

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

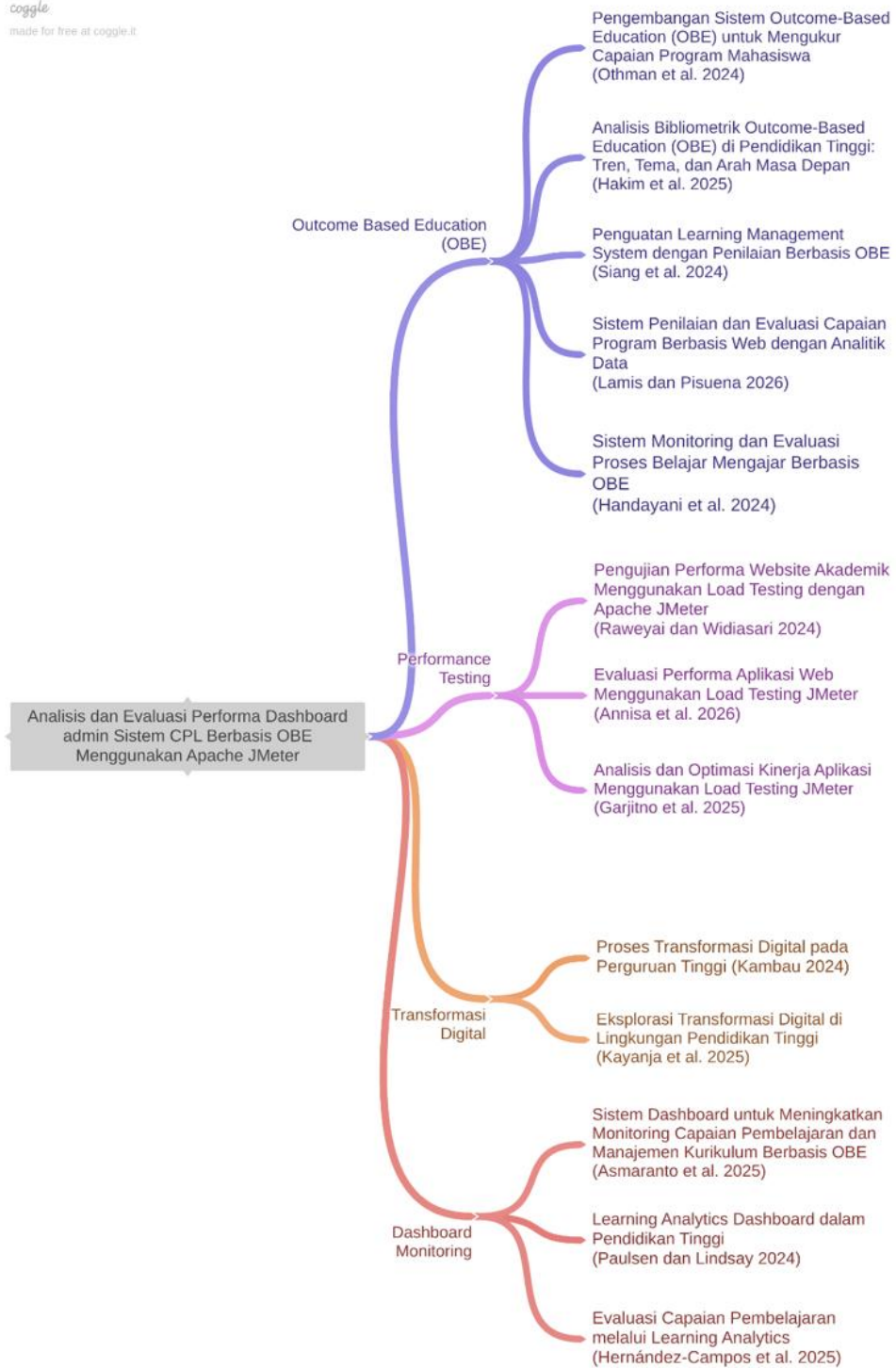
pengembangan sistem, evaluasi capaian pembelajaran, atau penyusunan indikator *monitoring*, belum pada pengujian performa *Endpoint* sistem ketika menerima *Request* pengguna secara simultan.

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.