BAB I
PENDAHULUAN

1.1 Latar Belakang

Transformasi digital dalam sektor pendidikan mendorong adopsi *Learning Management System* (*LMS*) sebagai pusat pengelolaan aktivitas akademik, mulai dari distribusi materi hingga evaluasi hasil belajar [1], [2]. *Moodle* sebagai *LMS open-source* yang paling banyak digunakan secara global [2] masih mempertahankan arsitektur monolitik berbasis *Hypertext Preprocessor* (PHP), di mana seluruh komponen aplikasi berbagi satu lingkungan proses dan basis data [3]. Arsitektur ini efektif pada skala kecil-menengah, namun rawan *bottleneck* ketika beban pengguna meningkat tajam dalam waktu singkat, seperti pada pekan ujian daring.

Di antara modul-modul *Moodle*, modul *grading* pada jalur *submit quiz* tergolong paling berat secara komputasi karena setiap pengiriman jawaban memicu operasi pencocokan jawaban, perhitungan skor, dan penulisan basis data yang jumlahnya sebanding dengan jumlah soal per pengguna [3]. Pada arsitektur monolitik, proses ini dijalankan secara sinkron dan berbagi sumber daya dengan modul lain seperti autentikasi dan navigasi konten, sehingga satu sesi pengiriman jawaban oleh ratusan pengguna secara bersamaan dapat memicu antrean permintaan yang menumpuk pada server.

Untuk memastikan permasalahan ini bukan isu yang telah diselesaikan pada versi *Moodle* terbaru, dilakukan penelusuran pada forum resmi komunitas *Moodle* ("Tips for increase quiz performance", moodle.org). Pengguna pada forum tersebut melaporkan penurunan performa saat ratusan siswa mengirim jawaban kuis secara bersamaan pada *Moodle* versi 4.2, dan salah satu kontributor dokumentasi resmi *Moodle* menjelaskan bahwa setiap pertanyaan yang dikirim memicu operasi basis data tersendiri, sehingga beban server meningkat sebanding dengan jumlah soal dan jumlah pengguna aktif [4]. Pengujian beban yang dibagikan pada forum yang sama turut menunjukkan bahwa satu sesi pengiriman jawaban dapat molor hingga rata-rata 100 detik (deviasi standar 20 detik) ketika diuji pada 1.000 pengguna simultan, dengan sebagian permintaan gagal diselesaikan pada beban ekstrem tersebut [4]. Temuan ini menegaskan bahwa karakteristik beban komputasi pada jalur *submit*

quiz bersumber dari desain arsitektur monolitik itu sendiri, bukan dari *bug* spesifik versi, sehingga tetap relevan pada *Moodle* versi 5.0.2 yang digunakan dalam penelitian ini [5].

Pendekatan konvensional seperti peningkatan spesifikasi perangkat keras (*vertical scaling*) atau *load balancing* pada level infrastruktur dapat meningkatkan ketersediaan layanan dalam jangka pendek, tetapi tidak mengatasi akar permasalahan pada level arsitektur aplikasi dan menimbulkan biaya operasional tinggi untuk menangani lonjakan beban yang bersifat temporer [3], [6].

Sebagai alternatif, penelitian terkini mengarah pada arsitektur *Microservice* yang memungkinkan modul berbeban tinggi seperti *grading* diisolasi dari sistem utama agar tidak mengganggu stabilitas layanan inti *LMS* [7], [8]. Namun, migrasi penuh (*greenfield*) pada sistem matang seperti *Moodle* dinilai kurang praktis karena tingginya kebutuhan sumber daya dan risiko gangguan operasional selama transisi [9], sehingga pola migrasi inkremental seperti *Strangler Fig* atau *Outpost Pattern* lebih sesuai untuk mengekstraksi fungsionalitas tertentu tanpa menulis ulang keseluruhan sistem [10].

Berdasarkan hal tersebut, penelitian ini mengkaji jalur *request submit quiz Moodle* melalui pendekatan *refactoring* parsial menggunakan arsitektur *Hybrid Microservice* dengan pola *Outpost*. *Moodle core* tetap dipertahankan sebagai sistem utama, sedangkan Node.js *Grading Service* ditambahkan sebagai layanan perantara pada jalur *request* menuju *endpoint processattempt.php*, menampung *request* ke dalam *in-memory queue*, dan meneruskannya secara terkontrol melalui *worker pool* [8], [11], [12]. Karena antrean yang digunakan masih bersifat *in-memory* dan belum menjamin persistensi data apabila layanan mengalami gangguan, penelitian ini difokuskan sebagai pengujian kelayakan *refactoring* parsial berupa *proof-of-concept*, bukan pemindahan penuh proses *grading* ke layanan mikro terpisah.

Untuk melaksanakan penelitian ini, digunakan metode *Design Science Research* (*DSR*), yaitu pendekatan yang berorientasi pada perancangan dan evaluasi artefak teknologi sebagai solusi atas permasalahan nyata, bukan pada pengujian hipotesis seperti pada metode *natural science* [13]. Metode ini dipilih karena kerangka tahapannya, identifikasi masalah, perumusan kebutuhan, perancangan-pengembangan artefak, demonstrasi, hingga evaluasi, selaras dengan alur penelitian

ini: mulai dari analisis *bottleneck* pada modul *grading*, perancangan Node.js *Grading Service*, hingga pengujian beban untuk mengevaluasi dampaknya terhadap performa dan ketahanan sistem *Moodle* [13].

Berdasarkan uraian tersebut, tugas akhir ini diarahkan pada *refactoring* parsial jalur *submit quiz Moodle* menggunakan pendekatan *Hybrid Microservice* dengan pola *Outpost* untuk mengevaluasi dampaknya terhadap performa dan ketahanan sistem *LMS*, serta menjadi referensi awal dalam penyesuaian arsitektur *LMS* berbasis monolitik secara bertahap.

1.2 Rumusan Masalah

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

1. Bagaimana mengimplementasikan arsitektur *Hybrid Microservice* mekanisme *queue-based load leveling* melalui Node.js *Grading Service* sebagai lapisan perantara untuk mengatur *request* menuju *endpoint processattempt.php* pada *Moodle*?
2. Bagaimana dampak penerapan arsitektur *Hybrid Microservice menggunakan* mekanisme *queue-based load leveling* dengan pola *Outpost* terhadap performa dan ketahanan sistem *LMS Moodle* pada kondisi beban tinggi dibandingkan dengan arsitektur monolitik *baseline*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan, tujuan utama dari penelitian ini adalah:

1. Mengimplementasikan arsitektur *Hybrid Microservice* menggunakan Node.js *Grading Service* sebagai lapisan *queue-based load leveling* untuk mengatur *request* menuju *endpoint processattempt.php* pada sistem *Moodle* tanpa melakukan perubahan total terhadap *Moodle core*.
2. Mengevaluasi dampak penerapan arsitektur *Hybrid Microservices* dengan pola *Outpost* terhadap performa dan ketahanan sistem Moodle dibandingkan dengan arsitektur monolitik *baseline*, untuk mengetahui efektivitas sistem berdasarkan metrik *p95 response time, error rate, throughput, CPU utilization*, dan *process attempt duration*.

1.4 Batasan Masalah

Penelitian ini memiliki sejumlah batasan agar ruang lingkup penelitian tetap terarah sesuai dengan tujuan, yaitu:

1. Lingkup Fungsional

   Penelitian terbatas pada *refactoring* parsial jalur *request submit quiz* pada tipe *Automated Quiz* pilihan ganda, khususnya *request* yang menuju *endpoint processattempt.php* . Penelitian ini tidak memindahkan logika kalkulasi nilai *Moodle* ke layanan eksternal. Penelitian juga tidak mencakup penilaian manual seperti esai, unggah tugas, atau forum diskusi.

2. Lingkup Arsitektur

   Penerapan arsitektur *Hybrid Microservice* dengan pola *Outpost* dilakukan dengan mempertahankan *Moodlecore* sebagai sistem utama, sementara jalur *request submit quiz* pada *endpoint processattempt.php* diarahkan melalui Node.js *Grading Service* sebagai lapisan antrean. Penelitian ini tidak melakukan pemindahan penuh logika penilaian *Moodle* maupun penulisan ulang total terhadap sistem *LMS*.

3. Lingkup Teknologi

   Komunikasi antar-komponen dirancang menggunakan pendekatan antrean berbasis *in-memory queue* pada Node.js sebagai *proof-of-concept*. Antrean ini digunakan untuk mengatur aliran *request submit quiz*.

4. Lingkup Pengujian

   Validasi performa dilakukan pada lingkungan lokal terkontrol menggunakan alat uji beban untuk mensimulasikan skenario ujian daring. Fokus pengukuran adalah performa sisi server, bukan aspek antarmuka pengguna. Metrik yang digunakan meliputi p95 *response time*, *error rate*, *throughput*, *CPU utilization*, dan *Process attempt duration* sebagai metrik custom untuk mengamati durasi *request submit quiz*.

5. Eksklusi Fitur Non-Esensial

   Tidak ada perubahan pada antarmuka pengguna *User Interface/User Experience* (UI/UX) *LMS frontend* tetap bawaan. Penelitian juga tidak membahas keamanan jaringan mendalam (seperti mitigasi *DDoS*, *OAuth advanced*) atau *deployment cloud* seperti Kubernetes atau *autoscaling*.

6. Lingkup Metodologi

Penelitian ini menggunakan *Design Science Research* sebagai kerangka utama. Namun, sub-tahap eksplorasi ide seperti *imagine and brainstorm* tidak digunakan sebagai sub-tahap terpisah, karena solusi penelitian telah ditentukan berdasarkan gap penelitian dan kebutuhan sistem, yaitu penerapan *Hybrid /Outpost Microservice* dengan Node.js *queue* pada proses *grading Moodle*. Oleh karena itu, penelitian difokuskan pada perumusan masalah, kebutuhan artefak, desain dan implementasi artefak, demonstrasi, serta evaluasi melalui *load testing*.

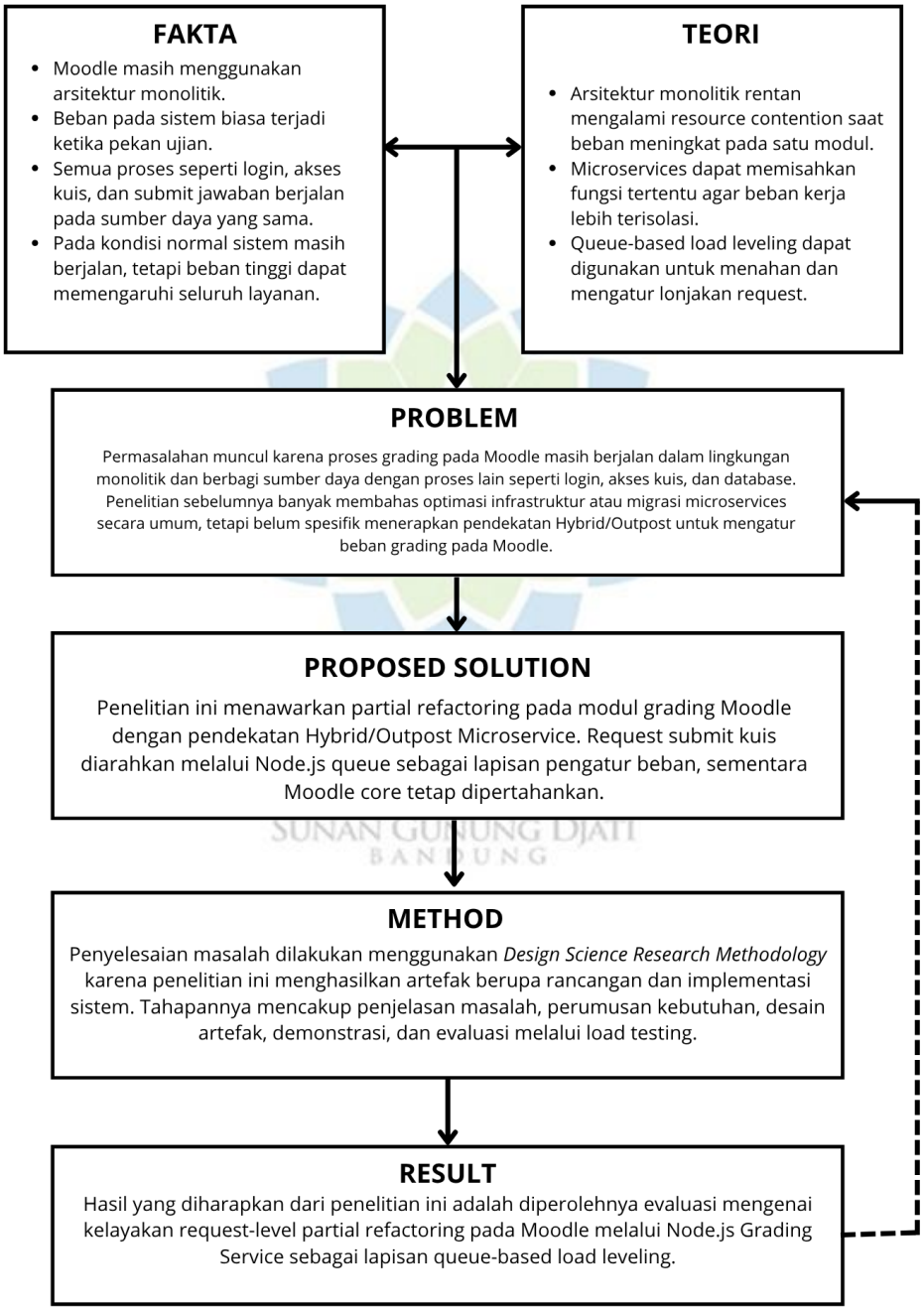
1.5 Manfaat

Secara teoritis, penelitian ini diharapkan dapat memberikan data empiris mengenai penerapan *refactoring* parsial menggunakan arsitektur *Hybrid Microservice* dengan pola *Outpost* pada sistem *LMS* berbasis monolitik. Hasil penelitian ini dapat digunakan sebagai referensi awal untuk memahami dampak penggunaan *queue-based load leveling* terhadap performa dan ketahanan sistem pada skenario beban tinggi.

Secara praktis, penelitian ini diharapkan dapat memberikan gambaran teknis mengenai penerapan Node.js *Grading Service* sebagai lapisan pengatur *request submit quiz* pada *Moodle*. Hasil penelitian juga dapat menjadi dasar pertimbangan bagi pengembangan selanjutnya, khususnya dalam penggunaan *persistent message broker*, pemisahan *resource*, dan *deployment* terdistribusi untuk meningkatkan skalabilitas sistem secara lebih menyeluruh.

1.6 Kerangka Pemikiran

Kerangka pemikiran pada penelitian ini berdasarkan fakta, literatur, masalah, solusi, metodologi yang digunakan, hingga hasil yang diharapkan dalam penelitian ini.



Gambar 1. 1 Kerangka Pemikiran

Gambar 1.1 memvisualisasikan kerangka berpikir penelitian yang disusun untuk menjawab permasalahan kinerja *LMS* sebagaimana telah diuraikan pada latar belakang. Kerangka ini diawali dengan identifikasi akar permasalahan teknis, yaitu terjadinya *resource contention* dalam satu *process space* arsitektur monolitik ketika terjadi lonjakan permintaan penilaian secara simultan, khususnya pada modul *grading* [14]. Berdasarkan kondisi tersebut, kerangka berpikir ini mengaitkan permasalahan performa *LMS* dengan keterbatasan pendekatan *vertical scaling* yang hanya menambah sumber daya infrastruktur tanpa mengubah desain arsitektur aplikasi. Pendekatan ini dinilai belum mampu mengisolasi beban komputasi berat, khususnya pada modul *grading* yang bersifat *CPU-intensive* dan dijalankan secara sinkron, sehingga berpotensi memengaruhi stabilitas layanan utama *LMS*.

Sebagai solusi, penelitian ini mengusulkan penerapan pola *Hybrid Microservice* dengan pola *Outpost*, yaitu pendekatan *refactoring* parsial yang menambahkan layanan eksternal pada jalur *request* tertentu tanpa mengganti keseluruhan sistem utama. Dalam penelitian ini, Node.js *Grading Service* digunakan sebagai lapisan *queue-based load leveling* yang menerima *request* menuju *endpoint* processattempt, menampungnya dalam antrean, lalu meneruskannya ke *Moodle* melalui *worker pool*. Pendekatan ini tidak memindahkan seluruh proses *grading* dari *Moodle*, tetapi menguji apakah pengaturan aliran *request* pada *endpoint* dapat membantu mengurangi tekanan langsung terhadap sistem monolitik. Pendekatan ini sejalan dengan prinsip arsitektur *Microservice* yang menekankan isolasi beban kerja, *scalability*, dan *fault tolerance* sebagaimana dijelaskan dalam penelitian mengenai perbandingan arsitektur monolitik dan *Microservice* [15] serta kajian sistematis terkait tantangan dan solusi arsitektur *Microservice* [16].

Realisasi solusi dilakukan melalui tahapan metodologis yang meliputi analisis masalah, perumusan kebutuhan artefak, perancangan arsitektur *Hybrid* , implementasi Node.js *Grading Service*, demonstrasi artefak pada skenario ujian daring, serta evaluasi menggunakan *load testing*. Hasil yang diharapkan adalah diperolehnya gambaran empiris mengenai dampak *refactoring* parsial terhadap performa dan ketahanan *Moodle*, sekaligus mengidentifikasi keterbatasan dari

pendekatan tersebut apabila *Moodle* core, database, dan *resource* host masih digunakan bersama. Pendekatan pengujian ini merujuk pada praktik evaluasi performa dan skalabilitas sistem *Microservice* yang direkomendasikan dalam literatur terkini [17].

Hasil akhir yang diharapkan dari kerangka berpikir ini adalah diperolehnya gambaran mengenai kelayakan penerapan *request*-level partial *refactoring* pada *Moodle* melalui Node.js *Grading Service*. Evaluasi dilakukan untuk mengetahui dampak pendekatan tersebut terhadap p95 *response time*, *error rate*, *throughput*, *CPU* utilization, dan *Process attempt duration*. Selain itu, kerangka berpikir ini juga diarahkan untuk mengidentifikasi keterbatasan pendekatan *Hybrid Microservice* dengan pola *Outpost* apabila *Moodle* core, database, session, dan *resource* host masih digunakan bersama dalam satu lingkungan pengujian lokal.

1.7 Sistematika Penulisan

Sistematika penulisan laporan memuat sistematika penulisan laporan tugas akhir dengan memberikan gambaran kandungan setiap bab, urutan penulisan, serta keterkaitan antara satu bab dengan bab lainnya dalam sebuah laporan tugas akhir. Berikut sistematika penulisan laporan tugas akhir.

BAB I PENDAHULUAN

Bab ini terdiri dari latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kerangka pemikiran, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas literatur atau penelitian terdahulu, konsep dan teori, model yang digunakan, dan rumus terkait yang menjadi landasan dalam proses analisis permasalahan dan kebutuhan penelitian.

BAB III METODOLOGI PENELITIAN

Bab ini berisi penjelasan langkah-langkah dan teknik yang diterapkan dalam penelitian, diuraikan secara sistematis dan terstruktur.

BAB IV HASIL DAN PEMBAHASAN

Bab ini memaparkan dua hal utama, yaitu temuan atau hasil penelitian berdasarkan langkah-langkah penelitian yang telah dilakukan, dan pembahasan hasil atau temuan penelitian sebagai jawaban terhadap rumusan masalah penelitian.

BAB V KESIMPULAN DAN SARAN

Bab ini berfokus pada penarikan simpulan dari hasil penelitian yang diperoleh serta menjawab pertanyaan penelitian, serta memberikan saran yang dapat dilakukan penelitian selanjutnya untuk meningkatkan kualitas penelitian.