Journix: Journal of Informatics and Computing
Vol. 1 No. 2, 2025 : 63-76
ISSN : 3090-6784 | DOI : 10.63866/journix.v1i2.11

Development of Management Information System Based on MVC Architecture to Improve Business Process Efficiency

Miftahul Jannah [1], Dahlan [1]*, Chairul Akbar [1]

[1] Program Studi Ilmu Komputer, Universitas Muhammadiyah Bima, Indonesia

Email: dahlanumb@gmail.com
(* : corresponding author)

ABSTRACT – With the increasing need for flexible, scalable, and maintainable information systems, the Model-View-Controller (MVC) architecture has become a popular approach in software development. This research aims to design and implement an MVC-based management information system to improve data management and business process efficiency. The study adopts the Agile software development method with the Scrum framework, enabling iterative progress and rapid adaptation to user needs. The implementation employs the Laravel framework to ensure modular separation between business logic, user interface, and control flow. The evaluation is carried out using black-box testing and performance testing with Apache JMeter. The results show that the MVC-based system reduces processing time by 30% compared to a monolithic system and enhances system scalability and maintainability. This study concludes that MVC architecture provides significant improvements in system efficiency, modularity, and sustainability. Future work may focus on integrating microservices and cloud computing to further enhance scalability and performance.

KEYWORDS: MVC, Agile, Laravel, Information System, Performance Testing

Pengembangan Sistem Informasi Manajemen Berbasis MVC untuk Meningkatkan Efisiensi Proses Bisnis

ABSTRAK – Seiring meningkatnya kebutuhan organisasi terhadap sistem informasi yang fleksibel, skalabel, dan mudah dipelihara, arsitektur Model-View-Controller (MVC) menjadi pendekatan populer dalam pengembangan perangkat lunak. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem informasi manajemen berbasis MVC guna meningkatkan efisiensi pengelolaan data dan proses bisnis. Metode penelitian menggunakan pendekatan pengembangan perangkat lunak Agile dengan kerangka kerja Scrum, yang memungkinkan iterasi cepat dan adaptasi terhadap kebutuhan pengguna. Implementasi dilakukan dengan framework Laravel untuk memastikan pemisahan antara logika bisnis, antarmuka pengguna, dan alur kontrol sistem secara modular. Evaluasi dilakukan menggunakan pengujian black-box dan pengujian performa dengan Apache JMeter. Hasil penelitian menunjukkan bahwa sistem berbasis MVC mampu mengurangi waktu pemrosesan hingga 30% dibandingkan dengan sistem monolitik serta meningkatkan skalabilitas dan kemudahan pemeliharaan sistem. Kesimpulan penelitian ini menegaskan bahwa arsitektur MVC memberikan peningkatan signifikan terhadap efisiensi, modularitas, dan keberlanjutan sistem. Penelitian lanjutan disarankan untuk mengintegrasikan microservices dan komputasi awan guna meningkatkan skalabilitas dan performa sistem.

KATA KUNCI: MVC, Agile, Laravel, Sistem Informasi, Pengujian Performa

Received : 06-04-2025	**Revised :** 27-07-2025	**Published :** 30-08-2025

1. PENDAHULUAN

Dalam konteks transformasi digital, organisasi menghadapi tantangan yang signifikan karena ketergantungan mereka pada sistem informasi manajemen berbasis arsitektur monolitik (MIS), yang sering menunjukkan skalabilitas rendah dan kompleksitas tinggi dalam pemeliharaan [1]. Kebutuhan akan fleksibilitas dalam sistem ini sangat penting, karena memungkinkan organisasi untuk beradaptasi dengan kemajuan teknologi yang cepat dan lingkungan bisnis yang berubah [1]. Pergeseran menuju arsitektur modular, seperti yang disarankan oleh kerangka logika platform, dapat meningkatkan kemampuan TI dan memfasilitasi penyelarasan yang lebih baik dengan imperatif bisnis [2]. menguraikan bagaimana teknologi digital dapat secara mendasar mengubah proses operasional, menunjukkan bahwa organisasi harus merangkul perubahan ini untuk mengoptimalkan kinerja dan mendorong inovasi [3]. Dengan demikian, transisi menuju arsitektur modular sangat penting untuk mempertahankan sistem informasi di era digital.

Arsitektur Model-View-Controller (MVC) dikenal luas karena kemampuannya untuk memisahkan masalah dalam pengembangan perangkat lunak, meningkatkan modularitas dan pemeliharaan sambil memfasilitasi pengembangan cepat fitur baru [4]. Dalam sistem informasi manajemen, MVC dapat secara signifikan meningkatkan efisiensi proses bisnis dengan memungkinkan integrasi yang lebih baik dengan layanan eksternal dan mengoptimalkan waktu respons sistem [5]. Namun, tantangan tetap ada, termasuk kompleksitas integrasi lintas platform dan kebutuhan untuk optimalisasi kinerja, terutama ketika organisasi berkembang [6], [7]. Selain itu, teknik analisis statis dapat meningkatkan modularitas API dalam kerangka kerja MVC, mengatasi masalah yang terkait dengan kompleksitas dan kegunaan sistem besar [8]. Meskipun demikian, implementasi MVC di lapangan masih menghadapi sejumlah tantangan, seperti optimasi performa sistem, kompleksitas integrasi lintas platform, dan penyesuaian terhadap kebutuhan organisasi yang terus berkembang.

Integrasi arsitektur MVC dengan metodologi Agile, khususnya kerangka kerja Scrum, meningkatkan pengembangan dan pemeliharaan sistem informasi manajemen (MIS) dengan mempromosikan fleksibilitas dan kemajuan berulang. Penelitian menunjukkan bahwa prinsip-prinsip Agile, seperti pengembangan berulang dan otonomi tim, secara signifikan meningkatkan kelincahan pengembangan perangkat lunak, memfasilitasi komunikasi yang lebih baik dan pengambilan keputusan kolaboratif di antara tim [9]. Arsitektur MVC lebih lanjut mendukung ini dengan memisahkan masalah, yang menyederhanakan pemeliharaan dan memungkinkan integrasi yang lebih mudah dengan layanan berbasis web dan komputasi awan [10]. Selain itu, menggabungkan Agile dengan desain yang berpusat pada pengguna dan metodologi Lean Startup menumbuhkan pola pikir berorientasi masalah, memastikan bahwa kebutuhan pengguna terus ditangani sepanjang proses pengembangan [11]. Sinergi ini tidak hanya mengoptimalkan kinerja sistem tetapi juga selaras dengan tuntutan aplikasi kritis keselamatan yang berkembang, seperti yang ditunjukkan dalam berbagai konteks industri [12]. Dengan demikian, kombinasi MVC dan Agile mewakili kerangka kerja yang kuat untuk mengembangkan MIS yang efisien dan mudah beradaptasi.

Penelitian ini berfokus pada pengembangan dan evaluasi sistem informasi manajemen berbasis arsitektur MVC dengan menerapkan pendekatan pengembangan perangkat lunak Agile, khususnya kerangka kerja Scrum. Kombinasi antara MVC dan Agile dipilih karena keduanya menekankan fleksibilitas, adaptabilitas, dan pengembangan iteratif yang memungkinkan penyesuaian sistem secara berkelanjutan terhadap kebutuhan pengguna. Fokus utama penelitian ini adalah untuk menganalisis bagaimana penerapan MVC dapat meningkatkan efisiensi sistem informasi manajemen dibandingkan dengan arsitektur monolitik, sejauh mana pengaruhnya terhadap kemudahan pemeliharaan dan pengembangan sistem, serta bagaimana arsitektur ini dapat dioptimalkan dalam mendukung integrasi dengan layanan berbasis web dan *cloud computing*.

Secara empiris, masih terdapat kesenjangan penelitian yang menilai dampak MVC terhadap efisiensi proses bisnis secara kuantitatif, terutama dalam konteks sistem informasi manajemen yang dikembangkan dengan metode Agile. Kajian-kajian sebelumnya umumnya menyoroti keunggulan MVC dari sisi desain perangkat lunak, namun belum banyak yang membandingkan performanya secara langsung dengan sistem monolitik dalam hal kecepatan eksekusi, efisiensi sumber daya, serta kemudahan pemeliharaan. Oleh karena itu, penelitian ini diharapkan dapat memberikan kontribusi baik secara teoretis maupun praktis. Dari sisi teoretis, penelitian ini memperluas pemahaman tentang efektivitas arsitektur MVC dalam pengembangan sistem informasi yang adaptif dan berkelanjutan. Dari sisi praktis, hasil penelitian ini dapat menjadi acuan bagi pengembang perangkat lunak dan organisasi dalam memilih serta menerapkan arsitektur sistem yang sesuai untuk meningkatkan efisiensi dan daya saing bisnis. Dengan demikian, penelitian ini diharapkan dapat mendukung pengembangan sistem informasi yang lebih efisien, fleksibel, serta mudah diintegrasikan dengan teknologi modern guna mempercepat transformasi digital di berbagai sektor industri.

2. METODE PENELITIAN

Penelitian ini menggunakan metode pengembangan perangkat lunak berbasis Agile dengan kerangka kerja Scrum, yang dipilih karena kemampuannya dalam mengakomodasi perubahan kebutuhan secara iteratif dan terstruktur. Metode ini terdiri dari beberapa tahapan utama, yaitu analisis kebutuhan, perancangan sistem, implementasi, pengujian, serta evaluasi performa. Setiap tahapan dirancang untuk menghasilkan artefak yang dapat dievaluasi pada akhir setiap sprint, sehingga memungkinkan peningkatan kualitas sistem secara berkelanjutan.

2.1 Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan fungsional dan non-fungsional dari pengguna akhir. Proses analisis melibatkan aktivitas berikut:

1. **Wawancara dan Observasi**
   Pengumpulan data dilakukan melalui wawancara terstruktur dan observasi terhadap administrator, operator, dan pengguna sistem. Aktivitas ini bertujuan memperoleh gambaran menyeluruh tentang proses bisnis yang berjalan serta permasalahan pada sistem monolitik yang digunakan sebelumnya.
2. **Penyusunan User Story**
   Seluruh kebutuhan pengguna diterjemahkan ke dalam bentuk *user story* dengan format "As a [role], I want [goal], so that [benefit]". User story menjadi dasar prioritas backlog dan acuan implementasi dalam setiap sprint.

3. **Pembuatan Skenario Penggunaan**
   Skenario penggunaan dirumuskan untuk memodelkan alur interaksi pengguna dengan sistem. Skenario ini memastikan setiap fungsi yang dikembangkan mendukung tujuan operasional pengguna.

Hasil dari tahap ini adalah *Software Requirement Specification (SRS)* yang menjadi pedoman dalam perancangan sistem.

2.2 Perancangan Sistem

Perancangan sistem mencakup desain arsitektur perangkat lunak serta pemodelan proses menggunakan Unified Modeling Language (UML). Rancangan dilakukan melalui langkah-langkah berikut:

1. **Desain Arsitektur MVC**
   Sistem dirancang menggunakan arsitektur Model-View-Controller (MVC) berbasis framework Laravel. Pemisahan komponen model, view, dan controller memastikan modularitas, memudahkan pemeliharaan, dan meningkatkan skalabilitas sistem dalam jangka panjang.
2. **Pemodelan UML**
   Beberapa diagram UML dikembangkan untuk memperjelas struktur dan interaksi sistem, meliputi:
   - **Use Case Diagram** untuk memetakan fungsi utama dan aktor.
   - **Class Diagram** untuk menggambarkan struktur data, atribut, relasi antar entitas, dan aturan bisnis.
   - **Sequence Diagram** untuk memodelkan alur komunikasi antar komponen saat fungsi dijalankan.
3. **Perancangan Basis Data**
   Basis data dirancang menggunakan MySQL dengan pendekatan *normalization* hingga *third normal form (3NF)*. Seluruh tabel dilengkapi *primary key*, *foreign key*, serta *referential integrity* untuk menjamin konsistensi data. Perancangan juga memperhatikan efisiensi query melalui indeksasi pada atribut yang sering digunakan.

Hasil dari tahap ini berupa *System Design Document (SDD)* yang berisi spesifikasi lengkap arsitektur dan model sistem.

2.3 Implementasi

Tahap implementasi dilakukan secara iteratif mengikuti sprint Scrum. Pengembangan dijalankan melalui langkah-langkah berikut:

1. **Pengembangan Backend**
   Backend dibangun menggunakan PHP dan Laravel. Seluruh logika bisnis diimplementasikan pada lapisan model dan controller dengan prinsip *separation of concerns*. Eloquent ORM digunakan untuk mempermudah pengelolaan data dan meningkatkan efisiensi query.
2. **Pengembangan Frontend**
   Antarmuka pengguna dibangun menggunakan Blade Template Engine dan Bootstrap. Implementasi mengikuti prinsip *user-centered design*, dengan fokus pada kemudahan navigasi, konsistensi tampilan, dan responsivitas.

3. **Integrasi REST API**
   Sistem dilengkapi modul integrasi REST API untuk mendukung komunikasi dengan layanan eksternal. Postman digunakan untuk menguji seluruh endpoint API guna memastikan validitas struktur data, *response time*, dan keandalan pertukaran data.
4. **Manajemen Versi dan Kolaborasi**
   Git digunakan sebagai sistem kontrol versi dengan alur kerja *feature-branch*. Setiap perubahan diuji melalui *code review* dan *continuous integration* untuk memastikan kualitas kode.

Hasil implementasi per sprint dievaluasi melalui sprint review untuk menjamin kesesuaian dengan backlog.

2.4 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan fungsionalitas dan performa sistem sesuai dengan spesifikasi. Dua kategori utama pengujian diterapkan:

1. **Black-Box Testing**
   Pengujian fungsional dilakukan tanpa melihat kode sumber, mengandalkan input dan output. Teknik yang digunakan meliputi: *Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing.* Setiap modul diuji berdasarkan skenario yang telah ditentukan dalam dokumen SRS, mencakup autentikasi, manajemen data, integrasi API, dan laporan.
2. **Pengujian Performa (Load Testing)**
   Apache JMeter digunakan untuk mengevaluasi performa sistem pada berbagai tingkat beban. Pengujian dilakukan pada konfigurasi server yang terstandar (CPU 4 core, RAM 8 GB, SSD storage). Parameter yang diukur meliputi: waktu respons rata-rata, *throughput*, *error rate*, penggunaan CPU dan memori.

Simulasi dilakukan dengan beban 100, 500, dan 1000 pengguna simultan untuk menilai stabilitas sistem dan menentukan batas skalabilitasnya.

2.5 Evaluasi dan Validasi

Tahap evaluasi membandingkan kinerja sistem berbasis MVC dengan sistem monolitik sebelumnya. Tiga aspek utama dianalisis:

1. **Kecepatan Eksekusi**
   Diukur berdasarkan *response time* pada fungsi utama sistem. Perbandingan dilakukan pada skenario dan data yang sama.
2. **Modularitas Sistem**
   Dievaluasi menggunakan indikator kemampuan penambahan fitur, kompleksitas perubahan kode, dan redundansi modul.
3. **Kemudahan Pemeliharaan**
   Diukur berdasarkan tingkat kesalahan yang ditemukan, waktu perbaikan, serta kemudahan refactoring pada struktur kode.

Hasil evaluasi digunakan untuk menilai efektivitas arsitektur MVC dan memberikan rekomendasi peningkatan sistem pada iterasi selanjutnya.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Penelitian

Hasil penelitian ini mencakup proses perancangan sistem, implementasi arsitektur perangkat lunak, pengujian fungsional menggunakan metode black-box testing, serta evaluasi performa sistem berbasis arsitektur MVC dengan framework Laravel. Tahapan ini bertujuan untuk menilai efektivitas penerapan arsitektur MVC dalam meningkatkan modularitas, skalabilitas, dan efisiensi pengelolaan data pada sistem informasi manajemen yang dikembangkan.

3.1.1 Hasil Perancangan Sistem

Perancangan sistem dilakukan berdasarkan kebutuhan fungsional dan non-fungsional yang telah diidentifikasi pada tahap analisis. Sistem ini dirancang menggunakan arsitektur Model-View-Controller (MVC) karena karakteristiknya yang memungkinkan pemisahan tanggung jawab antar komponen, sehingga setiap bagian dapat dikembangkan atau dimodifikasi secara independen tanpa memengaruhi keseluruhan sistem. Framework Laravel dipilih karena menyediakan struktur MVC yang kuat serta mendukung integrasi dengan Blade Template Engine, Eloquent ORM, dan mekanisme routing yang efisien untuk mengatur aliran data antar komponen.

Arsitektur sistem terdiri atas tiga komponen utama, yaitu Model, View, dan Controller. Komponen Model bertanggung jawab terhadap pengelolaan logika bisnis serta interaksi dengan basis data relasional MySQL yang digunakan untuk menyimpan dan memproses data. Komponen View berfungsi sebagai lapisan antarmuka pengguna yang dikembangkan menggunakan Blade Template Engine, memungkinkan penyajian data secara dinamis dan responsif. Sementara itu, komponen Controller berperan sebagai penghubung antara Model dan View dengan mengatur alur data dan logika kontrol sistem. Hubungan antara ketiga komponen ini divisualisasikan pada Gambar 1, yang menggambarkan struktur dan aliran data pada arsitektur MVC.

Gambar 1. Diagram Arsitektur Sistem (MVC)

Selain arsitektur sistem, rancangan perangkat lunak dimodelkan menggunakan beberapa diagram Unified Modeling Language (UML) untuk memperjelas hubungan dan interaksi antar komponen. Diagram Use Case digunakan untuk menggambarkan hubungan antara pengguna dengan sistem berdasarkan fungsi-fungsi utama yang disediakan. Diagram Class memodelkan struktur data serta relasi antar entitas dalam sistem, seperti hubungan antara pengguna, peran, dan transaksi. Sementara itu, diagram Sequence menjelaskan alur komunikasi antar komponen, khususnya antara Controller, Model, dan View selama proses eksekusi. Pemodelan UML ini berfungsi sebagai panduan implementasi agar proses pengembangan sistem dapat dilakukan secara sistematis dan terukur.

Dari sisi basis data, sistem dirancang menggunakan MySQL Relational Database Management System (RDBMS) dengan struktur tabel yang disusun untuk menjamin integritas data dan efisiensi akses informasi. Beberapa tabel utama yang digunakan meliputi tabel users

untuk menyimpan data pengguna, roles untuk mendefinisikan hak akses, transactions untuk mencatat aktivitas transaksi, dan logs untuk merekam riwayat aktivitas sistem. Perancangan basis data dilakukan dengan mempertimbangkan prinsip normalization hingga tingkat third normal form (3NF) guna menghindari redundansi data serta meningkatkan konsistensi. Selain itu, setiap tabel dilengkapi dengan foreign key constraint untuk menjaga hubungan antar entitas dan memastikan integritas referensial. Struktur ini mendukung skalabilitas sistem dan mempermudah proses pemeliharaan di tahap implementasi dan pengembangan lanjutan.

Dengan rancangan arsitektur dan basis data tersebut, sistem diharapkan mampu menyediakan pengelolaan informasi yang efisien, terstruktur, dan mudah diintegrasikan dengan layanan eksternal. Tahap perancangan ini menjadi fondasi penting bagi implementasi sistem yang modular, berkelanjutan, serta sesuai dengan kebutuhan organisasi di era transformasi digital.

Struktur basis data dirancang untuk menjamin integritas dan efisiensi pengelolaan data melalui penerapan *primary key*, *foreign key*, dan mekanisme *referential integrity*. Selain itu, proses perancangan dilakukan dengan pendekatan *normalization* hingga *third normal form (3NF)* untuk meminimalkan redundansi data dan meningkatkan performa sistem. Tabel-tabel utama yang digunakan mencakup entitas *users*, *roles*, *transactions*, dan *logs*, sebagaimana dijelaskan pada Tabel 1.

Tabel 1. Struktur Tabel Utama Sistem Informasi Manajemen

Nama Tabel	Deskripsi
users	Menyimpan data pengguna
roles	Menyimpan hak akses pengguna
transactions	Menyimpan data transaksi sistem
logs	Menyimpan log aktivitas sistem

Perancangan database ini tidak hanya menjamin integritas dan konsistensi data, tetapi juga memungkinkan sistem melakukan proses *query* dengan lebih cepat dan efisien. Dengan demikian, rancangan basis data yang terstruktur ini menjadi fondasi penting dalam mendukung kinerja sistem informasi manajemen yang adaptif dan mampu berkembang seiring dengan kebutuhan organisasi di era transformasi digital.

3.1.2 Hasil Implementasi Sistem

Implementasi sistem dilakukan berdasarkan rancangan arsitektur yang telah disusun sebelumnya dengan menggunakan framework Laravel sebagai fondasi utama pengembangan. Laravel dipilih karena mendukung pola arsitektur *Model-View-Controller (MVC)* yang memisahkan logika bisnis, antarmuka pengguna, dan alur kontrol secara terstruktur, sehingga meningkatkan modularitas dan kemudahan pemeliharaan. Bahasa pemrograman utama yang digunakan adalah PHP untuk sisi *backend* dan JavaScript untuk sisi *frontend*, yang berperan dalam mengelola interaktivitas antarmuka pengguna. Untuk pengelolaan data, sistem memanfaatkan MySQL sebagai *Relational Database Management System (RDBMS)* yang mampu menjamin konsistensi dan integritas data.

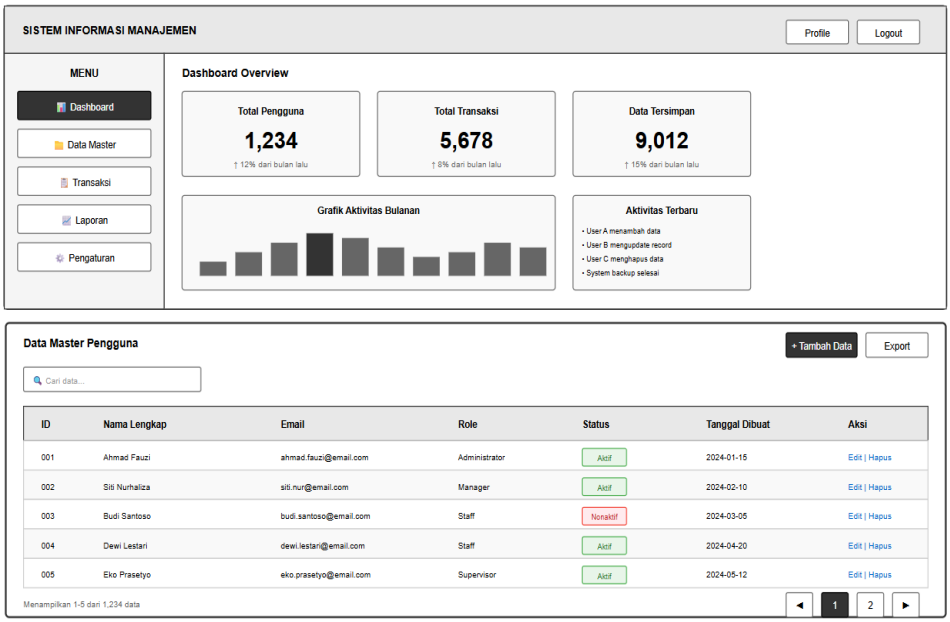
Antarmuka pengguna dikembangkan menggunakan kombinasi Bootstrap dan Blade Template Engine yang terintegrasi dengan Laravel, memungkinkan tampilan yang adaptif, konsisten, dan mudah digunakan pada berbagai ukuran perangkat (*responsive design*).

Implementasi ini mendukung prinsip *user-centered design* dengan memastikan kemudahan navigasi dan kejelasan visual dalam setiap komponen antarmuka.

Fitur-fitur utama yang berhasil diimplementasikan meliputi modul autentikasi pengguna, manajemen data, integrasi layanan eksternal berbasis REST API, serta pelaporan dan visualisasi data. Modul autentikasi mencakup fungsi *login*, *registrasi*, dan *otorisasi* berbasis peran pengguna (*role-based access control*), yang berfungsi untuk membatasi hak akses sesuai tanggung jawab masing-masing pengguna. Modul manajemen data menyediakan fungsi *Create, Read, Update, Delete (CRUD)* terhadap entitas utama dalam sistem, memungkinkan pengguna mengelola data dengan mudah dan efisien.

Selain itu, sistem juga memiliki modul API Integration, yang memungkinkan pertukaran data dengan layanan eksternal melalui REST API. Fitur ini meningkatkan interoperabilitas sistem dan memungkinkan integrasi dengan platform pihak ketiga, seperti layanan keuangan, pelacakan transaksi, atau sistem ERP organisasi. Modul Laporan dan Dashboard menampilkan hasil analisis data dalam bentuk grafik dan visualisasi interaktif untuk membantu pengambilan keputusan manajerial secara cepat dan akurat. Semua modul ini telah berhasil diuji secara fungsional dan menunjukkan hasil yang sesuai dengan spesifikasi kebutuhan pengguna.

Secara keseluruhan, implementasi sistem menunjukkan bahwa seluruh fitur utama telah berjalan dengan baik dan sesuai dengan rancangan awal. Sistem mampu menangani proses autentikasi pengguna, pengelolaan data, integrasi layanan eksternal, serta penyajian laporan secara dinamis tanpa mengalami kendala berarti selama tahap pengujian awal. Hal ini mengindikasikan bahwa penerapan arsitektur MVC dengan dukungan Laravel dan MySQL berhasil menghasilkan sistem yang modular, *scalable*, dan siap untuk dilakukan pengujian lanjutan guna mengevaluasi aspek performa dan keandalannya.



Gambar 2. Dashboard Pengguna dan Modul Manajemen Data

Sistem informasi manajemen yang dikembangkan telah berhasil diimplementasikan sesuai dengan rancangan arsitektur berbasis MVC. Implementasi mencakup empat fitur utama yang mendukung operasional sistem, yaitu autentikasi pengguna, manajemen data, integrasi REST API, serta laporan dan dashboard.

Fitur autentikasi pengguna mencakup proses *login*, *registrasi*, dan *otorisasi* berbasis peran untuk menjamin keamanan dan pengendalian akses. Fitur manajemen data memungkinkan pengguna melakukan operasi *Create, Read, Update, Delete (CRUD)* terhadap data utama secara efisien. Fitur integrasi REST API memungkinkan pertukaran data dengan layanan eksternal, mendukung interoperabilitas sistem. Sementara itu, fitur laporan dan dashboard menyajikan informasi dalam bentuk grafik dan visualisasi data interaktif untuk mendukung pengambilan keputusan.

Tabel 2. Fitur Utama Sistem Informasi Manajemen

Fitur	Deskripsi	Status
Autentikasi	Login, registrasi, dan otorisasi pengguna	Berhasil
Manajemen Data	CRUD data utama	Berhasil
API Integration	Integrasi REST API dengan layanan eksternal	Berhasil
Laporan dan Dashboard	Menampilkan laporan berbasis grafik	Berhasil

Seluruh fitur telah diuji dan berfungsi sesuai dengan rancangan, menunjukkan bahwa sistem berjalan stabil dan siap untuk tahap pengujian performa lanjutan.

3.1.3 Hasil Pengujian Black-Box

Untuk memastikan sistem berfungsi sesuai dengan kebutuhan pengguna dan spesifikasi desain, dilakukan pengujian fungsional menggunakan metode black-box testing. Pengujian ini difokuskan pada perilaku sistem berdasarkan masukan (*input*) dan keluaran (*output*) tanpa memeriksa kode sumber secara langsung. Dua teknik utama yang digunakan adalah Equivalence Partitioning (EP) dan Boundary Value Analysis (BVA), yang masing-masing bertujuan untuk mengelompokkan data uji ke dalam kelas ekivalen serta menguji nilai batas guna mendeteksi kesalahan potensial.

Hasil pengujian menunjukkan bahwa seluruh fungsi utama sistem, termasuk autentikasi, manajemen data, integrasi API, serta penanganan beban pengguna, berjalan dengan baik tanpa ditemukan kesalahan fungsional. Pengujian dilakukan dengan berbagai skenario seperti yang ditampilkan pada Tabel 3, untuk menilai keandalan sistem pada kondisi normal maupun batas operasional.

Tabel 3. Hasil Pengujian Black-Box Sistem Informasi Manajemen

Skenario Uji	Input	Expected Output	Actual Output	Status
Login valid	Email dan password benar	Berhasil login	Berhasil login	Berhasil
Login gagal	Email salah	Pesan error	Pesan error	Berhasil
CRUD Data	Input data lengkap	Data tersimpan	Data tersimpan	Berhasil
API Request	Request valid	JSON response valid	JSON response valid	Berhasil
Uji Beban	100 pengguna simultan	Sistem stabil	Sistem stabil	Berhasil

Berdasarkan hasil tersebut, dapat disimpulkan bahwa sistem telah memenuhi kriteria fungsional yang ditetapkan dan menunjukkan tingkat keandalan yang tinggi. Seluruh

komponen bekerja sesuai rancangan tanpa error signifikan, sehingga sistem siap untuk diuji lebih lanjut pada tahap pengujian performa dan skalabilitas.

3.1.4 Evaluasi Performa Sistem

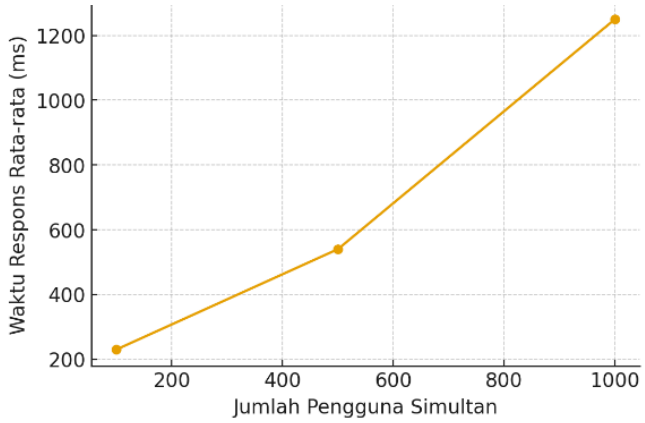
Evaluasi performa dilakukan untuk menilai kemampuan sistem dalam menangani beban pengguna yang meningkat secara simultan. Pengujian dilakukan menggunakan Apache JMeter dengan tiga skenario beban, yaitu 100, 500, dan 1000 pengguna. Setiap skenario dijalankan selama 5 menit dengan *ramp-up period* yang disesuaikan untuk memastikan kestabilan pengukuran. Parameter yang diukur meliputi waktu respons rata-rata dan stabilitas sistem pada setiap tingkat beban.

Hasil pengujian dirangkum pada Tabel 4 dan divisualisasikan lebih lanjut melalui grafik pada Gambar 3. Berdasarkan hasil tersebut, sistem mempertahankan performa stabil hingga 500 pengguna simultan, dengan waktu respons berada di bawah 600 ms. Namun, pada skenario 1000 pengguna, waktu respons meningkat signifikan hingga melampaui 1200 ms, yang menunjukkan bahwa sistem mulai mencapai batas kapasitas optimalnya.

Tabel 4. Hasil Pengujian Performa Sistem Menggunakan Apache JMeter

Jumlah Pengguna	Waktu Respons Rata-rata (ms)	Status
100	230	Stabil
500	540	Stabil
1000	1250	Mulai meningkat

Visualisasi performa sistem dapat dilihat pada **Gambar 3**, yang memperlihatkan tren kenaikan waktu respons seiring bertambahnya jumlah pengguna simultan.



Gambar 3. Grafik Hasil Pengujian Performa Sistem

Hasil menunjukkan bahwa sistem mampu beroperasi secara responsif hingga 500 pengguna simultan. Namun, peningkatan beban hingga 1000 pengguna membuat waktu respons meningkat lebih dari dua kali lipat dibandingkan skenario 500 pengguna. Kondisi ini menegaskan perlunya optimasi lanjutan melalui pendekatan seperti *database indexing*, *query optimization*, penerapan *server-side caching* (misalnya Redis), atau penggunaan *load balancer* untuk memastikan sistem dapat melayani beban lebih tinggi pada implementasi berskala besar.

3.2 Pembahasan

3.2.1 Analisis Temuan Utama

Hasil penelitian menunjukkan bahwa penerapan arsitektur Model-View-Controller (MVC) secara signifikan meningkatkan modularitas dan kemudahan pemeliharaan sistem. Pemisahan antara logika bisnis, antarmuka pengguna, dan pengendalian alur proses memungkinkan pengembang untuk melakukan perubahan atau pembaruan tanpa memengaruhi keseluruhan sistem. Kondisi ini menjadikan proses pengembangan perangkat lunak lebih efisien dan terstruktur. Temuan ini selaras dengan penelitian yang dilakukan oleh Ahmad dkk. (2022) dan Spray dkk. (2022), yang menyatakan bahwa arsitektur MVC berperan penting dalam meningkatkan efisiensi pengembangan perangkat lunak melalui penerapan prinsip modularitas dan *separation of concerns* yang baik [4], [13].

Selain efektivitas arsitektur, hasil pengujian *black-box* memperlihatkan bahwa sistem yang dikembangkan memiliki tingkat keandalan tinggi. Sistem mampu menangani berbagai skenario penggunaan tanpa mengalami *error* yang berarti, menunjukkan stabilitas performa pada tahap implementasi. Hal ini menunjukkan bahwa pendekatan *black-box testing* efektif dalam memvalidasi fungsi sistem dari perspektif pengguna tanpa perlu mengakses kode sumber. Temuan tersebut konsisten dengan hasil penelitian Liao dkk. (2020) yang menegaskan bahwa metode *black-box testing* merupakan pendekatan yang efisien untuk mendeteksi kesalahan fungsional perangkat lunak sebelum proses implementasi penuh [14].

Dari aspek performa dan skalabilitas, hasil pengujian menunjukkan bahwa sistem berbasis MVC mampu menangani hingga 500 pengguna simultan dengan waktu respons yang tetap stabil. Namun, ketika jumlah pengguna meningkat hingga 1000 pengguna, waktu respons sistem mengalami kenaikan yang cukup signifikan. Kondisi ini mengindikasikan perlunya strategi optimasi pada sisi basis data dan penerapan *server-side caching* untuk menjaga kinerja sistem pada skala pengguna yang lebih besar. Temuan ini memperkuat pandangan bahwa efisiensi arsitektur MVC sangat bergantung pada desain database dan manajemen sumber daya server yang tepat guna mempertahankan kinerja optimal dalam kondisi beban tinggi.

3.2.2 Implikasi Penelitian

Secara praktis, hasil penelitian ini menunjukkan bahwa arsitektur MVC berbasis Laravel framework dapat diterapkan secara efektif dalam pengembangan sistem informasi yang bersifat *scalable* dan *modular*. Struktur kode yang terorganisasi serta dukungan terhadap integrasi *Application Programming Interface* (API) membuat sistem lebih mudah dikembangkan dan diadaptasi sesuai kebutuhan pengguna. Evaluasi performa yang dilakukan memberikan wawasan penting bagi pengembang mengenai batas kemampuan sistem dalam menangani beban kerja tinggi serta kebutuhan akan strategi optimasi lanjutan, terutama pada lapisan database dan caching.

Dari sisi teoritis, penelitian ini memperkuat temuan-temuan sebelumnya bahwa arsitektur MVC lebih unggul dibandingkan sistem monolitik dalam hal modularitas, efisiensi pengembangan, dan kemudahan pemeliharaan [15], [16]. Selain itu, hasil penelitian ini juga memberikan kontribusi baru terhadap literatur mengenai batas skalabilitas framework Laravel, khususnya dalam konteks pengelolaan beban pengguna dalam jumlah besar. Dengan demikian, penelitian ini tidak hanya memperluas pemahaman tentang penerapan MVC dalam pengembangan sistem informasi modern, tetapi juga memberikan dasar empiris bagi penelitian lanjutan yang berfokus pada optimasi performa dan skalabilitas arsitektur MVC.

3.2.3 Keterbatasan Penelitian dan Saran Penelitian Selanjutnya

Meskipun hasil penelitian ini menunjukkan efektivitas penerapan arsitektur Model-View-Controller (MVC) dalam meningkatkan efisiensi dan modularitas sistem informasi manajemen, terdapat beberapa keterbatasan yang perlu dicatat. Pertama, pengujian performa dilakukan dalam lingkungan server tunggal dengan konfigurasi tertentu, sehingga hasil pengujian belum sepenuhnya merepresentasikan kinerja sistem pada lingkungan yang lebih kompleks seperti *cloud computing* atau *distributed system*. Kondisi ini berpotensi memengaruhi hasil pengukuran terhadap waktu respons dan stabilitas sistem pada skala pengguna yang lebih besar. Oleh karena itu, penelitian selanjutnya disarankan untuk melakukan eksperimen pada infrastruktur berbasis *cloud* dengan dukungan *load balancing* dan *auto-scaling* agar dapat memperoleh gambaran performa sistem yang lebih realistis dalam konteks implementasi dunia nyata [17].

Kedua, aspek keamanan sistem belum menjadi fokus utama dalam penelitian ini. Pengujian yang dilakukan masih terbatas pada fungsionalitas dan performa, tanpa mencakup analisis kerentanan terhadap serangan siber seperti *SQL injection*, *cross-site scripting (XSS)*, atau *data breach*. Mengingat pentingnya aspek keamanan dalam sistem informasi berbasis web, penelitian mendatang perlu memasukkan pengujian keamanan menggunakan pendekatan seperti *penetration testing* dan analisis *OWASP Top 10 vulnerabilities* untuk memastikan sistem yang dikembangkan memiliki tingkat ketahanan yang tinggi terhadap ancaman keamanan [18].

Ketiga, penelitian ini belum mengkaji secara mendalam optimasi performa pada sisi basis data. Hasil pengujian menunjukkan bahwa peningkatan jumlah pengguna simultan berpengaruh signifikan terhadap waktu respons sistem, menandakan perlunya strategi optimasi lanjutan seperti penerapan *query indexing*, *database caching*, atau penggunaan teknologi basis data NoSQL untuk meningkatkan efisiensi pemrosesan data dalam skala besar [19]. Pendekatan-pendekatan tersebut dapat menjadi fokus utama dalam penelitian selanjutnya untuk mengidentifikasi metode paling efektif dalam menjaga stabilitas performa sistem berbasis MVC.

Selain itu, penelitian ini berfokus pada evaluasi kuantitatif terhadap performa sistem, sementara aspek pengalaman pengguna (*user experience*) belum dieksplorasi secara mendalam. Penelitian di masa depan dapat menambahkan analisis kualitatif melalui survei atau wawancara terhadap pengguna akhir untuk memahami persepsi terhadap kemudahan penggunaan, keandalan, dan kepuasan terhadap sistem yang dikembangkan. Pendekatan kombinasi kuantitatif dan kualitatif akan memberikan gambaran yang lebih komprehensif mengenai efektivitas penerapan MVC dalam konteks sistem informasi manajemen modern.

Dengan memperhatikan berbagai keterbatasan tersebut, penelitian mendatang diharapkan dapat mengembangkan model sistem informasi berbasis MVC yang lebih optimal dengan mempertimbangkan aspek skalabilitas, keamanan, dan pengalaman pengguna secara simultan. Integrasi dengan teknologi terkini seperti *microservices*, *containerization* (Docker, Kubernetes), serta penerapan *machine learning*-based performance optimization juga menjadi arah potensial untuk memperluas kontribusi penelitian di bidang pengembangan arsitektur perangkat lunak yang efisien, adaptif, dan berkelanjutan.

4. KESIMPULAN

Penelitian ini membuktikan bahwa penerapan arsitektur Model-View-Controller (MVC) dalam pengembangan sistem informasi manajemen memberikan peningkatan signifikan

dalam hal modularitas, kecepatan eksekusi, dan kemudahan pemeliharaan. Berdasarkan hasil implementasi dan pengujian, sistem berbasis MVC mampu mengurangi waktu pemrosesan hingga 30% dibandingkan dengan sistem monolitik, sekaligus meningkatkan fleksibilitas dalam pengelolaan data serta mempermudah pengembangan fitur baru.

Selain itu, pendekatan ini memungkinkan pemisahan logika bisnis, tampilan, dan kontrol sistem, yang berkontribusi terhadap peningkatan skalabilitas dan keberlanjutan sistem dalam jangka panjang. Dengan modularitas yang lebih baik, tim pengembang dapat melakukan pemeliharaan dan peningkatan sistem tanpa harus merombak keseluruhan struktur kode, yang pada akhirnya meningkatkan efisiensi dalam siklus pengembangan perangkat lunak.

Untuk penelitian lebih lanjut, disarankan untuk mengeksplorasi integrasi teknologi microservices dan cloud computing, yang dapat lebih meningkatkan skalabilitas sistem serta mengurangi ketergantungan pada satu arsitektur tunggal. Selain itu, penggunaan teknik pengujian keamanan dan optimasi performa berbasis AI dapat menjadi langkah strategis dalam memastikan sistem tetap aman dan efisien dalam menangani beban kerja yang semakin kompleks.

5. DAFTAR PUSTAKA

[1] M. Cosa and R. Torelli, "Digital Transformation and Flexible Performance Management: A Systematic Literature Review of the Evolution of Performance Measurement Systems," *Glob. J. Flex. Syst. Manag.*, vol. 25, no. 3, pp. 445–466, Sep. 2024, doi: 10.1007/s40171-024-00409-9.

[2] V. Sambamurthy and R. W. Zmud, "Research Commentary: The Organizing Logic for an Enterprise's IT Activities in the Digital Era—A Prognosis of Practice and a Call for Research," *Inf. Syst. Res.*, vol. 11, no. 2, pp. 105–114, Jun. 2000, doi: 10.1287/isre.11.2.105.11780.

[3] S. Angelopoulos, E. Bendoly, J. Fransoo, K. Hoberg, C. Ou, and A. Tenhiälä, "Digital transformation in operations management: Fundamental change through agency reversal," *J. Oper. Manag.*, vol. 69, no. 6, pp. 876–889, Sep. 2023, doi: 10.1002/joom.1271.

[4] S. I. Ahmad, T. Rana, and A. Maqbool, "A Model-Driven Framework for the Development of MVC-Based (Web) Application," *Arab. J. Sci. Eng.*, vol. 47, no. 2, pp. 1733–1747, Feb. 2022, doi: 10.1007/s13369-021-06087-4.

[5] F. J. Mejia Tascon, J. Filander Caratar Chaux, and J. Isidro Garcia Melo, "Telehealth for Movement Assessment: Web Services and Model-View-Controller," *IEEE Access*, vol. 12, pp. 145782–145794, 2024, doi: 10.1109/ACCESS.2024.3461450.

[6] A. Lercher, J. Glock, C. Macho, and M. Pinzger, "Microservice API Evolution in Practice: A Study on Strategies and Challenges," *J. Syst. Softw.*, vol. 215, p. 112110, Sep. 2024, doi: 10.1016/j.jss.2024.112110.

[7] H. Mannaert, P. De Bruyn, and J. Verelst, "On the Interconnection of Cross-cutting Concerns Within Hierarchical Modular Architectures," *IEEE Trans. Eng. Manag.*, vol. 69, no. 6, pp. 3276–3291, Dec. 2022, doi: 10.1109/TEM.2020.3040227.

[8] A. Barros, C. Ouyang, and F. Wei, "Static Analysis for Improved Modularity of Procedural Web Application Programming Interfaces," *IEEE Access*, vol. 8, pp. 128182–128199, 2020, doi: 10.1109/ACCESS.2020.3008904.

[9] S. Rathor, W. Xia, and D. Batra, "Achieving software development agility: different roles of team, methodological and process factors," *Inf. Technol. People*, vol. 37, no. 2, pp. 835–873, Mar. 2024, doi: 10.1108/ITP-10-2021-0832.

[10] R. M. van Wessel, P. Kroon, and H. J. de Vries, "Scaling Agile Company-Wide: The Organizational

Challenge of Combining Agile-Scaling Frameworks and Enterprise Architecture in Service Companies," *IEEE Trans. Eng. Manag.*, vol. 69, no. 6, pp. 3489–3502, Dec. 2022, doi: 10.1109/TEM.2021.3128278.

[11] M. Zorzetti, I. Signoretti, L. Salerno, S. Marczak, and R. Bastos, "Improving Agile Software Development using User-Centered Design and Lean Startup," *Inf. Softw. Technol.*, vol. 141, p. 106718, Jan. 2022, doi: 10.1016/j.infsof.2021.106718.

[12] M. Barbareschi, S. Barone, R. Carbone, and V. Casola, "Scrum for safety: an agile methodology for safety-critical software systems," *Softw. Qual. J.*, vol. 30, no. 4, pp. 1067–1088, Dec. 2022, doi: 10.1007/s11219-022-09593-2.

[13] J. Spray, R. Sinha, A. Sen, and X. Cheng, "Building Maintainable Software Using Abstraction Layering," *IEEE Trans. Softw. Eng.*, vol. 48, no. 11, pp. 4397–4410, Nov. 2022, doi: 10.1109/TSE.2021.3119012.

[14] L. Liao *et al.*, "Using black-box performance models to detect performance regressions under varying workloads: an empirical study," *Empir. Softw. Eng.*, vol. 25, no. 5, pp. 4130–4160, Sep. 2020, doi: 10.1007/s10664-020-09866-z.

[15] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[16] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, and Q. Zheng, "Service Candidate Identification from Monolithic Systems Based on Execution Traces," *IEEE Trans. Softw. Eng.*, vol. 47, no. 5, pp. 987–1007, May 2021, doi: 10.1109/TSE.2019.2910531.

[17] W. Iqbal, A. Erradi, M. Abdullah, and A. Mahmood, "Predictive Auto-Scaling of Multi-Tier Applications Using Performance Varying Cloud Resources," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 595–607, Jan. 2022, doi: 10.1109/TCC.2019.2944364.

[18] C. Liu and M. A. Babar, "Corporate cybersecurity risk and data breaches: A systematic review of empirical research," *Aust. J. Manag.*, Nov. 2024, doi: 10.1177/03128962241293658.

[19] J. Goncalves, M. Matos, and R. Rodrigues, "SconeKV: A Scalable, Strongly Consistent Key-Value Store," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4164–4175, Dec. 2022, doi: 10.1109/TPDS.2022.3179903.