

PENERAPAN POLA MODULAR MONOLITH UNTUK SKALABILITAS SISTEM PEMBELAJARAN DARING DI SMA NEGERI JATIROGO

Bima Sena Adji, Retno Mumpuni, Afina Lina Nurlaili

Program Studi Informatika, Universitas Pembangunan Nasional “Veteran” Jawa Timur

Jl. Rungkut Madya No. 01 Gunung Anyar, Surabaya 60294

retnomumpuni.if@upnjatim.ac.id

ABSTRAK

Sistem pembelajaran daring pada institusi pendidikan menengah rentan terhadap degradasi arsitektur seiring pertumbuhan fitur, akibat struktur kode monolitik tradisional yang sangat terikat (*tightly coupled*). Arsitektur *microservices* yang lazim diusulkan sebagai solusi justru memunculkan kompleksitas operasional dan latensi jaringan yang tidak proporsional untuk ekosistem berskala sekolah menengah. Penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi penerapan arsitektur Modular *Monolith* pada platform *e-learning* SMA Negeri Jatirogo guna meningkatkan skalabilitas arsitektural dan *maintainability* tanpa kerumitan *deployment*. Sistem dikembangkan menggunakan metode *Prototyping* dengan dekomposisi domain bisnis menjadi empat modul independen (*Auth*, *Admin*, *Course*, *Quiz*) berdasarkan prinsip *Bounded Context* dari *Domain-Driven Design* (DDD). Evaluasi dilakukan melalui *Static Architecture Conformance Checking* (SACC), pengukuran *Cyclomatic Complexity* (CC), dan *User Acceptance Testing* (UAT). Arsitektur berhasil mencapai *zero circular dependencies* dan *zero cross-module violations*, dengan nilai CC tertinggi sebesar 19 (risiko sedang yang dapat ditoleransi) pada fungsi *Hybrid Grading*. UAT mencatatkan tingkat kesesuaian operasional 100%, membuktikan bahwa pola Modular *Monolith* merupakan solusi arsitektural yang tangguh dan terukur untuk sistem pembelajaran daring berskala sekolah menengah.

Kata kunci : *E-learning, Hybrid Grading, Modular Monolith, Software Architecture, Skalabilitas.*

1. PENDAHULUAN

Transformasi digital dalam sektor pendidikan menuntut institusi seperti SMA Negeri Jatirogo untuk beralih dari metode pembelajaran konvensional menuju sistem yang terotomatisasi [1]. Permasalahan utama yang dihadapi saat ini adalah tingginya ketergantungan pada pertemuan tatap muka fisik [2]. Pengelolaan administrasi pembelajaran seperti distribusi materi, pengumpulan tugas, dan rekapitulasi nilai juga masih terfragmentasi secara manual. Hal ini menimbulkan hambatan efisiensi ruang dan waktu, terutama dalam proses evaluasi kuis dan esai yang membebani kinerja tenaga pendidik [3]. Meskipun kebutuhan akan *Learning Management System* (LMS) sangat mendesak, perancangan perangkat lunak berskala sekolah menengah memunculkan tantangan teknis tersendiri, khususnya dalam pemilihan arsitektur sistem.

Penelitian-penelitian sebelumnya menunjukkan bahwa pengembangan sistem informasi sekolah yang mengadopsi arsitektur monolitik tradisional seringkali berakhir menjadi *Big Ball of Mud*, di mana tingkat ketergantungan antar-fungsi (*high coupling*) menyebabkan kode sistem sangat sulit dipelihara seiring bertambahnya fitur [4], [5]. Sebagai alternatif, arsitektur *microservices* sering diajukan untuk mencapai skalabilitas [6]. Namun, penerapan *microservices* pada platform *e-learning* sering kali menghadapi kendala akibat tingginya beban pemrosesan jaringan dan kerumitan manajemen layanan [4]. Lebih lanjut, implementasi *microservices* pada institusi dengan sumber daya infrastruktur terbatas justru memunculkan beban baru berupa latensi

jaringan dan kompleksitas operasional server. Literatur menegaskan bahwa *microservices* sejatinya adalah solusi untuk organisasi berskala raksasa, yang jika dipaksakan pada ekosistem berskala menengah justru dapat menghambat produktivitas dan membuang anggaran secara percuma [7]. Kondisi teknis inilah yang mendorong perlunya evaluasi ulang terhadap arsitektur sistem informasi akademik agar ketergantungan antar-modul dapat diatasi tanpa harus memaksakan penggunaan *microservices* yang berlebihan [8].

Sebagai rencana penyelesaian atas permasalahan tersebut, penelitian ini mengusulkan penerapan desain Modular *Monolith* sebagai solusi arsitektural yang menyeimbangkan kemudahan *deployment* dan kerapian kode [9]. Pendekatan ini mengadopsi prinsip *Bounded Context* dari *Domain-Driven Design* (DDD) untuk mengisolasi logika bisnis ke dalam modul-modul independen (seperti modul *Course* dan *Quiz*), sehingga menghasilkan struktur kode yang lincah (*agile*) dan mudah diuji (*testable*) [10]. Dalam konteks penelitian ini, skalabilitas didefinisikan secara spesifik sebagai skalabilitas arsitektural (*architectural scalability*), yakni kemampuan struktural sistem untuk berkembang baik dari sisi penambahan fitur maupun dari sisi eskalasi infrastruktur tanpa menyebabkan degradasi kode (*architecture erosion*). Definisi ini dibedakan secara eksplisit dari skalabilitas performa (*runtime scalability*) yang mengukur *throughput* dan latensi di bawah beban tinggi. Merujuk pada Newman (2020) [7], fondasi untuk mencapai skalabilitas performa yang sesungguhnya adalah arsitektur yang sudah terisolasi dengan baik sejak awal karena hanya

modul yang sudah memiliki batas domain yang tegas (*Bounded Context*) yang dapat diekstraksi menjadi layanan mandiri (*microservices*) ketika beban pengguna meningkat secara eksponensial di masa depan. Oleh karena itu, pencapaian *zero circular dependencies* dan *zero cross-module violations* yang dibuktikan melalui pengujian SACC pada penelitian ini merupakan prasyarat teknis dari skalabilitas, bukan sekadar indikator maintainability semata. Untuk memastikan arsitektur ini dapat menyelesaikan beban kerja spesifik guru, diimplementasikan pula fitur *Hybrid Grading* (penilaian otomatis dan manual terintegrasi). Pendekatan pengembangan sistem menggunakan metode *Prototyping* secara iteratif guna menjamin kesesuaian antara fungsionalitas aplikasi dengan kebutuhan nyata pengguna sebelum sistem akhir diluncurkan [11], [12]. Hal ini sejalan dengan penelitian terkini yang membuktikan bahwa pemodelan purwarupa secara iteratif sangat efektif dalam meminimalisir kegagalan fungsional [13], serta mampu mempercepat adopsi sistem informasi di lingkungan sekolah [14].

Oleh karena itu, penelitian ini bertujuan untuk merancang, mengimplementasikan, dan mengevaluasi penerapan arsitektur *Modular Monolith* pada platform *e-learning* di SMA Negeri Jatirogo. Kontribusi utama dari penelitian ini adalah menyajikan referensi empiris rekayasa perangkat lunak skala menengah yang dievaluasi secara komprehensif ganda. Validasi dilakukan melalui pengujian arsitektural (*White Box*) menggunakan analisis statis (*Static Architecture Compliance Checking*) dan pengukuran *Cyclomatic Complexity*. Serta divalidasi fungsionalitas bisnisnya melalui *User Acceptance Testing* (UAT), sebagaimana direkomendasikan dalam standar validasi perangkat lunak edukasi untuk mengukur tingkat penerimaan operasional pengguna akhir secara presisi [11]. Melalui pendekatan ini, diharapkan sistem tidak hanya responsif secara fungsional, tetapi juga memiliki tingkat *maintainability* tinggi untuk eskalasi di masa mendatang.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Pengembangan sistem *e-learning* berbasis *website* menggunakan metode *Prototyping* pada tingkat sekolah menengah kejuruan telah dilakukan untuk mengatasi keterbatasan waktu dan pengelolaan tugas manual [11]. Sistem dibangun menggunakan PHP dan MySQL, diuji dengan *Black Box Testing*, dan membuktikan efektivitas *Prototyping* dalam menggali kebutuhan pengguna secara iteratif. Penelitian lain memperkuat temuan tersebut melalui perancangan antarmuka *e-learning* berbasis *Prototyping* pada tingkat SMA, menekankan pentingnya evaluasi desain purwarupa sebelum sistem dibangun utuh agar antarmuka mudah diadopsi pengguna [12]. Kedua penelitian tersebut menjadi acuan metode pengembangan pada penelitian ini, namun keduanya

tidak membahas kualitas arsitektur kode yang dihasilkan.

Dari sisi arsitektur, perbandingan performa arsitektur monolitik dan *microservices* secara empiris mengidentifikasi bahwa monolitik tradisional rentan terhadap kegagalan total saat satu modul bermasalah [5]. Pengembangan LMS berbasis *microservices* menggunakan API Gateway juga telah dilakukan, namun menemukan kompleksitas tinggi dalam komunikasi data antar-layanan [4]. Sebagai alternatif pragmatis, arsitektur *Modular Monolith* yang menerapkan prinsip *Bounded Context* dari *Domain-Driven Design* (DDD) terbukti menghasilkan isolasi modul internal yang ketat tanpa kerumitan operasional *microservices* [6], dan divalidasi lebih lanjut bahwa arsitektur terisolasi merupakan prasyarat skalabilitas jangka panjang [7].

Pada aspek pengujian struktural, penelitian-penelitian terdahulu umumnya hanya menerapkan pengujian fungsional (*Black Box*) tanpa memvalidasi kualitas arsitektur kode secara internal. Evaluasi arsitektur *microservices* menggunakan *Microservices Scorecard* telah dilakukan [8], namun tidak pada konteks *Modular Monolith*. Celah inilah yang diisi oleh penelitian ini: menggabungkan *Static Architecture Conformance Checking* (SACC) dan pengukuran *Cyclomatic Complexity* sebagai kerangka evaluasi arsitektur *Modular Monolith* yang terukur pada sistem *e-learning* skala sekolah pendekatan yang belum pernah dilakukan pada penelitian-penelitian sebelumnya.

2.2. Sistem E-Learning

Sistem Informasi *E-Learning* merupakan metode pembelajaran yang memanfaatkan teknologi informasi dan komunikasi untuk mengelola, mengubah, dan mendistribusikan materi antar perangkat [2]. Karakteristik utamanya adalah kemampuan mendobrak batasan ruang dan waktu sehingga peserta didik dapat mengakses materi kapan saja dan di mana saja [2]. Dalam konteks sekolah menengah, *E-Learning* bertujuan menanggulangi keterbatasan jam tatap muka dan mengotomatisasi pengelolaan data akademik yang sebelumnya dilakukan secara manual, seperti distribusi materi, pengumpulan tugas, dan rekapitulasi nilai [3].

2.3. Arsitektur Perangkat Lunak

Arsitektur perangkat lunak mendefinisikan struktur dasar sistem yang mempengaruhi skalabilitas, *maintainability*, dan kinerja jangka panjang [9]. Arsitektur monolitik tradisional menyatukan seluruh fungsionalitas dalam satu *codebase* tunggal mudah di awal pengembangan namun rentan terhadap kegagalan total dan penskalaan yang tidak efisien seiring bertambahnya kompleksitas [5]. Sebagai alternatif, arsitektur *microservices* memecah aplikasi menjadi layanan-layanan kecil yang independen dan berkomunikasi via HTTP/REST API [4]. Meskipun menawarkan fleksibilitas tinggi, *microservices*

membawa beban operasional berupa orkestrasi kontainer dan latensi jaringan yang seringkali tidak proporsional untuk sistem berskala menengah [7].

2.4. Arsitektur Modular Monolith

Modular *Monolith* adalah gaya arsitektur di mana sistem tetap di-deploy sebagai satu unit tunggal, namun secara internal kode diorganisir secara ketat ke dalam modul-modul independen yang terisolasi berdasarkan domain bisnisnya [6]. Komunikasi antar-modul dilakukan melalui antarmuka publik internal (*contracts*) yang didefinisikan secara eksplisit, bukan melalui jaringan seperti pada *microservices* [6], [7]. Pendekatan ini menerapkan prinsip *Bounded Context* dari *Domain-Driven Design* (DDD) untuk memetakan batasan tanggung jawab setiap modul secara tegas [15], menghasilkan struktur kode dengan kohesi internal tinggi (*High Cohesion*) dan ketergantungan antar-modul yang rendah (*Low Coupling*).

2.5. Pengujian Perangkat Lunak

Pengujian struktural (*White Box*) pada penelitian ini menggunakan dua pendekatan. Pertama, *Static Architecture Conformance Checking* (SACC) memverifikasi kepatuhan kode sumber terhadap aturan arsitektur yang telah ditetapkan dengan mendeteksi pelanggaran isolasi modul dan dependensi melingkar sebelum sistem dioperasikan [8]. Kedua, *Cyclomatic Complexity* (CC) mengukur kompleksitas logika internal fungsi secara kuantitatif menggunakan rumus $V(G) = E - N + 2$; nilai CC yang tinggi mengindikasikan probabilitas kesalahan yang lebih besar dan keterpeliharaan yang lebih rendah [9]. Validasi fungsional dilengkapi dengan *User Acceptance Testing* (UAT) untuk mengukur tingkat kesesuaian operasional sistem dengan kebutuhan pengguna nyata di lapangan [11].

3. METODE PENELITIAN

Penelitian ini dilaksanakan melalui pendekatan rekayasa perangkat lunak yang sistematis, dimulai dari identifikasi kebutuhan di SMA Negeri Jatirogo hingga validasi kualitas kode. Tahapan penelitian dirancang untuk memastikan bahwa arsitektur yang dibangun memenuhi kaidah modularitas dan kebutuhan fungsional pengguna.

3.1. Metode Prototyping dalam Pemetaan Domain

Metodologi penelitian ini menggunakan pendekatan *Prototyping* sebagai landasan untuk menggali kebutuhan sistem secara iteratif. Proses diawali dengan observasi dan wawancara mendalam kepada stakeholder di SMA Negeri Jatirogo guna memetakan proses bisnis akademik. Hasil identifikasi kemudian diklasifikasikan menggunakan metode MoSCoW untuk menetapkan prioritas fitur. Berbeda dengan pendekatan tradisional, hasil analisis kebutuhan ini tidak langsung diimplementasikan ke dalam basis kode tunggal, melainkan digunakan

sebagai dasar untuk melakukan dekomposisi sistem berdasarkan fungsionalitas dan kepemilikan data.

3.2. Dekomposisi Sistem dan Pemetaan Bounded Context

Untuk menghindari terbentuknya *Spaghetti Code* dan *Tight Coupling*, sistem LMS ini dipecah menjadi wilayah-wilayah terisolasi secara logis yang disebut *Bounded Context*. Proses dekomposisi ini mengadopsi prinsip *Domain-Driven Design* (DDD) dengan mengelompokkan fitur ke dalam empat modul utama: *Auth*, *Admin*, *Course*, dan *Quiz*. Setiap modul memiliki otonomi penuh (*High Cohesion*) dan batasan tanggung jawab data yang ketat sebagaimana dirangkum dalam Tabel 1.

Tabel 1. Dekomposisi Sistem dan Batasan Kepemilikan Data

Nama Modul	Tanggung Jawab Utama	Batasan Kepemilikan Data
Modul <i>Auth</i>	Otentikasi identitas dan manajemen sesi.	Kredensial (<i>password hash</i>) dan <i>Roles</i> .
Modul <i>Admin</i>	Manajemen data induk sekolah (<i>Master Data</i>).	Profil <i>User</i> , entitas Kelas, dan Penjadwalan.
Modul <i>Course</i>	Distribusi bahan ajar dan penugasan.	Materi (PDF/Video), Silabus, dan Tugas.
Modul <i>Quiz</i>	Evaluasi akademik dan kalkulasi nilai.	Bank Soal, Jawaban Siswa, dan Nilai Rapor.

Penerapan integritas *Bounded Context* dijamin melalui hukum arsitektural yang melarang operasi manipulasi data secara langsung (SQL JOIN) antar-tabel milik modul yang berbeda. Sebagai solusi, komunikasi lintas modul diatur melalui mekanisme *Service Contract* (API Internal) menggunakan *Soft Reference*. Pendekatan ini memastikan bahwa setiap modul tetap independen secara fisik di dalam basis kode monolitik, namun memiliki karakteristik modularitas layaknya layanan mikro.

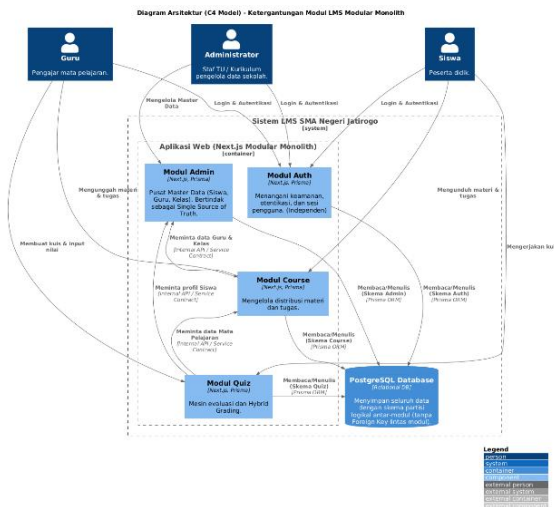
3.3. Perancangan Arsitektur dan Kontrak Komunikasi

Pemetaan wilayah domain (*Bounded Context*) yang telah dideklarasikan pada tahap dekomposisi perlu diterjemahkan ke dalam cetak biru struktural agar dapat diimplementasikan secara presisi pada tingkat kode. Oleh karena itu, perancangan pada tahap ini difokuskan pada dua aspek utama: visualisasi hierarki sistem menggunakan standar C4 Model dan penetapan aturan komunikasi lintas modul melalui *Service Contract* guna menjaga integritas data.

3.4. Cetak Biru Arsitektur (C4 Model)

Perancangan arsitektur sistem divisualisasikan menggunakan standar C4 Model pada tingkat *Container* dan *Component*. Pendekatan ini dipilih untuk memetakan tumpukan teknologi (Next.js dan Prisma ORM) serta batas sistem (*System Boundary*).

Sebagaimana ditunjukkan pada Gambar 1, seluruh modul berada dalam satu wadah aplikasi web terpadu, namun tetap memiliki garis ketergantungan yang terdefinisi dengan presisi.



Gambar 1. Cetak Biru Arsitektur (C4 Model)

Interaksi antar-modul dirancang dengan mematuhi *Dependency Rule* yang bersifat searah (*Directed Acyclic Graph*). Modul dikategorikan menjadi dua lapisan:

- Modul Hulu (*Upstream*): Bersifat independen, terdiri dari modul *Auth* (keamanan) dan modul *Admin* (*master data*). Modul *Admin* bertindak sebagai *Single Source of Truth* bagi modul lainnya.
- Modul Hilir (*Downstream*): Modul transaksional yang bergantung pada data dari lapisan hulu, yaitu modul *Course* (distribusi materi) dan modul *Quiz* (evaluasi). Modul *Quiz* memiliki tingkat ketergantungan tertinggi karena membutuhkan referensi data dari modul *Course* dan *Admin* secara bersamaan.

Penerapan *Dependency Rule* yang hierarkis dan searah ini merupakan mekanisme pertahanan utama untuk menjamin stabilitas sistem. Dengan melarang adanya ketergantungan melingkar (*circular dependency*), setiap modifikasi kode pada modul hilir dipastikan tidak akan mengganggu integritas fungsi pada modul hulu.

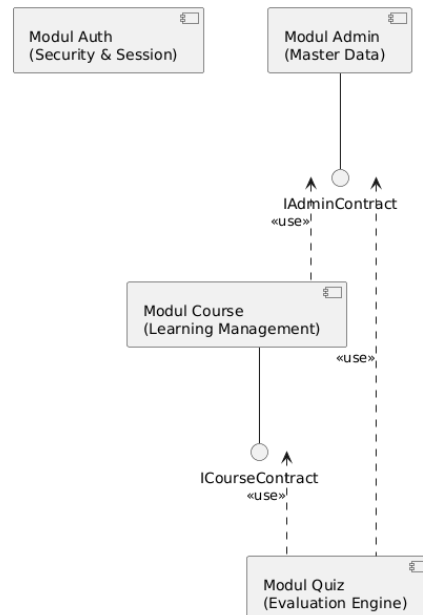
3.5. Mekanisme Komunikasi Antar-Modul (*Service Contract*)

Untuk menjamin integritas *Bounded Context*, sistem menerapkan hukum arsitektural yang melarang keras penggunaan operasi basis data secara langsung (seperti SQL JOIN) lintas tabel milik modul yang berbeda. Sebagai gantinya, komunikasi dilakukan melalui mekanisme *Service Contract* (API Internal) berbasis antarmuka (*Interface*).

Berdasarkan Gambar 2, setiap modul berinteraksi melalui *Provided Interface* dan *Required Interface*.

Modul penyedia (seperti *Admin*) mengekspos antarmuka publik bernama *IAdminContract* yang menjadi satu-satunya jalur resmi bagi modul konsumen (*Course* dan *Quiz*) untuk memvalidasi data.

UML Component Diagram (C4 Model) - Interaksi Lintas Modul LMS



Gambar 2. UML Component Diagram

Dalam menjaga prinsip *low coupling*, sistem menggunakan *Data Transfer Object* (DTO) sebagai kurir lintas modul. Saat data dikirimkan, ia dibungkus ke dalam objek DTO yang hanya mengekspos atribut spesifik, sehingga mencegah terjadinya paparan skema basis data mentah secara langsung antar-modul.

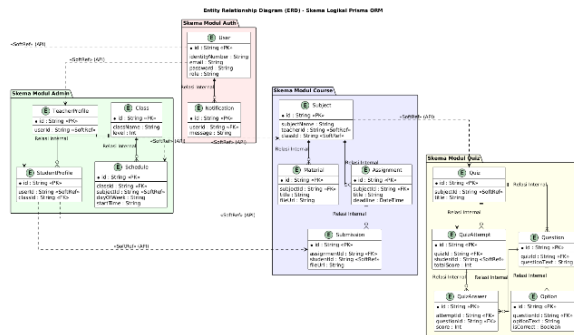
3.4. Perancangan Basis Data Terpartisi

Modularitas pada sistem ini tidak hanya diterapkan pada level aplikasi, tetapi diperluas hingga lapisan persistensi data melalui pola *Logically Partitioned Database*. Pendekatan ini bertujuan untuk menegaskan integritas *Bounded Context* dengan melarang adanya relasi *Foreign Key* fisik yang menyeberang antar-modul, guna menghindari *tight coupling* pada tingkat basis data.

Sebagai solusi untuk mempertahankan keterhubungan data tanpa menciptakan ketergantungan fisik, penelitian ini mengimplementasikan teknik *Soft Reference*. Dengan teknik ini, sebuah tabel hanya menyimpan referensi pengenalan unik (ID) dari entitas di modul lain sebagai atribut identitas biasa, bukan sebagai kunci tamu (*Foreign Key*) yang terikat secara fisik. Rancangan keterkaitan data ini divisualisasikan melalui *Entity Relationship Diagram* (ERD) pada gambar 3.

Berdasarkan Gambar 3, relasi fisik (garis solid) hanya diizinkan terjadi di dalam wilayah modul yang sama untuk menjaga kohesi internal. Relasi yang menyeberang antar-modul disimbolkan dengan garis putus-putus yang merepresentasikan *Soft Reference*.

Pemetaan kepemilikan data dibagi ke dalam empat skema utama:



Gambar 3. Entity Relationship Diagram (ERD)

- Skema Modul Auth: Mengelola entitas fundamental otentikasi seperti kredensial pengguna dan notifikasi.
- Skema Modul Admin: Bertindak sebagai pusat *Master Data* struktural sekolah yang mengelola profil personal, entitas kelas, dan penjadwalan akademik.
- Skema Modul Course: Mengelola kegiatan belajar-mengajar yang mencakup entitas bahan ajar, penugasan, dan pengumpulan tugas.
- Skema Modul Quiz: Mengelola seluruh entitas evaluasi dengan kompleksitas relasi internal yang tinggi, meliputi bank soal, rekam jejak jawaban, dan kalkulasi nilai.

Penerapan isolasi basis data melalui Prisma ORM ini memberikan keuntungan arsitektural berupa pencegahan kegagalan berantai (*cascading failure*). Rancangan ini memastikan sistem memiliki kesiapan masa depan (*future-proof*) yang maksimal.

3.5. Skenario Pengujian Sistem

Setelah tahap konstruksi purwarupa selesai, sistem dievaluasi melalui dua dimensi pengujian utama, yaitu validasi fungsional eksternal (*Black Box Testing*) dan validasi struktural internal (*White Box Testing*).

3.6. Pengujian Fungsionalitas (*Black Box*)

Pengujian fungsional dilaksanakan menggunakan metode *Equivalence Partitioning* (EP) untuk memvalidasi alur kerja standar, dan *Boundary Value Analysis* (BVA) untuk menguji ketahanan sistem terhadap masukan pada nilai batas ekstrem. Hasil eksekusi pengujian fungsional ini tidak dibahas secara mendetail dalam makalah ini, karena fokus evaluasi penelitian diarahkan pada validasi kualitas arsitektur melalui pengujian *White Box* dan *User Acceptance Testing* (UAT) sebagai representasi tertinggi dari validasi fungsional berbasis pengguna nyata.

3.7. Pengujian Arsitektur dan Kompleksitas (*White Box*)

Pengujian *White Box* bertujuan untuk memastikan bahwa implementasi kode sumber (*as-implemented*) mematuhi desain arsitektur (*as-designed*) guna mencegah terjadinya degradasi perangkat lunak. Pengujian ini dibagi menjadi dua metrik utama:

- Pengujian Struktur Arsitektur: Menerapkan metode *Static Architecture Conformance Checking* (SACC) menggunakan alat analisis statis *Dependency Cruiser*. Parameter keberhasilannya adalah tidak adanya pelanggaran isolasi antar-modul (*cross-module violation*) dan tercapainya 0 (nol) ketergantungan melingkar (*circular dependency*).
- Pengujian Kompleksitas Logika: Menggunakan pendekatan *Basis Path Analysis* untuk mengukur metrik *Cyclomatic Complexity* (CC) pada fungsi-fungsi bisnis yang kritis. Penghitungan nilai CC dilakukan berdasarkan jumlah jalur independen sesuai dengan Persamaan.

$$V(G) = E - N + 2 (1)$$

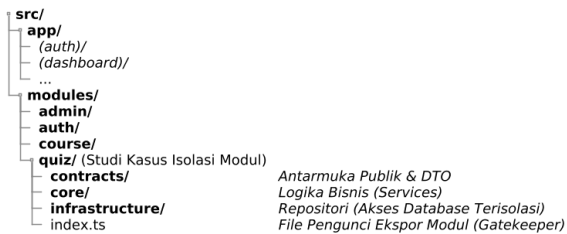
Evaluasi metrik ini diklasifikasikan ke dalam tiga kategori: nilai 1-10 mengindikasikan risiko rendah (*maintainability tinggi*), nilai 11-20 sebagai risiko sedang yang masih dapat ditoleransi untuk algoritma kompleks (seperti *Hybrid Grading*), dan nilai di atas 20 sebagai risiko tinggi yang membutuhkan *refactoring*. Selain sebagai indikator *maintainability*, kedua metrik pengujian *White Box* ini secara bersama-sama berfungsi sebagai proksi terukur dari skalabilitas arsitektural. Ketidadaan *circular dependency* yang dibuktikan melalui SACC memastikan bahwa setiap modul dapat dimodifikasi, diganti, atau diekstraksi secara independen tanpa menimbulkan *cascading failure* pada modul lainnya kondisi yang merupakan prasyarat wajib sebelum sebuah sistem monolitik dapat dimigrasi sebagian atau seluruhnya ke arsitektur terdistribusi. Sementara itu, nilai *Cyclomatic Complexity* yang terkendali memastikan bahwa setiap fungsi bisnis kritis tetap dapat dipahami dan diuji secara mandiri bahkan setelah sistem bertambah kompleks, sehingga biaya pengembangan tidak meningkat secara tidak proporsional seiring pertumbuhan fitur.

4. HASIL DAN PEMBAHASAN

Bab ini menguraikan hasil transformasi dari desain konseptual arsitektur Modular *Monolith* menjadi basis kode fisik yang fungsional. Pembahasan difokuskan pada implementasi hierarki direktori sistem, evaluasi kepatuhan struktural menggunakan analisis statis, pengukuran kompleksitas logika operasional, serta validasi akhir fungsionalitas melalui pengujian penerimaan pengguna di lingkungan SMA Negeri Jatirogo.

4.1. Implementasi Struktur Modul Terisolasi

Rancangan arsitektur yang telah dimodelkan melalui C4 Model dan *Package Diagram* direpresentasikan ke dalam struktur direktori fisik menggunakan pendekatan *Screaming Architecture*. Pendekatan ini memastikan bahwa basis kode secara eksplisit memperlihatkan batas wilayah domain bisnis (*Bounded Context*), bukan sekadar tumpukan teknologi kerangka kerja. Untuk merealisasikannya, sistem membelah basis kode menjadi dua lapisan utama, sebagaimana diilustrasikan pada Gambar 4.



Gambar 4. Isolasi Modul Terisolasi

Berdasarkan Gambar 4, penegakan arsitektur diimplementasikan melalui prinsip *Separation of Concerns* dan *Information Hiding* dengan pembagian sebagai berikut:

- Direktori *app/* (Lapisan Presentasi): Berfungsi murni menangani antarmuka pengguna (UI) dan perutean (*routing*). Komponen di lapisan ini dilarang keras melakukan operasi basis data secara langsung dan hanya diizinkan memanggil layanan publik dari lapisan domain.
- Direktori *modules/* (Lapisan Domain Terisolasi): Merupakan pusat operasional Modular *Monolith* yang menyimpan seluruh logika bisnis dan infrastruktur.

Mengambil Modul *Quiz* sebagai representasi, anatomi di dalam direktori modul dirancang dengan hierarki akses di dalam setiap direktori modul bersifat restriktif: lapisan *infrastructure/* menyimpan sintaks Prisma ORM yang tertutup penuh, lapisan *core/* menampung logika bisnis, lapisan *contracts/* menjadi satu-satunya area publik yang mengekspos *Interface* dan *DTO*, dan *index.ts* bertindak sebagai *Gatekeeper* yang memblokir impor ilegal secara otomatis hingga menghasilkan *compile-time error*.

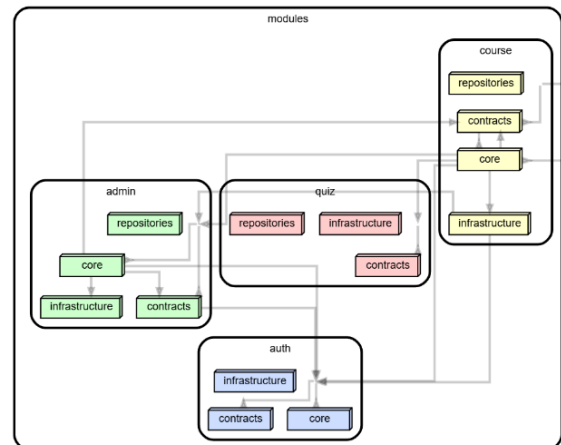
Melalui manajemen ekspor (*export management*) ini, kompiler TypeScript dikonfigurasi secara ketat agar lapisan UI maupun modul eksternal hanya dapat melakukan impor melalui fail *index.ts*. Mekanisme ini memberikan bukti konkret bahwa prinsip isolasi arsitektur telah tertanam kuat.

4.2. Hasil Evaluasi Kepatuhan Arsitektur (SACC)

Pengujian struktural dimensi pertama dieksekusi menggunakan alat bantu analisis statis *Dependency Cruiser* untuk memverifikasi kepatuhan kode sumber (*source code*) terhadap aturan arsitektur Modular *Monolith*. Evaluasi ini bertujuan untuk mendeteksi

potensi degradasi arsitektur (*architecture erosion*) sebelum sistem diimplementasikan di lingkungan produksi.

Berdasarkan hasil pemindaian sistem, grafik dependensi memberikan bukti empiris bahwa arsitektur yang diusulkan berhasil ditegakkan secara sempurna. Merujuk pada Gambar 5, temuan krusial dari evaluasi ini dapat disintesis ke dalam tiga parameter keberhasilan utama:



Gambar 5. Visualisasi Grafik Dependensi Tingkat Modul

- Isolasi Antar-Modul yang Konsisten: Keempat modul utama (*Auth*, *Admin*, *Course*, *Quiz*) terenkapsulasi secara tegas. Seluruh panah dependensi lintas domain terbukti hanya mengarah dan berhenti pada lapisan antarmuka publik (*contracts*). Tidak ditemukan adanya pemanggilan ilegal (*bypass*) menuju lapisan internal (*core* atau *infrastructure*) milik modul lain, sehingga sistem mencatatkan 0 (nol) pelanggaran lintas modul (*cross-module violations*).
- Aliran Dependensi Searah (*Unidirectional*): Arsitektur internal pada masing-masing modul secara konsisten mematuhi prinsip hierarki. Alur eksekusi selalu bergerak dari lapisan publik menuju lapisan logika bisnis, dan berakhir pada implementasi teknis basis data. Tidak ditemukan satupun dependensi yang mengarah terbalik (*upward dependency*).
- Pencapaian *zero circular dependencies*: Diverifikasi secara empiris melalui *Dependency Cruiser* yang memindai seluruh *import/export* statements antar-berkas TypeScript dan mencocokkannya dengan *ruleset* batas modul. Tidak ditemukan satu pun siklus dependensi bolak-balik, yang mengonfirmasi tingginya kohesi sistem dan menjamin stabilitas fungsionalitas saat modifikasi kode di masa depan.

4.3. Analisis Kompleksitas Logika (White Box)

Dimensi kedua menguji kualitas internal pada tingkat fungsi (metode) menggunakan metrik *Cyclomatic Complexity* (CC). Pengujian tidak diterapkan pada fungsi operasi CRUD (*Create, Read, Update, Delete*) dasar yang linier, melainkan dikhususkan pada fungsi-fungsi logika bisnis kritis yang bertindak sebagai "otak" penggerak utama pada masing-masing modul. Hasil perhitungan jalur independen disajikan pada Tabel 2.

Tabel 2. Rekapitulasi Hasil Pengukuran *Cyclomatic Complexity*

Modul	Nama Fungsi Kritis	Nilai CC	Kategori Risiko
Modul Auth	<i>authenticateUser()</i>	7	Risiko Rendah
Modul Admin	<i>getAllTeachers()</i>	11	Risiko Sedang
Modul Course	<i>getCourseDetailData()</i>	6	Risiko Rendah
Modul Quiz	<i>submitQuizAttempt()</i>	19	Risiko Sedang

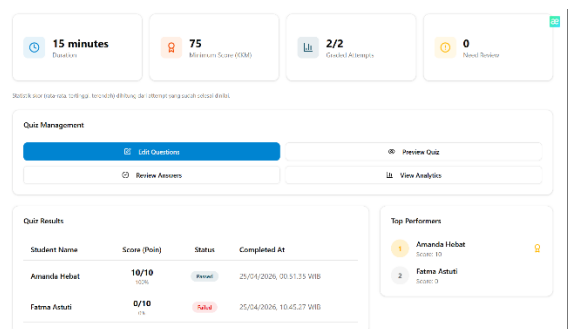
Berdasarkan Tabel 2, dapat dianalisis bahwa seluruh fungsi operasional sistem beroperasi dalam ambang batas toleransi kompleksitas, tanpa ada satupun yang melampaui kategori risiko tinggi (CC > 20). Pencapaian nilai metrik 6 pada modul *Course* merupakan hasil langsung dari optimasi kode (*refactoring*), di mana iterasi tumpukan data penugasan berhasil diekstraksi ke *helper functions* eksternal. Hal ini membuktikan implementasi prinsip *Clean Code* yang efektif.

Sementara itu, fungsi antarmuka publik pada modul *Quiz* mencatatkan nilai tertinggi sebesar 19 (Risiko Sedang). Nilai tersebut merupakan metrik yang sangat rasional dan dapat dipertanggungjawabkan, mengingat fungsi tersebut memikul beban komputasi terpusat. Secara asinkronus, fungsi ini harus memvalidasi keamanan ujian, mengkalkulasi selisih batas waktu (*grace time*), serta mengeksekusi otomatisasi penilaian ganda (*hybrid grading*) dalam satu urutan transaksi basis data relasional. Secara keseluruhan, metrik ini memvalidasi bahwa arsitektur logika sistem stabil, terukur, dan tidak memerlukan restrukturisasi mendesak.

4.4. Validasi Fungsional dan Penerimaan Pengguna

Tahap akhir dari evaluasi sistem bertujuan untuk memverifikasi bahwa ketangguhan arsitektur internal (Modular *Monolith*) berbanding lurus dengan keandalan fungsionalitas di sisi pengguna akhir. Sesuai dengan metodologi *Prototyping*, umpan balik dari pengguna tidak sekadar digunakan untuk merancang antarmuka, melainkan secara aktif mendikte pembentukan arsitektur logika bisnis guna memastikan sistem 100% relevan dengan masalah operasional sekolah.

Sebagai representasi tertinggi dari kompleksitas fungsional yang dihasilkan dari siklus iterasi tersebut, Modul *Quiz* berhasil diimplementasikan untuk mengakomodasi fungsionalitas penilaian ganda (*Hybrid Grading*).



Gambar 6. Antarmuka Operasional Modul Quiz pada Sisi Guru

Merujuk pada Gambar 6, antarmuka memfasilitasi guru untuk melakukan koreksi manual pada soal esai, yang nilainya kemudian diakumulasi secara otomatis oleh peladen dengan skor pilihan ganda. Fungsionalitas ini membuktikan bahwa isolasi domain yang ketat (*Bounded Context*) tidak menghalangi sistem untuk mengeksekusi komputasi relasional yang kompleks secara akurat dan responsif.

Sebagai tahap puncak, purwarupa sistem yang telah mencapai fungsi penuh divalidasi melalui *User Acceptance Testing* (UAT). Pengujian ini dirancang dengan melibatkan representasi langsung dari pemangku kepentingan utama di lingkungan SMA Negeri Jatirogo. Rekapitulasi hasil evaluasi terhadap 12 skenario operasional krusial disajikan pada Tabel 3.

Tabel 3. Rekapitulasi Hasil *User Acceptance Testing* (UAT)

Aktor / Peran	Fokus Skenario Pengujian Utama	Total Skenario	Tingkat
Administrator	Autentikasi, Impor CSV Massal, Plotting Kelas, Manajemen Jadwal.	4	100%
Guru	Unggah Materi, Distribusi Tugas, Pembuatan Soal, <i>Hybrid Grading</i> .	5	100%
Siswa	Ganti Sandi Wajib, Akses Materi, Eksekusi Kuis.	3	100%

Berdasarkan Tabel 3, seluruh skenario operasional (seperti migrasi data massal via CSV, distribusi tugas, dan eksekusi kuis berbatas waktu) mendapatkan persetujuan mutlak. Tingkat kesesuaian operasional mencapai 100%, yang secara formal

mengonfirmasi bahwa sistem berbasis Modular *Monolith* ini tidak hanya stabil secara struktural, tetapi juga telah layak pakai untuk menopang beban akademik sekolah secara komprehensif.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dan evaluasi, dapat disimpulkan bahwa penerapan arsitektur Modular *Monolith* yang diintegrasikan dengan metode *Prototyping* berhasil diimplementasikan dengan tingkat keandalan yang tinggi pada sistem *e-learning* SMA Negeri Jatirogo. Evaluasi menggunakan *Static Architecture Conformance Checking* (SACC) membuktikan bahwa pemetaan *Bounded Context* yang dijaga oleh mekanisme *Gatekeeper* efektif mencegah degradasi arsitektur, ditandai dengan pencapaian 0 (nol) pelanggaran lintas modul dan ketiadaan *circular dependencies*. Hasil ini secara langsung membuktikan skalabilitas arsitektural sistem: dengan batas domain yang terisolasi sempurna, setiap modul terutama Modul *Quiz* yang menanggung beban komputasi tertinggi telah siap untuk diekstraksi menjadi layanan *microservices* mandiri kapan pun dibutuhkan tanpa merombak logika bisnis yang ada. Selain itu, analisis kompleksitas logika menunjukkan bahwa seluruh fungsi beroperasi pada ambang batas optimal (nilai CC tertinggi sebesar 19 pada fitur *Hybrid Grading*), sementara validasi akhir melalui *User Acceptance Testing* (UAT) mencatatkan tingkat kesesuaian 100%, yang menegaskan bahwa ketangguhan fondasi sistem selaras dengan pemenuhan kebutuhan fungsional pengguna.

Berdasarkan pencapaian tersebut, terdapat beberapa saran strategis untuk pengembangan penelitian selanjutnya. Pertama, penelitian mendatang dapat mengkaji skenario ekstraksi fisik (*microservices extraction*) secara khusus pada Modul *Quiz* guna mengantisipasi lonjakan pengguna. Karena modul ini menanggung beban komputasi terberat melalui mekanisme penilaian ganda, pemisahannya menjadi layanan mandiri akan sangat menguntungkan dari sisi skalabilitas peladen. Kedua, guna menjaga stabilitas kode pada pengembangan jangka panjang, penelitian berikutnya perlu mengimplementasikan pengujian terotomatisasi (*Automated Testing*) yang terintegrasi secara langsung ke dalam alur *Continuous Integration/Continuous Deployment* (CI/CD).

Ketiga, pengembangan selanjutnya dapat mempertimbangkan eksplorasi integrasi kecerdasan buatan (AI), khususnya pemodelan pemrosesan bahasa alami (NLP), di dalam Modul *Quiz*. Penerapan teknologi ini ditujukan untuk mengotomatisasi proses koreksi pada butir soal bertipe esai, yang tidak hanya akan mereduksi beban administratif tenaga pendidik, tetapi juga menyempurnakan fitur evaluasi akademik menjadi sebuah ekosistem pembelajaran yang berkinerja tinggi dan sepenuhnya cerdas.

DAFTAR PUSTAKA

- [1] D. Suriyadi, A. Setiawan, and S. Wahyudi, "Peningkatan Hasil Belajar Siswa Melalui Penerapan E-Learning Pada Mata Pelajaran TIK Kelas X di SMA Negeri 1 Rambah," *Jurnal MediaTIK: Jurnal Media Pendidikan Teknik Informatika dan Komputer* Vol.6, vol. 6, no. 2, pp. 102–108, 2023, doi: 10.26858/jmtik.v6i2.46518.
- [2] I. Hakim, S. Assegaff, M. S. Informasi, S. Dinamika Bangsa, J. J. Jendral, and S. Thehok, "Analisis Dan Perancangan Sistem E-Learning Berbasis Web Pada Sma Negeri 4 Kota Jambi," *Jurnal Manajemen Sistem Informasi*, vol. 3, no. 4, pp. 1255–1261, 2018.
- [3] P. Garamesa, R. Y. Kalaway, and D. A. Sitaniapessy, "Rancang Bangun Sistem E-Learning Berbasis Web Di SMA Negeri 1 Waingapu," *Indonesian Journal of Informatic Research and Software Engineering (IJIRSE)*, vol. 2, no. 2, pp. 120–127, 2022, doi: 10.57152/ijirse.v2i2.424.
- [4] C. A. Dinova and I. C. Utomo, "Pengembangan Arsitektur Microservice pada Learning Management System E-learning Menggunakan Metode Web Service," *Jurnal Ilmu Komputer dan Informatika (JIKI)*, vol. 3, no. 2, pp. 125–141, 2023, doi: 10.54082/jiki.102.
- [5] L. Affandi, M. E. Apriyani, and A. M. Putra, "Analisis Response Metrics Terhadap Arsitektur Monolithic dan Microservices dalam Implementasi Aplikasi Kompen," *Jurnal Teknologi Informasi*, vol. 11, no. 2, pp. 48–53, Oct. 2020, doi: 10.36382/jti-tki.v11i2.495.
- [6] K. K. Pappula and S. Anasuri, "Modular Monoliths in Practice: A Middle Ground for Growing Product Teams," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 3, no. 4, pp. 53–63, 2022, doi: <https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106>.
- [7] S. Newman, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*. Sebastopol: O'Reilly Media, 2020. [Online]. Available: https://learning.oreilly.com/videos/monolith-to-microservices/1492047848/1492047848-1492047848_002_MP3/
- [8] K. R. Putra et al., "Perancangan dan Evaluasi Arsitektur Microservices Menggunakan Microservices Migration Pattern dan Microservices Scorecard," *Jurnal Aplikasi Teknologi Informasi dan Manajemen (JATIM)*, vol. 6, no. 2, pp. 70–83, 2025.
- [9] M. Richards and N. Ford, *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol: O'Reilly Media, 2020.
- [10] E. Evan, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston: Addison-Wesley, 2003.
- [11] K. Sofyan and S. R. Putri, "Rancang Bangun Sistem Pembelajaran E-Learning Berbasis Website Menggunakan Metode Prototype (Studi Kasus : SMK Sirajul Falah Parung)," *Jurnal TICOM: Technology of Information and Communication*, vol. 14, no. 1, pp. 37–43, 2025, doi: <https://doi.org/10.70309/ticom.v14i1.167>.

- [12] Frederica Rosabel Ramli, Fikri Hakim, and Ria Anggelina Hutabarat, "Perancangan Web Design Aplikasi E-Learning dengan Metode Prototype pada Tingkat SMA," *Majalah Ilmiah UPI YPTK*, vol. 28, no. 1, pp. 13–18, 2021, doi: 10.35134/jmi.v28i1.62.
- [13] H. I. T. Simamora, "PERANCANGAN SISTEM INFORMASI PENJUALAN CV MITRA TANI MENGGUNAKAN METODE PROTOTYPE," *JURTEKSI (Jurnal Teknologi dan Sistem Informasi)*, vol. 6, no. 2, pp. 173–178, 2020.
- [14] D. Anggoro and A. Hidayat, "Rancang Bangun Sistem Informasi Perpustakaan Sekolah Berbasis Web Guna Meningkatkan Efektivitas Layanan Pustakawan," *Edumatic : Jurnal Pendidikan Informatika*, vol. 4, no. 1, pp. 151–160, 2020, doi: 10.29408/edumatic.v4i1.2130.
- [15] V. Vernon, *Implementing Domain-Driven Design*, vol. 32, no. 3. Boston: Addison-Wesley Professional, 2013