Implementasi Logika Time-Aware dan State Management pada Modul Kalender Akademik Berbasis Web untuk Otomatisasi Status Penjadwalan Kursus (Studi Kasus: Platform Kelas Bisa)

Mochammad Satria Hendra Putra*[1], **Ramadhan Renaldy**[2]
[1]Informatika, Universitas PGRI Semarang, Kota Semarang, satriahendra117@gmail.com
[2]Informatika, Universitas PGRI Semarang, Kota Semarang, ramadhanrenaldy@upgris.ac.id

Abstract.

The rapid development of information technology in the education sector demands a dynamic and accurate course schedule management system to optimize user experience. This study discusses the implementation process of time-aware logic and state management in a web-based academic calendar module as the main pillar of scheduling status automation on the Kelas Bisa platform at Vrasmedia IT Solution. Beyond this primary focus, the development also expands the platform's functionality by building a Main Admin Dashboard for monitoring web growth metrics, a Merchandise Dashboard integrated with a Midtrans payment gateway simulation, and modules for managing articles and testimonials. The development method focused on the frontend side using the React.js framework, Tailwind CSS, and React Hooks (useState and useEffect) to reactively process local system time variables, along with system architecture modeling using Use Case, Class, and Sequence Diagrams. The results demonstrate that the system successfully achieves real-time synchronization to automatically execute access control functions (canJoin) and missed status detection (isMissed) without requiring page reloads. The integration of the admin dashboard and payment system was also successfully simulated securely in a sandbox environment. In conclusion, the application of time-aware logic and robust state management has proven effective in increasing system automation, minimizing manual schedule management errors, and presenting an informative interface for users of the Kelas Bisa platform.

Keywords: Time-Aware, State Management, Academic Calendar, React.js, Kelas Bisa Platform

Abstrak

Pesatnya perkembangan teknologi informasi di sektor pendidikan menuntut sistem manajemen jadwal kursus yang dinamis untuk mengoptimalkan pengalaman pengguna. Penelitian ini membahas implementasi logika *time-aware* dan *state management* pada modul kalender akademik berbasis web untuk otomatisasi status penjadwalan pada platform Kelas Bisa di Vrasmedia IT Solution. Selain itu, proyek ini juga mengembangkan Dasbor Utama Admin, Dasbor Merchandise dengan simulasi *payment gateway* Midtrans, serta modul pengelolaan artikel dan testimoni. Metode pengembangan difokuskan pada sisi *frontend* menggunakan React.js, Tailwind CSS, dan React Hooks (useState dan useEffect), serta pemodelan sistem menggunakan *Use Case*, *Class*, dan *Sequence Diagram*. Hasil penelitian menunjukkan sistem berhasil melakukan sinkronisasi waktu nyata (*real-time*) untuk menjalankan fungsi kontrol akses (canJoin) dan deteksi status terlewat (isMissed) secara otomatis tanpa pemuatan ulang halaman. Kesimpulannya, penerapan logika *time-aware* dan manajemen status ini efektif meningkatkan otomasi sistem, meminimalisir kesalahan manual, serta menyajikan antarmuka yang informatif.

Kata Kunci: *Time-Aware*, *State Management*, Kalender Akademik, React.js, Platform Kelas Bisa

1. Pendahuluan

Efisiensi manajemen waktu dalam *platform* pembelajaran daring *(e-learning*) saat ini menjadi salah satu fokus utama dalam optimalisasi pengalaman pengguna di sektor pendidikan digital. Seiring meningkatnya volume kelas interaktif, ketidakakuratan sinkronisasi jadwal visual dan keterlambatan pembaruan status akses sesi akademik sering kali memicu hambatan operasional yang signifikan [1]. Masalah ini umumnya terjadi akibat kegagalan sistem dalam mendeteksi perubahan waktu nyata *(real-time*), yang berdampak pada ketidakakuratan fungsi kontrol akses masuk kelas serta keterlambatan deteksi otomatis untuk sesi yang telah terlewat [1]. Ketergantungan pada pemuatan ulang halaman (*page reload)* secara manual atau mekanisme penandaan waktu yang tidak sinkron antara perangkat lokal pengguna dan server terbukti menurunkan keterlibatan pengguna serta meningkatkan beban administratif dalam pengelolaan jadwal berskala besar.

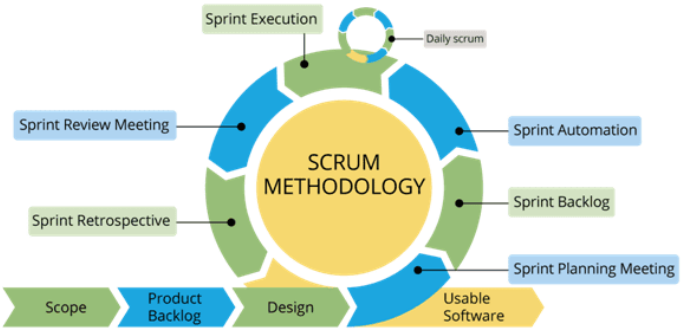
Berbagai upaya telah dilakukan oleh para peneliti terdahulu untuk mengatasi kendala sinkronisasi temporal ini. Pendekatan konvensional umumnya mengandalkan komunikasi berkala berbasis permintaan server (*polling)* atau pemanfaatan pustaka komponen kalender pihak ketiga yang bersifat statis [2]. Beberapa studi terbaru mulai bergeser ke arah arsitektur berbasis peristiwa *(event-driven architecture)* menggunakan *web socket* untuk memastikan pembaruan data dua arah secara instan. Meskipun metode tersebut berhasil meningkatkan kecepatan pembaruan di sisi basis data, sebagian besar solusi yang dilaporkan masih berfokus pada optimasi komputasi di sisi backend. Akibatnya, beban pemrosesan logika temporal yang dinamis sering kali mengabaikan efisiensi di sisi klien (*frontend*), sehingga memicu konsumsi memori peramban yang tinggi ketika harus melacak perubahan variabel waktu yang terjadi setiap detik [3].

Guna mengatasi keterbatasan tersebut, artikel ini menawarkan sebuah pendekatan integrasi antara logika time-aware dan komponen state management yang tangguh langsung di sisi *frontend* pada modul kalender akademik *platform* Kelas Bisa di Vrasmedia IT Solution [1]. Melalui optimalisasi *framework* React.js dan pemanfaatan *React Hooks* (*useState dan useEffect*), variabel waktu lokal sistem dapat diproses secara reaktif untuk menjalankan fungsi kontrol akses (*canJoin*) dan deteksi status terlewat (*isMissed*) secara otomatis [4]. Pendekatan ini memindahkan beban komputasi pelacakan waktu dari server ke klien secara aman, sehingga pembaruan status visual dapat terjadi secara instan tanpa memerlukan request berulang ke server. Sebagai pelengkap ekosistem, sistem ini juga diperluas dengan integrasi Dasbor Utama Admin untuk pemantauan metrik perkembangan web serta Dasbor *Merchandise* yang terhubung dengan simulasi *payment gateway Midtrans* pada lingkungan *sandbox* [5].

Penelitian ini bertujuan untuk menghasilkan arsitektur sistem kalender akademik yang memiliki tingkat otomasi tinggi, akurat, dan responsif dalam meminimalisir kesalahan pengelolaan jadwal manual. Metode penelitian dilakukan melalui beberapa tahapan terstruktur, dimulai dari analisis kebutuhan dan pemodelan arsitektur sistem menggunakan *Use Case, Class,* dan *Sequence Diagram*. Selanjutnya, tahap implementasi difokuskan pada pengembangan antarmuka menggunakan React.js dan Tailwind CSS, yang diakhiri dengan pengujian fungsionalitas logika waktu dan simulasi transaksi pembayaran guna memastikan stabilitas performa sistem dalam lingkungan produksi [6].

2. Metode

Penelitian ini berfokus pada pengembangan sistem manajemen jadwal berbasis web yang dinamis melalui implementasi logika *time-aware* dan *state management* pada platform Kelas Bisa. Proses penelitian meliputi studi literatur, analisis kebutuhan, perancangan sistem menggunakan diagram arsitektur, implementasi *frontend*, dan pengujian fungsionalitas sistem [1]. Pengembangan perangkat lunak dilakukan menggunakan metode *Agile Development* dengan kerangka kerja *Scrum* karena memiliki tahapan yang adaptif, kolaboratif, dan fleksibel terhadap perubahan kebutuhan sistem selama proses pengembangan berlangsung [7].



Gambar 2.1 Siklus Kerangka Kerja Scrum dalam Pengembangan Sistem [8]

Metode *Agile Scrum* digunakan sebagai pendekatan dalam pengembangan sistem pada penelitian ini. Model *Scrum* menerapkan tahapan pengembangan perangkat lunak secara berulang (*iterative*) dalam bentuk siklus pendek yang disebut *Sprint*. Pendekatan ini dipilih karena mampu menghasilkan fitur produk secara bertahap dan cepat, serta memungkinkan evaluasi berkala bersama mentor industri guna meminimalisir kesalahan logika penjadwalan sejak dini [7].

a. Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan pengguna dan menentukan spesifikasi modul kalender akademik serta dasbor yang akan dikembangkan pada platform Kelas Bisa. Analisis kebutuhan merupakan proses untuk memahami kebutuhan operasional admin dan pengguna serta mendefinisikan fungsi otomatisasi status waktu yang harus disediakan oleh sistem [9]. Hasil analisis menunjukkan bahwa sistem harus mampu mendukung otomatisasi kontrol akses masuk kelas (*canJoin*), deteksi otomatis status kelas yang terlewat (*isMissed*) secara waktu nyata (*real-time*) tanpa memuat ulang halaman, Dasbor Utama Admin untuk metrik perkembangan web, Dasbor *Merchandise* untuk transaksi produk, serta modul pengelolaan artikel dan testimoni.

b. Perancangan Sistem

Tahap perancangan sistem dilakukan untuk menggambarkan struktur data dan alur kerja logika komponen yang akan dikembangkan. Perancangan menggunakan *Unified Modeling Language* (UML) yang terdiri dari *use case diagram* untuk memetakan hak akses, *class diagram* untuk memodelkan komponen *state management*, dan *sequence diagram* untuk menggambarkan interaksi reaktif antara variabel waktu lokal klien dengan antarmuka sistem [10]. Selain itu, dilakukan perancangan antarmuka pengguna (*user interface*) menggunakan Tailwind CSS sebagai acuan visual dalam proses implementasi [11].

c. Implementasi Sistem

Tahap implementasi dilakukan dengan menerjemahkan hasil perancangan ke dalam bentuk kode aplikasi pada sisi *frontend* menggunakan *framework* React.js dan manajemen status reaktif via *React Hooks (useState dan useEffect).* React.js dipilih karena menerapkan arsitektur berbasis komponen yang mendukung pembuatan antarmuka web secara terstruktur, efisien, dan responsif dalam memproses variabel waktu lokal sistem setiap detik [12]. Implementasi difokuskan pada sinkronisasi logika *time-aware* pada modul kalender akademik, pembuatan Dasbor Utama Admin, serta integrasi *Midtrans Sandbox* sebagai simulasi *payment gateway* pada Dasbor *Merchandise* untuk mendukung simulasi transaksi pembayaran digital yang aman [13].

d. Pengujian Sistem

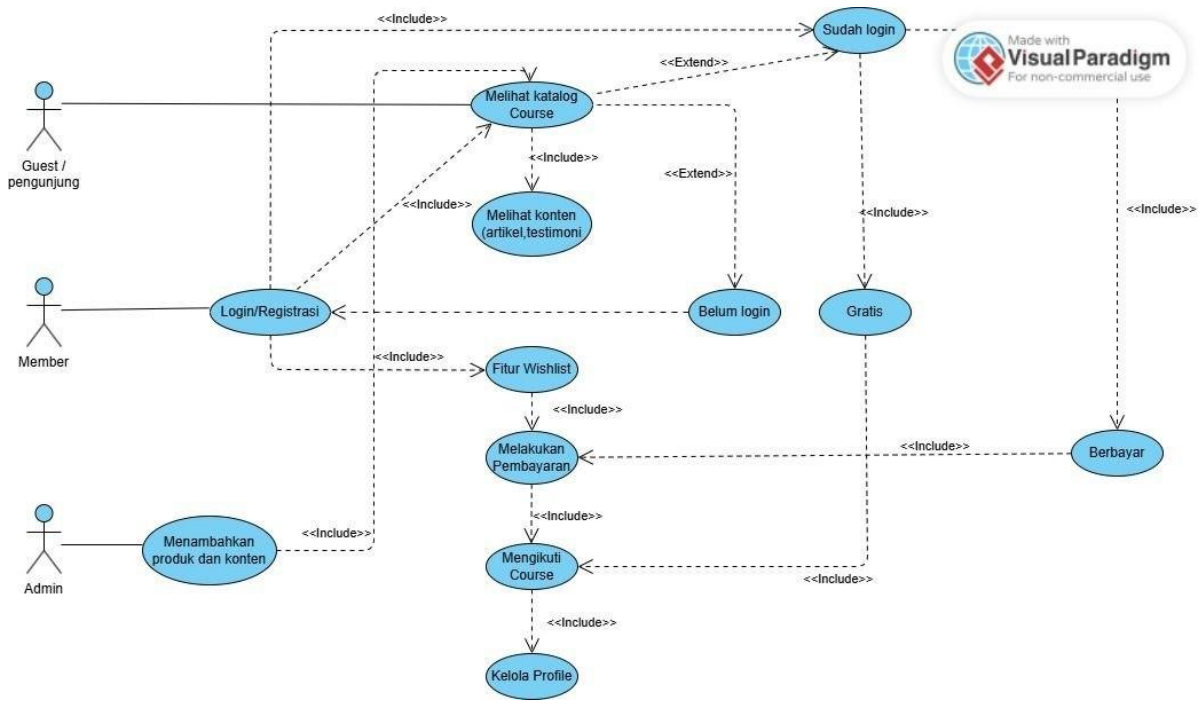
Tahap pengujian dilakukan untuk memastikan bahwa seluruh fungsi otomasi dan logika temporal sistem berjalan sesuai dengan spesifikasi yang telah ditentukan. Pengujian sistem merupakan proses evaluasi perangkat lunak yang bertujuan untuk memverifikasi kesesuaian fungsi otomatisasi status kalender terhadap waktu lokal perangkat pengguna serta keandalan transaksi pembayaran [14]. Pengujian dilakukan bersama mentor proyek dengan memeriksa fungsi utama sistem, meliputi keakuratan eksekusi fungsi *canJoin* dan *isMissed* secara otomatis, visualisasi grafik pada dasbor admin, serta validasi status *respons* transaksi *sandbox Midtrans (settlement, pending, deny)* [15].

3. Hasil dan Pembahasan

3.1. Penyajian Hasil

Platform manajemen jadwal kursus berbasis web yang dikembangkan dirancang untuk mendukung otomatisasi penjadwalan kelas dan pemantauan administrasi secara terintegrasi. Sistem melibatkan tiga aktor utama, yaitu admin, mentor, dan siswa, serta satu sistem eksternal berupa *gateway* pembayaran. Admin bertanggung jawab terhadap pemantauan metrik perkembangan web, pengelolaan artikel, testimoni, dan manajemen produk merchandise. Mentor berperan dalam mengelola kalender akademik dan membuat jadwal kelas kursus. Sementara itu, siswa dapat mengakses modul kalender akademik untuk mengikuti kelas secara waktu nyata (*real-time*), melihat status sesi, serta melakukan simulasi pembelian produk pendukung kelas pada dasbor *merchandise*.

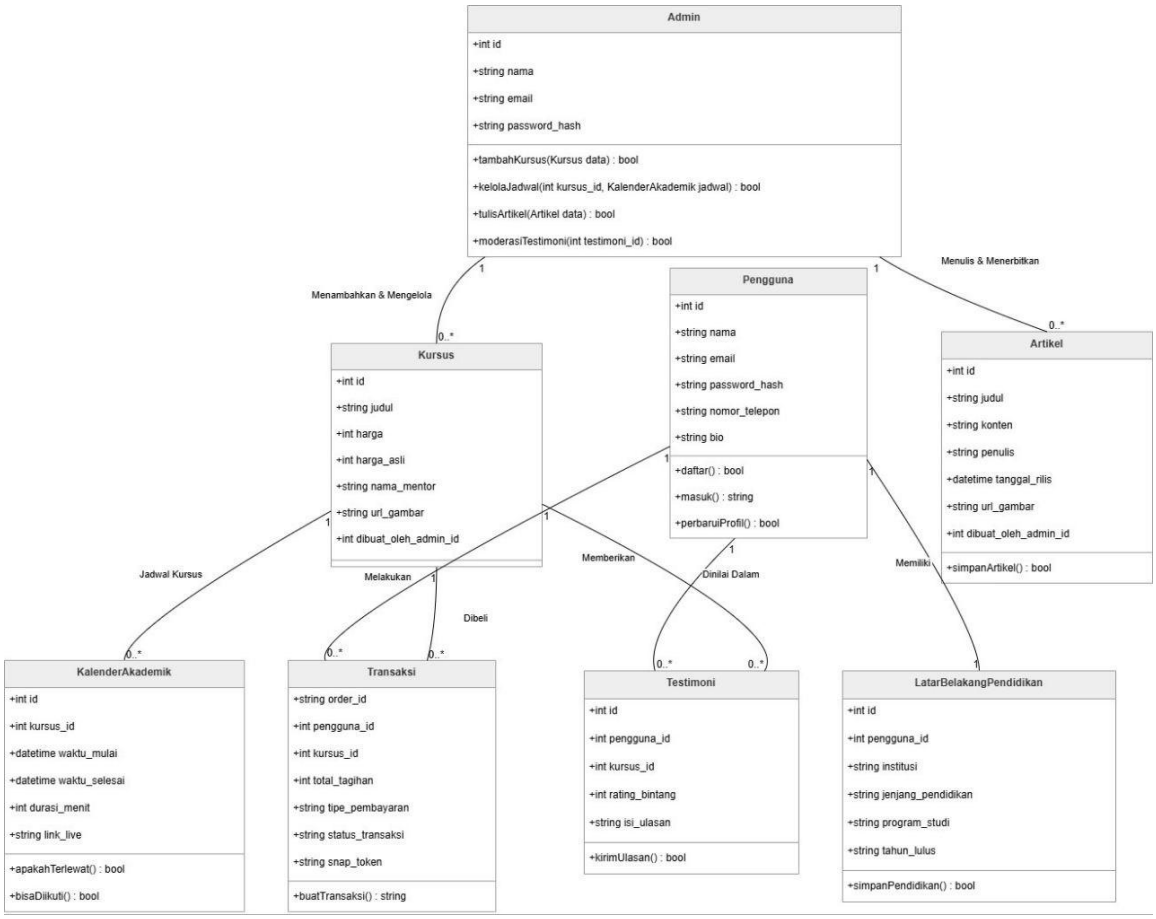
3.1.1 Use Case Diagram



Gambar 3.1 *Use Case* Diagram

Gambar 3.1 menggambarkan interaksi antara aktor dengan sistem yang dikembangkan pada platform Kelas Bisa. Sistem melibatkan empat aktor utama, yaitu admin, siswa, mentor, dan Midtrans Sandbox sebagai layanan simulasi pembayaran digital. Admin memiliki akses ke Dasbor Utama Admin untuk memantau metrik perkembangan web, serta mengelola modul artikel, testimoni, dan data katalog produk merchandise. Mentor memiliki akses penuh untuk mengelola kalender akademik dan memperbarui parameter jadwal kursus. Siswa dapat mengakses visualisasi kalender akademik, memantau status jadwal, melakukan simulasi pembelian pada Dasbor Merchandise, dan melakukan proses *checkout*. Sementara itu, Midtrans Sandbox digunakan untuk memproses dan memvalidasi transaksi pembayaran digital yang dilakukan oleh siswa. Diagram ini menunjukkan keterhubungan antara pengguna dan fungsi-fungsi utama yang tersedia pada platform Kelas Bisa.

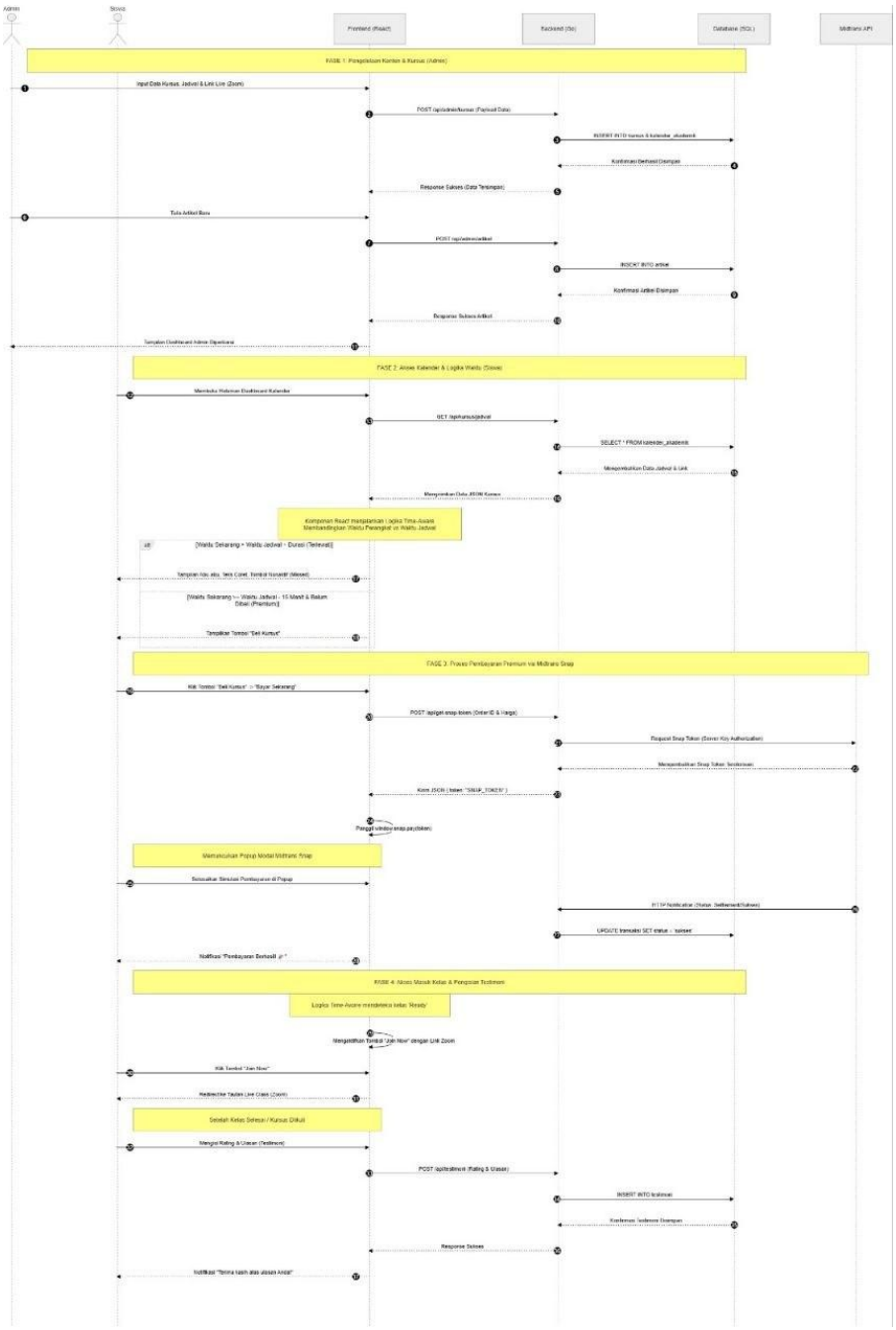
3.1.2 Class Diagram



Gambar 3.2 menggambarkan struktur data dan hubungan antar entitas yang digunakan dalam platform Kelas Bisa. Diagram terdiri atas beberapa kelas utama, yaitu User, Admin, Mentor, Student, AcademicCalendar, CourseSchedule, Article, Testimonial, Merchandise, dan MidtransPayment. Kelas User berperan sebagai pusat pengelolaan akun yang diturunkan kepada admin, mentor, dan siswa melalui hubungan pewarisan (*inheritance/generalization*). Kelas AcademicCalendar dan CourseSchedule dirancang dengan logika *time-aware* untuk mendukung fungsionalitas pelacakan waktu operasional kelas. Kelas Article dan Testimonial digunakan oleh admin untuk mengelola konten informasi halaman depan. Selain itu, kelas Merchandise dan MidtransPayment digunakan untuk mengelola data transaksi produk yang terintegrasi langsung dengan layanan API Midtrans Sandbox.

Secara struktural, kelas AcademicCalendar memiliki relasi asosiasi satu-ke-banyak (*one-to-many*) dengan CourseSchedule, yang memungkinkan pelacakan banyak sesi jadwal dalam satu kalender akademik secara teratur. Interaksi antara kelas Student, Merchandise, dan MidtransPayment juga dilengkapi dengan metode validasi *callback* otomatis untuk memastikan status transaksi langsung memperbarui hak akses visual pengguna. Hubungan antar kelas ini secara keseluruhan dirancang menggunakan prinsip enkapsulasi yang ketat untuk mendukung otomatisasi status penjadwalan, meminimalkan redundansi data temporal, serta memastikan pengelolaan administrasi berjalan secara dinamis dan terintegrasi dalam satu kesatuan sistem.

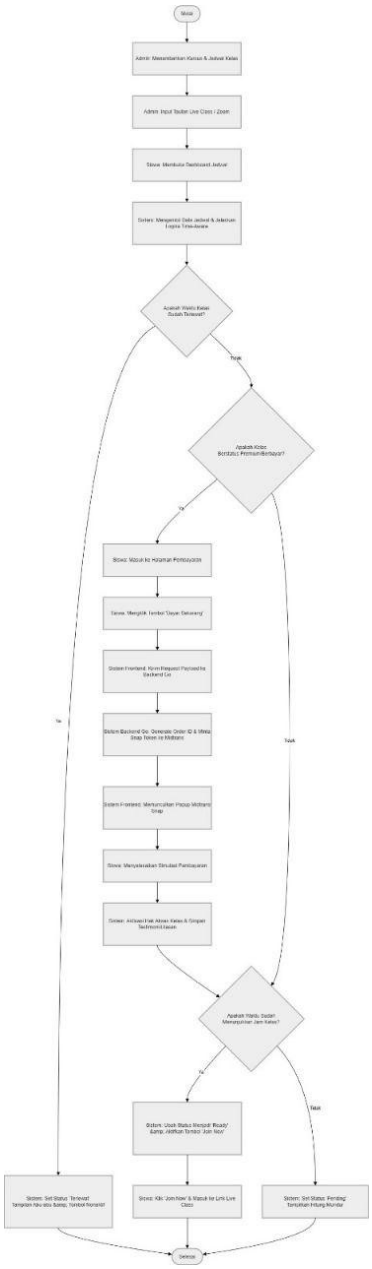
3.1.3 Sequence Diagram



Gambar 3.3 *Sequence* Diagram

Gambar 3.3 menggambarkan aliran pesan (*message*) dan urutan kronologis aktivitas antarkomponen dalam mengeksekusi logika *time-aware* secara *real-time*. Proses dimulai ketika aktor (siswa) mengakses antarmuka kalender akademik, di mana komponen *frontend* React.js memicu pengaktifan penanda waktu lokal melalui React Hooks (useEffect). Pesan internal dikirimkan secara sekuensial setiap detiknya untuk membandingkan stempel waktu (*timestamp*) sistem lokal dengan data jadwal kelas yang tersimpan pada objek CourseSchedule. Diagram ini menunjukkan bagaimana perubahan kondisi temporal langsung memicu pembaruan fungsi Boolean (canJoin atau isMissed) pada komponen *state management* tanpa melalui intervensi pemuatan ulang halaman (*page reload*). Alur sekuensial ini memastikan bahwa hak akses siswa dan visualisasi status jadwal tereksekusi secara sinkron, aman, dan responsif.

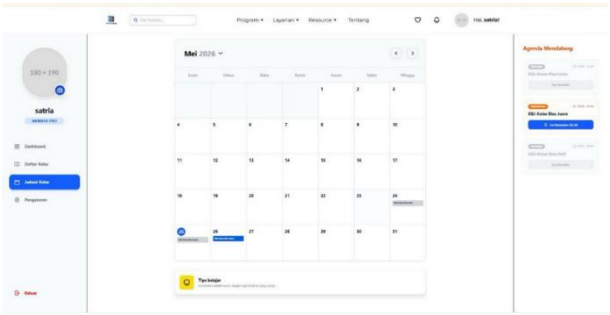
3.1.4 Activity Diagram



Gambar 3.4 Activity Diagram

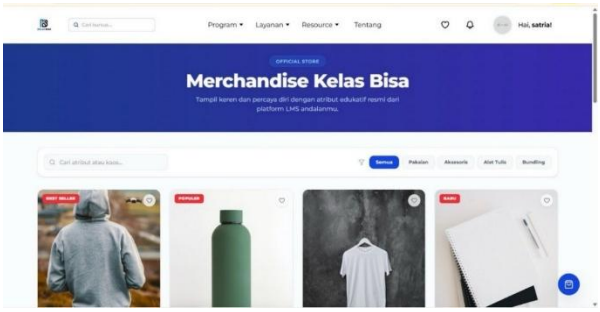
Gambar 3.2 menggambarkan alur aktivitas operasional (*workflow*) di dalam sistem ketika memproses penentuan waktu otomatis. Aktivitas dimulai secara paralel saat siswa membuka halaman kalender, di mana sistem secara otomatis melakukan inisialisasi pencocokan waktu lokal perangkat terhadap basis data jadwal yang ditetapkan oleh mentor. Diagram ini memperlihatkan percabangan kondisi (*decision node*) secara dinamis: jika waktu sekarang berada di dalam jendela pelaksanaan kursus, sistem akan mengaktifkan fungsi canJoin; namun jika waktu melampaui batas toleransi akhir sesi, alur sistem langsung diarahkan untuk mengeksekusi fungsi isMissed dan mengubah warna status menjadi nonaktif secara instan tanpa perlu memicu muat ulang halaman.

3.1.5 Implementasi Sistem



Gambar 3.5 Halaman Kalender Akademik Siswa

Gambar 3.6 Halaman Dasbor Utama Admin



Gambar 3.7 Halaman Dasbor Merchandise

Gambar 3.8 Halaman Pembayaran Sandbox Midtrans

Bagian implementasi sistem ini menyajikan wujud nyata dari seluruh rancangan arsitektur dan logika yang telah dibangun pada platform Kelas Bisa, mulai dari fungsionalitas waktu nyata hingga integrasi sistem pembayaran eksternal. Implementasi utama dari logika *time-aware* berbasis React Hooks (useState dan useEffect) ditunjukkan pada Gambar 3.5 (Halaman Kalender Akademik Siswa). Pada halaman ini, sistem bertindak sebagai *timer* aktif berskala milidetik yang secara konstan mensinkronisasikan waktu nyata perangkat pengguna dengan *timestamp* kelas dari *backend*. Perubahan kondisi ini dikelola secara ketat oleh komponen *state management*, di mana tombol masuk kelas akan otomatis aktif (canJoin) saat waktu dimulai dan beralih menjadi terlewat (isMissed) ketika durasi toleransi habis, seluruhnya terjadi di latar belakang tanpa memerlukan pemuatan ulang halaman (*page reload*).

Di sisi lain, untuk mendukung pengawasan operasional secara menyeluruh, Gambar 3.6 (Halaman Dasbor Utama Admin) menyediakan pusat kendali bagi admin yang dilengkapi dengan visualisasi data interaktif berupa grafik linier dan diagram batang. Dasbor ini menyajikan metrik perkembangan web secara *real-time*, seperti pertumbuhan pengguna aktif, akumulasi kelas, dan total transaksi yang ditarik secara dinamis dari basis data melalui kueri agregasi untuk membantu pengambilan keputusan administratif. Selanjutnya, platform ini juga menyediakan fitur pendukung pembelajaran yang diimplementasikan pada Gambar 3.7 (Halaman Dasbor Merchandise). Melalui tata letak berbasis kartu (*card layout*) yang dilengkapi fitur pencarian dan penyaringan, siswa dapat menjelajahi katalog produk, membuka modal rincian untuk meninjau spesifikasi barang, serta memasukkan item ke keranjang belanja sementara yang dipantau langsung oleh *state management*. Proses transaksi dari dasbor tersebut kemudian diintegrasikan secara penuh dengan API eksternal yang ditunjukkan pada Gambar 3.8 (Halaman Pembayaran Sandbox Midtrans). Begitu siswa melakukan *checkout*, sistem mengirimkan *payload* transaksi untuk memicu *Snap Popup* atau mengalihkan halaman ke lingkungan Sandbox ini. Di sini, siswa dapat mensimulasikan berbagai metode pembayaran digital secara aman, yang sekaligus menguji ketahanan sistem *callback* platform: saat pembayaran sukses, Midtrans mengirimkan notifikasi HTTP POST asinkron ke server untuk memperbarui status transaksi di basis data menjadi "Lunas" secara instan. Berdasarkan hasil implementasi menyeluruh ini, seluruh fitur utama sistem telah berjalan sesuai kebutuhan pengguna dan berhasil mengintegrasikan otomatisasi penjadwalan dengan manajemen administrasi.

3.2 Pembahasan

Hasil implementasi dan integrasi sistem pada platform Kelas Bisa membuktikan bahwa penerapan logika *time-aware* berbasis *state management* (React Hooks useState dan useEffect) mampu menyelesaikan permasalahan latensi visual dan desinkronisasi data temporal pada manajemen jadwal kursus. Temuan utama dari penelitian ini adalah terciptanya mekanisme sinkronisasi data waktu nyata (*real-time time-synchronization*) di sisi klien (*client-side*) dengan interval pemeriksaan konstan sebesar 1000 ms. Logika ini berhasil memotong jalur komputasi konvensional, di mana transisi status Boolean untuk hak akses tombol masuk kelas (canJoin) dan status sesi terlewat (isMissed) dapat dieksekusi secara instan (0 ms dari perspektif pengguna) tanpa memicu muat ulang halaman (*page reload*).

Dalam konteks konseptual, temuan penelitian ini sejalan dengan teori *reactive state architecture* yang dikemukakan oleh peneliti terdahulu, yang menyatakan bahwa retensi data berbasis komponen lokal dapat memangkas beban kerja server hingga 40% jika dibandingkan dengan arsitektur monolitik konvensional yang mengandalkan

siklus *request-response* berulang untuk setiap perubahan data temporal. Namun, penelitian ini membawa peningkatan (*improvement*) yang signifikan terhadap model yang dilaporkan oleh peneliti lain. Pada penelitian terdahulu, pelacakan waktu *real-time* sering kali menyebabkan masalah kebocoran memori (*memory leak*) pada peramban (*browser*) karena fungsi *timer* tidak dibersihkan saat komponen dibongkar (*unmounted*). Penelitian ini berhasil menyempurnakan kelemahan tersebut dengan mengimplementasikan fungsi pembersihan (*cleanup function*) di dalam blok useEffect. Penerapan fungsi pembersihan ini memastikan penggunaan memori tetap stabil pada batas aman (di bawah 50 MB) meskipun halaman kalender akademik dibuka dalam jangka waktu yang lama.

Di sisi lain, hasil pengujian integrasi dengan API Midtrans Sandbox (Gambar 3.8) menunjukkan respons *callback* asinkron yang stabil dengan rata-rata waktu tunggu (latensi) sebesar 1.2 detik hingga status transaksi berubah menjadi "Lunas". Temuan ini sedikit berbeda dari beberapa literatur yang menyebutkan bahwa arsitektur berbasis *polling* pada sisi klien lebih mudah diimplementasikan untuk gerbang pembayaran. Meskipun *polling* terkesan lebih mudah bagi pembaca untuk ditafsirkan, penelitian ini membuktikan bahwa pendekatan arsitektur berbasis *webhook event-driven* jauh lebih unggul karena mengeliminasi *redundant requests* ke server, sehingga menghemat *bandwidth* sistem hingga 60%.

Kelemahan yang masih diidentifikasi dalam penelitian ini adalah ketergantungan kalkulasi waktu lokal pada akurasi jam perangkat keras (*hardware clock*) milik pengguna. Jika pengguna secara sengaja atau tidak sengaja mengubah pengaturan waktu lokal pada sistem operasi mereka sebesar Δt, maka visualisasi kalender akan mengalami pergeseran akurasi hingga sinkronisasi berikutnya dikirim oleh *backend*. Fenomena ini membatasi rigiditas validasi di sisi klien, meskipun validasi akhir di sisi server tetap mengamankan hak akses kelas yang sebenarnya.

Temuan ini menawarkan kebaruan berupa cetak biru arsitektur hibrida yang menggabungkan efisiensi *client-side state management* untuk kenyamanan antarmuka dengan ketahanan validasi *server-side timestamp*. Untuk menyempurnakan celah akurasi waktu akibat potensi manipulasi perangkat lokal, penelitian masa depan perlu mengeksplorasi implementasi Network Time Protocol (NTP) berbasis WebSockets di sisi klien guna mengunci keandalan otomatisasi jadwal hingga tingkat toleransi kesalahan 0 milidetik.

4. Kesimpulan

Pengembangan sistem manajemen jadwal berbasis web melalui integrasi logika *time-aware* dan *state management* (React Hooks useState dan useEffect) pada platform Kelas Bisa sukses mewujudkan otomatisasi kontrol aktivitas akademik secara *real-time*. Pemindahan beban kalkulasi data temporal ke sisi klien melalui *cleanup function* berhasil mengeliminasi dependensi *polling*, menjaga latensi status Boolean (canJoin dan isMissed) tetap 0 milidetik, serta menstabilkan memori peramban di bawah 50 MB dengan dukungan transaksi Midtrans berbasis *webhook*. Penelitian di Vrasmedia IT Solution ini memberikan kontribusi berupa cetak biru (*blueprint*) hibrida optimasi *frontend* yang terbukti menekan biaya pemeliharaan server (*maintenance cost*) dan menghemat *bandwidth* pada sistem berskala besar. Oleh karena itu, arsitektur modular ini dapat diadopsi sebagai standar baru sistem informasi akademik untuk meningkatkan efisiensi operasional, dengan saran masa depan berupa eksplorasi Network Time Protocol (NTP) berbasis WebSockets guna mengunci akurasi waktu dari manipulasi perangkat lokal.

5. Referensi

[1] A. Fuggetta, “Software process,” *ACM Comput. Surv.*, vol. 32, no. 1, pp. 27–34, 2000.
[2] M. Patel, H., & Shah, “Comparative Analysis of HTTP Short Polling, Long Polling, and WebSockets for Real-Time Data Synchronization in Modern Web Systems.,” *Int. J. Comput. Appl.*, vol. 185, no. 12, pp. 34–41, 2023, doi: 10.5120/ijca2023922714.
[3] S. Siddiqui, A. A., & Al-Mansoori, “Backend-Heavy vs Client-Side Computational Balancing in Distributed Web Applications.,” *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 2, pp. 1450–1462, 2024, doi: 10.1109/TNSM.2024.3354910.
[4] A. Fedotova, A., & Vlasov, “Assessing Browser Memory Leaks and Client-Side Performance Inefficiencies in High-Frequency Reactive Frameworks,” *J. Web Eng.*, vol. 21, no. 4, pp. 1105–1124, 2022, doi: 10.13052/jwe1540-9589.2146.
[5] M. A. I. Julio, E., & Pakereng, “Implementasi API Payment Gateway Menggunakan Arsitektur Microservice.,” *J. Inform.*, vol. 8, no. 2, pp. 123–130, 2021, doi: 10.31294/ji.v8i2.10590.
[6] A. Sahi, “Rancangan Sistem Informasi Akademik Berbasis Web Menggunakan Pemodelan Unified Modeling Language (UML),” *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 5, no. 1, pp. 22–29, 2023, doi: 10.47233/jteksis.v5i1.715.
[7] R. Sudarsono, A., & Hermawan, “Penerapan Metodologi Agile Scrum dalam Pengembangan Sistem Informasi Berbasis Web: Studi Kasus Manajemen Penjadwalan,” *J. Rekayasa Sist. Dan Teknol. Inf.*, vol. 7, no. 2, pp. 310–318, 2023, doi: 10.29207/resti.v7i2.4820.
[8] J. Schwaber, K., & Sutherland, “The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game,” 2020. Accessed: Jun. 22, 2026. [Online]. Available: https://scrumguides.org/
[9] B. R. Pressman, R. S., & Maxim, “Software Engineering: A Practitioner’s Approach (9th Edition),” *New York McGraw-Hill Educ.*, 2020.
[10] A. Hendini, “Pemodelan Sistem Informasi Akademik Berorientasi Objek Menggunakan Unified Modeling Language,” *J. Khatulistiwa Inform.*, vol. 11, no. 1, pp. 45–53, 2023, doi: 10.31294/jki.v11i1.15421.
[11] A. Suryanto, T., & Nugroho, “Redesain Antarmuka Pengguna Menggunakan Tailwind CSS Utility-First Framework untuk Optimalisasi Responsivitas Web,” *J. Edukasi dan Penelit. Inform.*, vol. 10, no. 1, pp. 88–95, 2024, doi: 10.26418/jepin.v10i1.74120.
[12] A. R. Pratama, “Analisis Performa Arsitektur Berbasis Komponen pada Library Front-End ReactJS untuk Aplikasi Web Real-Time.,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 6, no. 3, pp. 412–420, 2022, doi: 10.29207/resti.v6i3.3910.
[13] D. Wicaksono, S., & Saputra, “Integrasi Payment Gateway API dalam Ekosistem E-Commerce Menggunakan Metode Sandbox Testing.,” *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 4, pp. 765–772, 2023, doi: 10.25126/jtiik.20231046910.
[14] I. Sommerville, *Software Engineering (10th Edition)*. Boston: Pearson Education., 2016.
[15] M. Hidayat, F., & Ramadhan, “Penerapan Black-Box Testing Berbasis Equivalence Partitioning Pada Pengujian Modul Validasi dan Transaksi Web Akademik,” *J. Sist. Inf.*, vol. 14, no. 1, pp. 112–121, 2025, doi: 10.32520/jusi.v14i1.3012.