

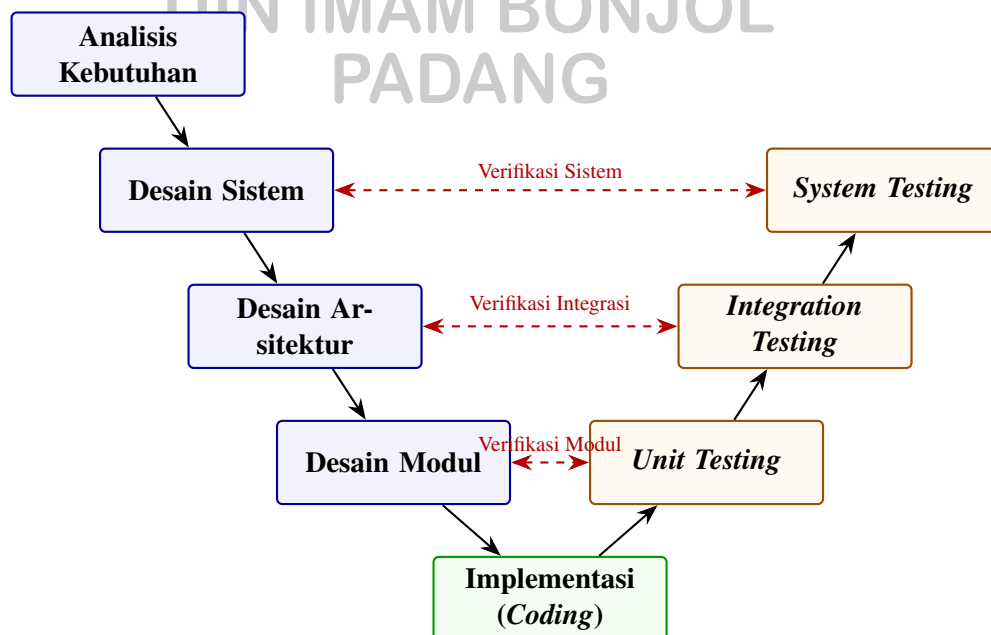
BAB III METODE PENELITIAN

A. Jenis Penelitian

Penelitian ini merupakan penelitian terapan (*applied research*) yang bertujuan untuk merancang dan membangun arsitektur *backend* serta menguji performa teknisnya secara kuantitatif berdasarkan ambang batas (*threshold*) terukur. Penggunaan istilah rancang bangun pada judul penelitian menitikberatkan pada perancangan dan implementasi prototipe arsitektur perangkat lunak yang terverifikasi keandalannya, bukan aplikasi siap pakai skala produksi penuh. Siklus pengembangan mengacu pada kerangka kerja V-Model (*Verification and Validation Model*), mulai dari tahap analisis kebutuhan hingga tahap *System Testing* (Anjasmara & Sumitro, 2023).

B. Tahapan Penelitian

Penelitian ini menerapkan kerangka kerja V-Model yang menyelaraskan setiap tahap perancangan di sisi kiri dengan tahap pengujian di sisi kanan secara berpasangan (Gambar 3.1). Pendekatan ini memastikan setiap keputusan perancangan memiliki mekanisme verifikasi yang terdefinisi dengan jelas.



Gambar 3.1. Diagram Alur Penelitian V-Model
Sumber: Dibuat oleh peneliti

Tahap pengujian pada penelitian ini difokuskan hingga *System Testing*. Pengujian tingkat akhir berupa *Acceptance Testing* belum dilakukan karena dua pertimbangan. Pertama, penelitian ini berfokus pada evaluasi teknis keandalan arsitektur *backend*, meliputi kinerja di bawah beban konkurensi, mekanisme antrean pesan asinkron, serta skalabilitas layanan. Kedua, dari pihak TIME Language Centre, silabus kurikulum dan modul materi pembelajaran saat ini masih dalam tahap perancangan internal, sehingga konten kursus belum siap untuk rilis produksi maupun diujikan secara langsung kepada peserta didik atau pengguna akhir.

Keberhasilan integrasi antarlayanan diverifikasi melalui *Integration Testing*, sedangkan *System Testing* memvalidasi sistem secara menyeluruh dari sisi fungsional *end-to-end* sekaligus mengukur ketercapaian target performa arsitektur. Rincian setiap tahapan diuraikan sebagai berikut:

1. Analisis Kebutuhan

Tahap awal penelitian dimulai dengan mengidentifikasi kebutuhan sistem melalui diskusi dan analisis proses bisnis di TIME Language Centre. Hasil analisis ini dirumuskan menjadi kebutuhan fungsional, pemodelan *use case*, dan kebutuhan non-fungsional sebagai acuan kualitas sistem.

Pada V-Model konvensional, analisis kebutuhan berpasangan simetris dengan *Acceptance Testing*. Dalam penelitian ini, validasi kebutuhan fungsional dialihkan ke *System Testing* melalui skenario *end-to-end* dan pengujian API fungsional, sedangkan kebutuhan non-fungsional divalidasi menggunakan pengujian beban.

2. Desain Sistem

Pada tahap ini, batas sistem (*system boundary*) dirancang untuk memperjelas komponen internal yang dikembangkan beserta interaksinya dengan entitas eksternal, seperti pengguna dan layanan *cloud* pihak ketiga (*payment gateway*, CDN video, autentikasi, dan pengiriman *email*).

Sistem kemudian dibagi ke dalam beberapa layanan mikro (*microservices*) mandiri. Alur komunikasi antarlayanan ditentukan: sinkron melalui *API Gateway*, dan asinkron berbasis *event-driven* melalui *message broker*. Perancangan sistem ini berpasangan dengan *System Testing* pada fase pengujian.

3. Desain Arsitektur

Fase desain arsitektur menetapkan pola dasar dan teknologi yang digunakan. Penelitian ini menerapkan arsitektur *Event-Driven Microservices* dengan prinsip utama: kemandirian layanan (*service autonomy*), penerapan pola *Database per Service*, keterikatan longgar (*loose coupling*), dan pemilihan bahasa pemrograman sesuai karakteristik tiap layanan (*polyglot programming*).

Komunikasi asinkron berbasis pesan (*asynchronous messaging*) antar-layanan menggunakan protokol AMQP pada RabbitMQ melalui pertukaran pesan *event* menggunakan pola *direct exchange*. Mekanisme antrean pesan gagal (*Dead Letter Queue/DLQ*) disiapkan untuk menjaga keandalan saat terjadi kegagalan pemrosesan pesan. Pada lapisan penyimpanan, sistem menerapkan pola *Database per Service* yang memisahkan pengelolaan basis data PostgreSQL pada setiap layanan, dengan dukungan Redis untuk penyimpanan *cache*. Seluruh rancangan arsitektur ini divalidasi pada tahap *Integration Testing*.

4. Desain Modul

Rancangan arsitektur kemudian diturunkan ke spesifikasi internal setiap layanan mikro. Struktur kelas, entitas domain, lapisan logika bisnis (*service layer*), repositori data, dan pengontrol (*controller*) digambarkan melalui *class diagram*.

Alur eksekusi pesan dan rincian transaksi antarkomponen dimodelkan menggunakan *sequence diagram*. Keseluruhan spesifikasi modul ini menjadi acuan utama saat pelaksanaan *Unit Testing*.

5. Implementasi

Tahap implementasi adalah proses menerjemahkan seluruh rancangan ke dalam kode program berbasis struktur *monorepo*. Setiap layanan mikro dikembangkan dan dikemas ke dalam kontainer Docker sehingga dapat dijalankan secara terintegrasi melalui Docker Compose.

Pada tahap ini diimplementasikan: *API Gateway* Nginx untuk merutekan lalu lintas HTTP, konfigurasi basis data PostgreSQL dan Redis, konfigurasi *exchange* serta antrean pada RabbitMQ, mekanisme keamanan (autentikasi berbasis JWT, *hashing* kata sandi menggunakan *bcrypt*, dan pembatasan laju per-

mintaan/*rate limiting*), serta instrumen pemantauan sistem (*system monitoring*) yang mencakup Prometheus, Grafana, dan Loki.

6. Pengujian

Fase pengujian mengeksekusi sisi kanan V-Model dari unit terkecil hingga sistem secara utuh, mencakup tiga tingkatan:

- a. *Unit Testing*: Memvalidasi logika internal pada komponen terkecil (*controller*, *service layer*, dan entitas) secara terisolasi, tanpa melibatkan basis data nyata atau *message broker*.
- b. *Integration Testing*: Memverifikasi komunikasi antarlayanan, baik alur permintaan REST API sinkron maupun pertukaran pesan asinkron melalui RabbitMQ, termasuk ketahanan antrean pesan saat terjadi kegagalan pemrosesan.
- c. *System Testing*: Menguji sistem yang telah terintegrasi melalui *API Gateway* pada lingkungan kontainer. Pengujian ini mencakup dua aspek utama:
 - 1) Pengujian fungsional *end-to-end* (E2E) secara *black-box* untuk memverifikasi alur proses bisnis dari pendaftaran hingga kelulusan kursus.
 - 2) Pengujian kinerja dan skalabilitas menggunakan Grafana k6 di bawah dua skenario beban konkuren (penelusuran katalog publik dan *pipeline webhook* pembayaran asinkron), untuk mengukur responsivitas, *throughput*, tingkat kegagalan permintaan, serta konsistensi data lintas layanan terdistribusi.

Beban pengujian disimulasikan menggunakan pengguna virtual (*Virtual Users*) yang ditentukan berdasarkan estimasi jumlah alumni TIME Language Centre dan rujukan dari literatur pengujian *microservices* sejenis. Pendekatan ini bertujuan melakukan verifikasi arsitektural terhadap keandalan dan daya tahan sistem.

C. Setting Penelitian

1. Tempat Penelitian

Penelitian ini mengambil studi kasus pada lembaga pendidikan TIME Language Centre yang berlokasi di Kota Padang, Sumatera Barat. Lembaga ini bergerak dalam bidang kursus persiapan IELTS, pembekalan *Work and Holiday Visa*

(WHV), dan persiapan studi ke luar negeri. Pemilihan TIME Language Centre didasarkan pada kebutuhan lembaga untuk beralih ke platform pembelajaran daring yang terukur (*scalable*) guna memperluas jangkauan layanan. Sistem yang dikembangkan dalam penelitian ini difokuskan sebagai prototipe arsitektur perangkat lunak yang teruji, dengan proses bisnis aktual pada lembaga dijadikan acuan utama dalam merumuskan kebutuhan fungsional dan non-fungsional sistem.

2. Waktu Penelitian

Penelitian ini dilaksanakan dalam kurun waktu enam bulan dengan mengikuti tahapan pada kerangka kerja V-Model. Rincian jadwal pelaksanaan setiap tahapan disajikan pada Tabel 3.1.

Tabel 3.1. Jadwal Pelaksanaan Penelitian

No	Tahapan Kegiatan	Bulan 1				Bulan 2				Bulan 3				Bulan 4				Bulan 5				Bulan 6			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Analisis Kebutuhan	•	•	•																					
2	<i>System Design</i>			•	•																				
3	<i>Architecture Design</i>				•	•	•																		
4	<i>Module Design</i>						•	•																	
5	Implementasi (<i>Coding</i>)							•	•	•	•	•	•	•	•	•									
6	<i>Unit and Integration Testing</i>										•	•	•	•	•	•									
7	<i>System Testing</i>															•	•	•							
8	Penyusunan Laporan	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

D. Alat/Instrumen Penelitian

Instrumen yang digunakan dalam penelitian ini mencakup perangkat keras, perangkat lunak, serta instrumen pengumpulan data.

1. Perangkat Keras

Spesifikasi perangkat keras yang digunakan pada tahap implementasi dan pengujian sistem (*System Testing*) disajikan pada Tabel 3.2.

Tabel 3.2. Spesifikasi Perangkat Keras

No	Komponen	Spesifikasi
1	Komputer Utama (<i>Host</i>)	Unit laptop dengan CPU AMD Ryzen 5 6600H (arsitektur 6 <i>core</i> / 12 <i>thread</i>), memori kerja RAM 16 GB LPDDR5, media penyimpanan 250 GB SSD NVMe, serta OS Windows 11 Home
2	Alokasi SUT (<i>System Under Test</i> - Docker Desktop)	Docker Desktop pada Windows 11, alokasi 4 vCPU, 4 GB RAM
3	Alokasi Klien Penguji (Grafana k6 Host)	Berjalan pada <i>host OS</i> (Windows 11) menggunakan sisa sumber daya <i>host</i>

2. Perangkat Lunak

Konfigurasi perangkat lunak yang digunakan dalam penelitian ini disajikan pada Tabel 3.3. Tabel ini merangkum komponen perangkat lunak utama arsitektur, basis data, infrastruktur, dan pengujian beban sistem, sedangkan pustaka pendukung pengujian unit pada tingkat kode sumber (seperti Jest dan Testify) dijabarkan secara kontekstual pada Bab 4.

Tabel 3.3. Spesifikasi Perangkat Lunak

No	Perangkat Lunak	Versi	Fungsi
Bahasa Pemrograman dan <i>Runtime</i>			
1	Node.js	18	<i>Runtime</i> JavaScript/TypeScript
2	Go (<i>Golang</i>)	1.23	Bahasa pemrograman untuk <i>Payment Service</i>
3	TypeScript	5.9.3	Bahasa pemrograman bertipe statis untuk layanan Node.js
Framework dan Pustaka			
4	Express.js	4.21	<i>Web framework</i> Node.js

No	Perangkat Lunak	Versi	Fungsi
5	Fiber	2.52	<i>Web framework Go</i>
6	TypeORM	0.3.20	ORM TypeScript
7	GORM	1.25.12	ORM Go
Infrastruktur dan Orkestrasi			
8	Docker	27.0	Kontainerisasi layanan
9	Docker Compose	2.29	Orkestrasi multi-kontainer
10	Nginx	1.27	<i>API Gateway</i> dan <i>reverse proxy</i>
Message Broker dan Basis Data			
11	RabbitMQ	3.11	<i>Message broker AMQP</i>
12	PostgreSQL	15	Basis data relasional
13	Redis	7.2	<i>In-memory cache</i>
Layanan Cloud dan Eksternal			
14	Firebase Auth	11.0	Autentikasi pengguna
15	Xendit	API v2	<i>Payment gateway</i>
16	Bunny.net	N/A	CDN dan penyimpanan video
17	Resend	4.0	Pengiriman <i>email</i> transaksional
Instrumen Pemantauan Sistem (<i>System Monitoring</i>) dan Pengujian Beban			
18	Prometheus	2.45	Pengumpul metrik performa
19	Grafana	10.0	Visualisasi metrik
20	Loki	2.9	Pengumpul log terpusat
21	Grafana k6	0.45	Pengujian beban sistem

E. Teknik Pengumpulan Data

Pengumpulan data dilakukan melalui tiga teknik utama: wawancara dengan pemangku kepentingan di lembaga, observasi langsung terhadap operasional kursus, dan studi pustaka teknis. Ketiga teknik ini saling melengkapi guna memastikan spesifikasi kebutuhan sistem memiliki landasan operasional dan teoretis yang kuat.

1. Wawancara

Wawancara dilakukan baik secara luring maupun daring melalui platform *video conference* (Zoom) bersama pendiri (*founder*) lembaga, Bapak Melqi Hamdani, S.T., serta staf administrasi TIME Language Centre. Sesi wawancara ini bertujuan untuk menggali alur operasional kursus, proses pendaftaran peserta didik, kendala operasional yang dihadapi, serta kebutuhan fitur pada platform pembelajaran daring yang akan dibangun.

Hasil wawancara dikonfirmasi kembali kepada pihak mitra, kemudian diterjemahkan menjadi spesifikasi kebutuhan fungsional dan non-fungsional pada tahap *Requirements Analysis V-Model*.

2. Observasi

Observasi dilakukan dengan mengunjungi langsung lokasi TIME Language Centre untuk mengamati proses operasional dan model pembelajaran yang sedang berjalan. Lembaga ini menyelenggarakan program *intensive camp* persiapan tes bahasa Inggris, yaitu kelas tatap muka terpadu dalam periode waktu tertentu.

Fokus observasi diarahkan pada alur pendaftaran peserta didik, proses verifikasi transaksi pembayaran, pengelolaan jadwal kelas, serta distribusi materi belajar. Hasil observasi lapangan ini menjadi dasar penyusunan daftar kebutuhan fungsional dan non-fungsional pada tahap *Requirements Analysis*.

3. Studi Pustaka

Studi pustaka dilakukan untuk menghimpun landasan teoretis serta pemahaman teknis terhadap teknologi yang diterapkan. Rujukan utama diperoleh dari artikel jurnal ilmiah bereputasi (terindeks Scopus dan SINTA) yang membahas arsitektur *microservices*, komunikasi asinkron berbasis *event-driven*, pengujian performa perangkat lunak, serta containerisasi.

Buku teks rekayasa perangkat lunak digunakan sebagai acuan metodologi pengembangan, sedangkan dokumentasi resmi teknologi (Node.js, Go, Docker, RabbitMQ, dan instrumen observabilitas) menjadi acuan teknis implementasi. Seluruh kajian kepustakaan ini menjadi landasan teoretis dalam perancangan arsitektur sistem.

Untuk menjaga keabsahan data, diterapkan teknik triangulasi terhadap ketiga metode tersebut. Informasi dari hasil wawancara diverifikasi silang dengan observasi lapangan guna memastikan kesesuaian antara kebutuhan yang disampaikan pemangku kepentingan dengan kondisi operasional nyata. Temuan dari kedua metode tersebut kemudian diperkuat dengan kajian literatur sebelum dirumuskan menjadi spesifikasi arsitektur sistem.

F. Teknik Analisis Data

Tahap analisis data berfokus pada pengolahan metrik kuantitatif yang diperoleh dari pengujian beban sistem menggunakan Grafana k6 pada tahap *System Testing*. Pendekatan yang digunakan adalah evaluasi deskriptif berbasis kriteria ambang batas (*threshold-based evaluation*), dengan parameter toleransi kualitas non-fungsional didefinisikan langsung di dalam skrip pengujian Grafana k6 untuk menentukan status kelulusan (*pass/fail*) secara otomatis dan objektif (Grafana Labs, 2025a).

Parameter kinerja yang diukur dan dievaluasi mengacu pada kerangka kerja Bjørndal dkk. (2021) dan panduan resmi Grafana k6 (Grafana Labs, 2025a), yang meliputi:

- a. Metrik Tren (*Custom Trend Metrics*): Latensi per skenario dan per langkah (*step*) interaksi pengguna, direkam menggunakan objek *Trend* pada Grafana k6 untuk menganalisis karakteristik performa masing-masing layanan mikro secara terperinci.
- b. *Throughput*: Total iterasi perjalanan pengguna lengkap per detik (*iterations/s*) dan total permintaan HTTP per detik (*req/s*) pada setiap tahap beban.
- c. Tingkat Kegagalan Permintaan HTTP (*HTTP Request Failure Rate*): Persentase permintaan HTTP yang gagal (*http_req_failed*) yang mengukur seluruh respons di luar kode status 2xx selama pengujian berlangsung.

1. Parameterisasi Data

Skenario B (*Pipeline Webhook Event-Driven*) memerlukan kumpulan data transaksi *PENDING* yang disiapkan sebelum pengujian dimulai (*pre-seeding*). Strategi *pre-seeding* ini dipilih karena dua pertimbangan teknis yang saling berkait-

tan.

Pertama, mekanisme idempotensi pada *Payment Service* dirancang untuk menolak konfirmasi pembayaran apabila transaksi yang dirujuk sudah berstatus *PAID*. Dengan demikian, satu ID transaksi hanya dapat digunakan dalam satu iterasi *webhook*. Apabila data tidak disiapkan di awal dan *webhook* dikirim berulang dengan ID yang sama, *Payment Service* akan melewati penerbitan pesan ke RabbitMQ sehingga alur asinkron tidak pernah diuji sesungguhnya.

Kedua, alur pembuatan transaksi baru (*POST /payments/checkout*) menerapkan validasi bisnis bahwa pengguna tidak dapat membeli kursus yang sama lebih dari satu kali. Pada beban 80 VU yang beroperasi selama 360 detik, simulasi konkurensi menunjukkan probabilitas tabrakan transaksi mendekati 100%, yang akan menghasilkan lonjakan respons HTTP 400. Respons tersebut tidak merepresentasikan kegagalan sistem melainkan penolakan validasi bisnis, dan berpotensi mengaburkan pengukuran NFR-02.

Solusi yang diterapkan adalah menyiapkan 30.000 data transaksi *PENDING* unik melalui skrip penyiapan data awal (*data seeding*) sebelum pengujian dimulai. ID transaksi tersebut dimuat ke dalam memori bersama menggunakan mekanisme *SharedArray* dari Grafana k6, kemudian dialokasikan secara sekuensial ke setiap iterasi menggunakan indeks iterasi unik skenario sehingga setiap iterasi dijamin mengonsumsi satu ID unik tanpa tabrakan (Grafana Labs, 2025a). Jumlah 30.000 transaksi ini disiapkan sebagai kapasitas penyangga (*buffer*) yang mencukupi untuk memenuhi seluruh kebutuhan iterasi selama durasi pengujian 360 detik, guna mencegah pengujian terhenti akibat kehabisan data unik.

2. Desain Skenario Pengujian

Pengujian kinerja dirancang ke dalam dua skenario perjalanan pengguna (*user journey*) yang dieksekusi secara konkuren menggunakan arsitektur multi-skenario Grafana k6 dengan alokasi *Virtual Users* (VU) terisolasi per skenario (Grafana Labs, 2025b):

- a. Skenario A: Penelusuran Katalog Publik (60% beban, hingga 120 VU): Setiap iterasi mengeksekusi *GET /courses* (pengambilan daftar katalog kursus) di-

ikuti *GET /courses/:id* (pengambilan rincian silabus materi) sebagai pengunjung publik tanpa autentikasi. Skenario ini menguji responsivitas *Course Service* dan efektivitas lapisan *caching* Redis di bawah beban baca (*read-heavy*) dari pengguna publik. Alokasi 60% ditetapkan sebagai porsi beban terbesar karena penelusuran katalog merepresentasikan lalu lintas publik yang paling dominan pada platform kursus daring. Setelah setiap langkah disisipkan jeda waktu (*think time*) 1 detik untuk mensimulasikan perilaku pengguna nyata saat membaca konten halaman.

- b. Skenario B: *Pipeline Webhook Event-Driven* Pembayaran (40% beban, hingga 80 VU): Setiap iterasi mengeksekusi satu permintaan *POST /payments/xendit-webhook* dengan mengambil ID transaksi unik dari data yang telah disiapkan sebelumnya (*pre-seeded*). Skenario ini mengukur dua hal secara bersamaan: (1) kecepatan *Payment Service* dalam memvalidasi token *webhook* dan menerbitkan pesan *event* ke RabbitMQ, dan (2) keandalan propagasi asinkron pesan tersebut ke *Course Service* dan *Notification Service* secara paralel. Latensi respons HTTP pada skenario ini merepresentasikan kapasitas komputasi *Payment Service* karena layanan ini langsung mengembalikan respons HTTP 200 segera setelah pesan berhasil diterbitkan ke *broker*, tanpa menunggu proses penyelesaian di layanan hilir (*fire-and-forget*).

Distribusi beban skenario diimplementasikan menggunakan eksekutor *ramping-vus* multi-skenario terpisah per rute perjalanan sesuai panduan Grafana k6 (Grafana Labs, 2025b). Proporsi 60% dan 40% (120 VU dan 80 VU pada beban uji 200 VU) ditetapkan untuk mengevaluasi dua karakteristik arsitektur utama: lapisan akses publik berbasis *cache* (Skenario A) dan pemrosesan transaksi asinkron berbasis antrean pesan (Skenario B).

3. Strategi Pengujian Beban

Sesuai rekomendasi metodologi Grafana k6 (Grafana Labs, 2025a), sebelum pengujian beban skala penuh dieksekusi, sistem terlebih dahulu divalidasi melalui pengujian awal (*smoke testing*) menggunakan mode uji cepat pada skrip pengujian beban. Pengujian awal ini menjalankan kedua skenario dengan beban minimal

(masing-masing 1 VU, 3 iterasi, durasi maksimum 30 detik) untuk memverifikasi bahwa seluruh rute proksi Nginx, konektivitas antarlayanan, dan koneksi basis data berfungsi dengan benar sebelum sistem menerima beban besar.

Setelah *smoke test* berhasil tanpa kesalahan, pengujian dilanjutkan ke pengujian beban bertahap (*staged load testing*) skala penuh, dilaksanakan dalam dua fase menggunakan mekanisme tahapan peningkatan beban (*ramp-up stages*):

- a. Fase 1 (Beban Nominal): Beban dinaikkan bertahap hingga total 100 VU (60 VU Skenario A + 40 VU Skenario B) selama 100 detik, dipertahankan 60 detik sebagai periode pengukuran kestabilan pada beban nominal, kemudian diturunkan ke 0 VU selama 20 detik sebagai jeda pemulihan sistem (*cool-down*).
- b. Fase 2 (Beban Uji 200 VU): Beban dinaikkan kembali bertahap hingga total 200 VU (120 VU Skenario A + 80 VU Skenario B) selama 100 detik, dipertahankan 60 detik sebagai periode pengukuran stabilitas pada beban uji 200 VU, kemudian diturunkan ke 0 VU selama 20 detik.

Target beban uji 200 VU ditetapkan sebagai skenario beban terkontrol yang dirancang untuk mengevaluasi ketahanan konkurensi arsitektur *event-driven microservices* pada kondisi beban yang melebihi kebutuhan operasional awal lembaga. Skala ini dipilih selaras dengan rentang beban evaluasi pada literatur arsitektur layanan mikro (Bjørndal dkk., 2021; Permana dkk., 2023), bukan sebagai representasi kapasitas produksi aktual. Total durasi pengujian berlangsung selama 360 detik (6 menit), memberikan rentang waktu pengamatan yang memadai pada setiap tahapan beban.

4. Kriteria Kelulusan Kinerja

Kriteria kelulusan ditetapkan sebagai ambang batas kinerja minimum yang dapat diterima untuk konteks prototipe arsitektur pada lingkungan pengujian lokal terkontrol. Pendekatan ini selaras dengan panduan Grafana k6 yang menganjurkan penetapan ambang batas (*thresholds*) berdasarkan tujuan keandalan sistem yang sedang diuji (Grafana Labs, 2025a). Dua ambang batas yang ditetapkan sebagai kriteria *pass/fail*:

- a. Waktu Respons Global (NFR-01): Persentil ke-95 (P95) agregat seluruh permintaan HTTP harus di bawah 200 ms, dan persentil ke-99 (P99) harus di bawah 400 ms (Grafana Labs, 2025c). Penetapan nilai 200 ms sebagai batas P95 merujuk pada panduan kinerja web Grafana k6 sebagai standar sistem yang responsif, sedangkan batas 400 ms pada P99 merepresentasikan batas atas toleransi sebelum pengguna merasakan jeda interaksi (Grafana Labs, 2025c). Kedua nilai ini berada di bawah ambang batas responsivitas 1,0 detik (Nielsen, 1993). Metrik yang digunakan adalah *http_req_duration* bawaan Grafana k6 sebagai representasi perilaku sistem secara menyeluruh di bawah beban multi-skenario konkuren. Metrik kustom per skenario (*dur_A_catalog_list_ms*, *dur_A_catalog_detail_ms*, *dur_B_webhook_ms*) direkam sebagai data diagnostik karakteristik latensi antarlayanan, namun tidak dijadikan kriteria *pass/fail* tersendiri.
- b. Tingkat Kegagalan Permintaan (NFR-02): Tingkat kegagalan permintaan HTTP yang diukur melalui metrik *http_req_failed* bawaan Grafana k6 harus di bawah 1% (Grafana Labs, 2025c). Pada rancangan pengujian ini, seluruh permintaan HTTP yang diharapkan mengembalikan respons 2xx (HTTP 200 OK). Respons HTTP 4xx tidak muncul dalam pengujian ini karena desain skenario *pre-seeding* memastikan setiap iterasi menggunakan ID transaksi yang valid dan belum pernah digunakan, sehingga mekanisme idempotensi tidak akan terpicu selama pengujian berlangsung. Dengan demikian, metrik *http_req_failed* secara langsung mencerminkan kesalahan nyata pada sistem.

Hasil pengujian dianalisis secara deskriptif untuk mengevaluasi karakteristik latensi per skenario, *throughput* agregat, keandalan pemrosesan pada beban uji 200 VU, dan konsistensi data lintas layanan (*eventual consistency*). Konsistensi data diverifikasi pascapengujian melalui kueri agregasi langsung ke masing-masing basis data terdistribusi (*payment_service_db*, *course_service_db*, dan *notifications_db*) untuk memastikan jumlah transaksi *PAID*, hak akses (*entitlement*) aktif, dan notifikasi *SENT* sesuai secara proporsional.