JURNAL FASILKOM
Volume 15 No. 3 | Desember 2025: 482-490

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Implementasi Otomatisasi *Data Lifecycle Management* (DLM) untuk Peningkatan Skalabilitas dan Keandalan Sistem Informasi Pemesanan Kelas

Atta Zulfahrizan[1], Dedy Kiswanto[2], Sybil Auzi[3], Josua Tampubolon[4]
[1,2,3,4] Ilmu Komputer, Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Negeri Medan
[1]latifhamzah106@gmail.com, [2]dedykiswanto@unimed.ac.id, [3]saibil1492@gmail.com*, [4]josuatampubolon30@gmail.com

Abstract

The rapid growth of historical data in class booking information systems can significantly degrade performance, impacting system scalability and reliability. This research addresses the issue by designing and implementing an automated Data Lifecycle Management (DLM) framework. The primary objectives are: (1) to develop a functional automated DLM prototype using the Laravel framework and its task scheduler, and (2) to analyze the extent to which this implementation enhances system scalability and reliability. This study adopts a system implementation method by applying the seven stages of DLM, supported by a Dual Connection database architecture that separates operational data (Hot Storage) from historical archives (Cold Storage). Benchmark test results indicate that this architecture maintains optimal read latency at approximately 14.8 ms by isolating operational data, effectively preventing the 47% performance degradation observed in stress tests with 100,000 records. The system automatically archives weekly transactional data and permanently deletes them after a retention period of one semester plus a 30-day grace period. Furthermore, a secure public API was developed to facilitate data sharing for academic purposes. The implementation of automated DLM proves effective in managing the data lifecycle, reducing the burden on the primary database, and maintaining system performance, thereby ensuring better scalability and reliability.

Keywords: data lifecycle management (DLM), DLM automation, automated data archiving, laravel scheduler.

Abstrak

Pertumbuhan data historis yang pesat pada sistem informasi pemesanan kelas dapat menurunkan performa secara signifikan, yang berdampak pada skalabilitas dan keandalan sistem. Penelitian ini mengatasi masalah tersebut dengan merancang dan mengimplementasikan kerangka kerja *Data Lifecycle Management* (DLM) yang terotomatisasi. Tujuan utama penelitian ini adalah: (1) mengembangkan prototipe DLM fungsional yang terotomatisasi menggunakan *framework* Laravel dan penjadwal tugasnya, serta (2) menganalisis sejauh mana implementasi ini meningkatkan skalabilitas dan keandalan sistem. Studi ini menggunakan metode implementasi sistem dengan menerapkan tujuh tahapan DLM yang didukung oleh arsitektur *database Dual Connection* untuk memisahkan data operasional (*Hot Storage*) dan arsip historis (*Cold Storage*). Hasil pengujian *benchmark* menunjukkan bahwa arsitektur ini mampu menjaga latensi pembacaan (*read latency*) tetap optimal di angka ~14.8 ms dengan mengisolasi data operasional, serta efektif mencegah degradasi performa sebesar 47% yang teramati pada uji beban dengan 100.000 data. Sistem secara otomatis mengarsipkan data transaksional mingguan dan menghapusnya secara permanen setelah melewati masa retensi satu semester ditambah periode tenggang 30 hari. Lebih lanjut, sebuah API publik yang aman dikembangkan untuk memfasilitasi pembagian data untuk kebutuhan akademik. Implementasi DLM terotomatisasi ini terbukti efektif dalam mengelola siklus hidup data, mengurangi beban database utama, dan menjaga performa sistem, sehingga menjamin skalabilitas dan keandalan yang lebih baik.

Kata kunci: *data lifecycle management* (DLM), otomatisasi DLM, pengarsipan data otomatis, laravel *scheduler*.

©This work is licensed under a Creative Commons Attribution - ShareAlike 4.0 International License

1. Pendahuluan

Pesatnya kemajuan teknologi informasi telah mendorong pemanfaatan sistem berbasis web sebagai solusi untuk optimalisasi efisiensi di berbagai sektor [1]. Sejalan dengan itu, sistem penjadwalan berbasis web yang terintegrasi dengan notifikasi otomatis telah diterapkan untuk mempermudah koordinasi dan penyampaian informasi secara real-time [2]. Pemanfaatan automasi dalam proses bisnis, seperti yang ditunjukkan pada sistem check-in dan check-out asrama, terbukti dapat mempersingkat waktu pelaksanaan tugas-tugas administratif secara signifikan [3]. Dalam konteks ini, framework modern seperti Laravel menjadi pilihan populer untuk membangun aplikasi manajemen data yang terstruktur dan andal, salah satunya pada sistem pengelolaan arsip digital [4].

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Konsep pengelolaan siklus hidup (*lifecycle management*) merupakan strategi penting untuk meningkatkan efisiensi dan keberlanjutan melalui integrasi teknologi [5]. Ketika diterapkan pada data, pendekatan *Data Lifecycle Management* (DLM) yang cerdas dan digerakkan oleh AI terbukti mampu mengoptimalkan efisiensi penyimpanan serta kinerja sistem secara keseluruhan melalui otomatisasi proses klasifikasi dan pengarsipan [6]. Kebutuhan akan DLM yang efektif didorong oleh evolusi arsitektur penyimpanan data yang terus beradaptasi untuk menangani volume dan format data yang semakin kompleks [7]. Selain itu, penerapan sistem manajemen data terpusat seperti *Document Management System* (DMS) berbasis *web* juga menunjukkan peran penting dalam mendukung digitalisasi arsip dan menjaga keandalan sistem informasi [8]. Performa sistem sangat bergantung pada format data yang digunakan, di mana pemilihan format yang tepat dapat mempercepat proses analisis dan meningkatkan efisiensi pemrosesan kueri [9]. Keamanan jaringan telah menjadi salah satu aspek krusial di era digital yang semakin terhubung [10]. Oleh karena itu, aspek keamanan data selama proses penyimpanan dan pengarsipan tetap menjadi perhatian utama, terutama untuk menjamin integritas dan keandalan sistem informasi yang terotomatisasi [11].

Di tengah transformasi digital ini, manajemen data yang efisien dan berkelanjutan merupakan aspek krusial dalam operasional sistem informasi modern, terutama ketika volume data terus bertumbuh secara eksponensial. Dalam berbagai domain, termasuk sistem informasi pemesanan sumber daya seperti ruang kelas, data historis yang tidak lagi aktif seringkali terakumulasi dalam *database* operasional. Fenomena ini secara signifikan dapat menurunkan performa sistem, yang tercermin dari lambatnya waktu respons kueri, penggunaan memori yang tidak efisien, dan meningkatnya kompleksitas pemeliharaan. Tanpa adanya kebijakan siklus hidup data yang terdefinisi dengan jelas, pengembang sistem dihadapkan pada tantangan besar untuk menjaga agar sistem tetap cepat, aman, dan andal.

Untuk mengatasi tantangan tersebut, kerangka kerja *Data Lifecycle Management* (DLM) menawarkan pendekatan sistematis untuk mengelola alur data, mulai dari penciptaan hingga penghancuran [12]. Tinjauan terhadap penelitian terdahulu menunjukkan bahwa DLM telah menjadi fokus penting dalam berbagai konteks. Pada penelitian [13] ditunjukkan bahwa implementasi DLM yang terotomatisasi pada platform enterprise seperti SAP mampu memberikan penghematan biaya penyimpanan hingga 40% dan peningkatan kinerja sistem yang signifikan. Kemudian pada penelitian [14] dijelaskan pendekatan yang lebih canggih dengan memanfaatkan model prediktif berbasis *machine learning* untuk menekan biaya penyimpanan sebesar 28-42% di lingkungan big data. Pada penelitian [15] diidentifikasi bahwa DLM standar seringkali tidak memadai untuk menangani dark data yang dikumpulkan namun tidak pernah digunakan yang cenderung menumpuk dan menurunkan efisiensi. Selain itu, pada penelitian [16] diperkenalkan kerangka kerja *Data Lifecycle Framework* (DaLiF) yang menekankan pentingnya otomatisasi untuk menghadapi kompleksitas volume data yang besar, sekaligus memastikan skalabilitas dan keandalan sistem pemerintahan.

Meskipun penelitian mengenai DLM telah mencakup kerangka kerja konseptual hingga solusi spesifik untuk *platform enterprise* dan lingkungan big data [13] [14] [16], terdapat celah penelitian yang signifikan. Mayoritas studi yang ada cenderung berfokus pada pembahasan tingkat tinggi atau solusi yang sangat spesifik untuk *platform proprietary*. Masih sangat sedikit studi kasus implementasi teknis *end-to-end* yang menekankan otomatisasi penuh dalam DLM pada sistem informasi skala kecil hingga menengah yang dikembangkan secara *custom*, terutama pada domain pendidikan seperti sistem pemesanan kelas. Penelitian ini bertujuan untuk mengisi celah tersebut.

Oleh karena itu, penelitian ini berfokus pada perancangan dan implementasi otomatisasi *Data Lifecycle Management* (DLM) pada sistem informasi pemesanan kelas berbasis web. Tujuan utama dari penelitian ini adalah: (1) Mengembangkan sebuah prototipe fungsional DLM yang terotomatisasi menggunakan framework Laravel dan penjadwal tugasnya. (2) Menganalisis sejauh mana implementasi DLM otomatis dapat meningkatkan skalabilitas dan keandalan sistem dengan mengelola, mengarsipkan, dan menghapus data historis secara efisien sesuai kebijakan retensi yang ditetapkan.

2. Metode Penelitian

Data Life Cycle Management (DLM) merupakan kerangka kerja sistematis untuk mengatur data mulai dari proses penciptaan hingga penghapusan, dengan tujuan menjaga efisiensi penyimpanan, integritas data, serta kepatuhan terhadap kebijakan keamanan. Dalam proyek *System Integrated Management* (SIM) untuk pemesanan kelas, DLM diimplementasikan secara otomatis melalui Laravel *Scheduler* dan arsitektur *dual connection MySQL*. Pendekatan ini diterapkan untuk mengatasi potensi penumpukan dark data, meningkatkan skalabilitas sistem, serta mendukung efisiensi pengelolaan data akademik yang terstruktur.

2.1. Arsitektur Data dan Infrastruktur Ganda (*Dual Connection*)

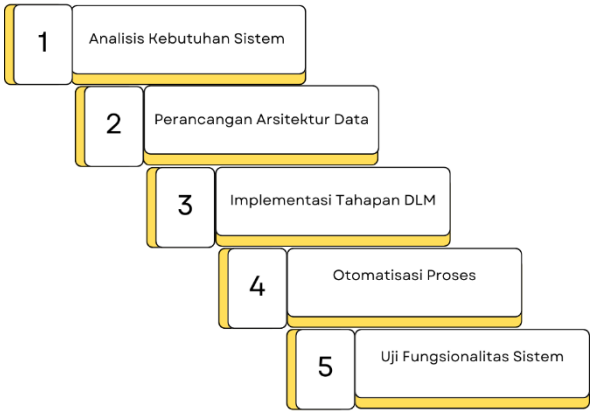
Tahap awal implementasi DLM difokuskan pada pemisahan antara data operasional (*Hot Storage*) dan data arsip (*Cold Storage*). Untuk mencapai tujuan tersebut, arsitektur database diubah dari koneksi tunggal menjadi koneksi ganda (*Dual Connection*) pada *framework Laravel*, dengan konfigurasi sebagai berikut:

P-ISSN : 2089-3353
E-ISSN : 2808-9162

- Database Aktif (*Hot Storage*): Menyimpan data operasional aktif, seperti pemesanan kelas tujuh hari terakhir.
- Database Arsip (*Cold Storage*): Menyimpan data historis jangka menengah dan panjang, dengan periode retensi lebih dari enam bulan.

Kedua koneksi ini didaftarkan dalam file config/database.php dengan nama koneksi khusus ('arsip'). *Model Eloquent* yang mengelola data arsip, seperti *Archived*Pesan, menggunakan koneksi tersebut untuk memastikan pemisahan akses dan keamanan antar database

Penelitian ini mengadopsi metode implementatif berbasis rekayasa sistem (*system implementation method*). Fokus studi ini adalah pada proses penerapan dan pembuktian fungsionalitas konsep *Data Lifecycle Management* (DLM) dalam sistem informasi Manajemen (SIM) pemesanan kelas, alih-alih melakukan pengujian komparatif performa sebelum dan sesudah implementasi.



Gambar 1. *Flowchart* Urutan Penelitian

Tahapan penelitian seperti yang ditampilkan di gambar 1 dilaksanakan secara sistematis sebagai berikut:

1. Analisis Kebutuhan Sistem: tahap ini bertujuan mengidentifikasi entitas data utama yang tunduk pada siklus hidup data secara penuh, meliputi data pemesanan kelas, mata kuliah pengganti, dan akun pengguna.

2. Perancangan Arsitektur Data: merancang struktur *database* menggunakan pendekatan *Dual Connection*, yang secara fungsional memisahkan penyimpanan data operasional (*Hot Storage*) dari data historis (*Cold Storage*).

3. Implementasi Tahapan DLM: menerapkan ketujuh fase *Data Lifecycle Management* (*Creation, Storage, Usage, Sharing, Archiving, Retention, Deletion*) menggunakan framework Laravel 11 dengan basis data MySQL.

4. Otomatisasi Proses: memanfaatkan laravel *scheduler* untuk menjadwalkan dan mengeksekusi proses pengarsipan dan penghapusan data secara mandiri.

5. Uji Fungsionalitas Sistem: melakukan serangkaian pengujian untuk memverifikasi bahwa setiap fase DLM—seperti validasi input, perpindahan data arsip, dan pemicu penghapusan otomatis berdasarkan *Retention Policy*—berfungsi sesuai desain.

Pendekatan ini dirancang untuk menjamin implementasi DLM dapat beroperasi secara efisien dan aman pada sistem berbasis web, menghasilkan model pengelolaan data yang mudah diintegrasikan dalam konteks akademik.

2.2. Struktur Tabel DB Aktif (*Hot Storage*) & Arsip (*Cold Storage*)

Pada *Hot Storage* yang *databasenya* dinamakan sim, data yang sering digunakan adalah data pada tabel matkulganti, pesan dan juga *users*. yang mana pada tabel ini lah setiap pemesanan keleas dilakukan.

Sebagai fondasi *Cold Storage*, telah dibuat tiga tabel arsip di database sim_archive. Struktur tabel arsip ini identik dengan tabel aktif, ditambah satu kolom penanda waktu kunci: archived_at (*TIMESTAMP*) yang mencatat kapan data dipindahkan.

Tabel 1.Tabel Penamaan pada Tabel *Database Hot Storage* & *Cold Storage*

Tabel Aktif	Tabel Arsip	Keterangan
matkulganti	matkulganti-archived	Menampung data matkul pengganti terkait pemesanan yang diarsipkan
pesan	pesan-archived	Menampung data pemesanan yang sudah selesai.
users	users-archived	Tabel *users* tidak akan dilakukan penghapusan otomatis, dan akan diterapkan pewarisan akun dari komting lama ke komting baru jika terjadi perubahan komting.

Tabel 1 menampilkan penamaan tabel pada *database* utama dan juga pada *database* arsip, yang mana perbedaannya ada di penamaannya yang mana tabel *database* arsip ada keterangan tambahan '*-archived'* sebagai pembeda dari database utama.

2.3. Implementasi Berdasarkan 7 Tahapan DLM Tabel

Pada subbab ini akan menjelaskan bagaimana kami mengimplementasikan *Data Lifecycle Management* (DLM) pada sistem kami

2.3.1 *Data Creation*

Tahap pertama adalah *data creation,* yaitu proses penciptaan data setiap kali pengguna melakukan pemesanan kelas. Data yang dimasukkan meliputi nama mata kuliah yang berpindah, kegiatan yang akan dilakukan, waktu mulai dan berakhirnya pemakaian kelas, serta tanggal pelaksanaan.

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Sistem secara otomatis menentukan hari dari tanggal yang dipilih dan menyimpan data tersebut dalam tabel yang sesuai. Proses ini dijalankan secara otomatis oleh Laravel dan tervalidasi agar tidak terjadi kesalahan input atau duplikasi data

2.3.2 *Data Storage*

Untuk penyimpanan, akan digunakan *database* MySQL. Pada peyimpanan juga diimplementasikan arsitektur *dual connection*, yang maksudnya pada sistem ini menggunakan dua database:

1. *Hot Storage* (DB Aktif): digunakan untuk menyimpan data aktif operasional dan juga transaksional (pesan, matkulganti, matakuliah, kelas).

2. *Cold Storage* (DB Arsip): Digunakan untuk menyimpan data historis yang tidak akan digunakan lagi dalam operasional. Database ini hanya berisi tabel transaksional yang isinya sama seperti yang ada di DB Aktif namun ditambahkan kolom *archived*_at untuk memberitahu kapan data ini diarsipkan.

Penerapan *dual connection* pada *database* ditujukan agar *database* utama tidak mengalami penurunan performa jika yang dapat disebabkan jika sudah telalu banyak data yang disimpan. Jadi dengan adanya *dual connection* dapat dipastikan *database* utama tetap berjalan dengan beban yang lebih ringan.

2.3.3 *Data Usage*

Tahap *data usage* pada penelitian ini berperan dalam menjaga integritas jadwal kelas. Sistem memastikan agar tidak terjadi tumpang tindih waktu pemesanan (*overlapping*). Misalnya, jika Komting kelas A telah memesan ruang A pada tanggal tertentu pukul 08.00–10.00, maka Komting lain tidak akan dapat memesan ruang yang sama pada waktu tersebut.

Selain itu, data yang tersimpan pada tahap ini dimanfaatkan secara aktif oleh sistem untuk menampilkan jadwal pemakaian kelas secara *real-time* kepada pengguna. Setiap permintaan pemesanan akan dibandingkan dengan data yang sudah ada di *database* aktif untuk memastikan ketersediaan ruangan. Proses validasi ini dilakukan secara otomatis oleh sistem sebelum data disimpan, sehingga dapat meminimalkan kesalahan jadwal serta meningkatkan keandalan sistem dalam pengelolaan pemakaian kelas.

2.3.4 *Data Sharing*

Pada tahap *data sharing*, sistem memungkinkan akses publik terhadap data yang sudah diarsipkan untuk keperluan akademik atau penelitian. Data arsip disediakan melalui *endpoint* API publik yang dirancang tanpa menyertakan informasi sensitif pengguna.

Dengan cara ini, data dapat dimanfaatkan secara transparan oleh universitas maupun pihak luar tanpa mengorbankan privasi pengguna.

2.3.5 *Data Archiving*

Data archiving pada projek ini akan dilakukan pada database MySQL yang hanya memuat data tabel transaksional yang kolomnya akan sama persis seperti yang ada di *hot storage* atau DB Aktif dengan tambahan *Timestamp archived*_at agar dapat memberitahu kapan data ini diarsipkan.

Proses pengarsipan dilakukan secara otomatis menggunakan mekanisme scheduler pada Laravel sesuai dengan kebijakan retensi yang telah ditentukan. Data yang sudah tidak digunakan dalam operasional harian akan disalin ke *database* arsip, kemudian dihapus dari database aktif. Dengan pendekatan ini, sistem dapat menjaga performa database utama tetap optimal sekaligus memastikan bahwa data historis tetap tersedia untuk kebutuhan pelaporan, audit akademik, dan penelitian di masa mendatang.

2.3.6 *Data Retention and Complience*

Data retention adalah sebuah peraturan yang menentukan seberapa lama data akan disimpan. Pada penelitian ini, kebijakan *data retention and Compliance* diterapkan untuk menentukan lama waktu penyimpanan data.

Tabel 2. *Retention Policy* yang Diterapkan Dalam Penelitian

Tahap DLM	Aturan Retensi	Pemicu Otomatis
Aktif (Hot Storage)	Data pemesanan (dan matkulganti terkait) harus berada di database aktif	Data minggu ini akan tetap berada di database aktif sampai Sabtu pukul 23:59:59.
Pengarsipan	Data dipindahkan dari database aktif ke database arsip.	Minggu Pagi (Pukul 08:00), Skrip akan menyalin dan menghapus semua data yang berada pada minggu ini. Skrip mengisi kolom archived_at.
Retensi Jangka Panjang	Data disimpan di arsip untuk pelaporan historis dan audit	Disimpan selama 1 Semester penuh.

Seperti yang sudah dikatakan dalam tabel 2, data aktif disimpan selama satu minggu di *hot storage,* sedangkan data arsip disimpan selama satu semester penuh. Setelah melewati masa penyimpanan dan tambahan waktu *grace period* selama 30 hari, sistem akan secara otomatis menghapus data tersebut. Kebijakan ini memastikan keseimbangan antara efisiensi penyimpanan dan kebutuhan audit akademik ataupun penelitian.

2.3.7 *Data Deletion*

Tahap terakhir pada DLM Adalah *deletion* atau penghapusan data. Penghapusan dilakukan pada saat perkuliahan semester selesai dan 30 hari setelahnya. Proses ini memberikan waktu *grace period* selama 30 hari sebelum data benar-benar dihapus secara permanen.

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Langkah ini dilakukan untuk memberikan kesempatan jika masih ada kebutuhan terhadap data pemakaian kelas.

2.4. Implementasi Berdasarkan 7 Tahapan DLM Tabel

Untuk memvalidasi klaim skalabilitas dan keandalan, penelitian ini menyertakan skenario pengujian beban (*load testing*) yang terukur. Pengujian dilakukan pada lingkungan lokal dengan spesifikasi server PHP 8.5 dan basis data MySQL. Metodologi pengujian dirancang untuk membandingkan metrik latensi (*latency*) antara database yang menerapkan DLM dengan *database* konvensional yang menumpuk data.

Proses pengujian menggunakan teknik *seeding* dengan algoritma *Collision-Free Strategy* untuk membangkitkan data *dummy* jadwal dalam jumlah besar (hingga 100.000 baris) tanpa melanggar batasan integritas relasional. Parameter yang diukur meliputi *Read Latency* (waktu respons pembacaan data) dan *Write Latency* (waktu simpan data baru termasuk pengecekan jadwal bentrok). Pengujian dibagi menjadi dua skenario utama:

1. Skenario ideal (*Archived*): merepresentasikan kondisi sistem dengan DLM aktif, di mana data operasional dijaga seminimal mungkin (0–5.000 data).
2. Skenario *stress test:* merepresentasikan kondisi sistem tanpa DLM, di mana data historis dibiarkan menumpuk hingga 55.000 dan 100.000 data.

3. Hasil dan Pembahasan

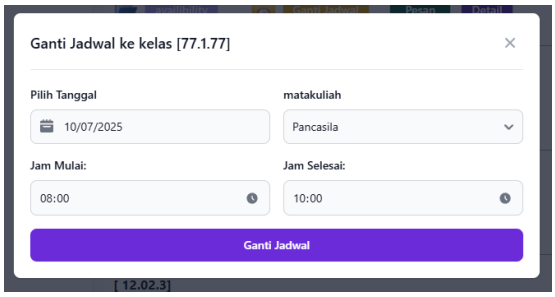
Bab ini membahas hasil implementasi dari setiap tahapan *Data Life Cycle Management* (DLM) yang telah dijelaskan pada bab sebelumnya. Implementasi dilakukan pada sistem *System Integrated Management* (SIM) berbasis *web* menggunakan *framework Laravel* 11, dengan MySQL sebagai basis data utama. Fokus pengujian diarahkan pada pengelolaan dan keamanan data, mencakup proses penciptaan, penyimpanan, penggunaan, berbagi, pengarsipan, penghapusan, serta tata kelola data secara menyeluruh.

3.1. Hasil Implementasi Tiap Tahapan DLM Spesifikasi

Pada subbab ini akan menjelaskan bagaimana tiap-tiap tahapan DLM sudah diterapkan pada *web* pemesanan kelas.

3.1.1 *Data Creation*

Tahap penciptaan (*creation*) data terjadi setiap kali pengguna melakukan pemesanan kelas melalui sistem. Setiap pemesanan akan menghasilkan entri baru yang berisi informasi pengguna, mata kuliah, jadwal, dan ruangan yang dipilih. Proses ini dilakukan secara otomatis melalui antarmuka aplikasi dan direkam dalam tabel utama di database aktif.



Gambar 2. Proses Terjadinya Penciptaan (*Creation*) Data

Pada gambar 2 dapat dilihat bahwa proses penciptaan (*Creation*) data pada penelitian ini membutuhkan *user* yang mana adalah komting untuk mengisi sebuah formulir yang harus diisi. Formulir mengandung tanggal pemesanan kelas, mata kuliah yang jadwalnya ingin dirubah, jam mata kuliah tersebut dimulai dan juga berakhir, pada saat user menekan ganti jadwal maka data yang dimasukkan akan dikirim ke *database* utama.

3.1.2 Data Storage

Data yang tercipta disimpan di dalam database berbasis MySQL. Data yang tercipta disimpan di dalam database utama berbasis MySQL. Database ini berfungsi sebagai Hot Storage yang menampung data operasional aktif selama satu minggu. Mekanisme penyimpanan dilakukan dengan mempertahankan konsistensi dan integritas data melalui relasi antar tabel serta validasi otomatis dari sistem.

Tabel 3. Matkulganti yang Menyimpan Data Pemindahan Jadwal

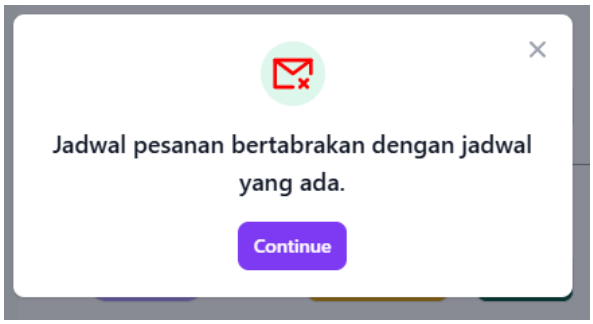
ID	Matkul ID	Kelas ID	Tanggal Ganti	Hari	Jam Mulai	Jam Akhir	Status	*User Class*
578	38	1	2025-10-07	Selasa	8:00:00	10:00:00	dibooking	PSIK 23D

Tabel 3 menampilkan tabel matkulganti yang menyimpan data pemindahan jadwal memperlihatkan bahwa data pemesanan tersimpan dengan struktur tabel yang konsisten dan terhubung secara relasional. Validasi otomatis Laravel menjaga integritas data agar tidak terjadi redundansi atau ketidaksesuaian antar tabel.

3.1.3 *Data Usage*

Data yang tersimpan digunakan oleh sistem untuk memproses setiap permintaan pemesanan baru. Fungsi utamanya adalah memastikan tidak terjadi *overlap* antara pemesanan kelas dalam waktu dan tempat yang sama. Dengan demikian, sistem dapat menjaga keakuratan jadwal pemakaian kelas dan mencegah konflik pemesanan.

P-ISSN : 2089-3353
E-ISSN : 2808-9162



Gambar 3. Pemesanan Kelas Ditolak

Gambar 3 menunjukkan penggunaan data yang mencegah *overlapping* pemakaian kelas menunjukkan hasil pengujian di mana sistem menolak pemesanan yang memiliki waktu bentrok dengan pemesanan lain pada ruang yang sama. Dengan mekanisme ini, jadwal pemakaian kelas menjadi teratur dan bebas konflik.

3.1.4 *Data Sharing*

Tahap *data sharing* diimplementasikan dengan menyediakan akses terprogram ke data arsip bagi pengguna yang berwenang. Tujuannya adalah agar data historis tetap dapat dimanfaatkan untuk keperluan administratif atau akademik tanpa harus mengakses database secara langsung. Mekanisme ini diwujudkan melalui pembangunan Antarmuka Pemrograman Aplikasi (API) yang aman dan terdokumentasi.

Untuk menyediakan akses yang fleksibel dan terukur, diimplementasikan sebuah *endpoint* API dinamis dengan arsitektur *RESTful*. Pendekatan ini dipilih karena kemampuannya menangani permintaan untuk berbagai jenis data arsip melalui satu struktur URL yang konsisten. Struktur *endpoint* yang digunakan Adalah:

GET /api/archive/{tableName}

Parameter {tableName} bersifat dinamis dan dapat diisi dengan nama tabel arsip yang valid, yaitu:

- pesan-archived
- matkulganti-archived
- users-archived

Implementasi ini menggunakan satu route dinamis pada Laravel yang diarahkan ke sebuah metode tunggal di dalam *ArchiveController*. Metode tersebut dilengkapi dengan mekanisme keamanan untuk memvalidasi nama tabel yang diminta, memastikan hanya tabel yang diizinkan yang dapat diakses. Desain ini secara signifikan meningkatkan skalabilitas, karena penambahan tabel arsip baru di masa depan hanya memerlukan pembaruan pada daftar tabel yang diizinkan tanpa mengubah logika inti dari *controller* atau *route*.

Potongan Kode routes/api.php

```
<?php

use App\Http\Controllers\Api\ArchiveController;
use Illuminate\Support\Facades\Route;

Route::get('/archive/{tableName}', [ArchiveController::class, 'getArchiveData']);
```

Potongan Kode ArchiveController.php

```
public function getArchiveData(Request $request, string $tableName)
    {
        // 1. Keamanan: Buat daftar tabel yang diizinkan untuk diakses
        $allowedTables = [
            'pesan-archived',
            'matkulganti-archived',
            'users-archived'
        ];
…
```

Pada potongan kode routes/api.php dan *ArchiveController.php* di atas menampilkan implementasi route dinamis pada file routes/api.php dan metode *getArchiveData* pada *ArchiveController* yang menangani permintaan secara fleksibel dan aman.

Pertama, *pengujian Endpoint* dan format data JSON

Sebelum diintegrasikan dengan aplikasi lain, *endpoint* API diuji secara langsung melalui peramban *web* (*browser*) untuk memvalidasi fungsionalitas dan format *respons*. Pengujian dilakukan dengan mengakses URL lengkap, termasuk parameter *start_date* dan *end_date*.

Contoh URL pengujian:

http://127.0.0.1:8000/api/archive/pesan-archived?start_date=2025-10-01&end_date=2025-10-31

Hasil pengujian yang berhasil mengembalikan *respons* data dalam format JSON (*JavaScript Object Notation*). Format ini dipilih karena merupakan standar industri untuk pertukaran data antar sistem. Sifatnya yang ringan, terstruktur, dan mudah dibaca oleh mesin membuatnya ideal untuk dikonsumsi oleh berbagai macam aplikasi klien, termasuk skrip otomatisasi.

Data JSON Hasil API


```
[
    {
        "id": 2,
        "kelas_id": 2,
        "user_id": 1,
        "tanggal_pesan": "2025-10-05",
        "jam_pesan": "15:00:00",
        "jamkeluar": "16:00:00",
        "kegiatan": "Rapat HIMA ",
        "status": "selesai",
        "user_class": "SI-4D",
        "created_at": "2025-10-05 12:00:00",
        "updated_at": "2025-10-05 12:00:00",
        "archived_at": "2025-10-10 08:00:00"
    }
]
```


Pada Data JSON Hasil API di atas menunjukkan *respons* yang berhasil dari API saat diakses melalui *browser*, menampilkan data arsip dari tabel pesan-*archived* dalam format JSON yang terstruktur. Untuk menjaga agar akses terhadap informasi API dan data arsip tetap aman, sistem dilengkapi dengan *middleware autentikasi* (auth) bawaan Laravel.

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Halaman panduan API (/panduan-api) ditempatkan dalam grup *route* yang dilindungi autentikasi, sehingga hanya pengguna yang telah login yang dapat melihat dokumentasi teknis dan mengunduh skrip. Jika pengguna yang tidak memiliki sesi aktif mencoba mengakses halaman tersebut, sistem secara otomatis mengarahkan mereka ke halaman *login*. Mekanisme ini memastikan bahwa hanya pihak berwenang yang dapat memanfaatkan fasilitas *data sharing,* tanpa mengorbankan keamanan dan integritas data arsip.

Kedua, otomatisasi pengunduhan arsip menggunakan *Skrip Python*

Untuk mempermudah pengguna dalam mengunduh dan mengolah data arsip, sebuah *skrip Python* disediakan. Skrip ini berfungsi sebagai klien API yang mengotomatiskan seluruh proses: mulai dari mengirim permintaan, menerima data JSON, hingga mengubahnya menjadi format .CSV yang mudah dibaca dan dapat diolah lebih lanjut menggunakan aplikasi seperti *Microsoft Excel* atau *Google Sheets*. Pengguna hanya perlu melakukan konfigurasi sederhana pada skrip:

1. Menentukan nama tabel yang akan diunduh pada variabel NAMA_TABEL.

2. Menyesuaikan rentang tanggal pada variabel *start* dan *end*.

Skript kemudian secara otomatis membangun *URL API* yang sesuai, mengirimkan permintaan *GET*, memparsing respons *JSON*, dan menuliskan hasilnya ke dalam sebuah file .csv yang namanya dibuat secara dinamis berdasarkan nama tabel dan rentang tanggal. Implementasi ini secara efektif memberdayakan pengguna non-teknis untuk dapat menarik data historis secara mandiri tanpa memerlukan akses langsung ke *database*.

Program Jurnal

```
NAMA_TABEL = 'pesan-archived'

start = '2025-10-01'
end = '2025-10-31'

base_url = 'http://127.0.0.1:8000'
```

Potongan kode di atas menampilkan bagian konfigurasi dan logika utama dari skrip Python, di mana pengguna dapat dengan mudah mengubah NAMA_TABEL untuk menargetkan data arsip yang berbeda.

Tabel 4. Hasil Eksekusi Skrip Python dan File CSV yang Dihasilkan

id	kelas_id	user_id	tanggal pesan	hari	Jam mulai	Jam akhir	kegiatan
2	2	2	10/5/2025	Jumat	13:00	15:00	Rapat Himpunan Mahasiswa
5	5	4	10/1/2025	Selasa	14:00	16:00	Diskusi Proyek Akhir
6	6	2	10/7/2025	Senin	8:00	9:30	Kelas Regular Sistem Operasi

Pada Tabel 4 di atas memperlihatkan *output* pada terminal setelah skrip berhasil dijalankan dan contoh file .csv yang dihasilkan, menunjukkan data arsip yang telah terstruktur dalam format tabel

3.1.5 *Data Archiving*

Proses pengarsipan dilakukan secara otomatis setiap minggu. Data yang sudah melewati masa aktif akan dipindahkan dari database utama ke *database* arsip (*cold storage*). Pemindahan ini juga mencatat waktu pengarsipan melalui kolom *archived_at,* sebagai penanda bahwa data telah dipindahkan untuk penyimpanan jangka menengah.

Tabel 5. Tabel Arsip matkulganti-archived

id	matkul_id	kelas_id	Tanggal ganti	hari	Jam mulai	Jam akhir	status
55	24	6	2024-10-21	senin	10:00	11:00	Dibook
56	36	2	2025-10-09	kamis	8:00	10:00	Dibook
57	38	2	2025-10-07	selasa	13:00	15:00	Dibook

Tabel 5 yang menampilkan tabel arsip *matkulganti_archived* memperlihatkan data yang telah dipindahkan ke database arsip dengan tambahan kolom archived_at sebagai penanda waktu pemindahan.

Pada tujuan pengotomatisasian pada projek ini akan dilakukan dengan command *schedule Laravel* yang akan melakukan pengarsipan secara otomatis pada hari minggu jam 8 pagi.

Potongan Kode Pengarsipan

```
foreach ($activeRecords as $record) {
try {
$archivedData = $record->toArray();
$archivedData['archived_at'] = $archivedAt;
```

Potongan kode pengarsipan diatas menunjukkan Potongan kode pengarsipan yang mengimplementasi scheduler pada Laravel yang mengeksekusi proses ini setiap hari Minggu pukul 08.00. Kode tersebut melakukan penyalinan data dari *hot storage* ke c*old storage,* lalu menghapus data aktif setelah pemindahan berhasil.

3.1.6 *Data Retention*

Kebijakan retensi data diterapkan dengan cara menyimpan data arsip selama satu semester penuh. Data yang berada di arsip tetap dapat diakses dan diunduh melalui *API* untuk kepentingan akademik atau administratif. Setelah masa retensi berakhir, data akan dijadwalkan untuk dihapus secara otomatis sesuai kebijakan *retention*.

Potongan Kode Otomatisasi Pengarsipan dan Penghapusan

P-ISSN : 2089-3353
E-ISSN : 2808-9162

```
Schedule::command('dlm:archivebookings')->w
eeklyOn(0, '8:00');

Schedule::command('dlm:deletearchived')->da
ily();
```

Potongan kode diatas adalah potongan kode otomatisasi pengarsipan dan penghapusan data yang sesuai pada kebijakan *retention* bahwa pengarsipan akan dilakukan pada hari minggu yang di dalam kode diwakilkan oleh angka nol (0), dan juga akan memeriksa apakah ada data yang cocok untuk dihapus sesuai kebijakan Retention.

3.1.7 *Data Deletion*

Pada tahap *deletion* akan dilakukan penghapusan data pada *cold storage*. Pada *retention policy* tertulis bahwa akan dihapus setelah 1 semester dan 30 hari berlalu, maka diasumsukan bahwa 1 semester terdapat 180 hari, dan ditambahkan grace period selama 30 hari, jadi total data akan dihapus setelah data berumur 210 hari.

Potongan Program Penghapusan Data Arsip

```
$deletedCount =
$archiveModelClass::where('archived_at',
'<', $cutoffDate)->delete();
```

Potongan kode diatas berisi potongan kode penghapusan data arsip memperlihatkan potongan kode yang memeriksa usia data berdasarkan kolom archived_at dan menghapusnya secara permanen dari database arsip apabila sudah melewati batas waktu tersebut

3.2. Analisis Performa dan Skalabilitas Sistem

Implementasi *Data Lifecycle Management* (DLM) pada sistem ini tidak hanya bertujuan untuk merapikan penyimpanan, tetapi secara fundamental menjaga performa basis data operasional. Dengan mengarsipkan data yang ada di basis data aktif (*hotsStorage*) ke basis data arsip (*cold storage*), kinerja basis data aktif dapat dijaga keoptimalannya dengan tidak membiarkannya menumpuk.

Pengujian performa dilakukan melalui metode *load testing* pada lingkungan lokal (PHP 8.5, MySQL) untuk membandingkan latensi sistem dengan dan tanpa DLM. Data uji dibangkitkan menggunakan algoritma *Collision-Free Strategy* hingga 100.000 baris untuk mensimulasikan beban nyata tanpa melanggar integritas data.

Untuk membuktikan hal tersebut, dilakukan pengukuran waktu eksekusi kueri (*query execution time*) pada berbagai volume data. Hasil pengujian performa disajikan pada Tabel 3 berikut ini:

Tabel 6. Hasil Perbandingan Latensi Basis Data (*Benchmark Results*)

Volume Data	Rata-rata *Read Latency*	Analisis Performa
0 - 5.000 baris	~14.8 ms	Optimal
55.000 baris	~19.6 ms	Melambat 32%
100.000 baris	~21.8 ms	Melambat 47%

Berdasarkan Tabel 6, terlihat korelasi linier antara penumpukan data dengan penurunan kinerja. Pada skenario *stress test* 2, di mana data dibiarkan menumpuk hingga 100.000 baris, waktu respons pembacaan (*read latency*) melambat hingga ~21.8 ms, atau mengalami degradasi performa sebesar 47% dibandingkan kondisi ideal. Sebaliknya, pada skenario Ideal di mana mekanisme *auto-archiving* DLM aktif, volume data pada tabel operasional berhasil ditekan (di bawah 5.000 baris), sehingga sistem mampu mempertahankan latensi stabil di kisaran ~14.8 ms. Hal ini membuktikan bahwa pemisahan data aktif dan arsip secara efektif mengisolasi beban kerja sistem, menjamin proses pengecekan jadwal (validasi *overlapping*) tetap instan meskipun sistem telah berjalan selama bertahun-tahun.

Selain performa, penerapan arsitektur *dual connection* memastikan bahwa proses pengarsipan yang berjalan di latar belakang tidak mengganggu aktivitas operasional pengguna (*non-blocking*), sehingga keandalan (*reliability*) sistem tetap terjaga selama proses pemeliharaan data berlangsung.

Pada hasil implementasi di atas, terlihat bahwa pada tahap *data archiving* maupun *data deletion* tidak dilakukan proses terhadap akun pengguna (Komting). Keputusan ini diambil berdasarkan hasil pengamatan yang menunjukkan bahwa pergantian Komting di setiap kelas tidak terjadi secara periodik setiap semester. Penerapan penghapusan atau penonaktifan otomatis justru berpotensi menimbulkan kendala administratif dan menambah beban bagi Komting yang masih aktif. Oleh karena itu, sistem dirancang agar akun Komting tetap aktif hingga terdapat pergantian resmi yang diinformasikan kepada admin. Apabila terjadi perubahan Komting, maka tanggung jawab atas akun tersebut diserahkan secara langsung kepada Komting baru yang ditunjuk, atau dapat dilakukan penonaktifan manual oleh admin apabila memang diperlukan. Pendekatan ini dinilai lebih efisien dan sesuai dengan kebutuhan operasional di lingkungan akademik

4. Kesimpulan

Berdasarkan hasil implementasi yang telah dilakukan, dapat disimpulkan bahwa penelitian ini berhasil mengimplementasikan konsep *Data Lifecycle Management* (DLM) pada sistem *System Integrated Management* (SIM) untuk pemesanan kelas berbasis web. Setiap tahapan DLM mulai dari *data creation*, *Storage*, *usage*, *sharing*, *archiving*, *retention*, hingga *Deletion* telah diterapkan sesuai dengan kerangka manajemen siklus hidup data. Proses implementasi dilakukan dengan memanfaatkan *framework Laravel* 11 dan *database* MySQL melalui arsitektur *dual connection*, yang memisahkan basis data aktif (*hot storage)* dan arsip (*cold storage*).

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Penerapan otomatisasi dengan Laravel *scheduler* memungkinkan proses pengarsipan dan penghapusan data berjalan terjadwal tanpa intervensi manual, sementara kebijakan *retention policy* mengatur masa simpan data (1 semester + 30 hari) agar sesuai dengan kebutuhan operasional akademik. Tahap Data Sharing juga berhasil diwujudkan melalui pengembangan endpoint API publik yang memungkinkan akses data arsip secara terstruktur dan aman. Dengan demikian, penelitian ini telah memenuhi tujuannya, yaitu menerapkan prinsip-prinsip DLM dalam sistem pemesanan kelas untuk mendukung pengelolaan data yang efisien, teratur, dan berkelanjutan di lingkungan akademik. Sebagai pengembangan lebih lanjut, sistem ini dapat ditingkatkan dengan menambahkan fitur pemantauan siklus data secara *real-time* atau integrasi notifikasi otomatis agar pengelolaan data arsip menjadi lebih adaptif dan informatif, dan juga pengembangan mekanisme pemulihan data (*data restoration*) yang memungkinkan data dari *cold storage* dikembalikan sementara ke *hot storage* untuk keperluan audit mendalam.

Daftar Rujukan

[1] K. J. Lie, Z. Rusdi, and D. A. Haris, "Website based scheduling system for the Living Word Community," *Int. J. Appl. Sci. Technol. Eng.*, vol. 1, no. 4, pp. 1438–1447, 2023.

[2] A. P. Tindi and O. Lumasuge, "Sistem Penjadwalan dan Notifikasi Petugas Ibadah Kelompok Berbasis Web dengan API Pesan Instan," *JUMINTAL J. Manaj. Inform. dan Bisnis Digit.*, vol. 4, no. 1, pp. 38–49, 2025, doi: 10.55123/jumintal.v4i1.5177.

[3] A. O. Yorizka, H. Y. Putra, and F. Akbar, "Implementasi Sistem Automasi Berbasis Web pada Proses Check-In dan Check-Out Asrama Universitas Andalas," *J. Nas. Teknol. dan Sist. Inf.*, vol. 7, no. 2, pp. 90–98, 2021, doi: 10.25077/teknosi.v7i2.2021.90-98.

[4] R. Ramadhani, "Designing a Web-Based Archive Management Application Using the Laravel Framework: A Case Study on a Recreational Park," *Inf. Technol. Syst.*, vol. 2, no. 1, pp. 16–24, 2024, doi: 10.58777/its.v2i1.300.

[5] I. Kabashkin, V. Perekrestov, T. Tyncherov, L. Shoshin, and V. Susanin, "Framework for Integration of Health Monitoring Systems in Life Cycle Management for Aviation Sustainability and Cost Efficiency," *Sustain.*, vol. 16, no. 14, 2024, doi: 10.3390/su16146154.

[6] H. Lian, T. Mo, and C. Zhang, "Intelligent Data Lifecycle Management in Cloud Storage : An AI- driven Approach to Optimize Cost and Performance," *Acad. Nexus J.*, vol. 3, no. 3, pp. 1–20, 2024.

[7] N. Janssen, T. Ilayperuma, J. Jayasinghe, F. Bukhsh, and M. Daneva, "The evolution of data storage architectures: examining the secure value of the Data Lakehouse," *J. Data, Inf. Manag.*, vol. 6, no. 4, pp. 309–334, 2024, doi: 10.1007/s42488-024-00132-1.

[8] W. T. Ramdan, D. Yusuf, and P. Purwantoro, "Rancang Bangun Portal Document Management System Berbasis Web Untuk Digitalisasi Arsip Menggunakan Frawework Laravel," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 4, pp. 8091–8097, 2024, doi: 10.36040/jati.v8i4.10499.

[9] C. Liu, A. Pavlenko, M. Interlandi, and B. Haynes, "Data formats in analytical DBMSs: performance trade-offs and future directions," *VLDB J.*, vol. 34, no. 3, 2025, doi: 10.1007/s00778-025-00911-1.

[10] D. Kiswanto, F. Ramadhani, N. M. Surbakti, and N. A. Nasution, "Pengembangan dan Implementasi Sistem Deteksi Serangan DDoS Berbasis Algoritma Random Forest," *Bull. Inf. Technol.*, vol. 6, no. 3, 2025, doi: 10.47065/bit.v5i2.2203.

[11] M. N. M. B. Sipahutar, A. L. P, and S. P. Sipayung, "Security Analysis of Data Storage in Cloud-Based Digital Archive Management Systems," *J. Adv. Comput. Knowl. Algorithms*, vol. 2, no. 3, pp. 80–83, 2025, doi: 10.29103/jacka.v2i3.22436.

[12] N. Chukwurah, A. B. Ige, and V. I. Adebayo, "Managing Data Lifecycle Effectively: Best Practices for Data Retention and Archival Processes," *Int. J. Eng. Res. Dev.*, vol. 20, no. 7, pp. 453–461, 2024, [Online]. Available: www.ijerd.com

[13] C. Sharma and A. Vaid, "Leveraging SAP Information Lifecycle Management (ILM): Latest insights and applications," *Int. J. Eng. Manag. Humanit.*, vol. 5, no. 6, pp. 167–173, 2024.

[14] Y. Zhang, H. Zhang, and E. Feng, "Cost-Effective Data Lifecycle Management Strategies for Big Data in Hybrid Cloud Environments," *Acad. Nexus J.*, vol. 3, no. 2, pp. 1–22, 2024.

[15] A. F. Md Ajis, S. Zakaria, and A. R. Ahmad, "Modelling Dark Data Lifecycle Management: A Malaysian Big Data Experience," *Int. J. Acad. Res. Bus. Soc. Sci.*, vol. 12, no. 3, pp. 328–344, 2022, doi: 10.6007/ijarbss/v12-i3/12363.

[16] S. I. H. Shah, V. Peristeras, and I. Magnisalis, "DaLiF: a data lifecycle framework for data-driven governments," *J. Big Data*, vol. 8, no. 1, 2021, doi: 10.1186/s40537-021-00481-3.