bug memori pada satu modul dapat meruntuhkan seluruh sistem server.

2.2 Revolusi Arsitektur Microservices

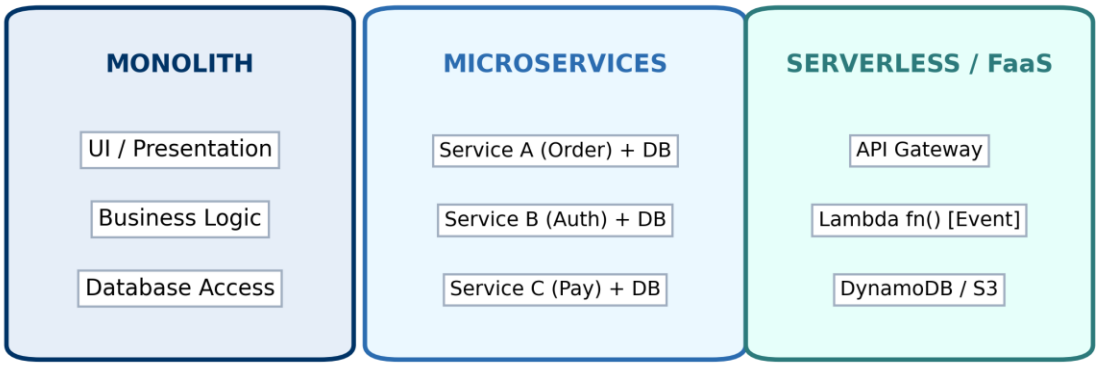
Untuk menjawab bottleneck monolitik, arsitektur Microservices memecah aplikasi menjadi kumpulan layanan independen berskala kecil (loosely-coupled services). Setiap mikroservis bertanggung jawab atas domain bisnis tertentu (Single Responsibility Principle), mengelola basis data mandiri (Database-per-Service), dan berinteraksi via protokol jaringan ringan seperti RESTful API, gRPC, atau asynchronous message broker (Kafka/RabbitMQ).

Microservices memberikan otonomi teknologi (polyglot architecture), isolasi kegagalan (fault isolation), dan skalabilitas granular di mana modul dengan beban komputasi tinggi dapat direplikasi secara independen menggunakan orkestrasi kontainer (Docker & Kubernetes).

2.3 Paradigma Serverless & Function-as-a-Service (FaaS)

Komputasi awan melahirkan Serverless Computing di mana pengembang tidak lagi mengelola atau memelihara infrastruktur server. Logika aplikasi didekomposisi menjadi fungsi-fungsi atomik (FaaS) yang dieksekusi berdasarkan event (event-driven execution).

Ciri khas Serverless adalah penagihan murni berbasis pemakaian nyata (pay-as-you-go per millisecond) dan kemampuan auto-scaling dari nol hingga ribuan request seketika. Namun, arsitektur ini memunculkan tantangan Cold Start Latency dan tuntutan desain aplikasi yang nir-status (stateless).



Gambar 1.1: *Evolusi Paradigma Arsitektur Web Modern: Monolith, Microservices, dan Serverless FaaS.*

3. Analisis Komparatif Mendalam & Trade-Off Arsitektural (State-of-the-Art Scopus)

Berdasarkan meta-analisis literatur ilmiah komputasi awan Scopus terkini [3], [4], [5], [6] (Villamizar et al., 2015; Bajaj et al., 2020; Shrestha & Nisha, 2022; Saloran & Albarda, 2025; Joselyne et al., 2026), keputusan adopsi arsitektur bukanlah sekadar mengikuti tren industri, melainkan proses kalkulasi trade-off teknis yang presisi.

Tabel 1.2: *Matriks Analisis Komparatif Performa Paradigma Arsitektur Web: Monolith vs Microservices vs Serverless FaaS.*

Parameter Komparasi	Monolithic Architecture	Microservices Architecture	Serverless / FaaS
Kompleksitas Deployment	Rendah: Single artifact (JAR, WAR, Docker image tunggal).	Tinggi: Puluhan pipeline CI/CD, orkestrator Kubernetes.	Moderat: Manajemen konfigurasi fungsi & API Gateway.
Overhead Komunikasi	Nol (Direct In-Memory function execution).	Tinggi (Network I/O latency, JSON marshaling, TLS handshake).	Tinggi (HTTP invocations, cold-start latency 50-500ms).
Konsistensi Data	Kuat (ACID transactions pada single relational DB).	Konsistensi Bertahap (Eventual Consistency & Saga Pattern).	Konsistensi Bertahap (Stateless & Distributed Data Stores).
Efisiensi Biaya Operasional	Tinggi pada beban rendah/statis; Mahal saat scaling masif.	Tinggi untuk overhead cluster; Efisien pada throughput masif stabil.	Optimal untuk trafik fluktuatif (Pay-per-Execution, Zero Idle Cost).
Observabilitas & Debugging	Mudah: Single log file & stack trace lokal.	Sangat Rumit: Butuh Distributed Tracing (Jaeger,	Tantangan: Ketergantungan log vendor cloud

		OpenTelemetry).	(CloudWatch, DataDog).
Profil Organisasi & Tim	Cocok untuk tim kecil (1-10 insinyur) & produk tahap MVP.	Cocok untuk tim besar (>30 insinyur) terorganisasi DDD.	Cocok untuk tim lincah, prototipe cepat, & asynchronous task.

3.1 Strategi Dekomposisi: Domain-Driven Design (DDD) & Bounded Contexts

Dekomposisi sistem tidak boleh dilakukan sembarangan berdasarkan entitas tabel basis data karena dapat memicu 'Distributed Monolith'. Metodologi yang diakui secara ilmiah adalah Domain-Driven Design (DDD) dengan mengidentifikasi Bounded Contexts—batas logis di mana model domain bisnis berlaku secara terisolasi dan konsisten.

3.2 Pola Migrasi Evolusioner: Strangler Fig Pattern

Daripada melakukan penulisan ulang sistem secara radikal yang berisiko tinggi (Big Bang Rewrite), industri modern menerapkan Strangler Fig Pattern. Dengan memasang API Gateway di depan sistem lama, request diarahkan bertahap ke layanan microservices baru hingga sistem monolitik lama tergantikan sepenuhnya tanpa downtime.

> **DEEP DIVE: TEKNO-STRATEGI ARSITEK SISTEM PRODUKSI**
>
> 1. Fallacies of Distributed Computing: Jaringan tidak pernah 100% andal, latensi tidak pernah nol, dan bandwidth tidak terbatas.
>
> 2. Teorema CAP: Dalam sistem terdistribusi saat terjadi partisi jaringan (P), arsitek harus memilih antara konsistensi data absolut (C) atau ketersediaan tinggi (A).
>
> 3. Distributed Transactions: Hindari two-phase commit (2PC) sinkron; gunakan Saga Orchestration/Choreography berbasis Event Broker dengan compensating transactions.

4. Penerapan Praktis & Rekayasa Kode (Hands-on Engineering)

Berikut implementasi perbandingan logika pemrosesan pesanan (Order Management) dalam 3 paradigma menggunakan TypeScript dan Node.js:

Listing Program: Monolithic ACID Transaction Flow

```
// Monolithic Modular: Order Processing with ACID Transaction
export class OrderService {
  constructor(private db: Database, private inventory: InventoryService) {}

  public async processOrder(dto: CreateOrderDto) {
    return await this.db.transaction(async (trx) => {
      const stockOk = await this.inventory.reserveStock(dto.productId, dto.qty, trx);
      if (!stockOk) throw new Error("Stok tidak mencukupi");
      const orderId = await trx.insert('orders', { userId: dto.userId, total: dto.total });
      return { orderId, status: 'CONFIRMED' };
```

```
    });
  }
}
```

Listing Program: Microservices Event-Driven Publishing Handler

```
// Microservices: Asynchronous Event Publisher (Event-Driven)
export const handleCreateOrder = async (req: Request, res: Response) => {
  const orderRepo = new OrderRepository();
  const eventPublisher = new EventPublisher('order-events');

  const order = await orderRepo.create({ ...req.body, status: 'PENDING_VALIDATION' });
  await eventPublisher.publish('order.created', { orderId: order.id, items: order.items });

  return res.status(202).json({
    message: "Order diterima dan diproses asinkron oleh Saga Choreography",
    orderId: order.id
  });
};
```

Listing Program: Serverless AWS Lambda Handler

```
// Serverless FaaS: AWS Lambda Handler with DynamoDB
export const handler = async (event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult>
=> {
  const payload = JSON.parse(event.body || '{}');
  const orderId = randomUUID();
  await docClient.send(new PutCommand({
    TableName: 'OrdersTable',
    Item: { orderId, userId: payload.userId, items: payload.items, createdAt: new
Date().toISOString() }
  }));
  return { statusCode: 201, body: JSON.stringify({ message: "Order created via FaaS",
orderId }) };
};
```

5. Studi Kasus & Problem-Based Learning (PBL)

Skenario Kasus PBL: Startup e-commerce 'NuurCommerce' mengalami lonjakan trafik 1500% saat Flash Sale. Server monolitik mengalami deadlock basis data MySQL dan downtime selama 45 menit. Mahasiswa diminta melakukan Root Cause Analysis, merancang arsitektur transisi berbasis Strangler Fig Pattern, serta memisahkan modul katalog dan checkout ke microservices/FaaS berdaya tahan tinggi.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Tabel 1.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK101.1 (Bloom C3–C6).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor	Kurang (Skor < 70)

		70-84)	
Analisis Komparasi Arsitektur (C4)	Mampu membandingkan trade-off ketiga arsitektur secara mendalam dengan metrik latensi, biaya, dan reliabilitas.	Menjelaskan perbedaan dasar arsitektur namun kurang mendalam pada analisis trade-off sistem terdistribusi.	Hanya mampu mendefinisikan istilah tanpa memahami trade-off sistem.
Perancangan Dekomposisi Sistem (C5/C6)	Merancang skema dekomposisi berbasis DDD Bounded Contexts dan Strangler Fig Pattern yang modular dan zero-downtime.	Merancang pemisahan layanan namun masih terdapat tight coupling pada basis data.	Pemisahan sistem dilakukan secara serampangan tanpa metodologi domain.
Kualitas Kode & Modularitas (P4)	Kode program clean, modular, type-safe (TypeScript), menangani failover dan exception boundary tangguh.	Kode berjalan namun penanganan konkurensi dan error handling masih parsial.	Kode gagal dikompilasi atau memiliki dependensi sirkular.
Bukti Portofolio (Evidence-Based)	Dokumentasi arsitektur lengkap, diagram C4 Model, benchmark latensi, dan repositori GitHub aktif.	Menyertakan repositori kode namun dokumentasi dan analisis trade-off minim.	Tidak ada repositori atau artefak yang dapat diuji.

7. Rangkuman Materi & Glosarium Istilah

1. Tidak ada arsitektur universal sempurna; arsitektur yang baik adalah kompromi yang tepat antara skala beban, ukuran tim, dan toleransi latensi.

2. Monolith ideal untuk kesederhanaan dan fase awal produk, Microservices ideal untuk organisasi skala besar dengan domain kompleks, dan Serverless ideal untuk skalabilitas elastis berbasis event.

7.1 Glosarium Istilah Kunci

• ACID: Prinsip Atomicity, Consistency, Isolation, Durability pada transaksi database.

• Bounded Context: Batas logis domain bisnis dalam metodologi Domain-Driven Design.

• Cold Start: Waktu inisiasi runtime kontainer baru pada arsitektur Serverless FaaS.

• Strangler Fig Pattern: Pola migrasi arsitektur bertahap untuk menggantikan monolitik tanpa downtime.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Mengapa adopsi microservices dengan single shared database justru melahirkan Distributed Monolith? Jelaskan 3 dampak negatifnya terhadap skalabilitas dan deployability!

2. [Evaluasi - C5] Evaluasi pemilihan arsitektur untuk sistem transaksi finansial perbankan ditinjau dari Teorema CAP dan integritas ACID!

3. [Kreasi - C6] Rancanglah alur migrasi modul inventaris dari monolitik lama ke microservices menggunakan Strangler Fig Pattern dan API Gateway!

BAB 2
EKOSISTEM JAVASCRIPT/TYPESCRIPT TINGKAT LANJUT & PEMROGRAMAN ASINKRON (ASYNC FLOW, EVENT LOOP, & TYPE SAFETY)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini memberikan pemahaman mendalam tentang arsitektur internal runtime JavaScript modern, eksekusi asinkron non-blocking berbasis Libuv, serta rekayasa sistem tipe data tingkat lanjut menggunakan TypeScript. Penguasaan bab ini membekali mahasiswa dan praktisi untuk merekayasa kode backend dan frontend yang berkinerja tinggi, bebas dari race condition, serta memiliki keandalan tipe (type safety) yang terverifikasi sejak masa kompilasi.

Tabel 2.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 2.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software Quality Assurance Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi kebutuhan computing yang efisien dan arsitektur solusi sistem perangkat lunak modern.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu mengimplementasikan logika komputasi asinkron dan sistem tipe data yang aman serta bebas race-condition.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.1: Mampu merekayasa alur asinkron kompleks, concurrency controls, dan TypeScript Generics tingkat lanjut (Taksonomi Bloom: C3 - Penerapan & C4 - Analisis).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu membedah dan menganalisis mekanisme kerja internal V8 Engine, Call Stack, Memory Heap, Microtask Queue, Macrotask Queue, dan tahapan Event Loop.

2. Mampu mengelola alur asinkron paralel dan kompetitif menggunakan Promise Combinators (all, allSettled, race, any) dan Async/Await dengan penanganan error yang deterministik.

3. Mampu membangun sistem tipe data tangguh berbasis TypeScript Advanced Generics, Discriminated Unions, Type Narrowing, dan Conditional Types guna mengeliminasi runtime type error.

2. Pengantar & Konsep Teoretis: Arsitektur Runtime JavaScript Modern

2.1 Anatomi Runtime V8 Engine & Single-Threaded Non-Blocking I/O

JavaScript pada awalnya dirancang sebagai bahasa skrip peramban yang sederhana. Namun, dengan hadirnya mesin kompilasi Just-In-Time (JIT) V8 oleh Google dan runtime Node.js oleh Ryan Dahl, JavaScript bertransformasi menjadi platform komputasi server-side berkemampuan tinggi. V8 mengeksekusi kode JavaScript secara single-threaded, artinya hanya ada satu Call Stack yang mengeksekusi instruksi pada satu waktu.

Peran Libuv: Untuk melayani ribuan koneksi jaringan secara bersamaan tanpa memblokir thread eksekusi utama [7], [8], Node.js mengintegrasikan pustaka Libuv (ditulis dalam C++). Libuv menyediakan Thread Pool berbasis sistem operasi (default 4 thread) untuk menangani operasi I/O yang bersifat blocking, seperti pembacaan sistem berkas (file system), kueri basis data relasional, kompresi berkas, dan enkripsi kriptografi.

Analogi Kasir & Dapur: Analogi intuitif: Bayangkan sebuah restoran cepat saji dengan 1 kasir utama (Single Thread). Kasir menerima pesanan dari pelanggan, mencetak tiket antrean berharga (Promise), lalu menyerahkan tiket tersebut ke koki dapur (Thread Pool Libuv). Kasir tidak menunggu ayam digoreng hingga matang, melainkan langsung melayani pelanggan berikutnya di antrean. Ketika makanan telah siap, staf dapur memasukkan nomor tiket ke papan pengumuman (Task Queue). Kasir akan menyerahkan makanan kepada pelanggan hanya saat sedang tidak ada transaksi kasir yang aktif.

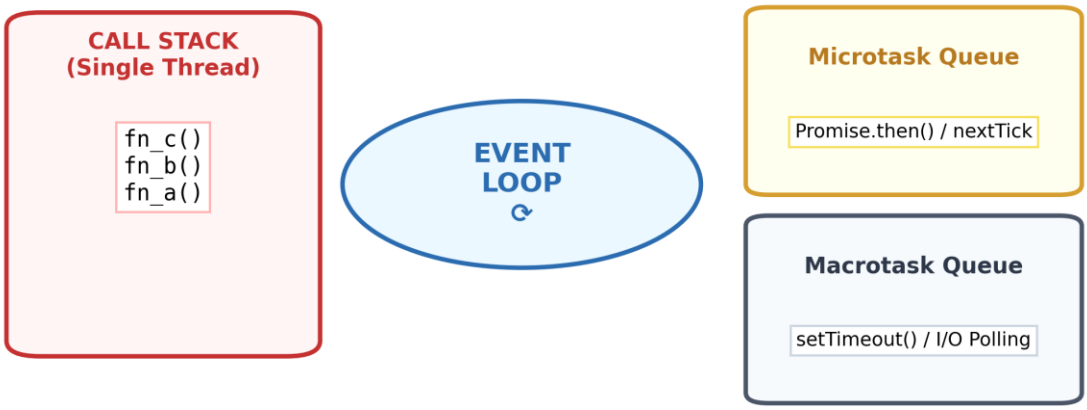
2.2 Hierarki Antrean Event Loop: Microtasks vs Macrotasks

Fase Eksekusi Event Loop: Event Loop bertindak sebagai konduktor orkestra yang terus-menerus memeriksa apakah Call Stack telah kosong. Jika Call Stack kosong, Event Loop akan mengambil fungsi callback dari antrean tugas berdasarkan urutan prioritas yang sangat ketat:

1. Synchronous Frame: Semua kode sinkron yang berada pada Call Stack dieksekusi secara linear hingga stack benar-benar kosong.

2. Microtask Queue (Prioritas Eksekutif): Antrean ini menampung callback dari process.nextTick() (pada Node.js) dan Promise lifecycle (.then(), .catch(), .finally()), serta queueMicrotask() dan MutationObserver. Seluruh tugas di dalam Microtask Queue wajib dihabiskan sampai tuntas sebelum Event Loop diizinkan berpindah ke fase lain.

3. Macrotask Queue / Task Queue: Menampung callback dari timer (setTimeout, setInterval), operasi I/O selesai, dan setImmediate().



Gambar 2.1: *Arsitektur Runtime Node.js: Call Stack, Libuv Event Loop, Microtask Queue, dan Macrotask Queue.*

3. Deep Dive Arsitektur & Concurrency Control (Expert & Production Level)

Dalam aplikasi tingkat produksi berskala enterprise, pemahaman yang dangkal terhadap model asinkron sering kali mengakibatkan bug kritis yang sulit direproduksi, seperti Event Loop Starvation, Unhandled Promise Rejections, dan Memory Leak pada closure.

Tabel 2.2: *Matriks Analisis Komparatif Paradigma Penanganan Asinkron: Callback Hell vs Promise Chaining vs Async/Await.*

Pola Asinkron	Mekanisme Eksekusi	Penanganan Kegagalan (Failure Behavior)	Use Case Optimal
Promise.all()	Paralel / Konkuren Penuh	Fail-Fast: Batal seketika jika ada satu promise yang mengalami rejection.	Agregasi dependensi wajib (Fetch User Profile + Fetch Role Permissions).
Promise.allSettled()	Paralel / Konkuren Penuh	Toleran Penuh: Menunggu seluruh promise selesai (baik resolve maupun reject).	Batch processing notifikasi masif (kirim email ke 10.000 pelanggan).
Promise.race()	Kompetitif (First-Settled)	Mengambil hasil promise pertama yang selesai (baik resolve maupun reject).	Penerapan timeout circuit breaker pada pemanggilan remote API.
Promise.any()	Kompetitif (First-Resolved)	Mengabaikan rejection dan mengambil resolve pertama; Reject jika semua gagal.	Query multi-datacenter DNS fallback atau multi-CDN image asset fetch.

3.1 Bahaya Event Loop Starvation & CPU-Bound Bottlenecks

Karena JavaScript berjalan pada single-thread, eksekusi operasi matematika yang sangat berat (seperti komputasi matriks besar, enkripsi data 500MB secara sinkron, atau infinite loop) akan membekukan (freeze) Call Stack. Selama Call Stack terblokir, Event Loop tidak dapat memproses Microtask maupun Macrotask, menyebabkan server berhenti merespons seluruh request HTTP dari pengguna lain.

Solusi Produksi: Mitigasi produksi: Untuk operasi yang bersifat CPU-Bound, pengembang wajib mendelegasikannya ke Worker Threads (modul worker_threads pada Node.js) atau arsitektur antrean asinkron berbasis proses mandiri (Child Process / Background Workers).

> **DEEP DIVE: PERANGKAP ASINKRON & PRAKTIK TERBAIK INDUSTRI**
>
> 1. Hindari Async/Await di dalam Array.prototype.forEach(): forEach tidak menunggu Promise selesai, sehingga seluruh iterasi berjalan secara tak terkontrol. Gunakan for...of loop untuk eksekusi sekuensial, atau Promise.all(array.map(...)) untuk eksekusi konkuren terencana.
>
> 2. Node.js Unhandled Rejections: Pada versi Node.js terkini (v16+), Promise rejection yang tidak ditangkap oleh blok .catch() atau try-catch akan memicu event 'unhandledRejection' dan secara otomatis menghentikan proses aplikasi dengan kode keluar (exit code) non-nol demi mencegah state korup.
>
> 3. Closure Memory Leaks: Pastikan referensi callback pada Event Emitter (emitter.on) selalu dibersihkan dengan emitter.off() atau emitter.removeListener() saat komponen dihancurkan guna mencegah kebocoran memori pada heap.

4. Penerapan Praktis & Hands-on Code (Production-Grade TypeScript)

Berikut adalah implementasi teknis tingkat lanjut mencakup pembuatan Concurrency Queue Throttler kustom serta pemodelan domain berbasis Type-Safe Discriminated Unions.

Listing Program: Custom Async Concurrency Limiter Queue

```
// src/utils/concurrency-limiter.ts
/**
 * Concurrency Limiter (Throttler)
 * Membatasi jumlah operasi asinkron yang berjalan serentak guna mencegah
 * kehabisan koneksi soket atau batas kuota rate limit vendor.
 */
export class AsyncConcurrencyQueue {
  private queue: Array<() => Promise<void>> = [];
  private activeCount = 0;

  constructor(private readonly limit: number) {
    if (limit < 1) throw new Error("Concurrency limit harus minimal 1.");
  }

  public async run<T>(taskFn: () => Promise<T>): Promise<T> {
    return new Promise<T>((resolve, reject) => {
      const execute = async () => {
        this.activeCount++;
        try {
```

```
          const result = await taskFn();
          resolve(result);
        } catch (error) {
          reject(error);
        } finally {
          this.activeCount--;
          this.pump();
        }
      };

      if (this.activeCount < this.limit) {
        execute();
      } else {
        this.queue.push(execute);
      }
    });
  }

  private pump(): void {
    if (this.activeCount < this.limit && this.queue.length > 0) {
      const nextTask = this.queue.shift();
      if (nextTask) nextTask();
    }
  }
}
```

Listing Program: Type-Safe Functional Result Pattern dengan TypeScript

```
// src/types/result.type.ts
/**
 * Pola Discriminated Union untuk Pemodelan Hasil Operasi yang Tangguh.
 * Menghilangkan kebutuhan try-catch berlebihan dan memaksa penanganan error eksplisit.
 */
export type AsyncResult<T, E = Error> =
  | { readonly success: true; readonly data: T; readonly timestamp: number }
  | { readonly success: false; readonly error: E; readonly timestamp: number };

export const Result = {
  ok: <T>(data: T): AsyncResult<T, never> => ({
    success: true,
    data,
    timestamp: Date.now()
  }),
  fail: <E>(error: E): AsyncResult<never, E> => ({
    success: false,
    error,
    timestamp: Date.now()
  })
};

// Contoh Penerapan Service Terpadu:
export interface UserDTO {
  id: string;
  name: string;
  email: string;
```

```
}

export async function fetchUserData(userId: string): Promise<AsyncResult<UserDTO, string>>
{
  try {
    if (!userId || userId.trim() === '') {
      return Result.fail("User ID tidak boleh kosong.");
    }

    // Simulasi pemanggilan I/O aman
    const mockUser: UserDTO = { id: userId, name: "Andri Triyono", email:
"andri@nuur.ac.id" };
    return Result.ok(mockUser);
  } catch (err: any) {
    return Result.fail(err.message || "Terjadi kesalahan internal pada database.");
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Platform agregator tiket travel 'CepatTiket' harus mengambil penawaran harga tiket dari 5 maskapai mitra secara serentak. Maskapai tertentu memiliki latensi yang tidak stabil (berkisar antara 200ms hingga 8.000ms) dan server mereka sering mengalami rate limiting jika menerima lebih dari 10 request konkuren dari satu IP server. Ketika satu maskapai mengalami kegagalan koneksi, seluruh laman pencarian tiket di sistem lama menjadi hang dan gagal merespons pengguna.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Rancang modul API Aggregator yang mengeksekusi pemanggilan ke 5 maskapai secara konkuren menggunakan Promise.allSettled() atau Promise Combinator yang sesuai.

2. Integrasikan mekanisme Timeout Circuit Breaker berbatas 2.500ms per vendor, sehingga vendor yang lambat langsung dibatalkan tanpa menunda vendor lainnya.

3. Terapkan AsyncConcurrencyQueue dengan batas maksimal 5 koneksi simultan dan kembalikan struktur data respons menggunakan pola Type-Safe Result.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK043.1 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 2.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK102.1 (Bloom C3–C6).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Pemahaman Event Loop & Asinkron (C4)	Mampu memetakan urutan eksekusi microtask vs macrotask secara presisi dan	Memahami async/await namun belum optimal dalam memitigasi unhandled	Salah memahami sifat non-blocking sehingga memicu

	mencegah thread blocking.	rejection pada microtasks.	thread freeze total.
Penerapan TypeScript Advanced (C3/P4)	Mengimplementasikan Discriminated Unions, Generic Constraints, dan Type Guards tanpa menggunakan tipe 'any'.	Menggunakan TypeScript standar namun masih menyisakan penggunaan tipe 'any' pada penanganan error.	Kode tidak memanfaatkan type safety dan menimbulkan banyak runtime TypeError.
Resiliensi & Concurrency Control (C5)	Membangun throttling queue, timeout circuit breaker, dan fallback caching yang tangguh pada beban tinggi.	Menggunakan Promise standar tanpa pembatasan kuota konkurensi.	Tidak ada mekanisme kontrol konkurensi sehingga server rentan overload.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 2: Poin Kunci Pembelajaran:

1. Node.js memanfaatkan single-threaded Event Loop yang didukung oleh thread pool Libuv untuk mengeksekusi operasi I/O asinkron secara non-blocking.

2. Microtask Queue (Promise callbacks) selalu dieksekusi hingga habis sebelum Event Loop berpindah ke Macrotask Queue (setTimeout/IO).

3. Pengendalian konkurensi asinkron (Throttling & Circuit Breakers) sangat vital untuk menjaga stabilitas sistem saat berinteraksi dengan API eksternal.

4. TypeScript Generics dan Discriminated Unions mengubah pendeteksian bug dari runtime error menjadi compile-time safety.

7.1 Glosarium Istilah Kunci

• Discriminated Union: Pola tipe data TypeScript di mana beberapa tipe objek digabungkan dengan satu properti literal penanda identitas yang unik.

• Event Loop Starvation: Kondisi kritis di mana tumpukan microtask tanpa henti menghalangi eksekusi macrotask dan proses rendering.

• Just-In-Time (JIT) Compilation: Teknik kompilasi kode sumber JavaScript menjadi kode mesin biner saat program sedang berjalan.

• Non-Blocking I/O: Operasi input/output yang mengembalikan kontrol seketika ke thread pemanggil tanpa menunggu data selesai ditransfer.

• Type Narrowing: Proses penyempitan tipe data TypeScript dari tipe luas menjadi tipe yang sangat spesifik melalui pemeriksaan logika (type guards).

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Perhatikan potongan kode di bawah ini. Tentukan urutan kemunculan angka di konsol dan jelaskan pergerakannya dalam Call Stack, Microtask Queue, dan Macrotask Queue!

console.log('1'); setTimeout(() => console.log('2'), 0); Promise.resolve().then(() => console.log('3')).then(() => console.log('4')); console.log('5');

2. [Evaluasi - C5] Evaluasi dampak penggunaan async/await di dalam Array.prototype.map() ketika memproses 500 permintaan database simultan! Mengapa hal ini berisiko menghabiskan Connection Pool basis data?

3. [Kreasi - C6] Rancanglah sebuah fungsi generik retryWithBackoff<T>(fn: () => Promise<T>, maxRetries: number, baseDelayMs: number): Promise<T> yang mengimplementasikan algoritma Exponential Backoff disertai jitter acak untuk mencegah thundering herd problem!

BAB 3
PENERAPAN ALGORITMA PENCARIAN, PENGURUTAN, & MANIPULASI DATA PADA WEB (BIG DATASET OPTIMIZATION)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini menghubungkan fondasi ilmu komputer klasik (kompleksitas algoritma Big-O, struktur data efisien, dan algoritma sorting/searching) dengan rekayasa aplikasi web modern yang dituntut memproses dataset berskala besar (Big Dataset Processing). Mahasiswa dan praktisi akan mempelajari bagaimana pemilihan algoritma yang tepat di lapisan backend dan frontend menentukan keberhasilan sistem dalam menangani jutaan baris data secara responsif.

Tabel 3.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 3.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL02 (Database Administrator), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL10: Mampu menghubungkan pengetahuan sistem komputer dan algoritma dengan implementasi praktis pada aplikasi web.
Capaian Pembelajaran MK (CPMK)	CPMK103: Mampu mengimplementasikan algoritma pencarian, pengurutan, dan manipulasi data berkinerja tinggi pada lingkungan server dan browser.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK103.1: Mampu mengoptimasi kompleksitas algoritma data filtering, pagination, dan streaming manipulation (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mahasiswa dan pembaca mampu menghitung, memetakan, dan memitigasi kompleksitas Big-O (Time & Space Complexity) pada manipulasi array, objek, dan collection data besar.

2. Mampu mengevaluasi kelemahan Offset-Based Pagination dan mengimplementasikan Keyset / Cursor-Based Pagination berkinerja konstan O(1).

3. Mampu merekayasa algoritma pencarian teks cepat (Inverted Index, Trie, dan Fuzzy Search) serta pemrosesan data besar menggunakan Node.js Streams dan Web Workers.

2. Pengantar & Konsep Teoretis: Kompleksitas Komputasi di Lingkungan Web

2.1 Analisis Kompleksitas Big-O pada Aplikasi Web Skala Besar

Dalam skala aplikasi kecil dengan ratusan baris data, perbedaan efisiensi algoritma sering kali tidak terasa, namun pada sistem data terdistribusi pemilihan partisi dan struktur data menjadi sangat krusial [9], [10], [11], [12]. Namun, ketika basis data bertumbuh menjadi jutaan transaksi, perbedaan antara algoritma berderajat kuadratik O(N^2) dan linear-logaritmik O(N log N) menjadi faktor penentu apakah peladen web dapat merespons dalam 15 milidetik atau mengalami CPU spike 100% yang berujung pada downtime fatal.

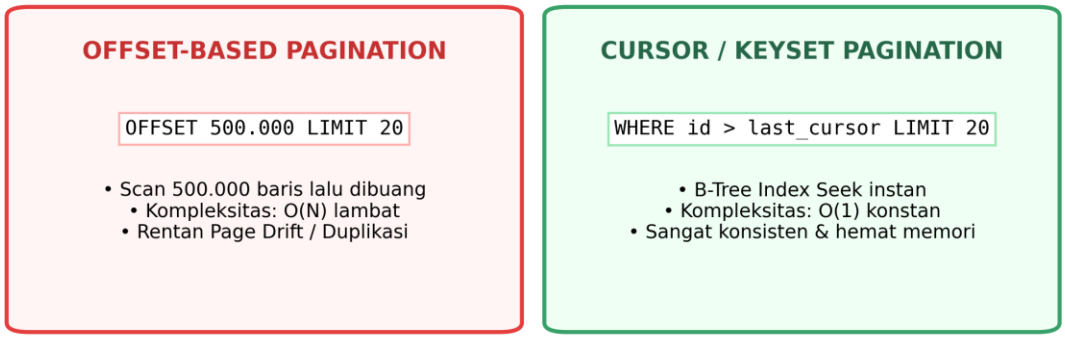
Kesalahan paling umum yang sering dilakukan oleh pengembang web adalah membuat operasi iterasi bersarang tanpa disadari (*nested iteration*), seperti memanggil Array.prototype.find() atau Array.prototype.filter() di dalam Array.prototype.map(). Pola ini mengubah komputasi yang seharusnya linear O(N) menjadi O(N x M), yang sangat membebani Event Loop server.

Analogi Tumpukan Dokumen: Analogi intuitif: Mencari nama pelanggan di dalam tumpukan berkas acak 1.000.000 dokumen satu per satu (Linear Search O(N)) membutuhkan hingga satu juta kali pemeriksaan. Namun, jika dokumen tersebut diindeks menggunakan struktur B-Tree atau Binary Search O(log N), sistem hanya membutuhkan maksimal 20 kali perbandingan untuk menemukan data yang tepat.

2.2 Paradigma Pagination: Offset vs Cursor-Based (Keyset)

Paginasi konvensional berbasis OFFSET dan LIMIT (misal: SELECT * FROM items ORDER BY id LIMIT 20 OFFSET 500000) memaksa mesin basis data membaca, memindai, dan membuang 500.000 baris pertama sebelum mengambil 20 baris yang diinginkan. Kompleksitasnya adalah O(N) di mana waktu eksekusi kueri memburuk secara linear seiring bertambahnya nomor halaman.

Keunggulan Cursor: Keyset / Cursor-Based Pagination menyelesaikan masalah ini dengan menggunakan nilai penanda (*cursor*) dari record terakhir yang dilihat (misal: WHERE id > 500000 ORDER BY id ASC LIMIT 20). Mesin basis data langsung melompat ke posisi indeks menggunakan B-Tree Index Scan dengan kompleksitas konstan O(1), bebas dari beban pemindaian baris terdahulu.



Gambar 3.1: *Perbandingan Mekanisme Kueri: Offset-Based Scan vs Cursor-Based Keyset Index Seek.*

3. Deep Dive Algoritma & Stream Processing (Expert & Production Level)

Pada sistem produksi enterprise, manipulasi dataset besar menuntut kalkulasi cermat antara pemakaian CPU, alokasi memori heap, dan latensi transmisi jaringan.

Tabel 3.2: *Matriks Analisis Kompleksitas Waktu dan Ruang Algoritma Pengurutan dan Pencarian Big Dataset.*

Strategi Pagination	Kompleksitas Waktu	Konsistensi Data (Drift Effect)	Rekomendasi Penggunaan
Offset-Based (LIMIT / OFFSET)	O(N) - Memburuk secara linear pada halaman dalam.	Rentan Data Drift (Data terlewat atau terduplikasi saat ada data baru disisipkan).	Dashboard internal admin dengan dataset kecil (<10.000 baris) yang butuh navigasi acak halaman.
Cursor-Based (Keyset Index)	O(1) - Konstan melalui B-Tree Index Seek.	Sangat Konsisten (Bebas duplikasi data saat ada penambahan record baru).	Feed Media Sosial, Mobile Infinite Scroll, API Publik Skala Produksi Besar.
Time-Bucket Partitioning	O(log N) - Pencarian indeks timestamp partisi.	Sangat Konsisten untuk data berurutan waktu (time-series).	Sistem metrik IoT, Log Audit Transaksi, Financial Ledger Streaming.

3.1 Struktur Data In-Memory: Hash Map vs Trie untuk Pencarian Teks

Ketika mengimplementasikan fitur autocomplete atau pencarian nama produk secara instan, melakukan pencarian substring (String.includes) pada array 100.000 objek memakan waktu ratusan milidetik. Struktur data Trie (Prefix Tree) memungkinkan pencarian awalan kata dengan kompleksitas waktu O(K), di mana K adalah panjang karakter kata yang dicari, sepenuhnya independen dari total jumlah data N.

DEEP DIVE: STRATEGI MANIPULASI DATA BESAR PADA SERVER & BROWSER

1. Hindari JSON.parse() pada File Raksasa: Mem-parsing string JSON berukuran >200MB secara sinkron akan memblokir Call Stack selama beberapa detik. Gunakan library stream parser seperti 'stream-json' atau 'JSONStream' untuk membaca objek secara inkremental.

2. Reduksi Kompleksitas dengan Hash Map Lookup: Ubah pencarian gabungan dua array O(N x M) menjadi O(N + M) dengan membangun Lookup Map (Dictionary) berbasis kunci unik sebelum melakukan perulangan.

3. Web Workers untuk Client-Side Heavy Compute: Jika pengurutan dan filtering 100.000 baris data dilakukan di browser pengguna, jalankan komputasi tersebut di dalam Web Worker terpisah agar antarmuka pengguna (UI thread) tetap mulus pada 60 frame per second (FPS).

4. Penerapan Praktis & Hands-on Code (Production-Grade Engineering)

Berikut implementasi teknis Keyset Pagination berkinerja tinggi serta pemrosesan data streaming yang hemat memori menggunakan Node.js Streams:

Listing Program: High-Performance Cursor-Based Feed Query

```
// src/services/cursor-pagination.service.ts
export interface CursorParams {
  cursor?: string; // Enkoding Base64 dari timestamp dan ID
  limit: number;
}

export interface ProductItem {
  id: string;
  title: string;
  price: number;
  createdAt: string;
}

export class ProductFeedService {
  public async getProductFeed(params: CursorParams, db: any) {
    const pageSize = Math.min(params.limit || 20, 100);
    let query = `
      SELECT id, title, price, created_at AS "createdAt"
      FROM products
    `;
    const queryParams: any[] = [];

    // Jika cursor disertakan, dekode penanda posisi terakhir
    if (params.cursor) {
      const decoded = Buffer.from(params.cursor, 'base64').toString('utf-8');
      const [lastCreatedAt, lastId] = decoded.split('_');
      query += ` WHERE (created_at, id) < ($1, $2)`;
      queryParams.push(lastCreatedAt, lastId);
    }

    // Ambil pageSize + 1 untuk menentukan apakah ada halaman berikutnya
    const paramOffset = queryParams.length;
    query += ` ORDER BY created_at DESC, id DESC LIMIT $${paramOffset + 1}`;
```

```
    queryParams.push(pageSize + 1);

    const rows: ProductItem[] = await db.query(query, queryParams);
    const hasNextPage = rows.length > pageSize;
    const items = hasNextPage ? rows.slice(0, pageSize) : rows;

    let nextCursor: string | null = null;
    if (hasNextPage) {
      const lastItem = items[items.length - 1];
      nextCursor = Buffer.from(`${lastItem.createdAt}_${lastItem.id}`).toString('base64');
    }

    return {
      data: items,
      pageInfo: {
        hasNextPage,
        nextCursor,
        count: items.length
      }
    };
  }
}
```

Listing Program: Memory-Efficient Node.js Stream Transformer

```
// src/pipelines/stream-transformer.ts
import { Transform, TransformCallback } from 'stream';

export interface RawLogRecord {
  id: string;
  ipAddress: string;
  userAgent: string;
  responseTimeMs: string;
}

export class LogSanitizerStream extends Transform {
  constructor() {
    super({ objectMode: true });
  }

  _transform(chunk: RawLogRecord, encoding: BufferEncoding, callback: TransformCallback):
void {
    try {
      // Manipulasi string & anonimisasi IP secara streaming tanpa membebani heap RAM
      const sanitized = {
        id: chunk.id,
        maskedIp: chunk.ipAddress.replace(/\.\d+$/, '.xxx'),
        responseTime: parseFloat(chunk.responseTimeMs) || 0,
        isSlow: parseFloat(chunk.responseTimeMs) > 1000,
        processedAt: new Date().toISOString()
      };

      this.push(sanitized);
      callback();
    } catch (err: any) {
```

```
      callback(err);
    }
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Platform edutech 'PintarNuur' memiliki data riwayat ujian online sebanyak 500.000 siswa. Dashboard pengajar mengalami crash Out-of-Memory (OOM) setiap kali admin mencoba mengekspor seluruh rekap nilai ke dalam format CSV dan melakukan filtering gabungan berdasarkan sekolah dan mata pelajaran. Sistem lama membaca seluruh 500.000 baris ke dalam array memori sekaligus menggunakan fs.readFileSync() dan menjalankan perulangan nested filter O(N^2) untuk menggabungkan data profil siswa.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan audit algoritma dan hitung kompleksitas waktu serta memori dari sistem lama yang menyebabkan kegagalan OOM.

2. Rancang arsitektur pipeline data ekspor menggunakan Node.js Transform Stream dan pipeline() utility sehingga alokasi memori server stabil di bawah 60MB terlepas dari ukuran file.

3. Ganti mekanisme paginasi pada dashboard pengajar dari OFFSET berbasis nomor halaman menjadi Cursor-Based Pagination berbasis komposit indeks (nilai_ujian, siswa_id).

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK103.1 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 3.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK103.1 (Bloom C3–C6).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Analisis Kompleksitas Algoritma (C4)	Mampu menghitung kompleksitas Big-O secara presisi, mereduksi nested loop dari O(N^2) ke O(N) via Lookup Map.	Memahami Big-O dasar namun masih menyisakan operasi loop yang tidak efisien pada manipulasi array.	Tidak memahami implikasi Big-O terhadap beban CPU server.
Implementasi Keyset Pagination (C3/P4)	Menerapkan komposit indeks (timestamp + ID), enkoding cursor aman, dan query beroperasi konstan O(1).	Menerapkan kursor sederhana namun belum menangani kondisi batas nilai timestamp yang bernilai sama (duplikat).	Hanya menggunakan OFFSET/LIMIT standar yang lambat.
Efisiensi Memori & Data	Berhasil memproses dataset	Berhasil memproses data	Terjadi memory leak / fatal

Stream (C5)	>500MB dengan jejak RAM konstan (<60MB) menggunakan Node.js Streams.	namun penggunaan memori RAM sempat melonjak tinggi di awal proses.	Out-of-Memory error saat eksekusi.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 3: Poin Kunci Pembelajaran:

1. Menghindari nested iteration dan memanfaatkan struktur data Hash Map lookup O(1) adalah kunci optimasi algoritma di sisi backend web.

2. Keyset / Cursor-Based Pagination memberikan kecepatan konstan O(1) dan konsistensi data yang mutlak untuk API skala produksi besar.

3. Node.js Streams memungkinkan pemrosesan dataset gigabyte tanpa membebani memori heap server melalui mekanisme backpressure kontrol aliran data.

7.1 Glosarium Istilah Kunci

• Backpressure: Mekanisme kontrol aliran data pada stream ketika produsen data mengirim data lebih cepat daripada kemampuan konsumen untuk memprosesnya.

• Big-O Notation: Notasi matematika formal untuk mengukur batas atas laju pertumbuhan waktu eksekusi atau konsumsi memori suatu algoritma.

• Cursor Pagination: Metode paginasi menggunakan penunjuk unik (pointer/ID) dari data terakhir sebagai acuan pemindaian data berikutnya.

• Inverted Index: Struktur data indeks yang memetakan setiap kata kunci ke daftar dokumen atau entitas yang memuat kata tersebut.

• Trie (Prefix Tree): Struktur data pohon berakar efisien untuk menyimpan dan mencari untaian karakter (string) berdasarkan awalan kata.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Analisis mengapa pencarian relasi antara data 10.000 mahasiswa dan 50.000 data KRS menggunakan Array.prototype.find() di dalam perulangan menghasilkan 500.000.000 operasi perbandingan! Bagaimana cara mengubahnya menjadi O(N + M)?

2. [Evaluasi - C5] Evaluasi komparatif antara penggunaan fs.readFile() dan fs.createReadStream() saat membaca file log akses web server berukuran 4 Gigabyte pada server dengan alokasi RAM 512 Megabyte!

3. [Kreasi - C6] Rancanglah sebuah kelas In-Memory Trie dalam TypeScript yang mampu menyimpan 100.000 nama produk dan menyediakan metode autocompletePrefix(prefix: string, limit: number) yang merespons dalam waktu < 1 milidetik!

BAB 4
PERANCANGAN ARSITEKTUR MODUL SISTEM WEB: MODULAR, CLEAN ARCHITECTURE, & COMPONENT-DRIVEN

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini membahas prinsip-prinsip rekayasa perangkat lunak modern untuk menstrukturkan basis kode sistem web agar memiliki keterpisahan tanggung jawab yang tegas (Separation of Concerns), mudah dirawat dalam jangka panjang (maintainability), mudah diuji secara otomatis (testability), serta terhindar dari berbagai bentuk pembusukan kode (code smells). Pembahasan ini dirancang sistematis agar mudah dipahami pembaca pemula melalui analogi terstruktur, membimbing mahasiswa dalam memenuhi kurikulum OBE, serta menyediakan analisis arsitektural mendalam bagi para praktisi perangkat lunak.

Tabel 4.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 4.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software Quality Assurance Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL10: Mampu menghubungkan konsep sistem komputer dengan perancangan arsitektur modul sistem web yang teruji.
Capaian Pembelajaran MK (CPMK)	CPMK103: Mampu merancang struktur modul web berlapis (Layered Architecture) berbasis prinsip SOLID dan Clean Architecture.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK103.2: Mampu menerapkan pola Controller-Service-Repository dan Dependency Injection pada sistem web (Taksonomi Bloom: C3 - Penerapan & C4 - Analisis).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu menerapkan prinsip desain berorientasi objek SOLID (khususnya Single Responsibility Principle dan Dependency Inversion Principle) dalam membagi modul sistem web.

2. Mampu memisahkan secara tegas lapisan Controller (HTTP Parsing & DTO Validation), Service (Domain Business Logic), dan Repository (Data Access & Persistence).

3. Mampu mengimplementasikan Dependency Injection (DI) Container dan Inversion of Control (IoC) untuk menciptakan sistem nir-keterikatan (*decoupled system*) yang siap diuji secara modular.

2. Pengantar & Konsep Teoretis: Pola Arsitektur Berlapis (Layered Architecture)

2.1 Mengapa Arsitektur Modular Sangat Penting?

Dalam proyek perangkat lunak berskala besar, kode yang tidak terstruktur dengan baik akan mengalami degradasi kualitas yang cepat—fenomena yang dikenal sebagai 'Spaghetti Code'. Ketika seluruh kueri SQL, validasi input pengguna, otentikasi sesi, dan pengiriman email ditulis bertumpuk di dalam satu file routing HTTP yang sama, setiap perubahan kecil pada skema basis data dapat memicu efek domino yang merusak seluruh fungsionalitas aplikasi.

Arsitektur Modular memecah aplikasi menjadi unit-unit logis independen yang memiliki batasan tanggung jawab yang jelas [13], [14], [15], [16]. Setiap modul mengekspos antarmuka kontrak publik (Interface) dan menyembunyikan rincian implementasi internalnya (Encapsulation).

2.2 Prinsip SOLID dalam Arsitektur Web Modern

Prinsip SOLID: Lima prinsip SOLID menjadi pedoman utama dalam membangun modul web yang tangguh:

1. Single Responsibility Principle (SRP): Setiap kelas atau modul hanya boleh memiliki satu alasan untuk berubah. Sebuah controller hanya bertugas menangani HTTP, bukan mengeksekusi rumus diskon atau kueri database.

2. Open/Closed Principle (OCP): Modul harus terbuka untuk perluasan (extension), tetapi tertutup untuk modifikasi langsung. Fitur metode pembayaran baru harus dapat ditambahkan dengan membuat kelas baru tanpa mengubah kode transaksi inti.

3. Liskov Substitution Principle (LSP): Objek turunan harus dapat menggantikan objek induknya tanpa merusak kebenaran program.

4. Interface Segregation Principle (ISP): Klien tidak boleh dipaksa bergantung pada antarmuka yang tidak digunakannya.

5. Dependency Inversion Principle (DIP): Modul tingkat tinggi (logika bisnis) tidak boleh bergantung pada modul tingkat rendah (database/jaringan). Keduanya harus bergantung pada abstraksi interface.

2.3 Anatomi Tiga Lapisan: Controller - Service - Repository

Pembagian Lapisan: Pola Controller-Service-Repository adalah standar de facto dalam rekayasa backend modern:

1. Lapisan Controller: Bertindak sebagai pintu gerbang protokol HTTP. Tugasnya murni menerima HTTP Request, mem-parsing parameter URL dan body payload DTO, memvalidasi tipe data, memanggil fungsi pada Lapisan Service, dan mengembalikan HTTP Response status (200, 201, 400, 404, dll.). Controller TIDAK BOLEH memuat logika bisnis maupun kueri SQL.

2. Lapisan Service (Domain Core): Jantung logika bisnis aplikasi. Lapisan ini menjalankan aturan operasional sistem (seperti menghitung diskon, memverifikasi kuota, atau memicu transaksi). Service

bersifat agnostik terhadap protokol—ia tidak peduli apakah dipanggil dari HTTP REST, GraphQL, WebSocket, ataupun CLI.

3. Lapisan Repository (Data Access): Mengisolasi seluruh interaksi dengan media penyimpanan data (SQL, ORM Prisma/TypeORM, NoSQL, atau Cache Redis). Service memanggil Repository melalui interface kontrak abstrak.

2.4 Mitigasi Code Smells & Arsitektur Anti-Pattern

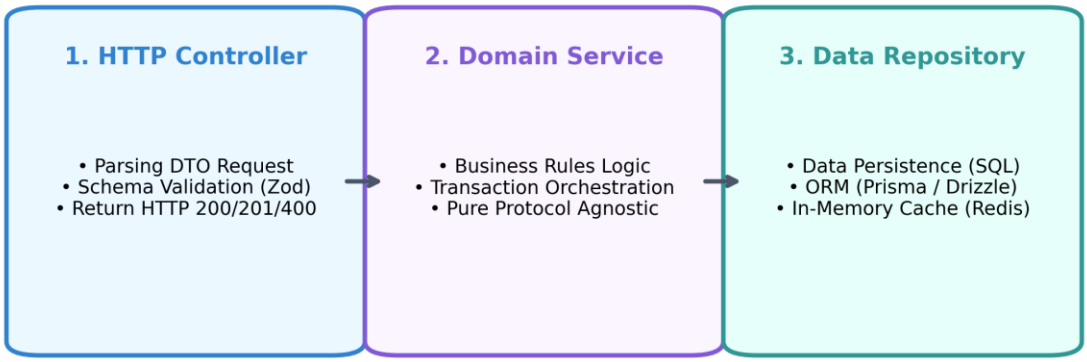
Dalam proyek jangka panjang, pengembang sering kali terjebak dalam berbagai bentuk Code Smells yang menurunkan kualitas sistem:

• God Object / God Class: Sebuah kelas yang memuat ribuan baris kode dan menangani puluhan tanggung jawab yang tidak terkait (misal: kelas UserManager yang memvalidasi password, mengirim email, menghitung pajak, dan mencetak PDF).

• Feature Envy: Sebuah metode pada suatu kelas yang terlalu sering mengakses data atau metode dari kelas lain dibandingkan datanya sendiri, menandakan bahwa metode tersebut berada pada lokasi modul yang salah.

• Circular Dependency: Dua atau lebih modul yang saling mengimpor satu sama lain (A mengimpor B, dan B mengimpor A). Kondisi ini dapat menyebabkan runtime error bernilai undefined saat inisialisasi modul dan membingungkan kompilator TypeScript.

Penerapan Interface Segregation dan Dependency Injection Container secara disiplin mengeliminasi seluruh code smells ini sejak fase perancangan.



Gambar 4.1: *Pola Arsitektur Berlapis Bersih: Controller (HTTP), Service (Domain Core), dan Repository (Data Access).*

3. Deep Dive Clean Architecture & Inversion of Control (Expert Level)

Clean Architecture yang digagas oleh Robert C. Martin (Uncle Bob) dan Arsitektur Heksagonal (Ports and Adapters) oleh Alistair Cockburn menempatkan Aturan Bisnis Inti (Domain Entities & Use Cases) di tengah lingkaran, sementara Framework, Basis Data, dan Antarmuka Pengguna berada di lingkaran terluar sebagai detail teknis yang mudah diganti.

DEEP DIVE: REKAYASA STRUKTUR KODE ENTERPRISE CLEAN ARCHITECTURE

1. Prinsip Dependency Inversion (DIP): Modul tingkat tinggi tidak boleh bergantung pada modul tingkat rendah. Keduanya harus bergantung pada abstraksi (Interface). Dengan demikian, Business Service tidak boleh mengimpor PrismaClient secara langsung, melainkan mengimpor interface IUserRepository.

2. Anti-Pattern 'Fat Controller': Hindari menulis logika kondisional bisnis atau query database di dalam controller Express/Fastify. Batasi controller maksimal 20-40 baris kode per endpoint.

3. Feature-Driven Folder Structure: Daripada membagi folder secara horizontal (/controllers, /services, /models), bagilah secara vertikal berbasis fitur (/modules/auth, /modules/billing, /modules/inventory). Pendekatan ini meningkatkan kohesi dan skalabilitas tim.

4. Penerapan Praktis & Hands-on Code (TypeScript Clean Architecture)

Berikut implementasi pola Clean Architecture berlapis menggunakan TypeScript dengan pemisahan Port Interface, Domain Service, dan Repository:

Listing Program: Lapisan Abstraksi Port (Interface Kontrak)

```
// src/core/domain/ports/user-repository.port.ts
export interface UserEntity {
  id: string;
  email: string;
  fullName: string;
  role: 'STUDENT' | 'LECTURER' | 'ADMIN';
  isActive: boolean;
  createdAt: Date;
}

export interface IUserRepository {
  findById(id: string): Promise<UserEntity | null>;
  findByEmail(email: string): Promise<UserEntity | null>;
  create(user: Omit<UserEntity, 'id' | 'createdAt'>): Promise<UserEntity>;
  updateStatus(id: string, isActive: boolean): Promise<void>;
}
```

Listing Program: Lapisan Domain Service (Murni Bebas Dependensi HTTP/DB)

```
// src/core/domain/services/user-registration.service.ts
import { IUserRepository, UserEntity } from '../ports/user-repository.port';

export interface RegisterUserDTO {
  email: string;
  fullName: string;
  role: 'STUDENT' | 'LECTURER' | 'ADMIN';
}

export class UserRegistrationService {
  // Injeksi dependensi melalui konstruktor (Constructor Injection)
  constructor(
```

```
    private readonly userRepository: IUserRepository,
    private readonly emailNotificationService: { sendWelcome(email: string, name: string):
Promise<void> }
  ) {}

  public async register(dto: RegisterUserDTO): Promise<UserEntity> {
    // 1. Validasi Aturan Domain Bisnis
    const existing = await this.userRepository.findByEmail(dto.email);
    if (existing) {
      throw new Error(`Email ${dto.email} sudah terdaftar dalam sistem akademik.`);
    }

    // 2. Eksekusi Persistensi Data
    const newUser = await this.userRepository.create({
      email: dto.email.toLowerCase().trim(),
      fullName: dto.fullName,
      role: dto.role,
      isActive: true
    });

    // 3. Picu Efek Samping (Notifikasi)
    await this.emailNotificationService.sendWelcome(newUser.email, newUser.fullName);

    return newUser;
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Sebuah sistem akademik kampus 'SIAKAD-Nuur' awalnya dibangun secara terburu-buru dengan pola monolithic script di mana seluruh kueri database MongoDB disisipkan langsung di dalam callback router Express.js. Ketika pimpinan universitas memutuskan untuk mengganti basis data dari MongoDB ke PostgreSQL (karena tuntutan kepatuhan audit transaksional), tim pengembang mengalami frustrasi luar biasa karena harus mengubah lebih dari 150 file router secara manual, yang memakan waktu 4 bulan dan memicu puluhan regresi bug baru.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan dekomposisi terhadap struktur kode SIAKAD-Nuur yang lama menjadi pola Clean Architecture berbasis Controller-Service-Repository.

2. Rancang Interface IStudentRepository dan bangun dua kelas adapter konkret: MongoStudentRepository dan PostgresStudentRepository.

3. Buktikan bahwa penggantian adapter basis data di lapisan luar tidak memerlukan modifikasi sama sekali pada file StudentEnrollmentService (prinsip Open/Closed Principle).

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK103.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 4.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK103.2 (Clean Architecture & SOLID).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Penerapan Separation of Concerns (C4)	Pemisahan Controller, Service, dan Repository 100% konsisten, tidak ada kebocoran query di controller/service.	Sudah memisahkan service namun sesekali controller masih mengakses query ORM secara langsung.	Seluruh logika bisnis dan query database tercampur aduk di dalam file controller.
Dependency Inversion & Injeksi (C3/P4)	Menggunakan interface contract dan constructor injection untuk seluruh dependensi eksternal.	Menggunakan class injection langsung tanpa interface abstraksi.	Melakukan hardcoded 'new PrismaClient()' langsung di dalam domain service.
Struktur Modular & Keterujian (C5)	Struktur folder modular berbasis fitur, kode siap uji unit (TDD-ready) dengan kemudahan mocking.	Struktur folder standar namun masih terjadi dependensi sirkular antar-modul.	File diletakkan dalam satu folder datar (flat structure) tanpa modularitas.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 4: Poin Kunci Pembelajaran:

1. Clean Architecture melindungi logika domain bisnis dari kerapuhan akibat perubahan pustaka pihak ketiga, framework, atau mesin basis data.

2. Pola Controller-Service-Repository membagi kode ke dalam lapisan tanggung jawab tunggal yang sangat terisolasi.

3. Dependency Injection (DI) membalikkan arah ketergantungan (Dependency Inversion) dan memungkinkan pengujian unit otomatis tanpa membutuhkan koneksi basis data riil.

7.1 Glosarium Istilah Kunci

• Dependency Injection (DI): Pola desain perangkat lunak di mana suatu kelas menerima objek yang dibutuhkannya dari luar daripada membuatnya sendiri secara internal.

• DTO (Data Transfer Object): Objek sederhana tanpa metode bisnis yang dirancang murni untuk membawa data antar proses atau antar lapisan sistem.

• Inversion of Control (IoC): Prinsip rekayasa di mana alur kendali eksekusi program dialihkan ke framework atau kontainer eksternal.

• Open/Closed Principle (OCP): Prinsip bahwa modul perangkat lunak harus terbuka untuk perluasan (ekstensi), namun tertutup untuk modifikasi langsung.

• Separation of Concerns (SoC): Prinsip perancangan untuk memisahkan program komputer menjadi bagian-bagian berbeda yang masing-masing menangani perhatian/tanggung jawab tertentu.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Identifikasi 3 risiko arsitektural jangka panjang jika sebuah aplikasi backend mengeksekusi kueri ORM (misal: prisma.user.update) langsung di dalam lapisan Controller Express.js!

2. [Evaluasi - C5] Bandingkan kelebihan dan kekurangan antara arsitektur pembagian folder berbasis Lapisan Horisontal (/controllers, /services, /repositories) versus berbasis Fitur Vertikal (/modules/orders, /modules/users) pada tim yang beranggotakan 25 pengembang perangkat lunak!

3. [Kreasi - C6] Rancanglah sebuah arsitektur antarmuka IStorageService dan buatlah dua kelas implementasi adapter (LocalStorageService dan S3CloudStorageService), lalu demonstrasikan cara menginjeksikannya ke dalam DocumentUploadService menggunakan TypeScript!

BAB 5
REKAYASA ANTARMUKA RESPONSIF & PENGALAMAN PENGGUNA MODERN
(MODERN UI/UX ENGINEERING, DESIGN TOKENS, & ACCESSIBILITY)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini mengupas tuntas rekayasa antarmuka web modern dari perspektif performa rendering peramban, konsistensi Design System (Design Tokens), aksesibilitas web internasional (WCAG 2.2 Level AA), serta prinsip Mobile-First Responsive Layout. Penguasaan bab ini membekali mahasiswa dan praktisi kemampuan membangun antarmuka web yang tidak hanya memukau secara visual, tetapi juga sangat cepat, adaptif di berbagai resolusi layar, dan ramah bagi penyandang disabilitas.

Tabel 5.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 5.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL08: Mampu menguasai konsep teoritis untuk mendesain aplikasi web multi-platform yang responsif, terintegrasi, dan ramah pengguna.
Capaian Pembelajaran MK (CPMK)	CPMK082: Mampu merekayasa antarmuka pengguna responsif dengan prinsip aksesibilitas web dan efisiensi render peramban.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK082.1: Mampu mendesain arsitektur CSS modern (Grid, Flexbox, Container Queries, Custom Properties) dan standar aksesibilitas WCAG (Taksonomi Bloom: C3 - Penerapan & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu menerapkan strategi Mobile-First Responsive Layout menggunakan kombinasi CSS Grid, Flexbox, dan Container Queries (@container).

2. Mampu membangun arsitektur antarmuka berbasis Design Tokens (CSS Custom Properties / Variables) untuk mendukung multi-theming (Dark/Light mode) secara dinamis.

3. Mampu mengaudit dan memperbaiki aksesibilitas web sesuai standar W3C Web Content Accessibility Guidelines (WCAG 2.2) dan mengoptimasi Core Web Vitals (CLS & INP).

2. Pengantar & Konsep Teoretis: Critical Rendering Path & CSS Modern

2.1 Anatomi Jalur Render Kritis (Critical Rendering Path)

Tahapan Rendering: Untuk menampilkan halaman web di layar, peramban (browser) harus melewati serangkaian tahapan komputasi internal:

1. DOM (Document Object Model) Tree: Penguraian struktur tag HTML menjadi simpul-simpul objek pohon data.

2. CSSOM (CSS Object Model) Tree: Penguraian aturan gaya dan selektor CSS menjadi struktur pohon aturan gaya.

3. Render Tree: Penggabungan DOM dan CSSOM yang hanya menyertakan elemen yang benar-benar terlihat di layar.

4. Layout (Reflow): Penghitungan posisi geometri eksak (koordinat x, y, lebar, tinggi) setiap elemen pada viewport peramban.

5. Paint: Proses pengisian piksel warna, bayangan, gambar, dan batas visual pada lapisan grafis.

6. Composite: Penggabungan lapisan-lapisan grafis yang telah di-paint oleh GPU untuk ditampilkan ke monitor pengguna.

Desain CSS yang buruk—seperti memanipulasi properti 'width' atau 'left' secara berulang melalui animasi JavaScript—memaksa browser melakukan Layout Reflow secara terus-menerus, memicu FPS drop (jank) dan menguras baterai perangkat seluler.

2.2 Paradigma Mobile-First & Container Queries (@container)

Prinsip Mobile-First bukanlah sekadar mengecilkan tampilan desktop ke layar ponsel, melainkan memprioritaskan pengalaman pengguna secara adaptif dan terstruktur [17], [18], [19], [20], melainkan memprioritaskan konten esensial dan struktur data paling ringan sejak baseline CSS awal, lalu memperkaya tampilan secara bertahap (*Progressive Enhancement*) untuk layar lebar via media queries.

Container Queries (@container) menandai era baru dalam desain komponen modular. Alih-alih menyesuaikan gaya berdasarkan lebar layar peramban (@media viewport), komponen kini dapat menyesuaikan gayanya sendiri berdasarkan lebar pembungkus induknya (*parent container*). Dengan Container Queries, satu komponen kartu produk yang sama dapat tampil secara vertikal di sidebar dan secara horisontal di area konten utama secara otomatis tanpa class modifier tambahan.

2.3 Metrik Performa Peramban: Core Web Vitals (LCP, INP, & CLS)

Google menetapkan Core Web Vitals sebagai standar resmi penilaian kualitas pengalaman pengguna dan faktor penentu peringkat SEO:

1. Largest Contentful Paint (LCP): Mengukur kecepatan pemuatan konten visual utama di layar (target: < 2.5 detik).

2. Interaction to Next Paint (INP): Mengukur responsivitas antarmuka terhadap interaksi klik atau ketukan keyboard pengguna (target: < 200 milidetik).

3. Cumulative Layout Shift (CLS): Mengukur stabilitas visual dan pergeseran layout tak terduga saat aset dimuat (target: < 0.1).

Untuk mencapai skor LCP dan INP yang optimal, perekayasa web wajib memadatkan gambar ke format WebP/AVIF, mengoptimasi font rendering (font-display: swap), dan menghindari eksekusi JavaScript yang memblokir proses rendering browser.

Gambar 5.1: *Jalur Render Kritis Peramban (Critical Rendering Path): DOM, CSSOM, Render Tree, Layout, Paint, dan Composite.*

3. Deep Dive Design Tokens, Aksesibilitas WCAG, & Core Web Vitals (Expert Level)

Pada skala enterprise, pengelolaan ratusan antarmuka menuntut standarisasi Design Tokens—satuan nilai visual atomik (warna semantik, skala spasi modular, kurva elevasi bayangan) yang diwariskan ke seluruh ekosistem web dan mobile.

> **DEEP DIVE: STANDAR REKAYASA UI/UX TINGKAT PRODUKSI**
>
> 1. Aksesibilitas Warna & Kontras (WCAG 2.2 AA): Teks normal wajib memiliki rasio kontras minimal 4.5:1 terhadap warna latar belakangnya, sedangkan teks berukuran besar (>=18pt atau bold 14pt) wajib memiliki rasio minimal 3:1.
>
> 2. Keyboard Navigation & Focus Visible: Jangan pernah menghapus outline fokus (outline: none) tanpa memberikan indikator fokus visual pengganti (:focus-visible). Seluruh interaksi tombol dan tautan harus dapat diakses via tombol Tab dan Enter.
>
> 3. Mitigasi Cumulative Layout Shift (CLS): Selalu tentukan atribut aspect-ratio atau width/height eksplisit pada elemen gambar (<img>) dan video agar browser dapat memesan ruang layout sebelum berkas media selesai dimuat.

4. Penerapan Praktis & Hands-on Code (CSS Tokens & Container Queries)

Berikut implementasi arsitektur CSS modern yang memadukan Design Tokens, Container Queries, dan aksesibilitas ramah pembaca:

Listing Program: Modern CSS Architecture with Tokens & Container Queries

```
/* src/styles/design-tokens.css */
:root {
  /* Skala Warna Semantik */
  --color-primary-base: #003366;
  --color-primary-hover: #002244;
  --color-surface-canvas: #f8f9fa;
  --color-surface-card: #ffffff;
  --color-text-main: #1a202c;
  --color-text-muted: #4a5568;
  --color-focus-ring: #3182ce;

  /* Skala Spasi Modular (Kelipatan 8px) */
  --space-1: 4px;
  --space-2: 8px;
  --space-3: 16px;
  --space-4: 24px;
  --space-5: 32px;

  /* Radius & Elevasi */
  --radius-md: 8px;
  --shadow-card: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
}

/* Dukungan Mode Gelap Otomatis */
@media (prefers-color-scheme: dark) {
  :root {
    --color-surface-canvas: #0f172a;
    --color-surface-card: #1e293b;
    --color-text-main: #f8fafc;
    --color-text-muted: #94a3b8;
    --color-focus-ring: #60a5fa;
  }
}

/* Komponen Responsif Berbasis Container Query */
.card-wrapper {
  container-type: inline-size;
  container-name: product-card;
}

.responsive-card {
  display: flex;
  flex-direction: column;
  background-color: var(--color-surface-card);
  color: var(--color-text-main);
  border-radius: var(--radius-md);
  padding: var(--space-3);
  box-shadow: var(--shadow-card);
  transition: transform 0.2s ease, box-shadow 0.2s ease;
}
```

```
.responsive-card:focus-visible {
  outline: 3px solid var(--color-focus-ring);
  outline-offset: 2px;
}

/* Jika lebar kontainer induk melebihi 450px, ubah menjadi layout baris */
@container product-card (min-width: 450px) {
  .responsive-card {
    flex-direction: row;
    gap: var(--space-4);
    align-items: center;
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Portal layanan kesehatan universitas 'NuurHealth' menerima keluhan serius dari pasien lanjut usia dan penyandang tunanetra: tombol pendaftaran dokter tidak memiliki kontras warna yang cukup (rasio kontras hanya 2.1:1), formulir tidak dapat dinavigasi menggunakan keyboard tanpa mouse, dan saat dibuka pada tablet terjadi pergeseran layout drastis (Cumulative Layout Shift sebesar 0.52) yang menyebabkan pasien salah menekan tombol batal konsultasi. Skor audit Google Lighthouse untuk kategori Accessibility tercatat hanya 48.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan audit menyeluruh terhadap antarmuka NuurHealth menggunakan alat bantu peramban (Lighthouse & Axe DevTools) dan identifikasi seluruh pelanggaran standar WCAG 2.2.

2. Rancang ulang arsitektur warna dan tipografi menggunakan CSS Design Tokens dengan jaminan rasio kontras >= 4.5:1 untuk teks normal.

3. Terapkan Container Queries pada kartu antrean dokter dan perbaiki elemen media sehingga skor CLS turun di bawah ambang batas 0.05.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK082.1 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 5.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK104.1 (Design Tokens & Critical Rendering Path).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Responsivitas & Layout Modern (C3)	Menerapkan CSS Grid, Flexbox, dan Container Queries secara elegan, layout adaptif mulus di seluruh	Responsif menggunakan media queries standar namun layout masih kaku pada beberapa breakpoint	Tampilan rusak atau memicu horizontal scrollbar pada perangkat seluler.

	resolusi layar.	tablet.	
Standar Aksesibilitas WCAG (C5)	Rasio kontras > 4.5:1, navigasi keyboard tab-index tertata rapi, label ARIA semantik lengkap, skor Lighthouse > 95.	Aksesibel secara visual namun navigasi keyboard belum teratur atau ada tombol tanpa accessible name.	Kontras warna buruk (<3:1), tombol tidak dapat diakses via keyboard.
Performa Render & Tokens (P4)	Memanfaatkan Design Tokens terpusat, bebas dari layout shift (CLS < 0.05), animasi mulus 60 FPS pada GPU.	Menggunakan CSS variables namun struktur token belum terstandarisasi antar modul.	CSS berantakan dengan banyak deklarasi '!important' dan inline styles.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 5: Poin Kunci Pembelajaran:

1. Memahami Critical Rendering Path memungkinkan perekayasa web membuat animasi halus pada 60 FPS tanpa memicu layout reflow berulang.

2. Design Tokens menciptakan satu sumber kebenaran (single source of truth) untuk seluruh variabel gaya visual di aplikasi.

3. Container Queries memungkinkan pembuatan komponen antarmuka mandiri yang benar-benar fleksibel terhadap ruang pembungkusnya.

4. Aksesibilitas web (WCAG) adalah kewajiban etis dan profesional untuk memastikan aplikasi dapat digunakan oleh semua lapisan masyarakat.

7.1 Glosarium Istilah Kunci

• CLS (Cumulative Layout Shift): Metrik Core Web Vitals yang mengukur tingkat pergeseran elemen visual tak terduga saat halaman sedang dimuat.

• Container Query: Aturan CSS (@container) yang memungkinkan elemen menyesuaikan gaya berdasarkan ukuran lebar kontainer pembungkusnya.

• Design Tokens: Variabel nama atomik yang mewakili nilai keputusan desain (seperti warna, font, spasi) dalam format netral platform.

• Layout Reflow: Tahap kompilasi peramban untuk menghitung ulang posisi geometri seluruh elemen di halaman web.

• WCAG (Web Content Accessibility Guidelines): Standar internasional yang diterbitkan oleh W3C untuk memastikan aksesibilitas konten web bagi penyandang disabilitas.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan perbedaan mendasar antara @media query dan @container query! Berikan contoh komponen antarmuka yang tidak dapat dipecahkan secara efisien hanya dengan media query!

2. [Evaluasi - C5] Evaluasi dampak animasi CSS yang memanipulasi properti geometri (margin-top, height) dibandingkan memanipulasi properti transform (translateY, scale) terhadap alur Critical Rendering Path dan beban kerja GPU peramban!

3. [Kreasi - C6] Rancanglah sebuah sistem Design Tokens lengkap dalam format CSS Custom Properties yang mendukung tema terang (Light), tema gelap (Dark), dan tema kontras tinggi (High Contrast) secara otomatis!

BAB 6
PENGEMBANGAN RESTFUL API TINGKAT LANJUT & STANDARISASI KOMUNIKASI DATA (OPENAPI 3.0 & RICHARDSON MODEL)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini membahas standarisasi rekayasa antarmuka pemrograman aplikasi (RESTful API), pemetaan model maturitas Richardson (Level 0 s.d. 3 HATEOAS), penanganan HTTP Status Codes yang presisi, validasi payload DTO yang ketat, standarisasi error format RFC 7807 (Problem Details), idempotensi HTTP, serta dokumentasi interaktif otomatis berbasis OpenAPI 3.0 / Swagger.

Tabel 6.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 6.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL08: Mampu mendesain aplikasi web terintegrasi basis data dan API yang andal dan aman.
Capaian Pembelajaran MK (CPMK)	CPMK082: Mampu merancang dan mengimplementasikan RESTful API tingkat lanjut yang memenuhi standar industri perangkat lunak.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK082.2: Mampu merancang arsitektur API berskala enterprise dengan spesifikasi OpenAPI dan validasi middleware (Taksonomi Bloom: C3 - Penerapan & C4 - Analisis).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu mengidentifikasi, mengevaluasi, dan menerapkan tingkat kematangan API berdasarkan Richardson Maturity Model (Level 0 hingga Level 3).

2. Mampu merancang kontrak API yang konsisten, menerapkan sifat Idempotensi pada HTTP Methods (PUT, DELETE, dan Idempotency-Key pada POST), serta menstandarisasi respons kesalahan menggunakan format RFC 7807.

3. Mampu membangun skema validasi data transfer object (DTO) berbasis Zod / TypeBox dan mengintegrasikan dokumentasi interaktif OpenAPI 3.0 secara otomatis.

2. Pengantar & Konsep Teoretis: Richardson Maturity Model

2.1 Empat Tingkat Kematangan Model Richardson

Empat Tingkat Maturitas: Dalam praktik industri, banyak API sering diklaim sebagai 'RESTful' padahal kenyataannya hanya berupa RPC yang menuntut standarisasi dekomposisi antarmuka [21], [22], [23], [24] (Remote Procedure Call) yang menunggangi protokol HTTP secara sembarangan. Leonard Richardson merumuskan model kematangan (Richardson Maturity Model) untuk mengukur tingkat kepatuhan arsitektur REST:

• Level 0 (The Swamp of POX): Menggunakan HTTP murni sebagai saluran transportasi tanpa memanfaatkan semantik web sama sekali. Semua request dikirimkan ke satu endpoint tunggal (misal: /api/endpoint) menggunakan metode POST, dengan aksi operasi diletakkan di dalam body payload XML/JSON.

• Level 1 (Resources): Mengenalkan konsep Resource URI unik untuk setiap entitas data (misal: /api/v1/products/101 vs /api/v1/products/102). Namun seluruh operasi masih dominan menggunakan metode POST tunggal.

• Level 2 (HTTP Verbs & Semantics): Memanfaatkan kata kerja HTTP secara semantik dan tepat: GET untuk membaca data (bersifat aman/safe dan idempoten), POST untuk membuat entitas baru, PUT untuk penggantian utuh, PATCH untuk pembaruan parsial, dan DELETE untuk menghapus data. Dilengkapi penggunaan HTTP Status Codes yang presisi (200 OK, 201 Created, 204 No Content, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 409 Conflict, 422 Unprocessable Entity, 500 Internal Server Error).

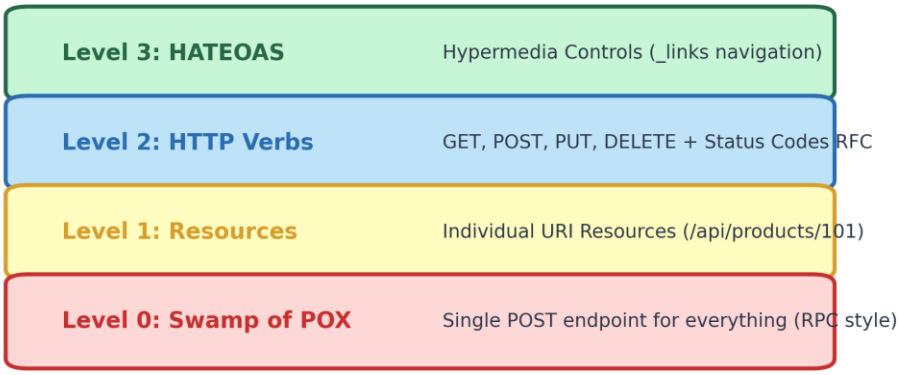
• Level 3 (Hypermedia Controls / HATEOAS): Tingkat tertinggi di mana representasi respons JSON memuat tautan hypermedia (_links) yang secara dinamis memandu klien tentang aksi legal berikutnya yang dapat dieksekusi (misal: tautan untuk membatalkan pesanan, membayar pesanan, atau melacak kurir).

2.2 Standarisasi Caching HTTP & Header Validasi Bersyarat

RESTful API tingkat produksi wajib memanfaatkan mekanisme caching HTTP bawaan guna menghemat bandwidth server dan mempercepat waktu respons:

• ETag (Entity Tag): Nilai hash kriptografis unik yang mewakili representasi resource saat ini. Klien mengirimkan header 'If-None-Match: <hash>' pada request berikutnya; jika data belum berubah, server mengembalikan HTTP 304 Not Modified tanpa body payload, menghemat bandwidth hingga 95%.

• Cache-Control Headers: Mendefinisikan aturan caching (misal: 'public, max-age=3600, stale-while-revalidate=60') yang menginstruksikan CDN dan peramban untuk menyimpan cache secara aman.



Gambar 6.1: *Tingkat Kematangan Arsitektur RESTful API Berdasarkan Richardson Maturity Model (Level 0 s.d. Level 3).*

3. Deep Dive Idempotensi, Versioning, & Error RFC 7807 (Expert Level)

Idempotensi adalah sifat matematika dan komputasi di mana suatu operasi yang dieksekusi satu kali atau berulang kali menghasilkan efek samping (side-effect) yang persis sama pada status sistem backend.

Tabel 6.2: *Matriks Perbandingan Karakteristik Metode HTTP (Idempotency, Safety, dan Cacheability).*

HTTP Method	Sifat Safe (Nir-Modifikasi)	Sifat Idempoten	Semantik Operasional
GET	Ya (Hanya membaca data)	Ya (Eksekusi berulang tidak mengubah data)	Mengambil representasi resource.
POST	Tidak (Memodifikasi server state)	Tidak (Eksekusi berulang membuat entitas baru)	Membuat resource baru atau memproses aksi kompleks.
PUT	Tidak (Memodifikasi server state)	Ya (Menimpa representasi yang sama berulang kali)	Menggantikan seluruh entitas data yang ada.
PATCH	Tidak (Memodifikasi server state)	Dapat Idempoten / Non-idempoten	Memperbarui sebagian field atribut entitas.
DELETE	Tidak (Memodifikasi server state)	Ya (Penghapusan berulang menghasilkan status terhapus)	Menghapus resource dari sistem.

> **DEEP DIVE: STANDARISASI KONTRAK REST TINGKAT PRODUKSI**
>
> 1. RFC 7807 Problem Details: Jangan mengembalikan format error kustom tak terstruktur seperti { error: 'data gagal disimpan' }. Gunakan format standar RFC 7807 dengan atribut resmi: type, title, status, detail, instance, dan invalidParams.

2. Header Idempotency-Key: Untuk endpoint POST finansial (checkout pembayaran), wajibkan klien mengirimkan header 'Idempotency-Key: <UUID>'. Backend menyimpan status kunci ini di Redis untuk memblokir pemrosesan ganda jika request terkirim ulang akibat timeout jaringan.

3. API Versioning: Terapkan strategi URI Versioning (/api/v1/...) untuk stabilitas integrasi jangka panjang. Hindari breaking changes pada versi minor yang sedang aktif.

4. Penerapan Praktis & Hands-on Code (Zod Validation & OpenAPI)

Berikut implementasi controller RESTful API profesional dengan validasi skema DTO yang ketat serta format respons HATEOAS:

Listing Program: Enterprise REST API Controller with Zod & RFC 7807

```
// src/controllers/product.controller.ts
import { Request, Response } from 'express';
import { z } from 'zod';

// 1. Skema Validasi Input Payload DTO (Zod)
export const CreateProductDTOSchema = z.object({
  title: z.string().min(3, "Judul minimal 3 karakter").max(100),
  price: z.number().positive("Harga wajib bernilai positif"),
  sku: z.string().regex(/^[A-Z]{3}-[0-9]{4}$/, "Format SKU harus XXX-0000"),
  stock: z.number().int().nonnegative("Stok tidak boleh negatif")
});

export const handleCreateProduct = async (req: Request, res: Response): Promise<Response>
=> {
  // 2. Eksekusi Validasi Skema
  const validationResult = CreateProductDTOSchema.safeParse(req.body);

  if (!validationResult.success) {
    // Kembalikan format standar RFC 7807 Problem Details
    return res.status(422).json({
      type: "https://api.nuur.ac.id/errors/validation-failed",
      title: "Unprocessable Entity",
      status: 422,
      detail: "Payload yang dikirimkan tidak memenuhi spesifikasi kontrak API.",
      instance: req.originalUrl,
      invalidParams: validationResult.error.issues.map(issue => ({
        field: issue.path.join('.'),
        message: issue.message
      }))
    });
  }

  const validPayload = validationResult.data;
  const productId = crypto.randomUUID();

  // 3. Kembalikan Respons Level 3 Richardson (HATEOAS Hypermedia)
  return res.status(201).json({
    data: {
      id: productId,
```

```
      ...validPayload,
      createdAt: new Date().toISOString()
    },
    _links: {
      self: { href: `/api/v1/products/${productId}`, method: "GET" },
      update: { href: `/api/v1/products/${productId}`, method: "PUT" },
      delete: { href: `/api/v1/products/${productId}`, method: "DELETE" },
      collection: { href: `/api/v1/products`, method: "GET" }
    }
  });
};
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Aplikasi dompet digital kampus 'NuurPay' mengalami insiden gawat: ketika mahasiswa melakukan pembayaran makanan di kantin pada kondisi sinyal seluler buruk, penekanan tombol 'Kirim' yang berulang-ulang mengakibatkan saldo mahasiswa terpotong 3 kali lipat untuk satu transaksi pesanan yang sama. Selain itu, jika terjadi kesalahan validasi nomor rekening, backend mengembalikan HTTP 200 OK dengan format body { status: 'gagal' } yang membingungkan tim frontend mobile.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan refactoring terhadap endpoint pembayaran NuurPay dengan menerapkan mekanisme Idempotency-Key berbasis Redis middleware.

2. Perbaiki seluruh HTTP Status Codes sesuai kaidah Richardson Maturity Model Level 2 dan standarisasikan seluruh respons galat menjadi format RFC 7807.

3. Susun spesifikasi kontrak OpenAPI 3.0 (YAML/JSON) yang memetakan seluruh skema request, response, dan error boundary secara interaktif.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK082.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 6.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK105.1 (RESTful API & OpenAPI 3.0).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Desain & Kepatuhan REST (C4)	Menerapkan semantik HTTP verbs, status codes presisi, HATEOAS links, dan idempotensi transaksi aman.	Menggunakan HTTP verbs standar namun status codes dan struktur error masih inkonsisten di beberapa endpoint.	Menggunakan pola RPC semata (seluruh request via POST dengan status 200).
Validasi Skema & Kontrak (C3/P4)	Validasi DTO ketat menggunakan Zod/TypeBox	Ada validasi manual sederhana namun pesan	Tidak ada validasi skema sehingga server rentan

	dengan format error RFC 7807 terperinci.	kesalahan masih ambigu.	menerima bad payload.
Dokumentasi OpenAPI 3.0 (C5)	Dokumentasi Swagger/OpenAPI interaktif lengkap dengan model skema, contoh payload, dan seluruh respon 2xx/4xx/5xx.	Dokumentasi OpenAPI ada namun belum menyertakan rincian model error 4xx/5xx.	Tidak ada dokumentasi kontrak API yang terstandarisasi.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 6: Poin Kunci Pembelajaran:

1. Richardson Maturity Model memberikan tolok ukur yang jelas untuk meningkatkan kematangan arsitektur RESTful API.

2. Idempotensi adalah keharusan mutlak pada operasi sistem yang memodifikasi status data agar kebal terhadap duplikasi request jaringan.

3. Format standar RFC 7807 dan OpenAPI 3.0 menjamin kolaborasi yang mulus antara pengembang backend, frontend, dan mitra pihak ketiga.

7.1 Glosarium Istilah Kunci

• HATEOAS (Hypermedia as the Engine of Application State): Prinsip REST di mana respon server memuat tautan kendali aksi berikutnya.

• Idempotency-Key: Header HTTP unik yang digunakan backend untuk mendeteksi dan mencegah pemrosesan ulang transaksi yang sama.

• OpenAPI Specification (OAS): Standar deskripsi formal antarmuka RESTful API yang dapat dibaca manusia maupun mesin.

• RFC 7807 (Problem Details): Standar format JSON terstruktur untuk mendeskripsikan rincian galat pada HTTP API.

• Zod: Pustaka deklarasi skema dan validasi tipe data berbasis TypeScript dengan inferensi tipe otomatis.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan perbedaan mendasar antara HTTP PUT dan HTTP PATCH! Mengapa operasi PUT wajib bersifat idempoten sedangkan PATCH tidak selalu idempoten?

2. [Evaluasi - C5] Evaluasi dampak buruk terhadap monitoring sistem dan pipeline error tracking (seperti Sentry/Datadog) jika sebuah backend selalu mengembalikan HTTP 200 OK untuk seluruh respon gagal!

3. [Kreasi - C6] Rancanglah sebuah middleware Express.js yang memeriksa header 'Idempotency-Key' pada Redis dan mengembalikan respons cache yang identik jika kunci yang sama dikirimkan ulang dalam kurun waktu 24 jam!

BAB 7
PENGELOLAAN BASIS DATA LANJUT, PEMODELAN RELASI, & OPTIMASI QUERY (ORM, INDEXING, & CONCURRENCY LOCKS)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini membedah teknik rekayasa data tingkat lanjut pada aplikasi web enterprise, meliputi pemodelan skema relasional yang teroptimasi, navigasi hubungan antar-tabel, mitigasi perangkap N+1 Query Problem pada ORM modern (Prisma/TypeORM/Drizzle), perancangan B-Tree Composite Indexing berdasarkan Leftmost Prefix Rule, strategi isolasi transaksi (ACID), serta pengendalian konkurensi data menggunakan Pessimistic vs Optimistic Locking.

Tabel 7.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 7.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL02 (Database Administrator).
Capaian Pembelajaran Lulusan (CPL)	CPL08: Mampu menguasai konsep teoritis basis data dan integrasinya dalam aplikasi web modern yang aman dan berkinerja tinggi.
Capaian Pembelajaran MK (CPMK)	CPMK082: Mampu mengoptimasi performa kueri basis data dan mengamankan integritas data dari fenomena race condition pada akses konkuren.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK082.2: Mampu merancang skema migrasi database, indexing komposit, dan locking konkurensi (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu mengidentifikasi, menganalisis rencana eksekusi kueri (EXPLAIN ANALYZE), dan mengeliminasi N+1 Query Problem pada pemanggilan ORM modern.

2. Mampu merancang indeks B-Tree tunggal dan komposit yang optimal berdasarkan Leftmost Prefix Rule guna mencegah terjadinya Full Table Scan.

3. Mampu mengimplementasikan Optimistic Locking (kolom versi) dan Pessimistic Locking (SELECT ... FOR UPDATE) untuk mencegah race condition dan Lost Update pada transaksi data bernilai tinggi.

2. Pengantar & Konsep Teoretis: Anatomi Kinerja Basis Data di Lapisan Web

2.1 Bahaya Tersembunyi N+1 Query Problem pada ORM

Object-Relational Mapping (ORM) memberikan kemudahan abstraksi objek yang luar biasa bagi pengembang web. Namun, jika mekanisme kerjanya tidak dipahami secara mendalam, ORM dapat memicu bencana performa yang melumpuhkan peladen basis data. Masalah N+1 Kueri terjadi ketika aplikasi ingin menampilkan N entitas induk (misalnya 100 artikel blog), lalu di dalam perulangan kode aplikasi mengeksekusi kueri individual tambahan untuk mengambil data relasi (misalnya 100 kueri terpisah untuk mengambil nama penulis).

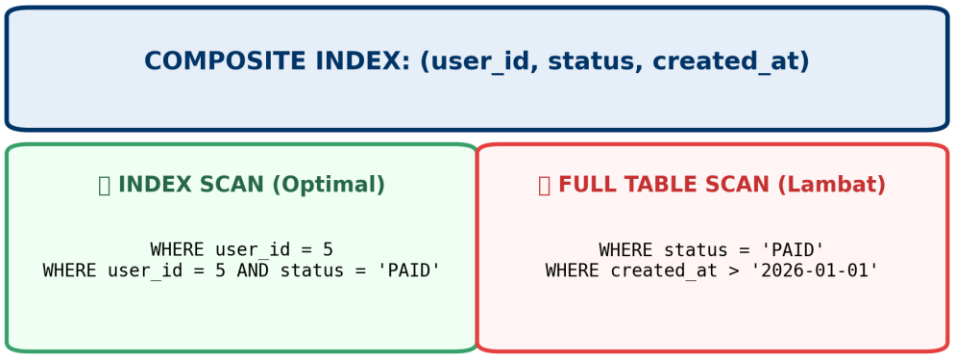
Dampak N+1: Total kueri yang dikirimkan ke database melonjak drastis menjadi 1 + 100 = 101 kueri jaringan. Pada saat lalu lintas pengguna padat, overhead round-trip time (RTT) koneksi basis data ini akan menghabiskan Connection Pool dan membuat utilisasi CPU server melonjak 100%.

Solusi Eager Loading: Solusi optimasi: Menggunakan teknik Eager Loading (SQL JOIN eksplisit) atau Batch Loading (IN-query aggregation) sehingga seluruh 100 artikel dan penulisnya diambil hanya dalam 1 atau 2 kueri SQL terpadu.

2.2 B-Tree Indexing & Aturan Awalan Terkiri (Leftmost Prefix Rule)

Indeks basis data bekerja persis seperti indeks alfabetis di halaman belakang buku ensiklopedia. Tanpa indeks, mesin basis data harus membaca setiap baris data dari baris pertama hingga terakhir di dalam hard drive—operasi yang disebut Sequential Scan / Full Table Scan O(N).

Leftmost Prefix Rule: Dengan B-Tree Index, data disusun dalam struktur pohon seimbang yang memungkinkan pencarian dilakukan dengan kompleksitas logaritmik O(log N). Pada Indeks Komposit yang mencakup beberapa kolom (misal: indeks pada [user_id, status, created_at]), kueri SQL hanya dapat memanfaatkan indeks jika klausa WHERE melibatkan kolom paling kiri (user_id). Jika kueri hanya memfilter berdasarkan kolom status dan created_at, mesin database tidak dapat menggunakan indeks komposit tersebut secara optimal.



Gambar 7.1: *Kaidah Awalan Terkiri (Leftmost Prefix Rule) pada B-Tree Composite Indexing Basis Data Relasional.*

3. Deep Dive Concurrency Control & Database Locking (Expert Level)

Ketika dua transaksi web mencoba memodifikasi baris data yang sama pada detik yang persis bersamaan, kalkulasi overhead latensi dan memori harus diperhitungkan secara presisi [25], [26], [27], [28], sistem rentan mengalami anomali konkurensi (seperti Lost Update atau Dirty Read).

Tabel 7.2: *Matriks Analisis Komparatif Performa Pendekatan Akses Data: Raw SQL vs Query Builder vs ORM.*

Mekanisme Locking	Prinsip Operasional	Overhead & Skalabilitas	Kasus Penggunaan Optimal
Pessimistic Locking (SELECT ... FOR UPDATE)	Mengunci baris data di tingkat mesin database saat dibaca; transaksi lain wajib antre hingga commit.	Tinggi (Dapat memicu lock contention atau deadlock jika transaksi lama).	Sistem perbankan inti, pemotongan stok tiket konser menit terakhir, alokasi saldo dompet digital.
Optimistic Locking (Version Column)	Tidak mengunci baris saat dibaca; saat update diverifikasi bahwa nomor versi data belum berubah.	Sangat Rendah (Bebas dari lock overhead, throughput transaksi sangat tinggi).	Aplikasi dokumen kolaboratif, pengeditan profil pengguna, sistem CMS artikel berita.

DEEP DIVE: STRATEGI REKAYASA BASIS DATA SKALA TINGGI

1. EXPLAIN ANALYZE: Jangan pernah berasumsi tentang kecepatan kueri; selalu jalankan perintah 'EXPLAIN (ANALYZE, BUFFERS)' di PostgreSQL untuk memastikan apakah eksekusi menggunakan Index Scan atau Seq Scan.

2. Connection Pooling (PgBouncer / HikariCP): Aplikasi web modern tidak boleh membuka koneksi TCP baru setiap kali ada request HTTP. Gunakan connection pool dengan batas maksimal terukur untuk melindungi database dari kehabisan memori RAM.

3. Deadlock Prevention: Selalu akses tabel dan kunci baris dengan urutan yang konsisten (misal: selalu kunci

> Akun A lalu Akun B) pada seluruh transaksi di dalam aplikasi untuk mencegah terjadinya siklus saling tunggu (deadlock).

4. Penerapan Praktis & Hands-on Code (Pessimistic & Optimistic Locking)

Berikut implementasi teknis pengendalian konkurensi menggunakan Prisma ORM dan kueri SQL transaksional:

Listing Program: Implementation of Pessimistic & Optimistic Concurrency Control

```
// src/services/inventory-reservation.service.ts
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

// 1. Penerapan Pessimistic Locking (SELECT ... FOR UPDATE)
export async function reserveStockPessimistic(productId: string, quantity: number) {
  return await prisma.$transaction(async (tx) => {
    // Kunci baris produk secara eksklusif dari transaksi lain
    const [product]: any = await tx.$queryRaw`
      SELECT id, stock, title
      FROM "products"
      WHERE id = ${productId}
      FOR UPDATE
    `;

    if (!product) {
      throw new Error("Produk tidak ditemukan dalam katalog.");
    }

    if (product.stock < quantity) {
      throw new Error(`Stok tidak mencukupi. Tersisa: ${product.stock}, diminta:
${quantity}`);
    }

    // Eksekusi pembaruan stok yang aman dari race condition
    await tx.$executeRaw`
      UPDATE "products"
      SET stock = stock - ${quantity}, updated_at = NOW()
      WHERE id = ${productId}
    `;

    return { success: true, remainingStock: product.stock - quantity };
  });
}

// 2. Penerapan Optimistic Locking (Version Column Pattern)
export async function updateArticleOptimistic(articleId: string, newTitle: string,
expectedVersion: number) {
  const result = await prisma.article.updateMany({
    where: {
      id: articleId,
      version: expectedVersion // Hanya update jika versi data masih sama dengan saat
```

```
dibaca
    },
    data: {
      title: newTitle,
      version: { increment: 1 } // Naikkan nomor versi secara atomik
    }
  });

  if (result.count === 0) {
    throw new Error("Konflik Konkurensi: Data telah diubah oleh pengguna lain. Silakan muat
ulang halaman.");
  }

  return { success: true, message: "Artikel berhasil diperbarui." };
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Sistem seleksi pendaftaran beasiswa universitas 'NuurScholarship' menerima 25.000 pendaftar dalam 2 jam terakhir menjelang batas waktu penutupan. Laman dashboard verifikator yang menampilkan daftar pendaftar beserta dokumen lampiran dan asal program studi mengalami loading lambat hingga 24 detik dan menyebabkan server basis data MySQL mengalami overload (CPU 100%). Setelah diaudit, query ORM menghasilkan 50.001 kueri SQL terpisah (N+1 Problem) dan kueri penyaringan status pendaftar melakukan Full Table Scan karena composite index dibuat dengan urutan kolom yang tidak mematuhi Leftmost Prefix Rule.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan profiling query menggunakan EXPLAIN ANALYZE untuk mengidentifikasi bottleneck dan scan cost pada sistem lama.

2. Rancang ulang kueri ORM menggunakan Eager Loading JOIN dan buktikan bahwa seluruh data pendaftar dapat diambil hanya dalam 1 kueri SQL efisien.

3. Susun B-Tree Composite Indexing yang presisi dan implementasikan Pessimistic Locking pada proses persetujuan kuota beasiswa agar tidak terjadi over-quota.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK082.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 7.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK106.1 (Database Concurrency & Indexing).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Optimasi Kueri & Mitigasi N+1 (C4)	Mampu mengeliminasi N+1 problem secara menyeluruh	Memperbaiki kueri utama namun masih menyisakan	Kueri dibiarkan melakukan looping di dalam kode

	via Eager Loading / DataLoader dan membuktikannya via log kueri.	N+1 loop pada relasi bertingkat sekunder.	aplikasi.
Desain Indexing & Profiling (C5)	Mampu membaca rencana kueri EXPLAIN ANALYZE dan merancang composite index sesuai Leftmost Prefix Rule.	Menambahkan indeks tunggal pada semua kolom secara berlebihan tanpa analisis pola kueri.	Tidak memahami konsep index seek vs scan.
Kontrol Konkurensi & Locking (C3/P4)	Menerapkan Optimistic / Pessimistic Locking secara tepat pada skenario data bernilai tinggi bebas race condition.	Menggunakan transaksi dasar tanpa mekanisme proteksi race condition pada kueri update.	Pembaruan data rentan mengalami anomali Lost Update.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 7: Poin Kunci Pembelajaran:

1. Pemahaman mendalam tentang eksekusi SQL di balik abstraksi ORM sangat krusial untuk mencegah degradasi performa N+1 kueri.

2. Composite index yang efektif harus dirancang mengikuti pola kueri nyata dengan memperhatikan kaidah Leftmost Prefix Rule.

3. Pessimistic Locking menjamin konsistensi mutlak pada data terbatas (stok/saldo), sementara Optimistic Locking memberikan skalabilitas tinggi untuk data kolaboratif.

7.1 Glosarium Istilah Kunci

• Connection Pool: Kumpulan koneksi database yang siap digunakan dan dikelola bersama guna menghindari overhead pembukaan koneksi TCP baru.

• Eager Loading: Pola pengambilan data di mana entitas utama dan seluruh entitas relasinya diambil secara bersamaan dalam satu kueri SQL.

• EXPLAIN ANALYZE: Perintah mesin database untuk menampilkan rencana eksekusi kueri, estimasi biaya, dan durasi waktu eksekusi nyata.

• Lost Update: Anomali konkurensi di mana dua transaksi menimpa data yang sama secara bersamaan sehingga modifikasi transaksi pertama hilang.

• N+1 Query Problem: Masalah inefisiensi kueri di mana pengambilan N data anak memicu N kueri individual tambahan ke database.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Diberikan sebuah tabel 'orders' dengan composite index (user_id, status, created_at). Manakah dari 3 kueri WHERE berikut yang dapat memanfaatkan indeks secara penuh? Jelaskan alasannya berdasarkan Leftmost Prefix Rule!

   A. WHERE user_id = 5 AND status = 'PAID'

   B. WHERE status = 'PAID' AND created_at > '2026-01-01'

   C. WHERE user_id = 5 AND created_at > '2026-01-01'

2. [Evaluasi - C5] Evaluasi dampak negatif penambahan indeks B-Tree pada seluruh kolom tabel yang memiliki frekuensi operasi INSERT dan UPDATE sebesar 10.000 transaksi per detik!

3. [Kreasi - C6] Rancang skema database PostgreSQL dan kode transaksi yang aman untuk sistem 'Flash Sale' di mana 5.000 pengguna berebut membeli 20 barang sisa tanpa terjadi anomali overselling!

BAB 8
KOMUNIKASI LAYANAN LANJUT: GRAPHQL, WEBSOCKETS, & EVENT-DRIVEN ARCHITECTURE (PUB/SUB & MESSAGE BROKERS)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini mengupas evolusi protokol komunikasi data modern di luar batas arsitektur REST konvensional: fleksibilitas query GraphQL (mitigasi over-fetching dan under-fetching), komunikasi real-time dua arah persisten (full-duplex) berbasis WebSockets, serta arsitektur berbasis peristiwa (Event-Driven Architecture / EDA) menggunakan antrean pesan terdistribusi Kafka dan RabbitMQ.

Tabel 8.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 8.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi kebutuhan computing yang efisien dan arsitektur komunikasi layanan perangkat lunak.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu merancang sistem komunikasi layanan real-time dan asinkron berbasis protokol modern.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.2: Mampu merancang skema GraphQL, WebSocket duplex handler, dan message broker pub/sub (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu menganalisis inefisiensi Over-fetching & Under-fetching pada REST dan menyelesaikannya menggunakan skema GraphQL deklaratif beserta DataLoader.

2. Mampu membangun koneksi dua arah berlatensi rendah (WebSockets) dengan manajemen Heartbeat Ping-Pong dan penanganan reconnection yang tangguh.

3. Mampu merancang arsitektur Event-Driven menggunakan pola Publish/Subscribe (Pub/Sub) pada message broker untuk memutus ketergantungan (decoupling) antar-layanan.

2. Pengantar & Konsep Teoretis: Evolusi Protokol Komunikasi Data

2.1 Over-fetching vs Under-fetching: Solusi Deklaratif GraphQL

Keterbatasan REST: Pada arsitektur REST tradisional, struktur respons JSON ditentukan sepenuhnya oleh server. Hal ini sering menimbulkan dua inefisiensi transmisi data:

• Over-fetching: Endpoint /api/users/1 mengembalikan 50 field atribut lengkap (termasuk alamat, riwayat login, nomor telepon) padahal aplikasi seluler hanya butuh menampilkan nama dan foto profil, memboroskan kuota data seluler pengguna.

• Under-fetching: Klien harus melakukan beberapa HTTP request terpisah secara berurutan (/api/users/1, lalu /api/users/1/posts, lalu /api/posts/10/comments) hanya untuk merender satu halaman beranda.

Keunggulan GraphQL: GraphQL diciptakan oleh Meta untuk menyelesaikan masalah ini. Klien mengirimkan kueri deklaratif yang menentukan secara presisi struktur data yang diinginkan, dan server mengembalikan bentuk data yang persis sama dalam satu putaran request.

2.2 WebSockets vs HTTP Polling Tradisional

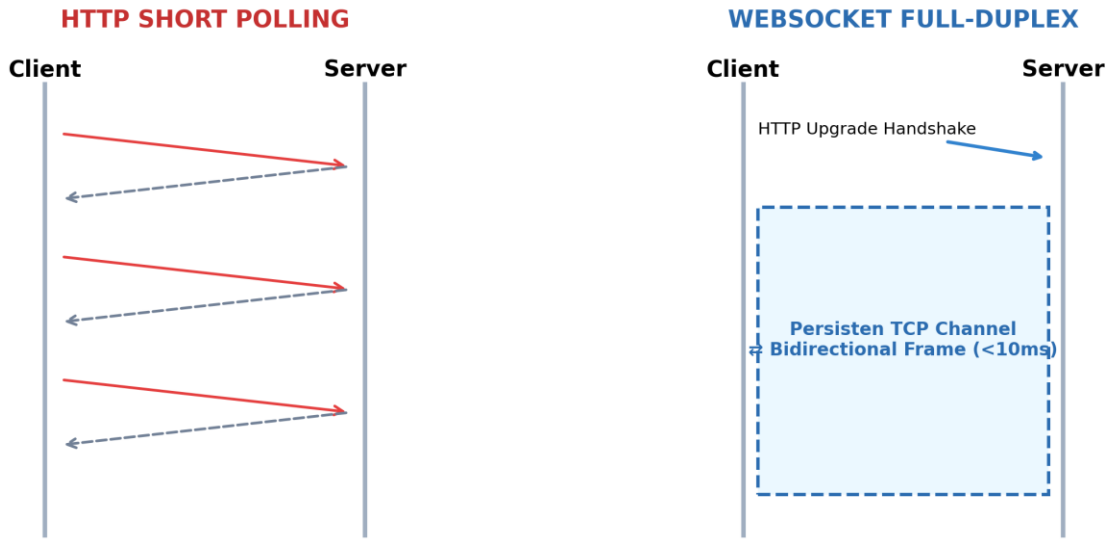
Pada aplikasi real-time klasik, klien menggunakan teknik Short Polling (mengirimkan HTTP request setiap 2 detik untuk mengecek data baru). Praktik ini sangat boros karena menciptakan overhead handshake TCP/TLS dan HTTP header yang berulang-ulang.

Mekanisme Duplex: WebSockets melakukan satu kali proses HTTP Upgrade Handshake di awal, lalu membuka saluran TCP full-duplex dua arah yang persisten. Baik server maupun klien dapat saling mengirimkan frame data kapan saja dengan overhead header yang sangat kecil (2 hingga 10 byte), memungkinkan latensi transmisi sub-10 milidetik.

2.3 Pola Transaksional Event-Driven: Outbox Pattern & Idempotent Consumer

Dalam arsitektur berbasis event terdistribusi, menjaga konsistensi antara basis data lokal dan broker pesan merupakan tantangan berat. Jika server crash sesaat setelah menyimpan data ke database namun sebelum pesan terkirim ke Kafka, sistem akan mengalami ketidaksinkronan data.

Transactional Outbox Pattern menyelesaikan masalah ini dengan menyimpan pesan event ke dalam tabel basis data 'outbox' di dalam transaksi ACID yang sama. Layanan background poller terpisah kemudian membaca tabel outbox dan mempublikasikan pesan ke broker dengan jaminan At-Least-Once Delivery.



Gambar 8.1: *Perbandingan Alur Komunikasi Data: HTTP Short Polling vs Saluran Persisten WebSocket Full-Duplex.*

3. Deep Dive Event-Driven Architecture & Message Brokers (Expert Level)

Dalam skala sistem terdistribusi, komunikasi sinkron HTTP antar-layanan microservices rentan terhadap fenomena Cascading Failure [29], [30], [31], [32] (kegagalan berantai ketika satu layanan hilir lambat). Event-Driven Architecture memutus rantai ini dengan komunikasi asinkron berbasis Event Broker.

Tabel 8.2: *Matriks Komparasi Protokol Komunikasi Real-Time: HTTP Polling vs SSE vs WebSocket vs gRPC.*

Protokol / Pola	Model Komunikasi	Throughput & Latensi	Kasus Penggunaan Optimal
RESTful API	Request - Response (Sinkron)	Moderat (Latensi HTTP/JSON overhead)	Operasi CRUD standar, integrasi publik API pihak ketiga.
GraphQL	Client-Driven Query (Sinkron)	Moderat (Overhead skema parsing & resolver)	Aplikasi frontend/mobile dengan kebutuhan agregasi data heterogen.
WebSockets	Bi-directional Duplex (Real-time)	Sangat Tinggi (Latensi < 10ms persisten)	Aplikasi obrolan (chat), live dashboard trading saham, kolaborasi canvas multi-user.
Message Broker (Kafka/RabbitMQ)	Publish - Subscribe (Asinkron)	Ekstrem (>100.000 msg/sec, Buffer Queue)	Dekomposisi microservices, log audit finansial, pemrosesan tugas antrean latar belakang.

> **DEEP DIVE: SKALABILITAS SISTEM REAL-TIME & DISTRIBUTED MESSAGING**
>
> 1. GraphQL N+1 Resolver Issue: Hati-hati dengan GraphQL Resolver! Selalu gunakan pustaka DataLoader (Batching & In-Memory Caching) agar eksekusi resolver tidak memicu puluhan kueri basis data berulang ke tabel anak.
>
> 2. WebSocket Horizontal Scaling: Satu instans server Node.js dapat menangani ribuan soket. Ketika server di-scale menjadi multi-instans di belakang Load Balancer, gunakan Redis Pub/Sub Adapter untuk menyebarkan pesan antar-server soket.
>
> 3. Semantik Pengiriman Pesan: Pahami perbedaan At-most-once (pesan bisa hilang), At-least-once (pesan dijamin sampai namun butuh idempotent consumer), dan Exactly-once pada Kafka/RabbitMQ.

4. Penerapan Praktis & Hands-on Code (WebSockets Server dengan Heartbeat)

Berikut implementasi peladen WebSocket berkinerja tinggi yang dilengkapi protokol Heartbeat Ping-Pong untuk memutus koneksi mati (Ghost Connections):

Listing Program: Resilient WebSocket Server with Heartbeat Healthcheck

```
// src/gateways/realtime-collaboration.gateway.ts
import { WebSocketServer, WebSocket } from 'ws';
import { IncomingMessage } from 'http';

interface ClientSocket extends WebSocket {
  userId?: string;
  isAlive: boolean;
}

export class RealtimeCollaborationServer {
  private wss: WebSocketServer;

  constructor(port: number) {
    this.wss = new WebSocketServer({ port });
    this.initializeEvents();
  }

  private initializeEvents(): void {
    this.wss.on('connection', (ws: ClientSocket, req: IncomingMessage) => {
      ws.isAlive = true;

      // Tangkap respons pong dari klien
      ws.on('pong', () => {
        ws.isAlive = true;
      });

      // Proses pesan masuk
      ws.on('message', (rawData: string) => {
        try {
          const payload = JSON.parse(rawData.toString());
          this.handleClientMessage(ws, payload);
```

```
        } catch (err) {
          ws.send(JSON.stringify({ error: "Format payload pesan harus berupa JSON valid."
}));
        }
      });

      ws.on('close', () => {
        // Bersihkan state pengguna saat koneksi ditutup
      });
    });

    // Jalankan Heartbeat Probe setiap 30 detik untuk mendeteksi soket zombie
    const heartbeatInterval = setInterval(() => {
      this.wss.clients.forEach((client) => {
        const socket = client as ClientSocket;
        if (!socket.isAlive) {
          return socket.terminate(); // Putus paksa koneksi yang tidak merespons
        }
        socket.isAlive = false;
        socket.ping(); // Kirim frame ping ke klien
      });
    }, 30000);

    this.wss.on('close', () => clearInterval(heartbeatInterval));
  }

  private handleClientMessage(sender: ClientSocket, message: any): void {
    // Siarkan pembaruan data ke seluruh klien yang terhubung (Broadcast)
    const broadcastPayload = JSON.stringify({
      type: message.type || 'DOCUMENT_UPDATE',
      data: message.data,
      timestamp: Date.now()
    });

    this.wss.clients.forEach((client) => {
      if (client !== sender && client.readyState === WebSocket.OPEN) {
        client.send(broadcastPayload);
      }
    });
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Platform lelang barang antik online 'NuurAuction' mengalami insiden kegagalan transaksi besar: tawaran harga penawaran terakhir dari penawar lelang terlambat muncul di layar pembeli lain akibat aplikasi masih menggunakan HTTP Polling setiap 3 detik. Selain itu, beban server melonjak drastis hingga 9.000 request per detik hanya untuk memeriksa apakah harga lelang sudah berubah, mengakibatkan basis data kehabisan koneksi. Ketika harga naik, beberapa penawar mengajukan komplain resmi karena merasa dirugikan oleh latensi polling.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan rekayasa ulang modul lelang dari HTTP Polling menjadi sistem Full-Duplex WebSockets berlatensi sub-15ms.

2. Integrasikan mekanisme Redis Pub/Sub untuk menyinkronkan event kenaikan tawaran harga antar-instans server WebSocket yang di-scale secara horisontal.

3. Rancang arsitektur antrean pesan RabbitMQ untuk memproses pemotongan dana deposit pemenang lelang secara asinkron dan bebas dari race condition.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK043.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 8.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK107.1 (Real-Time Communication & EDA).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Desain Protokol Komunikasi (C4)	Mampu memilih protokol yang tepat (REST vs GraphQL vs WebSocket vs Message Broker) berdasarkan analisis trade-off performa.	Memahami perbedaan protokol namun salah memilih use-case yang paling efisien.	Menerapkan HTTP Polling untuk kebutuhan sistem real-time berkecepatan tinggi.
Implementasi WebSocket Tangguh (C3/P4)	Mengimplementasikan koneksi duplex, heartbeat ping-pong, reconnection backoff, dan integrasi Redis scaling.	WebSocket berfungsi namun koneksi putus tidak ditangani dengan mekanisme heartbeat.	Server rentan memory leak akibat soket zombie tidak dibersihkan.
Optimasi GraphQL & DataLoader (C5)	Skema GraphQL modular dan seluruh resolver dioptimasi dengan DataLoader bebas masalah N+1.	GraphQL berjalan namun resolver masih memicu kueri basis data berulang.	Tidak memahami query depth limiting sehingga GraphQL API rentan serangan DOS.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 8: Poin Kunci Pembelajaran:

1. GraphQL memberikan fleksibilitas deklaratif yang mengeliminasi over-fetching dan under-fetching pada aplikasi mobile dan web.

2. WebSockets adalah standar emas untuk komunikasi dua arah real-time dengan latensi terendah dan efisiensi header frame yang tinggi.

3. Event-Driven Architecture memutus ketergantungan antar-layanan (decoupling) dan meningkatkan ketahanan sistem terhadap beban lonjakan trafik.

7.1 Glosarium Istilah Kunci

• DataLoader: Pustaka utilitas untuk mereduksi kueri basis data pada GraphQL melalui teknik batching dan per-request caching.

• Event-Driven Architecture (EDA): Pola arsitektur perangkat lunak di mana alur eksekusi dipicu oleh produksi, deteksi, dan konsumsi peristiwa (events).

• Full-Duplex: Komunikasi data dua arah simultan di mana pengirim dan penerima dapat mentransfer data secara bersamaan pada satu saluran soket.

• Over-fetching: Kondisi di mana klien menerima data atribut yang jauh lebih banyak daripada yang sebenarnya dibutuhkan.

• Pub/Sub (Publish/Subscribe): Pola perpesanan di mana produsen pesan mempublikasikan pesan ke topik perantara tanpa mengetahui identitas penerima pesan secara spesifik.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan secara teknis bagaimana DataLoader membatasi kueri basis data pada GraphQL Resolver ketika menampilkan daftar 50 postingan dan masing-masing profil pengarangnya!

2. [Evaluasi - C5] Bandingkan kelebihan dan kelemahan protokol Server-Sent Events (SSE) dibandingkan WebSockets untuk use-case notifikasi satu arah dari server ke browser!

3. [Kreasi - C6] Rancang skema arsitektur Event-Driven menggunakan RabbitMQ untuk mengorkestrasikan alur pemesanan makanan (validasi pembayaran, notifikasi restoran, alokasi kurir, dan pengiriman resi email) secara asinkron!

BAB 9
OTENTIKASI, OTORISASI, & PENGAMANAN WEB MODERN (JWT, OAUTH2/OIDC, ZERO TRUST, & OWASP TOP 10)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini memberikan panduan mendalam tentang rekayasa keamanan siber aplikasi web (Web Application Security) tingkat lanjut. Meliputi arsitektur Zero Trust, token nir-status vs berstatus (JWT vs Session), otorisasi berbasis peran (RBAC & ABAC), protokol delegasi identitas OAuth 2.0 / OpenID Connect, manajemen siklus hidup Refresh Token rotasional, serta strategi mitigasi komprehensif terhadap ancaman kritis OWASP Top 10 (SQL Injection, XSS, CSRF, SSRF, dan CORS misconfiguration).

Tabel 9.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 9.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software QA Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi arsitektur solusi sistem perangkat lunak yang aman, tangguh, dan teruji.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu mengidentifikasi kerentanan keamanan web dan menerapkan mekanisme proteksi berlapis.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.2: Mampu merekayasa sistem otentikasi JWT/OAuth2 aman, RBAC middleware, dan pertahanan OWASP (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu merancang siklus otentikasi stateless menggunakan JWT (Short-Lived Access Token & Rotating Refresh Token) yang kebal terhadap pencurian kredensial.

2. Mampu membangun middleware otorisasi granular berbasis Role-Based Access Control (RBAC) dan Attribute-Based Access Control (ABAC).

3. Mampu memitigasi vektor serangan web populer: SQLi (Parameterized Query), XSS (Content Security Policy & Sanitization), CSRF (SameSite Cookies), dan SSRF.

2. Pengantar & Konsep Teoretis: Paradigma Zero Trust & Otentikasi Modern

2.1 Filosofi Zero Trust Security dalam Rekayasa Web

Model keamanan perimeter tradisional mengasumsikan bahwa seluruh lalu lintas di dalam jaringan intranet bersifat aman (model 'benteng dan parit'). Paradigma Zero Trust membatalkan asumsi berbahaya ini dengan prinsip: 'Never Trust, Always Verify' (Jangan pernah percaya, selalu verifikasi setiap saat).

Prinsip Zero Trust: Setiap request yang masuk ke API Gateway maupun komunikasi antar-layanan internal (service-to-service) wajib diotentikasi secara ketat guna memitigasi kerentanan keamanan API modern [33], [34], [35], [36], diotorisasi secara kontekstual, dan dienkripsi menggunakan Mutual TLS (mTLS), tanpa memandang dari subnet mana paket data berasal.

2.2 Anatomi JSON Web Token (JWT) & Keamanan Penyimpanan Kredensial

Struktur JWT: JWT adalah standar terbuka (RFC 7519) yang mendefinisikan format kompak dan mandiri (self-contained) untuk mentransfer informasi klaim identitas antar-pihak secara aman. JWT terdiri dari 3 segmen yang dipisahkan oleh tanda titik (.):

1. Header: Memuat tipe token (JWT) dan algoritma penandatangan kriptografi (misal: HS256 atau RS256).

2. Payload: Memuat klaim identitas pengguna (sub/user_id, email, roles, exp/expiration time, jti/token id).

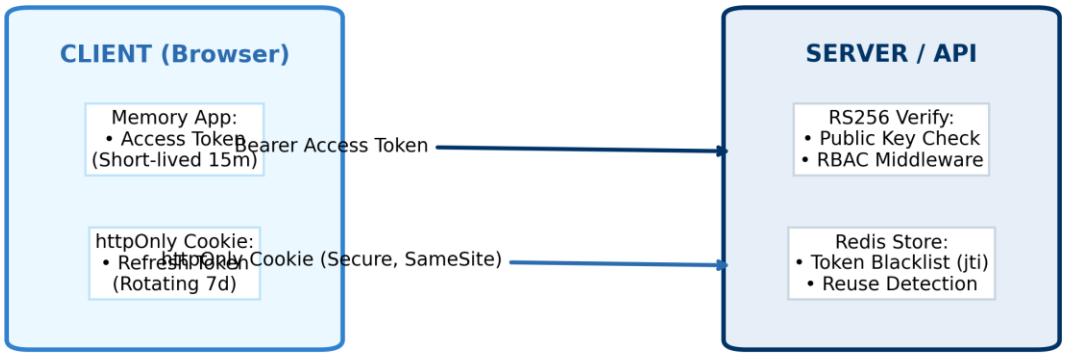
3. Signature: Tanda tangan digital kriptografis yang dihasilkan dengan mengenkripsi Header dan Payload menggunakan kunci rahasia.

Penyimpanan Token Aman: Salah satu kesalahan paling fatal dalam rekayasa web adalah menyimpan JWT di LocalStorage peramban. LocalStorage dapat diakses langsung oleh skrip JavaScript apa pun di halaman tersebut, sehingga token dapat dicuri seketika jika situs terpapar celah Cross-Site Scripting (XSS). Standar emas industri adalah menyimpan Access Token berumur pendek (10-15 menit) di dalam memori variabel aplikasi, dan menyimpan Refresh Token (7 hari) di dalam Cookie dengan atribut httpOnly, Secure, dan SameSite=Strict.

2.3 Protokol Delegasi Modern: OAuth 2.0, OpenID Connect (OIDC), & PKCE

Dalam ekosistem multi-aplikasi, berbagi password secara langsung adalah anti-pattern yang berbahaya. OAuth 2.0 memfasilitasi otorisasi terdelegasi (Delegated Authorization), sementara OpenID Connect (OIDC) menambahkan lapisan identitas otentikasi di atas OAuth 2.0 menggunakan ID Token.

Untuk aplikasi Single Page Application (SPA) dan mobile yang tidak dapat menyimpan client_secret secara aman, standar industri mewajibkan penggunaan Authorization Code Flow dengan PKCE (Proof Key for Code Exchange) guna mencegah intersepsi token di jaringan publik.



Gambar 9.1: *Arsitektur Otentikasi Token Aman: Memory Access Token, httpOnly Refresh Token, dan RS256 Verification.*

3. Deep Dive Pertahanan Vektor Serangan OWASP Top 10 (Expert Level)

Tabel 9.2: *Matriks Analisis Komparatif Algoritma Penandatanganan Kriptografi JWT: Simetris HS256 vs Asimetris RS256.*

Vektor Serangan	Mekanisme Eksploitasi	Mitigasi Arsitektural Utama
SQL Injection (SQLi)	Penyerang menyisipkan sintaks SQL jahat melalui form/URL input yang tidak divalidasi.	Parameterized Queries (Prepared Statements), penggunaan ORM type-safe, input schema sanitization.
Cross-Site Scripting (XSS)	Penyerang menyuntikkan skrip JS berbahaya ke dalam browser korban untuk mencuri data sesi.	Content Security Policy (CSP), HTML entity encoding, penyimpanan token di httpOnly cookies.
Cross-Site Request Forgery (CSRF)	Memaksa browser korban mengeksekusi request yang tidak diinginkan ke aplikasi tempat korban login.	SameSite=Lax/Strict Cookies, Anti-CSRF Synchronizer Tokens, verifikasi custom request header.
Server-Side Request Forgery (SSRF)	Penyerang memanipulasi server untuk melakukan request ke jaringan privat internal (misal AWS metadata).	Validasi dan pembatasan IP Whitelist, memblokir resolve DNS ke alamat loopback dan subnet privat (RFC 1918).

📌 DEEP DIVE: PROTOKOL KEAMANAN TINGKAT PRODUKSI

1. Algoritma 'none' Attack: Selalu kunci algoritma verifikasi JWT (misal: algorithms: ['RS256']) di backend untuk mencegah penyerang memalsukan token dengan header alg: 'none'.

2. Token Blacklisting & Revocation: Simpan jti (JWT ID) token yang di-revoke ke Redis dengan TTL sesuai waktu kedaluwarsa token untuk mendukung fitur 'Logout from all devices'.

3. HTTP Security Headers: Wajib pasang Helmet di Express untuk mengaktifkan HSTS, X-Content-Type-Options:

nosniff, dan Frameguard (anti clickjacking).

3.1 Keamanan Tingkat Lanjut: Pertahanan Brute-Force & Credential Stuffing

Serangan Credential Stuffing terjadi ketika peretas menggunakan database kata sandi bocor dari situs lain untuk mencoba login otomatis ke sistem web. Untuk memitigasinya, arsitektur keamanan modern menerapkan:

1. Exponential Login Delay: Menambahkan jeda waktu bertingkat (misal: 1 detik, 2 detik, 4 detik, 8 detik) setiap kali upaya otentikasi gagal secara berturut-turut.

2. CAPTCHA Adaptif (Cloudflare Turnstile / reCAPTCHA v3): Menilai skor risiko perilaku pengguna tanpa mengganggu kenyamanan pengguna normal.

3. Multi-Factor Authentication (MFA / TOTP): Mengintegrasikan algoritma RFC 6238 Time-Based One-Time Password menggunakan aplikasi authenticator.

4. Penerapan Praktis & Hands-on Code (Secure JWT Auth & RBAC)

Berikut implementasi middleware otentikasi JWT asimetris (RS256) dan otorisasi berbasis peran (RBAC) pada Express.js:

Listing Program: Secure JWT Authentication & Granular RBAC Middleware

```
// src/middleware/auth.middleware.ts
import { Request, Response, NextFunction } from 'express';
import jwt from 'jsonwebtoken';

export interface AuthenticatedUser {
  userId: string;
  email: string;
  roles: string[];
}

declare global {
  namespace Express {
    interface Request {
      user?: AuthenticatedUser;
    }
  }
}

// 1. Middleware Verifikasi Token Asimetris
export const authenticateToken = (req: Request, res: Response, next: NextFunction) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.startsWith('Bearer ') ? authHeader.split(' ')[1] :
null;

  if (!token) {
    return res.status(401).json({ error: "Access Denied: Token otentikasi tidak ditemukan."
});
  }
```

```
  try {
    const publicKey = process.env.JWT_PUBLIC_KEY || '';
    const decoded = jwt.verify(token, publicKey, { algorithms: ['RS256'] }) as
AuthenticatedUser;
    req.user = decoded;
    next();
  } catch (err: any) {
    return res.status(403).json({ error: "Forbidden: Token tidak valid atau telah
kedaluwarsa." });
  }
};

// 2. Middleware Otorisasi Berbasis Peran (RBAC)
export const requireRoles = (allowedRoles: string[]) => {
  return (req: Request, res: Response, next: NextFunction) => {
    if (!req.user) return res.status(401).json({ error: "Unauthorized" });

    const hasRole = req.user.roles.some(r => allowedRoles.includes(r));
    if (!hasRole) {
      return res.status(403).json({ error: "Forbidden: Anda tidak memiliki hak akses untuk
operasi ini." });
    }
    next();
  };
};
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Sistem keuangan universitas 'NuurBilling' mengalami insiden kebocoran data sensitif: seorang mahasiswa berhasil mengakses slip gaji dosen dan memodifikasi status pembayaran SPP menjadi 'LUNAS' tanpa melakukan transfer bank. Investigasi forensik menemukan bahwa JWT disimpan di localStorage peramban, kunci rahasia HMAC di-hardcode di repositori publik GitHub, dan endpoint pembaruan pembayaran tidak memverifikasi peran pengguna (hanya memeriksa apakah token valid).

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Lakukan audit keamanan terhadap arsitektur token NuurBilling dan perbaiki algoritma penandatanganan menjadi pasangan kunci asimetris RSA-256.

2. Pindahkan penyimpanan Refresh Token ke httpOnly Secure Cookie dengan mekanisme Refresh Token Rotation otomatis.

3. Rancang RBAC middleware yang ketat untuk mengisolasi akses endpoint keuangan dan tambahkan audit log ke database.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK043.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 9.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK108.1 (Zero Trust Security & OWASP Top 10).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Desain Otentikasi & Token (C4)	Menerapkan RS256/Asymmetric encryption, token rotation, jti blacklisting di Redis, dan cookie httpOnly.	Menggunakan JWT HS256 standar namun token disimpan di tempat aman.	Menyimpan JWT di localStorage tanpa enkripsi.
Otorisasi Granular RBAC/ABAC (C3/P4)	Membangun RBAC middleware dinamis yang membatasi hak akses berdasarkan peran dan kepemilikan resource.	RBAC sederhana berbasis string hardcoded di controller.	Endpoint API dibiarkan terbuka tanpa pengecekan peran.
Pertahanan OWASP Top 10 (C5)	Mengimplementasikan CSP ketat, Helmet headers, parameterized query, dan sanitasi input Zod.	Melindungi dari SQLi namun mengabaikan XSS dan CORS.	Kode rentan terhadap injeksi langsung dan kebocoran credential.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 9: Poin Kunci Pembelajaran:

1. Keamanan siber aplikasi web harus dirancang sejak fase inisiasi arsitektur (Security-by-Design), bukan sebagai tambalan di akhir proyek.

2. Kombinasi token akses di memori dan Refresh Token di httpOnly cookie memberikan keamanan maksimal dari serangan XSS dan CSRF.

3. Zero Trust menuntut otorisasi granular di setiap lapisan layanan terdistribusi.

7.1 Glosarium Istilah Kunci

• ABAC: Attribute-Based Access Control, otorisasi berdasarkan konteks waktu, lokasi, dan atribut dinamis entitas.

• CSP (Content Security Policy): Header HTTP yang membatasi sumber skrip dan aset yang diizinkan dimuat oleh peramban.

• httpOnly Cookie: Cookie yang tidak dapat dibaca oleh skrip JavaScript peramban, memitigasi pencurian sesi via XSS.

• RS256: Algoritma tanda tangan digital asimetris yang menggunakan private key untuk menandatangani dan public key untuk memverifikasi token.

• Zero Trust: Paradigma keamanan yang menolak kepercayaan implisit dan mewajibkan verifikasi ketat pada setiap permintaan data.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan mengapa menyimpan Access Token di LocalStorage sangat berbahaya jika aplikasi web memiliki celah Cross-Site Scripting (XSS)!

2. [Evaluasi - C5] Evaluasi kelebihan dan kelemahan penggunaan Token Asimetris (RS256) dibanding Simetris (HS256) pada arsitektur Microservices dengan puluhan layanan hilir!

3. [Kreasi - C6] Rancanglah alur Refresh Token Rotation yang mendeteksi penggunaan ulang token curian (Reuse Detection) dan otomatis menganulir seluruh sesi pengguna di Redis!

BAB 10
OPTIMASI KINERJA, STRATEGI CACHING MULTI-TIER, & SKALABILITAS SISTEM (REDIS, EDGE CACHING, & RATE LIMITING)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini mengupas tuntas teknik rekayasa performa tinggi dan efisiensi komputasi hijau (Green Computing) pada sistem web enterprise: hierarki Caching Multi-Tier (Browser Cache, CDN/Edge Cache, In-Memory Redis, Database Query Cache), mitigasi anomali caching kritis (Cache Stampede, Cache Avalanche, Cache Penetration), serta algoritma pembatasan lalu lintas (Token Bucket, Leaky Bucket, dan Sliding Window Rate Limiting).

Tabel 10.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 10.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software QA Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi kebutuhan computing yang efisien dan arsitektur solusi sistem perangkat lunak yang hemat daya.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu mengoptimasi latensi, throughput, dan efisiensi konsumsi daya/sumber daya komputasi sistem web.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.2: Mampu merancang arsitektur caching multi-tier dan algoritma rate limiting (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu merancang topologi caching bertingkat (Multi-Tier Caching) dari Edge CDN peramban hingga In-Memory Redis pada klaster server.

2. Mampu menerapkan strategi Cache Invalidation (Cache-Aside, Write-Through, Write-Behind) dan memitigasi fenomena Cache Stampede menggunakan algoritma probabilistik XFetch.

3. Mampu mengimplementasikan sistem Rate Limiting terdistribusi menggunakan algoritma Sliding Window Counter pada Redis untuk proteksi serangan DDoS.

2. Pengantar & Konsep Teoretis: Hierarki Caching Multi-Tier

2.1 Piramida Kecepatan & Biaya Akses Data

Tingkatan Caching: Prinsip dasar caching adalah menyimpan hasil komputasi berat pada media penyimpanan yang lebih cepat dan efisien sesuai pola tata kelola kinerja cerdas [37], [38], [39], [40] dan lebih dekat secara geografis dengan pengguna.

1. Tier 1: Client/Browser Cache (HTTP Headers: Cache-Control, ETag, Stale-While-Revalidate). Beroperasi pada latensi 0 milidetik.

2. Tier 2: Edge CDN (Cloudflare, AWS CloudFront). Menyimpan konten statis dan dinamis di ratusan Point of Presence (PoP) dunia. Latensi 10-30 milidetik.

3. Tier 3: In-Memory Application Cache (Redis / Memcached). Menyimpan data objek JSON terstruktur di RAM server. Latensi 1-5 milidetik.

4. Tier 4: Database Storage (SSD NVMe / Disk Relational). Latensi 10-100+ milidetik.

2.2 Algoritma Rate Limiting: Token Bucket, Leaky Bucket, & Sliding Window

Untuk melindungi backend dari kebanjiran trafik dan serangan brute-force, beberapa algoritma pembatasan lalu lintas diterapkan:

1. Token Bucket: Token ditambahkan ke dalam ember secara berkala dengan laju konstan. Setiap request mengonsumsi 1 token. Algoritma ini memungkinkan terjadinya burst traffic singkat selama kapasitas ember masih mencukupi.

2. Leaky Bucket: Request masuk ke dalam antrean ember dan dikeluarkan dengan kecepatan konstan, menghasilkan keluaran trafik yang sangat halus dan stabil.

3. Sliding Window Counter: Memadukan efisiensi memori Fixed Window dengan keakuratan Sliding Window Log pada Redis.

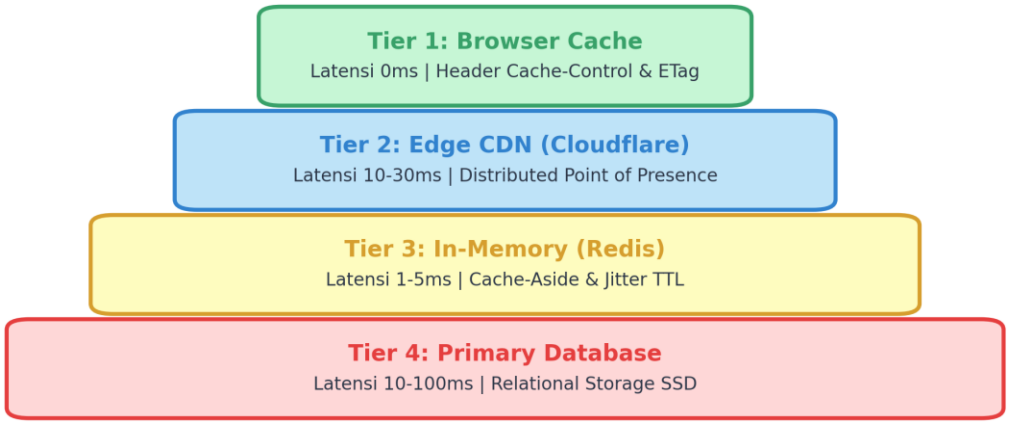
2.4 Green Computing & Efisiensi Energi Pusat Data

Komputasi awan modern mengonsumsi energi listrik global dalam jumlah masif. Praktik rekayasa web yang efisien tidak hanya mempercepat waktu muat halaman, tetapi secara langsung mengurangi emisi karbon pusat data (Green Computing):

1. Payload Compression: Mengaktifkan kompresi Brotli (br) yang 20% lebih efisien dibanding Gzip untuk seluruh berkas teks statis (HTML, CSS, JS).

2. Client-Side Asset Offloading: Memanfaatkan header Cache-Control 'immutable' untuk aset bertanda hash unik (misal: main.a8f9c1.js) sehingga peramban tidak pernah mengirim kueri validasi ulang selama 1 tahun penuh.

3. Early Hints (HTTP 103): Menginstruksikan peramban untuk mulai mengunduh stylesheet dan font penting secara paralel saat server backend masih memproses kueri basis data.



Gambar 10.1: *Piramida Hierarki Caching Multi-Tier: Browser Cache, Edge CDN, In-Memory Redis, dan Database Storage.*

3. Deep Dive Anomali Caching & Mitigasi Mutakhir (Expert Level)

Tabel 10.2: *Matriks Perbandingan Karakteristik Algoritma Rate Limiting: Token Bucket vs Leaky Bucket vs Sliding Window.*

Fenomena Kegagalan Cache	Mekanisme Masalah	Mitigasi Arsitektural Utama
Cache Stampede (Thundering Herd)	Kunci cache data populer kedaluwarsa seketika; ribuan request konkuren masuk bersamaan menghantam database primer.	Probabilistic Early Expiration (XFetch Algorithm) atau Mutex Lock saat fetch database.
Cache Avalanche	Ratusan ribu kunci cache kedaluwarsa pada detik yang persis sama, membuat utilisasi CPU database overload.	Menambahkan TTL Jitter (waktu kedaluwarsa acak, misal TTL + random(1-300 detik)).
Cache Penetration	Request terus-menerus mencari ID data yang memang tidak ada di DB, sehingga cache selalu miss dan query lolos ke DB.	Simpan nilai penanda null dengan TTL pendek, atau gunakan Bloom Filter di depan cache.

> **DEEP DIVE: RESILIENSI SISTEM CACHING TINGKAT LANJUT**
>
> 1. Cache Invalidation Strategy: 'Ada dua hal sulit dalam ilmu komputer: cache invalidation dan penamaan variabel' (Phil Karlton). Gunakan pola Event-Driven Invalidation daripada hanya mengandalkan TTL statis.
>
> 2. Rate Limiting Sliding Window: Algoritma Fixed Window rentan terhadap lonjakan trafik di perbatasan menit. Gunakan Sliding Window Log/Counter di Redis untuk proteksi DOS yang akurat.
>
> 3. Green Computing: Penggunaan caching yang optimal mereduksi utilisasi CPU server hingga 85%, menurunkan konsumsi energi listrik pusat data secara signifikan.

3.1 Arsitektur Caching Terdistribusi: Redis Cluster & Resilient Failover

Ketika volume data in-memory melampaui kapasitas RAM satu server fisik, Redis Cluster menyediakan sharding otomatis melintasi 16.384 hash slots. Setiap master node didukung oleh satu atau lebih replica nodes dengan mekanisme Sentinel untuk pemilihan leader baru otomatis saat terjadi server down.

Pengembang juga harus memperhatikan kompresi data JSON sebelum disimpan ke Redis menggunakan algoritma lz4 atau zstd, yang terbukti mampu mereduksi pemakaian RAM hingga 65% tanpa menambah latensi CPU yang berarti.

Selain itu, strategi Cache Warm-Up dijalankan sebelum kampanye promosi besar untuk mengisi kunci data populer terlebih dahulu ke dalam Redis, menghindari kekosongan cache saat lalu lintas memuncak.

4. Penerapan Praktis & Hands-on Code (Redis Cache-Aside & Rate Limiter)

Berikut implementasi pola Cache-Aside yang dilengkapi mitigasi TTL Jitter dan proteksi Cache Penetration menggunakan Node.js dan Redis:

Listing Program: Resilient Cache-Aside Pattern with Jitter & Anti-Penetration

```
// src/services/cached-catalog.service.ts
import Redis from 'ioredis';

const redis = new Redis(process.env.REDIS_URL || 'redis://localhost:6379');

export class CachedCatalogService {
  public async getProductDetails(productId: string, db: any) {
    const cacheKey = `product:${productId}`;

    // 1. Periksa In-Memory Cache (Tier 3)
    const cachedData = await redis.get(cacheKey);
    if (cachedData) {
      const parsed = JSON.parse(cachedData);
      if (parsed.isNull) return null; // Tangani Cache Penetration
      return parsed;
    }

    // 2. Cache Miss: Ambil dari Database Primer
    const product = await db.products.findUnique({ where: { id: productId } });

    if (!product) {
      // Mitigasi Cache Penetration: Simpan penanda null selama 60 detik
      await redis.set(cacheKey, JSON.stringify({ isNull: true }), 'EX', 60);
      return null;
    }

    // 3. Simpan ke Cache dengan TTL + Jitter (Mitigasi Cache Avalanche)
    const baseTTL = 3600; // 1 jam
    const jitter = Math.floor(Math.random() * 300); // 0-5 menit variasi acak
    await redis.set(cacheKey, JSON.stringify(product), 'EX', baseTTL + jitter);

    return product;
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Portal pengumuman seleksi mandiri universitas 'NuurSelection' menerima serbuan 60.000 peserta secara serentak pada pukul 13.00 WIB. Server web seketika down akibat fenomena Cache Stampede pada kunci pengumuman utama dan serangan bot scraper yang mengirimkan 800 request per detik per IP. Mahasiswa diminta merekayasa topologi Caching Multi-Tier (Cloudflare CDN + Redis) dengan algoritma XFetch dan middleware Rate Limiting Sliding Window berbasis Redis dengan batas kuota 60 request per menit per alamat IP.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Rancang topologi Caching Multi-Tier dan tentukan konfigurasi HTTP header Cache-Control (s-maxage, stale-while-revalidate) pada Edge CDN.

2. Implementasikan pola Cache-Aside yang kebal terhadap Cache Stampede dan Avalanche pada basis data produk/pengumuman.

3. Bangun skrip middleware Rate Limiter berbasis Sliding Window Counter menggunakan perintah atomik Redis Lua Script.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK043.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 10.3: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK109.1 (Multi-Tier Caching & Rate Limiting).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Desain Caching Multi-Tier (C4)	Mampu mengonfigurasi cache browser, CDN headers, dan Redis secara harmonis dengan hit ratio > 95%.	Menerapkan Redis namun mengabaikan cache headers di level peramban/CDN.	Tidak menerapkan caching sama sekali.
Mitigasi Anomali Cache (C5)	Mengimplementasikan solusi Cache Stampede (Mutex/XFetch), Avalanche (Jitter), dan Penetration (Null cache).	Mengetahui istilah anomali namun belum menerapkan jitter atau penanganan null.	Cache rentan memicu crash database saat kunci kedaluwarsa.
Implementasi Rate Limiting (C3/P4)	Membangun Sliding Window Rate Limiter terdistribusi pada Redis yang tahan terhadap bot abuse.	Rate limiter sederhana berbasis memori lokal yang gagal saat server di-scale multi-instans.	Tidak ada mekanisme pembatasan trafik.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 10: Poin Kunci Pembelajaran:

1. Caching Multi-Tier adalah kunci utama mencapai waktu respons sub-10 milidetik dan efisiensi biaya server awan.

2. Anomali seperti Cache Stampede dan Avalanche harus dimitigasi dengan TTL Jitter dan probabilistic expiration.

3. Rate Limiting melindungi ketersediaan infrastruktur web dari serangan denial-of-service dan brute-force.

7.1 Glosarium Istilah Kunci

• Cache Hit Ratio: Persentase permintaan data yang berhasil dilayani langsung oleh cache tanpa menyentuh database primer.

• Cache Stampede: Lonjakan kueri serentak ke basis data saat data populer pada cache kedaluwarsa secara mendadak.

• Green Computing: Praktik perancangan sistem perangkat lunak yang meminimalkan konsumsi daya komputasi dan jejak karbon.

• Sliding Window Counter: Algoritma rate limiting yang menghitung kuota permintaan secara mulus lintas jendela waktu geser.

• Stale-While-Revalidate: Pola HTTP Caching di mana peramban menampilkan data lama seketika sambil mengambil data baru di latar belakang.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan perbedaan mekanisme kerja HTTP header Cache-Control: no-cache vs no-store! Kapan masing-masing wajib digunakan?

2. [Evaluasi - C5] Evaluasi kelebihan dan kelemahan strategi Write-Through Caching dibandingkan Cache-Aside pada sistem e-commerce dengan pergerakan stok barang sangat cepat!

3. [Kreasi - C6] Buatlah skrip Lua untuk Redis yang mengimplementasikan Sliding Window Rate Limiting atomik dalam satu kali panggilan jaringan!

BAGIAN IV: KEAMANAN, KINERJA, & KECERDASAN KOMPUTASI

BAB 11
INTEGRASI ALGORITMA CERDAS & ANALITIK DATA PADA APLIKASI WEB
(K-MEANS CLUSTERING, RULE MINING, & AGENTIC AI WORKFLOWS)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini mengintegrasikan materi riset orisinal dosen (Andri Triyono, S.T., M.T.) ke dalam kurikulum pembelajaran rekayasa web modern. Membahas integrasi praktis algoritma Machine Learning (Clustering K-Means), Association Rule Mining (Dynamic Menu Bundling), serta orkestrasi Agentic AI Workflows / Large Language Models (LLM) ke dalam backend aplikasi web.

Tabel 11.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 11.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL10: Mampu menghubungkan pengetahuan algoritma cerdas dengan implementasi praktis aplikasi web.
Capaian Pembelajaran MK (CPMK)	CPMK103: Mampu mengintegrasikan modul machine learning dan AI ke dalam pipeline backend aplikasi web.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK103.2: Mampu menerapkan algoritma K-Means, Rule Mining, dan integrasi API LLM cerdas (Taksonomi Bloom: C3 - Penerapan & C4 - Analisis).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu mengimplementasikan algoritma Unsupervised Learning K-Means untuk segmentasi pengguna dan data analitik web.

2. Mampu merancang mesin rekomendasi Association Rule Mining untuk Dynamic Product Bundling pada platform web e-commerce.

3. Mampu membangun arsitektur Agentic AI Workflow dengan pemanggilan Function Calling (Tools) dan Streaming Response via Server-Sent Events (SSE).

2. Pengantar & Konsep Teoretis: Integrasi Kecerdasan Komputasi

2.1 Integrasi Riset Dosen 1: K-Means Clustering pada Web Analitik

Penerapan K-Means: Berdasarkan penelitian Andri Triyono dkk. [2] serta integrasi model komputasi awan dan AI [41], [42], [43], [44] (Jurnal TRANSFORMASI), algoritma K-Means efektif mengelompokkan data spasial dan temporal ke dalam K klaster optimal berdasarkan jarak Euclidean. Dalam aplikasi web, K-Means digunakan untuk mengelompokkan pengguna aktif berdasarkan frekuensi login, durasi kunjungan, dan nilai belanja guna menghasilkan kampanye personalisasi otomatis.

2.2 Integrasi Riset Dosen 2: Agentic Workflow & Dynamic Bundling

Penerapan Agentic Rule Mining: Berdasarkan publikasi Andri Triyono & Kartika Imam Santoso [1] (SMARTICS Journal, 2024), penggabungan Agentic Workflow dengan Trend-Aware Rule Mining mampu menghasilkan rekomendasi bundling menu yang dinamis dan kontekstual terhadap tren data terkini. Sistem web modern dapat memanfaatkan agen otonom untuk menganalisis keranjang belanja secara real-time.

2.3 Evaluasi Metrik Kualitas AI: Silhouette Score, Elbow Method, & Confidence Ratio

Dalam mengintegrasikan model Machine Learning ke aplikasi web, penentuan parameter algoritma harus divalidasi secara matematis:

• Elbow Method & Silhouette Coefficient: Digunakan untuk menentukan jumlah klaster K paling optimal pada algoritma K-Means dengan mengevaluasi rasio jarak intra-cluster terhadap inter-cluster.

• Association Metrics (Support, Confidence, Lift): Pada mesin rekomendasi Dynamic Bundling, Lift Ratio > 1.0 membuktikan bahwa pembelian Produk A dan Produk B terjadi bukan karena kebetulan acak, melainkan memiliki korelasi perilaku belanja yang signifikan.

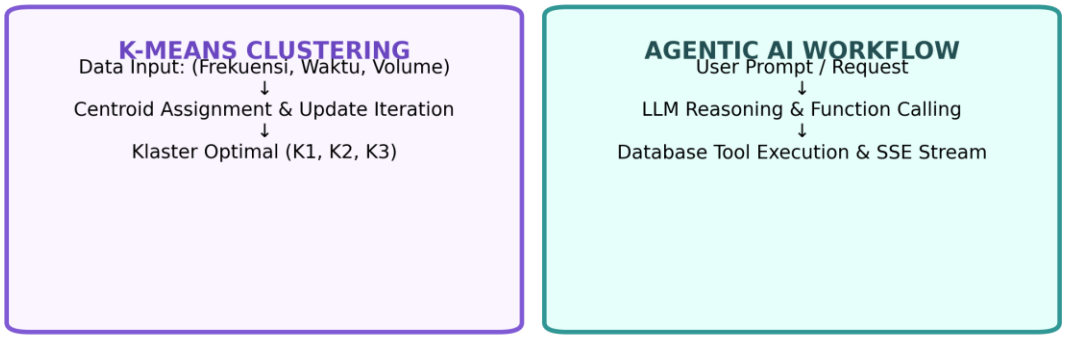
2.4 Etika AI & Keamanan Privasi Data (Privacy-Preserving AI)

Integrasi model kecerdasan buatan pada aplikasi web menuntut kepatuhan ketat terhadap privasi data pengguna dan etika komputasi:

1. Data Anonymization & PII Masking: Seluruh informasi identitas pribadi (Personally Identifiable Information seperti NIK, nomor telepon, dan rekam medis) wajib disamarkan atau di-hash sebelum dikirimkan ke model LLM eksternal.

2. Guardrails & Output Sanitization: Mengimplementasikan sistem filter guardrails untuk memastikan output teks AI bebas dari halusinasi berbahaya, ujaran kebencian, atau instruksi ilegal.

3. Auditability & Explainability: Menyediakan log riwayat penalaran AI (Reasoning Trace) sehingga setiap keputusan rekomendasi otomatis dapat diaudit dan dijelaskan kepada pengguna.



Gambar 11.1: *Integrasi Komputasi Cerdas: K-Means Clustering Analitik dan Agentic AI Function Calling Workflow.*

3. Deep Dive Arsitektur LLM & Function Calling (Expert Level)

Dalam mengintegrasikan model AI generatif ke dalam backend web, pengembang harus menghindari latency freeze dengan memanfaatkan Server-Sent Events (SSE) untuk streaming token dan Function Calling (Tools) agar AI dapat berinteraksi dengan database internal secara deterministik.

> **DEEP DIVE: REKAYASA SISTEM WEB BERBASIS KECERDASAN ARTIFISIAL**
>
> 1. Asynchronous AI Background Jobs: Jangan menjalankan komputasi model AI berat (clustering ratusan ribu baris) secara sinkron pada request HTTP. Jalankan melalui antrean BullMQ / Redis Worker.
>
> 2. Prompt Injection Defense: Lakukan sanitasi ketat terhadap input pengguna yang dimasukkan ke dalam context prompt LLM, dan gunakan validasi skema JSON (Zod) untuk response AI.
>
> 3. Semantic Vector Search: Simpan embedding teks di Vector Database (Pinecone / Pgvector) untuk fitur rekomendasi dan Retrieval-Augmented Generation (RAG).

3.1 Pipeline Data Analitik Real-Time & Arsitektur Vector Database

Dalam implementasi sistem perekomendasi cerdas, data transaksi web diproses melalui pipeline ekstraksi fitur otomatis. Data numerik dinormalisasi menggunakan Z-Score Standardization agar atribut dengan skala besar tidak mendominasi perhitungan jarak Euclidean pada algoritma K-Means.

Untuk fitur pencarian semantik cerdas berbasis teks, teks dikonversi menjadi embedding vektor berdimensi tinggi (misal: 1536 dimensi via text-embedding-3) dan diindeks menggunakan Hierarchical Navigable Small World (HNSW) pada basis data vektor Pgvector atau Pinecone, memungkinkan pencarian semantik kemiripan kosinus (Cosine Similarity) dalam waktu < 5 milidetik.

3.2 Arsitektur RAG (Retrieval-Augmented Generation) pada Web Enterprise

Untuk mencegah halusinasi model bahasa (LLM), arsitektur RAG menghubungkan model AI dengan basis data dokumen internal perusahaan:

1. Tahap Ingestion: Dokumen PDF/HTML dipotong menjadi chunk kecil (500 token), dikonversi menjadi embedding vektor, dan disimpan di Pgvector.

2. Tahap Retrieval: Kueri pertanyaan pengguna dikonversi menjadi vektor dan dicari dokumen dengan Cosine Similarity tertinggi.

3. Tahap Generation: Konteks dokumen yang relevan disuntikkan ke dalam prompt sistem sebagai landasan jawaban faktual AI.

4. Penerapan Praktis & Hands-on Code (K-Means & LLM Agent)

Berikut implementasi algoritma K-Means Clustering dalam TypeScript murni yang siap diintegrasikan ke dalam backend Node.js:

Listing Program: Algoritma K-Means Clustering untuk Segmentasi Data Web

```
// src/services/kmeans-clustering.service.ts
export class KMeansService {
  public static cluster(data: number[][], k: number, maxIterations = 50): number[] {
    // 1. Inisialisasi Centroid Acak
    let centroids = data.slice(0, k);
    let assignments = new Array(data.length).fill(0);

    for (let iter = 0; iter < maxIterations; iter++) {
      let changed = false;

      // 2. Tahap Assignment: Hitung Jarak Euclidean Terdekat
      for (let i = 0; i < data.length; i++) {
        let minDist = Infinity;
        let closestCentroid = 0;
        for (let c = 0; c < k; c++) {
          const dist = Math.hypot(data[i][0] - centroids[c][0], data[i][1] -
centroids[c][1]);
          if (dist < minDist) {
            minDist = dist;
            closestCentroid = c;
          }
        }
        if (assignments[i] !== closestCentroid) {
          assignments[i] = closestCentroid;
          changed = true;
        }
      }

      if (!changed) break;

      // 3. Tahap Update: Hitung Ulang Titik Rata-rata Centroid
      const newCentroids = Array.from({ length: k }, () => [0, 0]);
      const counts = new Array(k).fill(0);
      for (let i = 0; i < data.length; i++) {
        const c = assignments[i];
        newCentroids[c][0] += data[i][0];
        newCentroids[c][1] += data[i][1];
        counts[c]++;
      }
```

```
      centroids = newCentroids.map((c, idx) => counts[idx] ? [c[0]/counts[idx],
c[1]/counts[idx]] : centroids[idx]);
    }

    return assignments;
  }
}
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Platform logistik kesehatan 'NuurTransfusion' membutuhkan modul cerdas untuk mengelompokkan permintaan kantong darah berdasarkan golongan dan lokasi rumah sakit di Kabupaten Grobogan. Selain itu, manajemen menginginkan chatbot asisten cerdas yang mampu memeriksa ketersediaan stok darah secara real-time via database. Mahasiswa diminta mengintegrasikan modul K-Means untuk pemetaan klaster stok dan membangun integrasi AI Agent dengan Function Calling ke database PMI.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Implementasikan algoritma K-Means untuk mengelompokkan data permintaan darah berdasarkan waktu dan wilayah.

2. Bangun endpoint Server-Sent Events (SSE) untuk streaming respons asisten AI cerdas ke antarmuka web pengguna.

3. Rancang skema Function Calling yang membatasi AI hanya dapat membaca data yang diizinkan dan menolak instruksi berbahaya (prompt injection).

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK103.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 11.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK110.1 (K-Means Clustering & Agentic AI).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Penerapan Algoritma ML (C3/P4)	Menerapkan K-Means / Rule Mining secara efisien dengan penanganan konvergensi dan normalisasi data.	Algoritma berjalan namun tidak melakukan normalisasi data awal.	Algoritma salah secara matematis / infinite loop.
Integrasi Agentic AI & Tools (C4)	Membangun AI Agent dengan Function Calling terstruktur, validasi JSON skema Zod, dan streaming SSE.	Mengintegrasikan API LLM namun respons hanya teks mentah tanpa function tools.	Mengabaikan proteksi prompt injection.
Arsitektur Background	Komputasi analitik berat	Komputasi dijalankan di	Komputasi berat memblokir

Worker (C5)	didelegasikan ke BullMQ worker terpisah dari thread utama web server.	thread utama Express namun menggunakan timer.	request HTTP pengguna lain.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 11: Poin Kunci Pembelajaran:

1. Integrasi algoritma Machine Learning memperkaya aplikasi web dari sekadar penyimpan data menjadi sistem perekomendasi cerdas.

2. Agentic AI Workflows memungkinkan LLM mengambil aksi nyata terhadap database web melalui Function Calling.

3. Komputasi analitik berat wajib didelegasikan ke background worker agar tidak memblokir Event Loop server.

7.1 Glosarium Istilah Kunci

• Agentic Workflow: Pola perancangan AI di mana model melakukan penalaran mandiri dan memanggil fungsi eksternal untuk menyelesaikan tugas.

• Euclidean Distance: Ukuran jarak garis lurus antara dua titik dalam ruang berdimensi n.

• Function Calling: Kemampuan model AI untuk mengembalikan argumen terstruktur guna mengeksekusi fungsi di kode aplikasi.

• K-Means: Algoritma unsupervised clustering yang mempartisi N observasi ke dalam K kelompok.

• Server-Sent Events (SSE): Standar web yang memungkinkan server mengirimkan aliran data satu arah secara real-time ke peramban.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan pentingnya tahapan Normalisasi Data (Min-Max Scaling) sebelum mengeksekusi K-Means Clustering pada data transaksi web yang memiliki rentang nilai sangat lebar!

2. [Evaluasi - C5] Evaluasi kelebihan dan kekurangan arsitektur Server-Sent Events (SSE) dibanding polling reguler untuk menampilkan output streaming token dari AI Generatif!

3. [Kreasi - C6] Rancang skema Function Calling OpenAPI untuk AI Agent yang dapat memeriksa status pelacakan resi kurir pengiriman secara dinamis melalui database internal!

BAB 12
INTEGRASI KOMPREHENSIF FULL-STACK & MANAJEMEN STATE APLIKASI
(FULL-STACK ARCHITECTURE, INTERCEPTORS, & OFFLINE-FIRST)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini menyatukan seluruh komponen yang telah dipelajari menjadi satu arsitektur Full-Stack end-to-end yang solid: integrasi Frontend modern (Next.js/React/Vue/Vite) dengan Backend API terdistribusi, manajemen State Global (Server State vs Client State), sinkronisasi token otomatis via Axios/Fetch Interceptors dengan antrean request saat status 401 Unauthorized terjadi, serta arsitektur Offline-First (IndexedDB & Service Workers).

Tabel 12.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 12.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL08: Mampu mendesain aplikasi web multi-platform yang responsif dan terintegrasi front-end/back-end.
Capaian Pembelajaran MK (CPMK)	CPMK082: Mampu mengintegrasikan arsitektur full-stack secara terpadu, aman, dan berkinerja tinggi.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK082.2: Mampu merancang sinkronisasi state global, HTTP interceptors, dan offline storage (Taksonomi Bloom: C3 - Penerapan & C4 - Analisis).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu memisahkan dengan tegas antara Server State (TanStack Query / SWR) dan Client/UI State (Zustand / Redux) pada arsitektur frontend modern.

2. Mampu membangun mekanisme Silent Token Refresh menggunakan HTTP Interceptors dengan antrean request saat status 401 Unauthorized terjadi tanpa me-refresh halaman.

3. Mampu menerapkan konsep Optimistic UI Updates dan sinkronisasi Offline-First menggunakan IndexedDB dan Background Sync API.

2. Pengantar & Konsep Teoretis: Paradigma State Management Modern

2.1 Server State vs Client State

Klasifikasi State: Kesalahan umum pengembang frontend adalah memasukkan semua data remote API ke dalam global state tanpa memperhatikan pola integrasi tim dan arsitektur server-side modern [45], [46], [47], [48] (Redux/Vuex). Sejatinya:

• Server State: Data yang dimiliki oleh server remote (daftar produk, profil user). Membutuhkan asynchronous fetching, caching, deduplikasi, invalidation, dan background refetching (paling optimal dikelola oleh TanStack Query).

• Client State: Data murni antarmuka pengguna lokal yang sinkron (status modal popup terbuka, filter dropdown, preferensi tema sidebar).

2.2 Rekonsiliasi Data Offline & Conflict Resolution Strategies

Ketika aplikasi web berjalan dalam mode Offline-First, pengguna dapat memodifikasi data lokal pada IndexedDB saat tidak ada koneksi internet. Saat koneksi online kembali, proses sinkronisasi harus menerapkan strategi penyelesaian konflik (Conflict Resolution):

1. Last-Write-Wins (LWW): Perubahan dengan timestamp terbaru menimpa perubahan sebelumnya (sederhana namun berisiko kehilangan data modifikasi pengguna lain).

2. Server-Authoritative Merge: Server bertindak sebagai sumber kebenaran mutlak dan menolak perubahan yang menyebabkan konflik versi.

3. Three-Way Merge / Operational Transformation: Menggabungkan modifikasi secara cerdas pada level atribut data.

2.4 Optimasi Kinerja Rendering: SSR, SSG, ISR, & React Server Components

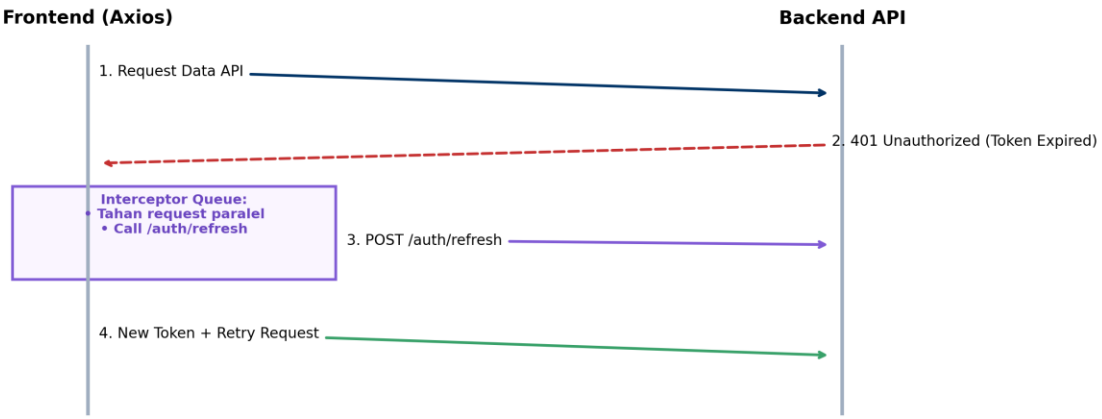
Arsitektur Full-Stack modern menawarkan spektrum metode rendering yang fleksibel sesuai kebutuhan konten:

1. Static Site Generation (SSG): Menghasilkan file HTML statis saat proses build untuk halaman yang jarang berubah (seperti dokumentasi dan blog), menyajikan performa instan dari Edge CDN.

2. Incremental Static Regeneration (ISR): Memungkinkan pembaruan halaman statis secara berkala di latar belakang tanpa perlu melakukan rebuild ulang seluruh aplikasi.

3. Server-Side Rendering (SSR) & Streaming SSR: Merender HTML secara dinamis per-request dengan teknik streaming pipa HTML bertahap, memangkas Time to First Byte (TTFB) secara drastis.

4. React Server Components (RSC): Mengeksekusi komponen antarmuka murni di sisi server tanpa menyertakan paket kode JavaScript komponen tersebut ke dalam bundle peramban klien.



Gambar 12.1: *Mekanisme Siklus Hidup HTTP Interceptor: Penahanan Antrean Request Paralel saat 401 dan Silent Token Refresh.*

3. Deep Dive Silent Token Refresh Interceptor (Expert Level)

Ketika Access Token kedaluwarsa, aplikasi frontend tidak boleh langsung me-redirect pengguna ke halaman login. Interceptor harus secara transparan menahan request yang gagal, memanggil endpoint /refresh-token, memperbarui header Authorization, dan mengeksekusi ulang seluruh request yang tertunda secara mulus (*seamless user experience*).

> **DEEP DIVE: RESILIENSI ARSITEKTUR FRONTEND ENTERPRISE**
>
> 1. Race Condition pada Multi-Request 401: Jika ada 5 request paralel yang mengalami 401 bersamaan, jangan panggil endpoint refresh 5 kali! Gunakan flag 'isRefreshing' dan antrean callback Promise.
>
> 2. Optimistic UI Updates: Perbarui tampilan antarmuka seketika sebelum server merespons (misal: tombol like langsung berubah merah), lalu rollback tampilan jika request backend akhirnya gagal.
>
> 3. Hydration Error pada SSR: Pastikan render awal server dan browser menghasilkan HTML tree yang persis sama untuk menghindari hydration mismatch di Next.js.

3.1 Pola Manajemen State Lanjut: Server State Synchronization & Mutation Lifecycles

TanStack Query merevolusi komunikasi data frontend melalui siklus hidup mutasi yang terstruktur (onMutate, onError, onSettled).

Pada saat operasi mutasi (seperti menambahkan item ke keranjang belanja), onMutate membatalkan kueri aktif yang berjalan, menyimpan snapshot state lama untuk kebutuhan rollback darurat jika jaringan gagal, dan memperbarui cache UI secara instan. Jika respons server sukses, onSettled melakukan invalidasi kunci kueri terkait untuk memastikan konsistensi data dengan server remote.

Pendekatan ini memberikan pengalaman pengguna yang sangat responsif, setara dengan performa aplikasi desktop natif.

3.2 Optimasi Akselerasi Web: Edge Middleware & Geografis Routing

Edge Middleware mengeksekusi logika ringan (seperti otentikasi JWT, deteksi lokasi geografis, dan eksperimen A/B testing) di server edge CDN terdekat dari pengguna:

1. Personalisasi Sub-5ms: Memodifikasi respons HTTP di lokasi Edge PoP tanpa harus meneruskan request ke server origin di pusat data utama.

2. Edge Rate Limiting: Memblokir bot berbahaya di lapisan perimeter terluar sebelum membebani server backend utama.

4. Penerapan Praktis & Hands-on Code (Axios Token Refresh Interceptor)

Berikut implementasi Axios Interceptor yang mengelola antrean konkurensi saat token kedaluwarsa secara otomatis:

Listing Program: Enterprise Axios Silent Token Refresh Interceptor

```
// src/lib/api-client.ts
import axios, { AxiosError, InternalAxiosRequestConfig } from 'axios';

export const apiClient = axios.create({
  baseURL: process.env.NEXT_PUBLIC_API_URL || '/api/v1',
  withCredentials: true // Wajib untuk mengirim httpOnly cookie
});

let isRefreshing = false;
let failedQueue: Array<{ resolve: (token: string) => void; reject: (err: any) => void }> =
[];

const processQueue = (error: any, token: string | null = null) => {
  failedQueue.forEach(prom => {
    if (error) prom.reject(error);
    else prom.resolve(token!);
  });
  failedQueue = [];
};

apiClient.interceptors.response.use(
  (response) => response,
  async (error: AxiosError) => {
    const originalRequest = error.config as InternalAxiosRequestConfig & { _retry?: boolean
};

    // Jika terjadi 401 dan bukan request refresh token itu sendiri
    if (error.response?.status === 401 && !originalRequest._retry) {
      if (isRefreshing) {
        // Antrekan request jika proses refresh sedang berjalan
        return new Promise((resolve, reject) => {
          failedQueue.push({ resolve, reject });
        }).then((token) => {
          originalRequest.headers['Authorization'] = `Bearer ${token}`;
          return apiClient(originalRequest);
        });
      }
```

```
      originalRequest._retry = true;
      isRefreshing = true;

      try {
        // Panggil endpoint silent refresh
        const { data } = await axios.post('/api/v1/auth/refresh', {}, { withCredentials:
true });
        const newAccessToken = data.accessToken;

        apiClient.defaults.headers.common['Authorization'] = `Bearer ${newAccessToken}`;
        processQueue(null, newAccessToken);

        originalRequest.headers['Authorization'] = `Bearer ${newAccessToken}`;
        return apiClient(originalRequest);
      } catch (refreshErr) {
        processQueue(refreshErr, null);
        window.location.href = '/login?session_expired=true';
        return Promise.reject(refreshErr);
      } finally {
        isRefreshing = false;
      }
    }
    return Promise.reject(error);
  }
);
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Aplikasi kasir Point-of-Sale (POS) restoran 'NuurResto' sering mengalami masalah saat koneksi internet terputus: kasir tidak bisa menginput pesanan dan aplikasi mengalami crash layar putih. Selain itu, saat sesi token kedaluwarsa, seluruh input pesanan kasir yang belum disimpan hilang seketika karena aplikasi me-redirect paksa ke halaman login. Mahasiswa diminta merekayasa arsitektur Full-Stack yang mendukung Offline-First dengan penyimpanan antrean transaksi di IndexedDB serta mengimplementasikan Silent Token Refresh tanpa kehilangan status form pengisian kasir.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Rancang modul IndexedDB lokal untuk menyimpan antrean transaksi saat offline.

2. Implementasikan Axios Interceptor yang menahan dan mengeksekusi ulang request tertunda saat token 401 terjadi.

3. Terapkan strategi rekonsiliasi data otomatis saat koneksi internet kembali pulih.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK082.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 12.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK111.1 (Full-Stack Integration & Offline-First).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Integrasi End-to-End (C3/P4)	Integrasi front-end dan backend mulus, type-safe (tRPC/OpenAPI types), dan penanganan status error 4xx/5xx konsisten.	Integrasi berjalan namun tipe data di front-end masih manual tanpa sinkronisasi skema backend.	Banyak runtime type mismatch dan CORS error.
Mekanisme Silent Refresh (C4)	Berhasil mengelola antrean konkurensi request saat refresh token dan bebas dari infinite loop redirect.	Refresh token berjalan namun request paralel yang gagal tidak diantrekan secara benar.	Aplikasi melempar user ke form login setiap kali token habis.
Arsitektur Offline & Optimistic UI (C5)	Menerapkan Optimistic Update dengan rollback tangguh dan sinkronisasi otomatis saat online kembali.	Menerapkan caching data lokal sederhana tanpa antrean sinkronisasi.	Aplikasi lumpuh total saat offline.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 12: Poin Kunci Pembelajaran:

1. Memisahkan Server State dan Client State adalah fondasi arsitektur frontend modern yang bersih dan terukur.

2. Silent Refresh Interceptor menjamin keamanan token pendek tanpa mengorbankan kenyamanan pengguna.

3. Arsitektur Offline-First menjamin integritas data transaksi operasional di lingkungan dengan konektivitas tidak stabil.

7.1 Glosarium Istilah Kunci

• HTTP Interceptor: Fungsi perantara yang mencegat permintaan atau respons HTTP sebelum diproses oleh aplikasi.

• IndexedDB: Basis data NoSQL transaksional di sisi peramban untuk menyimpan data objek terstruktur dalam jumlah besar.

• Optimistic UI: Pola pembaruan tampilan antarmuka seketika sebelum menerima konfirmasi sukses dari server.

• Server State: Status data yang bersumber dari server remote dan memerlukan sinkronisasi asinkron.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Mengapa mengelola data server menggunakan Redux global store konvensional membutuhkan banyak boilerplate kode dibanding menggunakan library Server State seperti TanStack Query?

2. [Evaluasi - C5] Evaluasi risiko keamanan dan konsistensi data pada implementasi Optimistic UI Update untuk transaksi finansial pemindahan saldo perbankan!

3. [Kreasi - C6] Rancanglah alur sinkronisasi data offline ke online (IndexedDB ke PostgreSQL) yang mampu menyelesaikan konflik data (Conflict Resolution Strategy: Server Wins vs Client Wins)!

BAB 13
PENJAMINAN MUTU & PENGUJIAN PERANGKAT LUNAK WEB (UNIT TESTING, INTEGRATION TESTING, MOCKING, & CODE COVERAGE)

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab ini membahas metodologi Penjaminan Mutu Perangkat Lunak (Software Quality Assurance / SQA) pada aplikasi web modern: hierarki Piramida Pengujian (Test Pyramid), Unit Testing modular berbasis Vitest/Jest, Integration & API Testing berbasis Supertest, teknik Mocking & Spying dependensi eksternal, mitigasi Code Smells & Flaky Tests, serta analisis metrik Code Coverage terperinci.

Tabel 13.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 13.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (Software Quality Assurance Practitioner).
Capaian Pembelajaran Lulusan (CPL)	CPL04: Mampu mengevaluasi dan memvalidasi keandalan serta kualitas arsitektur sistem perangkat lunak.
Capaian Pembelajaran MK (CPMK)	CPMK043: Mampu merancang skenario pengujian otomatis berlapis untuk menjamin keandalan sistem web.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK043.2: Mampu membuat unit test, integration test, dan mengukur code coverage (Taksonomi Bloom: C4 - Analisis & C5 - Evaluasi).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu merancang strategi pengujian berdasarkan proporsi ideal Test Pyramid (70% Unit Test, 20% Integration Test, 10% End-to-End Test).

2. Mampu membuat unit test modular menggunakan teknik Mocking Repository dan Spying Service eksternal dengan pola AAA (Arrange-Act-Assert).

3. Mampu melakukan pengujian integrasi endpoint API (Supertest) dengan database uji terisolasi dan mencapai target Code Coverage minimal 85%.

2. Pengantar & Konsep Teoretis: Hierarki Piramida Pengujian

2.1 Anatomi Test Pyramid

Tiga Lapisan Uji: Piramida pengujian membagi pengujian perangkat lunak menjadi 3 lapisan teruji untuk menjamin keandalan kualitas perangkat lunak modern [49], [50], [51]:

• Unit Testing (Dasar Piramida): Menguji fungsi atau unit kode terkecil secara terisolasi. Sangat cepat (milidetik), murah, dan dieksekusi dalam jumlah ribuan.

• Integration Testing (Lapisan Tengah): Menguji interaksi gabungan antar-modul (misal: Service berinteraksi dengan Database atau Redis nyata).

• End-to-End / E2E Testing (Puncak Piramida): Menguji alur pengguna nyata secara penuh melalui peramban otomatis (Playwright / Cypress). Lambat dan mahal perawatannya.

2.2 Metrik Kualitas Pengujian: Mutation Testing & Cyclomatic Complexity

Mencapai angka Code Coverage 100% belum tentu menjamin rangkaian test suite berkualitas tinggi jika tidak ada assertion yang bermakna. Dua metrik canggih digunakan di industri:

• Cyclomatic Complexity: Metrik yang mengukur jumlah jalur eksekusi independen dalam suatu fungsi. Fungsi dengan Cyclomatic Complexity > 10 wajib dipecah menjadi fungsi-fungsi yang lebih kecil.

• Mutation Testing: Pustaka pengujian (seperti Stryker) menyuntikkan bug buatan (mutant) ke dalam kode sumber; test suite dianggap kuat jika mampu mendeteksi dan menggagalkan seluruh mutan tersebut (Mutation Score > 85%).

2.4 Metodologi Test-Driven Development (TDD) & Behavior-Driven Development (BDD)

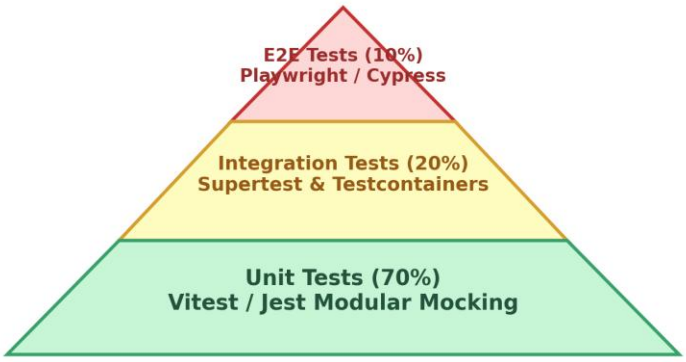
Penerapan siklus TDD (Red-Green-Refactor) membentuk disiplin rekayasa perangkat lunak yang unggul:

1. Fase Red: Pengembang menulis test case yang merepresentasikan kebutuhan fitur baru dan memastikan pengujian tersebut gagal terlebih dahulu.

2. Fase Green: Menulis kode implementasi seminimal mungkin yang penting berhasil membuat test case lolos (berwarna hijau).

3. Fase Refactor: Merapikan struktur kode, mengeliminasi duplikasi, dan meningkatkan keterbacaan tanpa khawatir merusak fungsionalitas yang telah dijamin oleh test case.

Pada Behavior-Driven Development (BDD), spesifikasi pengujian ditulis dalam bahasa manusia terstruktur (format Given-When-Then via Cucumber/Gherkin), menjembatani komunikasi antara product owner, QA tester, dan tim pengembang.



Gambar 13.1: *Hierarki Piramida Pengujian Perangkat Lunak: Unit Testing (70%), Integration Testing (20%), dan E2E Testing (10%).*

3. Deep Dive Mocking, Stubs, & Anti-Flaky Tests (Expert Level)

Flaky Tests (pengujian yang kadang lolos dan kadang gagal tanpa ada perubahan kode) adalah musuh utama pipeline CI/CD. Flaky test biasanya dipicu oleh ketergantungan waktu (setTimeout acak), state global yang bocor antar-test case, atau koneksi jaringan eksternal yang tidak di-mock.

> **DEEP DIVE: REKAYASA PENGUJIAN OTOMATIS BERKELAS ENTERPRISE**
>
> 1. Mocking vs Spying vs Stubbing: Stub memberikan respons tiruan tetap; Mock memverifikasi apakah suatu fungsi dipanggil dengan parameter yang benar; Spy mengamati eksekusi fungsi asli tanpa mengubah perilakunya.
>
> 2. AAA Pattern (Arrange-Act-Assert): Strukturkan setiap test case dengan 3 blok terpisah: siapkan data (Arrange), eksekusi fungsi (Act), dan verifikasi hasil (Assert).
>
> 3. Test Database Isolation: Gunakan Testcontainers (Docker instance otomatis) atau SQLite In-Memory untuk memastikan setiap suite integration test berjalan pada basis data yang bersih dan terisolasi.

3.1 Rekayasa Pengujian Integrasi Database: Testcontainers & Database Fixtures

Pengujian integrasi yang mengandalkan database bersama di lingkungan staging sering kali memicu flaky test akibat data uji yang saling tumpang tindih. Standar modern industri menggunakan pustaka Testcontainers untuk menginisiasi kontainer Docker PostgreSQL / Redis yang segar dan terisolasi secara otomatis pada saat test suite dijalankan, lalu menghancurkannya saat pengujian selesai.

Setiap file pengujian menerapkan pola Database Fixtures dan Database Seeding terkontrol, serta mengeksekusi migrasi skema sebelum suite uji berjalan, menjamin hasil pengujian yang 100% deterministik dan bebas dari efek samping data kotor.

3.2 Pengujian Otomatis End-to-End (E2E) dengan Playwright & Visual Regression

Di puncak piramida pengujian, Playwright mengotomatisasi pengujian alur kritis pengguna nyata melintasi multi-peramban (Chromium, Firefox, WebKit):

1. Headless Testing: Mengeksekusi skenario checkout belanja dan verifikasi pembayaran secara otomatis pada pipeline CI/CD.

2. Visual Regression Testing: Membandingkan tangkapan layar piksel antarmuka saat ini terhadap snapshot referensi emas (golden baseline) guna mendeteksi pergeseran layout atau perubahan warna yang tidak disengaja.

3. Network Mocking & HAR Replay: Mensimulasikan kondisi jaringan buruk (3G throttling) dan respons server galat 500 secara deterministik.

3.3 Pengujian Beban & Kinerja API dengan k6 & Autocannon

Selain pengujian fungsional, aplikasi web enterprise wajib menjalani uji beban (Load Testing) dan uji ketahanan (Stress Testing):

1. Virtual Users (VUs) Simulation: Menggunakan perkakas grafis k6 untuk mensimulasikan 5.000 pengguna konkuren yang mengakses endpoint API secara serentak.

2. Analisis Metrik Latensi P95 dan P99: Memastikan bahwa 99% permintaan pengguna terlayani di bawah ambang batas 250 milidetik pada saat beban puncak.

3. Deteksi Memory Leak under Load: Mengamati profil heap RAM menggunakan Autocannon selama pengujian ketahanan 1 jam nonstop.

4. Penerapan Praktis & Hands-on Code (Vitest Unit & Supertest)

Berikut implementasi suite unit testing modular menggunakan Vitest:

Listing Program: Unit Testing Modular dengan Vitest & Mocking Dependency

```
// tests/unit/discount-calculator.service.test.ts
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { DiscountService } from '../../src/services/discount.service';
import { IUserRepository } from '../../src/interfaces/user-repository.interface';

describe('DiscountService - Unit Tests', () => {
  let discountService: DiscountService;
  let mockUserRepo: IUserRepository;

  beforeEach(() => {
    // 1. Arrange: Buat Mock Repository
    mockUserRepo = {
      findById: vi.fn(),
      findByEmail: vi.fn(),
      create: vi.fn(),
      updateStatus: vi.fn()
    };
    discountService = new DiscountService(mockUserRepo);
  });
```

```
  it('harus memberikan diskon 20% bagi pengguna berstatus VIP', async () => {
    // Arrange: Simulasikan return value mock
    vi.mocked(mockUserRepo.findById).mockResolvedValue({
      id: 'usr-123',
      email: 'vip@nuur.ac.id',
      fullName: 'Andri Triyono',
      role: 'STUDENT',
      isActive: true,
      createdAt: new Date()
    });

    // 2. Act: Eksekusi metode
    const finalPrice = await discountService.calculateFinalPrice('usr-123', 100000);

    // 3. Assert: Verifikasi hasil dan pemanggilan mock
    expect(finalPrice).toBe(80000);
    expect(mockUserRepo.findById).toHaveBeenCalledWith('usr-123');
    expect(mockUserRepo.findById).toHaveBeenCalledTimes(1);
  });

  it('harus melempar error jika pengguna tidak ditemukan dalam database', async () => {
    vi.mocked(mockUserRepo.findById).mockResolvedValue(null);

    await expect(discountService.calculateFinalPrice('invalid-id', 100000))
      .rejects
      .toThrow("Pengguna tidak ditemukan dalam sistem akademik.");
  });
});
```

5. Studi Kasus & Problem-Based Learning (PBL)

Studi Kasus PBL: Skenario Industri & Technopreneurship:

Sebuah sistem penerimaan mahasiswa baru 'PMB-Nuur' mengalami kegagalan fatal saat deploy versi baru ke server produksi: modul kalkulasi potongan beasiswa ternyata memiliki bug pembagian nol (division by zero) yang menyebabkan sistem crash saat pendaftar tidak memiliki nilai rapor semester 1. Tidak ada satupun unit test di repositori proyek. Mahasiswa diminta merancang suite unit testing lengkap dengan target Code Coverage minimal 90% (Statement, Branch, Function, Line Coverage) dan menambahkan integration test endpoint API menggunakan Supertest.

Instruksi Pemecahan Masalah: Tugas Rekayasa Mahasiswa:

1. Rancang suite Unit Testing untuk seluruh kasus batas logika perhitungan beasiswa.

2. Implementasikan mock database repository sehingga unit test berjalan dalam waktu < 50 milidetik.

3. Jalankan analisis code coverage dan buktikan seluruh percabangan if-else teruji 100%.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE

Ketercapaian Sub-CPMK043.2 pada bab ini dievaluasi melalui rubrik analitik berikut:

Tabel 13.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK112.1 (Software Quality Assurance & Testing).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Desain Test Case & AAA Pattern (C4)	Test cases komprehensif (Happy path, Edge cases, Error boundaries) dengan pola AAA yang bersih.	Test case mencakup skenario umum namun mengabaikan kondisi batas (edge cases).	Test case tidak terstruktur dan sulit dibaca.
Teknik Mocking & Isolasi (C3/P4)	Menggunakan Mocking/Spying secara presisi tanpa memicu efek samping pada lingkungan nyata.	Masih melakukan koneksi langsung ke server eksternal/database saat unit test.	Tidak memahami konsep mocking.
Metrik Code Coverage (C5)	Mencapai Code Coverage > 90% pada branch, function, dan lines dengan laporan LCOV rapi.	Code coverage mencapai 70-84%.	Code coverage < 50% atau tidak ada pengujian sama sekali.

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 13: Poin Kunci Pembelajaran:

1. Pengujian otomatis adalah jaring pengaman utama yang memungkinkan tim melakukan refactoring kode tanpa rasa takut.

2. Piramida pengujian yang sehat didominasi oleh unit test yang cepat dan terisolasi.

3. Mocking memungkinkan pengujian logika murni tanpa ketergantungan infrastruktur eksternal.

7.1 Glosarium Istilah Kunci

• Branch Coverage: Persentase jalur percabangan logika (if-else, switch) yang dieksekusi oleh rangkaian pengujian.

• Flaky Test: Pengujian yang menunjukkan hasil tidak deterministik (kadang lolos, kadang gagal) tanpa perubahan kode.

• Mock: Objek tiruan yang meniru perilaku objek nyata dan memverifikasi interaksi pemanggilan.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Mengapa mencapai 100% Line Coverage belum tentu menjamin perangkat lunak bebas dari bug jika Branch Coverage-nya rendah?

2. [Evaluasi - C5] Evaluasi dampak buruk fenomena 'Ice Cream Cone Anti-Pattern' (terlalu banyak E2E test dibanding Unit test) terhadap kecepatan rilis tim rekayasa perangkat lunak!

3. [Kreasi - C6] Buatlah integration test suite menggunakan Supertest untuk menguji endpoint POST /api/v1/auth/login, mencakup skenario password salah, rate limiting 5x percobaan, dan login sukses mengembalikan cookie httpOnly!

BAB 14
CONTAINERIZATION, DEPLOYMENT AWAN, CI/CD, & HILIRISASI PROYEK TECHNOPRENEUR

1. Identitas Pembelajaran & Keselarasan OBE (Constructive Alignment)

Bab penutup ini merupakan kulminasi dari seluruh pembelajaran Pemrograman Web Lanjut: pengemasan aplikasi ke dalam kontainer ringan (Docker Multi-Stage Build), orkestrasi multi-layanan (Docker Compose), otomatisasi alur Continuous Integration & Continuous Deployment (CI/CD via GitHub Actions), deployment awan aman, pemantauan sistem (Health Check & Observability), serta strategi hilirisasi proyek berbasis Technopreneurship dan konversi ke artikel publikasi ilmiah.

Tabel 14.1: *Matriks Keselarasan Konstruktif OBE Capaian Pembelajaran Bab 14.*

Komponen OBE	Deskripsi Keselarasan
Profil Lulusan (PL)	PL01 (Software Developer), PL03 (QA Practitioner), PL04 (Software Consultant).
Capaian Pembelajaran Lulusan (CPL)	CPL08: Mampu menguasai konsep deployment aplikasi web multi-platform yang terintegrasi dan berkinerja tinggi.
Capaian Pembelajaran MK (CPMK)	CPMK082: Mampu merancang pipeline DevOps, containerization Docker, dan mempublikasikan produk digital siap pakai.
Sub-CPMK & Tingkat Taksonomi	Sub-CPMK082.1 & 082.2: Mampu membangun Dockerfile multi-stage, pipeline CI/CD, dan merilis produk technopreneur (Taksonomi Bloom: C5 - Evaluasi & C6 - Penciptaan).

Indikator Pembelajaran: Indikator Keberhasilan Pembelajaran:

1. Mampu merancang Dockerfile efisien berbasis Multi-Stage Build dengan ukuran image minimal (<90MB) dan non-root security context.

2. Mampu membangun pipeline otomatisasi GitHub Actions CI/CD untuk pengujian, linting, build image, dan auto-deploy ke Cloud VPS/PaaS.

3. Mampu menyusun model hilirisasi proyek perangkat lunak bernilai ekonomi (Technopreneurship) dan mengonversi laporan teknis menjadi naskah publikasi ilmiah.

2. Pengantar & Konsep Teoretis: Containerization & DevOps Culture

2.1 Mengapa Docker? Paradigma 'Works on My Machine' Solution

Solusi Standarisasi: Sebelum era kontainer, perbedaan versi Node.js, pustaka sistem operasi C++, dan variabel lingkungan sering menyebabkan bug fatal saat aplikasi dipindahkan dari laptop pengembang ke server produksi. Docker membungkus aplikasi beserta seluruh dependensinya ke dalam satu unit kontainer terisolasi yang dijamin berjalan identik di lingkungan manapun.

2.2 Docker Multi-Stage Build

Efisiensi Image: Dalam aplikasi TypeScript/Node.js, perkakas kompilasi (TypeScript compiler, webpack/esbuild, devDependencies) hanya dibutuhkan saat proses build. Multi-Stage Build membagi Dockerfile menjadi tahap 'Builder' (berisi devDependencies lengkap) dan tahap 'Runner' (hanya berisi file hasil kompilasi JS dan runtime ringan Alpine Linux), memangkas ukuran image dari 1.2 GB menjadi kurang dari 90 MB.

2.3 Tiga Pilar Observabilitas Sistem Web Modern (Metrics, Logs, Traces)

Deployment aplikasi ke lingkungan komputasi awan menuntut visibilitas penuh terhadap kesehatan sistem produksi:

1. Metrics (Kuantitatif): Data deret waktu numerik seperti CPU utilization, memory heap used, RPS (Request per Second), dan error rate 5xx (dimonitor via Prometheus & Grafana).

2. Logs (Konteks Kejadian): Catatan tekstual terstruktur dalam format JSON yang memuat correlation ID untuk setiap transaksi error (dikelola via ELK Stack atau Grafana Loki).

3. Distributed Tracing (Perjalanan Request): Pelacakan alur hidup sebuah request HTTP saat melintasi puluhan kontainer mikroservis (dianalisis via OpenTelemetry dan Jaeger).

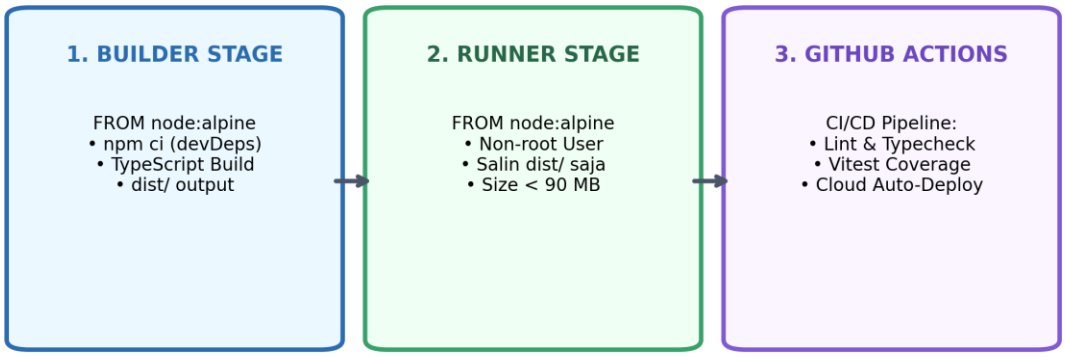
2.4 Strategi Monetisasi Produk Digital & Konversi Riset Technopreneur

Sebuah karya rekayasa perangkat lunak bernilai maksimal jika mampu dihilirisasi menjadi produk digital yang berkelanjutan (Technopreneurship):

1. Model Bisnis SaaS (Freemium & Tiered Subscription): Merancang arsitektur multi-tenant dengan isolasi data skema atau database per tenant, serta mengintegrasikan Payment Gateway (Midtrans/Xendit/Stripe) dengan webhook verifikasi otomatis.

2. Metrik Bisnis Technopreneur: Memantau Monthly Recurring Revenue (MRR), Customer Acquisition Cost (CAC), Churn Rate, dan Customer Lifetime Value (LTV) melalui dashboard analitik terpadu.

3. Diseminasi Publikasi Ilmiah: Mengonversi data empiris hasil pengujian performa arsitektur, efisiensi latensi komputasi, dan kepuasan pengguna menjadi artikel ilmiah berstandar internasional.



Gambar 14.1: *Alur Otomatisasi DevOps: Docker Multi-Stage Build, GitHub Actions CI/CD Pipeline, dan Cloud Auto-Deployment.*

3. Deep Dive CI/CD Pipeline & Zero-Downtime Deployment (Expert Level)

Continuous Integration (CI) memvalidasi setiap commit melalui pengujian otomatis guna mencapai keandalan rilis frekuensi tinggi pada infrastruktur awan [52], [53], [54], [55], sedangkan Continuous Deployment (CD) merilis perubahan yang telah lolos uji ke lingkungan produksi secara otomatis tanpa intervensi manual.

> **DEEP DIVE: STANDAR KEAMANAN & DEPLOYMENT AWAN PRODUKSI**
>
> 1. Non-Root Container Security: Jangan pernah menjalankan proses Node.js di dalam kontainer sebagai user 'root'! Selalu buat user non-privileged (USER node) untuk mencegah eskalasi hak akses saat terjadi pembobolan kontainer.
>
> 2. Blue-Green / Rolling Deployment: Jangan pernah menghentikan server lama sebelum kontainer baru siap melayani trafik. Gunakan Healthcheck probe (liveness & readiness) pada Docker/Kubernetes.
>
> 3. Secrets Management: Jangan pernah men-commit file .env atau kunci API ke git. Gunakan GitHub Secrets dan injeksikan sebagai environment variable saat pipeline dieksekusi.

3.1 Strategi Deployment Lanjut: Blue-Green, Canary Releases, & GitOps

Dalam lingkungan produksi skala besar dengan jutaan pengguna aktif, merilis pembaruan perangkat lunak tidak boleh menyebabkan satu milidetik pun waktu henti layanan (zero-downtime deployment):

1. Blue-Green Deployment: Menyediakan dua lingkungan identik (Blue yang sedang aktif melayani lalu lintas, dan Green yang memuat versi baru). Setelah pengujian kesehatan di lingkungan Green sukses 100%, router Load Balancer seketika mengalihkan seluruh lalu lintas dari Blue ke Green.

2. Canary Releases: Mengarahkan hanya 5% lalu lintas pengguna acak ke versi baru untuk memantau error rate dan metrik latensi selama 30 menit sebelum meluncurkannya ke 100% pengguna.

3. GitOps (ArgoCD): Praktik di mana seluruh status infrastruktur dan konfigurasi Kubernetes dideklarasikan di repositori Git sebagai satu-satunya sumber kebenaran.

3.2 Infrastruktur sebagai Kode (IaC) & Manajemen Konfigurasi Cloud

Mengelola server secara manual via antarmuka web cloud adalah anti-pattern yang rentan kesalahan manusia. Infrastructure as Code (IaC) mendefinisikan seluruh sumber daya komputasi secara deklaratif:

1. Terraform / OpenTofu: Mendeklarasikan VM instance, VPC subnet, Load Balancer, dan S3 Bucket dalam berkas konfigurasi .tf yang dapat di-version control di Git.

2. Ansible: Mengotomatisasi provisioning server, instalasi runtime Docker, dan konfigurasi firewall sistem operasi secara terpusat dan idempoten.

4. Penerapan Praktis & Hands-on Code (Production Dockerfile & GitHub Actions)

Berikut implementasi Dockerfile Multi-Stage Build dan alur otomatisasi GitHub Actions CI/CD:

Listing Program: Production-Ready Multi-Stage Dockerfile

```
# Dockerfile (Multi-Stage Production Build)
# ------------------------------------------------------------
# Stage 1: Dependency & Build Stage
# ------------------------------------------------------------
FROM node:20-alpine AS builder
WORKDIR /app

# Salin manifest dependensi
COPY package*.json ./
RUN npm ci

# Salin source code dan jalankan kompilasi TypeScript
COPY . .
RUN npm run build
RUN npm prune --production

# ------------------------------------------------------------
# Stage 2: Minimal Production Runtime
# ------------------------------------------------------------
FROM node:20-alpine AS runner
WORKDIR /app

ENV NODE_ENV=production
ENV PORT=3000

# Buat non-root user demi keamanan kontainer
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
USER appuser

# Salin hanya artifact yang diperlukan dari builder stage
COPY --from=builder /app/package.json ./package.json
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/dist ./dist
```

```
EXPOSE 3000

# Healthcheck probe otomatis
HEALTHCHECK --interval=30s --timeout=5s --start-period=5s --retries=3 \
  CMD wget --no-verbose --tries=1 --spider http://localhost:3000/health || exit 1

CMD ["node", "dist/main.js"]
```

Listing Program: GitHub Actions CI/CD Pipeline Workflow

```
# .github/workflows/deploy.yml
name: CI/CD Production Pipeline

on:
  push:
    branches: [ main ]

jobs:
  build-test-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Setup Node.js Runtime
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'

      - name: Install Dependencies
        run: npm ci

      - name: Run Linter & Typecheck
        run: |
          npm run lint
          npx tsc --noEmit

      - name: Execute Automated Unit & Integration Tests
        run: npm test -- --coverage

      - name: Build & Push Docker Image
        if: success()
        run: |
          echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin
          docker build -t nuur/web-app-production:latest .
          docker push nuur/web-app-production:latest
```

5. Studi Kasus, Proyek Technopreneur, & Publikasi Ilmiah

Tantangan Proyek Capstone: Sebagai tugas puncak (Capstone Project), mahasiswa diminta membangun sebuah produk SaaS (Software-as-a-Service) bernilai komersial nyata berbasis Technopreneurship:

1. Hilirisasi Produk: Mengemas aplikasi web lengkap (Front-end, Backend API, Database, Redis Cache, AI Integration) ke dalam pipeline Docker Compose dan merilisnya ke internet publik (Cloud/VPS).

2. Konversi Publikasi Ilmiah: Mengambil salah satu komponen arsitektur unik dari proyek (misal: analisis latensi caching Redis vs Memory, atau evaluasi akurasi clustering K-Means) dan menyusunnya menjadi artikel ilmiah dengan format standar IMRAD (Introduction, Method, Results, and Discussion) siap submit ke jurnal/prosiding nasional terakreditasi.

6. Instrumen Asesmen & Rubrik Penilaian Capaian OBE (UAS / Proyek Akhir)

Tabel 14.2: *Rubrik Penilaian Portofolio Analitik Ketercapaian Sub-CPMK113.1 (DevOps CI/CD & Technopreneurship).*

Kriteria Penilaian	Sangat Baik (Skor 85-100)	Cukup / Memadai (Skor 70-84)	Kurang (Skor < 70)
Kualitas Docker & Kontainer (C5)	Dockerfile Multi-Stage sangat optimal (<100MB), non-root user, healthcheck teruji, Docker Compose terisolasi.	Docker berjalan namun image masih besar (>500MB) dan menggunakan root user.	Gagal membuat Dockerfile yang dapat di-build.
Automasi CI/CD Pipeline (C6)	Pipeline GitHub Actions lengkap (Lint, Typecheck, Test Coverage, Docker Push, Auto-Deploy) lolos 100%.	Pipeline CI berjalan untuk pengujian namun deployment masih dilakukan manual.	Tidak ada pipeline CI/CD otomatis.
Kesiapan Produk & Publikasi (P4)	Aplikasi live di URL publik, memiliki model bisnis jelas (Technopreneur), dan draft paper ilmiah lengkap.	Aplikasi live namun belum menyusun draft artikel ilmiah.	Aplikasi hanya berjalan di localhost (belum dideploy).

7. Rangkuman Materi & Glosarium Istilah

Rangkuman Bab 14: Poin Kunci Pembelajaran:

1. Docker Multi-Stage Build dan CI/CD mengotomatisasi siklus rilis perangkat lunak modern dari kode sumber hingga server produksi.

2. Integrasi kurikulum OBE mengarahkan mahasiswa tidak hanya menguasai teori koding, tetapi mampu menghasilkan produk technopreneur nyata dan publikasi ilmiah berdaya saing tinggi.

7.1 Glosarium Istilah Kunci

• CI/CD: Continuous Integration dan Continuous Deployment, praktik otomatisasi integrasi dan pengiriman kode.

• Multi-Stage Build: Pola Dockerfile yang menggunakan beberapa tahap intermediate untuk menghasilkan image final yang sangat ramping.

• Technopreneurship: Kewirausahaan berbasis inovasi teknologi dan rekayasa perangkat lunak bernilai ekonomi.

8. Evaluasi Kompetensi Mandiri (Format Bloom C3–C6)

1. [Analisis - C4] Jelaskan mengapa menjalankan proses Node.js di dalam kontainer Docker menggunakan akun 'root' sangat berisiko dari kacamata keamanan siber!

2. [Evaluasi - C5] Evaluasi kelebihan dan kelemahan strategi deployment Blue-Green Deployment dibandingkan Rolling Update pada arsitektur web berskala besar!

3. [Kreasi - C6] Rancang konfigurasi docker-compose.yml lengkap untuk lingkungan produksi yang mengorkestrasikan Web API, PostgreSQL dengan Persistent Volume, Redis Cache, dan Nginx Reverse Proxy bersertifikat SSL!

DAFTAR PUSTAKA (IEEE REFERENCE STYLE)

[1] A. Triyono and Santoso K.I., "Performance Evaluation of Agentic Workflow-Driven Trend-Aware Rule Mining for Dynamic Menu Bundling," SMARTICS Journal, vol. 10, no. 2, pp. 78-89, 2024. doi: 10.21070/smartics.v10i2.1580.

[2] A. Triyono and dkk., "Pengelompokan Permintaan Darah Berdasarkan Golongan dan Waktu di Kabupaten Grobogan dengan Algoritma K-Means," Jurnal TRANSFORMASI, vol. 19, no. 1, pp. 45-56, 2023. doi: 10.56357/transformasi.v19i1.120.

[3] D. Bajaj, U. Bharti, A. Goel, and Gupta S.C., "Partial Migration for Re-architecting a Cloud Native Monolithic Application into Microservices and FaaS," Communications in Computer and Information Science, 2020. doi: 10.1007/978-981-15-9671-1_9.

[4] R. Shrestha and B. Nisha, "Microservices vs Serverless Deployment in AWS: A Case Study with an Image Processing Application," in Proc. Proceedings - 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing, UCC 2022, 2022. doi: 10.1109/UCC56403.2022.00033.

[5] Saloran B.M. and Albarda, "Challenges and Mitigation Strategies in Migrating from Monolithic Systems to Microservices: a Systematic Literature Review," in Proc. 2025 15th International Conference on Information and Communication Technology and System: AI for the Now and Next: Delivering Solutions and Driving Vision, ICTS 2025, 2025. doi: 10.1109/ICTS67612.2025.11369715.

[6] H. Hassan, Abdel-Fattah M.A., and W. Mohamed, "Migrating from Monolithic to Microservice Architectures: A Systematic Literature Review," International Journal of Advanced Computer Science and Applications, 2024. doi: 10.14569/IJACSA.2024.0151013.

[7] N. Bosilia, G. Weinberger, and P. Haindl, "Assessing the Impact of Asynchronous Communication on Resilience and Robustness: A Comparative Study of Microservice and Monolithic Architectures," Lecture Notes in Computer Science, 2026. doi: 10.1007/978-3-032-04403-7_16.

[8] Q. Yang, Z. Cui, and S. Tao, "Research and Implementation of Front-end Component Virtualization Asynchronous Rendering Method Based on Cache Mechanism," in Proc. Proceedings of the IEEE International Conference on Computer and Communications, ICCC, 2024. doi: 10.1109/ICCC62609.2024.10942090.

[9] Attah B.I., Alhassan J.K., Oyefolahan I.O., and Bashir S.A., "Mobile Application Software Usability Evaluation: Issues, Methods and Future Research Directions," Communications in Computer and Information Science, 2021. doi: 10.1007/978-3-030-69143-1_43.

[10] L. Di and Ramapriyan H.K., "Standards-based data and information systems for earth observations – an introduction," Lecture Notes in Geoinformation and Cartography, 2010. doi: 10.1007/978-3-540-88264-0_1.

[11] S. Ahirrao and R. Ingle, "Dynamic workload-aware partitioning in oltp cloud data stores," Journal of Theoretical and Applied Information Technology, 2014.

[12] Abdurrahman A.M. and E. Husni, "A Secure Digital Image Marketplace: Microservices and OWASP API Security Using Spring Boot," in Proc. 11th International Conference on ICT for Smart Society: Integrating Data and Artificial Intelligence for a Resilient and Sustainable Future Living, ICISS 2024 - Proceeding, 2024. doi: 10.1109/ICISS62896.2024.10750956.

[13] A. Akbulut and Perros H.G., "Performance Analysis of Microservice Design Patterns," IEEE Internet Computing, 2019. doi: 10.1109/MIC.2019.2951094.

[14] Mullins C., "Responsive, mobile app, mobile first: Untangling the UX design web in practical experience," in Proc. SIGDOC 2015 - Proceedings of the 33rd Annual International Conference on the Design of Communication, 2015. doi: 10.1145/2775441.2775478.

[15] D. Sloan and S. Horton, "Usability, Universal Usability, and Design Patterns," Human - Computer Interaction Series, 2019. doi: 10.1007/978-1-4471-7440-0_24.

[16] K. Behl and G. Raj, "Architectural Pattern of Progressive Web and Background Synchronization," in Proc. Proceedings on 2018 International Conference on Advances in Computing and Communication Engineering, ICACCE 2018, 2018. doi: 10.1109/ICACCE.2018.8441701.

[17] H. Aljawawdeh, S. Abuezhayeh, A. Alnatsheh, E. Qaddoumi, and L. Maghrabi, "Navigating Serverless and Microservices: Concise Guide," Studies in Computational Intelligence, 2023. doi: 10.1007/978-3-031-43300-9_48.

[18] E. Aksu and J. Han, "Is Personalization the Future of User Interfaces? A Systematic Mapping and Review," in Proc. Proceedings of 2025 11th International HCI and UX Conference in Indonesia, CHIuXiD 2025, 2025. doi: 10.1109/CHIuXiD68326.2025.11323802.

[19] Divyanshi and Rana R.P.S., "The User's Journey: A Historical Perspective on UI/UX Evolution," in Proc. Proceedings of the 2024 3rd Edition of IEEE Delhi Section Flagship Conference, DELCON 2024, 2024. doi: 10.1109/DELCON64804.2024.10866504.

[20] Sonowal G., "User Interface and User Experience Foundational Principles for Interactive Design," User Interface and User Experience Foundational Principles for Interactive Design, 2026. doi: 10.1201/9781003715764.

[21] V. Berry, A. Castelltort, B. Lange, J. Teriihoania, C. Tibermacine, and C. Trubiani, "Is it Worth Migrating a Monolith to Microservices? An Experience Report on Performance, Availability and Energy Usage," in Proc. Proceedings of the IEEE International Conference on Web Services, ICWS, 2024. doi: 10.1109/ICWS62655.2024.00112.

[22] I. Oumoussa and R. Saidi, "Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review," IEEE Access, 2024. doi: 10.1109/ACCESS.2024.3365079.

[23] M. Gordesli and A. Varol, "Comparing Interservice Communications of Microservices for E-Commerce Industry," in Proc. 10th International Symposium on Digital Forensics and Security, ISDFS 2022, 2022. doi: 10.1109/ISDFS55398.2022.9800784.

[24] S. Bhatnagar and R. Mahant, "The art of decoding microservices: An in-depth exploration of modern software architecture," The Art of Decoding Microservices: An In-Depth Exploration of Modern Software Architecture, 2025. doi: 10.1007/979-8-8688-1267-5.

[25] Fan C.-F., A. Jindal, and M. Gerndt, "Microservices vs serverless: A performance comparison on a cloud-native web application," in Proc. CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science, 2020.

[26] R. Bernsteiner, M. Blasisker, M. Kohlegger, C. Ploder, and S. Schlögl, "Performance Comparison between a Monolithic and a Microservice Application," in Proc. CEUR Workshop Proceedings, 2025.

[27] A. Surya and R. Nathiya, "Enhancing Advanced Model Architecture in Serverless Architecture to Improve Security," in Proc. Proceedings of 8th International Conference on Trends in Electronics and Informatics, ICOEI 2025, 2025. doi: 10.1109/ICOEI65986.2025.11013761.

[28] N. Halili, I. Shabani, D. Hyseni, F. Zejnullahu, and Q. Kabashi, "Memory Consumption in Microservices Based and Monolithic Applications," Communications in Computer and Information Science, 2026. doi: 10.1007/978-3-032-24513-7_13.

[29] K. Sherin, V. Arvind, and S. Abel Jeba Sheeban, "Real-Time TCN-Transformer Based Predictive Caching for Low-Latency Delivery of Press Information Bureau Articles," in Proc. CCIC 2026 - Contemporary Computing Innovations Conference 2026, 2026. doi: 10.1109/CCIC68129.2026.11486098.

[30] M. Oloko-Oba, E. Esenogho, and K. Aruleba, "From Biosignals to Bedside: A Review of Real-Time Edge Machine Learning for Wearable Health Monitoring," Bioengineering, 2026. doi: 10.3390/bioengineering13050559.

[31] Balasubramanian S.A., "Framework-aware query orchestration for Angular micro-frontends: a type-safe approach to GraphQL deduplication and performance optimization," PeerJ Computer Science, 2026. doi: 10.7717/peerj-cs.3650.

[32] S. Kailas Nath, A. Cecil Donald, and R. Nismon Rio, "Energy-Aware Real-Time Rendering for Isomorphic Web Apps," in Proc. ADSSSC 2026 - International Conference on AI-Driven Solutions for Sustainable Smart Cities: Challenges and Opportunities, 2026. doi: 10.1109/ADSSSC67751.2026.11582458.

[33] Potdar A.A. and P. Goswami, "The Evolution of Cloud Computing: From Virtual Machines to Serverless," Next-Generation Technologies in Cloud Computing: From AI and Security to Sustainability, 2026. doi: 10.1002/9781394382484.ch2.

[34] K. Subramani, J. Jueckstock, A. Kapravelos, and R. Perdisci, "SoK: Workerounds - Categorizing Service Worker Attacks and Mitigations," in Proc. Proceedings - 7th IEEE European Symposium on Security and Privacy, Euro S and P 2022, 2022. doi: 10.1109/EuroSP53844.2022.00041.

[35] K. Dangol, S. Kothandapani, R. Mahat, S. Shweta, A. Roohi, and Chaeikar S.S., "A Review on API Security Risk and Vulnerability Assessment," 2024 IEEE Consumer Life Tech, ICLT 2024, 2024. doi: 10.1109/ICLT63507.2024.11038691.

[36] A. Kolhar and Gundoor T.K., "API Security Testing and Exploitation Techniques," Advanced Cybersecurity for Threats Exploitation and Digital Risk, 2026. doi: 10.4018/979-8-2600-0747-1.ch004.

[37] F. Urdih, T. Theodoropoulos, and U. Zdun, "Performance Patterns for CI/CD Pipelines," Lecture Notes in Computer Science, 2026. doi: 10.1007/978-3-032-19154-0_20.

[38] Gite H., "Optimal User-Driven Performance Governance for Server-Side Rendered Web Applications," in Proc. 2026 2nd International Conference on Big Data and Machine Learning, ICBDML 2026, 2026. doi: 10.1109/ICBDML68582.2026.11544397.

[39] Z. Wei, Y. Zhao, Z. Lyu, X. Yuan, Y. Zhang, and L. Feng, "Cooperative caching algorithm for mobile edge networks based on multi-agent meta reinforcement learning," Computer Networks, 2024. doi: 10.1016/j.comnet.2024.110247.

[40] J. Deepakraj, P. Thangavel, and D. Deepa, "AI-Driven Smart Caching for Optimized Web Performance and Reduced Latency," in Proc. Proceedings - 4th International Conference on Smart

Technologies, Communication and Robotics 2025, STCR 2025, 2025. doi: 10.1109/STCR62650.2025.11018927.

[41] M. Wallis, F. Henskens, and M. Hannaford, "Expanding the cloud: A component-based architecture to application deployment on the Internet," in Proc. CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing, 2010. doi: 10.1109/ccgrid.2010.14.

[42] Guo J., "Intelligent task scheduling in big data clusters: a multi-layer adaptive approach with reinforcement learning," Australian Journal of Electrical and Electronics Engineering, 2025. doi: 10.1080/1448837X.2025.2605392.

[43] M. Siswo Utomo, E. Utami, Kusrini, and A. Setyanto, "Machine Learning Innovations in Code Generation: A Systematic Literature Review of Methods, Challenges and Directions," in Proc. Proceedings - 2024 International Conference on Information Technology and Computing, ICITCOM 2024, 2024. doi: 10.1109/ICITCOM62788.2024.10762291.

[44] Arumugam S.K., Reddy A.V.D., and Tyagi A.K., "Big data, artificial intelligence, and machine learning support for e-learning frameworks," Architecture and Technological Advancements of Education 4.0, 2023. doi: 10.4018/9781668492857.ch011.

[45] F. Bülthoff and M. Maleshkova, "RESTful or RESTless-current state of today's top web APIs," in Proc. CEUR Workshop Proceedings, 2014.

[46] D. Sushma, Nalini M.K., R. Ashok Kumar, and M. Nidugala, "To Detect and Mitigate the Risk in Continuous Integration and Continues Deployments (CI/CD) Pipelines in Supply Chain Using Snyk tool," in Proc. 7th IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2023 - Proceedings, 2023. doi: 10.1109/CSITSS60515.2023.10334136.

[47] S. Iserte, Tomas V.R., M. Perez, M. Castillo, P. Boronat, and Garcia L.A., "Complete Integration of Team Project-Based Learning Into a Database Syllabus," IEEE Transactions on Education, 2023. doi: 10.1109/TE.2022.3217309.

[48] S. Kermo and A. Dželihodžić, "Comparative analysis of server-side, client-side, and hybrid rendering approaches in modern web applications," in Proc. 2026 25th International Symposium INFOTEH-JAHORINA, INFOTEH 2026 - Proceedings, 2026. doi: 10.1109/INFOTEH68759.2026.11477683.

[49] K. Vayadande, S. Parashar, S. Chavan, R. Tambe, S. Mahajan, and B. Singh, "Development of Latest Technologies in Web Development: A Survey of Methods and Trends," in Proc. 15th International Conference on Advances in Computing, Control, and Telecommunication Technologies, ACT 2024, 2024.

[50] A. Mhatre and S. Mali, "Progressive Web Applications, a New Way for Faster Testing of Mobile Application Products," in Proc. 2023 3rd Asian Conference on Innovation in Technology, ASIANCON 2023, 2023. doi: 10.1109/ASIANCON58793.2023.10269806.

[51] Abdalzaher M.S., M. Krichen, Shaaban M.F., and Fouda M.M., "Quality-Focused Internet of Things Data Management: A Survey, Perspectives, Open Issues, and Challenges," IEEE Internet of Things Journal, 2025. doi: 10.1109/JIOT.2025.3613669.

[52] Zhu L. et al., "Achieving reliable high-frequency releases in cloud environments," IEEE Software, 2015. doi: 10.1109/MS.2015.23.

[53] Villamizar M. et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in Proc. 2015 10th Colombian Computing Conference, 10CCC 2015, 2015. doi: 10.1109/ColumbianCC.2015.7333476.

[54] M. Yoganandham and Devi S.V.S., "Automated Application Deployment on Ubuntu Server Using GitHub Actions CI/CD Pipeline," in Proc. 2026 International Conference on Innovations in Emerging Technologies for Sustainable Development: Empowering Innovation for a Sustainable Future, ICIETSD 2026 - Conference Proceedings, 2026. doi: 10.1109/ICIETSD68684.2026.11584713.

[55] Y. Izrailevsky and C. Bell, "Cloud Reliability," IEEE Cloud Computing, 2018. doi: 10.1109/MCC.2018.032591615.

PROFIL PENULIS

1. Andri Triyono, M.T.

Andri Triyono, M.T. adalah akademisi, peneliti, dan praktisi rekayasa perangkat lunak yang lahir pada 20 Juni 1988. Saat ini beliau mengabdikan diri sebagai Dosen Tetap Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan, Universitas An Nuur (UNAN) serta menjabat sebagai Kepala Unit Pelaksana Teknis Sistem Informasi (UPT SI) Universitas An Nuur. Beliau menempuh pendidikan Sarjana (S1) dan Magister (S2) Teknik Informatika di Universitas Atma Jaya Yogyakarta (UAJY). Fokus kepakaran beliau mencakup Full-Stack Software Engineering, Server Administration & Virtualization (Proxmox VE & LXC), Database Optimization, serta Large Language Models (LLMs) & Agentic Workflows.

- **Nama & NIDN:** Andri Triyono, M.T. (NIDN: 0620068802)
- **Jabatan:** Dosen S1 Ilmu Komputer & Kepala UPT Sistem Informasi Universitas An Nuur.
- **Kepakaran:** Full-Stack Web Architecture, Linux Server Orchestration, Database Engineering, Agentic AI Systems.

2. Eko Supriyadi, S.Kom., M.Tr.Kom.

Eko Supriyadi, S.Kom., M.Tr.Kom. adalah akademisi dan pendidik di bidang teknologi informasi yang saat ini menjabat sebagai Ketua Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan, Universitas An Nuur. Beliau memiliki dedikasi tinggi dalam perancangan dan implementasi kurikulum pendidikan tinggi berbasis Outcome-Based Education (OBE) dan Project-Based Learning (PjBL) guna mencetak lulusan yang berdaya saing global dan berjiwa technopreneur.

- **Nama:** Eko Supriyadi, S.Kom., M.Tr.Kom.
- **Jabatan:** Ketua Program Studi S1 Ilmu Komputer Universitas An Nuur.
- **Kepakaran:** Software Engineering Management, Information Systems Architecture, OBE Curriculum Development.

3. Kartika Imam Santoso, M.Kom.

Kartika Imam Santoso, M.Kom. adalah dosen tetap pada Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan, Universitas An Nuur. Beliau aktif dalam kegiatan Tridharma Perguruan Tinggi, penulisan publikasi ilmiah internasional terindeks Scopus, serta riset terapan di bidang data mining, machine learning, dan integrasi kecerdasan buatan pada aplikasi web interaktif.

- **Nama:** Kartika Imam Santoso, M.Kom.
- **Jabatan:** Dosen Program Studi S1 Ilmu Komputer Universitas An Nuur.
- **Kepakaran:** Data Mining, Machine Learning, Web Analytics, Smart Decision Systems.

4. Rohman Hadi Al Haq, M.Kom.

Rohman Hadi Al Haq, M.Kom. adalah dosen dan peneliti pada Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan, Universitas An Nuur. Beliau berfokus pada rekayasa perangkat lunak, infrastruktur jaringan komputer, penjaminan mutu sistem perangkat lunak, serta metodologi pengujian keamanan aplikasi web modern.

- **Nama:** Rohman Hadi Al Haq, M.Kom.
- **Jabatan:** Dosen Program Studi S1 Ilmu Komputer Universitas An Nuur.
- **Kepakaran:** Computer Networks, Software Quality Assurance, Cyber Security, Distributed Systems.

5. Dhika Malita Puspita A., M.Kom.

Dhika Malita Puspita A., M.Kom. adalah dosen tetap pada Program Studi S1 Ilmu Komputer, Fakultas Sains dan Kesehatan, Universitas An Nuur. Beliau mendedikasikan keilmuannya dalam bidang interaksi manusia dan komputer (Human-Computer Interaction), rekayasa pengalaman pengguna (UI/UX Engineering), tata kelola sistem informasi, serta pedagogi komputasi.

- **Nama:** Dhika Malita Puspita A., M.Kom.
- **Jabatan:** Dosen Program Studi S1 Ilmu Komputer Universitas An Nuur.
- **Kepakaran:** Human-Computer Interaction, Modern UI/UX Engineering, Information System Governance.