

**ANALISIS DAN EVALUASI PERFORMA *DASHBOARD*
ADMIN SISTEM CPL BERBASIS OBE MENGGUNAKAN
APACHE JMETER**

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



**AHMAD FAJRUL 'ULUM
22EO10027**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

**ANALISIS DAN EVALUASI PERFORMA *DASHBOARD*
ADMIN SISTEM CPL BERBASIS OBE MENGGUNAKAN
APACHE JMETER**

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



**AHMAD FAJRUL 'ULUM
22EO10027**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa penulisan skripsi dengan judul “Analisis dan evaluasi performa *dashboard* admin sistem CPL berbasis OBE menggunakan Apache JMeter” adalah hasil karya saya dengan arahan dari pembimbing dan belum diajukan kepada pihak manapun. Sumber informasi yang dikutip dalam skripsi ini telah dicantumkan dalam Daftar Pustaka.

Demikian pernyataan ini dibuat dengan sebenarnya. Apabila di kemudian hari terdapat ketidaksesuaian dalam pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Cilacap, 4 Agustus 2026

Ahmad Fajrul ‘Ulum
22EO10027

LEMBAR PENGESAHAN

Skripsi Saudara

Nama : Ahmad Fajrul 'Ulum
NIM : 22EO10027
Fakultas/Prodi : Fakultas MIKOM / Informatika
Judul : Analisis dan evaluasi performa *dashboard* admin sistem
CPL berbasis OBE menggunakan Apache JMeter

Telah disidangkan oleh Dewan Penguji Fakultas Matematika dan Ilmu Komputer Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap pada hari / tanggal :

Selasa, 28 Juli 2026

Dan dapat diterima sebagai pemenuhan tugas akhir mahasiswa Program Strata Satu (S.1) Program Studi Informatika (S.Kom.) Fakultas Matematika dan Ilmu Komputer (FMIKOM) pada Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap.

Cilacap, 4 Agustus 2026



Mengetahui,
Dekan Fakultas Matematika dan Ilmu Komputer

Mochamad T. A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

HALAMAN NOTA KONSULTAN

Mochamad T. A. Aziz Zein, S.Si., M.Kom.

Dosen Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap

NOTA KONSULTAN

Hal : Skripsi Saudara Ahmad Fajrul 'Ulum
Lampiran : -

Kepada
Yth. Dekan Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap
di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah membaca, memeriksa dan melakukan perbaikan seperlunya maka skripsi saudara:

Nama : Ahmad Fajrul 'Ulum
NIM : 22EO10027
Prodi : Informatika
Judul : Analisis dan evaluasi performa *dashboard* admin sistem CPL berbasis OBE menggunakan Apache JMeter

Dapat diajukan ke Fakultas Matematika dan Ilmu Komputer, Universitas Nahdlatul Ulama Al Ghazali Cilacap untuk memenuhi syarat memperoleh gelar Strata Satu (S1).

Wassalamu'alaikum Wr. Wb.

Cilacap, 4 Agustus 2026
Konsultan

Mochamad T. A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

NOTA PEMBIMBING

Cilacap, 17 Juli 2026

Kepada Yth :
Fakultas Matematika dan Ilmu Komputer (FMIKOM)
UNUGHA Cilacap
di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah melakukan bimbingan, telaah, arahan dan koreksi tahap penulisan skripsi saudara:

Nama : Ahmad Fajrul 'Ulum
NIM : 22EO10027
Fakultas : Matematika dan Ilmu Komputer
Prodi : Informatika
Judul : Analisis dan evaluasi performa *dashboard* admin sistem CPL berbasis OBE menggunakan Apache JMeter

Kami berpendapat bahwa skripsi tersebut sudah dapat diajukan ke sidang skripsi. Bersama ini kami kirimkan skripsi tersebut, semoga dapat segera disidangkan.

Atas perhatiannya kami ucapkan terima kasih.
Wassalamu'alaikum Wr. Wb.

Mengetahui,

Pembimbing I

Pembimbing II

Tri Anggoro, M.Kom
NIDN. 0601128103

Riski Aspriyani, M.Pd
NIDN. 0616118901

HALAMAN MOTO

لَا يُكَلِّفُ اللَّهُ نَفْسًا إِلَّا وُسْعَهَا

“Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya.”

(QS. Al-Baqarah: 286)

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT, Rabb semesta alam yang senantiasa memberikan karunia sehingga penulis mampu menyelesaikan penulisan skripsi ini. Karya ini saya persembahkan kepada :

1. Orang tua Bapak Sohidin dan Ibu Muji Supriyatin yang selalu mendidik saya, memberikan do'a, dukungan, nasihat dan semangat yang tiada henti.
2. Adik tercinta Ahmad Solehudin dan saudara tersayang yang tidak bisa saya sebutkan satu persatu yang selalu memberikan do'a dan semangat tiada henti di setiap detik langkahku.
3. Keluarga FMIKOM Angkatan 2022 yang selalu memberikan keceriaan, kebersamaan dan motivasi.
4. Keluarga program studi Informatika, Sistem Informasi, Matematika yang saya banggakan
5. Seluruh teman UNUGHA yang telah memberikan do'a, dukungan, dan semangat.

KATA PENGANTAR

Puji syukur kepada Allah SWT atas segala limpahan nikmat dan karuniaNya, sehingga penulis dapat menyelesaikan penulisan skripsi. Sholawat dan salam senantiasa penulis haturkan kepada Nabi Muhammad SAW sebagai pembimbing seluruh umat manusia.

Skripsi ini tidak akan selesai tanpa adanya bantuan dari banyak pihak, karena itu penulis menyampaikan terima kasih kepada :

1. Bapak Mochamad T. A. Aziz Zein, S.Si., M.Kom, Dekan FMIKOM UNUGHA
2. Bapak Tri Anggoro, M.Kom, pembimbing I yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
3. Ibu Riski Aspriyani, M.Pd, pembimbing II yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
4. Bapak Verry, S.T., M.Kom, sebagai Pembimbing Akademis.
5. Bapak Agun Nurul Widiyanto (Pengelola SSDI / LP2M UNUGHA), yang telah membantu memfasilitasi lingkungan pengujian berupa menyediakan akses VPS UNUGHA sebagai media utama penunjang penelitian ini.
6. Bapak dan Ibu Dosen Program Studi Informatika FMIKOM UNUGHA, yang telah memberikan ilmu dan bekal berharga selama masa perkuliahan.
7. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu, atas segala bantuan dan dukungan yang telah diberikan.

Semoga Allah SWT membalas jerih payah dan pengorbanan yang telah diberikan dengan balasan yang lebih baik. Amiin. Penulis berharap semoga karya kecil ini bermanfaat bagi pembaca.

Cilacap, 4 Agustus 2026

Ahmad Fajrul 'Ulum

ABSTRAK

AHMAD FAJRUL 'ULUM. Analisis dan Evaluasi Performa *Dashboard* Admin Sistem CPL Berbasis OBE Menggunakan Apache JMeter. Dibimbing oleh Bapak Tri Anggoro, M.Kom dan Ibu Riski Aspriyani, M.Pd

Sistem Penilaian Capaian Pembelajaran Lulusan (CPL) berbasis *Outcome-Based Education* (OBE) membutuhkan Dashboard Admin yang mampu menyajikan data agregat secara cepat dan stabil. Penelitian ini bertujuan menganalisis performa tiga endpoint utama Dashboard Admin, yaitu `/api/dashboard/stats`, `/api/dashboard/dosen`, dan `/api/dashboard/students`, serta mengidentifikasi dugaan *bottleneck* dan menyusun rekomendasi perbaikan. Pengujian dilakukan menggunakan Apache JMeter dengan skenario 1, 10, 20, 50, 100, 200, dan 500 pengguna virtual pada server lokal dan VPS UNUGHA. Kedua lingkungan menggunakan aplikasi dan basis data MariaDB yang sama dengan sekitar 341.502 baris data *dummy*. Parameter yang dianalisis meliputi *response time*, *throughput*, *error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem. Pemantauan sumber daya dilakukan menggunakan Glances melalui SSH.

Hasil penelitian menunjukkan bahwa server lokal mulai mengalami penurunan performa pada 50 pengguna, sedangkan VPS UNUGHA mulai menunjukkan penurunan pada 100 pengguna. Pada skenario 500 pengguna, seluruh endpoint pada kedua lingkungan menghasilkan *response time* lebih dari 10 detik dan *throughput* cenderung stagnan atau menurun. Endpoint `/api/dashboard/students` menghasilkan *response time* tertinggi, yaitu 15.894,4 ms pada server lokal dan 10.714,6 ms pada VPS UNUGHA. Seluruh permintaan tetap berhasil diproses dengan *error rate* sebesar 0%, sedangkan penggunaan memori relatif stabil. Penggunaan CPU proses Node.js mencapai sekitar 101% pada server lokal dan 110–112% pada VPS UNUGHA, sementara CPU global VPS UNUGHA tetap berada di bawah 6%. Temuan tersebut mengindikasikan dugaan keterbatasan pada pemrosesan aplikasi dan akses data, bukan kehabisan kapasitas CPU VPS secara keseluruhan. Rekomendasi yang disusun meliputi *query refactoring*, penyesuaian pemanfaatan prosesor, penerapan PM2 *cluster mode* pada VPS UNUGHA, dan migrasi cache menuju *external in-memory datastore*. Rekomendasi tersebut belum diimplementasikan maupun diuji ulang dalam penelitian ini.

Kata kunci: Apache JMeter, Dashboard Admin, evaluasi performa, *bottleneck*, Node.js, VPS.

ABSTRACT

AHMAD FAJRUL 'ULUM. Performance Analysis and Evaluation of Outcome-Based Education (OBE) CPL System Admin *Dashboard* Using Apache JMeter. Supervised by Tri Anggoro, M.Kom. and Riski Aspriyani, M.Pd.

An *Outcome-Based Education* (OBE)-based Graduate Learning Outcomes assessment system requires an Admin Dashboard capable of presenting aggregated data rapidly and reliably. This study aimed to analyze the performance of three main Admin Dashboard endpoints, namely /api/dashboard/stats, /api/dashboard/dosen, and /api/dashboard/students, identify potential bottlenecks, and formulate technical improvement recommendations. Performance testing was conducted using Apache JMeter with virtual user scenarios of 1, 10, 20, 50, 100, 200, and 500 on a local server and the UNUGHA VPS. Both environments used the same application and MariaDB database containing approximately 341,502 rows of dummy data. The analyzed parameters included response time, throughput, error rate, CPU utilization, memory utilization, and system stability. Resource utilization was monitored using Glances through SSH.

The results showed that the local server began to experience performance degradation at 50 users, whereas the UNUGHA VPS began to decline at 100 users. At 500 users, all endpoints in both environments produced response times exceeding 10 seconds, while throughput tended to stagnate or decrease. The /api/dashboard/students endpoint produced the highest response time, reaching 15,894.4 ms on the local server and 10,714.6 ms on the UNUGHA VPS. All requests were successfully processed with an error rate of 0%, while memory utilization remained relatively stable. Node.js CPU utilization reached approximately 101% on the local server and 110–112% on the UNUGHA VPS, while the overall CPU utilization of the VPS remained below 6%. These findings indicate potential limitations in application-level processing and data access rather than exhaustion of the VPS's overall CPU capacity. The proposed recommendations include query refactoring, processor utilization adjustment, PM2 cluster mode implementation on the UNUGHA VPS, and cache migration to an external in-memory datastore. These recommendations were not implemented or retested in this study.

Keywords: Apache JMeter, Admin Dashboard, performance evaluation, bottleneck, Node.js, VPS.

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI	iii
LEMBAR PENGESAHAN	iv
HALAMAN NOTA KONSULTAN.....	v
HALAMAN NOTA PEMBIMBING.....	vi
HALAMAN MOTO	vii
HALAMAN PERSEMBAHAN	viii
KATA PENGANTAR	ix
ABSTRAK.....	x
ABSTRACT.....	xi
DAFTAR ISI.....	xii
DAFTAR GAMBAR	xiv
DAFTAR TABEL.....	xv
BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Rumusan Masalah.....	3
C. Batasan Masalah	4
D. Tujuan Penelitian	4
E. Manfaat Penelitian	5
BAB II TINJAUAN PUSTAKA	7
A. Penelitian Terkait	7
B. Landasan Teori.....	15
BAB III METODOLOGI.....	25
A. Waktu dan Tempat Penelitian.....	25
B. Alat dan Bahan.....	26
C. Prosedur Penelitian	28
D. Analisis Data.....	58
E. Jadwal Penelitian	60
BAB IV HASIL DAN PEMBAHASAN.....	62
A. Hasil.....	62
B. Pembahasan.....	67
BAB V KESIMPULAN.....	84

A.	Kesimpulan	84
B.	Saran	85
	DAFTAR PUSTAKA	86
	LAMPIRAN	94

DAFTAR GAMBAR

Gambar 1 Mindmap Penelitian	14
Gambar 2 Lingkungan Pengujian pada Server Lokal dan VPS UNUGHA.....	25
Gambar 3 Diagram Alir Prosedur Penelitian	29
Gambar 4 Tampilan Dashboard Admin Sistem CPL.....	32
Gambar 5 Activity Diagram Pengujian Performa.....	39
Gambar 6 Sequence Diagram Pengujian Performa.....	48
Gambar 7 Jadwal Penelitian.....	61

DAFTAR TABEL

Tabel 1 Rangkuman Penelitian Terdahulu	7
Tabel 2 Bentuk Data.....	17
Tabel 3 Alat Penelitian.....	26
Tabel 4 Bahan Penelitian	28
Tabel 5 Skenario Pengujian	30
Tabel 6 <i>Endpoint</i>	32
Tabel 7 Ringkasan Data <i>Dummy</i> Sistem CPL	33
Tabel 8 Karakteristik Data yang Diproses oleh Endpoint Dashboard Admin	34
Tabel 9 Keterlibatan Tabel Database pada Endpoint Dashboard Admin	35
Tabel 10 Kelompok Data Agregat dan Kontrol <i>Dashboard</i> Admin.....	36
Tabel 11 Spesifikasi Lingkungan Pengujian.....	41
Tabel 12 Konfigurasi Apache JMeter	42
Tabel 13 Konfigurasi Thread Group Apache JMeter.....	44
Tabel 14 Konfigurasi HTTP Request Apache JMeter	45
Tabel 15 Konfigurasi Header dan Ekstraksi Token JWT	45
Tabel 16 Konfigurasi Uniform Random Timer	46
Tabel 17 Metrik Performa yang Dikumpulkan dari Pengujian Apache JMeter....	47
Tabel 18 Skenario Pengujian Performa.....	49
Tabel 19 Parameter Analisis Performa	52
Tabel 20 Kriteria Operasional Evaluasi Performa	53
Tabel 21 Kriteria Identifikasi Dugaan <i>Bottleneck</i> Berdasarkan Hasil Pengujian .	55
Tabel 22 Karakteristik Teknis <i>Endpoint</i> yang Dianalisis	56
Tabel 23 Perbandingan Hasil Pengujian Endpoint /api/dashboard/stats pada Server Lokal dan VPS UNUGHA	62
Tabel 24 Perbandingan Hasil Pengujian Endpoint /api/dashboard/dosen pada Server Lokal dan VPS UNUGHA.....	63
Tabel 25 Perbandingan Hasil Pengujian Endpoint /api/dashboard/students pada Server Lokal dan VPS UNUGHA.....	65
Tabel 26 Perbandingan Hasil Pemantauan CPU dan Memori pada Server Lokal dan VPS UNUGHA	66
Tabel 27 Ringkasan Hasil Pengujian Response time.....	68
Tabel 28 Ringkasan Hasil Pengujian Throughput	70
Tabel 29 Ringkasan Hasil Pengujian Error rate pada Server Lokal dan VPS UNUGHA	73
Tabel 30 Ringkasan Penggunaan CPU dan Memori Server Lokal dan VPS UNUGHA	74
Tabel 31 Ringkasan Kestabilan Server Lokal dan VPS UNUGHA	76
Tabel 32 Identifikasi Dugaan Bottleneck pada Server Lokal dan VPS UNUGHA	77
Tabel 33 Rekomendasi Perbaikan Berdasarkan Hasil Analisis Parameter Performa dan Identifikasi Dugaan Bottleneck	80

BAB I PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi informasi telah mendorong transformasi digital pada pendidikan tinggi [1]. Perguruan tinggi membutuhkan sistem informasi untuk mengelola data akademik secara terstruktur, mulai dari data mahasiswa, kurikulum, mata kuliah, penilaian, hingga evaluasi capaian pembelajaran [2]. Dalam konteks tersebut, *Outcome Based Education* atau OBE digunakan karena menekankan ketercapaian luaran pembelajaran sebagai dasar perencanaan dan evaluasi pendidikan [3]. Pada tingkat program studi, luaran tersebut direpresentasikan dalam bentuk Capaian Pembelajaran Lulusan atau CPL yang digunakan untuk menilai ketercapaian kompetensi lulusan [4].

Sistem Penilaian Capaian Pembelajaran Lulusan berbasis OBE merupakan sistem informasi akademik yang membantu pengelolaan data CPL, CPMK, mata kuliah, nilai, dosen, mahasiswa, dan relasi akademik lainnya [5]. Dalam sistem seperti ini, *Dashboard* berperan menyajikan informasi akademik dalam bentuk ringkasan, grafik, tren, dan indikator *monitoring* [6]. Pada objek penelitian, *Dashboard* admin menampilkan data agregat seperti statistik CPL, jumlah mata kuliah aktif, jumlah mahasiswa, rata-rata CPL global, grafik CPL, tren semester, kelengkapan data, *alerts*, *insights*, analisis dosen, evaluasi mahasiswa terbatas, serta *Filter Dashboard*.

Dashboard admin tidak mengambil seluruh data mentah dari *Database* secara langsung, tetapi menampilkan hasil pengolahan *Backend* dalam bentuk ringkasan dan agregasi. Pemisahan antara data mentah *Database* dan data agregat *Dashboard* penting agar evaluasi performa tidak keliru, karena *Dashboard* berbasis data agregat memiliki karakteristik pemrosesan yang berbeda dengan halaman web sederhana [7]. Proses pemuatan *Dashboard* melibatkan pembacaan parameter *Filter*, akses data melalui Prisma ORM, *Query* ke MariaDB, perhitungan agregat, penyusunan *Response*, dan pengiriman data ke *Frontend*. Proses tersebut dapat memengaruhi *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem ketika *Endpoint* diakses secara simultan. Selain itu, peningkatan beban pemrosesan juga dapat tercermin dari perubahan penggunaan sumber daya *Server*, seperti CPU dan memori, yang diamati sebagai indikator pendukung selama proses pengujian [8].

Sistem yang menjadi objek penelitian ini adalah Sistem Penilaian CPL berbasis OBE yang dikembangkan di lingkungan Universitas Nahdlatul Ulama Al Ghazali Cilacap. Sistem menggunakan React.js dan TypeScript pada *Frontend*, Node.js dengan Express.js pada *Backend*, Prisma ORM, serta MariaDB sebagai *Database*. Data yang digunakan berupa data *dummy* karena sistem belum digunakan secara

nyata dalam proses akademik. Data *dummy* digunakan sebagai konteks skala data sistem, bukan untuk menilai validitas akademik perhitungan CPL.

Permasalahan performa perlu dikaji karena *Dashboard* admin berpotensi diakses secara simultan atau diproses berulang ketika pengguna menggunakan *Filter* tertentu. Apabila *Endpoint Dashboard* membutuhkan waktu lama untuk menghasilkan *Response*, proses *monitoring* akademik dapat menjadi kurang efektif dan pengambilan keputusan berbasis data dapat terhambat [9]. Oleh karena itu, diperlukan evaluasi performa untuk mengetahui kemampuan *Endpoint Dashboard* Admin dalam menangani *Request* pengguna secara simultan. Evaluasi dilakukan melalui pengukuran metrik performa *Endpoint* dan didukung oleh pengamatan penggunaan CPU serta memori *Server* untuk membantu menginterpretasikan perubahan performa yang terjadi selama pengujian.

Evaluasi performa sistem berbasis web dilakukan melalui pengujian performa pada berbagai tingkat beban pengguna. Salah satu perangkat lunak yang banyak digunakan untuk tujuan tersebut adalah Apache JMeter [10]. Pengujian performa dilakukan untuk mengamati kemampuan sistem pada beban pengguna bertahap, sedangkan *Stress testing* digunakan untuk mengetahui perilaku sistem pada beban tinggi [11]. Dalam penelitian ini, Apache JMeter digunakan untuk menyimulasikan pengguna virtual dan mengirim *Request* HTTP/API ke *Endpoint Dashboard* admin [12].

Pengujian dilakukan pada dua lingkungan, yaitu *server* lokal dan *Virtual Private Server* (VPS) milik Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap yang diakses melalui jaringan internet menggunakan domain obe.unugha.ac.id dan selanjutnya disebut VPS UNUGHA, menggunakan *endpoint*, data, skenario jumlah pengguna, dan parameter pengukuran yang sama. Pengujian pada kedua lingkungan diperlukan untuk mengetahui perubahan performa sistem pada konfigurasi *server* yang berbeda serta membandingkan kemampuan masing-masing lingkungan dalam menangani beban yang sama. Perbandingan tersebut menggambarkan performa sistem pada dua lingkungan aktual dan tidak dimaksudkan sebagai pengujian perangkat keras yang dikendalikan secara eksperimental.

Pengujian difokuskan pada performa proses *request-response endpoint* yang menghasilkan data agregat dari *backend*, bukan pada proses *rendering* React, tampilan *frontend*, atau keseluruhan fitur sistem. Parameter utama yang digunakan meliputi *response time*, *throughput*, dan *error rate*. Pengujian juga didukung dengan pemantauan penggunaan CPU dan memori *server* menggunakan Glances melalui *Secure Shell* atau SSH untuk membantu menginterpretasikan perubahan performa pada setiap skenario.

Penelitian sebelumnya telah membahas implementasi OBE, *dashboard* akademik, dan pengujian performa aplikasi web. Namun, kajian yang secara khusus mengevaluasi performa *endpoint Dashboard* Admin Sistem CPL berbasis OBE sebagai penghasil data agregat pada dua lingkungan *server* masih terbatas. Penelitian mengenai OBE umumnya berfokus pada implementasi kurikulum, sistem pemantauan capaian pembelajaran, atau penyajian *dashboard*. Sementara itu, penelitian *performance testing* umumnya menguji aplikasi web secara keseluruhan tanpa menyoroti karakteristik *dashboard* akademik berbasis data agregat [13].

Berdasarkan keterbatasan tersebut, kontribusi penelitian ini terletak pada evaluasi performa *endpoint API Dashboard* Admin Sistem CPL berbasis OBE pada *server* lokal dan VPS UNUGHA. Evaluasi dilakukan melalui pengukuran *response time*, *throughput*, dan *error rate*, serta didukung pemantauan penggunaan CPU dan memori. Hasil pengujian pada masing-masing *server* dianalisis dan dibandingkan pada skenario beban yang sama untuk mengetahui titik awal penurunan performa dan kestabilan sistem.

Berdasarkan hasil analisis parameter performa, penelitian selanjutnya mengidentifikasi dugaan *bottleneck* yang berkaitan dengan pemrosesan *endpoint*, proses Node.js, dan kapasitas lingkungan *server*. Identifikasi tersebut menjadi dasar dalam menyusun rekomendasi perbaikan teknis berupa optimalisasi akses data, penyesuaian pemanfaatan prosesor berdasarkan lingkungan *server*, dan pengembangan mekanisme *caching*. Rekomendasi tersebut tidak diimplementasikan maupun diuji ulang dalam penelitian ini.

Dengan demikian, penelitian ini penting dilakukan untuk memberikan gambaran mengenai kemampuan *Dashboard* Admin Sistem CPL berbasis OBE dalam menangani permintaan pengguna secara simultan pada dua lingkungan *server*. Hasil penelitian diharapkan dapat menunjukkan pola perubahan performa, perbedaan kemampuan *server* lokal dan VPS UNUGHA, titik awal penurunan performa, dugaan *bottleneck*, serta rekomendasi teknis untuk pengembangan sistem selanjutnya.

B. Rumusan Masalah

Berdasarkan permasalahan yang telah diuraikan pada latar belakang, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana performa *endpoint Dashboard* Admin Sistem CPL berbasis OBE pada *server* lokal dan VPS UNUGHA berdasarkan parameter *response time*, *throughput*, *error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem pada berbagai tingkat beban pengguna?

2. Pada skenario jumlah pengguna berapa *endpoint Dashboard* Admin mulai menunjukkan penurunan performa pada *server* lokal dan VPS UNUGHA, serta bagaimana perbandingan performa kedua lingkungan tersebut?
3. Faktor teknis apa yang diduga berkontribusi terhadap munculnya *bottleneck* berdasarkan hasil analisis parameter performa, serta rekomendasi perbaikan apa yang dapat disusun untuk pengembangan sistem selanjutnya?

C. Batasan Masalah

Agar penelitian dapat dilakukan secara lebih terarah dan sesuai dengan ruang lingkup skripsi, maka penelitian ini memiliki beberapa batasan sebagai berikut:

1. Penelitian difokuskan pada evaluasi performa tiga *endpoint API Dashboard* Admin Sistem CPL berbasis OBE, yaitu `/api/dashboard/stats`, `/api/dashboard/dosen`, dan `/api/dashboard/students`.
2. Pengujian menggunakan data *dummy* yang disusun untuk merepresentasikan struktur, relasi, volume, dan karakteristik data pada Sistem CPL.
3. Pengujian dilakukan menggunakan Apache JMeter dengan skenario beban sebesar 1, 10, 20, 50, 100, 200, dan 500 pengguna virtual.
4. Parameter utama yang dianalisis meliputi *response time*, *throughput*, dan *error rate*, sedangkan penggunaan CPU, penggunaan memori, dan kestabilan sistem digunakan sebagai parameter pendukung.
5. Pengujian dilakukan pada server lokal dan VPS UNUGHA menggunakan *endpoint*, data, parameter, serta skenario beban yang sama. Server lokal diakses melalui jaringan lokal, sedangkan VPS UNUGHA diakses oleh laptop penguji melalui jaringan internet menggunakan domain `obe.unugha.ac.id`. Nilai *response time* pada VPS UNUGHA merupakan waktu respons *end-to-end* dan penelitian tidak memisahkan latensi jaringan dari waktu pemrosesan server.
6. Penelitian difokuskan pada proses *request-response endpoint backend* dan tidak mencakup proses *rendering frontend*, pengujian antarmuka pengguna, *code profiling*, penelusuran kueri secara *real-time*, implementasi rekomendasi perbaikan, maupun pengujian ulang setelah optimasi diterapkan.

D. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dijelaskan, maka tujuan penelitian ini adalah sebagai berikut:

1. Menganalisis performa *endpoint Dashboard* Admin Sistem CPL berbasis OBE pada *server* lokal dan VPS UNUGHA berdasarkan parameter *response*

time, throughput, error rate, penggunaan CPU, penggunaan memori, dan kestabilan sistem pada berbagai tingkat beban pengguna.

2. Menentukan skenario jumlah pengguna yang menyebabkan *endpoint Dashboard Admin* mulai mengalami penurunan performa pada masing-masing *server* serta membandingkan performa *server* lokal dan VPS UNUGHA pada skenario yang sama.
3. Mengidentifikasi dugaan *bottleneck* berdasarkan hasil analisis parameter performa dan menyusun rekomendasi teknis untuk meningkatkan performa sistem pada pengembangan selanjutnya.

E. Manfaat Penelitian

1. Manfaat Teoritis

Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan ilmu pengetahuan, khususnya pada bidang rekayasa perangkat lunak dan pengujian performa sistem. Manfaat teoretis penelitian ini adalah sebagai berikut:

- a. Menambah referensi mengenai evaluasi performa aplikasi web berbasis *Outcome-Based Education*.
- b. Memberikan contoh penerapan *load testing* dan *stress testing* pada *endpoint API* sistem akademik berbasis web.
- c. Memperkuat pemahaman mengenai hubungan antara peningkatan beban pengguna, perubahan parameter performa, dan penggunaan sumber daya *server*.
- d. Memberikan gambaran mengenai perbandingan perilaku sistem yang sama ketika dijalankan pada dua lingkungan *server* dengan kapasitas yang berbeda.
- e. Menjadi acuan bagi penelitian selanjutnya yang membahas identifikasi *bottleneck* dan optimasi performa *dashboard* akademik berbasis data agregat.

2. Manfaat Praktis

Penelitian ini diharapkan memberikan manfaat praktis bagi pihak yang berkaitan dengan pengembangan dan penggunaan Sistem CPL. Manfaat tersebut adalah sebagai berikut:

- a. Bagi peneliti, penelitian ini dapat meningkatkan pemahaman mengenai pengujian, analisis, dan perbandingan performa sistem menggunakan Apache JMeter.
- b. Bagi pengembang sistem, penelitian ini memberikan informasi mengenai pola penurunan performa, *endpoint* yang memiliki beban

pemrosesan lebih tinggi, dan bagian sistem yang diduga mengalami *bottleneck*.

- c. Bagi program studi atau instansi, penelitian ini memberikan gambaran awal mengenai kemampuan *Dashboard* Admin ketika dijalankan pada *server* lokal dan VPS UNUGHA sebelum sistem digunakan secara lebih luas.
- d. Bagi pengguna sistem, hasil penelitian dapat mendukung peningkatan kualitas layanan *Dashboard* Admin pada pengembangan berikutnya.
- e. Bagi pengembangan sistem selanjutnya, rekomendasi teknis yang dihasilkan dapat digunakan sebagai arahan awal dalam melakukan *query refactoring*, penggunaan indeks basis data, migrasi menuju *external caching*, penyesuaian jumlah proses aplikasi, penerapan PM2 *cluster mode*, dan peningkatan kapasitas *server* sesuai kebutuhan.

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Penelitian terkait merupakan bagian yang menjelaskan perkembangan penelitian sebelumnya yang relevan dengan topik penelitian. Pada penelitian ini, kajian penelitian terkait diarahkan pada tiga kelompok utama, yaitu sistem berbasis *Outcome Based Education* (OBE), *Dashboard* akademik atau *Learning analytics*, serta pengujian performa aplikasi web menggunakan metode *Load testing* dan *Stress testing*. Pengelompokan tersebut digunakan agar posisi penelitian dapat terlihat secara jelas dalam konteks evaluasi performa *Endpoint Dashboard* admin Sistem CPL berbasis OBE.

Penelitian terdahulu digunakan untuk mengetahui ruang lingkup kajian yang sudah dilakukan, metode yang digunakan, hasil yang diperoleh, serta celah penelitian yang masih dapat dikembangkan. Beberapa penelitian sebelumnya telah membahas pengembangan sistem OBE, sistem evaluasi capaian pembelajaran, *Dashboard* akademik, dan pengujian performa aplikasi web. Namun, penelitian yang secara khusus mengevaluasi performa *Endpoint Dashboard* admin Sistem CPL berbasis OBE yang memuat data agregat dari *Backend* masih terbatas. Oleh karena itu, rangkuman penelitian terdahulu disajikan pada Tabel 1 sebagai dasar untuk menunjukkan keterkaitan, perbedaan, dan posisi penelitian ini.

Tabel 1 Rangkuman Penelitian Terdahulu

No	Penulis (Tahun)	Keterangan
(1)	(2)	(3)
1	Othman et al. (2024)[14]	Judul: The Development an <i>Outcome-Based Education</i> (OBE) System for Measuring Student Programme Attainment. Metode: Pengembangan sistem OBE menggunakan pendekatan Rapid Application Development (RAD). Hasil: Penelitian menghasilkan sistem CO-AIMS yang digunakan untuk membantu pemetaan course outcome ke program outcome, memantau capaian mahasiswa, dan mendukung pengelolaan assessment dalam konteks OBE. Keterkaitan/Perbedaan: Relevan karena sama-sama membahas sistem berbasis OBE dan capaian pembelajaran. Perbedaannya, penelitian tersebut berfokus pada

No	Penulis (Tahun)	Keterangan
		pengembangan sistem pengukuran attainment, sedangkan penelitian ini berfokus pada evaluasi performa <i>Endpoint Dashboard</i> admin Sistem CPL berbasis OBE.
2	Hakim et al. (2025)[15]	Judul: Bibliometric Analysis of <i>Outcome-Based Education</i> (OBE) in Higher Education: Trends, Themes, and Future Directions. Metode: Analisis bibliometrik menggunakan data Scopus, VOSviewer, dan Bibliometrix R. Hasil: Penelitian menunjukkan perkembangan tren, tema, jaringan penulis, kata kunci, dan arah penelitian OBE di pendidikan tinggi. Keterkaitan/Perbedaan: Relevan untuk menunjukkan perkembangan kajian OBE, tetapi tidak membahas implementasi sistem <i>Dashboard</i> maupun evaluasi performa aplikasi.
3	Lamis dan Pisuena (2026)[16]	Judul: Web-based Program Outcomes Assessment and Evaluation System with Data Analytics. Metode: Penelitian pengembangan deskriptif dengan pendekatan Agile SDLC, evaluasi kualitas sistem, dan evaluasi usability. Hasil: Penelitian menghasilkan sistem berbasis web untuk mengelola, mengevaluasi, dan memvisualisasikan data program outcomes melalui analitik data. Keterkaitan/Perbedaan: Relevan karena membahas sistem evaluasi capaian program berbasis web dan data analytics. Perbedaannya, penelitian tersebut mengevaluasi kualitas dan usability sistem, sedangkan penelitian ini mengevaluasi performa <i>Endpoint Dashboard</i> berdasarkan hasil <i>Load testing</i> dan <i>Stress testing</i>.
4	Siang et al. (2024)[17]	Judul: Reinforcing Learning Management System With OBE-Based <i>Learning analytics</i>

No	Penulis (Tahun)	Keterangan
		and RIASEC Integration. Metode: Kajian dan pengembangan kerangka LMS yang mengintegrasikan OBE-based Learning analytics dan model RIASEC. Hasil: Penelitian menunjukkan bahwa integrasi OBE Learning analytics dan RIASEC dapat membantu pemantauan capaian pembelajaran, interpretasi data peserta didik, dan penyusunan laporan berbasis Learning analytics. Keterkaitan/Perbedaan: Relevan karena membahas OBE, Learning analytics, dan sistem pembelajaran. Perbedaannya, penelitian tersebut tidak menguji performa <i>Endpoint</i> sistem dari sisi <i>Response time</i>, <i>Throughput</i>, <i>Error rate</i>, dan kestabilan sistem.
5	Asmaranto et al. (2025)[18]	Judul: <i>A Dashboard System for Enhancing Learning Outcome Monitoring and Curriculum Management Under the Outcome-Based Education Framework.</i> Metode: Berdasarkan judul dan metadata publikasi, penelitian ini membahas sistem <i>Dashboard</i> dalam konteks OBE untuk monitoring learning outcome dan manajemen kurikulum. Hasil: Berdasarkan metadata yang dapat diakses, penelitian tersebut berfokus pada <i>Dashboard</i> untuk mendukung monitoring learning outcome dan curriculum management dalam kerangka OBE. Keterkaitan/Perbedaan: Relevan karena sama-sama membahas <i>Dashboard</i> dalam konteks OBE. Perbedaannya, penelitian ini tidak berfokus pada pengembangan <i>Dashboard</i> baru dalam tahapan skripsi, tetapi mengevaluasi performa <i>Dashboard</i> admin Sistem CPL berbasis OBE yang telah dikembangkan sebelumnya oleh peneliti pada kegiatan Praktik Kerja Lapangan.

No	Penulis (Tahun)	Keterangan
6	Handayani et al. (2024)[19]	Judul: Sistem <i>Monitoring</i> dan Evaluasi Proses Belajar Mengajar Berbasis <i>Outcome Based Education</i> di Fakultas Teknik Universitas Diponegoro. Metode: Studi kasus dengan pendekatan kualitatif, studi dokumentasi, dan wawancara untuk membangun indikator monitoring dan evaluasi berbasis OBE. Hasil: Penelitian menghasilkan indikator dan rubrik monitoring evaluasi proses belajar mengajar berbasis OBE. Keterkaitan/Perbedaan: Relevan karena membahas monitoring dan evaluasi berbasis OBE. Perbedaannya, penelitian tersebut tidak membahas performa <i>Dashboard</i> atau pengujian beban menggunakan Apache JMeter.
7	Raweyai dan Widiyari (2024)[12]	Judul: <i>Load testing</i> Method Supported by Apache JMeter at XYZ University. Metode: Pengujian performa website akademik menggunakan metode Load testing dengan Apache JMeter melalui skenario penggunaan normal, puncak, dan stres. Hasil: Website akademik yang diuji masih memenuhi kriteria pengujian, dengan <i>Response time</i> GET dan POST yang berada di bawah batas maksimal pada beberapa skenario. Keterkaitan/Perbedaan: Relevan karena menggunakan Apache JMeter untuk mengevaluasi performa website akademik. Perbedaannya, penelitian tersebut belum membahas <i>Dashboard</i> admin berbasis OBE yang memproses data agregat CPL.
8	Annisa et al. (2026)[20]	Judul: Performance Evaluation of Web Applications Using JMeter <i>Load testing</i> for Server Capacity and Response Efficiency. Metode: Load testing menggunakan Apache JMeter dengan skenario 50 sampai 5000 pengguna virtual selama satu jam. Hasil: Website ICOMTA PPNS stabil

No	Penulis (Tahun)	Keterangan
		hingga 200 pengguna, tetapi setelah beban melebihi 300 pengguna <i>Response time</i> meningkat tajam, error mulai muncul, dan kualitas layanan menurun pada beban tinggi. Keterkaitan/Perbedaan: Relevan karena menggunakan parameter <i>Response time</i>, <i>Error rate</i>, <i>Throughput</i>, dan <i>bandwidth usage</i>. Perbedaannya, objek penelitian tersebut adalah website konferensi/jurnal, sedangkan penelitian ini berfokus pada <i>Endpoint Dashboard</i> admin Sistem CPL berbasis OBE.
9	Garjitno et al. (2025)[21]	Judul: Analisis dan Optimasi Kinerja Aplikasi POTIO Menggunakan <i>Load testing</i> JMeter. Metode: Load testing menggunakan Apache JMeter pada enam <i>Endpoint</i> utama dengan skenario 50, 100, dan 150 pengguna serentak, dilanjutkan eksperimen peningkatan jaringan serta optimasi <i>Database</i> dan kode. Hasil: Aplikasi POTIO mulai mengalami penurunan performa pada 150 pengguna, ditandai dengan peningkatan <i>Response time</i>, penurunan <i>Throughput</i>, dan munculnya error. Optimasi <i>Database</i> dan kode lebih konsisten memperbaiki performa dibanding hanya peningkatan jaringan. Keterkaitan/Perbedaan: Relevan karena membahas Load testing, <i>Bottleneck</i>, dan rekomendasi/perbaikan performa. Perbedaannya, penelitian ini tidak melakukan implementasi optimasi penuh, tetapi menyusun rekomendasi terbatas berdasarkan hasil pengujian dan dugaan teknis.
10	Paulsen dan Lindsay (2024)[22]	Judul: <i>Learning analytics</i> Dashboards Are Increasingly Becoming About Learning and Not Just Analytics: A Systematic Review. Metode: Systematic literature review terhadap <i>Learning analytics Dashboard</i> di pendidikan tinggi.

No	Penulis (Tahun)	Keterangan
		Hasil: Penelitian membahas tema-tema dalam desain dan implementasi <i>Dashboard Learning analytics</i>, serta menunjukkan pergeseran <i>Dashboard</i> dari sekadar analitik menuju dukungan pembelajaran yang lebih pedagogis. Keterkaitan/Perbedaan: Relevan karena membahas <i>Dashboard Learning analytics</i> di pendidikan tinggi. Perbedaannya, penelitian tersebut tidak membahas performa teknis <i>Endpoint Dashboard</i> atau pengujian beban <i>Server</i>.
11	Hernández-Campos et al. (2025)[23]	Judul: Learning Outcomes Evaluation Through <i>Learning analytics</i> Systems in Higher Education: A Systematic Literature Review. Metode: Systematic literature review terhadap studi yang mengevaluasi learning outcomes melalui Learning analytics systems di pendidikan tinggi. Hasil: Penelitian mengidentifikasi faktor-faktor yang berkaitan dengan learning outcomes, ruang lingkup, modalitas pembelajaran, dan latar teori dalam sistem Learning analytics. Keterkaitan/Perbedaan: Relevan karena membahas evaluasi learning outcomes berbasis Learning analytics. Perbedaannya, penelitian tersebut tidak membahas evaluasi performa <i>Endpoint Dashboard</i> atau pengujian <i>Request-Response</i> menggunakan Apache JMeter.

Sumber : Hasil olahan peneliti, 2026.

Berdasarkan Tabel 1, penelitian terdahulu dapat dikelompokkan ke dalam tiga fokus utama. Kelompok pertama berkaitan dengan sistem berbasis OBE dan evaluasi capaian pembelajaran. Penelitian pada kelompok ini menunjukkan bahwa sistem informasi dapat digunakan untuk membantu pengukuran capaian program, pemetaan capaian pembelajaran, serta evaluasi proses akademik secara lebih terstruktur. Namun, fokus penelitian tersebut umumnya masih berada pada

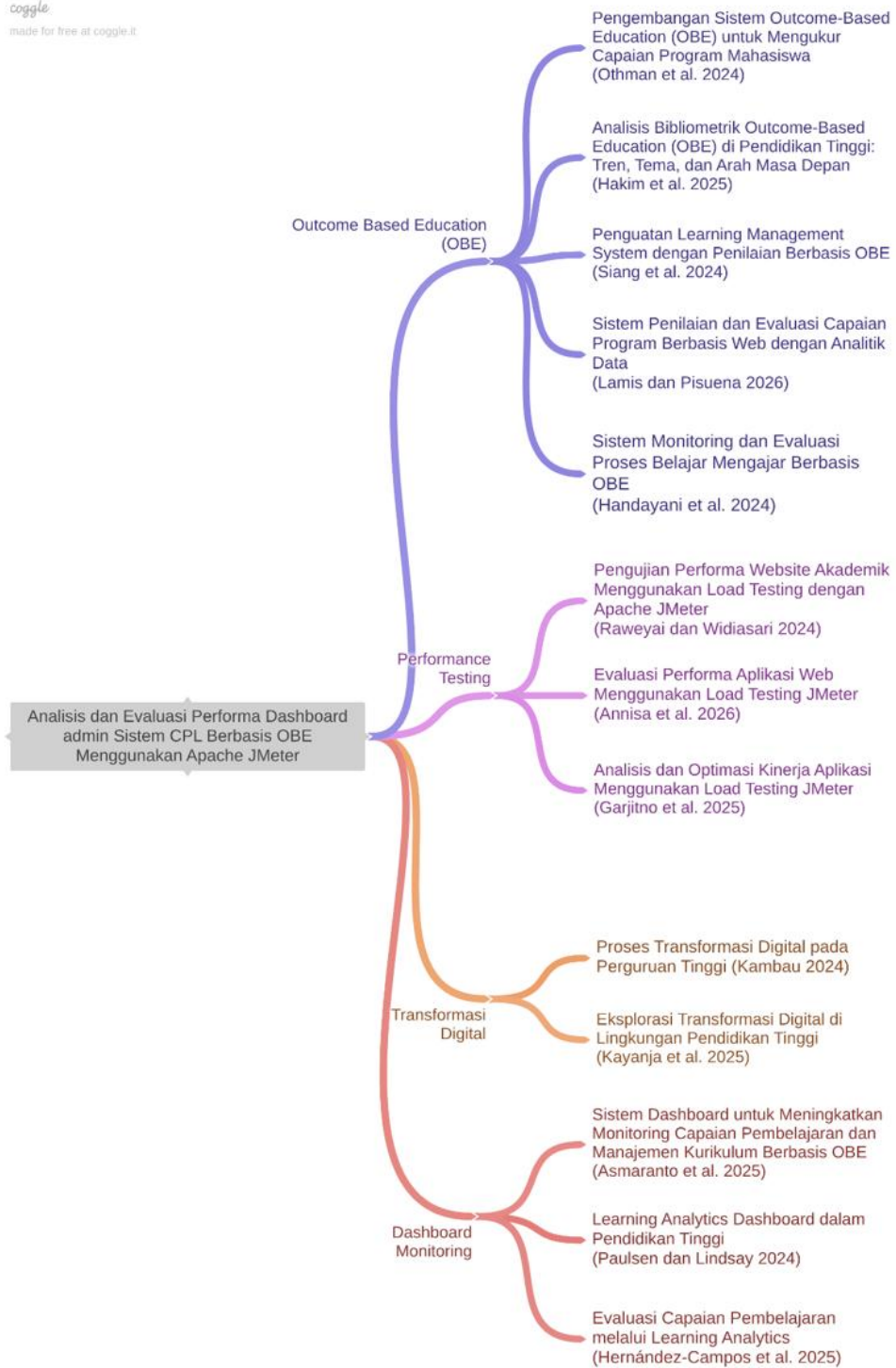
pengembangan sistem, evaluasi capaian pembelajaran, atau penyusunan indikator *monitoring*, belum pada pengujian performa *Endpoint* sistem ketika menerima *Request* pengguna secara simultan.

Kelompok kedua berkaitan dengan *Dashboard* akademik dan *Learning analytics*. Penelitian dalam kelompok ini menunjukkan bahwa *Dashboard* dapat membantu penyajian informasi akademik melalui ringkasan data, visualisasi, *monitoring* capaian, dan dukungan pengambilan keputusan. Akan tetapi, penelitian tersebut lebih banyak membahas fungsi *Dashboard*, desain informasi, dan peran analitik pembelajaran. Aspek teknis seperti *Response time*, *Throughput*, *Error rate*, dan kestabilan *Endpoint Dashboard* ketika memproses data agregat belum menjadi fokus utama.

Kelompok ketiga berkaitan dengan pengujian performa aplikasi web menggunakan Apache JMeter. Penelitian pada kelompok ini menunjukkan bahwa *Load testing* dan *Stress testing* dapat digunakan untuk mengevaluasi kemampuan sistem dalam menangani beban pengguna, mengukur *Response time*, *Throughput*, *Error rate*, serta mengidentifikasi indikasi penurunan performa. Meskipun demikian, objek penelitian pada kajian tersebut umumnya berupa website akademik, aplikasi web umum, atau aplikasi layanan tertentu, belum secara spesifik membahas *Dashboard* admin Sistem CPL berbasis OBE yang memproses data agregat akademik.

Dengan demikian, posisi penelitian ini berada pada irisan antara *Dashboard* berbasis OBE dan pengujian performa aplikasi web. Penelitian ini tidak berfokus pada pengembangan *Dashboard* baru, karena *Dashboard* admin Sistem CPL telah dikembangkan sebelumnya oleh peneliti pada kegiatan Praktik Kerja Lapangan. Fokus penelitian ini adalah mengevaluasi performa *Endpoint Dashboard* admin menggunakan Apache JMeter, menganalisis perubahan *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem, mengidentifikasi dugaan *Bottleneck*, serta menyusun rekomendasi perbaikan secara terbatas berdasarkan pola hasil pengujian. Pendekatan tersebut menjadi pembeda utama penelitian ini dibandingkan penelitian terdahulu.

Untuk memperjelas hubungan antarpelitian terdahulu, kajian penelitian terkait disajikan dalam bentuk mind map. Mind map ini memetakan beberapa kelompok kajian yang berhubungan dengan topik penelitian, yaitu transformasi digital, *Outcome Based Education* (OBE), *Dashboard monitoring*, dan *Performance testing*. Pemetaan tersebut digunakan untuk menunjukkan ruang lingkup penelitian sebelumnya serta keterkaitan antarbidang yang mendukung penelitian ini.



Gambar 1 Mindmap Penelitian

Gambar 1 menunjukkan bahwa penelitian terdahulu tersebar pada empat kelompok utama, yaitu transformasi digital, OBE, *Dashboard monitoring*, dan *Performance testing*. Pemetaan tersebut memperlihatkan bahwa belum banyak penelitian yang secara khusus menghubungkan *Dashboard monitoring* berbasis OBE dengan evaluasi performa *Endpoint* menggunakan Apache JMeter. Hal ini menjadi dasar bagi penelitian ini untuk mengevaluasi performa *Dashboard* admin Sistem CPL berbasis OBE berdasarkan hasil pengujian beban.

B. Landasan Teori

1. Outcome Based Education (OBE)

Outcome Based Education (OBE) merupakan pendekatan pendidikan yang menempatkan hasil pembelajaran sebagai dasar dalam perencanaan, pelaksanaan, dan evaluasi pendidikan [24]. Pendekatan ini menilai proses pembelajaran berdasarkan ketercapaian kompetensi yang harus dimiliki mahasiswa setelah menyelesaikan program pendidikan [25]. OBE juga menekankan keselarasan antara capaian pembelajaran, proses pembelajaran, metode penilaian, dan evaluasi hasil belajar agar seluruh kegiatan akademik mengarah pada capaian yang telah ditetapkan [26].

Dalam konteks pendidikan tinggi, OBE berkaitan erat dengan pengelolaan data akademik seperti kurikulum, mata kuliah, capaian pembelajaran, indikator penilaian, dan nilai mahasiswa. Data tersebut perlu dikelola secara sistematis agar dapat digunakan dalam pemantauan capaian pembelajaran dan perbaikan berkelanjutan [27]. Pada penelitian ini, OBE digunakan sebagai dasar konseptual karena sistem yang diuji merupakan Sistem CPL berbasis OBE. Pembahasan OBE tidak diarahkan pada aspek pedagogis secara mendalam, tetapi sebagai konteks kebutuhan sistem *monitoring* dan *Dashboard* yang menyajikan informasi capaian pembelajaran secara terukur.

2. Capaian Pembelajaran Lulusan (CPL)

Capaian Pembelajaran Lulusan (CPL) merupakan rumusan kemampuan yang harus dimiliki mahasiswa setelah menyelesaikan pendidikan pada suatu program studi [28]. CPL umumnya mencakup aspek sikap, pengetahuan, keterampilan umum, dan keterampilan khusus yang menjadi acuan dalam penyusunan kurikulum serta evaluasi keberhasilan pembelajaran [29].

Dalam sistem berbasis OBE, CPL berperan sebagai rujukan utama dalam pemetaan mata kuliah, CPMK, metode penilaian, dan evaluasi hasil belajar mahasiswa [30]. Setiap mata kuliah dapat berkontribusi terhadap satu atau beberapa CPL, sehingga pengelolaan CPL membutuhkan relasi data yang jelas antara mata kuliah, CPMK, CPL, dan nilai mahasiswa.

Pada penelitian ini, CPL digunakan sebagai konteks data yang diproses oleh Sistem CPL berbasis OBE, khususnya pada *Dashboard* admin. Pembahasan CPL

tidak diarahkan pada validitas akademik perhitungannya, tetapi untuk menjelaskan bahwa informasi pada *Dashboard* merupakan hasil pengolahan data akademik yang saling berelasi dan dapat memengaruhi proses pemuatan data agregat pada *Endpoint Dashboard*.

3. Sistem *Monitoring* Capaian Pembelajaran

Sistem penilaian CPL berbasis web merupakan sistem informasi akademik yang digunakan untuk mengelola data capaian pembelajaran secara terstruktur dan terpusat [31]. Dalam konteks OBE, sistem ini menghubungkan data CPL, CPMK, mata kuliah, teknik penilaian, mahasiswa, dosen, nilai, dan kurikulum agar proses pemantauan serta evaluasi capaian pembelajaran dapat dilakukan secara lebih sistematis [32].

Pada penelitian ini, sistem penilaian CPL diposisikan sebagai objek evaluasi performa, bukan sebagai objek pengembangan fitur baru. Fokus pembahasan diarahkan pada karakteristik sistem yang memengaruhi proses pemuatan *Dashboard* admin, terutama alur kerja antara *Frontend*, *Backend*, dan *Database*. *Frontend* berperan sebagai antarmuka pengguna, *Backend* memproses *Request* dan data agregat, sedangkan *Database* menyimpan data akademik yang dibutuhkan sistem. Alur *Request-Response* dari *Frontend* ke *Backend*, proses pengambilan data, perhitungan agregat, dan pengiriman *Response* menjadi bagian penting yang dapat memengaruhi performa *Dashboard* [33].

4. *Dashboard* Sistem Informasi

Dashboard merupakan antarmuka informasi yang menyajikan data penting secara ringkas, visual, dan mudah dipahami untuk mendukung pemantauan serta pengambilan keputusan [34]. Dalam sistem akademik berbasis OBE, *Dashboard* dapat digunakan untuk menampilkan informasi capaian pembelajaran, seperti rata-rata CPL, distribusi capaian, tren semester, grafik CPL, serta hubungan antara mata kuliah dan capaian pembelajaran [35]. *Dashboard* membantu pengguna memahami kondisi akademik tanpa harus membaca seluruh data mentah, karena informasi telah diringkas dan divisualisasikan sesuai kebutuhan *monitoring*.

Pada penelitian ini, *Dashboard* admin Sistem CPL diposisikan sebagai modul utama yang menampilkan data agregat dari *Backend*, seperti statistik global, grafik CPL, tren capaian, kelengkapan data, *alerts*, *insights*, analisis dosen, evaluasi mahasiswa terbatas, serta *Filter Dashboard*. Pengujian tidak diarahkan pada komponen tampilan React atau performa *rendering* browser, tetapi pada *Endpoint Dashboard* admin yang menghasilkan *Response* data agregat. Dengan demikian, performa yang dianalisis adalah performa *Request-Response Endpoint Dashboard*, bukan performa visual antarmuka secara penuh [36].

Karakteristik *Dashboard* admin lebih kompleks dibandingkan halaman sederhana karena memuat beberapa kelompok informasi yang membutuhkan proses

Query, relasi data, perhitungan, *Filter*, dan penyusunan *Response*. Oleh karena itu, *Endpoint Dashboard* admin relevan untuk diuji menggunakan skenario beban pengguna simultan [7].

5. Data Mentah, Data *Dummy*, dan Data Agregat *Dashboard*

Data mentah *Database* merupakan data asli yang tersimpan pada tabel sebelum melalui proses pengolahan, perhitungan, atau peringkasan informasi [37]. Dalam penelitian ini, data yang digunakan bukan data akademik asli, melainkan data *dummy* yang disusun mengikuti struktur data pada Sistem CPL berbasis OBE. Data *dummy* digunakan sebagai data utama pengujian untuk menggambarkan skala *Database*, relasi antardata, dan bentuk data yang diproses oleh *Dashboard*.

Tabel 2 Bentuk Data

Jenis Data	Contoh Data	Keterangan
Data angka	nilai CPL, nilai CPMK, bobot penilaian, jumlah mahasiswa, rata-rata capaian	Digunakan untuk proses perhitungan dan agregasi <i>Dashboard</i>
Teks pendek	nama mahasiswa, nama dosen, kode mata kuliah, kode CPL, semester, kelas	Digunakan sebagai identitas atau label data
Teks panjang	deskripsi CPL, deskripsi CPMK, keterangan evaluasi	Digunakan sebagai informasi penjelas dalam sistem
Data relasional	relasi mata kuliah-CPL, CPMK-CPL, dosen-mata kuliah, mahasiswa-nilai	Digunakan untuk menghubungkan data antarentitas

Berbeda dengan data mentah, data agregat merupakan data hasil pengolahan yang telah diringkas, dihitung, diFilter, atau dikelompokkan sesuai kebutuhan analisis [38]. Dalam konteks *Dashboard* admin Sistem CPL, data *dummy* tidak dikirim seluruhnya ke *Frontend*, tetapi diproses terlebih dahulu oleh *Backend* melalui *Query Database*, relasi data, perhitungan, *Filter*, dan penyusunan *Response*.

Dengan demikian, objek pengujian dalam penelitian ini difokuskan pada *Endpoint Dashboard* admin yang menghasilkan *Response* data agregat, seperti statistik global, grafik CPL, tren semester, analisis dosen, dan evaluasi mahasiswa terbatas, bukan pada seluruh isi *Database* atau seluruh fitur sistem.

6. Teknologi Sistem yang Diuji

Teknologi sistem yang diuji perlu dijelaskan untuk memahami arsitektur aplikasi dan alur pemrosesan data pada *Dashboard* admin. Sistem terdiri dari *Frontend*, *Backend*, ORM, dan *Database* yang saling berperan dalam proses *Request-Response*, mulai dari permintaan data, pemrosesan *Server*, pengambilan data, hingga pengiriman *Response* kembali ke *Frontend*.

a) *Frontend*: React.js dan TypeScript

Frontend berfungsi sebagai antarmuka pengguna. Sistem menggunakan React.js untuk membangun antarmuka berbasis komponen dan TypeScript untuk mendukung struktur tipe data dalam pengembangan aplikasi [39]. React memang dirancang untuk membangun antarmuka berbasis komponen, sedangkan TypeScript menambahkan dukungan tipe pada JavaScript.

b) *Backend*: Node.js dan Express.js

Backend berfungsi menerima *Request*, memproses logika aplikasi, mengambil data, melakukan agregasi, dan mengirimkan *Response*. Pada sistem ini, Node.js digunakan sebagai *Runtime Server* dan Express.js digunakan sebagai *Framework API* [40]. Dalam konteks *Dashboard* admin, *Backend* berperan penting dalam membaca parameter *Filter*, memproses data, dan menyusun *Response Dashboard*.

c) ORM: Prisma ORM

Prisma ORM digunakan sebagai penghubung antara *Backend* dan *Database*. Prisma membantu akses data melalui kode aplikasi dan menyediakan *Type-safe Query*, tetapi *Query* yang dihasilkan tetap dapat memengaruhi performa sistem, terutama pada proses relasi data, *Filter*, dan agregasi [41].

d) *Database*: MariaDB

MariaDB digunakan sebagai sistem manajemen basis data relasional untuk menyimpan data akademik seperti mahasiswa, dosen, mata kuliah, CPL, CPMK, relasi, dan nilai [42]. Karena data tersebut saling berhubungan, proses pemuatan *Dashboard* admin dapat melibatkan *Query* dan relasi data yang cukup kompleks.

e) Alur *Request-Response Dashboard*

Alur kerja dimulai dari *Request Frontend* ke *Backend*. *Backend* kemudian mengambil data melalui Prisma ORM, memproses *Query Database*, melakukan perhitungan atau agregasi, menyusun *Response*, lalu mengirimkan data kembali ke *Frontend*.

f) Batasan Pembahasan Teknologi

Pembahasan teknologi tidak diarahkan pada pembangunan ulang sistem atau analisis kode secara rinci. Teknologi hanya dijelaskan sebagai konteks arsitektur yang memengaruhi performa *Endpoint Dashboard* admin. Fokus

utama penelitian tetap pada evaluasi performa *Request-Response* menggunakan Apache JMeter.

7. *Performance testing*

Performance testing merupakan metode pengujian untuk mengevaluasi kemampuan sistem dalam menangani beban kerja tertentu, terutama ketika jumlah *Request* atau pengguna meningkat [43]. Pengujian ini digunakan untuk melihat respons, kestabilan, dan kemampuan sistem dalam mempertahankan performa pada berbagai kondisi beban [44].

Dalam aplikasi berbasis web, *Performance testing* penting dilakukan karena sistem yang berjalan normal pada satu pengguna belum tentu tetap stabil ketika diakses banyak pengguna secara simultan. Evaluasi biasanya dilakukan menggunakan parameter seperti *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem. Selain parameter tersebut, penelitian performa juga dapat memanfaatkan informasi penggunaan sumber daya *Server*, seperti CPU dan memori, sebagai indikator pendukung untuk membantu menginterpretasikan perubahan performa aplikasi ketika menerima peningkatan beban kerja. Pengamatan terhadap kedua sumber daya tersebut tidak menggantikan metrik utama pengujian, tetapi digunakan untuk memperkuat analisis terhadap perilaku sistem selama proses pengujian [45].

Pada penelitian ini, *Performance testing* digunakan untuk mengevaluasi *Endpoint Dashboard* admin Sistem CPL berbasis OBE. Hasil pengujian tidak digunakan untuk memastikan dugaan penyebab *Bottleneck* secara mutlak, tetapi untuk membaca pola penurunan performa dan menyusun dugaan teknis berdasarkan data pengujian. JMeter juga menyediakan laporan pengujian yang memuat grafik dan statistik seperti *Response time*, *Throughput*, serta error untuk mendukung analisis performa [46].

8. *Load testing*

Load testing merupakan bagian dari *Performance testing* yang dilakukan dengan memberikan beban pengguna secara bertahap untuk mengetahui kemampuan sistem pada kondisi beban normal hingga meningkat [47]. Pengujian ini digunakan untuk melihat respons *Endpoint* ketika jumlah pengguna virtual yang mengakses sistem bertambah secara simultan [48].

Dalam penelitian ini, konsep *Load testing* digunakan sebagai salah satu kategori skenario pengujian performa untuk mengevaluasi perubahan performa *Endpoint* pada kondisi peningkatan beban pengguna secara bertahap. Pengujian difokuskan pada *Endpoint Dashboard* yang menghasilkan data agregat dari *Backend*, bukan pada seluruh fitur sistem atau seluruh isi *Database*. Parameter yang diamati meliputi *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem [49].

Apache JMeter digunakan untuk mengatur skenario pengguna virtual melalui *Thread Group* dan mengirim *Request* ke *Endpoint* menggunakan *HTTP Request*. Dokumentasi JMeter menjelaskan bahwa *Thread Group* digunakan untuk menentukan jumlah pengguna yang disimulasikan, sedangkan *HTTP Request* digunakan untuk mengirim *Request* HTTP/HTTPS ke *Server*.

9. Stress testing

Stress testing merupakan bagian dari *Performance testing* yang dilakukan dengan memberikan beban melebihi kondisi normal untuk mengetahui batas kemampuan dan kestabilan sistem [50]. Pengujian ini digunakan untuk mengamati perilaku sistem ketika menerima tekanan tinggi, seperti peningkatan *Response time*, kenaikan *Error rate*, penurunan *Throughput*, atau munculnya indikasi penurunan kestabilan sistem [11].

Pada penelitian ini, konsep *Stress testing* diterapkan sebagai salah satu kategori dalam skenario pengujian performa untuk mengevaluasi perilaku *Endpoint* ketika menerima beban yang melebihi kondisi operasional normal. Pengujian ini tidak bertujuan untuk membuktikan penyebab *Bottleneck* secara langsung, melainkan untuk mengidentifikasi dugaan *Bottleneck* berdasarkan perubahan parameter performa, yaitu *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem. Hasil pengujian kemudian dianalisis bersama seluruh skenario pengujian performa sebagai dasar penyusunan dugaan teknis dan rekomendasi perbaikan [51]. Dalam JMeter, skenario beban dapat diatur melalui *Thread Group* yang menentukan jumlah pengguna virtual dan pola pengiriman *Request*.

10. Parameter Pengujian Performa

Parameter pengujian performa digunakan untuk mengukur kinerja *Endpoint Dashboard* admin secara kuantitatif pada setiap skenario pengujian [52]. Dalam penelitian ini, parameter utama yang digunakan meliputi *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem karena mampu menggambarkan kecepatan respons, kapasitas pemrosesan, tingkat kegagalan, serta konsistensi performa sistem [53].

a) *Response time*

Response time merupakan waktu yang dibutuhkan sistem untuk merespons *Request* pengguna. Parameter ini digunakan untuk mengetahui seberapa cepat *Endpoint Dashboard* admin memproses dan mengirimkan *Response* data agregat dari *Backend* [54].

b) *Throughput*

Throughput menunjukkan jumlah *Request* yang dapat diproses sistem dalam satuan waktu tertentu. Parameter ini digunakan untuk melihat kapasitas *Endpoint* dalam menangani *Request* ketika jumlah pengguna simultan meningkat [55].

c) *Error rate*

Error rate merupakan persentase *Request* yang gagal selama pengujian. Nilai ini digunakan untuk melihat tingkat keberhasilan *Endpoint* dalam memproses *Request* pada setiap skenario beban.

d) Penggunaan CPU

Penggunaan CPU menunjukkan tingkat aktivitas prosesor ketika aplikasi memproses *Request*. Dalam penelitian performa aplikasi web, peningkatan utilisasi CPU dapat menunjukkan bertambahnya beban komputasi akibat meningkatnya jumlah *Request* yang diproses. Pada penelitian ini, penggunaan CPU diamati sebagai indikator pendukung dalam menginterpretasikan perubahan performa *Endpoint Dashboard Admin*.

e) Penggunaan Memori

Penggunaan memori menunjukkan besarnya kapasitas RAM yang digunakan aplikasi selama menjalankan proses. Pada pengujian performa, peningkatan penggunaan memori dapat memberikan gambaran mengenai kebutuhan sumber daya aplikasi ketika memproses beban pengguna yang semakin besar. Dalam penelitian ini, penggunaan memori digunakan sebagai indikator pendukung dalam analisis performa *Endpoint*.

f) Kestabilan Sistem

Kestabilan sistem diamati dari pola perubahan *Response time*, *Throughput*, dan *Error rate*. Sistem dianggap relatif stabil apabila peningkatan jumlah pengguna tidak menyebabkan lonjakan waktu respons, penurunan *Throughput* signifikan, atau peningkatan error yang tinggi.

g) Batasan Interpretasi

Parameter tersebut digunakan untuk mengevaluasi performa *Request-Response Endpoint Dashboard admin*, bukan performa seluruh sistem, tampilan *Frontend*, atau proses *rendering* browser. Apache JMeter sendiri menyediakan laporan pengujian yang memuat metrik seperti ringkasan keberhasilan/gagal, statistik transaksi, *Response time*, dan *Throughput*.

11. Apache JMeter

Apache JMeter merupakan perangkat lunak *open-source* berbasis Java yang digunakan untuk melakukan *Load testing* dan mengukur performa aplikasi, termasuk aplikasi web, *API*, *Database*, dan layanan jaringan [56]. JMeter dapat mensimulasikan beban tinggi pada *Server* untuk menguji kekuatan sistem dan menganalisis performa pada berbagai jenis beban. Selain itu, JMeter bekerja pada level protokol, sehingga tidak mengukur *rendering* halaman seperti browser.

Dalam penelitian ini, Apache JMeter dipilih sebagai alat utama karena sesuai dengan kebutuhan pengujian performa *Endpoint Dashboard admin Sistem CPL* berbasis OBE. Objek pengujian berupa *Endpoint HTTP/API* yang menghasilkan data agregat dari *Backend*, yaitu */api/dashboard/stats*, */api/dashboard/dosen*, dan

/api/dashboard/students. JMeter mendukung pengaturan jumlah pengguna virtual, pengiriman *Request* ke *Endpoint*, serta pencatatan metrik performa seperti *Response time*, *Throughput*, *Error rate*, dan status *Request* [57].

Beberapa komponen JMeter yang relevan dalam penelitian ini adalah sebagai berikut:

a. *Thread Group*

Merupakan komponen utama pada Apache JMeter yang digunakan untuk mengatur simulasi pengguna virtual (virtual user) dalam suatu pengujian performa. Komponen ini digunakan untuk menentukan jumlah pengguna yang disimulasikan, waktu peningkatan pengguna (ramp-up period), durasi pengujian, serta pola eksekusi *Request* dalam test plan. Konfigurasi *Thread Group* menentukan bagaimana pengguna virtual dibangkitkan dan bagaimana *Request* dikirimkan ke sistem selama proses pengujian berlangsung [58].

b. *HTTP Request*

Digunakan untuk menentukan *Endpoint* yang diuji, yaitu *Endpoint Dashboard* admin yang menghasilkan data agregat dari *Backend*.

c. *HTTP Header Manager*

Digunakan untuk mengatur header *Request*, seperti token autentikasi atau tipe konten yang diperlukan *Server*.

d. *JSON Extractor*

JSON Extractor digunakan untuk mengambil nilai tertentu dari *Response* berbentuk JSON. Pada penelitian ini, *JSON Extractor* digunakan untuk mengekstraksi token JWT dari *Response* proses login sehingga token tersebut dapat digunakan pada *Request* berikutnya.

e. *Timer*

Timer digunakan untuk memberikan jeda antar *Request* sehingga pola akses pengguna virtual dapat dibuat lebih mendekati aktivitas pengguna sebenarnya. Pada penelitian ini digunakan *Uniform Random Timer* untuk mensimulasikan waktu tunggu (*think time*) sebelum pengguna menjalankan *Request* berikutnya

f. *Listener dan Report*

Listener atau mekanisme pelaporan pada JMeter digunakan untuk menyimpan hasil pengujian yang diperoleh selama proses eksekusi. Data hasil pengujian digunakan untuk menganalisis parameter performa seperti *Response time*, *Throughput*, *Error rate*, dan jumlah *Request* yang berhasil diproses.

Berdasarkan kemampuan tersebut, Apache JMeter digunakan dalam penelitian ini untuk membangun skenario pengujian performa secara terukur pada *Endpoint Dashboard Admin Sistem CPL*. Data hasil pengujian kemudian digunakan sebagai dasar evaluasi performa sistem, analisis perubahan karakteristik performa berdasarkan tingkat beban pengguna, serta identifikasi dugaan *Bottleneck* berdasarkan pola hasil pengujian.

12. *Bottleneck* Sistem

Bottleneck merupakan kondisi ketika salah satu bagian sistem menjadi titik hambat yang membatasi performa sistem secara keseluruhan [59]. Pada aplikasi berbasis web, *Bottleneck* dapat terjadi pada proses *Backend*, *Query Database*, relasi antartabel, ukuran *Response*, konfigurasi *Server*, jaringan, atau proses pemrosesan data yang tidak efisien.

Dalam konteks *Dashboard* admin Sistem CPL berbasis OBE, potensi *Bottleneck* dapat muncul karena *Endpoint Dashboard* memproses data agregat dari beberapa sumber data sebelum dikirimkan ke *Frontend*. Proses tersebut dapat melibatkan pembacaan parameter *Filter*, akses data melalui Prisma ORM, *Query* ke MariaDB, relasi antartabel, perhitungan agregat, dan penyusunan *Response*.

Pada penelitian ini, *Bottleneck* tidak diidentifikasi melalui profiling kode atau tracing *Query* secara mendalam. Identifikasi dilakukan sebagai dugaan teknis berdasarkan pola hasil pengujian menggunakan Apache JMeter. Dugaan *Bottleneck* ditelusuri dengan membandingkan performa tiga *Endpoint Dashboard* admin, yaitu */api/dashboard/stats*, */api/dashboard/dosen*, dan */api/dashboard/students*. [60].

Endpoint yang menunjukkan *Response time* paling tinggi, *Throughput* paling rendah, *Error rate* lebih besar, atau kestabilan paling buruk pada skenario beban tertentu diposisikan sebagai *Endpoint* yang berpotensi menjadi sumber *Bottleneck*. Dengan demikian, analisis *Bottleneck* dalam penelitian ini digunakan untuk membantu menyusun rekomendasi perbaikan, bukan sebagai pembuktian final terhadap kode program atau *Query Database* tertentu.

13. Rekomendasi Perbaikan Performa Terbatas

Rekomendasi perbaikan performa merupakan arahan teknis yang akan disusun berdasarkan hasil evaluasi pengujian, karakteristik *Endpoint Dashboard* admin, dan dugaan *Bottleneck* yang diidentifikasi [61]. Rekomendasi ini tidak diposisikan sebagai hasil optimasi yang sudah diimplementasikan, tetapi sebagai arahan awal untuk pengembangan sistem berikutnya.

Dalam penelitian ini, rekomendasi diarahkan pada *Endpoint* yang menunjukkan performa paling lemah berdasarkan hasil pengujian. Apabila *Endpoint* tertentu memiliki *Response time* tinggi, *Throughput* menurun, *Error rate* meningkat, atau performa tidak stabil, maka rekomendasi disusun sesuai dengan karakteristik proses pada *Endpoint* tersebut.

Beberapa bentuk rekomendasi yang dapat disusun dalam penelitian ini meliputi:

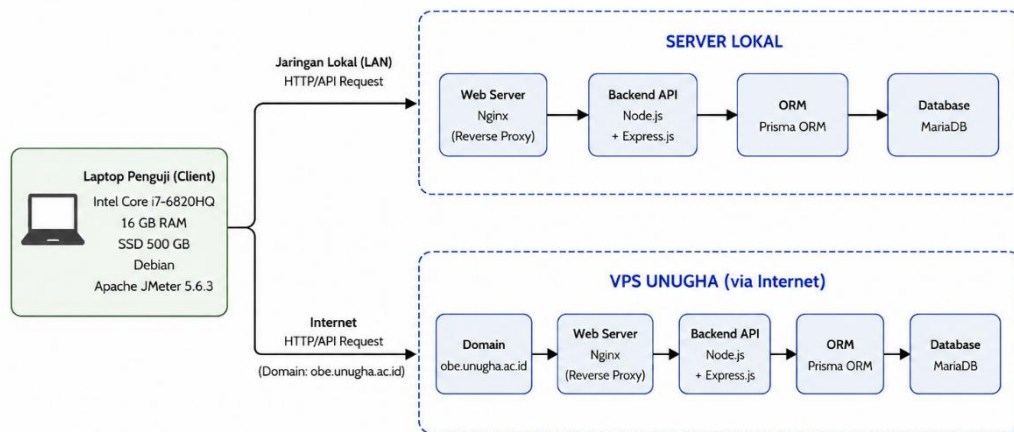
- a) Evaluasi *Query Database*
Evaluasi dilakukan untuk melihat kemungkinan adanya *Query* yang terlalu berat, relasi yang kompleks, atau proses pengambilan data yang belum efisien.
- b) *Indexing* pada kolom relasi dan *Filter*
Indexing dapat diarahkan pada kolom yang sering digunakan dalam proses pencarian, *Filter*, atau relasi data, seperti program studi, semester, mata kuliah, CPL, CPMK, dan nilai [62]. *Index* pada MariaDB dapat membantu meningkatkan performa *Query* dan mempercepat pengambilan data.
- c) Pengurangan *Over-fetching*
Over-fetching terjadi ketika sistem mengambil data lebih banyak daripada yang dibutuhkan. Rekomendasi ini diarahkan agar *Endpoint Dashboard* admin hanya mengambil data yang diperlukan untuk membentuk *Response* [63].
- d) *Caching* data agregat
Bersifat relatif statis dapat dipertimbangkan sebagai salah satu alternatif untuk mengurangi proses perhitungan dan pengambilan data yang dilakukan secara berulang. Pendekatan ini berpotensi menurunkan beban *Query* pada *Backend* dan basis data sehingga dapat membantu meningkatkan waktu respons sistem, terutama pada fitur/halaman yang sering mengakses data agregat [64].
- e) Pembatasan *Payload Response*
Data yang dikirim dari *Backend* ke *Frontend* perlu dibatasi sesuai kebutuhan tampilan agar ukuran *Response* tidak terlalu besar dan proses pengiriman data lebih efisien.
- f) Evaluasi *Endpoint* yang berat
Apabila salah satu *Endpoint* memiliki proses yang lebih berat dibanding *Endpoint* lainnya, maka perlu dilakukan evaluasi terhadap proses agregasi, relasi antartabel, penggunaan *Filter*, atau kemungkinan pemisahan proses agar pemuatan data lebih terkontrol.
- g) Penyesuaian konfigurasi *Server*
Jika hasil pengujian menunjukkan keterbatasan sumber daya, rekomendasi dapat diarahkan pada penyesuaian konfigurasi atau spesifikasi *Server*.

Dengan demikian, rekomendasi perbaikan dalam penelitian ini berfungsi sebagai arahan teknis awal untuk meningkatkan kesiapan performa *Endpoint Dashboard* admin. Rekomendasi tetap dibatasi pada hasil evaluasi pengujian dan dugaan teknis, bukan berdasarkan implementasi optimasi atau pembuktian langsung melalui profiling kode secara mendalam.

BAB III METODOLOGI

A. Waktu dan Tempat Penelitian

Waktu penelitian direncanakan dimulai setelah pelaksanaan seminar proposal, dengan durasi kurang lebih 45 hari kerja. Selama waktu tersebut, kegiatan penelitian meliputi tahap persiapan, pelaksanaan pengujian, analisis hasil pengujian, penyusunan rekomendasi perbaikan secara terbatas, penulisan laporan, serta persiapan sidang.



Gambar 2 Lingkungan Pengujian pada Server Lokal dan VPS UNUGHA

Penelitian dilakukan pada dua lingkungan pengujian, yaitu server lokal dan VPS UNUGHA. Server lokal merupakan perangkat server yang diakses melalui jaringan lokal, sedangkan VPS UNUGHA merupakan server virtual milik Universitas Nahdlatul Ulama Al Ghazali Cilacap yang menjalankan API Sistem CPL dan diakses melalui jaringan internet menggunakan domain obe.unugha.ac.id.

Laptop penguji digunakan untuk menjalankan Apache JMeter sebagai alat simulasi pengguna virtual. Pengujian terhadap server lokal dilakukan melalui jaringan lokal, sedangkan pengujian terhadap VPS UNUGHA mengharuskan laptop penguji terhubung ke internet.

B. Alat dan Bahan

1. Alat

Alat yang digunakan dalam penelitian ini terdiri atas perangkat keras, perangkat lunak, dan jaringan yang mendukung pengujian performa *Dashboard Admin Sistem Capaian Pembelajaran Lulusan (CPL) berbasis Outcome Based Education (OBE)*. *Server* lokal dan VPS UNUGHA digunakan sebagai lingkungan sistem yang diuji, sedangkan laptop penguji digunakan untuk menjalankan Apache JMeter dan utilitas pemantauan.

Laptop penguji mengirimkan *request* HTTP/API menuju masing-masing *server* menggunakan skenario pengujian yang sama. Pemantauan penggunaan CPU dan memori *server* dilakukan menggunakan Glances melalui koneksi *Secure Shell* (SSH).

Tabel 3 Alat Penelitian

No	Alat	Spesifikasi	Fungsi
1	<i>Server</i> lokal	Intel Core 2 Duo P8400, RAM 3 GB, SSD 256 GB, Debian <i>Server</i>	Menjalankan Sistem CPL berbasis OBE sebagai lingkungan pengujian lokal, termasuk <i>backend</i> Node.js dan Express.js, Prisma ORM, serta basis data MariaDB
2	VPS UNUGHA	Intel Xeon E5-2650 v4 2,20 GHz, 24 pemroses logis, RAM 12 GB, kapasitas partisi sistem 97,9 GB, Ubuntu 24.04 64-bit	Menjalankan Sistem CPL berbasis OBE sebagai lingkungan pengujian VPS UNUGHA
3	Laptop penguji	Intel Core i7-6820HQ, RAM 16 GB, SSD 500 GB, Debian	Menjalankan Apache JMeter untuk menyimulasikan pengguna virtual dan mengirimkan <i>request</i> ke <i>endpoint Dashboard Admin</i>
4	Apache JMeter	Versi 5.6.3	Menyimulasikan pengguna virtual serta mengumpulkan nilai <i>response time</i> , <i>throughput</i> , dan <i>error rate</i>
5	<i>Web browser</i>	Google Chrome	Memverifikasi fungsi sistem dan memastikan <i>endpoint</i> dapat diakses sebelum pengujian

No	Alat	Spesifikasi	Fungsi
6	Jaringan pengujian	Jaringan lokal dan jaringan internet menuju VPS UNUGHA	Menghubungkan laptop penguji dengan server lokal melalui jaringan lokal serta dengan VPS UNUGHA melalui domain obe.unugha.ac.id menggunakan jaringan internet
7	<i>Secure Shell</i>	SSH melalui terminal	Mengakses <i>server</i> dan melakukan pemantauan selama pengujian berlangsung
8	Glances	Versi 3.3.1.1	Memantau penggunaan CPU dan memori <i>server</i> secara <i>real-time</i> selama setiap skenario pengujian

2. Bahan

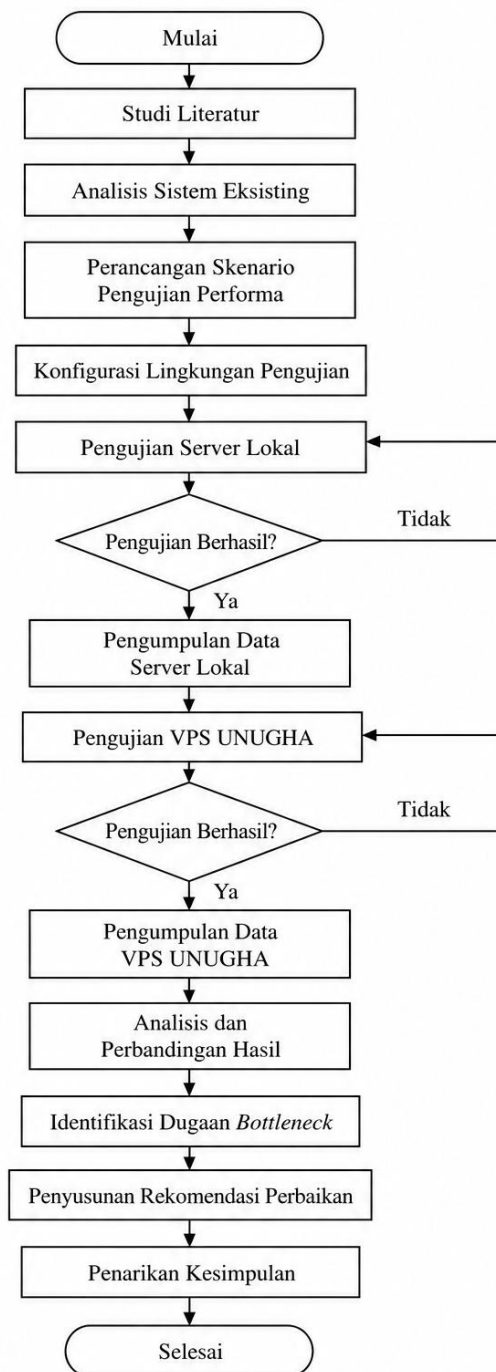
Bahan penelitian merupakan objek dan data yang digunakan selama proses pengujian performa sistem. Bahan penelitian yang digunakan dalam penelitian ini meliputi sistem yang diuji, data pengujian, serta *Endpoint Dashboard* Admin yang menjadi fokus evaluasi performa.

Tabel 4 Bahan Penelitian

No	Bahan	Keterangan
1	Sistem CPL Berbasis OBE	Sistem yang menjadi objek penelitian dan digunakan untuk mengelola data capaian pembelajaran lulusan
2	<i>Dashboard</i> Admin	Modul <i>Dashboard</i> yang menampilkan statistik CPL, grafik capaian, tren semester, analisis dosen, evaluasi mahasiswa terbatas, dan informasi <i>monitoring</i> lainnya
3	Data <i>Dummy</i>	Data pengujian yang disusun sesuai struktur <i>Database</i> Sistem CPL untuk menggambarkan relasi dan skala data sistem
4	<i>Endpoint Dashboard</i> Admin	<i>Endpoint</i> yang diuji meliputi /api/dashboard/stats, /api/dashboard/dosen, dan /api/dashboard/students
5	<i>Database</i> MariaDB	Basis data yang digunakan untuk menyimpan dan mengelola data Sistem CPL
6	Token Autentikasi JWT	Digunakan untuk mengakses <i>Endpoint</i> yang memerlukan autentikasi selama proses pengujian

C. Prosedur Penelitian

Prosedur penelitian merupakan tahapan operasional yang disusun secara sistematis dari awal hingga akhir penelitian. Penelitian ini difokuskan pada evaluasi dan perbandingan performa *endpoint Dashboard* Admin Sistem CPL berbasis OBE pada *server* lokal dan VPS UNUGHA menggunakan metode *load testing* dan *stress testing*. Pengujian dilakukan menggunakan skenario yang sama untuk memperoleh nilai *response time*, *throughput*, *error rate*, serta penggunaan CPU dan memori pada masing-masing *server*.



Gambar 3 Diagram Alir Prosedur Penelitian

Diagram alir pada Gambar 3 menunjukkan tahapan penelitian yang diawali dengan studi literatur, analisis sistem eksisting, dan perancangan skenario pengujian. Setelah lingkungan pengujian dikonfigurasi, pengujian dilakukan secara bergantian pada *server* lokal dan VPS UNUGHA menggunakan skenario yang sama. Data dari kedua *server* kemudian dianalisis dan dibandingkan berdasarkan *response time*, *throughput*, *error rate*, penggunaan CPU, dan penggunaan memori. Hasil analisis digunakan untuk mengidentifikasi dugaan *bottleneck*, menyusun rekomendasi perbaikan, dan menarik kesimpulan.

Tabel 5 Skenario Pengujian

No	Jumlah Pengguna Simultan	Jenis Pengujian	Tujuan Pengujian
1	1	<i>Baseline</i>	Mengetahui kondisi dasar performa <i>Endpoint Dashboard</i> admin
2	10	<i>Load testing</i> ringan	Mengamati performa <i>Endpoint</i> pada beban awal
3	20	<i>Load testing</i> sedang	Mengamati perubahan performa ketika jumlah pengguna meningkat
4	50	<i>Load testing</i> tinggi	Mengetahui kemampuan <i>Endpoint</i> pada beban menengah menuju tinggi
5	100	<i>Stress testing</i>	Mengamati perilaku <i>Endpoint</i> pada beban tinggi
6	200	<i>Stress testing</i> tinggi	Mengamati potensi penurunan performa dan kestabilan sistem
7	500	<i>Stress testing</i> ekstrem	Menguji batas kemampuan <i>Endpoint</i> pada beban maksimum pengujian

Seluruh skenario pengujian diterapkan pada *server* lokal dan VPS UNUGHA menggunakan konfigurasi Apache JMeter yang sama.

Pengujian dilakukan secara bertahap (*incremental load*), dimulai dari kondisi *baseline*, dilanjutkan dengan skenario *Load testing*, kemudian *Stress testing*. Melalui skenario tersebut perubahan *Response time*, *Throughput*, *Error rate*, dan

kestabilan sistem dianalisis untuk mengidentifikasi indikasi penurunan performa dan dugaan *Bottleneck*.

1. Studi Literatur

Tahap studi literatur dilakukan untuk memperoleh dasar teori, konsep, dan referensi yang mendukung pelaksanaan penelitian. Literatur yang dikaji berkaitan dengan *Outcome Based Education* (OBE), Capaian Pembelajaran Lulusan (CPL), sistem *monitoring* capaian pembelajaran, *Dashboard* Admin, data agregat, *Performance testing*, parameter pengujian performa, skenario pengujian performa, serta penggunaan Apache JMeter.

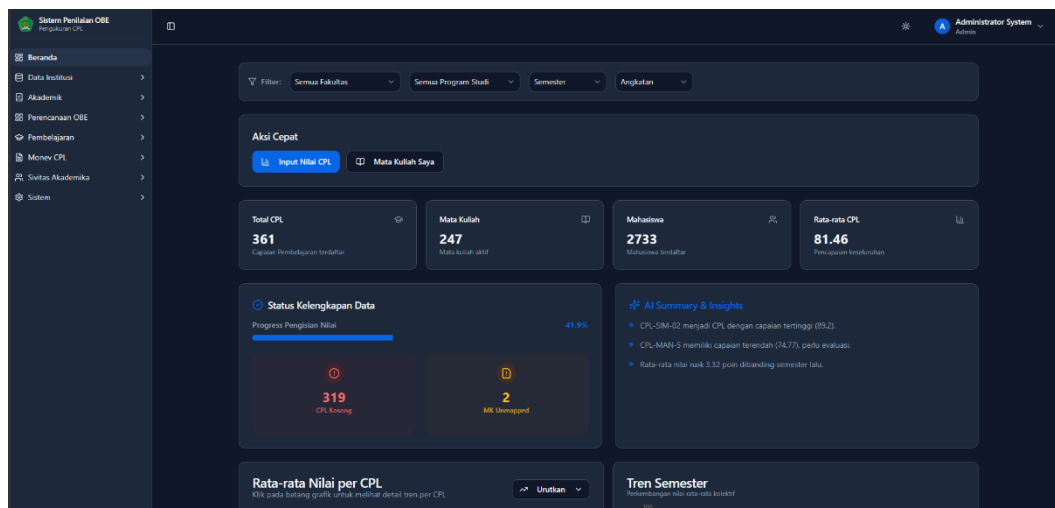
Studi literatur dilakukan dengan menelaah jurnal ilmiah, artikel konferensi, buku, dan dokumentasi resmi perangkat lunak yang relevan. Referensi mengenai OBE dan CPL digunakan untuk memahami konteks sistem yang diuji, sedangkan referensi mengenai *Dashboard* dan data agregat digunakan untuk memahami karakteristik pemuatan informasi pada *Dashboard* Admin. Selain itu, referensi mengenai *Performance testing*, termasuk konsep *Load testing* dan *Stress testing*, digunakan sebagai landasan dalam menyusun skenario pengujian performa serta menentukan parameter yang akan dianalisis.

Pada tahap ini juga dipelajari dokumentasi teknis yang berkaitan dengan teknologi sistem dan alat pengujian, seperti Apache JMeter, Node.js, Express.js, Prisma ORM, dan MariaDB. Dokumentasi tersebut digunakan untuk memahami alur kerja sistem berbasis web, proses *Request-Response*, akses basis data, serta kemungkinan faktor teknis yang dapat memengaruhi performa *Endpoint Dashboard* Admin.

Hasil dari tahap studi literatur digunakan sebagai dasar dalam menentukan ruang lingkup penelitian, menyusun skenario pengujian performa, menetapkan parameter pengujian, serta merumuskan dasar analisis terhadap hasil pengujian. Dengan demikian, tahap studi literatur menjadi landasan awal sebelum dilakukan analisis sistem eksisting dan perancangan skenario pengujian performa.

2. Analisis Sistem Eksisting

Tahap analisis sistem eksisting dilakukan untuk memahami karakteristik Sistem Penilaian Capaian Pembelajaran Lulusan berbasis OBE, khususnya pada bagian *Dashboard* admin. Analisis difokuskan pada alur kerja *Dashboard*, *Endpoint API*, struktur data, serta proses pemuatan data dari *Backend* agar pengujian performa sesuai dengan ruang lingkup penelitian.



Gambar 4 Tampilan *Dashboard* Admin Sistem CPL

Berdasarkan Gambar 4, *Dashboard* admin memuat beberapa kelompok informasi yang diproses oleh *Backend* sebelum dikirimkan ke *Frontend*. Oleh karena itu, pengujian difokuskan pada *Endpoint Dashboard* admin yang menghasilkan *Response* data agregat, bukan pada seluruh fitur sistem atau proses *rendering* antarmuka. *Endpoint* yang diuji dalam penelitian ini ditunjukkan pada tabel 6.

Tabel 6 *Endpoint*

No	<i>Endpoint</i>	Metode	Fokus Data yang Diuji
1	/api/dashboard/stats	GET	Statistik global, grafik CPL, tren semester, alert, dan insight
2	/api/dashboard/dosen	GET	Analisis dosen, jumlah kelas, rata-rata nilai kelas, dan progres input nilai
3	/api/dashboard/students	GET	Evaluasi mahasiswa terbatas dan ringkasan nilai mahasiswa

Pada tahap ini, *Dashboard* admin dipahami sebagai modul yang menampilkan data agregat, bukan seluruh data mentah dari *Database*. Data mentah seperti mahasiswa, dosen, mata kuliah, CPL, CPMK, relasi, dan nilai tetap tersimpan pada *Database*, sedangkan data yang ditampilkan pada *Dashboard* merupakan hasil

pengolahan *Backend* berupa ringkasan, perhitungan, *Filter*, dan susunan *Response* sesuai kebutuhan tampilan. Pendekatan ini sejalan dengan konsep *Dashboard* berbasis agregasi, karena pengolahan data ringkasan dapat memengaruhi performa *Query* dan pemuatan *Dashboard*.

Selain itu, tahap ini juga mengidentifikasi data *dummy* yang digunakan sebagai data utama pengujian. Data *dummy* disusun mengikuti struktur data pada Sistem CPL berbasis OBE, sehingga dapat menggambarkan skala *Database*, relasi antardata, dan bentuk data yang diproses oleh *Dashboard*. Jumlah data dibuat tetap selama pengujian agar perubahan performa lebih dipengaruhi oleh jumlah pengguna simultan, bukan oleh perubahan volume data pada *Database*.

Tabel 7 Ringkasan Data *Dummy* Sistem CPL

Komponen	Jumlah	Keterangan
Jumlah tabel <i>Database</i>	50 tabel	Struktur data Sistem CPL berbasis OBE
Total baris data	~341.502 baris	Total seluruh tabel pada <i>Database</i>
Ukuran <i>Database</i>	417,8 MiB	Ukuran keseluruhan <i>Database</i>
Mahasiswa	2.733 data	Sumber data ringkasan mahasiswa pada <i>Dashboard</i>
Users	2.873 data	Akun pengguna sistem
Profiles	2.868 data	Profil pengguna
CPL	367 data	Capaian Pembelajaran Lulusan
Mata kuliah aktif	247 data	Sumber data ringkasan mata kuliah pada <i>Dashboard</i>
nilai_cpmk	~226.396 baris	Data nilai CPMK
nilai_cpl	~66.512 baris	Data rekap nilai CPL

Komponen	Jumlah	Keterangan
nilai_sub_cpmk	7.628 baris	Data nilai sub-CPMK
nilai_teknik_penilaian	7.527 baris	Data nilai teknik penilaian
cpmk_cpl_mapping	3.246 baris	Relasi CPMK ke CPL

Berdasarkan Tabel 7, data *dummy* menunjukkan bahwa *Database* sistem memiliki struktur cukup besar dan saling berelasi. Selain data utama, terdapat tabel pendukung seperti fakultas, prodi, angkatan, kurikulum, semester, rubrik, teknik penilaian, role, role_permissions, dan audit_logs. Keberadaan tabel relasional seperti cpl_mata_kuliah, mata_kuliah_pengampu, cpmk_cpl_mapping, dan profil_lulusan_cpl menunjukkan bahwa sistem memiliki keterkaitan data akademik yang kompleks.

Selain jumlah data dan struktur relasi basis data, karakteristik data yang diproses oleh *Endpoint Dashboard Admin* juga perlu diperhatikan karena dapat memengaruhi proses *Query*, relasi antartabel, dan agregasi data. Sistem CPL berbasis *Outcome Based Education* (OBE) menggunakan berbagai jenis data yang terdiri atas string, numerik, dan boolean. Sebagian besar relasi antarentitas menggunakan UUID dengan panjang 36 karakter sebagai primary key dan foreign key, sedangkan data identitas seperti nama, NIM, NIDN, email, kode CPL, dan kode mata kuliah menggunakan tipe VARCHAR dengan panjang yang bervariasi. Selain itu, sistem juga memproses data numerik berupa semester, SKS, nilai CPL, dan bobot kontribusi yang digunakan dalam perhitungan capaian pembelajaran.

Tabel 8 Karakteristik Data yang Diproses oleh *Endpoint Dashboard Admin*

Kategori Data	Tipe Basis Data	Panjang Data	Contoh Data
String Pendek	VARCHAR(36)	36 karakter	UUID, ID relasi
String Pendek	VARCHAR(10–20)	5–20 karakter	Kode CPL, kode mata kuliah

Kategori Data	Tipe Basis Data	Panjang Data	Contoh Data
String Sedang	VARCHAR(191)	10–100 karakter	Nama mahasiswa, nama dosen, NIM, NIDN, email
String Panjang	TEXT	>100 karakter	Deskripsi CPL
Angka Bulat	INT	1–8 digit	Semester, SKS
Angka Desimal	DECIMAL(5,2)	0,00–100,00	Nilai CPL
Angka Desimal	DECIMAL(3,2)	0,00–1,00	Bobot kontribusi CPL
Boolean	TINYINT(1)	0 atau 1	Status aktif

Berdasarkan karakteristik tersebut, *Endpoint Dashboard* Admin tidak hanya memproses data teks sederhana, tetapi juga melibatkan relasi data dan agregasi numerik dari beberapa tabel akademik yang berpotensi memengaruhi performa sistem ketika menerima *Request* secara simultan.

Dalam proses penyusunan informasi *Dashboard*, sistem tidak mengambil data dari satu tabel saja, melainkan melakukan pengolahan data dari beberapa tabel yang saling berelasi. Setiap *Endpoint* memiliki kebutuhan data yang berbeda sehingga jumlah tabel dan kompleksitas *Query* yang digunakan juga berbeda.

Tabel 9 Keterlibatan Tabel Database pada *Endpoint Dashboard* Admin

<i>Endpoint</i>	Tabel yang Diakses	Fungsi Utama
/api/dashboard/stats	nilai_cpl, cpl, mata_kuliah, cpl_mata_kuliah, users, user_roles	Menampilkan statistik CPL, mata kuliah, dan pengguna
/api/dashboard/dosen	nilai_cpl, profiles, mata_kuliah, mata_kuliah_pengampu	Menampilkan analisis capaian pembelajaran

<i>Endpoint</i>	Tabel yang Diakses	Fungsi Utama
		berdasarkan dosen pengampu
/api/dashboard/students	nilai_cpl, profiles	Menampilkan evaluasi dan ringkasan capaian mahasiswa

Berdasarkan Tabel 9, *Endpoint* /api/dashboard/stats memiliki kompleksitas pengolahan data yang lebih tinggi karena melibatkan lebih banyak tabel dan proses agregasi dibandingkan *Endpoint* lainnya. Perbedaan kompleksitas tersebut berpotensi menghasilkan karakteristik performa yang berbeda ketika sistem menerima beban pengguna secara bersamaan.

Meskipun volume data *dummy* di dalam *Database* cukup besar, data tersebut tidak seluruhnya dikirimkan ke sisi *Frontend*. Data yang dipetakan pada Tabel 7, Tabel 8, dan Tabel 9 murni digunakan untuk menggambarkan skala, karakteristik fisik, serta kompleksitas relasi pada *Database* internal sistem. Sementara itu, fokus pengujian tetap diarahkan pada *Endpoint Dashboard Admin* yang menghasilkan *Response* berupa data agregat. Dengan demikian, analisis performa dalam penelitian ini mutlak difokuskan pada kemampuan proses *Backend* dalam mengambil, mengolah, menghitung, memFilter, dan mengirimkan data ringkasan tersebut. Adapun kelompok data agregat dan kontrol interaksi pada *Dashboard Admin* ditunjukkan pada Tabel 10.

Tabel 10 Kelompok Data Agregat dan Kontrol *Dashboard Admin*

Kelompok Data	Objek Data / State	Fungsi
Statistik global	stats.cpl	Menampilkan jumlah keseluruhan CPL yang terdaftar pada sistem
Statistik global	stats.mataKuliah	Menampilkan jumlah mata kuliah aktif yang digunakan dalam pemetaan nilai
Statistik global	stats.users	Menampilkan jumlah mahasiswa atau pengguna sesuai cakupan <i>Filter</i>

Kelompok Data	Objek Data / State	Fungsi
Statistik global	stats.avgScore	Menampilkan rata-rata capaian CPL secara global berdasarkan data nilai yang telah diproses
Visualisasi	chartData	Menampilkan rata-rata nilai tiap CPL dalam bentuk grafik
Visualisasi	trendData	Menampilkan tren rata-rata capaian dari semester ke semester
Visualisasi	distributionData	Menampilkan distribusi nilai berdasarkan kategori tertentu
Visualisasi	performanceData	Menampilkan CPL dengan performa tertinggi atau terendah
Kelengkapan data	cplEmpty	Menampilkan jumlah atau daftar CPL yang belum memiliki data nilai referensi
Kelengkapan data	mkUnmapped	Menampilkan jumlah mata kuliah yang belum terpetakan dengan CPMK atau CPL
Kelengkapan data	progressPengisian	Menampilkan progres pengisian nilai oleh dosen dalam bentuk persentase
Peringatan sistem	<i>alerts</i>	Menampilkan kondisi penting, seperti CPL rendah atau mata kuliah yang perlu ditinjau

Kelompok Data	Objek Data / State	Fungsi
Wawasan otomatis	<i>insights</i>	Menampilkan ringkasan analitis, seperti CPL tertinggi, CPL terendah, atau perubahan tren
<i>Monitoring</i> operasional	dosenAnalysis	Menampilkan analisis dosen, jumlah kelas, rata-rata nilai kelas, dan progres input nilai
<i>Monitoring</i> mahasiswa	studentEvaluation	Menampilkan evaluasi mahasiswa secara terbatas, misalnya ringkasan nilai mahasiswa tertentu
Kontrol interaksi	activeFilters	Menyimpan parameter <i>Filter</i> seperti fakultas, program studi, semester, angkatan, kelas, atau mata kuliah
Kontrol interaksi	cplSortMode	Mengatur pengurutan grafik CPL berdasarkan nilai tertinggi atau terendah

Tabel 10 menunjukkan bahwa *Dashboard* admin memuat beberapa kelompok data agregat sesuai kebutuhan *monitoring*, seperti statistik global, visualisasi CPL, tren semester, distribusi nilai, kelengkapan data, peringatan sistem, insight otomatis, analisis dosen, evaluasi mahasiswa terbatas, serta kontrol *Filter*. Data yang ditampilkan bukan seluruh data mentah dari *Database*, melainkan hasil pengolahan *Backend* yang telah diringkaskan dan disusun sesuai kebutuhan tampilan *Dashboard*.

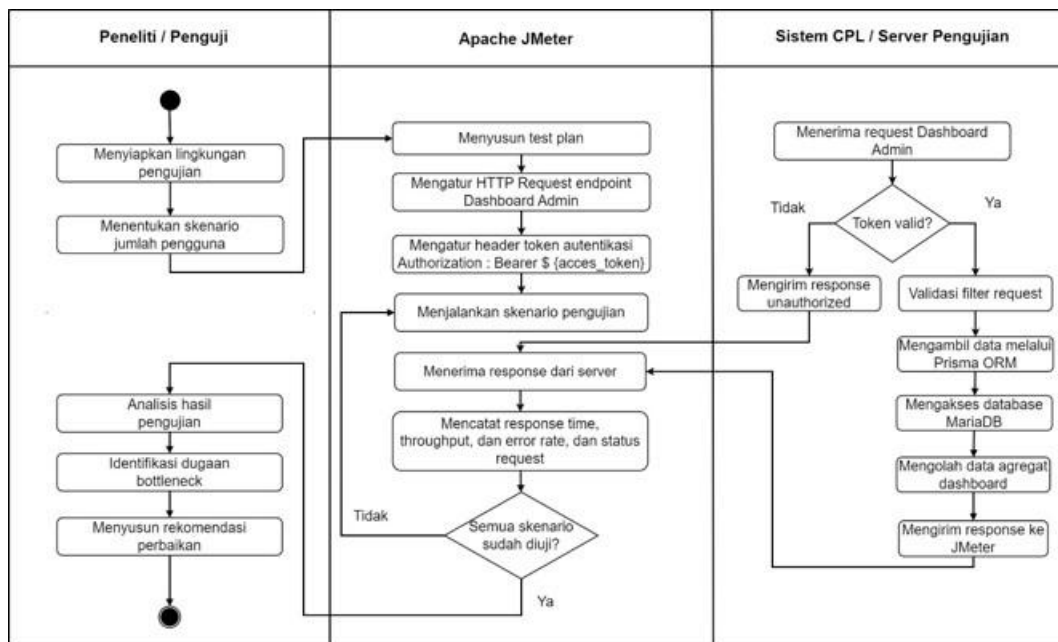
Selain itu, kontrol seperti *activeFilters* dan *cplSortMode* digunakan untuk menentukan cakupan dan urutan data yang ditampilkan. Perubahan *Filter* dapat menghasilkan *Response* yang berbeda karena *Backend* perlu memproses ulang data sesuai parameter yang digunakan. Oleh karena itu, pada tahap ini dilakukan identifikasi karakteristik *Endpoint Dashboard* admin, meliputi alamat *Endpoint*, metode *Request*, autentikasi, parameter *Filter*, dan jenis *Response* yang dihasilkan.

Analisis sistem eksisting juga mencakup pemahaman terhadap teknologi yang digunakan, yaitu React.js dan TypeScript pada *Frontend*, Node.js dan Express.js pada *Backend*, Prisma ORM, serta MariaDB. Pemahaman ini diperlukan karena

performa *Dashboard* dapat dipengaruhi oleh proses *Request-Response*, *Query Database*, relasi antardata, penggunaan *Filter*, dan ukuran *Response*. Prisma sendiri menyediakan panduan optimasi *Query* untuk mengidentifikasi dan memperbaiki *Query* yang lambat atau mahal, sedangkan JMeter menggunakan *Thread Group* dan *HTTP Request* untuk menyusun skenario pengujian web berbasis *Request*.

Dengan demikian, *Dashboard* admin dipahami sebagai modul informasi akademik yang lebih kompleks dibandingkan fitur ringan karena memuat ringkasan, visualisasi, peringatan, *monitoring*, dan data agregat. Hasil analisis sistem eksisting digunakan sebagai dasar penyusunan skenario pengujian agar evaluasi performa tetap terarah pada *Endpoint Dashboard* admin dan tidak melebar ke fitur lain di luar ruang lingkup penelitian.

3. Perancangan Skenario Pengujian



Gambar 5 Activity Diagram Pengujian Performa

Pada tahap ini dilakukan perancangan skenario pengujian performa menggunakan Apache JMeter. Skenario dirancang untuk mensimulasikan akses pengguna secara simultan terhadap *Endpoint Dashboard* admin Sistem CPL berbasis OBE. Pengujian difokuskan pada proses *Request-Response Endpoint* yang menghasilkan data agregat dari *Backend*.

Endpoint yang diuji dalam skenario ini meliputi */api/dashboard/stats*, */api/dashboard/dosen*, dan */api/dashboard/students*. Ketiga *Endpoint* tersebut dipilih karena digunakan dalam proses pemuatan *Dashboard* admin dan menghasilkan data agregat yang dibutuhkan untuk menampilkan statistik CPL, analisis dosen, serta evaluasi mahasiswa.

Activity diagram pada Gambar 5 menggambarkan alur pengujian performa *Endpoint Dashboard* admin menggunakan Apache JMeter. Alur dimulai dari penyiapan lingkungan, penentuan skenario pengguna, penyusunan test plan, pengaturan *Request* dan header autentikasi, hingga pelaksanaan pengujian. Sistem memproses *Request Dashboard* melalui validasi token, Prisma ORM, dan MariaDB, kemudian *Response* dicatat oleh JMeter berdasarkan *Response time*, *Throughput*, *Error rate*, dan status *Request*. Jika seluruh skenario telah selesai, hasil pengujian dianalisis untuk mengidentifikasi dugaan *Bottleneck* dan menyusun rekomendasi perbaikan.

Skenario pengujian performa disusun menggunakan pendekatan peningkatan beban pengguna secara bertahap (incremental load). Pengujian diawali dengan 1 pengguna sebagai kondisi *baseline*, kemudian dilanjutkan dengan 10, 20, dan 50 pengguna sebagai kategori *Load testing*, serta 100, 200, dan 500 pengguna sebagai kategori *Stress testing*. Pengelompokan tersebut bertujuan untuk mengamati perubahan karakteristik performa *Endpoint* mulai dari kondisi operasional normal hingga kondisi beban tinggi dalam satu rangkaian skenario pengujian performa. Pada Apache JMeter, jumlah pengguna virtual diatur menggunakan *Thread Group*, sedangkan pengiriman *Request* ke setiap *Endpoint* dilakukan menggunakan *HTTP Request*.

Parameter yang diukur dalam pengujian ini meliputi:

- a) *Response time*, yaitu waktu yang dibutuhkan sistem untuk memberikan *Response* terhadap *Request*.
- b) *Throughput*, yaitu jumlah *Request* yang dapat diproses oleh sistem dalam satuan waktu tertentu.
- c) *Error rate*, yaitu persentase *Request* yang gagal selama proses pengujian.
- d) Kestabilan sistem, yaitu konsistensi performa *Endpoint Dashboard* berdasarkan pola perubahan *Response time*, *Throughput*, dan *Error rate*.

Hasil dari tahap ini berupa rancangan skenario pengujian performa beserta konfigurasi test plan Apache JMeter yang digunakan untuk melaksanakan seluruh skenario pengujian, mulai dari kondisi *baseline*, kategori *Load testing*, hingga kategori *Stress testing*.

4. Konfigurasi Lingkungan Pengujian

Pada tahap ini dilakukan Konfigurasi lingkungan pengujian pada *server* lokal dan VPS UNUGHA. Kedua lingkungan menggunakan versi aplikasi, struktur basis data, jumlah data, *endpoint*, autentikasi, serta skenario Apache JMeter yang sama. *Server* digunakan untuk menjalankan Sistem CPL berbasis OBE beserta seluruh layanan pendukungnya, sedangkan laptop penguji digunakan sebagai mesin pengujian yang menjalankan Apache JMeter untuk menghasilkan simulasi pengguna virtual. Pemisahan perangkat *Server* dan laptop penguji dilakukan agar proses pembangkitan beban tidak memberikan tambahan beban komputasi pada

Server aplikasi sehingga hasil pengukuran lebih merepresentasikan kemampuan sistem dalam menangani permintaan pengguna.

Pengujian menggunakan dua jalur komunikasi jaringan yang berbeda. Pengujian server lokal dilakukan melalui jaringan lokal dengan konfigurasi LAN antara laptop penguji dan server lokal. Sementara itu, pengujian VPS UNUGHA dilakukan melalui jaringan internet dengan mengirimkan permintaan HTTP/API menuju domain obe.unugha.ac.id.

Apache JMeter dijalankan pada laptop penguji untuk mengirimkan permintaan kepada kedua lingkungan. Pemantauan CPU dan memori dilakukan menggunakan Glances melalui koneksi SSH pada masing-masing server. Perbedaan jalur jaringan tersebut menjadi bagian dari karakteristik lingkungan pengujian yang perlu diperhatikan dalam menginterpretasikan hasil *response time*.

Endpoint yang diuji pada penelitian ini meliputi:

- */api/dashboard/stats*
- */api/dashboard/dosen*
- */api/dashboard/students*

Ketiga *Endpoint* tersebut merupakan layanan *Dashboard Admin* yang membutuhkan proses autentikasi. Oleh karena itu, konfigurasi Apache JMeter mencakup proses login untuk memperoleh token JWT, ekstraksi token dari *Response* autentikasi, serta penggunaan token tersebut pada setiap *Request Endpoint Dashboard Admin*.

a. Spesifikasi Lingkungan Pengujian

Tabel berikut menunjukkan spesifikasi perangkat yang digunakan dalam proses pengujian

Tabel 11 Spesifikasi Lingkungan Pengujian

Komponen	Server Lokal	VPS UNUGHA
Prosesor	Intel Core 2 Duo P8400	Intel Xeon E5-2650 v4
RAM	3 GB	12 GB
Penyimpanan	SSD 256 GB	Partisi sistem 97,9 GB
Sistem operasi	Debian Server	Ubuntu 24.04 64-bit
Aplikasi	Sistem CPL berbasis OBE	Sistem CPL berbasis OBE
Backend	Node.js dan Express.js	Node.js dan Express.js
ORM	Prisma ORM	Prisma ORM
Basis data	MariaDB	MariaDB

Komponen	Server Lokal	VPS UNUGHA
Jalur akses pengujian	Jaringan lokal/LAN	Jaringan internet
Alamat akses	192.168.11.181:8081	obe.unugha.ac.id
Client penguji	Laptop Intel Core i7-6820HQ, RAM 16 GB, Debian	Laptop Intel Core i7-6820HQ, RAM 16 GB, Debian
Alat pengujian	Apache JMeter 5.6.3	Apache JMeter 5.6.3
Pemantauan	Glances melalui SSH	Glances melalui SSH

b. Konfigurasi Apache JMeter

Apache JMeter versi 5.6.3 digunakan sebagai alat pengujian performa untuk melakukan simulasi pengguna virtual terhadap *Endpoint Dashboard Admin*. Konfigurasi pengujian dibuat berdasarkan skenario peningkatan jumlah pengguna virtual mulai dari kondisi beban rendah hingga beban tinggi.

Komponen utama Apache JMeter yang digunakan dalam penelitian ini ditunjukkan pada Tabel 12.

Tabel 12 Konfigurasi Apache JMeter

Komponen JMeter	Fungsi
<i>Thread Group</i>	Mengatur jumlah pengguna virtual (virtual user), ramp-up period, durasi pengujian, dan pola eksekusi <i>Request</i>
<i>HTTP Request</i>	Mengirimkan <i>Request</i> HTTP ke <i>Endpoint</i> yang diuji sesuai alur transaksi pengguna
<i>HTTP Header Manager</i>	Menambahkan header <i>Request</i> seperti Content-Type dan token JWT pada <i>Endpoint</i> yang membutuhkan autentikasi
<i>JSON Extractor</i>	Mengambil token JWT dari <i>Response</i> proses login untuk digunakan pada <i>Request</i> berikutnya

Komponen JMeter	Fungsi
<i>Uniform Random Timer</i>	Mensimulasikan waktu tunggu pengguna (<i>think time</i>) dengan interval 1–3 detik

Berdasarkan komponen utama Apache JMeter pada Tabel 12, setiap komponen memiliki peran dalam membentuk test plan yang digunakan pada proses pengujian performa. Konfigurasi tersebut disusun untuk memastikan proses simulasi pengguna virtual dapat berjalan sesuai dengan skenario pengujian yang telah dirancang. Salah satu komponen utama dalam pembentukan skenario beban adalah *Thread Group*, karena komponen ini menentukan jumlah pengguna virtual, pola peningkatan beban, serta durasi eksekusi pengujian. Oleh karena itu, konfigurasi *Thread Group* pada penelitian ini dijelaskan secara lebih rinci pada bagian berikut.

1) Konfigurasi *Thread Group*

Thread Group digunakan untuk mengatur simulasi pengguna virtual pada Apache JMeter. Parameter yang dikonfigurasi meliputi jumlah pengguna virtual (virtual user), waktu peningkatan pengguna (ramp-up period), jumlah pengulangan *Request* (loop count), dan durasi pengujian.

Jumlah pengguna virtual yang diuji terdiri dari:

- 1 pengguna
- 10 pengguna
- 20 pengguna
- 50 pengguna
- 100 pengguna
- 200 pengguna
- 500 pengguna

Konfigurasi pengujian ditunjukkan pada Tabel 13.

Tabel 13 Konfigurasi *Thread Group* Apache JMeter

Virtual User	Ramp-up Period	Loop Count	Durasi
1	1 detik	Tidak Terbatas (Infinite)	61 detik
10	5 detik	Tidak Terbatas (Infinite)	65 detik
20	10 detik	Tidak Terbatas (Infinite)	70 detik
50	25 detik	Tidak Terbatas (Infinite)	85 detik
100	50 detik	Tidak Terbatas (Infinite)	110 detik
200	100 detik	Tidak Terbatas (Infinite)	160 detik
500	250 detik	Tidak Terbatas (Infinite)	310 detik

Penggunaan Loop Count Tidak Terbatas (Infinite) menunjukkan bahwa setiap pengguna virtual melakukan pengulangan *Request* selama durasi pengujian berlangsung. Mekanisme ramp-up period digunakan agar peningkatan jumlah pengguna berlangsung secara bertahap, kemudian sistem dipertahankan pada kondisi beban tersebut selama 60 detik (hold load) untuk mengamati kestabilan performa.

2) Konfigurasi *HTTP Request*

HTTP Request digunakan untuk menentukan *Request* HTTP yang dikirimkan oleh pengguna virtual menuju layanan *API* yang diuji. Pada penelitian ini, setiap pengguna virtual menjalankan rangkaian *Request* yang diawali dengan proses autentikasi untuk memperoleh token JWT, kemudian dilanjutkan dengan akses terhadap *Endpoint Dashboard Admin*.

Konfigurasi *HTTP Request* yang digunakan dalam penelitian ini ditunjukkan pada Tabel 14.

Tabel 14 Konfigurasi *HTTP Request* Apache JMeter

Parameter	Server Lokal	VPS UNUGHA
Target pengujian	192.168.11.181:8081	obe.unugha.ac.id
Jalur akses	Jaringan lokal (LAN)	Jaringan Internet
Method	POST dan GET	POST dan GET
Authentication Endpoint	/api/auth/login	/api/auth/login
Dashboard Endpoint	/api/dashboard/stats /api/dashboard/dosen /api/dashboard/students	/api/dashboard/stats /api/dashboard/dosen /api/dashboard/students
Mekanisme autentikasi	JWT	JWT

Pada proses pengujian, *Request* pertama yang dijalankan oleh setiap pengguna virtual adalah POST /api/auth/login untuk memperoleh token autentikasi. Setelah token JWT berhasil diperoleh, pengguna virtual melanjutkan *Request* GET menuju *Endpoint Dashboard* Admin sesuai dengan skenario pengujian. Setiap rangkaian *Request* tersebut dijalankan secara berulang selama durasi pengujian berdasarkan konfigurasi *Thread Group*.

3) Konfigurasi *HTTP Header Manager* dan *JSON Extractor*

HTTP Header Manager digunakan untuk menambahkan informasi header pada setiap *Request* yang dikirimkan oleh Apache JMeter. Konfigurasi ini diperlukan karena *Endpoint Dashboard* Admin menggunakan mekanisme autentikasi berbasis JWT.

Pada proses autentikasi, *Response* dari *Endpoint* login berupa data JSON yang berisi token JWT. Token tersebut kemudian diambil menggunakan komponen *JSON Extractor* untuk digunakan pada *Request* berikutnya.

Konfigurasi autentikasi ditunjukkan pada Tabel 15.

Tabel 15 Konfigurasi Header dan Ekstraksi Token JWT

Komponen	Konfigurasi	Fungsi
<i>HTTP Header Manager</i>	Content-Type: application/json	Menentukan format data <i>Request</i>
<i>HTTP Header Manager</i>	Accept: application/json	Menentukan format <i>Response</i> yang diterima

Komponen	Konfigurasi	Fungsi
<i>HTTP Header Manager</i>	Authorization: Bearer \${AUTH_TOKEN}	Mengirimkan token JWT pada <i>Endpoint Dashboard</i>
<i>JSON Extractor</i>	JSON Path: \$.token	Mengambil nilai token JWT dari <i>Response login</i>

Setelah token berhasil diekstraksi, nilai token disimpan dalam variabel \${AUTH_TOKEN} dan digunakan secara otomatis pada setiap *Request* menuju *Endpoint Dashboard* Admin. Mekanisme tersebut memastikan setiap *Request* yang dikirimkan oleh pengguna virtual memiliki autentikasi yang sesuai dengan mekanisme keamanan sistem.

4) Konfigurasi *Think Time*

Untuk mendekati pola interaksi pengguna sebenarnya, pengujian menggunakan mekanisme waktu tunggu (*think time*) antar *Request*. *Think time* digunakan untuk memberikan jeda sebelum pengguna virtual melanjutkan *Request* berikutnya sehingga pola beban yang dihasilkan tidak berupa pengiriman *Request* secara terus-menerus tanpa jeda.

Pada penelitian ini digunakan *Uniform Random Timer* dengan konfigurasi waktu tunggu antara 1 hingga 3 detik. Konfigurasi tersebut terdiri dari waktu tunggu awal sebesar 1000 ms dan tambahan waktu acak sebesar 2000 ms.

Konfigurasi *Think Time* ditunjukkan pada Tabel 16.

Tabel 16 Konfigurasi *Uniform Random Timer*

Parameter	Nilai
Komponen	<i>Uniform Random Timer</i>
Constant Delay Offset	1000 ms
Random Delay Maximum	2000 ms
Rentang <i>Think Time</i>	1–3 detik

Dengan konfigurasi tersebut, setiap pengguna virtual memiliki variasi waktu tunggu sebelum mengirimkan *Request* berikutnya sehingga simulasi beban lebih mendekati aktivitas pengguna pada kondisi nyata.

5) Mekanisme Pengumpulan Data Pengujian

Pengumpulan data hasil pengujian dilakukan menggunakan Apache JMeter dalam mode Non-GUI untuk mengurangi penggunaan sumber daya pada laptop penguji selama proses pembangkitan beban. Hasil eksekusi pengujian disimpan dalam format file .jtl (JMeter Test Log) yang berisi data transaksi selama pengujian berlangsung.

Data hasil pengujian tersebut kemudian digunakan untuk menghasilkan laporan HTML *Dashboard Report* yang memuat informasi performa sistem. Parameter yang digunakan dalam analisis penelitian meliputi:

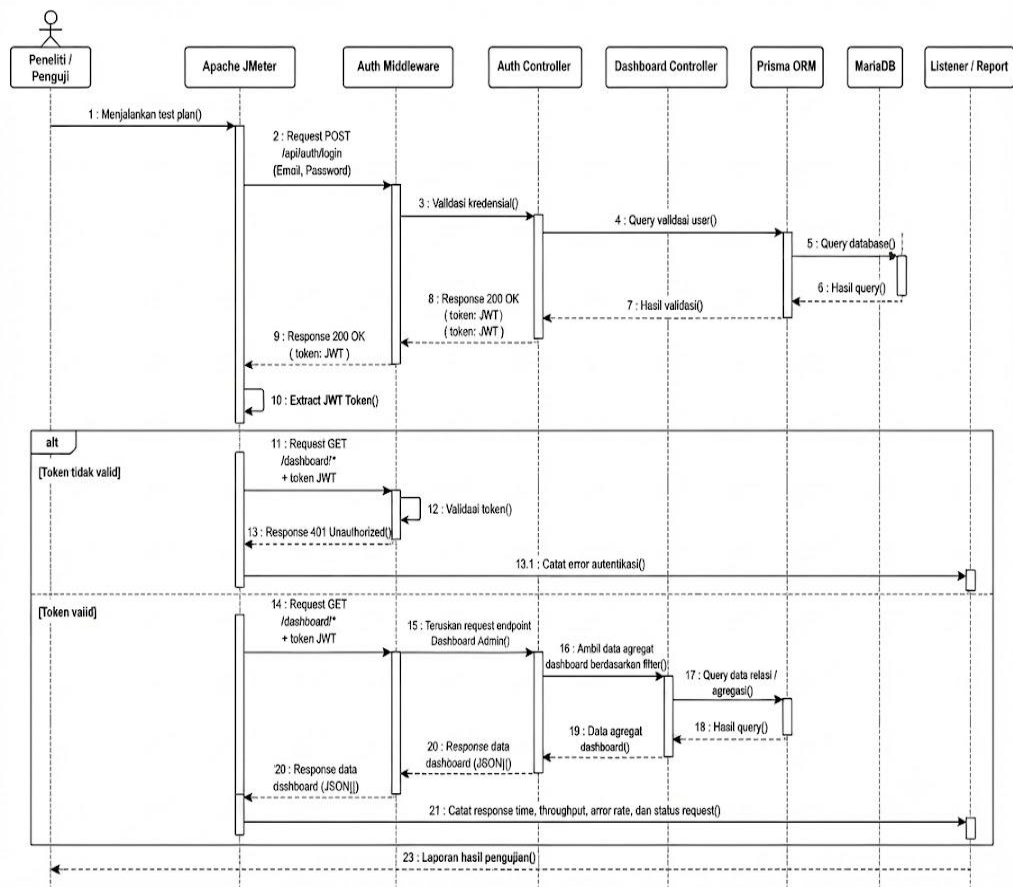
Tabel 17 Metrik Performa yang Dikumpulkan dari Pengujian Apache JMeter

Parameter	Keterangan
<i>Response time</i>	Waktu rata-rata yang dibutuhkan sistem dalam memberikan <i>Response</i> terhadap <i>Request</i>
<i>Throughput</i>	Jumlah <i>Request</i> yang berhasil diproses dalam satu satuan waktu
<i>Error rate</i>	Persentase <i>Request</i> yang mengalami kegagalan selama pengujian
Jumlah <i>Request</i>	Total <i>Request</i> yang dikirimkan selama proses pengujian

Data tersebut selanjutnya digunakan sebagai dasar evaluasi performa *Endpoint Dashboard Admin* berdasarkan kriteria operasional evaluasi performa pada Tabel 20.

c. Sequence Diagram Pengujian *Endpoint Dashboard Admin*

Sequence diagram pada Gambar 6 menggambarkan alur interaksi antar komponen dalam proses pengujian performa *Endpoint Dashboard Admin* menggunakan Apache JMeter. Diagram tersebut menjelaskan tahapan komunikasi antara Apache JMeter sebagai pembangkit beban (load generator) dengan komponen *Backend Sistem CPL*, mulai dari proses autentikasi pengguna virtual, pengiriman *Request* menuju *Endpoint Dashboard Admin*, proses pengolahan data, hingga pencatatan *Response* sebagai hasil pengujian performa.



Gambar 6 Sequence Diagram Pengujian Performa

Pada proses pengujian, Apache JMeter menjalankan test plan berdasarkan konfigurasi *Thread Group* yang telah ditentukan. Setiap pengguna virtual melakukan proses autentikasi melalui *Endpoint* login untuk memperoleh token JWT. *Response* hasil autentikasi kemudian diproses menggunakan *JSON Extractor* untuk mengambil nilai token yang akan digunakan pada *Request* berikutnya.

Setelah token JWT diperoleh, Apache JMeter mengirimkan *Request* menuju *Endpoint Dashboard Admin* dengan menyertakan token autentikasi melalui *HTTP Header Manager*. *Request* tersebut diterima oleh Auth Middleware untuk dilakukan validasi token. Apabila token tidak valid, sistem memberikan *Response* 401 Unauthorized dan kondisi tersebut tercatat sebagai *Request* gagal pada hasil pengujian.

Apabila token valid, *Request* diteruskan menuju *Dashboard Controller* untuk melakukan proses pengambilan data agregat. *Dashboard Controller* melakukan pengolahan data melalui Prisma ORM yang berkomunikasi dengan *Database MariaDB*. Setelah proses pengambilan data selesai, sistem mengirimkan *Response* dalam format JSON kembali kepada Apache JMeter.

Response yang diterima oleh Apache JMeter kemudian disimpan dalam hasil pengujian dan dianalisis berdasarkan parameter performa yang digunakan dalam penelitian, yaitu *Response time*, *Throughput*, *Error rate*, jumlah *Request*, dan status keberhasilan *Request*. Data tersebut selanjutnya menjadi dasar dalam mengevaluasi performa *Endpoint Dashboard Admin* pada berbagai tingkat beban pengguna serta mengidentifikasi indikasi dugaan *Bottleneck* pada sistem.

5. Pelaksanaan Pengujian Performa

Pelaksanaan Pengujian performa dilakukan terlebih dahulu pada *server* lokal, kemudian dilanjutkan pada VPS UNUGHA menggunakan skenario yang sama. Setiap skenario dijalankan secara bergantian dan hasilnya dicatat untuk proses analisis. Pengujian dilaksanakan menggunakan Apache JMeter dengan memberikan beban secara bertahap pada setiap *Endpoint*, sehingga perubahan performa sistem dapat diamati pada berbagai tingkat beban pengguna.

Pada pengujian server lokal, Apache JMeter mengirimkan permintaan menuju alamat API melalui jaringan lokal. Setelah seluruh skenario server lokal selesai, pengujian dilanjutkan terhadap VPS UNUGHA. Pada tahap tersebut, laptop penguji dihubungkan ke internet dan Apache JMeter mengirimkan permintaan HTTP/API melalui domain *obe.unugha.ac.id*. Kedua lingkungan tetap menggunakan versi aplikasi, data, *endpoint*, konfigurasi skenario, dan parameter pengukuran yang sama.

Pada penelitian ini, *Load testing* dan *Stress testing* tidak dilaksanakan sebagai dua tahapan pengujian yang terpisah, melainkan sebagai satu rangkaian pengujian performa. Pendekatan ini dipilih agar pengujian mampu menggambarkan perubahan karakteristik performa sistem secara menyeluruh, mulai dari kondisi beban rendah hingga beban tinggi. Hasil pengujian yang diperoleh selanjutnya digunakan sebagai dasar untuk menganalisis parameter performa, mengidentifikasi dugaan *Bottleneck*, serta menyusun rekomendasi perbaikan.

Pelaksanaan pengujian diawali dengan pengujian *baseline* menggunakan satu pengguna virtual untuk memperoleh gambaran performa awal sistem. Selanjutnya, jumlah pengguna virtual ditingkatkan secara bertahap menjadi 10, 20, 50, 100, 200, dan 500 pengguna. Peningkatan beban dilakukan secara bertingkat agar perubahan performa pada setiap *Endpoint* dapat diamati secara sistematis, sekaligus memudahkan identifikasi titik awal terjadinya penurunan performa. Rincian skenario pengujian yang digunakan dalam penelitian ini disajikan pada Tabel 18.

Tabel 18 Skenario Pengujian Performa

Tingkat Beban Pengguna	Kategori Pengujian	Tujuan
1 User	<i>Baseline</i>	Mengetahui performa awal <i>Endpoint</i>

Tingkat Beban Pengguna	Kategori Pengujian	Tujuan
10 User	<i>Load testing</i>	Mengamati perubahan performa pada beban rendah
20 User	<i>Load testing</i>	Mengamati perubahan performa pada beban operasional
50 User	<i>Load testing</i>	Mengidentifikasi awal penurunan performa
100 User	<i>Stress testing</i>	Mengevaluasi performa pada beban tinggi
200 User	<i>Stress testing</i>	Mengamati kestabilan <i>Endpoint</i> pada beban tinggi
500 User	<i>Stress testing</i>	Mengidentifikasi indikasi batas kemampuan <i>Endpoint</i>

Sumber: Diolah oleh penulis (2026)

Selain merekam metrik performa menggunakan Apache JMeter, penelitian ini juga melakukan pemantauan terhadap penggunaan sumber daya *Server* selama proses pengujian berlangsung. Pemantauan dilakukan melalui koneksi *Secure Shell* (SSH) menuju *Server* pengujian dengan memanfaatkan utilitas Glances. Parameter yang diamati meliputi utilisasi CPU dan penggunaan memori (RAM). Data penggunaan sumber daya direkam melalui tangkapan layar (screenshot) pada saat kondisi beban puncak (*Peak load*) untuk setiap skenario jumlah pengguna virtual. Selanjutnya, data tersebut dikonversi menjadi data kuantitatif yang digunakan sebagai informasi pendukung dalam analisis parameter performa, khususnya untuk mengidentifikasi dugaan *Bottleneck* serta menyusun rekomendasi perbaikan sistem.

6. Pengumpulan Data Hasil Pengujian

Setelah seluruh skenario pengujian performa selesai dilaksanakan, tahap berikutnya adalah pengumpulan data hasil pengujian. Data penelitian diperoleh dari dua sumber utama, yaitu hasil pengujian performa menggunakan Apache JMeter dan hasil pemantauan sumber daya *Server* menggunakan Glances.

Data yang diperoleh dari Apache JMeter berupa metrik performa *Endpoint* yang diuji, meliputi *Response time*, *Throughput*, *Error rate*, jumlah *Request*, dan status keberhasilan *Request* pada setiap variasi jumlah pengguna virtual. Data tersebut diperoleh dari hasil eksekusi pengujian yang telah disimpan dalam laporan hasil pengujian JMeter.

Selain data dari Apache JMeter, dilakukan pengumpulan data pendukung berupa penggunaan sumber daya *Server* menggunakan Glances. Data yang diamati meliputi utilisasi CPU dan penggunaan memori (RAM) selama proses pengujian, terutama pada kondisi beban tinggi (*peak load*). Data tersebut digunakan untuk membantu menganalisis kondisi sistem ketika terjadi perubahan performa akibat peningkatan jumlah pengguna virtual.

Seluruh data hasil pengujian kemudian didokumentasikan dan dikelompokkan berdasarkan *Endpoint* yang diuji serta skenario jumlah pengguna virtual. Pengelompokan tersebut dilakukan untuk mempermudah proses perbandingan dan analisis setiap parameter performa. Data yang telah terkumpul selanjutnya digunakan sebagai dasar evaluasi performa sistem, analisis kestabilan layanan, identifikasi indikasi dugaan *Bottleneck*, serta penyusunan rekomendasi perbaikan berdasarkan hasil pengujian.

7. Analisis Parameter Performa

Pada tahap ini dilakukan analisis terhadap data hasil pengujian yang diperoleh dari *server* lokal dan VPS UNUGHA. Data yang dianalisis meliputi hasil pengujian menggunakan Apache JMeter serta data pemantauan penggunaan sumber daya *server* menggunakan Glances. Analisis difokuskan pada tiga *endpoint Dashboard Admin*, yaitu `/api/dashboard/stats`, `/api/dashboard/dosen`, dan `/api/dashboard/students`.

Analisis dilakukan dengan membandingkan hasil pada setiap skenario pengguna virtual, yaitu 1, 10, 20, 50, 100, 200, dan 500 pengguna. Perbandingan dilakukan dalam dua tahap, yaitu membandingkan perubahan performa pada masing-masing *server* dan membandingkan hasil antara *server* lokal dan VPS UNUGHA pada skenario yang sama. Parameter yang digunakan meliputi *response time*, *throughput*, *error rate*, kestabilan sistem, penggunaan CPU, dan penggunaan memori.

Parameter *response time* digunakan untuk mengetahui perubahan waktu yang dibutuhkan setiap *endpoint* dalam memproses *request*. Parameter *throughput* digunakan untuk mengevaluasi jumlah *request* yang dapat diproses per satuan waktu, sedangkan *error rate* digunakan untuk mengetahui persentase *request* yang gagal. Kestabilan sistem dianalisis berdasarkan pola perubahan ketiga parameter tersebut ketika beban pengguna meningkat.

Data penggunaan CPU dan memori digunakan sebagai indikator pendukung dalam mengidentifikasi dugaan *bottleneck* pada masing-masing *server*. Karena *server* lokal dan VPS UNUGHA memiliki spesifikasi perangkat keras yang berbeda, perbandingan penggunaan sumber daya digunakan untuk menjelaskan karakteristik performa setiap lingkungan dan tidak dimaksudkan sebagai perbandingan perangkat keras secara terkontrol.

Parameter yang digunakan dalam analisis performa beserta bentuk analisis yang dilakukan disajikan pada Tabel 19.

Tabel 19 Parameter Analisis Performa

No	Parameter	Bentuk Analisis	Indikasi yang Diamati
1	<i>Response time</i>	Membandingkan waktu respons pada setiap skenario pengguna	Kenaikan waktu respons ketika beban meningkat
2	<i>Throughput</i>	Membandingkan jumlah <i>Request</i> yang diproses per satuan waktu	Kapasitas pemrosesan <i>Endpoint</i> terhadap beban pengguna
3	<i>Error rate</i>	Membandingkan persentase <i>Request</i> gagal pada setiap skenario	Kegagalan <i>Request</i> pada beban tertentu
4	Kestabilan sistem	Mengamati pola perubahan <i>Response time</i> , <i>Throughput</i> , dan <i>Error rate</i>	Konsistensi performa saat jumlah pengguna meningkat
5	Utilisasi CPU dan Memori	Membandingkan penggunaan sumber daya <i>Server</i> pada setiap skenario	Indikasi keterbatasan sumber daya dan dugaan <i>Bottleneck</i>

Hasil analisis digunakan untuk menilai kemampuan setiap *endpoint* dalam mempertahankan performa pada masing-masing lingkungan *server*. Hasil tersebut juga digunakan untuk mengetahui perbedaan karakteristik performa antara *server* lokal dan VPS UNUGHA pada skenario pengujian yang sama.

Untuk menginterpretasikan hasil analisis pada setiap parameter, penelitian ini menggunakan kriteria operasional evaluasi performa sebagai instrumen evaluasi. Kriteria tersebut digunakan sebagai pedoman dalam mengklasifikasikan kondisi performa *Endpoint* pada setiap skenario pengujian sehingga hasil analisis dapat diinterpretasikan secara objektif dan konsisten. Evaluasi difokuskan pada mekanisme *Request-Response API* dalam menghasilkan data agregat dari *Backend*, sehingga tidak mencakup aspek *Frontend*, proses *rendering* pada browser, maupun komponen lain di luar *Endpoint API* yang diuji [65].

Kriteria operasional evaluasi performa disusun berdasarkan karakteristik Performance Efficiency dan Reliability pada ISO/IEC 25010 [66], serta sintesis berbagai penelitian mengenai evaluasi performa aplikasi web dan *API* menggunakan *Load testing* [20], [67]. Kriteria tersebut tidak dimaksudkan sebagai standar baku yang berlaku secara universal, melainkan sebagai kriteria operasional yang disesuaikan dengan karakteristik objek penelitian.

Berdasarkan landasan tersebut, disusun kriteria operasional evaluasi yang mengintegrasikan parameter *Response time*, *Throughput*, *Error rate*, dan kestabilan sistem sehingga evaluasi performa dapat dilakukan secara menyeluruh terhadap *Endpoint Dashboard Admin*. Kriteria operasional evaluasi performa yang digunakan pada penelitian ini disajikan pada Tabel 20.

Tabel 20 Kriteria Operasional Evaluasi Performa

Parameter	Baik	Cukup	Kurang	Buruk / Tidak Stabil
<i>Response time</i>	≤ 3 detik	$> 3-5$ detik	$> 5-10$ detik	> 10 detik
<i>Error rate</i>	0%	$> 0-1\%$	$> 1-5\%$	$> 5\%$
<i>Throughput</i>	<i>Throughput</i> meningkat secara proporsional seiring bertambahnya jumlah pengguna virtual.	<i>Throughput</i> masih meningkat, namun laju peningkatannya mulai melambat atau mendekati kondisi jenuh (saturation) pada beban tinggi.	<i>Throughput</i> tidak lagi meningkat secara berarti (stagnan) atau mulai mengalami penurunan meskipun jumlah pengguna terus bertambah.	<i>Throughput</i> menurun secara signifikan akibat kegagalan layanan, timeout, crash, atau sistem tidak mampu mempertahankan proses <i>request</i> .
Kestabilan Sistem	Seluruh <i>request</i> berhasil	Seluruh <i>request</i> masih berhasil diproses, tetapi	<i>Response time</i> meningkat signifikan dan	<i>Response time</i> telah berada pada kategori

Parameter	Baik	Cukup	Kurang	Buruk / Tidak Stabil
	diproses tanpa <i>error</i> , <i>response time</i> masih memenuhi kriteria baik, dan <i>throughput</i> mampu dipertahankan selama pengujian.	terjadi peningkatan <i>response time</i> atau perlambatan peningkatan <i>throughput</i> seiring bertambahnya beban.	<i>throughput</i> mulai stagnan atau menurun sehingga kualitas layanan mengalami penurunan, meskipun <i>request</i> masih dapat diproses.	buruk (>10 detik) disertai <i>throughput</i> yang stagnan atau menurun secara nyata, atau terjadi kegagalan <i>request</i> , <i>timeout</i> , <i>crash</i> , maupun sistem tidak mampu mempertahankan layanan pada beban tinggi.

Sumber : Disusun peneliti berdasarkan karakteristik Performance Efficiency dan Reliability pada ISO/IEC 25010 [65], serta sintesis penelitian mengenai evaluasi performa aplikasi web dan *API* menggunakan *Load testing* [20], [66], [67].

Kriteria pada Tabel 20 digunakan untuk mengevaluasi hasil pengujian pada *server* lokal dan VPS UNUGHA. Hasil evaluasi selanjutnya digunakan untuk menentukan kategori performa setiap *endpoint*, mengidentifikasi dugaan *bottleneck* pada masing-masing lingkungan, serta menyusun rekomendasi perbaikan *Dashboard Admin Sistem CPL*.

8. Identifikasi Dugaan *Bottleneck*

Identifikasi dugaan *bottleneck* dilakukan berdasarkan hasil pengujian pada *server* lokal dan VPS UNUGHA. Analisis dilakukan dengan menghubungkan perubahan *response time*, *throughput*, *error rate*, penggunaan CPU, dan penggunaan memori pada setiap skenario pengujian.

Identifikasi dilakukan secara terpisah pada masing-masing *server* karena keduanya memiliki spesifikasi perangkat keras dan konfigurasi lingkungan yang berbeda. Hasil identifikasi kemudian dibandingkan untuk mengetahui faktor yang diduga memengaruhi penurunan performa pada setiap lingkungan.

Identifikasi dalam penelitian ini bersifat dugaan teknis berdasarkan hasil pengujian dan pemantauan sumber daya *server*. Penelitian tidak mencakup *code profiling*, penelusuran *query* secara langsung, maupun pengujian terhadap implementasi optimasi.

Faktor yang dianalisis meliputi proses agregasi pada *backend*, kompleksitas *query* basis data, relasi antartabel, penggunaan *filter*, ukuran *payload response*, antrean *request*, serta penggunaan sumber daya *server*. Faktor-faktor tersebut dianalisis karena *endpoint Dashboard Admin* tidak hanya mengambil data dari basis data, tetapi juga melakukan pengolahan dan agregasi sebelum respons dikirimkan kepada *frontend*. Kriteria identifikasi dugaan *bottleneck* disajikan pada Tabel 21.

Tabel 21 Kriteria Identifikasi Dugaan *Bottleneck* Berdasarkan Hasil Pengujian

No	Indikasi Hasil Pengujian	Interpretasi Hasil	Dugaan Faktor Penyebab
1	Salah satu <i>endpoint</i> memiliki <i>response time</i> lebih tinggi dibandingkan <i>endpoint</i> lainnya	<i>Endpoint</i> membutuhkan waktu pemrosesan lebih lama dan memiliki proses yang lebih kompleks	Proses agregasi pada <i>backend</i> , kompleksitas <i>query</i> , relasi antartabel, penggunaan <i>filter</i> , atau ukuran <i>payload response</i>
2	<i>Response time</i> meningkat secara signifikan ketika jumlah pengguna bertambah	Kapasitas pemrosesan mulai mendekati batas kemampuan sistem	Peningkatan beban <i>backend</i> , proses basis data, atau antrean <i>request</i>
3	<i>Throughput</i> tidak meningkat secara proporsional terhadap penambahan pengguna	Penambahan beban tidak diikuti peningkatan jumlah <i>request</i> yang dapat diselesaikan	Kapasitas pemrosesan <i>endpoint</i> atau sumber daya <i>server</i> mulai mencapai kondisi jenuh
4	<i>Error rate</i> tetap rendah, tetapi <i>response time</i> meningkat	Sistem masih berhasil memproses <i>request</i> , tetapi membutuhkan waktu lebih lama	Antrean <i>request</i> , proses agregasi yang berat, atau keterbatasan kapasitas pemrosesan
5	<i>Error rate</i> meningkat pada skenario tertentu	Sistem mulai mengalami kegagalan dalam mempertahankan layanan	Timeout, kegagalan layanan, keterbatasan koneksi, atau ketidakstabilan <i>backend</i>

No	Indikasi Hasil Pengujian	Interpretasi Hasil	Dugaan Faktor Penyebab
6	<i>Response time</i> meningkat dan <i>throughput</i> stagnan atau menurun pada beban tinggi	Sistem mulai mengalami kejenuhan dan penurunan kestabilan performa	Keterbatasan sumber daya <i>server</i> , proses <i>backend</i> , atau beban <i>query</i> yang semakin tinggi
7	Penggunaan CPU meningkat bersamaan dengan peningkatan <i>response time</i>	Proses komputasi menjadi salah satu faktor pembatas performa	Proses Node.js, pengolahan data agregat, atau eksekusi <i>query</i> yang membutuhkan CPU tinggi
8	Penggunaan memori mendekati kapasitas maksimum atau meningkat secara terus-menerus	Kapasitas memori berpotensi membatasi proses aplikasi	Penggunaan memori aplikasi, data yang diproses, <i>caching</i> , atau pengelolaan memori yang kurang efisien

Selain berdasarkan indikasi umum pada Tabel 21, analisis juga diarahkan pada karakteristik masing-masing *endpoint* dan lingkungan *server*. Hal ini dilakukan agar dugaan *bottleneck* dapat dikaitkan dengan proses *Dashboard* Admin yang diuji serta kondisi sumber daya pada *server* lokal dan VPS UNUGHA.

Tabel 22 Karakteristik Teknis *Endpoint* yang Dianalisis

<i>Endpoint</i>	Karakteristik Proses <i>Backend</i>	Parameter yang Menjadi Fokus Analisis
/api/dashboard/stats	Agregasi statistik <i>Dashboard</i>	<i>Response time</i> , <i>Throughput</i> , <i>Error rate</i>
/api/dashboard/dosen	Agregasi data dosen dan progres nilai	<i>Response time</i> , <i>Throughput</i> , <i>Error rate</i>
/api/dashboard/students	Agregasi data evaluasi mahasiswa	<i>Response time</i> , <i>Throughput</i> , <i>Error rate</i>

Selain analisis terhadap masing-masing *endpoint*, penelitian ini juga mengevaluasi kestabilan sistem secara keseluruhan dengan mengintegrasikan nilai

response time, throughput, error rate, penggunaan CPU, dan penggunaan memori pada setiap skenario pengujian. Evaluasi dilakukan secara terpisah pada *server* lokal dan VPS UNUGHA, kemudian hasil keduanya dibandingkan untuk memperoleh gambaran mengenai kemampuan sistem dalam mempertahankan performa pada lingkungan yang berbeda.

Hasil identifikasi dugaan *bottleneck* selanjutnya digunakan sebagai dasar dalam penyusunan rekomendasi perbaikan. Dengan demikian, rekomendasi yang diberikan didasarkan pada hasil pengujian empiris dan kondisi sumber daya pada masing-masing *server*.

9. Penyusunan Rekomendasi Perbaikan

Penyusunan rekomendasi perbaikan dilakukan setelah proses analisis performa dan identifikasi dugaan *bottleneck* pada *server* lokal dan VPS UNUGHA. Tahap ini diawali dengan menentukan *endpoint* dan skenario pengujian yang menunjukkan penurunan performa berdasarkan nilai *response time, throughput, error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem.

Selanjutnya, hasil pengujian dikaitkan dengan karakteristik proses masing-masing *endpoint* dan kondisi lingkungan *server*. Berdasarkan hubungan tersebut, disusun rekomendasi teknis yang sesuai dengan dugaan penyebab penurunan performa.

Rekomendasi disusun secara terpisah untuk setiap temuan agar sesuai dengan kondisi *server* lokal dan VPS UNUGHA. Rekomendasi dalam penelitian ini bersifat arahan perbaikan dan tidak mencakup implementasi maupun pengujian ulang setelah optimasi.

Hasil tahap ini berupa rekomendasi teknis yang didasarkan pada temuan empiris hasil pengujian dan dapat digunakan sebagai dasar pengembangan sistem selanjutnya.

10. Penarikan Kesimpulan

Penarikan kesimpulan dilakukan berdasarkan seluruh hasil analisis pengujian pada *server* lokal dan VPS UNUGHA. Kesimpulan disusun untuk menjawab rumusan masalah mengenai performa *endpoint Dashboard Admin*, titik awal penurunan performa, perbedaan hasil pada kedua lingkungan *server*, serta dugaan *bottleneck* yang ditemukan.

Kesimpulan didasarkan pada hasil pengujian terhadap *endpoint /api/dashboard/stats, /api/dashboard/dosen, dan /api/dashboard/students* pada skenario 1, 10, 20, 50, 100, 200, dan 500 pengguna virtual. Parameter yang menjadi dasar penarikan kesimpulan meliputi *response time, throughput, error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem.

Hasil pengujian dianalisis pada masing-masing *server*, kemudian dibandingkan pada skenario yang sama. Berdasarkan hasil tersebut, ditentukan kondisi performa setiap *endpoint*, skenario yang menunjukkan penurunan performa, serta faktor yang diduga menjadi penyebab *bottleneck*.

Hasil akhir penelitian berupa kesimpulan mengenai performa *Dashboard Admin* pada *server* lokal dan VPS UNUGHA serta rekomendasi perbaikan yang disusun berdasarkan temuan hasil pengujian.

D. Analisis Data

Analisis data dilakukan terhadap hasil pengujian Apache JMeter dan data pemantauan sumber daya *server* menggunakan Glances. Analisis difokuskan pada *endpoint* `/api/dashboard/stats`, `/api/dashboard/dosen`, dan `/api/dashboard/students`.

Analisis dilakukan dalam dua tahap. Pertama, hasil setiap skenario pada *server* lokal dan VPS UNUGHA dianalisis secara terpisah. Kedua, hasil kedua *server* dibandingkan pada skenario pengguna virtual yang sama. Parameter yang dianalisis meliputi *response time*, *throughput*, *error rate*, kestabilan sistem, penggunaan CPU, dan penggunaan memori.

Perbandingan hasil menggambarkan performa *end-to-end* pada masing-masing lingkungan aktual. Pengukuran server lokal mencakup komunikasi melalui jaringan lokal, sedangkan pengukuran VPS UNUGHA mencakup komunikasi melalui jaringan internet dari laptop penguji menuju domain `obe.unugha.ac.id` hingga respons diterima kembali. Oleh karena itu, nilai *response time* VPS UNUGHA mencakup waktu pemrosesan aplikasi dan latensi jaringan. Penelitian ini tidak memisahkan kedua komponen waktu tersebut.

Tahapan analisis data terdiri atas analisis parameter performa, identifikasi dugaan *bottleneck*, dan penyusunan rekomendasi perbaikan.

1. Analisis Parameter Performa

Analisis parameter performa dilakukan menggunakan kriteria operasional evaluasi yang telah ditetapkan. Hasil setiap parameter dibandingkan pada skenario 1, 10, 20, 50, 100, 200, dan 500 pengguna virtual, baik pada *server* lokal maupun VPS UNUGHA.

Tahapan analisis parameter performa meliputi:

a. Analisis *Response time*

Analisis *Response time* dilakukan untuk mengetahui waktu yang dibutuhkan sistem dalam memberikan respons terhadap *Request* yang dikirimkan. Nilai *Response time* pada setiap skenario dibandingkan untuk melihat perubahan performa seiring meningkatnya jumlah pengguna virtual

b. Analisis *Throughput*

Analisis *Throughput* dilakukan untuk mengetahui jumlah *Request* yang dapat diproses sistem dalam satuan waktu tertentu. Nilai *Throughput* digunakan untuk mengevaluasi kemampuan sistem dalam menangani peningkatan beban pengguna.

c. Analisis *Error rate*

Analisis *Error rate* dilakukan untuk mengetahui persentase *Request* yang gagal diproses oleh sistem selama pengujian berlangsung. Peningkatan *Error rate* digunakan sebagai salah satu indikator penurunan performa sistem.

d. Analisis Penggunaan CPU dan Memori

Analisis penggunaan CPU dan memori dilakukan berdasarkan data hasil pemantauan menggunakan *Glances* pada setiap skenario pengujian. Data utilisasi CPU dan penggunaan memori kemudian dikorelasikan dengan parameter *Response time*, *Throughput*, dan *Error rate* untuk membantu mengidentifikasi indikasi keterbatasan sumber daya *Server* maupun dugaan penyebab penurunan performa sistem.

e. Analisis Kestabilan Sistem

Analisis kestabilan sistem dilakukan dengan mengamati konsistensi *Response time*, *Throughput*, dan *Error rate* pada setiap skenario pengujian. Sistem dianggap mengalami penurunan kestabilan apabila terjadi peningkatan *Response time* yang signifikan, penurunan *Throughput*, atau peningkatan *Error rate* pada tingkat beban tertentu.

2. Identifikasi Dugaan *Bottleneck*

Identifikasi dugaan *bottleneck* dilakukan dengan menghubungkan hasil analisis *response time*, *throughput*, *error rate*, penggunaan CPU, dan penggunaan memori pada setiap *endpoint*. Identifikasi dilakukan secara terpisah pada *server* lokal dan VPS UNUGHA karena kedua lingkungan memiliki spesifikasi dan konfigurasi yang berbeda.

Hasil kedua *server* kemudian dibandingkan untuk mengetahui kesamaan atau perbedaan pola penurunan performa. Identifikasi ini bersifat dugaan teknis berdasarkan hasil pengujian empiris dan tidak mencakup *code profiling* maupun penelusuran *query* secara langsung.

3. Penyusunan Rekomendasi Perbaikan

Rekomendasi perbaikan disusun berdasarkan hasil evaluasi performa dan identifikasi dugaan *bottleneck*. Setiap rekomendasi dikaitkan dengan temuan pada *endpoint* dan lingkungan *server* yang mengalami penurunan performa.

Rekomendasi dalam penelitian ini bersifat arahan teknis dan tidak mencakup implementasi maupun pengujian ulang setelah optimasi.

Hasil analisis disajikan dalam bentuk tabel dan uraian pembahasan untuk menggambarkan performa setiap *endpoint* pada *server* lokal dan VPS UNUGHA. Hasil tersebut digunakan untuk menentukan kategori performa, mengidentifikasi dugaan *bottleneck*, serta menyusun rekomendasi perbaikan sesuai temuan penelitian.

Hasil analisis data disajikan dalam bentuk tabel dan uraian pembahasan untuk menggambarkan performa setiap *endpoint* pada *server* lokal dan VPS UNUGHA dalam seluruh skenario pengujian. Hasil tersebut dievaluasi menggunakan kriteria operasional yang telah ditetapkan untuk menentukan kategori performa, mengidentifikasi dugaan *bottleneck*, serta menjadi dasar penyusunan rekomendasi perbaikan.

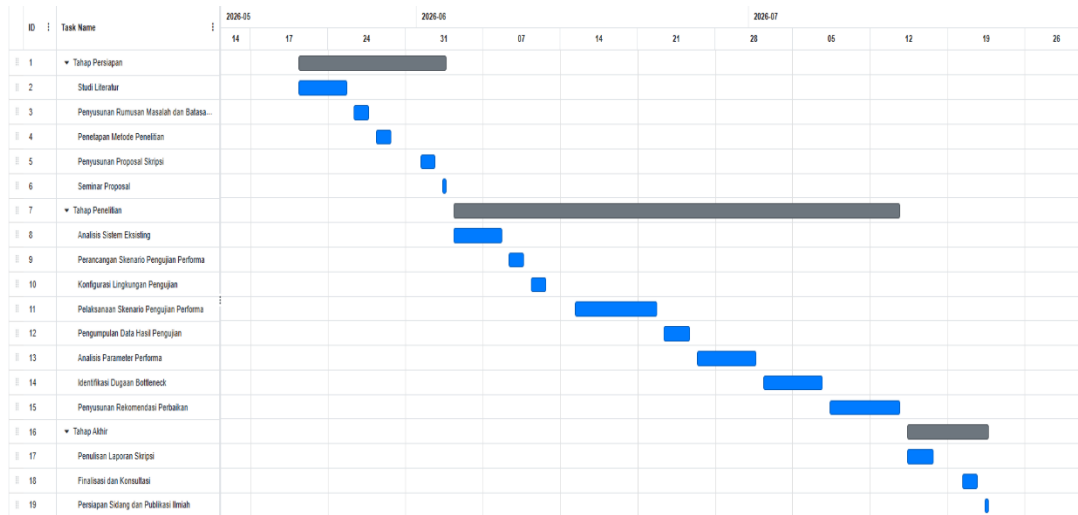
E. Jadwal Penelitian

Jadwal penelitian disusun untuk menggambarkan tahapan kegiatan mulai dari penyusunan proposal hingga penyelesaian laporan akhir. Penelitian dilaksanakan selama kurang lebih 45 hari kerja dan dibagi menjadi tiga tahap, yaitu tahap persiapan, tahap penelitian, dan tahap akhir.

Tahap persiapan mencakup studi literatur, penyusunan rumusan masalah dan batasan penelitian, penetapan metode, penyusunan proposal, serta seminar proposal. Tahap penelitian mencakup analisis sistem eksisting, perancangan skenario pengujian, konfigurasi lingkungan pengujian, pelaksanaan pengujian pada *server* lokal dan VPS UNUGHA, pengumpulan data, analisis dan perbandingan performa, identifikasi dugaan *bottleneck*, serta penyusunan rekomendasi perbaikan.

Tahap akhir mencakup penulisan laporan skripsi, konsultasi dan finalisasi, persiapan sidang, serta publikasi ilmiah.

Rincian waktu dan pembagian kegiatan penelitian ditunjukkan pada Gambar 7.



Gambar 7 Jadwal Penelitian

BAB IV HASIL DAN PEMBAHASAN

A. Hasil

Pada bagian ini disajikan hasil pengujian performa *Dashboard* Admin Sistem Capaian Pembelajaran Lulusan (CPL) berbasis *Outcome Based Education* (OBE) menggunakan Apache JMeter pada *server* lokal dan VPS UNUGHA. Pengujian dilakukan terhadap tiga *endpoint* *Dashboard* Admin, yaitu */api/dashboard/stats*, */api/dashboard/dosen*, dan */api/dashboard/students*, menggunakan skenario 1, 10, 20, 50, 100, 200, dan 500 pengguna virtual.

Hasil pengujian meliputi nilai response time, throughput, error rate, serta data penggunaan CPU dan memori. Data pada server lokal dan VPS UNUGHA disajikan secara berdampingan pada setiap skenario pengguna agar hasil kedua lingkungan dapat dibandingkan secara langsung.

Hasil pada bagian ini disajikan dalam bentuk tabel perbandingan dan klasifikasi performa berdasarkan kriteria operasional yang telah ditetapkan pada Bab III. Analisis lebih lanjut, identifikasi dugaan bottleneck, dan penyusunan rekomendasi perbaikan disajikan pada bagian pembahasan.

1. Hasil Pengujian *Endpoint* */api/dashboard/stats*

Pengujian *endpoint* */api/dashboard/stats* dilakukan untuk memperoleh data performa dalam melayani permintaan statistik *Dashboard* Admin, seperti statistik umum, grafik capaian pembelajaran, tren semester, alert, dan informasi pendukung lainnya.

Hasil pengujian *endpoint* */api/dashboard/stats* pada server lokal dan VPS UNUGHA disajikan secara berdampingan pada Tabel 23.

Tabel 23 Perbandingan Hasil Pengujian *Endpoint* */api/dashboard/stats* pada Server Lokal dan VPS UNUGHA

No	Pengguna virtual	Server lokal	VPS UNUGHA
1	1	<i>Response time</i> : 13,0 ms (Baik) <i>Throughput</i> : 0,14 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)	<i>Response time</i> : 100,3 ms (Baik) <i>Throughput</i> : 0,15 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)
2	10	<i>Response time</i> : 35,4 ms (Baik) <i>Throughput</i> : 1,23 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)	<i>Response time</i> : 131,6 ms (Baik) <i>Throughput</i> : 1,25 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)
3	20	<i>Response time</i> : 69,0 ms (Baik) <i>Throughput</i> : 2,37 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)	<i>Response time</i> : 204,7 ms (Baik) <i>Throughput</i> : 2,23 req/s (Baik) <i>Error rate</i> : 0,0% (Baik)

No	Pengguna virtual	Server lokal	VPS UNUGHA
4	50	<i>Response time:</i> 613,2 ms (Baik) <i>Throughput:</i> 4,16 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 324,9 ms (Baik) <i>Throughput:</i> 4,84 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
5	100	<i>Response time:</i> 2.484,1 ms (Baik) <i>Throughput:</i> 4,41 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 1.349,1 ms (Baik) <i>Throughput:</i> 5,96 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)
6	200	<i>Response time:</i> 5.987,6 ms (Kurang) <i>Throughput:</i> 4,41 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 3.692,5 ms (Cukup) <i>Throughput:</i> 6,18 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)
7	500	<i>Response time:</i> 15.548,8 ms (Buruk / Tidak Stabil) <i>Throughput:</i> 4,40 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 10.532,2 ms (Buruk/Tidak Stabil) <i>Throughput:</i> 6,15 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)

Berdasarkan Tabel 23, hasil server lokal dan VPS UNUGHA dapat dibandingkan secara langsung pada setiap skenario pengguna. Data tersebut selanjutnya dianalisis pada bagian pembahasan.

2. Hasil Pengujian *Endpoint* /api/dashboard/dosen

Pengujian *endpoint* /api/dashboard/dosen dilakukan untuk memperoleh data performa dalam melayani permintaan informasi dosen pada *Dashboard Admin*. *Endpoint* ini menyajikan data dosen, mata kuliah yang diampu, dan progres pengisian nilai.

Hasil pengujian *endpoint* /api/dashboard/dosen pada server lokal dan VPS UNUGHA disajikan secara berdampingan pada Tabel 24.

Tabel 24 Perbandingan Hasil Pengujian *Endpoint* /api/dashboard/dosen pada Server Lokal dan VPS UNUGHA

No	Pengguna virtual	Server lokal	VPS UNUGHA
1	1	<i>Response time:</i> 4,7 ms (Baik) <i>Throughput:</i> 0,14 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 52,3 ms (Baik) <i>Throughput:</i> 0,15 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
2	10	<i>Response time:</i> 29,0 ms (Baik) <i>Throughput:</i> 1,23 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 84,4 ms (Baik) <i>Throughput:</i> 1,18 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)

No	Pengguna virtual	Server lokal	VPS UNUGHA
3	20	<i>Response time:</i> 43,4 ms (Baik) <i>Throughput:</i> 2,39 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 141,8 ms (Baik) <i>Throughput:</i> 2,24 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
4	50	<i>Response time:</i> 521,4 ms (Baik) <i>Throughput:</i> 4,12 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 252,4 ms (Baik) <i>Throughput:</i> 4,86 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
5	100	<i>Response time:</i> 2.482,0 ms (Baik) <i>Throughput:</i> 4,24 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 1.233,8 ms (Baik) <i>Throughput:</i> 5,87 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)
6	200	<i>Response time:</i> 6.019,6 ms (Kurang) <i>Throughput:</i> 4,17 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 3.697,7 ms (Cukup) <i>Throughput:</i> 6,01 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)
7	500	<i>Response time:</i> 15.777,8 ms (Buruk / Tidak Stabil) <i>Throughput:</i> 4,10 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 10.490,5 ms (Buruk/Tidak Stabil) <i>Throughput:</i> 5,80 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)

Berdasarkan Tabel 24, hasil server lokal dan VPS UNUGHA dapat dibandingkan secara langsung pada setiap skenario pengguna. Data tersebut selanjutnya dianalisis pada bagian pembahasan.

3. Hasil Pengujian *Endpoint* /api/dashboard/students

Pengujian *endpoint* /api/dashboard/students dilakukan untuk memperoleh data performa dalam melayani permintaan informasi mahasiswa dan data evaluasi capaian pembelajaran pada *Dashboard* Admin.

Hasil pengujian *endpoint* /api/dashboard/students pada server lokal dan VPS UNUGHA disajikan secara berdampingan pada Tabel 25.

Tabel 25 Perbandingan Hasil Pengujian Endpoint /api/dashboard/students pada Server Lokal dan VPS UNUGHA

No	Pengguna virtual	Server lokal	VPS UNUGHA
1	1	<i>Response time:</i> 11,1 ms (Baik) <i>Throughput:</i> 0,15 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 93,4 ms (Baik) <i>Throughput:</i> 0,14 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
2	10	<i>Response time:</i> 37,2 ms (Baik) <i>Throughput:</i> 1,20 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 125,3 ms (Baik) <i>Throughput:</i> 1,19 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
3	20	<i>Response time:</i> 52,0 ms (Baik) <i>Throughput:</i> 2,39 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 225,0 ms (Baik) <i>Throughput:</i> 2,25 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
4	50	<i>Response time:</i> 539,9 ms (Baik) <i>Throughput:</i> 4,00 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 284,2 ms (Baik) <i>Throughput:</i> 4,86 req/s (Baik) <i>Error rate:</i> 0,0% (Baik)
5	100	<i>Response time:</i> 2.514,7 ms (Baik) <i>Throughput:</i> 4,15 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 1.290,5 ms (Baik) <i>Throughput:</i> 5,79 req/s (Cukup) <i>Error rate:</i> 0,0% (Baik)
6	200	<i>Response time:</i> 6.170,7 ms (Kurang) <i>Throughput:</i> 3,86 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 3.768,1 ms (Cukup) <i>Throughput:</i> 5,74 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)
7	500	<i>Response time:</i> 15.894,4 ms (Buruk / Tidak Stabil) <i>Throughput:</i> 3,69 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)	<i>Response time:</i> 10.714,6 ms (Buruk/Tidak Stabil) <i>Throughput:</i> 5,49 req/s (Kurang) <i>Error rate:</i> 0,0% (Baik)

Berdasarkan Tabel 25, hasil server lokal dan VPS UNUGHA dapat dibandingkan secara langsung pada setiap skenario pengguna. Data tersebut selanjutnya dianalisis pada bagian pembahasan.

4. Hasil Pemantauan Penggunaan CPU dan Memori *Server*

Pemantauan penggunaan CPU dan memori dilakukan menggunakan Glances pada saat beban puncak setiap skenario pengujian. Data pemantauan digunakan sebagai informasi pendukung dalam analisis performa sistem.

Hasil pemantauan penggunaan CPU dan memori pada server lokal dan VPS UNUGHA disajikan secara berdampingan pada Tabel 26.

Tabel 26 Perbandingan Hasil Pemantauan CPU dan Memori pada Server Lokal dan VPS UNUGHA

No	Skenario pengguna	Server lokal	VPS UNUGHA
1	Kondisi awal (idle)	CPU global: 8,7% CPU Node.js: 4,6% Memori: 1.80G / 64,8%	CPU global: 0,7% CPU Node.js: 6,3% Memori: 1,68 GB / 14,4%
2	1	CPU global: 11,2% CPU Node.js: 4,6% Memori: 1.80G (~1843 MB) / 64,8%	CPU global: 0,9% CPU Node.js: 8,6% Memori: 1,60 GB / 13,7%
3	10	CPU global: 32,5% CPU Node.js: 23% Memori: 1.78G (~1822 MB) / 64,1%	CPU global: 2,0% CPU Node.js: 40,9% Memori: 1,49 GB / 12,8%
4	20	CPU global: 47,2% CPU Node.js: 67,7% Memori: 1.77G (~1812 MB) / 63,7%	CPU global: 3,0% CPU Node.js: 50,2% Memori: 1,48 GB / 12,7%
5	50	CPU global: 59,5% CPU Node.js: 95% Memori: 1.79G (~1832 MB) / 64,4%	CPU global: 5,1% CPU Node.js: 91,5% Memori: 1,50 GB / 12,8%
6	100	CPU global: 61,2% CPU Node.js: 101% Memori: 1.82G (~1863 MB) / 65,6%	CPU global: 5,7% CPU Node.js: 110,0% Memori: 1,53 GB / 13,1%
7	200	CPU global: 65,3% CPU Node.js: 98,5% Memori: 1.82G (~1863 MB) / 65,5%	CPU global: 5,6% CPU Node.js: 112,0% Memori: 1,52 GB / 13,0%
8	500	CPU global: 61,3% CPU Node.js: 101% Memori: 1.84G (~1884 MB) / 66,4%	CPU global: 5,4% CPU Node.js: 111,0% Memori: 1,55 GB / 13,3%

Berdasarkan Tabel 26, penggunaan CPU global tertinggi pada server lokal mencapai 65,3%, sedangkan pada VPS UNUGHA mencapai 5,7%. Penggunaan CPU proses Node.js tertinggi masing-masing mencapai sekitar 101% dan 112%.

Penggunaan memori relatif stabil pada kedua lingkungan, yaitu 1,77–1,84 GB atau 63,7–66,4% pada server lokal dan 1,48–1,68 GB atau 12,7–14,4% pada VPS UNUGHA. Dokumentasi pemantauan Glances pada setiap skenario disajikan pada bagian lampiran.

B. Pembahasan

Pada bagian ini dilakukan pembahasan terhadap hasil pengujian performa *Dashboard Admin Sistem CPL* pada *server* lokal dan VPS UNUGHA. Pembahasan mencakup parameter *response time*, *throughput*, *error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem.

Hasil pengujian pada masing-masing *server* dianalisis terlebih dahulu, kemudian dibandingkan pada skenario jumlah pengguna virtual yang sama. Selanjutnya, hasil tersebut dievaluasi berdasarkan kriteria operasional yang telah ditetapkan pada Bab III. Hasil analisis dan perbandingan digunakan untuk mengidentifikasi dugaan *bottleneck* serta menyusun rekomendasi perbaikan yang sesuai dengan kondisi setiap lingkungan *server*.

1. Analisis Parameter Performa

Analisis parameter performa dilakukan terhadap hasil pengujian pada *server* lokal dan VPS UNUGHA. Parameter yang dianalisis meliputi *response time*, *throughput*, *error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem.

Analisis dilakukan untuk mengetahui pola perubahan performa setiap *endpoint* akibat peningkatan jumlah pengguna virtual. Hasil pada kedua *server* kemudian dibandingkan untuk mengetahui perbedaan kemampuan sistem dalam menangani beban pengujian yang sama. Penilaian kondisi performa mengacu pada kriteria operasional evaluasi performa yang telah ditetapkan pada Bab III.

a. Analisis *Response time*

Response time merupakan waktu yang dibutuhkan sistem sejak menerima permintaan hingga mengirimkan respons kepada pengguna. Nilai *response time* yang lebih rendah menunjukkan bahwa sistem dapat memberikan respons dengan lebih cepat.

Nilai *response time* pada VPS UNUGHA merupakan waktu respons *end-to-end* yang diukur sejak Apache JMeter mengirimkan permintaan dari laptop penguji melalui jaringan internet menuju domain obe.unugha.ac.id hingga respons diterima kembali. Dengan demikian, nilai tersebut mencakup waktu pemrosesan aplikasi dan latensi jaringan. Sementara itu, pengujian server lokal dilakukan melalui jaringan lokal dengan jalur komunikasi yang lebih pendek. Perbedaan jalur jaringan perlu diperhatikan terutama ketika menginterpretasikan hasil pada skenario beban rendah

Berdasarkan hasil pengujian, ketiga *endpoint* menunjukkan peningkatan *response time* seiring bertambahnya jumlah pengguna virtual. Namun, titik awal peningkatan yang signifikan berbeda antara *server* lokal dan VPS UNUGHA. Pada *server* lokal, peningkatan mulai terlihat pada skenario 50 pengguna, sedangkan pada VPS UNUGHA peningkatan yang lebih besar mulai terlihat pada skenario 100 pengguna. Ringkasan hasil pengujian *response time* disajikan pada Tabel 27.

Tabel 27 Ringkasan Hasil Pengujian Response time

<i>Server</i>	<i>Endpoint</i>	<i>Response time terendah (ms)</i>	<i>Response time tertinggi (ms)</i>	Pola perubahan
Lokal	/api/dashboard /stats	13,0	15.548,8	Relatif rendah hingga 20 pengguna, kemudian meningkat tajam mulai 50 pengguna
Lokal	/api/dashboard /dosen	4,7	15.777,8	Relatif rendah hingga 20 pengguna, kemudian meningkat tajam mulai 50 pengguna
Lokal	/api/dashboard /students	11,1	15.894,4	Relatif rendah hingga 20 pengguna, kemudian meningkat tajam mulai 50 pengguna
VPS UNUGHA	/api/dashboard /stats	100,3	10.532,2	Relatif rendah hingga 50 pengguna, kemudian meningkat tajam mulai 100 pengguna

<i>Server</i>	<i>Endpoint</i>	<i>Response time terendah (ms)</i>	<i>Response time tertinggi (ms)</i>	Pola perubahan
VPS UNUGHA	/api/dashboard /dosen	52,3	10.490,5	Relatif rendah hingga 50 pengguna, kemudian meningkat tajam mulai 100 pengguna
VPS UNUGHA	/api/dashboard /students	93,4	10.714,6	Relatif rendah hingga 50 pengguna, kemudian meningkat tajam mulai 100 pengguna

Berdasarkan Tabel 27, *server* lokal menghasilkan *response time* yang lebih rendah dibandingkan VPS UNUGHA pada skenario 1, 10, dan 20 pengguna. Pada skenario tersebut, seluruh *endpoint* pada *server* lokal masih memberikan respons di bawah 70 ms, sedangkan VPS UNUGHA berada pada kisaran 52,3 ms hingga 225,0 ms.

Perbedaan kondisi mulai terlihat pada skenario 50 pengguna. *Response time server* lokal meningkat menjadi 521,4 ms hingga 613,2 ms, sedangkan VPS UNUGHA masih berada pada kisaran 252,4 ms hingga 324,9 ms. Hasil tersebut menunjukkan bahwa peningkatan waktu respons pada *server* lokal terjadi lebih awal daripada VPS UNUGHA.

Pada skenario 100 pengguna, seluruh *endpoint* pada *server* lokal menghasilkan *response time* sekitar 2,48 hingga 2,51 detik. Sementara itu, VPS UNUGHA menghasilkan *response time* sekitar 1,23 hingga 1,35 detik. Berdasarkan kriteria operasional pada Tabel 20, hasil kedua *server* pada skenario ini masih termasuk kategori baik karena nilai *response time* masih berada di bawah 3 detik.

Pada skenario 200 pengguna, *response time* seluruh *endpoint* pada *server* lokal meningkat menjadi 5,99 hingga 6,17 detik sehingga dikategorikan kurang. Pada VPS UNUGHA, *response time* berada pada kisaran 3,69 hingga 3,77 detik sehingga masih termasuk kategori cukup. Hasil tersebut menunjukkan bahwa VPS UNUGHA mampu mempertahankan waktu respons yang lebih baik pada beban 200 pengguna.

Pada skenario 500 pengguna, seluruh *endpoint* pada kedua *server* menghasilkan *response time* lebih dari 10 detik sehingga dikategorikan buruk atau tidak stabil. Meskipun demikian, nilai *response time* VPS UNUGHA masih lebih rendah dibandingkan *server* lokal. Pada *server* lokal, nilai tertinggi terdapat pada *endpoint* */api/dashboard/students* sebesar 15.894,4 ms. Pada VPS UNUGHA, *endpoint* yang sama juga menghasilkan nilai tertinggi sebesar 10.714,6 ms.

Secara keseluruhan, *server* lokal memberikan waktu respons yang lebih rendah pada beban awal, tetapi mengalami peningkatan lebih cepat ketika jumlah pengguna bertambah. VPS UNUGHA memiliki waktu respons awal yang lebih tinggi, tetapi menunjukkan kemampuan yang lebih baik dalam menangani beban 50 hingga 200 pengguna. Namun, pada skenario 500 pengguna, kedua *server* telah mengalami penurunan performa hingga mencapai kondisi buruk atau tidak stabil.

b. Analisis *Throughput*

Throughput merupakan jumlah permintaan yang dapat diproses sistem dalam satuan waktu. Nilai *throughput* yang meningkat secara proporsional menunjukkan bahwa sistem mampu memanfaatkan penambahan beban untuk memproses lebih banyak permintaan. Sebaliknya, nilai yang meningkat secara lambat, stagnan, atau menurun menunjukkan bahwa kapasitas pemrosesan sistem mulai mencapai kondisi jenuh.

Berdasarkan hasil pengujian, *throughput* pada kedua *server* meningkat pada skenario beban rendah. Namun, *server* lokal mulai mengalami perlambatan peningkatan pada skenario 50 pengguna, sedangkan VPS UNUGHA masih menunjukkan peningkatan yang lebih baik hingga skenario 100 pengguna. Ringkasan hasil analisis *throughput* disajikan pada Tabel 28.

Tabel 28 Ringkasan Hasil Pengujian Throughput

<i>Server</i>	<i>Endpoint</i>	<i>Throughput</i> tertinggi (req/s)	Skenario	Pola perubahan
Lokal	<i>/api/dashboard/stats</i>	4,41	100 dan 200 pengguna	Meningkat hingga 100 pengguna, stagnan pada 200 pengguna, kemudian sedikit menurun pada 500 pengguna
Lokal	<i>/api/dashboard/dosen</i>	4,24	100 pengguna	Meningkat hingga 100 pengguna,

<i>Server</i>	<i>Endpoint</i>	<i>Throughput tertinggi (req/s)</i>	<i>Skenario</i>	<i>Pola perubahan</i>
				kemudian menurun
Lokal	/api/dashboard/students	4,15	100 pengguna	Meningkat hingga 100 pengguna, kemudian menurun
VPS UNUGHA	/api/dashboard/stats	6,18	200 pengguna	Meningkat hingga 200 pengguna, kemudian sedikit menurun
VPS UNUGHA	/api/dashboard/dosen	6,01	200 pengguna	Meningkat hingga 200 pengguna, kemudian menurun
VPS UNUGHA	/api/dashboard/students	5,79	100 pengguna	Meningkat hingga 100 pengguna, kemudian menurun

Pada server lokal, nilai throughput seluruh endpoint meningkat secara proporsional hingga skenario 20 pengguna sehingga dikategorikan baik. Pada skenario 50 dan 100 pengguna, throughput masih mengalami peningkatan, tetapi laju pertumbuhannya mulai melambat sehingga dikategorikan cukup.

Pada endpoint /api/dashboard/stats, nilai throughput meningkat menjadi 4,41 req/s pada skenario 100 pengguna dan tetap sebesar 4,41 req/s pada 200 pengguna. Pada skenario 500 pengguna, nilainya sedikit menurun menjadi 4,40 req/s. Pola tersebut menunjukkan bahwa kapasitas pemrosesan endpoint mulai mengalami stagnasi sejak skenario 100 pengguna dan dikategorikan kurang pada skenario 200 serta 500 pengguna.

Pada endpoint /api/dashboard/dosen, nilai throughput tertinggi sebesar 4,24 req/s dicapai pada skenario 100 pengguna, kemudian menurun menjadi 4,17 req/s pada 200 pengguna dan 4,10 req/s pada 500 pengguna. Pola serupa terjadi pada endpoint /api/dashboard/students, yaitu menurun dari 4,15 req/s pada 100 pengguna menjadi 3,86 req/s pada 200 pengguna dan 3,69 req/s pada 500 pengguna. Hasil tersebut menunjukkan bahwa server lokal mulai mengalami kejenuhan setelah skenario 100 pengguna.

Pada VPS UNUGHA, nilai throughput seluruh endpoint meningkat secara proporsional hingga skenario 50 pengguna sehingga dikategorikan baik. Pada skenario 100 pengguna, nilai throughput masih meningkat, tetapi laju pertumbuhannya mulai melambat sehingga dikategorikan cukup.

Endpoint `/api/dashboard/stats` menghasilkan throughput sebesar 5,96 req/s pada 100 pengguna dan meningkat menjadi 6,18 req/s pada 200 pengguna. Meskipun masih meningkat, pertumbuhannya mulai melambat dan mendekati kondisi jenuh sehingga dikategorikan cukup. Pada skenario 500 pengguna, nilainya sedikit menurun menjadi 6,15 req/s sehingga dikategorikan kurang.

Endpoint `/api/dashboard/dosen` menunjukkan pola serupa. Nilai throughput meningkat dari 5,87 req/s pada 100 pengguna menjadi 6,01 req/s pada 200 pengguna, kemudian menurun menjadi 5,80 req/s pada 500 pengguna. Sementara itu, endpoint `/api/dashboard/students` mengalami penurunan lebih awal, yaitu dari 5,79 req/s pada 100 pengguna menjadi 5,74 req/s pada 200 pengguna dan 5,49 req/s pada 500 pengguna. Oleh karena itu, endpoint `/api/dashboard/students` pada VPS UNUGHA dikategorikan kurang mulai skenario 200 pengguna.

Pada skenario 1 hingga 20 pengguna, perbedaan nilai throughput antara server lokal dan VPS UNUGHA relatif kecil. Perbedaan mulai terlihat pada skenario 50 pengguna, ketika seluruh endpoint pada VPS UNUGHA menghasilkan throughput sebesar 4,84–4,86 req/s, sedangkan server lokal berada pada kisaran 4,00–4,16 req/s. Pada skenario 100 pengguna, VPS UNUGHA menghasilkan throughput sebesar 5,79–5,96 req/s, lebih tinggi dibandingkan server lokal yang berada pada kisaran 4,15–4,41 req/s.

Pada skenario 200 pengguna, VPS UNUGHA menghasilkan throughput sebesar 5,74–6,18 req/s, sedangkan server lokal berada pada kisaran 3,86–4,41 req/s. Pada skenario 500 pengguna, VPS UNUGHA masih menghasilkan throughput sebesar 5,49–6,15 req/s, lebih tinggi dibandingkan server lokal yang berada pada kisaran 3,69–4,40 req/s.

Secara keseluruhan, VPS UNUGHA memiliki kapasitas pemrosesan permintaan yang lebih tinggi dibandingkan server lokal, terutama pada skenario 50 hingga 500 pengguna. Meskipun demikian, peningkatan throughput pada VPS UNUGHA mulai melambat pada skenario 100 hingga 200 pengguna dan menurun pada skenario 500 pengguna. Endpoint `/api/dashboard/students` menunjukkan penurunan throughput paling awal pada kedua lingkungan sehingga mengindikasikan beban pemrosesan yang lebih tinggi dibandingkan endpoint lainnya.

c. Analisis *Error rate*

Error rate merupakan persentase permintaan yang gagal diproses oleh sistem selama pengujian. Semakin kecil nilai *error rate*, semakin baik kemampuan sistem dalam menyelesaikan permintaan pengguna. Berdasarkan kriteria operasional pada Bab III, nilai *error rate* sebesar 0% termasuk kategori baik.

Ringkasan hasil pengujian *error rate* pada *server* lokal dan VPS UNUGHA disajikan pada Tabel 29.

Tabel 29 Ringkasan Hasil Pengujian Error rate pada Server Lokal dan VPS UNUGHA

<i>Server</i>	<i>Endpoint</i>	<i>Error rate</i> terendah (%)	<i>Error rate</i> tertinggi (%)	Kategori
Lokal	/api/dashboard/stats	0,0	0,0	Baik
Lokal	/api/dashboard/dosen	0,0	0,0	Baik
Lokal	/api/dashboard/students	0,0	0,0	Baik
VPS UNUGHA	/api/dashboard/stats	0,0	0,0	Baik
VPS UNUGHA	/api/dashboard/dosen	0,0	0,0	Baik
VPS UNUGHA	/api/dashboard/students	0,0	0,0	Baik

Berdasarkan Tabel 29, seluruh *endpoint* pada *server* lokal dan VPS UNUGHA menghasilkan *error rate* sebesar 0% pada skenario 1 hingga 500 pengguna virtual. Hasil tersebut menunjukkan bahwa seluruh permintaan yang dikirimkan selama pengujian berhasil diterima dan diproses tanpa menghasilkan kegagalan permintaan.

Meskipun tidak ditemukan kesalahan permintaan, nilai *error rate* sebesar 0% tidak menunjukkan bahwa performa sistem secara keseluruhan tetap baik. Pada skenario beban tinggi, terutama 200 dan 500 pengguna, sistem masih

berhasil memberikan respons, tetapi membutuhkan waktu yang semakin lama. Kondisi tersebut terlihat dari peningkatan *response time* dan perlambatan atau penurunan *throughput* pada kedua *server*.

Dengan demikian, penurunan performa yang terjadi pada penelitian ini tidak ditandai oleh kegagalan permintaan, melainkan oleh peningkatan waktu respons dan kejenuhan kapasitas pemrosesan. VPS UNUGHA maupun *server* lokal masih dapat menyelesaikan seluruh permintaan hingga 500 pengguna, tetapi pada skenario tersebut waktu respons telah berada dalam kategori buruk atau tidak stabil.

d. Analisis Penggunaan CPU dan Memori

Analisis penggunaan CPU dan memori dilakukan untuk mengetahui perubahan pemanfaatan sumber daya *server* selama peningkatan jumlah pengguna virtual. Data ini digunakan sebagai pendukung untuk menginterpretasikan perubahan *response time* dan *throughput* pada *server* lokal dan VPS UNUGHA.

Ringkasan penggunaan CPU dan memori pada kedua *server* disajikan pada Tabel 30.

Tabel 30 Ringkasan Penggunaan CPU dan Memori Server Lokal dan VPS UNUGHA

<i>Server</i>	CPU global awal	CPU global tertinggi	CPU Node.js tertinggi	Kisaran penggunaan memori
Lokal	8,7%	65,3%	101%	1,77–1,84 GB atau 63,7–66,4%
VPS UNUGHA	0,7%	5,7%	112%	1,48–1,68 GB atau 12,7–14,4%

Berdasarkan Tabel 30, penggunaan CPU global pada *server* lokal meningkat seiring bertambahnya jumlah pengguna virtual. Penggunaan CPU meningkat dari 8,7% pada kondisi awal menjadi 59,5% pada skenario 50 pengguna dan mencapai nilai tertinggi sebesar 65,3% pada skenario 200 pengguna. Pada skenario 500 pengguna, penggunaan CPU global tercatat sebesar 61,3%.

Peningkatan penggunaan CPU proses Node.js pada *server* lokal terlihat lebih besar dibandingkan penggunaan CPU global. Nilainya meningkat dari 4,6% pada kondisi awal menjadi 95% pada 50 pengguna dan mencapai sekitar 101% pada skenario 100 dan 500 pengguna. Nilai tersebut menunjukkan bahwa proses Node.js telah menggunakan kapasitas sekitar satu inti prosesor secara

penuhi. Kondisi ini terjadi bersamaan dengan peningkatan *response time* dan perlambatan *throughput*, terutama mulai skenario 50 hingga 100 pengguna.

Pada VPS UNUGHA, penggunaan CPU global relatif rendah selama seluruh pengujian. Nilainya meningkat dari 0,7% pada kondisi awal menjadi 5,1% pada 50 pengguna dan mencapai nilai tertinggi sebesar 5,7% pada 100 pengguna. Penggunaan CPU global kemudian berada pada 5,6% untuk 200 pengguna dan 5,4% untuk 500 pengguna.

Meskipun penggunaan CPU global VPS UNUGHA relatif rendah, penggunaan CPU proses Node.js meningkat dari 6,3% pada kondisi awal menjadi 91,5% pada 50 pengguna, 110% pada 100 pengguna, dan mencapai 112% pada 200 pengguna. Nilai penggunaan CPU proses yang melebihi 100% menunjukkan bahwa proses Node.js menggunakan kapasitas lebih dari satu inti logis. Namun, kondisi tersebut tidak berarti seluruh kapasitas CPU VPS UNUGHA telah digunakan karena VPS UNUGHA memiliki 24 prosesor logis dan nilai CPU global tetap berada di bawah 6%.

Pola tersebut menunjukkan bahwa penurunan performa pada VPS UNUGHA tidak disebabkan oleh habisnya kapasitas CPU *server* secara keseluruhan. Peningkatan penggunaan CPU proses Node.js yang terjadi bersamaan dengan kenaikan *response time* dan stagnasi *throughput* lebih mengarah pada keterbatasan pemrosesan pada tingkat aplikasi atau proses Node.js.

Penggunaan memori pada kedua *server* relatif stabil selama pengujian. Pada *server* lokal, penggunaan memori berada pada kisaran 1,77–1,84 GB atau 63,7–66,4%. Pada VPS UNUGHA, penggunaan memori berada pada kisaran 1,48–1,68 GB atau 12,7–14,4%.

Tidak terdapat peningkatan penggunaan memori yang signifikan ketika jumlah pengguna bertambah. Oleh karena itu, memori tidak menunjukkan indikasi sebagai penyebab utama penurunan performa pada kedua *server*. Penurunan performa lebih berkaitan dengan peningkatan waktu pemrosesan dan keterbatasan kapasitas proses Node.js dalam menangani permintaan secara bersamaan

e. Analisis Kestabilan Sistem

Analisis kestabilan dilakukan untuk mengetahui kemampuan sistem dalam mempertahankan layanan ketika jumlah pengguna virtual meningkat. Penilaian dilakukan berdasarkan perubahan *response time*, pola *throughput*, *error rate*, serta penggunaan sumber daya *server*. Ringkasan kestabilan sistem disajikan pada Tabel 31.

Tabel 31 Ringkasan Kestabilan Server Lokal dan VPS UNUGHA

Skenario pengguna	Server lokal	VPS UNUGHA	Indikator utama
1–20	Stabil	Stabil	<i>Response time</i> rendah, <i>throughput</i> meningkat, dan <i>error rate</i> 0%
50	Mulai menurun	Stabil	Server lokal mengalami peningkatan <i>response time</i> , sedangkan VPS UNUGHA masih memberikan respons di bawah 500 ms
100	Mendekati kondisi jenuh	Mulai menurun	Pertumbuhan <i>throughput</i> melambat dan penggunaan CPU proses Node.js meningkat
200	Kurang stabil	Mendekati kondisi jenuh	<i>Response time</i> server lokal melebihi 5 detik, sedangkan VPS UNUGHA berada pada kisaran 3,69–3,77 detik
500	Buruk/Tidak stabil	Buruk/Tidak stabil	<i>Response time</i> seluruh <i>endpoint</i> melebihi 10 detik dan <i>throughput</i> stagnan atau menurun

Berdasarkan Tabel 31, *server* lokal mampu mempertahankan kestabilan performa hingga skenario 20 pengguna dan mulai mengalami penurunan pada 50 pengguna. Pada skenario 100 pengguna, pertumbuhan *throughput* mulai melambat dan penggunaan CPU proses Node.js mencapai sekitar 101%, sehingga sistem mulai mendekati kondisi jenuh.

Pada VPS UNUGHA, performa masih stabil hingga skenario 50 pengguna. Penurunan mulai terlihat pada 100 pengguna dan kondisi mendekati jenuh terjadi pada 200 pengguna. Meskipun demikian, VPS UNUGHA masih menghasilkan *response time* dan *throughput* yang lebih baik dibandingkan *server* lokal pada beban menengah hingga tinggi.

Pada skenario 500 pengguna, kedua *server* dikategorikan buruk atau tidak stabil karena seluruh *endpoint* menghasilkan *response time* lebih dari 10 detik dan *throughput* tidak lagi mengalami peningkatan. Nilai *error rate* tetap sebesar 0%, tetapi seluruh permintaan membutuhkan waktu yang semakin lama untuk diproses. Dengan demikian, VPS UNUGHA lebih stabil daripada *server* lokal, tetapi keduanya belum mampu mempertahankan performa yang baik pada beban 500 pengguna.

2. Identifikasi Dugaan *Bottleneck*

Identifikasi dugaan *bottleneck* dilakukan dengan menghubungkan perubahan *response time*, *throughput*, penggunaan CPU, penggunaan memori, dan karakteristik pemrosesan setiap *endpoint*. Identifikasi dilakukan secara terpisah pada *server* lokal dan VPS UNUGHA karena keduanya memiliki spesifikasi perangkat keras yang berbeda.

Ringkasan hasil identifikasi dugaan *bottleneck* disajikan pada Tabel 32.

Tabel 32 Identifikasi Dugaan *Bottleneck* pada Server Lokal dan VPS UNUGHA

<i>Server</i>	Temuan pengujian	Indikator	Dugaan <i>bottleneck</i>
Lokal	Penurunan performa mulai terlihat pada 50 pengguna	<i>Response time</i> meningkat di atas 500 ms dan CPU Node.js mencapai 95%	Kapasitas pemrosesan proses Node.js mulai mendekati batas satu inti prosesor
Lokal	<i>Throughput</i> stagnan atau menurun setelah 100 pengguna	<i>Throughput</i> hanya berada pada kisaran 3,69–4,41 req/s	Kapasitas pemrosesan aplikasi tidak bertambah meskipun jumlah permintaan meningkat
Lokal	<i>Endpoint</i> /api/dashboard/students memiliki beban tertinggi	<i>Response time</i> mencapai 15.894,4 ms dan <i>throughput</i> turun menjadi 3,69 req/s	Proses pengambilan, penggabungan, atau pengolahan data pada <i>endpoint</i> lebih kompleks
Lokal	Penggunaan memori relatif stabil	Memori berada pada kisaran 63,7–66,4%	Memori tidak menunjukkan indikasi sebagai penyebab utama penurunan performa
VPS UNUGHA	Penurunan performa mulai terlihat pada 100 pengguna	<i>Response time</i> meningkat di atas 1 detik dan CPU Node.js mencapai 110%	Keterbatasan pemrosesan pada tingkat proses Node.js atau aplikasi
VPS UNUGHA	CPU global tetap rendah ketika CPU Node.js meningkat	CPU global di bawah 6%, sedangkan CPU Node.js mencapai 112%	Kapasitas CPU <i>server</i> secara keseluruhan belum habis, tetapi proses aplikasi belum memanfaatkan seluruh prosesor secara optimal

<i>Server</i>	<i>Temuan pengujian</i>	<i>Indikator</i>	<i>Dugaan bottleneck</i>
VPS UNUGHA	<i>Throughput</i> mulai jenuh pada 100–200 pengguna	Nilai tertinggi hanya sekitar 5,79–6,18 req/s dan menurun pada 500 pengguna	Kapasitas pemrosesan aplikasi atau akses data mulai mencapai titik jenuh
VPS UNUGHA	<i>Endpoint</i> /api/dashboard/students mengalami penurunan lebih awal	<i>Throughput</i> menurun mulai 200 pengguna dan <i>response time</i> mencapai 10.714,6 ms	Kompleksitas proses atau kueri data pada <i>endpoint</i> lebih tinggi dibandingkan <i>endpoint</i> lainnya
VPS UNUGHA	Penggunaan memori relatif stabil	Memori berada pada kisaran 12,7–14,4%	Memori tidak menunjukkan indikasi sebagai penyebab utama penurunan performa

Berdasarkan Tabel 32, dugaan *bottleneck* pada *server* lokal mulai terlihat pada skenario 50 pengguna. Kondisi tersebut ditandai dengan peningkatan *response time*, perlambatan pertumbuhan *throughput*, dan penggunaan CPU proses Node.js yang mencapai 95%. Pada skenario 100 pengguna, penggunaan CPU proses Node.js mencapai sekitar 101%, sedangkan *throughput* mulai stagnan atau menurun.

Pada VPS UNUGHA, penurunan performa terjadi lebih lambat, yaitu mulai skenario 100 pengguna. Penggunaan CPU proses Node.js mencapai 110–112%, tetapi penggunaan CPU global tetap berada di bawah 6%. Kondisi tersebut menunjukkan bahwa sumber daya CPU VPS UNUGHA secara keseluruhan masih tersedia, tetapi pemrosesan pada tingkat aplikasi atau proses Node.js mulai mencapai keterbatasan.

Endpoint /api/dashboard/students menunjukkan indikasi beban pemrosesan paling tinggi pada kedua *server*. *Endpoint* ini memiliki *response time* tertinggi pada skenario 500 pengguna dan mengalami penurunan *throughput* lebih awal dibandingkan *endpoint* lainnya. Kondisi tersebut diduga berkaitan dengan jumlah data, proses penggabungan data, agregasi, atau kueri basis data yang lebih kompleks.

Penggunaan memori pada kedua *server* relatif stabil dan tidak mengalami peningkatan yang signifikan. Oleh karena itu, memori tidak menunjukkan indikasi sebagai penyebab utama penurunan performa. Dugaan *bottleneck* lebih mengarah pada keterbatasan proses Node.js, kapasitas pemrosesan aplikasi, dan kompleksitas akses data pada *endpoint* tertentu.

Identifikasi ini merupakan dugaan teknis berdasarkan hubungan antara hasil pengujian dan penggunaan sumber daya *server*. Penelitian ini tidak melakukan *code*

profiling maupun penelusuran kueri basis data secara langsung, sehingga penyebab *bottleneck* belum dapat dinyatakan secara pasti.

3. Rekomendasi Perbaikan

Rekomendasi perbaikan disusun berdasarkan hasil analisis parameter performa dan identifikasi dugaan *bottleneck* pada *server* lokal dan VPS UNUGHA. Parameter yang dianalisis meliputi *response time*, *throughput*, *error rate*, penggunaan CPU, penggunaan memori, dan kestabilan sistem.

Hasil analisis menunjukkan bahwa peningkatan jumlah pengguna virtual menyebabkan kenaikan *response time* serta perlambatan atau penurunan *throughput* pada kedua *server*. Seluruh permintaan tetap berhasil diproses dengan *error rate* sebesar 0%, sedangkan penggunaan memori relatif stabil. Penggunaan CPU proses Node.js meningkat hingga mendekati atau melebihi kapasitas satu inti prosesor. Selain itu, *endpoint* /api/dashboard/students menghasilkan *response time* tertinggi dan mengalami penurunan *throughput* lebih awal dibandingkan *endpoint* lainnya.

Berdasarkan hasil analisis parameter tersebut, dugaan *bottleneck* mengarah pada proses akses dan pengolahan data, pemrosesan aplikasi yang terkonsentrasi pada proses Node.js, serta keterbatasan mekanisme *in-app caching* apabila aplikasi dijalankan dalam beberapa proses. Oleh karena itu, rekomendasi perbaikan difokuskan pada optimalisasi lapisan akses data, penyesuaian pemanfaatan prosesor berdasarkan lingkungan *server*, dan migrasi mekanisme *caching* ke layanan eksternal.

Rekomendasi yang diberikan merupakan arahan teknis berdasarkan temuan empiris dan didukung oleh penelitian terdahulu. Rekomendasi tersebut tidak diimplementasikan maupun diuji ulang dalam penelitian ini. Ringkasan hubungan antara hasil analisis parameter performa, identifikasi dugaan *bottleneck*, dan rekomendasi perbaikan disajikan pada Tabel 33.

Tabel 33 Rekomendasi Perbaikan Berdasarkan Hasil Analisis Parameter Performa dan Identifikasi Dugaan Bottleneck

Lingkup penerapan	Hasil analisis parameter performa	Dugaan bottleneck	Rekomendasi perbaikan
Kedua server	<i>Endpoint</i> /api/dashboard/students menghasilkan <i>response time</i> tertinggi dan mengalami penurunan <i>throughput</i> lebih awal.	Proses pengambilan, penggabungan, atau pengolahan data pada <i>endpoint</i> tersebut diduga memiliki beban pemrosesan lebih tinggi.	Melakukan <i>query refactoring</i> melalui penggunaan <i>select projection</i> , pembatasan relasi, pengurangan <i>over-fetching</i> , penerapan <i>pagination</i> , dan penggunaan indeks basis data.
Server lokal	Penurunan performa mulai terlihat pada 50 pengguna. CPU global mencapai 61–65%, sedangkan CPU proses Node.js mencapai sekitar 101%.	Kapasitas pemrosesan server lokal diduga mulai terbatas ketika jumlah permintaan meningkat.	Menyesuaikan jumlah proses aplikasi dengan jumlah inti prosesor serta meningkatkan kapasitas prosesor apabila server digunakan sebagai lingkungan produksi.
VPS UNUGHA	CPU proses Node.js mencapai 110–112%, sedangkan CPU global tetap berada di bawah 6%.	Pemrosesan aplikasi diduga terkonsentrasi pada proses Node.js sehingga sumber daya prosesor <i>multi-core</i> belum dimanfaatkan secara optimal.	Menjalankan beberapa <i>worker process</i> menggunakan PM2 <i>cluster mode</i> agar permintaan dapat diproses secara paralel.

Lingkup penerapan	Hasil analisis parameter performa	Dugaan <i>bottleneck</i>	Rekomendasi perbaikan
Kedua <i>server</i>	<i>Throughput</i> mulai stagnan atau menurun pada beban tinggi meskipun sistem telah menggunakan <i>in-app caching</i> .	Mekanisme cache masih bergantung pada memori proses aplikasi dan tidak dapat digunakan bersama oleh beberapa <i>worker process</i> .	Memindahkan mekanisme cache ke <i>external in-memory datastore</i> , seperti Redis atau Garnet[68].

Berdasarkan Tabel 33, rekomendasi optimalisasi akses data dan migrasi mekanisme *caching* berlaku pada kedua *server* karena keduanya menggunakan aplikasi dan struktur pemrosesan data yang sama. Sementara itu, rekomendasi pemanfaatan prosesor disesuaikan dengan karakteristik masing-masing lingkungan. *Server* lokal memiliki keterbatasan kapasitas perangkat keras, sedangkan VPS UNUGHA masih memiliki sumber daya CPU global yang belum dimanfaatkan secara optimal.

Selanjutnya, setiap rekomendasi perbaikan dijelaskan berdasarkan urutan hasil analisis parameter performa, identifikasi dugaan *bottleneck*, usulan perbaikan teknis, dan dukungan penelitian terdahulu sebagai berikut.

a. Optimalisasi Lapisan Akses Data melalui *Query Refactoring*

Hasil analisis parameter performa menunjukkan bahwa *endpoint* `/api/dashboard/students` menghasilkan *response time* tertinggi pada kedua lingkungan pengujian. Pada skenario 500 pengguna, *response time endpoint* tersebut mencapai 15.894,4 ms pada *server* lokal dan 10.714,6 ms pada VPS UNUGHA. *Endpoint* ini juga mengalami penurunan *throughput* lebih awal dibandingkan *endpoint* lainnya. Temuan tersebut mengindikasikan adanya beban pengambilan, penggabungan, atau pengolahan data yang lebih tinggi pada jalur pemrosesan *endpoint* `/api/dashboard/students`.

Berdasarkan dugaan *bottleneck* tersebut, perbaikan dapat dilakukan melalui *query refactoring* pada lapisan akses data. Langkah yang dapat diterapkan meliputi penggunaan atribut yang lebih selektif melalui *select projection*, pembatasan relasi yang dimuat, pengurangan *over-fetching*, penerapan *pagination*, serta penggunaan indeks pada kolom yang digunakan dalam proses relasi, penyaringan, dan pengurutan. Perbaikan tersebut bertujuan mengurangi volume data yang diambil, diproses, dan dikirimkan dalam setiap permintaan.

Rekomendasi ini sejalan dengan penelitian Yusmita et al. [69] yang menunjukkan bahwa strategi akses basis data berpengaruh terhadap efisiensi

eksekusi aplikasi. Penelitian tersebut membandingkan penggunaan *Raw SQL* dan Prisma ORM serta menunjukkan bahwa pemilihan strategi akses data yang sesuai dengan karakteristik kueri dapat meningkatkan performa aplikasi. Oleh karena itu, *query refactoring* diprioritaskan pada *endpoint* `/api/dashboard/students`, tetapi juga dapat diterapkan pada *endpoint* lain yang memiliki pola akses data serupa.

b. Penyesuaian Pemanfaatan Prosesor Berdasarkan Lingkungan *Server*

Hasil analisis penggunaan sumber daya menunjukkan bahwa CPU proses Node.js mencapai sekitar 101% pada *server* lokal dan 110–112% pada VPS UNUGHA. Namun, penggunaan CPU global pada kedua *server* menunjukkan kondisi yang berbeda. Pada *server* lokal, CPU global berada pada kisaran 61–65%, sedangkan pada VPS UNUGHA tetap berada di bawah 6%. Perbedaan tersebut menunjukkan bahwa rekomendasi pemanfaatan prosesor perlu disesuaikan dengan karakteristik setiap lingkungan *server*.

Pada *server* lokal, penurunan performa mulai terlihat pada skenario 50 pengguna dan semakin besar pada beban berikutnya. *Server* lokal menggunakan prosesor Intel Core 2 Duo P8400 dengan jumlah inti dan kapasitas pemrosesan yang terbatas. Oleh karena itu, jumlah proses aplikasi perlu disesuaikan dengan jumlah inti prosesor yang tersedia agar tidak menimbulkan persaingan sumber daya antarporses. Apabila *server* lokal akan digunakan sebagai lingkungan produksi dengan beban pengguna yang lebih tinggi, peningkatan kapasitas prosesor menjadi rekomendasi yang lebih sesuai.

Pada VPS UNUGHA, penggunaan CPU proses Node.js mencapai 110–112%, tetapi penggunaan CPU global tetap berada di bawah 6%. Kondisi tersebut menunjukkan bahwa kapasitas prosesor *server* secara keseluruhan masih tersedia, sementara pemrosesan aplikasi cenderung terkonsentrasi pada proses Node.js. Oleh karena itu, aplikasi dapat dijalankan dalam beberapa *worker process* menggunakan PM2 *cluster mode* agar permintaan didistribusikan dan diproses secara paralel. Jumlah *worker process* perlu disesuaikan dengan jumlah inti prosesor serta kapasitas layanan basis data.

Rekomendasi tersebut didukung oleh penelitian Rozak et al [70] yang menunjukkan bahwa penerapan teknik *clustering* pada aplikasi Node.js dapat meningkatkan pemanfaatan sumber daya prosesor dan kemampuan aplikasi dalam menangani permintaan secara bersamaan pada lingkungan *multi-core*. Dengan demikian, rekomendasi pada *server* lokal difokuskan pada penyesuaian jumlah proses dan peningkatan kapasitas prosesor, sedangkan pada VPS UNUGHA difokuskan pada pemanfaatan beberapa *worker process* melalui PM2 *cluster mode*.

c. Migrasi Arsitektur *Caching* ke Layanan *In-Memory Datastore* Eksternal

Hasil analisis *throughput* menunjukkan bahwa kemampuan pemrosesan kedua *server* mulai stagnan atau menurun pada beban tinggi. Pada *server* lokal, *throughput* ketiga *endpoint* berada pada kisaran 3,69–4,40 permintaan per detik pada skenario 500 pengguna. Pada VPS UNUGHA, nilainya berada pada kisaran 5,49–6,15 permintaan per detik. Kondisi tersebut menunjukkan bahwa peningkatan jumlah pengguna tidak lagi diikuti peningkatan kapasitas pemrosesan secara proporsional.

Sistem telah menggunakan mekanisme *in-app caching*, tetapi data cache masih tersimpan dalam memori proses aplikasi. Mekanisme tersebut memiliki keterbatasan ketika aplikasi dijalankan dalam beberapa *worker process* karena setiap proses memiliki ruang memori yang terpisah. Oleh karena itu, cache dapat dipindahkan ke *external in-memory datastore* agar dapat digunakan bersama oleh seluruh proses aplikasi dan tidak bergantung pada satu instance Node.js.

Pendekatan tersebut didukung oleh penelitian Prilyansyah [71], yang menunjukkan bahwa penerapan Redis sebagai *external Caching layer* mampu meningkatkan performa *API* melalui penurunan *Response time* dan peningkatan *Throughput* dibandingkan mekanisme tanpa *Caching*. Selain Redis, Microsoft Garnet juga dapat dipertimbangkan sebagai alternatif karena kompatibel dengan protokol RESP (Redis Serialization Protocol). Berdasarkan hasil RESP Bench yang dipublikasikan oleh Microsoft Research [72] menunjukkan bahwa Garnet menghasilkan *throughput* dan latensi yang kompetitif pada beberapa skenario pengujian *in-memory datastore*. Dengan demikian, penggunaan cache eksternal dapat menjadi arah pengembangan untuk meningkatkan efisiensi akses data dan mendukung penerapan beberapa proses aplikasi pada kedua *server*.

Secara keseluruhan, optimalisasi akses data dan migrasi cache eksternal berlaku pada kedua *server* karena menggunakan aplikasi dan struktur data yang sama. Sementara itu, penerapan PM2 *cluster mode* lebih diprioritaskan pada VPS UNUGHA karena masih memiliki kapasitas prosesor yang belum dimanfaatkan secara optimal. Seluruh rekomendasi tersebut disusun berdasarkan hasil analisis parameter performa dan identifikasi dugaan *bottleneck*, serta belum diimplementasikan maupun diuji ulang dalam penelitian ini.

BAB V KESIMPULAN

A. Kesimpulan

Berdasarkan hasil pengujian dan analisis performa *Dashboard* Admin Sistem CPL berbasis OBE pada *server* lokal dan VPS UNUGHA, diperoleh kesimpulan sebagai berikut:

1. Peningkatan jumlah pengguna virtual menyebabkan kenaikan response time serta perlambatan, stagnasi, atau penurunan throughput pada endpoint `/api/dashboard/stats`, `/api/dashboard/dosen`, dan `/api/dashboard/students`. Seluruh permintaan tetap berhasil diproses dengan error rate sebesar 0%, sedangkan penggunaan memori pada kedua lingkungan relatif stabil. Dengan demikian, penurunan performa terutama ditunjukkan oleh peningkatan waktu respons, kejenuhan kapasitas pemrosesan, dan peningkatan penggunaan CPU proses Node.js.
2. Server lokal menghasilkan response time lebih rendah pada skenario 1 hingga 20 pengguna, tetapi mulai mengalami penurunan performa pada 50 pengguna. VPS UNUGHA masih mempertahankan performa yang lebih baik hingga 50 pengguna dan mulai menurun pada 100 pengguna. Pada 200 pengguna, server lokal dikategorikan kurang dengan response time 5.987,6–6.170,7 ms, sedangkan VPS UNUGHA masih dikategorikan cukup dengan nilai 3.692,5–3.768,1 ms. Pada 500 pengguna, kedua lingkungan dikategorikan buruk atau tidak stabil karena response time melebihi 10 detik dan throughput tidak lagi meningkat. Hasil VPS UNUGHA merupakan pengukuran end-to-end melalui internet menuju domain `obe.unugha.ac.id`, sehingga turut mencakup latensi jaringan dan tidak dimaksudkan sebagai perbandingan murni perangkat keras.
3. Dugaan bottleneck mengarah pada beban akses dan pengolahan data endpoint `/api/dashboard/students`, keterbatasan pemrosesan pada proses Node.js, serta mekanisme in-app caching yang masih bergantung pada memori proses aplikasi. Endpoint `/api/dashboard/students` menghasilkan response time tertinggi, yaitu 15.894,4 ms pada server lokal dan 10.714,6 ms pada VPS UNUGHA. Berdasarkan temuan tersebut, rekomendasi perbaikan meliputi query refactoring, penyesuaian jumlah proses dan kapasitas prosesor pada server lokal, penerapan PM2 cluster mode pada VPS UNUGHA, serta migrasi cache menuju external in-memory datastore. Rekomendasi tersebut belum diimplementasikan maupun diuji ulang dalam penelitian ini.

B. Saran

Berdasarkan hasil pengujian, analisis parameter performa, identifikasi dugaan *bottleneck*, dan keterbatasan penelitian, saran yang dapat diberikan adalah sebagai berikut:

1. Pengembang sistem disarankan menerapkan perbaikan pada lapisan akses data, pemanfaatan prosesor, dan mekanisme caching. Perbaikan akses data diprioritaskan pada endpoint `/api/dashboard/students` melalui query refactoring, penggunaan select projection, pengurangan over fetching, penerapan pagination, dan penggunaan indeks basis data yang sesuai. Pada server lokal, jumlah proses aplikasi perlu disesuaikan dengan jumlah inti prosesor dan kapasitas prosesor dapat ditingkatkan apabila digunakan sebagai lingkungan produksi. Sementara itu, pada VPS UNUGHA, aplikasi dapat dijalankan menggunakan beberapa worker process melalui PM2 cluster mode. Mekanisme in-app caching juga disarankan untuk dikembangkan menjadi external caching menggunakan layanan in-memory datastore, seperti Redis.
2. Setelah rekomendasi teknis diterapkan, perlu dilakukan pengujian ulang menggunakan endpoint, data, parameter, dan skenario beban yang sama untuk mengetahui pengaruh perbaikan terhadap performa sistem. Penelitian selanjutnya juga disarankan menggunakan data operasional serta menambahkan endurance testing dan spike testing agar kestabilan sistem dapat dievaluasi pada kondisi yang lebih mendekati penggunaan nyata.

DAFTAR PUSTAKA

- [1] W. Kayanja, M. Kyambade, Dan T. Kiggundu, “Exploring Digital Transformation In Higher Education Setting: The Shift To Fully Automated And Paperless Systems,” *Cogent Education*, Vol. 12, No. 1, Des 2025, Doi: 10.1080/2331186x.2025.2489800.
- [2] B. Kushari Dan L. Septiadi, “A Learning Outcome Assessment Information System To Facilitate Outcome-Based Education (Obe) Implementation,” *Jurnal Pendidikan Teknologi Dan Kejuruan*, Vol. 28, No. 2, Okt 2022, Doi: 10.21831/Jptk.V28i2.42339.
- [3] M. M. Syeed, A. Shihavuddin, M. F. Uddin, M. Hasan, Dan R. H. Khan, “Outcome Based Education (Obe): Defining The Process And Practice For Engineering Education,” *Ieee Access*, Vol. 10, Hlm. 119170–119192, 2022, Doi: 10.1109/Access.2022.3219477.
- [4] A. Naim, M. M. Alnfai, Dan N. S. Almalki, “Information Systems Based Model For The Assessment Of Program Learning Outcomes In Measuring The Quality In Higher Education,” *Humanit. Soc. Sci. Commun.*, Vol. 12, No. 1, Hlm. 1975, Nov 2025, Doi: 10.1057/S41599-025-06259-9.
- [5] A. Safiudin, . S., M. E. Sulisty, S. Pramono, Dan A. Ramelan, “The Development Of Web-Based Outcome Based Education Information System,” *Journal Of Electrical, Electronic, Information, And Communication Technology*, Vol. 2, No. 2, Hlm. 61, Okt 2020, Doi: 10.20961/Jeeict.2.2.45291.
- [6] O. M. Yigitbasioglu Dan O. Velcu, “A Review Of Dashboards In Performance Management: Implications For Design And Research,” *International Journal Of Accounting Information Systems*, Vol. 13, No. 1, Hlm. 41–59, Mar 2012, Doi: 10.1016/J.Accinf.2011.08.002.
- [7] Z. Huang Dan E. Wu, “Lightweight Materialization For Fast Dashboards Over Joins,” *Proceedings Of The Acm On Management Of Data*, Vol. 1, No. 4, Hlm. 1–27, Des 2023, Doi: 10.1145/3626735.
- [8] H. Alnuhait, W. Alzyadat, A. Althunibat, H. Kahtan, B. Zaqaibeh, Dan H. A. Al-Khawaja, “Web Application Performance Assessment: A Study Of Responsiveness, Throughput, And Scalability,” *International Journal Of Advanced And Applied Sciences*, Vol. 11, No. 9, Hlm. 214–226, Sep 2024, Doi: 10.21833/Ijaas.2024.09.023.
- [9] J. Van Riet, I. Malavolta, Dan T. A. Ghaleb, “Optimize Along The Way: An Industrial Case Study On Web Performance,” *Journal Of*

- Systems And Software*, Vol. 198, Hlm. 111593, Apr 2023, Doi: 10.1016/J.Jss.2022.111593.
- [10] M. Bolanowski, M. Ćmil, Dan A. Starzec, “New Model For Defining And Implementing Performance Tests,” *Future Internet*, Vol. 16, No. 10, Hlm. 366, Okt 2024, Doi: 10.3390/Fi16100366.
 - [11] M. Hendayun, A. Ginanjar, Dan Y. Ihsan, “Analysis Of Application Performance Testing Using Load Testing And Stress Testing Methods In Api Service,” *Jurnal Sisfotek Global*, Vol. 13, No. 1, Hlm. 28, Mar 2023, Doi: 10.38101/Sisfotek.V13i1.2656.
 - [12] S. S. Raweyai Dan I. R. Widiyari, “Performance Testing Of Academic Website Using Load Testing Method Supported By Apache Jmeter At Xyz University,” *Jurnal Teknik Informatika (Jutif)*, 2024, [Daring]. Tersedia Pada: [Http://Jutif.If.Unsoed.Ac.Id/Index.Php/Jurnal/Article/View/1796](http://Jutif.If.Unsoed.Ac.Id/Index.Php/Jurnal/Article/View/1796)
 - [13] S. Di Meglio, L. L. L. Starace, V. Pontillo, R. Opdebeeck, C. De Roover, Dan S. Di Martino, “Performance Testing In Open-Source Web Projects: Adoption, Maintenance, And A Change Taxonomy,” Dalam *2025 Ieee International Conference On Software Maintenance And Evolution (Icsme)*, Ieee, Sep 2025, Hlm. 199–210. Doi: 10.1109/Icsme64153.2025.00027.
 - [14] E. H. K. A. N. Othman, N. A. Mat Saat, N. H. Muhamad, Dan M. Abdul Majid, “The Development An Outcome-Based Education (Obe) System For Measuring Student Programme Attainment,” *Journal Of Computing Research And Innovation*, Vol. 9, No. 2, Hlm. 177–187, Sep 2024, Doi: 10.24191/Jcrinn.V9i2.472.
 - [15] N. Hakim, L. Hakim, N. D. Made Santi Diwyarthi, A. R. Purnamasari, Dan N. K. Ningsih, “Bibliometric Analysis Of Outcome-Based Education (Obe) In Higher Education: Trends, Themes, And Future Directions,” *The Eastasouth Journal Of Learning And Educations*, Vol. 3, No. 03, Hlm. 171–185, Nov 2025, Doi: 10.58812/Esle.V3i03.800.
 - [16] A. M. M. Lamis Dan J. S. Pisueno, “Sistem Penilaian Dan Evaluasi Capaian Program Berbasis Web Dengan Analitik Data,” *World Journal Of Advanced Research And Reviews*, Vol. 29, No. 3, Hlm. 481–492, Mar 2026, Doi: 10.30574/Wjarr.2026.29.3.0572.
 - [17] S. E. R. Siang, A. R. L. Reyes, Dan C. E. Dum Dumaya, “Reinforcing Learning Management System With Obe-Based Learning Analytics And Riasec Integration,” *Tem Journal*, Vol. 13, No. 4, Hlm. 3346–3358, Nov 2024, Doi: 10.18421/Tem134-69.

- [18] N. A. Asmaranto, T. F. Kusumasari, Dan E. N. Alam, “A Dashboard System For Enhancing Learning Outcome Monitoring And Curriculum Management Under The Outcome-Based Education Framework,” Dalam *2025 International Seminar On Intelligent Technology And Its Applications (Isitia)*, Ieee, Jul 2025, Hlm. 661–666. Doi: 10.1109/Isitia66279.2025.11137419.
- [19] N. Utami Handayani Dan N. Abyor Handayani, “Sistem Monitoring Dan Evaluasi Proses Belajar Mengajar Berbasis Outcome Based Education Di Fakultas Teknik Universitas Diponegoro,” *Jpii*, Vol. 2, No. 3, Hlm. 194–200, 2024, Doi: 10.14710/Jpii.2024.24263.
- [20] A. Annisa, M. Ardiana, P. N. Rahayu, T. Brian, Dan M. A. Jami'in, “Performance Evaluation Of Web Applications Using Jmeter Load Testing For Server Capacity And Response Efficiency,” *Journal Of Applied Informatics And Computing*, Vol. 10, No. 1, Hlm. 585–590, Feb 2026, Doi: 10.30871/Jaic.V10i1.11840.
- [21] M. Garjitno, H. Angriani, Dan Y. Saharaeni, “Analisis Dan Optimasi Kinerja Aplikasi Potio Menggunakan Load Testing Jmeter,” *Jtriste*, Vol. 12, No. 2, Hlm. 56–68, Okt 2025, Doi: 10.55645/Jtriste.V12i2.625.
- [22] L. Paulsen Dan E. Lindsay, “Learning Analytics Dashboards Are Increasingly Becoming About Learning And Not Just Analytics - A Systematic Review,” *Educ. Inf. Technol. (Dordr)*, Vol. 29, No. 11, Hlm. 14279–14308, Agu 2024, Doi: 10.1007/S10639-023-12401-4.
- [23] M. Hernández-Campos, A. Gonzalez-Torres, Dan F. J. García-Peñalvo, “Learning Outcomes Evaluation Through Learning Analytics Systems In Higher Education: A Systematic Literature Review,” *Sage Open*, Vol. 15, No. 3, Jan 2025, Doi: 10.1177/21582440251347374.
- [24] N. J. Rao, “Outcome-Based Education: An Outline,” *Higher Education For The Future*, Vol. 7, No. 1, Hlm. 5–21, Jan 2020, Doi: 10.1177/2347631119886418.
- [25] R. Mufanti, D. Carter, Dan N. England, “Outcomes-Based Education In Indonesian Higher Education: Reporting On The Understanding, Challenges, And Support Available To Teachers,” *Social Sciences & Humanities Open*, Vol. 9, Hlm. 100873, 2024, Doi: 10.1016/J.Ssaho.2024.100873.
- [26] S. Suharsono, O. S. Simanjuntak, R. I. Perwira, S. Pambudi, M. Fachrurradjie, Dan Y. P. Aqillasari, “Development Of A Semester Learning Plans System Based On Obe (Outcome-Based Education),” 2024, Hlm. 485–501. Doi: 10.2991/978-2-38476-247-7_51.

- [27] S. Aminah, A. Alfa Krisnadhi, Dan A. Nizar Hidayanto, “Ontological Framework For The Analysis Of Outcome-Based Curriculum In Higher Education,” *Ieee Access*, Vol. 13, Hlm. 31497–31516, 2025, Doi: 10.1109/Access.2025.3542881.
- [28] D. Tanu Wijaya, I. Sumadikarta, Dan B. Panjaitan, “Analisa Dan Perancangan Aplikasi Evaluasi Capaian Pembelajaran Lulusan,” *Prosiding*, Vol. 4, Hlm. 137–147, Nov 2023, Doi: 10.59134/Prosiding.V4i.558.
- [29] O. Pritasari, B. Y. Wilujeng, Dan N. R. Windayani, “Penerapan Kurikulum Outcome Based Education (Obe) Dalam Kurikulum Merdeka Belajar Kurikulum Merdeka Di Prodi S1 Pendidikan Tata Rias,” *Journal Of Vocational And Technical Education (Jvte)*, Vol. 5, No. 1, Hlm. 41–48, Mar 2023, Doi: 10.26740/Jvte.V5n1.P41-48.
- [30] I. Simangunsong, “Rancang Bangun Frontend Sistem Informasi Capaian Pembelajaran Lulusan (Cpl) Berbasis Web Menggunakan Metode Agile Kanban,” *Jurnal Informatika Dan Teknik Elektro Terapan*, Vol. 14, No. 2, Apr 2026, Doi: 10.23960/Jitet.V14i2.9221.
- [31] M. Pertiwi, N. Efranda, B. Erwin Slam, N. Ritha, Dan M. Bettiza, “Ui/Ux Design Of Web-Based For Outcome-Based Education (Obe) Assessment Information System In Universitas Maritim Raja Ali Haji (Umrah) Using Prototype Method,” *Bio Web Conf.*, Vol. 134, Hlm. 01006, Okt 2024, Doi: 10.1051/Bioconf/202413401006.
- [32] M. Naufal Gibran, “Sistem Informasi Penilaian Berbasis Outcome Based Learning Guna Monitoring Pemenuhan Capaian Pembelajaran,” *Jurnal Nasional Komputasi Dan Teknologi Informasi (Jnkti)*, Vol. 8, No. 6, 2025.
- [33] T. Rak, J. Drabek, Dan M. Charytanowicz, “Performance-Based Classification Of Users In A Containerized Stock Trading Application Environment Under Load,” *Electronics (Basel)*, Vol. 14, No. 14, Hlm. 2848, Jul 2025, Doi: 10.3390/Electronics14142848.
- [34] A. Sarikaya, M. Correll, L. Bartram, M. Tory, Dan D. Fisher, “What Do We Talk About When We Talk About Dashboards?,” *Ieee Trans. Vis. Comput. Graph.*, Vol. 25, No. 1, Hlm. 682–692, Jan 2019, Doi: 10.1109/Tvcg.2018.2864903.
- [35] R. M. Harden, J. R. Crosby, Dan M. H. Davis, “Amees Guide No. 14: Outcome-Based Education: Part 1 - An Introduction To Outcome-Based Education,” *Med. Teach.*, Vol. 21, No. 1, Hlm. 7–14, Jan 1999, Doi: 10.1080/01421599979969.
- [36] A. S. Azzahidi, B. Wijayanto, Dan A. Darmawan, “Performance Evaluation Of Backend Frameworks For Rest Api: A Comparative

- Study Of Spring Boot, Flask, Express.Js, Laravel Frankenphp, And Gin,” *Jurnal Teknik Informatika (Jutif)*, Vol. 6, No. 4, Hlm. 2405–2419, Agu 2025, Doi: 10.52436/1.Jutif.2025.6.4.4811.
- [37] A. Lamer, C. Saint-Dizier, N. Paris, Dan E. Chazard, “Data Lake, Data Warehouse, Datamart, And Feature Store: Their Contributions To The Complete Data Reuse Pipeline,” *Jmir Med. Inform.*, Vol. 12, Hlm. E54590–E54590, Jul 2024, Doi: 10.2196/54590.
 - [38] A. Barbaro Dkk., “Data Acquisition, Processing, And Aggregation In A Low-Cost Iot System For Indoor Environmental Quality Monitoring,” *Applied Sciences*, Vol. 14, No. 10, Hlm. 4021, Mei 2024, Doi: 10.3390/App14104021.
 - [39] F. Zammetti, *Modern Full-Stack Development*. Berkeley, Ca: Apress, 2020. Doi: 10.1007/978-1-4842-5738-8.
 - [40] E. Nurhayati Dan A. Agussalim, “Rancang Bangun Back-End Api Pada Aplikasi Mobile Ayamhub Menggunakan Framework Node Js Express,” *Jurnal Sistem Dan Teknologi Informasi (Justin)*, Vol. 11, No. 3, Hlm. 524, Jul 2023, Doi: 10.26418/Justin.V11i3.66823.
 - [41] A. D. Ramadhan Dan Y. Prayudi, “Implementasi Object-Relational Mapping (Orm) Prisma Dalam Perancangan Restful Api Untuk Web Sda Division Di Pt Telkom Indonesia Tbk,” *Technologia : Jurnal Ilmiah*, Vol. 16, No. 2, Hlm. 256, Apr 2025, Doi: 10.31602/Tji.V16i2.17880.
 - [42] K. Kroc, O. Kizun, Dan M. Skublewska-Paszkowska, “Perfomance Analysis Of Relational Databases Mysql, Postgresql, Mariadb And H2,” *Journal Of Computer Sciences Institute*, Vol. 14, Hlm. 1–7, Mar 2020, Doi: 10.35784/Jcsi.1565.
 - [43] S. Di Meglio, L. L. L. Starace, Dan S. Di Martino, “Web App Performance Testing In Industrial Contexts: Supporting Workload Generation With E2e-Loader++,” *Journal Of Systems And Software*, Vol. 232, Hlm. 112684, Feb 2026, Doi: 10.1016/J.Jss.2025.112684.
 - [44] A. Ali, H. A. Maghawry, Dan N. Badr, “Performance Testing As A Service Using Cloud Computing Environment: A Survey,” *Journal Of Software: Evolution And Process*, Vol. 34, No. 12, Des 2022, Doi: 10.1002/Smr.2492.
 - [45] Ekky Novriza Alam Dan Fitriyana Dewi, “Performance Testing Analysis Of Bandungtanginas Application With Jmeter,” *International Journal Of Innovation In Enterprise System*, Vol. 6, No. 2, Hlm. 157–166, Okt 2024, Doi: 10.25124/Ijies.V6i02.172.

- [46] S. Matam Dan J. Jain, *Pro Apache Jmeter*. Berkeley, Ca: Apress, 2017. Doi: 10.1007/978-1-4842-2961-3.
- [47] Joko Yuwono, “Analisis Kinerja Sistem E-Learning Universitas Pamulang Menggunakan Load Testing Berbasis Apache Jmeter,” *Jurnal Informatika Dan Teknologi Komputer (Jitek)*, Vol. 6, No. 1, Hlm. 53–66, Mar 2026, Doi: 10.55606/Jitek.V6i1.8599.
- [48] A. R. Azizi, N. Andriana, Dan Kristanto, “Performance Testing Aplikasi Sistem Informasi Penatausahaan Dan Perbendaharaan Online (Sippol),” *Jurnal Informatika Polinema*, Vol. 12, No. 1, Hlm. 121–126, Nov 2025, Doi: 10.33795/Jip.V12i1.7432.
- [49] I. Hamidah, I. Haromain, Dan I. M. Drehem, “Evaluasi Pengujian Kinerja Menggunakan Jmeter Untuk Menunjang Stabilitas Aplikasi Layanan Perbankan Pada Pt Bank Rakyat Indonesia Tbk,” *Dbesti: Journal Of Digital Business And Technology Innovation*, Vol. 2, No. 1, Hlm. 114–126, Jun 2025, Doi: 10.54914/Dbesti.V2i1.1621.
- [50] N. L. A. Sonia Ginasari, K. Suar Wibawa, Dan N. K. Ayu Wirdiani, “Pengujian Stress Testing Api Sistem Pelayanan Dengan Apache Jmeter,” *Jitter : Jurnal Ilmiah Teknologi Dan Komputer*, Vol. 2, No. 3, Hlm. 552, Nov 2021, Doi: 10.24843/Jtrti.2021.V02.I03.P14.
- [51] M. A. Puspita, M. Mardiana, R. Ariesta, Dan M. A. Muhammad, “User Acceptance Testing Dan Performance Testing Pada Pengembangan Website Kanal Pengetahuan Dikti,” *Journal Of Information Engineering And Educational Technology*, Vol. 8, No. 2, Hlm. 96–101, Mei 2025, Doi: 10.26740/Jieet.V8n2.P96-101.
- [52] I. Indrianto, “Performance Testing On Web Information System Using Apache Jmeter And Blazemeter,” *Jurnal Ilmiah Ilmu Terapan Universitas Jambi*, Vol. 7, No. 2, Hlm. 138–149, Des 2023, Doi: 10.22437/Jiituj.V7i2.28440.
- [53] F. P. E. Putra, A. Zulfikri, A. Rohman, Dan R. Alim, “Analysis Comparative Of Performance Optimization Techniques For Php Framework Testing: Laravel, Codeigniter, Symfony,” *Brilliance: Research Of Artificial Intelligence*, Vol. 5, No. 1, Hlm. 242–248, Jun 2025, Doi: 10.47709/Brilliance.V5i1.5989.
- [54] R. C. Fitonah, Y. Ardilla, M. Ridwan, Dan A. T. Wibowo, “Evaluasi Kinerja Website E-Procurement Pada Pt Xyz Di Surabaya Berbasis Iso 25010,” *Journal Of Informatics Management And Information Technology*, Vol. 5, No. 4, Hlm. 437–446, 2025, Doi: 10.47065/Jimat.V5i4.802.
- [55] Apache Software Foundation, “Apache Jmeter User’s Manual: Glossary,” Apache Jmeter Documentation. Diakses: 1 Mei 2026.

[Daring]. Tersedia Pada:
<https://jmeter.apache.org/usermanual/glossary.html>

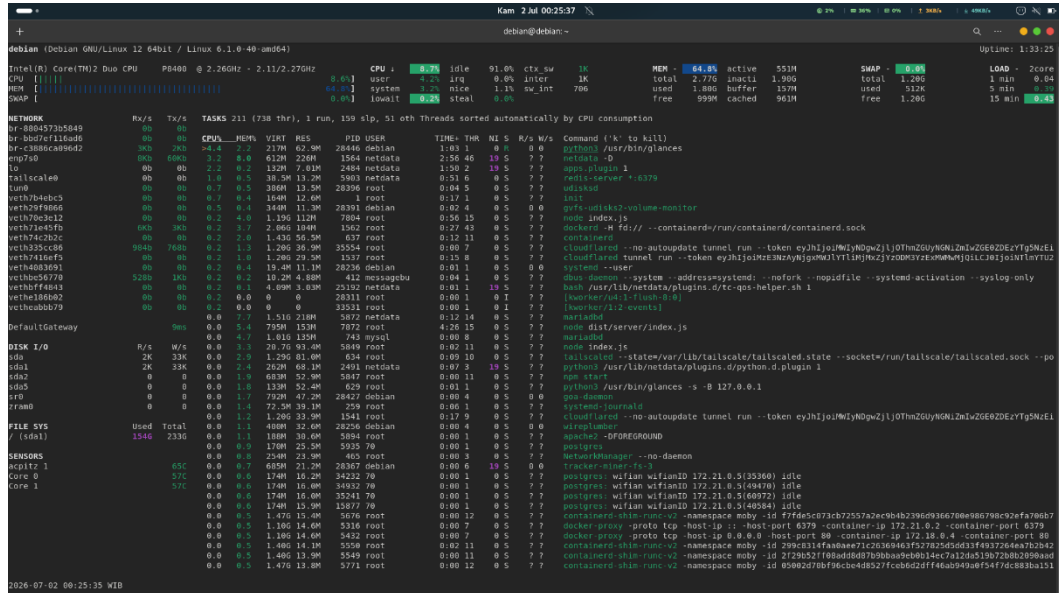
- [56] D. I. Permatasari, “Pengujian Aplikasi Menggunakan Metode Load Testing Dengan Apache Jmeter Pada Sistem Informasi Pertanian,” *Jurnal Sistem Dan Teknologi Informasi (Justin)*, Vol. 8, No. 1, Hlm. 135, Jan 2020, Doi: 10.26418/Justin.V8i1.34452.
- [57] S. A. Sentosa, E. Subyantoro, Dan I. Asrowardi, “Pengukuran Kinerja Pada Aplikasi Video Pembelajaran Umkm Berbasis Web Dengan Metode Pengujian Beban,” *Routers: Jurnal Sistem Dan Teknologi Informasi*, Hlm. 95–102, Apr 2024, Doi: 10.25181/Rt.V2i2.3407.
- [58] R. Juniantika, N. S. Wibowo, V. Dewangga, Dan D. Rizky, “Optimalisasi Performa Website Lsp Smkn 2 Kraksaan Melalui Pengujian Beban Dengan Apache Jmeter,” *El Sains: Jurnal Elektro*, Vol. 7, No. 1, Hlm. 9–14, Mei 2025, Doi: 10.30996/Elsains.V7i1.13183.
- [59] J. Tang *Dkk.*, “A Comprehensive Review Of Theories, Methods, And Techniques For Bottleneck Identification And Management In Manufacturing Systems,” *Applied Sciences*, Vol. 14, No. 17, Hlm. 7712, Agu 2024, Doi: 10.3390/App14177712.
- [60] Albert Yakobus Chandra Dan Putry Wahyu Setyaningsih, “Benchmarking Local Development Environments: Analyzing The Performance Of Xampp, Mamp, And Laragon,” *Bulletin Of Computer Science Research*, Vol. 5, No. 3, Hlm. 193–206, Apr 2025, Doi: 10.47065/Bulletincsr.V5i3.493.
- [61] N. V. R. S. C. Gupta Lakkimsetty, “Database Optimization Strategies: Enhancing Performance And Scalability,” *International Journal Of Computer Science And Mobile Computing*, Vol. 12, No. 11, Hlm. 69–89, Nov 2023, Doi: 10.47760/Ijcsmc.2023.V12i11.006.
- [62] A. Anchlia, “Enhancing Query Performance Through Relational Database Indexing,” *International Journal Of Computer Trends And Technology*, Vol. 72, No. 8, Hlm. 130–133, Agu 2024, Doi: 10.14445/22312803/Ijctt-V72i8p119.
- [63] R. N. Muzaki Dan A. Salam, “Reducing Under-Fetching And Over-Fetching In Rest Api With GraphQL For Web-Based Software Development,” *Jurnal Teknik Informatika (Jutif)*, Vol. 5, No. 2, Hlm. 447–453, Apr 2024, Doi: 10.52436/1.Jutif.2024.5.2.1725.
- [64] M. V. Privalov Dan M. V. Stupina, “Improving Web-Oriented Information Systems Efficiency Using Redis Caching Mechanisms,” *Indonesian Journal Of Electrical Engineering And Computer Science*,

Vol. 33, No. 3, Hlm. 1667, Mar 2024, Doi: 10.11591/Ijeecs.V33.I3.Pp1667-1675.

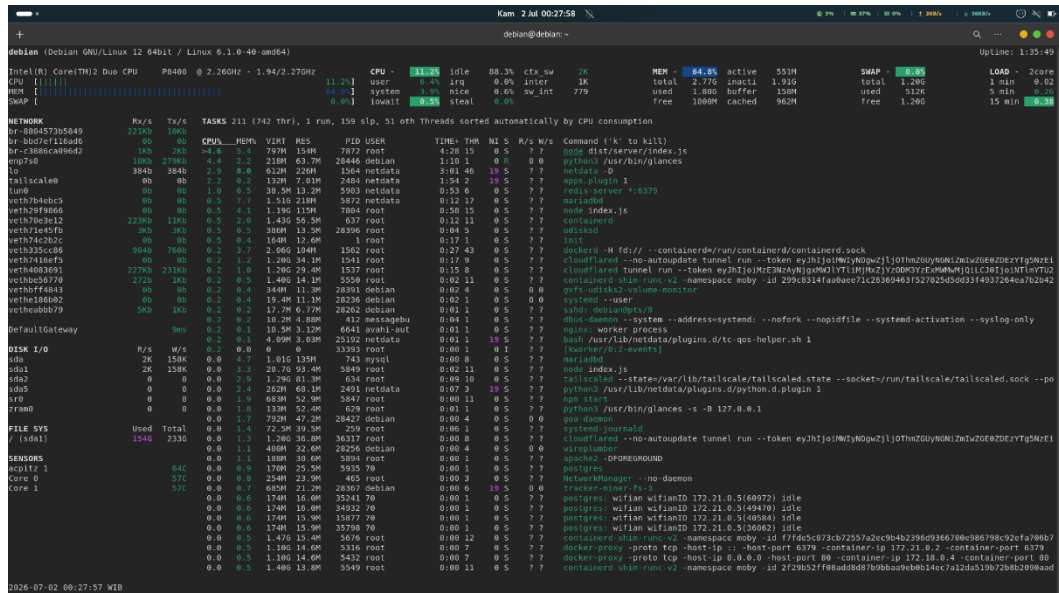
- [65] D. S. Pereira, L. F. V. Bezerra, J. S. Nunes, I. M. Barroca Filho, Dan F. A. S. Lopes, "Performance Efficiency Evaluation Based On Iso/Iec 25010:2011 Applied To A Case Study On Load Balance And Resilient," Dalam *Anais Do Xxiv Workshop De Testes E Tolerância A Falhas (Wtf 2023)*, Sociedade Brasileira De Computação - Sbc, Mei 2023, Hlm. 108–120. Doi: 10.5753/Wtf.2023.787.
- [66] D. Yuniasri, P. Damayanti, Dan S. Rochimah, "Performance Efficiency Evaluation Frameworks Based On Iso 25010," Dalam *2020 10th Electrical Power, Electronics, Communications, Controls And Informatics Seminar (Eeccis)*, Ieee, Agu 2020, Hlm. 254–258. Doi: 10.1109/Eeccis49483.2020.9263432.
- [67] A. N. Rachman, R. N. Shofa, I. N. Sjamsuddin, G. N. Tarempa, I. T. Julianto, Dan B. A. Athoillah, "Load Testing-Based Performance Evaluation Of The Siukt Api System," *Journal Of Intelligent Systems Technology And Informatics*, Vol. 2, No. 1, Hlm. 22–29, Mar 2026, Doi: 10.64878/Jistics.V2i1.89.
- [68] Microsoft Research, "Garnet: A High-Performance, Multi-Threaded Cache Store," Microsoft Research. Diakses: 5 Juli 2026. [Daring]. Tersedia Pada: [Https://Microsoft.Github.Io/Garnet/](https://Microsoft.Github.Io/Garnet/)
- [69] J. C. Yusmita, R. Arya, J. M. Wijaya, K. M. Suryaningrum, Dan R. R. Siswanto, "Optimizing Database Access Strategy: A Performance Analysis Comparison Of Raw Sql And Prisma Orm," *Procedia Comput. Sci.*, Vol. 269, Hlm. 1201–1210, 2025, Doi: 10.1016/J.Procs.2025.09.061.
- [70] B. Rozak, E. Erizal, Dan F. Noor Hasan, "Implementasi Teknik Clustering Untuk Meningkatkan Performa Aplikasi Node Js," *Smart Comp: Jurnalnya Orang Pintar Komputer*, Vol. 12, No. 4, Okt 2023, Doi: 10.30591/Smartcomp.V12i4.4532.
- [71] Y. Prilyansyah, "Optimasi Performa Api Aplikasi E-Commerce Menggunakan Redis Caching," 2025. Diakses: 14 Juli 2026. [Daring]. Tersedia Pada: [Https://Newjournal.Lppmunindra.Ac.Id/Index.Php/String/Article/View/957/134](https://Newjournal.Lppmunindra.Ac.Id/Index.Php/String/Article/View/957/134)
- [72] Microsoft Research, "Garnet Benchmarking Results: Resp Bench," Microsoft Garnet Official Documentation. Diakses: 14 Juli 2026. [Daring]. Tersedia Pada: [Https://Microsoft.Github.Io/Garnet/Docs/Benchmarking/Results-Resp-Bench](https://Microsoft.Github.Io/Garnet/Docs/Benchmarking/Results-Resp-Bench)

LAMPIRAN

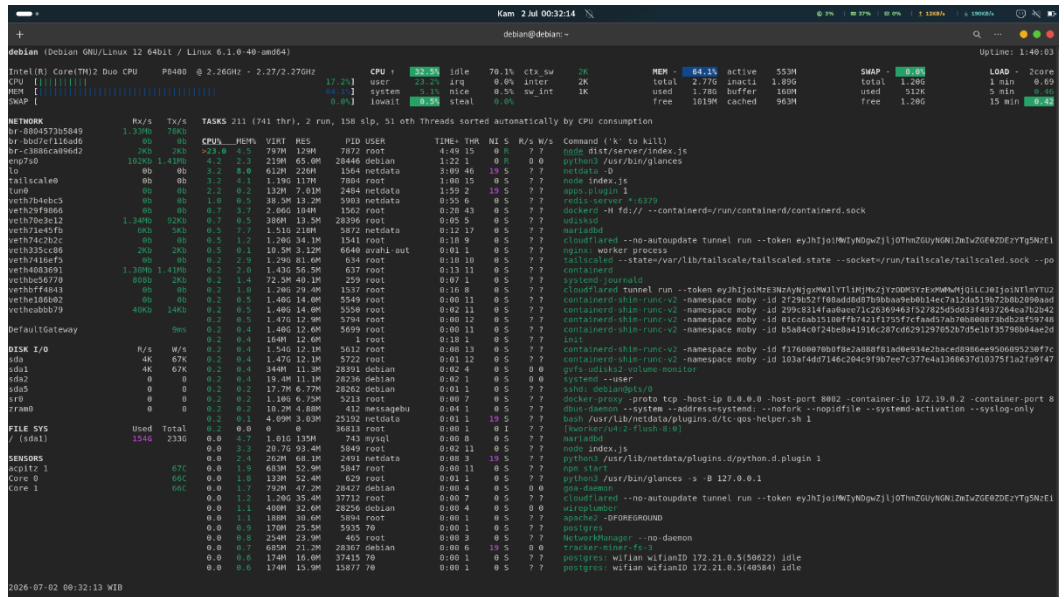
Lampiran 1 Tangkapan Layar Pemantauan CPU dan Memori *Server* Lokal Menggunakan Glances pada Kondisi *Idle* Sebelum Pengujian



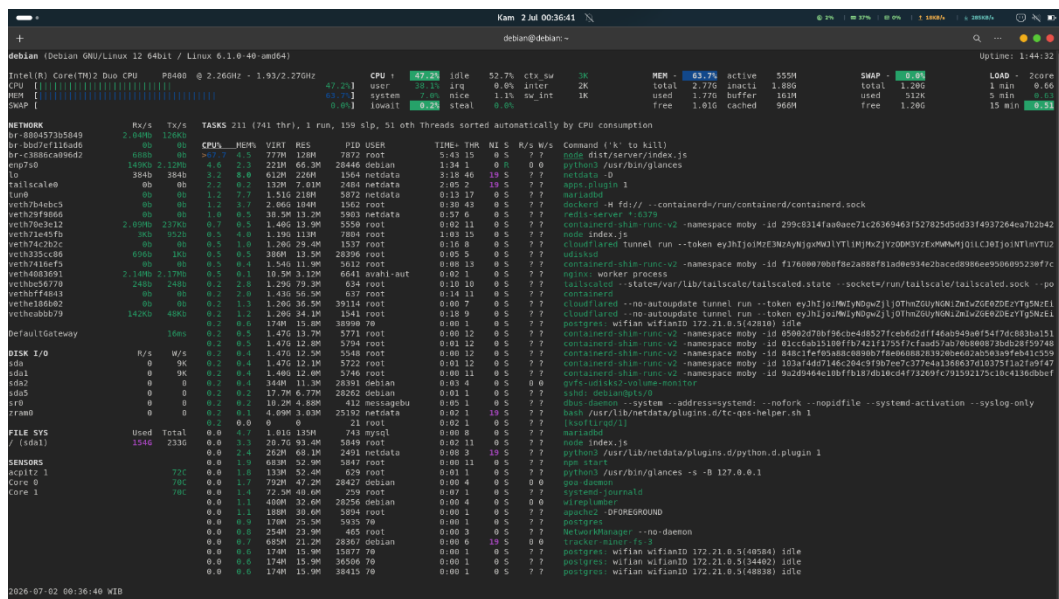
Lampiran 2 Tangkapan Layar Pemantauan CPU dan Memori *Server* Lokal Menggunakan Glances pada Skenario *Baseline* 1 Pengguna Virtual



Lampiran 3 Tangkapan Layar Pemantauan CPU dan Memori Server Lokal Menggunakan Glances pada Skenario 10 Pengguna Virtual

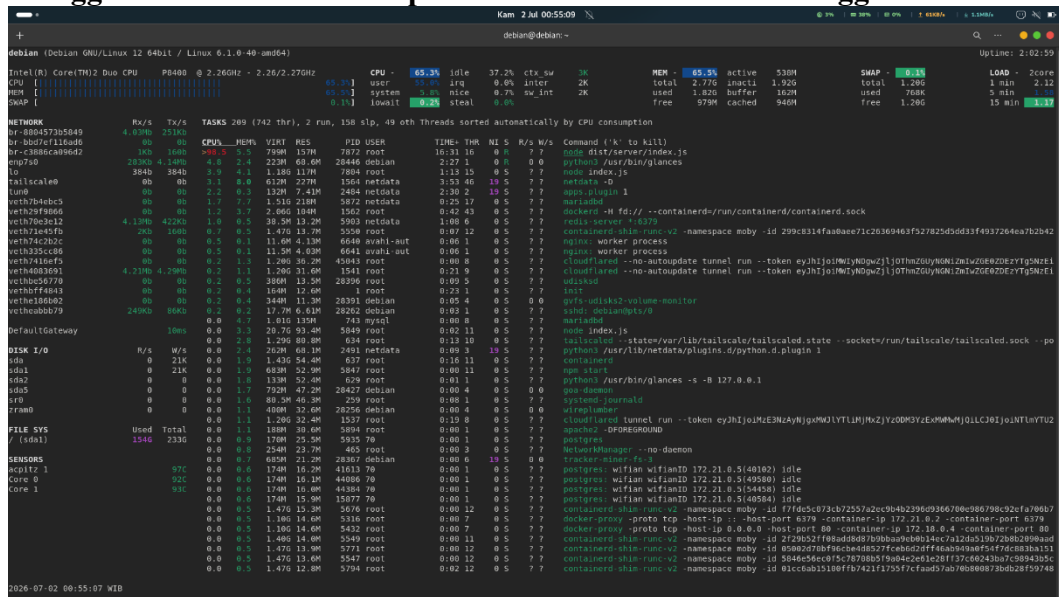


Lampiran 4 Tangkapan Layar Pemantauan CPU dan Memori Server Lokal Menggunakan Glances pada Skenario 20 Pengguna Virtual

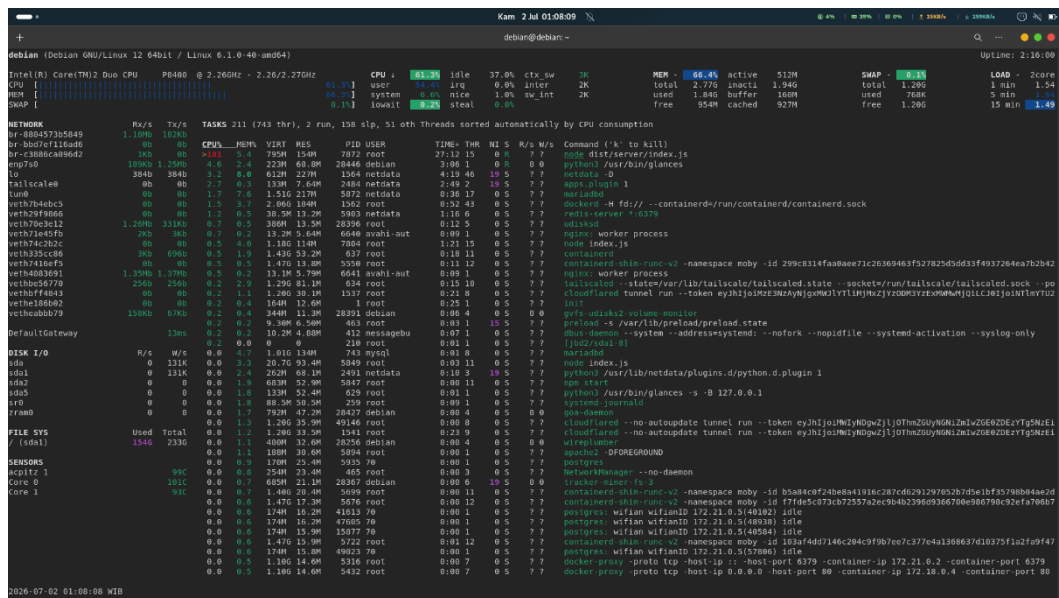


[illegible][illegible]

Lampiran 7 Tangkapan Layar Pemantauan CPU dan Memori *Server* Lokal Menggunakan Glances pada Skenario 200 Pengguna Virtual



Lampiran 8 Tangkapan Layar Pemantauan CPU dan Memori *Server* Lokal Menggunakan Glances pada Skenario 500 Pengguna Virtual

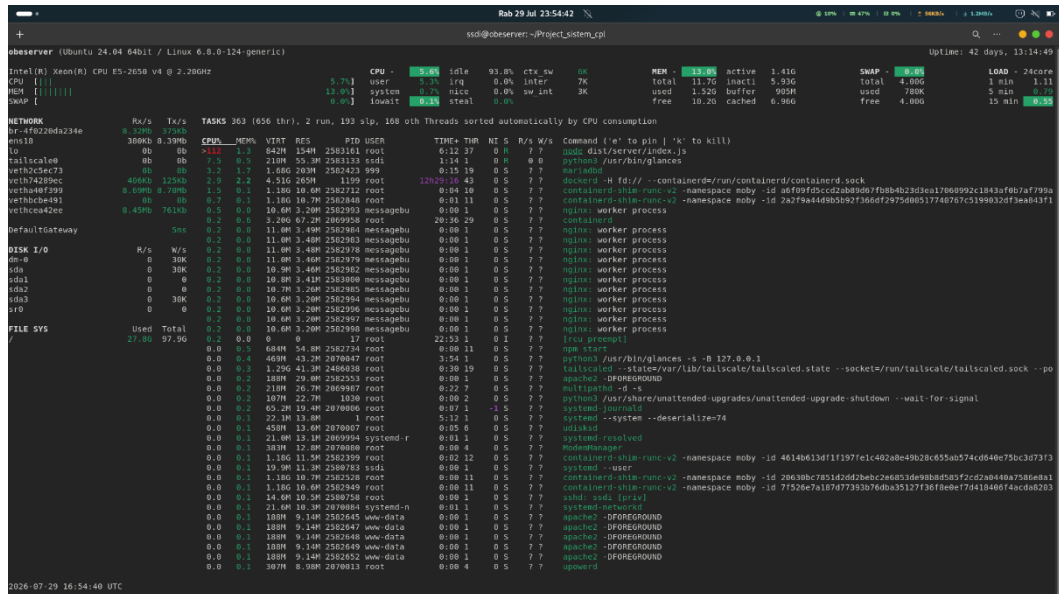


[illegible][illegible]

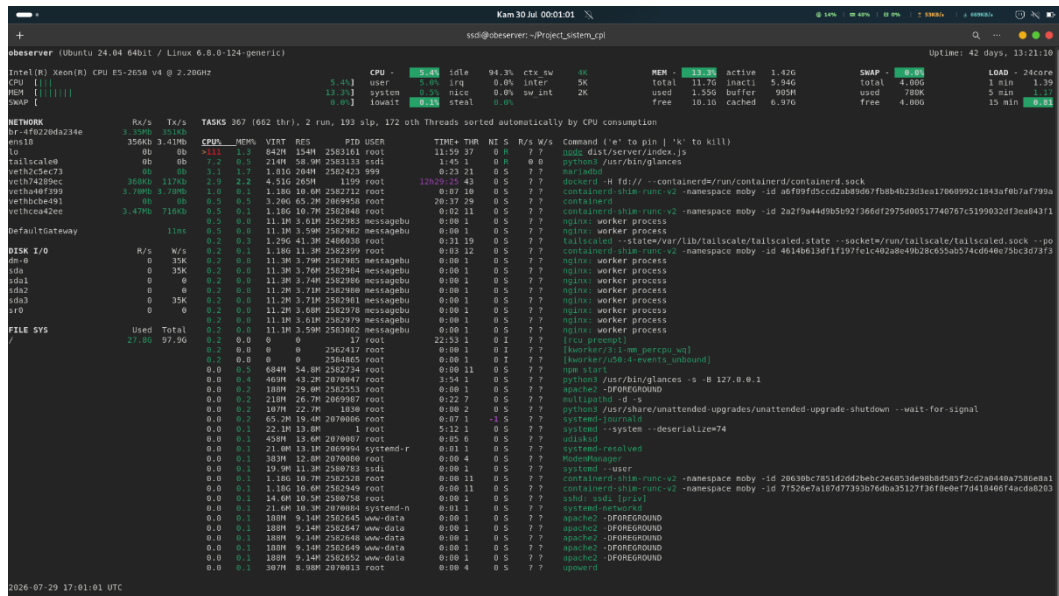
[illegible][illegible]

[illegible][illegible]

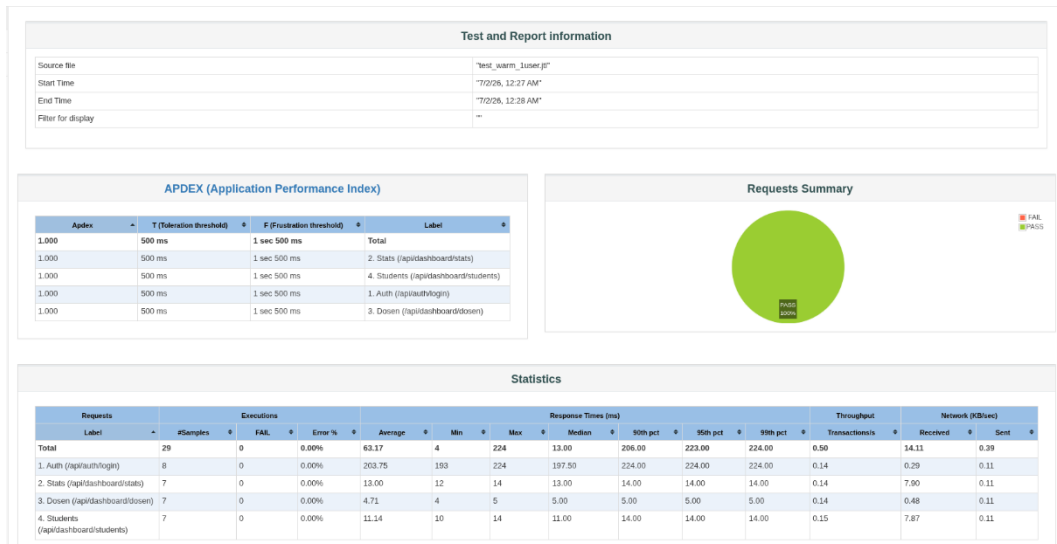
Lampiran 15 Tangkapan Layar Pemantauan CPU dan Memori VPS UNUGHA Menggunakan Glances pada Skenario 200 Pengguna Virtual



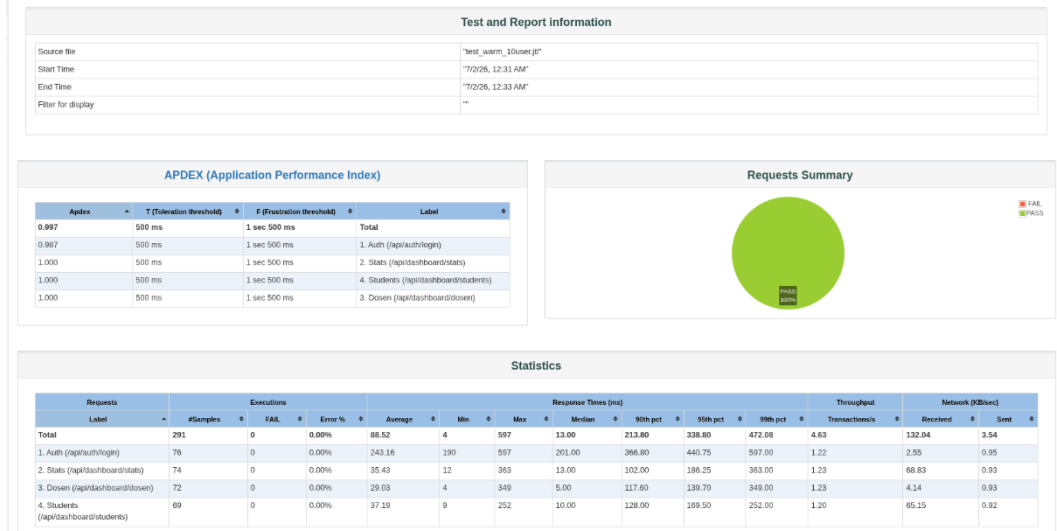
Lampiran 16 Tangkapan Layar Pemantauan CPU dan Memori VPS UNUGHA Menggunakan Glances pada Skenario 500 Pengguna Virtual



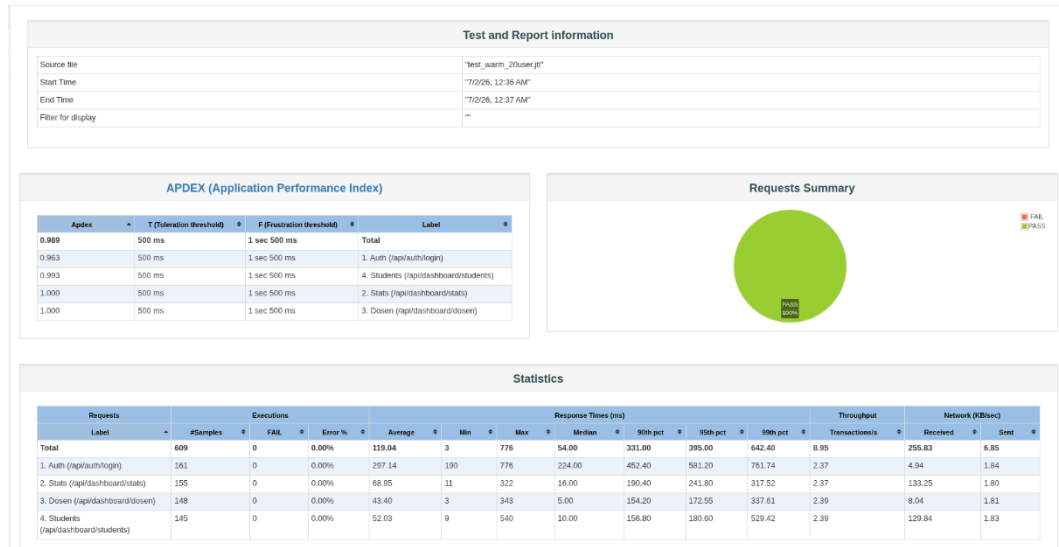
Lampiran 17 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario *Baseline* 1 Pengguna Virtual



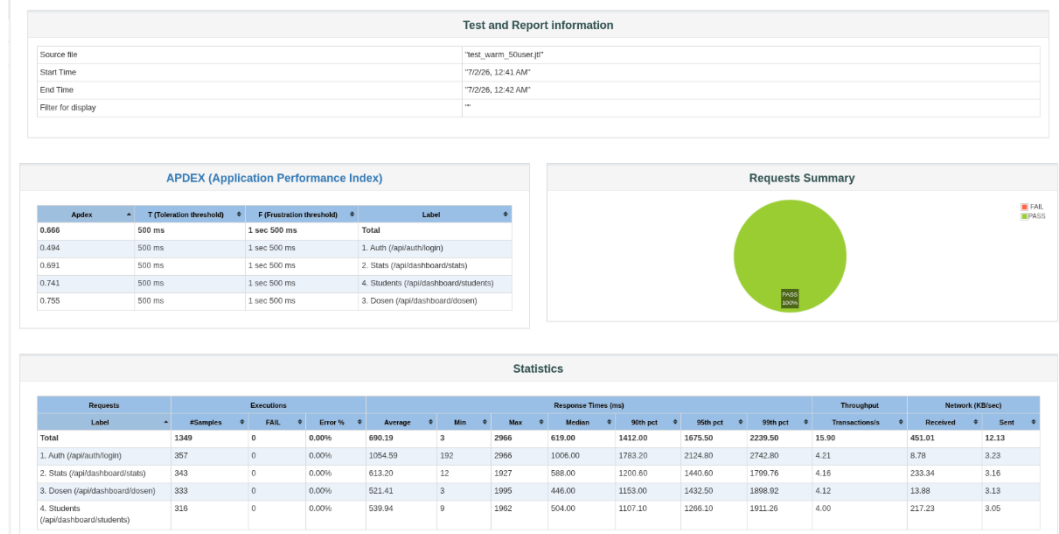
Lampiran 18 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 10 Pengguna Virtual



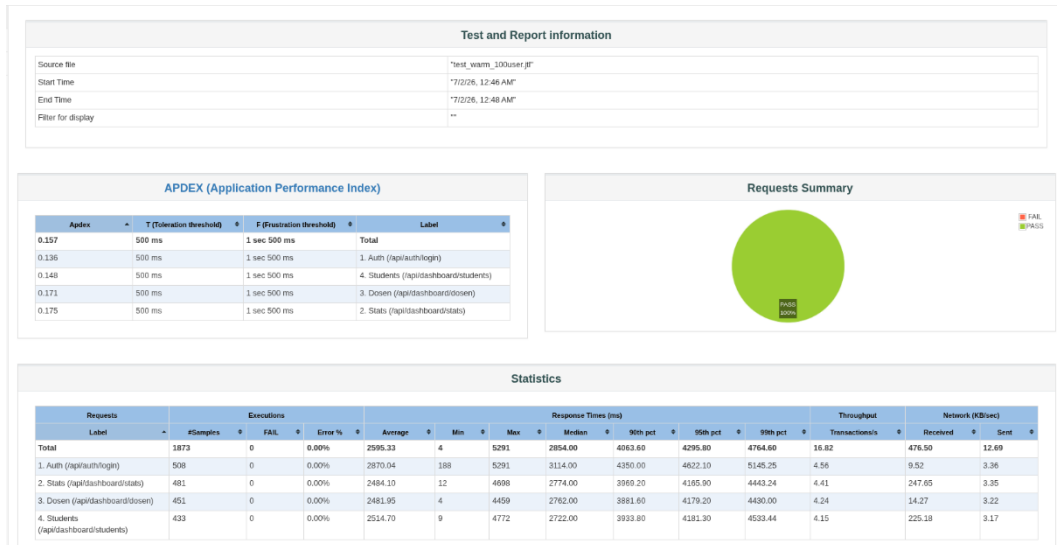
Lampiran 19 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 20 Pengguna Virtual



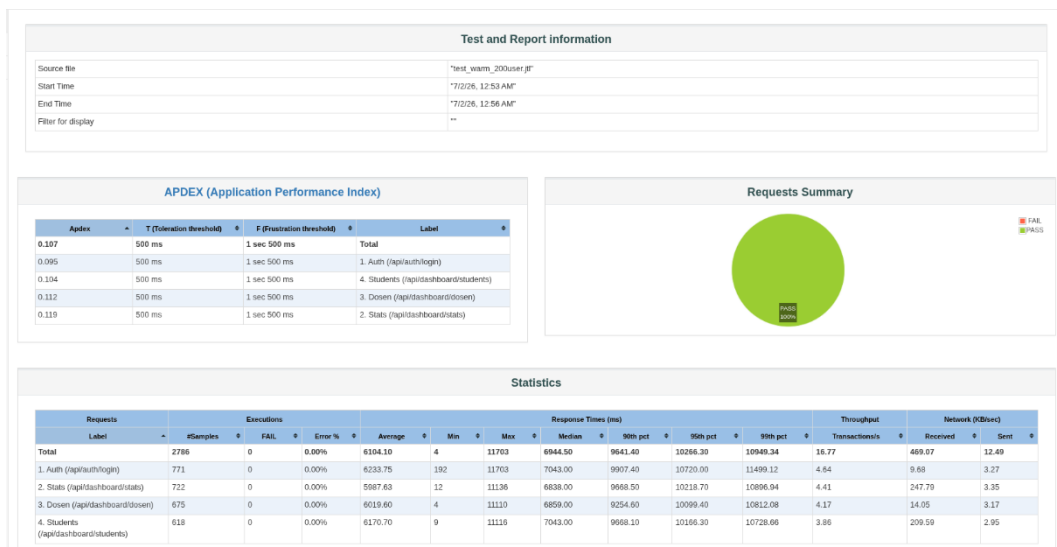
Lampiran 20 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 50 Pengguna Virtual



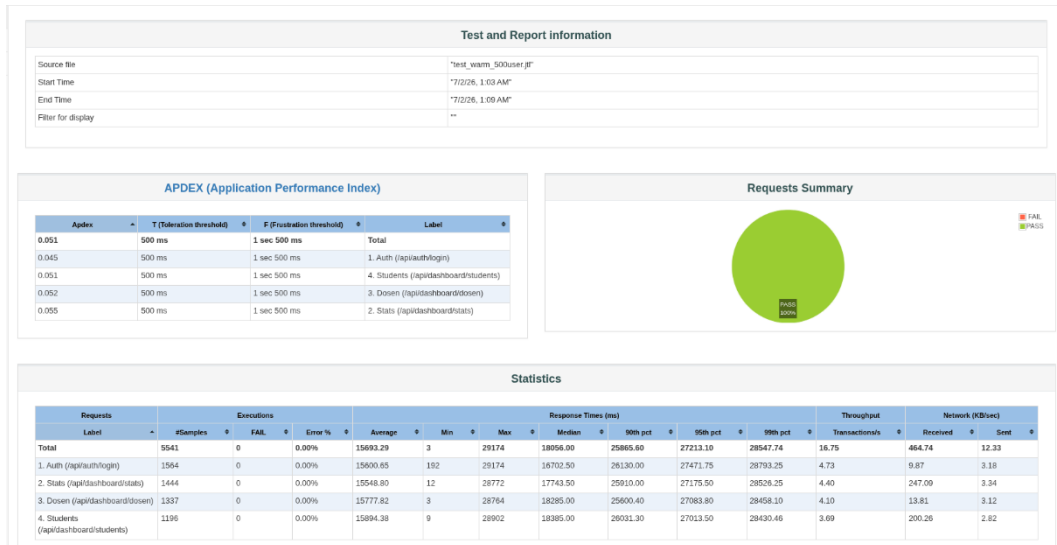
Lampiran 21 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 100 Pengguna Virtual



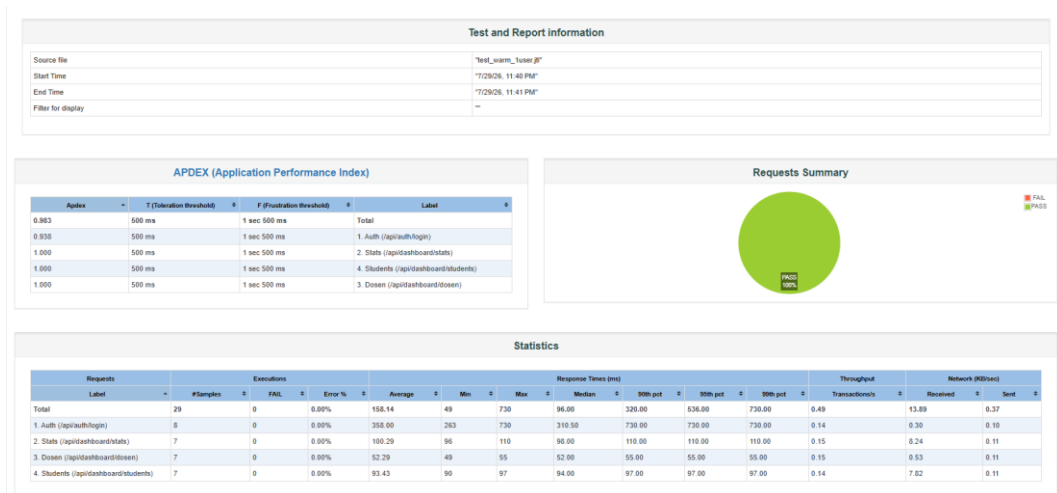
Lampiran 22 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 200 Pengguna Virtual



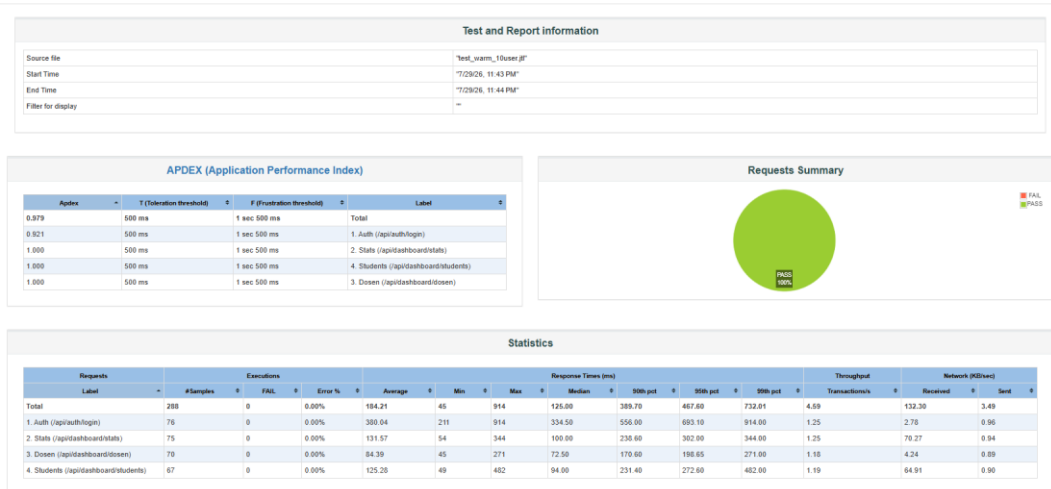
Lampiran 23 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada *Server* Lokal dengan Skenario 500 Pengguna Virtual



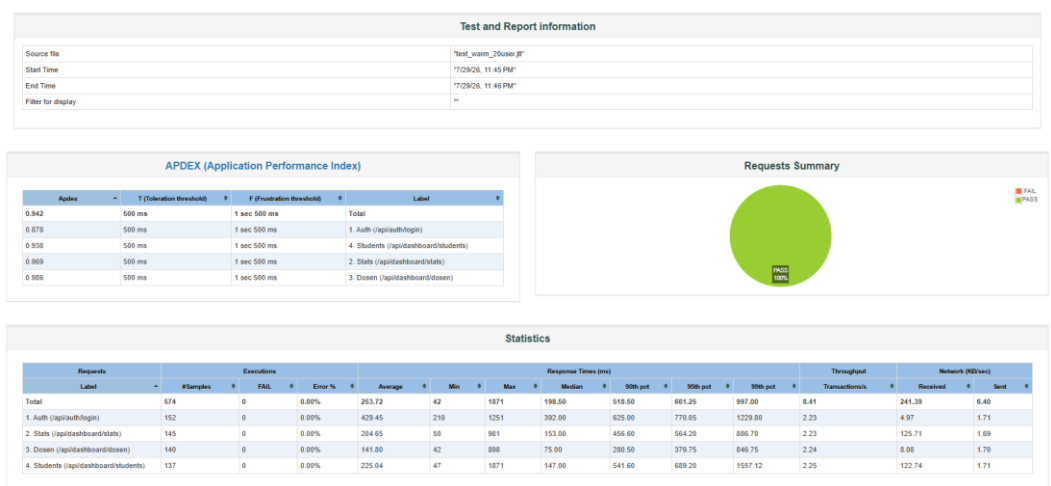
Lampiran 24 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario *Baseline* 1 Pengguna Virtual



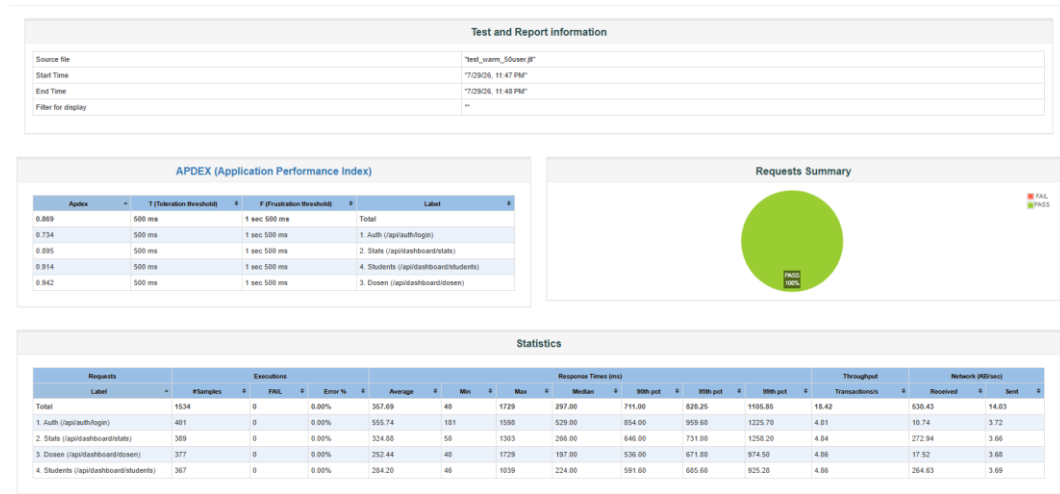
Lampiran 25 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 10 Pengguna Virtual



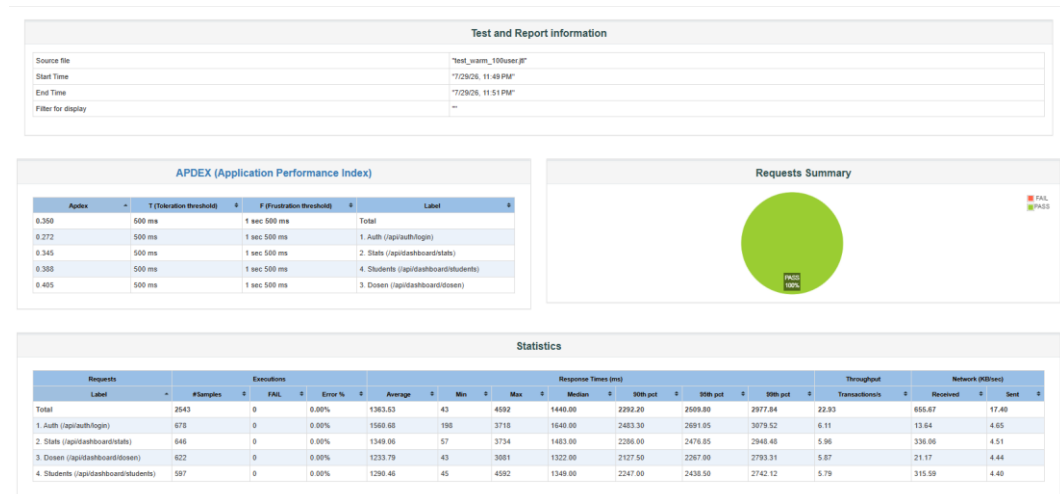
Lampiran 26 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 20 Pengguna Virtual



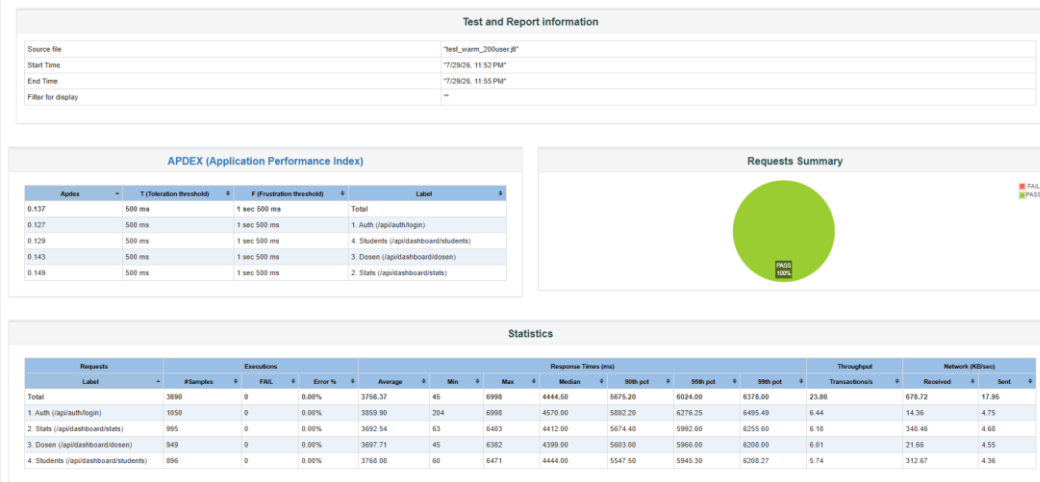
Lampiran 27 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 50 Pengguna Virtual



Lampiran 28 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 100 Pengguna Virtual



Lampiran 29 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 200 Pengguna Virtual



Lampiran 30 Tangkapan Layar Laporan Hasil Pengujian Apache JMeter pada VPS UNUGHA dengan Skenario 500 Pengguna Virtual

