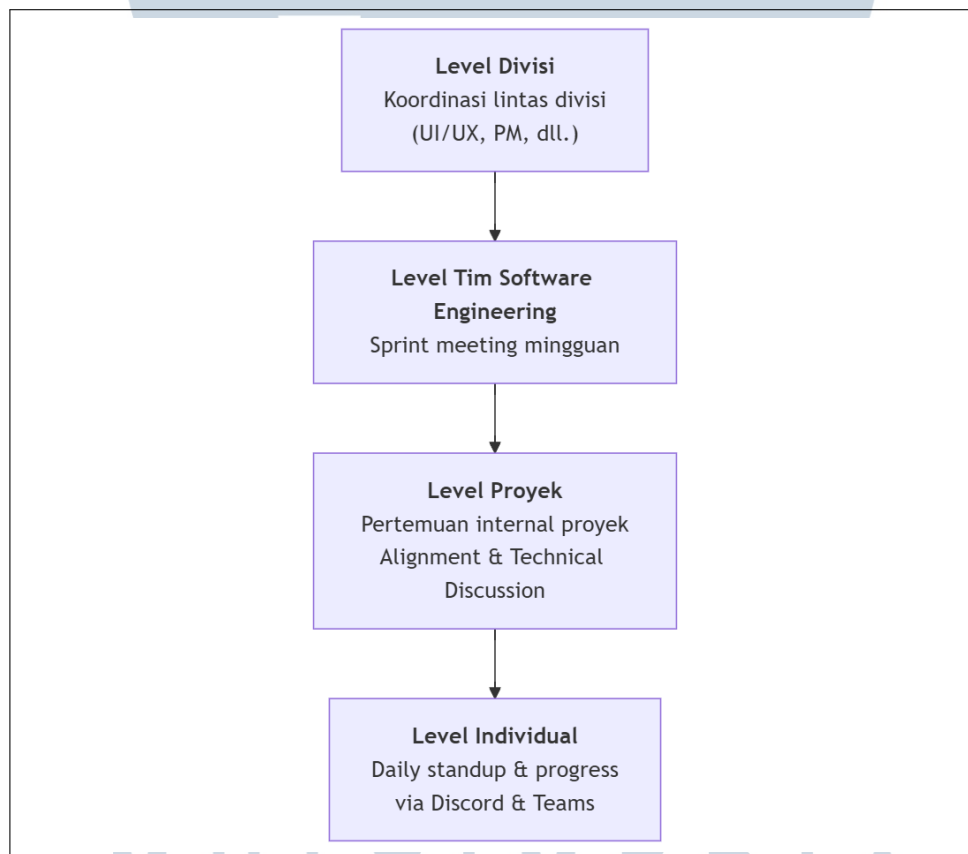


BAB 3

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam struktur organisasi CDT, posisi Software Engineer Intern berada di bawah supervisi langsung dari divisi Software Engineering (SWE), yang dipimpin oleh Lead Software Engineer Russell Otniel Tjakra. Koordinasi kerja dilakukan secara lintas divisi yang melibatkan tim UI/UX, Product Management (PM), serta divisi-divisi pendukung lainnya. Alur koordinasi kerja selama masa magang dapat dilihat pada Gambar 3.1 di bawah ini.



Gambar 3.1. Bagan Alur Koordinasi Kerja Magang CDT SWE

Pekerjaan dilakukan dalam struktur tim yang mengadopsi metodologi Agile development dengan sprint meetings mingguan. Koordinasi harian dilakukan melalui platform Microsoft Teams dan Discord untuk memfasilitasi komunikasi real-time, terutama mengingat pelaksanaan magang yang dilakukan secara remote.

Setiap proyek memiliki struktur pelaporan yang jelas dengan supervisor teknis dari divisi SWE dan PM yang bertanggung jawab atas timeline dan kebutuhan user.

Sistem pelaporan progres dilakukan secara berkala melalui presentasi mingguan selama fase pelatihan dan biweekly meeting untuk setiap proyek spesifik. Alur komunikasi vertikal dilakukan melalui supervisor langsung, sedangkan koordinasi horizontal dilakukan antar sesama peserta dalam tim yang sama.

3.2 Tugas yang Dilakukan

Selama masa pelaksanaan kerja magang, berbagai tugas telah dilaksanakan sesuai dengan fase dan proyek yang ditugaskan, terutama pada pembuatan POC sistem kolaborasi real-time pada ELN. Setiap tugas bersifat eksploratif, dengan tujuan untuk memastikan bahwa solusi yang dirancang dapat diimplementasikan secara teknis dan memenuhi kebutuhan user. Adapun uraian tugas yang dilaksanakan adalah sebagai berikut:

1. Penyempurnaan UI Penampil DICOM

Selama masa penugasan singkat di proyek Radiology Information System & Picture Archiving and Communication System (RIS & PACS), dilakukan penyempurnaan UI penampil Digital Imaging and Communications in Medicine (DICOM) sebagai bagian dari upaya peningkatan pengalaman user dalam bekerja dengan gambar medis. Fokus penugasan ini mencakup optimalisasi UI pengguna untuk meningkatkan aksesibilitas dan responsivitas tampilan lintas perangkat.

2. Pembuatan Modul Kolaborasi Real-Time

Tugas ini mencakup pembuatan modul kolaborasi real-time yang memungkinkan lebih dari satu user melakukan penyuntingan sebuah modul, dalam bentuk form, secara bersamaan. Modul ini dirancang untuk menangani sinkronisasi data secara langsung antar pengguna serta meminimalkan konflik selama proses pengeditan. Implementasi dilakukan melalui dua tahap iteratif untuk menyesuaikan dengan kebutuhan user, dengan pendekatan *field-level locking* pada iterasi pertama dan *page-level locking* pada iterasi kedua.

3. Pembuatan Rich Text Editor dengan Kolaborasi Real-Time

Tugas ini mencakup pembuatan rich text editor dengan kemampuan kolaborasi real-time untuk memungkinkan lebih dari satu pengguna dalam menulis

dan menyunting dokumen secara simultan. Fitur ini memungkinkan setiap perubahan yang dilakukan oleh pengguna disinkronkan secara instan tanpa menimbulkan konflik sekaligus melakukan penyimpanan dokumen berkala di database. Implementasi dibangun dengan arsitektur modern yang mengadopsi pendekatan Conflict-free Replicated Data Type (CRDT) serta komunikasi dua arah berbasis WebSocket untuk memastikan sinkronisasi data yang stabil dan efisien selama kolaborasi berlangsung. Selain itu, text editor ini juga memungkinkan user untuk melakukan export dokumen menjadi PDF.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang Dilakukan Tiap Minggu Selama Magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Onboarding tim, pelatihan dasar HTML/CSS/JavaScript, serta presentasi hasil pelatihan awal.
2	Pelatihan Git dan TypeScript, persiapan presentasi, serta eksplorasi awal framework frontend.
3	Pengembangan aplikasi menggunakan framework UI, pelatihan Next.js, serta pengembangan UI.
4	Melanjutkan proyek Next.js, presentasi, serta mulai integrasi Supabase.
5	Integrasi Supabase ke aplikasi, presentasi hasil, serta onboarding ke proyek riset AI internal.
6	Pengembangan proyek grup berbasis Supabase self-hosted, implementasi fitur autentikasi dan produk, presentasi.
7	Eksplorasi repositori internal RIS & PACS, diskusi redesign sistem imaging, dan sprint meeting.
8	Dikerahkan ke proyek e-Laboratory Notebook (eLN), eksplorasi fitur kolaborasi real-time, debugging UI.
9	Implementasi awal fitur kolaborasi real-time, perbaikan UI responsif, eksplorasi keamanan sistem.
10	Refactor modul kolaborasi, penyesuaian berdasarkan feedback, serta eksplorasi fitur penguncian.

Bersambung ke halaman berikutnya

Tabel 3.1 Pekerjaan yang Dilakukan Tiap Minggu Selama Magang (Lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
11	Melanjutkan pengembangan editor kolaboratif, eksplorasi text editor real-time, serta lanjutan POC.
12	Implementasi text editor kolaboratif, eksplorasi teknologi pendukung, dan integrasi dengan desain.
13	Reimplementasi modul kolaborasi, setup infrastruktur pendukung, serta perbaikan awal.
14	Integrasi modul kolaborasi, perbaikan bug serta caret tracking, merge dengan fitur visualisasi data.
15	Perbaikan bug modul kolaborasi, eksplorasi penyimpanan dokumen, serta alternatif provider.
16	Setup server kolaborasi internal, integrasi dengan client editor, sinkronisasi data backend.
17	Perbaikan koneksi server-client, konversi format dokumen, penyempurnaan modul kolaborasi.
18	Konversi dokumen ke format universal, penyesuaian konten kompleks, serta pembuatan halaman demo.
19	Penyempurnaan modul kolaborasi, debugging fitur PDF, eksplorasi hosting server internal.

1. Minggu 1: 3 Februari – 7 Februari 2025

- (a) Sesi orientasi dan pengenalan terhadap struktur organisasi serta ruang lingkup kerja divisi CDT.
- (b) Penentuan peran sesuai latar belakang dan minat mahasiswa dalam pengembangan perangkat lunak.
- (c) Pelatihan teknis mandiri dasar web development menggunakan HTML dan CSS.
- (d) Implementasi dasar untuk evaluasi pemahaman, dilanjutkan dengan JavaScript dan pengenalan Git.
- (e) Presentasi recap pembelajaran HTML, CSS, dan JavaScript.

2. Minggu 2: 10 Februari – 14 Februari 2025

- (a) Pendalaman TypeScript dengan fokus pada sintaksis dan aplikasi modern.

- (b) Persiapan materi presentasi teknis Git dan TypeScript.
 - (c) Pelatihan mandiri React dengan TypeScript.
 - (d) Diskusi teknis informal dan pengenalan proyek mini internal.
3. Minggu 3: 17 Februari – 21 Februari 2025
- (a) Pengembangan proyek mini menggunakan React dan Material UI (tema aplikasi wallet).
 - (b) Pembelajaran Next.js dengan pendekatan modular.
 - (c) Implementasi awal Next.js dengan Material UI dan Tailwind CSS.
4. Minggu 4: 24 Februari – 28 Februari 2025
- (a) Pengembangan aplikasi dashboard dengan Next.js.
 - (b) Pembuatan aplikasi pelacak kebiasaan harian.
 - (c) Penggunaan TypeScript, Material UI, dan Tailwind CSS.
 - (d) Presentasi recap dan evaluasi integrasi backend.
5. Minggu 5: 3 Maret – 7 Maret 2025
- (a) Integrasi backend menggunakan Supabase ke aplikasi Next.js.
 - (b) Penyusunan materi presentasi integrasi dan pengujian fitur CRUD dan autentikasi.
 - (c) Onboarding proyek AI internal dan RIS & PACS serta riset platform AI eksternal.
6. **Minggu 6: 10 Maret – 14 Maret 2025**
- (a) Proyek mini kelompok dengan Next.js dan Supabase (self-hosted).
 - (b) Implementasi fitur tampilan produk, pencarian, dan autentikasi.
 - (c) Presentasi akhir proyek kelompok.
7. Minggu 7: 17 Maret – 21 Maret 2025
- (a) Eksplorasi sistem RIS & PACS dan analisis kode repo.
 - (b) Pertemuan sprint dan diskusi standar DICOM.
 - (c) Diskusi pengembangan fitur dengan tim SWE internal.

8. Minggu 8: 24 Maret – 28 Maret 2025
 - (a) Realokasi ke proyek ELN dan onboarding teknis.
 - (b) Eksplorasi sistem realtime Supabase dan bug fixing toolbar viewer.
 - (c) Diskusi fase POC.
9. Minggu 9: 2 April – 5 April 2025
 - (a) Pertemuan rutin SWE ELN dan pengembangan fitur kolaborasi real-time.
 - (b) Perbaikan bug UI dan eksplorasi fitur presence dan sinkronisasi.
 - (c) Evaluasi implikasi keamanan modul realtime.
10. Minggu 10: 7 April – 11 April 2025
 - (a) Restrukturisasi modul kolaborasi.
 - (b) Peningkatan efisiensi komunikasi dan penguncian modul.
 - (c) Diskusi struktur data untuk pengguna simultan.
11. Minggu 11: 14 April – 18 April 2025
 - (a) Implementasi text editor collaborative.
 - (b) Eksplorasi arsitektur multiple user editing.
 - (c) Validasi integrasi UI berdasarkan desain Figma.
12. Minggu 12: 21 April – 25 April 2025
 - (a) Implementasi lanjutan editor.
 - (b) Eksplorasi protokol sinkronisasi dan uji skenario kolaboratif.
 - (c) Sinkronisasi hasil uji coba dengan UI.
13. Minggu 13: 28 April – 2 Mei 2025
 - (a) Reimplementasi modul kolaborasi untuk real-time editing.
 - (b) Setup sistem eksternal dengan containerization.
 - (c) Debugging awal dan pengujian unit.
14. Minggu 14: 5 Mei – 9 Mei 2025
 - (a) Integrasi modul kolaborasi ke proyek utama.

- (b) Implementasi user caret tracking.
 - (c) Penggabungan branch eksperimen grafik/chart editor.
15. Minggu 15: 13 Mei – 17 Mei 2025
- (a) Evaluasi sistem penyimpanan kolaboratif.
 - (b) Konsultasi solusi hosting dan penyimpanan.
 - (c) Uji sistem fallback koneksi realtime.
16. Minggu 16: 20 Mei – 24 Mei 2025
- (a) Setup server kolaborasi realtime untuk uji lokal.
 - (b) Integrasi modul server dengan client editor.
 - (c) Penyimpanan data kolaboratif secara persisten.
17. Minggu 17: 26 Mei – 30 Mei 2025
- (a) Debugging koneksi server-client.
 - (b) Implementasi konversi dokumen ke format serialisasi.
 - (c) Perbaikan sinkronisasi jaringan tidak stabil.
18. Minggu 18: 2 Juni – 6 Juni 2025
- (a) Penyempurnaan modul konversi dokumen (JSON, HTML, Markdown).
 - (b) Eksplorasi decoding diagram dan chart.
 - (c) Pembuatan halaman demo hasil konversi.
19. Minggu 19: 9 Juni – 11 Juni 2025
- (a) Penyelesaian modul kolaborasi sesuai kebutuhan sistem ELN terbaru.
 - (b) Penyelesaian isu ekspor PDF.
 - (c) Eksplorasi hosting server kolaborasi eksternal.

3.3.1 Penyempurnaan UI Penampil DICOM

Setelah menyelesaikan masa pelatihan, penugasan berlanjut pada proyek RIS & PACS yang berfungsi sebagai platform terintegrasi untuk pengelolaan informasi radiologi serta penyimpanan dan visualisasi gambar medis seperti MRI, CT Scan, dan X-ray. Proyek ini berada dalam fase pengembangan lanjutan dengan fokus pada perbaikan fungsionalitas, penyempurnaan antarmuka pengguna, serta peningkatan responsivitas sistem berdasarkan umpan balik dari pengguna awal. Salah satu kontribusi utama adalah pengembangan modul penampil DICOM yang dikustomisasi untuk disesuaikan dengan kebutuhan user.

Berikut adalah ringkasan kontribusi teknis utama selama pengerjaan proyek:

Tabel 3.2. Rangkuman Kontribusi dalam Penyempurnaan UI Penampil DICOM

Area Kontribusi	Deskripsi
UI Redesign	Reposisi toolbar dari atas ke bawah untuk meningkatkan aksesibilitas, perubahan desain dari posisi <i>fixed</i> menjadi <i>floating</i> untuk efisiensi ruang.
Perbaikan Tooltip Bug	Penyelesaian bug tooltip yang muncul di posisi tidak tepat saat interaksi pertama melalui <i>prop-drilling</i> minimal guna memicu ulang proses render tanpa mengganggu struktur komponen.
Peningkatan Responsivitas Tampilan	Penyesuaian tampilan tooltip dan toolbar untuk mendukung berbagai ukuran layar dan orientasi perangkat, memastikan konsistensi visual cross-platform.

3.3.2 Modul Kolaborasi Real-Time

Modul kolaborasi real-time ini merupakan bagian dari sistem kolaborasi real-time di ELN untuk memungkinkan lebih dari satu user menyunting sebuah modul berbentuk form secara bersamaan. Implementasi kolaborasi ini berfokus pada sinkronisasi data secara real-time antar user serta pengelolaan konflik dalam penyuntingan.

Fitur kolaborasi ini dikembangkan melalui dua iterasi, yang masing-masing mencerminkan respons terhadap kebutuhan user yang terus berkembang:

1. Iterasi pertama mengadopsi pendekatan *field-level locking*, yaitu pengguna

dapat menyunting modul yang sama secara bersamaan selama mereka tidak menyunting kolom yang sama.

2. Iterasi kedua mengadopsi pendekatan *page-level locking*, di mana hanya satu pengguna yang dapat menyunting modul pada satu waktu sementara pengguna lain hanya dapat melihat perubahan secara real-time.

A Kebutuhan User

Secara umum, modul kolaborasi real-time ini diharapkan dapat menyokong kolaborasi dan komunikasi lebih dari user dalam proses pencatatan eksperimen. Berikut adalah kebutuhan user yang telah diperoleh dari divisi PM dan didiskusikan dengan divisi SWE terkait implementasi modul kolaborasi real-time:

1. Semua user yang aktif di modul dapat melihat perubahan penyuntingan modul secara real-time.
2. User dapat melihat keberadaan user lain.
3. Penyuntingan dapat dilakukan tanpa konflik data dan persistensi state modul.
4. Lebih dari satu user dapat menyunting modul yang sama secara bersamaan (iterasi 1); berkembang menjadi pembatasan satu user dalam satu modul (iterasi 2).

B Riset Teknologi dan Implementasi

Dalam tahap awal pengembangan modul kolaborasi real-time, riset dilakukan untuk menentukan pendekatan teknologi yang paling sesuai dengan kebutuhan sistem. Tujuan utama dari riset ini adalah untuk menemukan solusi yang memungkinkan sinkronisasi data antarpengguna secara efisien, tanpa menimbulkan konflik atau beban berlebih pada sistem.

Dari hasil eksplorasi, terdapat dua pendekatan utama yang dipertimbangkan.

1. Pendekatan pertama menggunakan layanan sinkronisasi data waktu nyata yang sudah terintegrasi dalam infrastruktur sistem perusahaan. Meskipun secara fitur lebih terbatas, pendekatan ini menawarkan integrasi yang sangat ringan dan cepat, serta dinilai memadai untuk kebutuhan modul saat ini, mengingat bentuk kolaborasi yang lebih tersegmentasi antarpengguna sehingga risiko konflik data sangat minim.

2. Pendekatan kedua implementasi berbasis CRDT, yang sangat kuat dan cocok untuk kolaborasi simultan tingkat lanjut. Namun, pendekatan ini memerlukan konfigurasi yang lebih kompleks dan dianggap kurang relevan untuk skala kebutuhan saat ini. Kendati demikian, pendekatan ini tetap dicatat sebagai opsi yang layak apabila fitur kolaborasi ingin ditingkatkan di masa mendatang.

Setelah mempertimbangkan kelebihan dan kekurangan masing-masing, pendekatan pertama dipilih karena lebih sesuai dengan arsitektur sistem yang ada dan cukup untuk memenuhi kebutuhan kolaborasi dasar.

Beberapa strategi implementasi sinkronisasi data juga dievaluasi:

1. Pendekatan berbasis penyiaran status pengguna: Metode ini menggunakan siaran langsung antar pengguna untuk mempertahankan status kolaboratif secara lokal. Sinkronisasi dilakukan tanpa menyimpan data secara langsung di basis data, kecuali untuk penyimpanan otomatis berkala. Pendekatan ini sangat ringan namun bergantung pada satu pengguna sebagai sumber utama, sehingga rawan ketidakkonsistenan.
2. Pendekatan berbasis perubahan data terpusat: Dalam pendekatan ini, setiap perubahan langsung dicatat melalui pembaruan ke basis data. Meskipun sangat akurat dan konsisten, pendekatan ini menimbulkan beban performa yang signifikan, terutama pada interaksi intens seperti pengetikan, karena tingginya frekuensi pemanggilan ke server.
3. Pendekatan gabungan: Strategi yang akhirnya dipilih adalah kombinasi antara penyiaran lokal dan perubahan berbasis basis data. Informasi disinkronkan secara efisien melalui siaran langsung antar pengguna untuk performa optimal, sementara data penting tetap dikirim ke basis data dengan frekuensi yang dikendalikan. Pendekatan ini menawarkan keseimbangan antara efisiensi, kecepatan, dan keandalan.

Rangkaian riset dan uji coba ini menjadi dasar pemilihan arsitektur akhir modul kolaborasi, yang tidak hanya mempertimbangkan kebutuhan teknis saat ini, tetapi juga fleksibilitas untuk pengembangan lebih lanjut di masa depan.

C Iterasi 1: *Field-Level Locking*

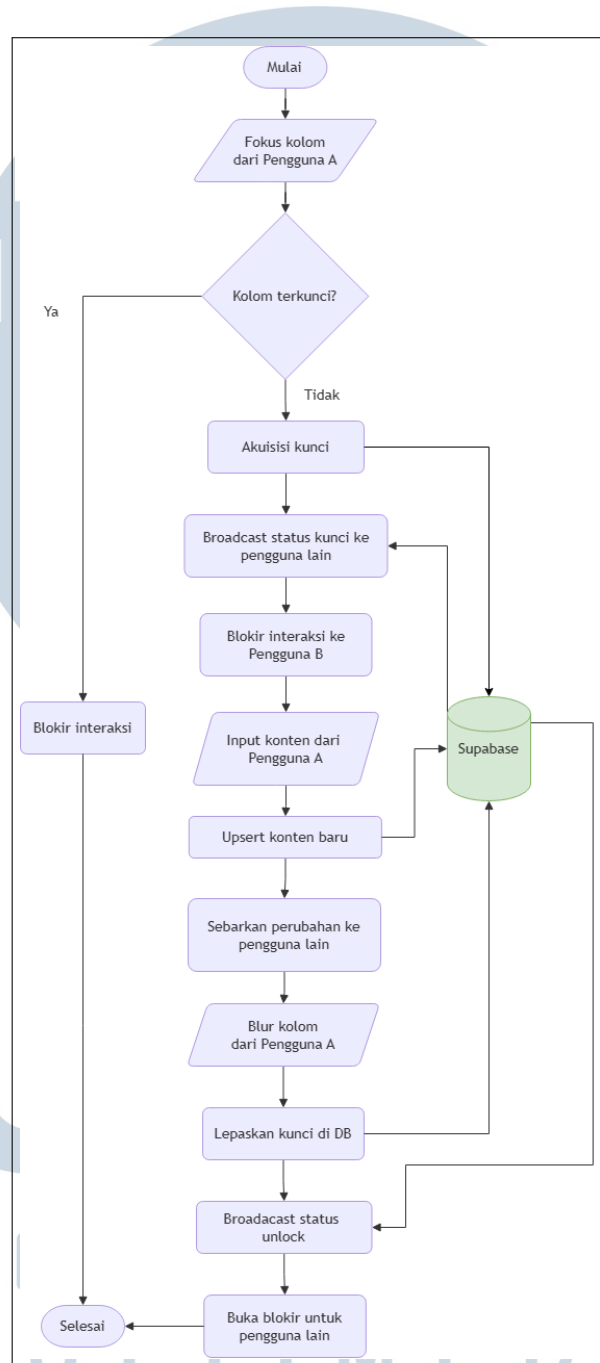
Iterasi pertama dari sistem kolaborasi real-time berfokus pada implementasi mekanisme penguncian per kolom (*field-level locking*). Tujuan utamanya adalah

memungkinkan lebih dari satu pengguna untuk mengakses dan menyunting satu modul secara bersamaan, namun dengan pencegahan konflik data melalui mekanisme kunci untuk setiap kolom secara individual.

Pendekatan ini menghasilkan sistem kolaborasi yang fleksibel dan scalable, di mana pengguna dapat berinteraksi dalam modul yang sama, namun tidak dapat menyunting kolom yang sedang dikunci oleh pengguna lain. Mekanisme ini juga disertai dengan update visual secara real-time.



C.1 Alur Interaksi Pengguna

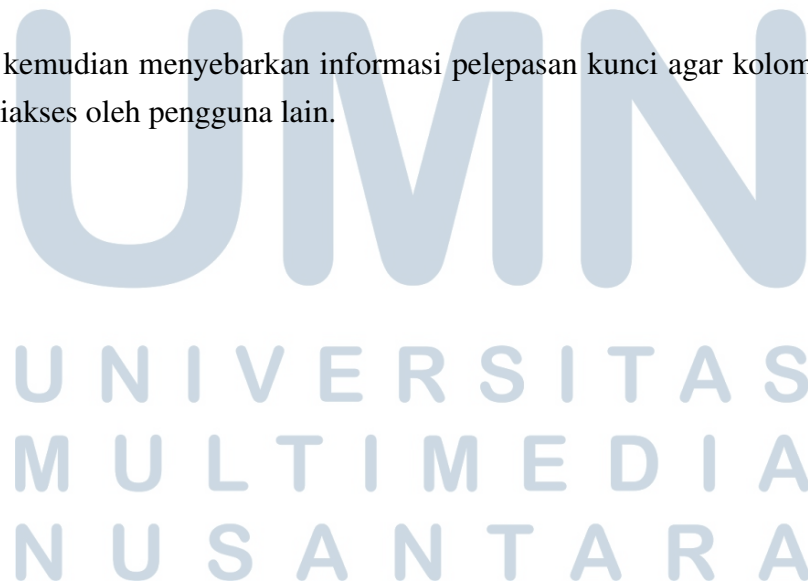


Gambar 3.2. User Flow pada Iterasi 1 Modul Kolaborasi Real-Time

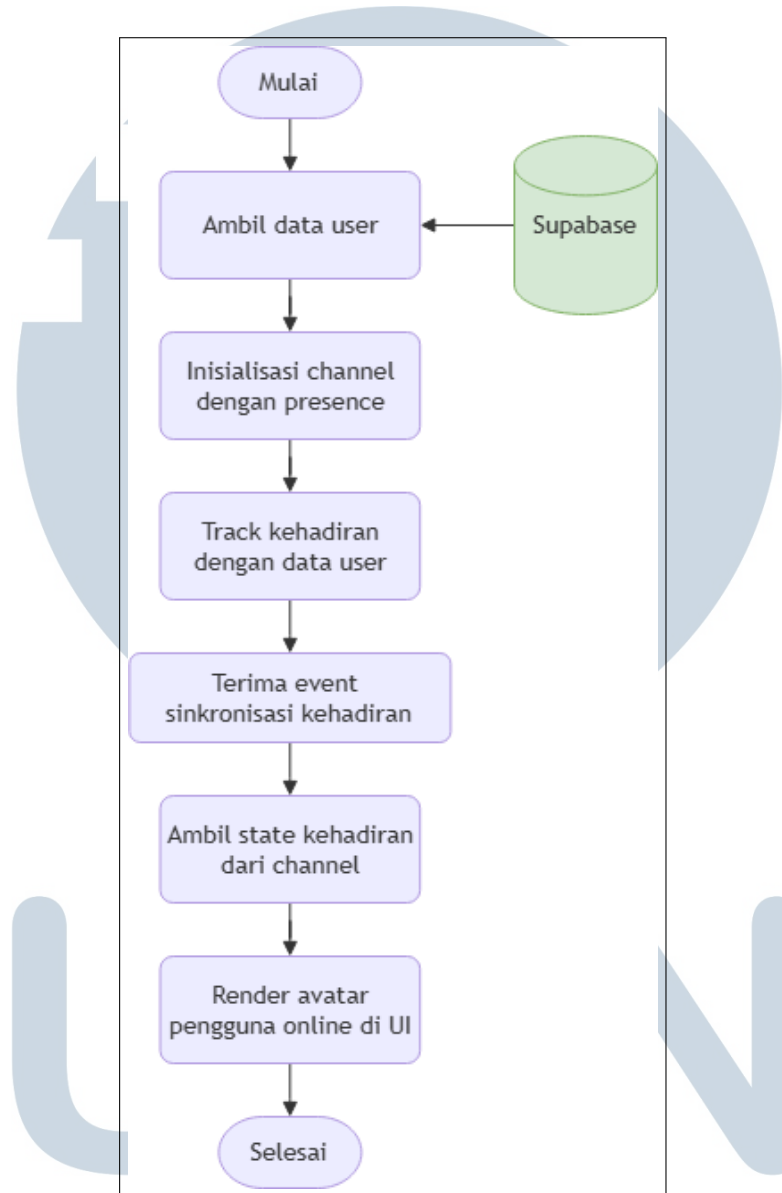
Diagram alur pada Gambar 3.2 menggambarkan langkah-langkah interaksi pengguna dalam proses kolaborasi pengisian formulir secara real-time. Alur ini menunjukkan bagaimana sistem menangani permintaan kunci, penyebaran

perubahan, serta pelepasan kunci secara otomatis untuk menjaga konsistensi antar pengguna. Berikut adalah rincian tiap langkah dalam proses tersebut:

1. Pengguna A mengarahkan fokus ke suatu kolom input dalam formulir.
2. Sistem memeriksa apakah kolom tersebut sedang dikunci oleh pengguna lain.
3. Jika kolom sedang dikunci, maka interaksi pengguna diblokir dan sistem menunggu kolom tersedia.
4. Jika kolom tidak terkunci, sistem akan mengakuisisi kunci dengan menyimpan data kunci ke basis data.
5. Setelah kunci berhasil diperoleh, sistem menyebarkan status kunci melalui layanan real-time ke seluruh pengguna lain.
6. Pengguna lain (misalnya pengguna B) menerima informasi tersebut dan otomatis antarmukanya akan memblokir interaksi terhadap kolom tersebut.
7. Selama pengguna A mengetik, perubahan ditampilkan secara langsung di layar dan secara berkala dikirim ke server serta disebarkan ke pengguna lain.
8. Ketika pengguna A keluar dari kolom atau menutup halaman, sistem melepaskan kunci dari basis data.
9. Sistem kemudian menyebarkan informasi pelepasan kunci agar kolom kembali dapat diakses oleh pengguna lain.



C.2 Fitur Presence Tracker



Gambar 3.3. Flowchart Pelacakan Presence User

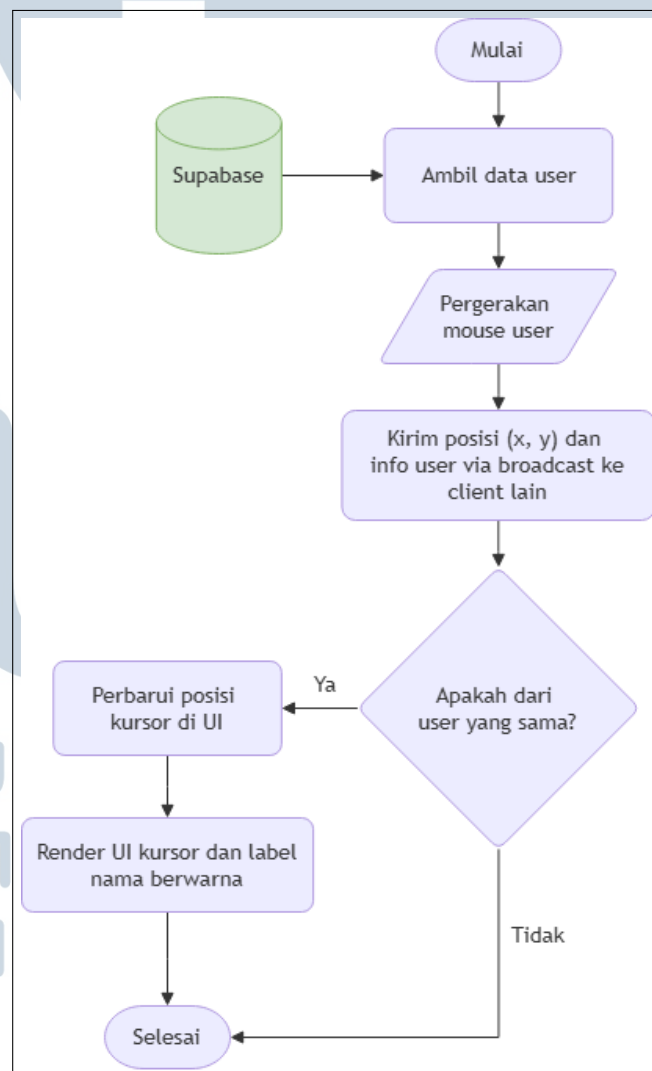
Fitur presence tracker menampilkan daftar pengguna yang sedang online di modul yang sama untuk meningkatkan pengalaman kolaborasi. Gambar 3.3 menjelaskan alur kerjanya yang dapat dirincikan sebagai berikut:

1. Saat user membuka halaman, sistem menginisialisasi koneksi ke channel Supabase untuk pelacakan kehadiran.

2. Client melakukan tracking ke channel dengan mengirimkan informasi pengguna seperti nama, ID, dan warna identitas.
3. Saat server menyinkronkan kehadiran pengguna, event sinkronisasi dipicu.
4. State kehadiran dari channel dikumpulkan dan diproses.
5. Daftar pengguna online ditampilkan dalam UI berupa avatar warna dengan inisial atau nama pengguna.

Tampilan fitur ini dapat dilihat pada Gambar 3.5.

C.3 Fitur Cursor Tracker



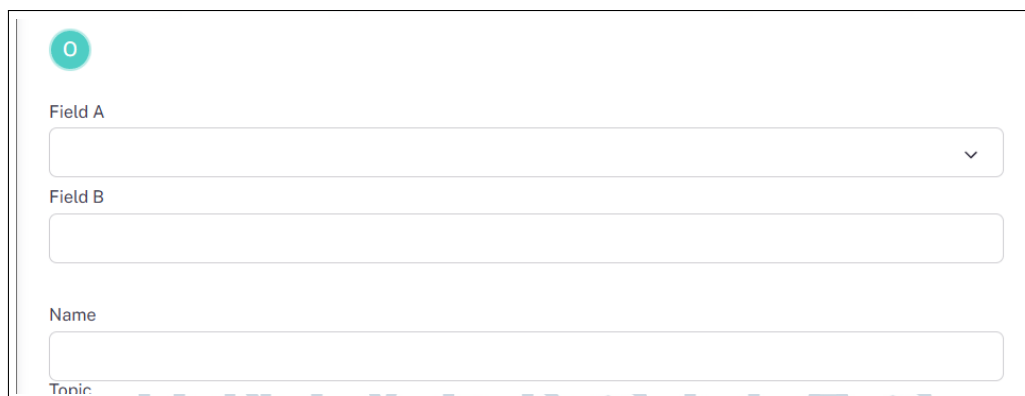
Gambar 3.4. Flowchart Pelacakan Cursor User

Fitur cursor tracker melacak pergerakan kursor dari user lain dalam modul yang sama dan menampilkannya secara langsung di UI untuk meningkatkan pengalaman kolaborasi dan komunikasi user. Gambar 3.4 menjelaskan alur kerjanya yang dapat dirincikan sebagai berikut:

1. Setiap kali pengguna menggerakkan mouse, sistem mengirimkan data posisi kursor (x, y), bersama dengan informasi pengguna, melalui channel siaran.
2. Client lain menerima event pergerakan cursor yang dikirimkan oleh server.
3. Jika data tersebut berasal dari user yang sama, maka event diabaikan agar tidak ditampilkan ganda.
4. Jika berasal dari user lain, maka data disimpan di state lokal.
5. UI kemudian merender posisi kursor user lain dengan warna dan label nama sesuai identitas pengguna.

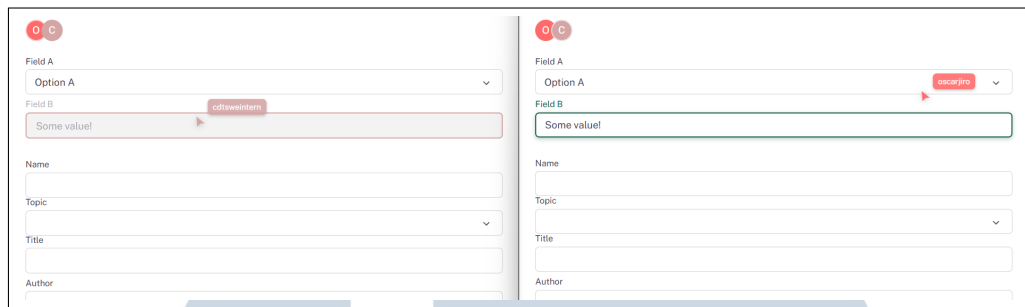
Tampilan fitur ini dapat dilihat pada Gambar 3.6.

C.4 Tampilan dan Simulasi Kolaborasi Antar Pengguna



Gambar 3.5. Cuplikan UI Modul dengan Kolom Interaktif untuk Kolaborasi Real-Time

Gambar 3.5 menunjukkan cuplikan UI pengguna dari modul kolaborasi. Terlihat terdapat sebuah presence tracker di atas form untuk menunjukkan pengguna mana saja yang sedang online di modul tersebut. Masing-masing kolom dapat dikunci secara individual, dan perubahan akan terlihat langsung ketika pengguna lain sedang melakukan penyuntingan.



Gambar 3.6. Simulasi Dua Pengguna dalam Tampilan Split-Window dengan Kolaborasi Real-Time

Gambar 3.6 menunjukkan simulasi dua pengguna yang sedang bekerja di halaman yang sama. Di sisi kiri, pengguna A telah mengunci dan sedang menyunting sebuah kolom, ditandai dengan tampilan aktif dan kursor pengetikan. Sementara di sisi kanan, pengguna B melihat kolom yang sama dalam kondisi terkunci/dinonaktifkan, tidak dapat diedit, dan memiliki warna border sesuai dengan warna profil pengguna yang sedang menyunting. Terlihat juga cursor setiap pengguna yang bergerak secara real-time untuk meningkatkan rasa kehadiran, kesadaran konteks, dan komunikasi.

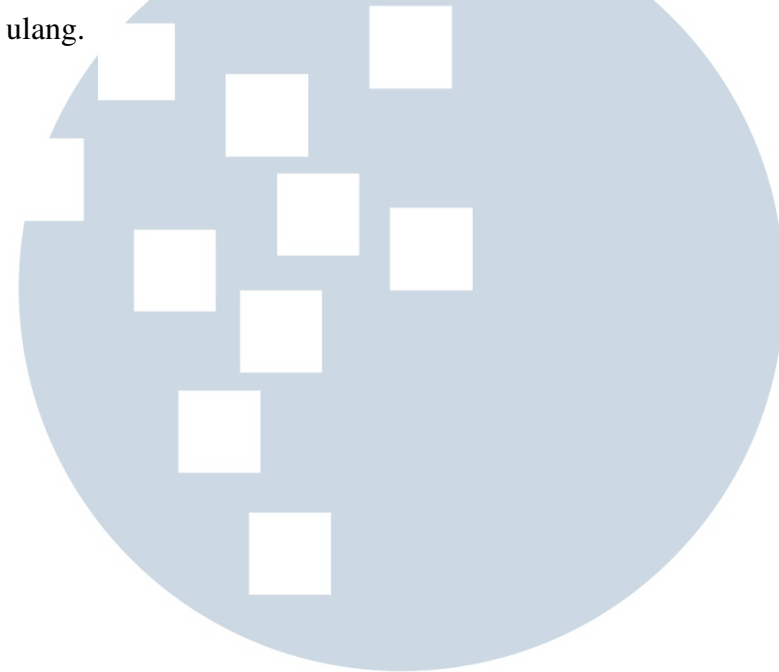
C.5 Kesimpulan Iterasi 1

Pendekatan *field-level locking* memberikan dasar yang kokoh untuk sistem kolaboratif yang aman dan efisien. Sistem ini menghindari konflik data dengan membatasi akses terhadap kolom tertentu secara eksklusif per pengguna, namun tetap memungkinkan kolaborasi dalam satu modul secara simultan. Keberhasilan dari iterasi ini membuka jalan untuk eksplorasi dan perancangan iterasi berikutnya sesuai kebutuhan pengguna yang terus berkembang.

D Iterasi 2: *Page-Level Locking*

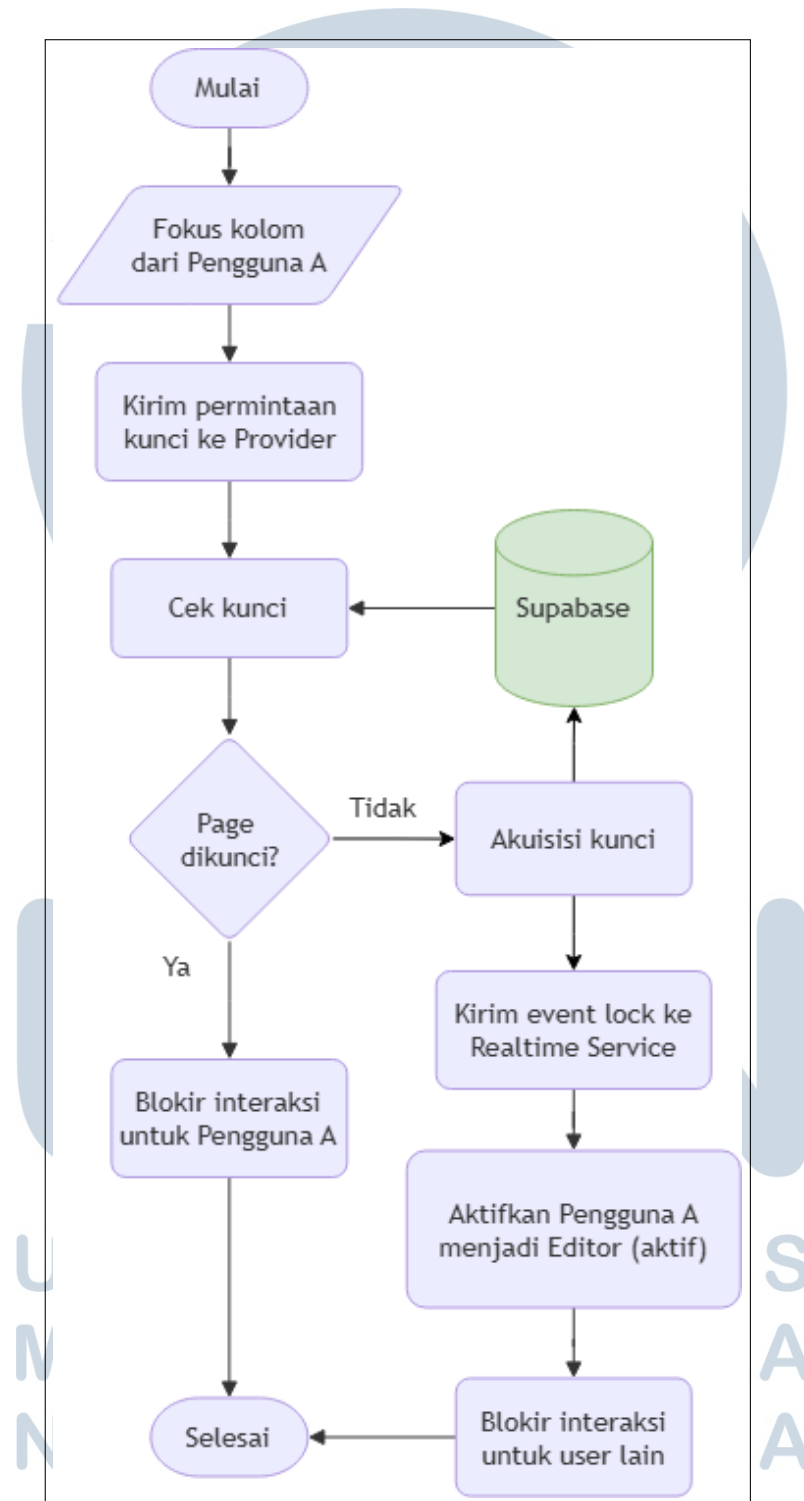
Pada iterasi kedua, fokus bergeser dari eksperimen kolaborasi real-time multi-pengguna pada level kolom ke pendekatan yang lebih konservatif: penguncian pada level page secara keseluruhan. Pergeseran ini dipicu oleh kekhawatiran dari pihak manajemen terkait kompleksitas teknis, kebutuhan akan integritas data yang lebih tinggi, serta potensi konflik sinkronisasi jika kolaborasi granular tetap diterapkan tanpa pengujian menyeluruh.

Alih-alih mengembangkan lebih lanjut sistem dari iterasi pertama, implementasi pada iterasi ini dilakukan melalui refactoring besar-besaran dari sisi arsitektur frontend, dengan fokus pada abstraksi yang lebih baik, penggunaan Context Provider, dan komponen-komponen yang lebih modular dan dapat digunakan ulang.



UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

D.1 Alur Interaksi Pengguna



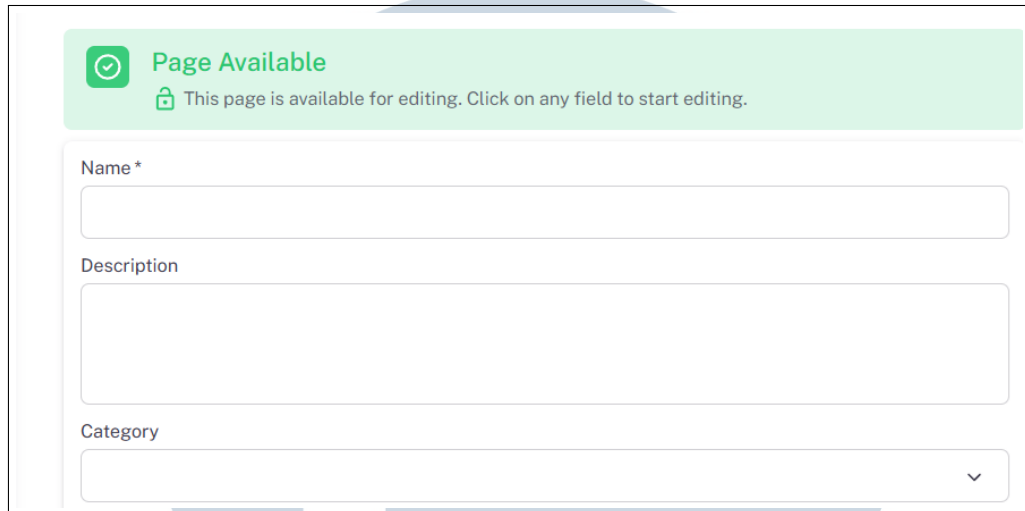
Gambar 3.7. User Flow pada Iterasi 2 Modul Kolaborasi Real-Time

Diagram alur pada Gambar 3.7 menjelaskan proses penguncian kolom dan sinkronisasi status antara pengguna yang aktif mengedit dan pengguna lain yang sedang mengamati. Proses ini memastikan hanya satu pengguna yang dapat mengedit konten dalam satu waktu, sementara pengguna lain mendapatkan tampilan baca-saja. Berikut adalah langkah-langkah rinci dari alur interaksinya:

1. Pengguna A mengarahkan fokus ke salah satu kolom dalam halaman formulir.
2. Aplikasi client mengirim permintaan penguncian ke penyedia konteks (Provider).
3. Provider meneruskan permintaan tersebut ke database untuk melakukan akuisisi kunci.
4. Jika kolom belum dikunci dan permintaan berhasil, sistem akan memicu siaran status melalui layanan real-time.
5. Pengguna A menerima konfirmasi bahwa ia telah menjadi editor aktif dan dapat mengubah isi kolom.
6. Secara bersamaan, pengguna B menerima informasi bahwa kolom telah dikunci oleh pengguna A.
7. UI pengguna B diperbarui sehingga semua kolom menjadi tidak aktif dan hanya dapat dilihat (viewer mode).

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

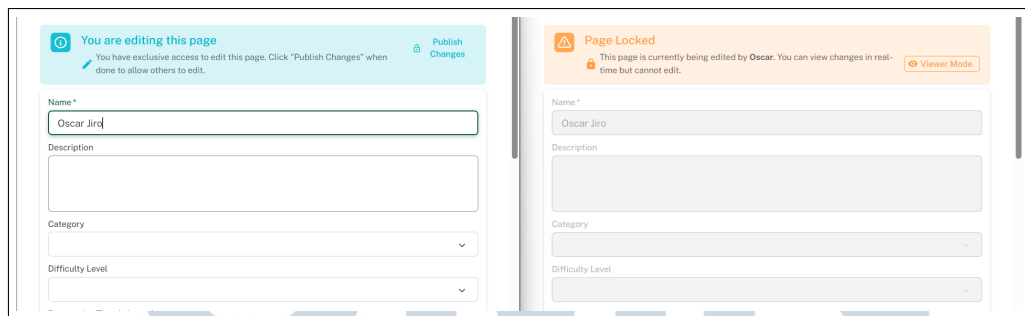
D.2 Tampilan dan Simulasi Kolaborasi Antar Pengguna



The screenshot shows a green banner at the top with a checkmark icon and the text "Page Available". Below the banner, there is a lock icon and the text "This page is available for editing. Click on any field to start editing." The form below contains three input fields: "Name *" (a text box), "Description" (a larger text box), and "Category" (a dropdown menu).

Gambar 3.8. Tampilan UI Iterasi 2 dengan Status Penguncian Halaman

Gambar 3.8 menunjukkan cuplikan UI pengguna dari modul kolaborasi iterasi 2 yang sekarang memiliki *page-level locking*. Terlihat sebuah *page lock status* di atas form yang masih tersedia saat pengguna membuka modul.



The image shows two side-by-side screenshots of the collaboration interface. The left screenshot shows the "You are editing this page" status with a blue banner, a pencil icon, and the text "You have exclusive access to edit this page. Click 'Publish Changes' when done to allow others to edit." The right screenshot shows the "Page Locked" status with an orange banner, a lock icon, and the text "This page is currently being edited by Oscar. You can view changes in real-time but cannot edit." Both screenshots show the same form with fields for "Name *" (containing "Oscar Jiro"), "Description", "Category", and "Difficulty Level".

Gambar 3.9. Simulasi Kolaborasi: Kiri sebagai Editor, Kanan dalam Mode Terkunci

Gambar 3.9 memperlihatkan simulasi dua pengguna dalam satu halaman. Ketika pengguna kiri aktif menyunting, status halaman menunjukkan warna biru, sementara pengguna kanan melihat status oranye dengan status halaman sedang dikunci oleh pengguna lain.

D.3 Kesimpulan Iterasi 2

Implementasi iterasi 2 ini berhasil dilakukan dan sesuai dengan kebutuhan user yang didefinisikan. POC modul kolaborasi real-time ini dinyatakan layak untuk

dilanjutkan ke tahap pengembangan produksi untuk penyempurnaan dan integrasi ke sistem utama.

Pendekatan iterasi 2 ini memberikan sejumlah keuntungan dibandingkan iterasi sebelumnya, yakni konsistensi data lebih tinggi, penyederhanaan sinkronisasi, arsitektur yang lebih modular, dan visual feedback yang lebih jelas.

Namun demikian, iterasi ini juga memiliki keterbatasan, seperti menurunnya produktivitas saat banyak pengguna ingin berkolaborasi dalam waktu bersamaan. Oleh karena itu, sistem tetap dirancang fleksibel untuk memungkinkan pengembangan model optimistic locking atau penguncian lebih granular di masa mendatang.

3.3.3 Rich Text Editor dengan Kolaborasi Real-Time

Rich text editor dengan kolaborasi real-time ini merupakan bagian dari sistem kolaborasi real-time di ELN yang memungkinkan lebih dari satu user menyunting sebuah dokumen dengan konten yang kompleks secara bersamaan. Implementasi text editor ini berfokus pada sinkronisasi dokumen tanpa konflik dengan tetap dapat mendukung kebutuhan fitur-fitur user dengan pendekatan berbasis CRDT dan komunikasi dua arah menggunakan WebSocket.

A Kebutuhan User

Secara umum, rich text editor kolaboratif ini diharapkan dapat menyokong kolaborasi lebih dari user dalam proses pelaporan dan dokumentasi pekerjaan. Berikut adalah kebutuhan user yang telah diperoleh dari divisi PM dan didiskusikan dengan divisi SWE terkait implementasi text editor kolaboratif:

1. Text editor mampu menyokong berbagai kebutuhan seperti formatting teks, anotasi gambar, dan visualisasi data.
2. Lebih dari satu user dapat menyunting dokumen secara bersamaan tanpa konflik.
3. User dapat melihat keberadaan user lain.
4. Konten dokumen disimpan pada database dan dapat digunakan ulang di bagian lain aplikasi.
5. Konten dokumen dapat di-export menjadi PDF yang mempertahankan formatting.

B Riset Teknologi dan Implementasi

Untuk mendukung kebutuhan pembuatan rich text editor dengan kolaborasi real-time, dilakukan riset terhadap berbagai teknologi yang dapat mendukung integrasi fitur lanjutan seperti penyisipan grafik, anotasi gambar, dan kolaborasi simultan.

Dalam aspek text editor, beberapa pustaka modern dievaluasi berdasarkan fleksibilitas, ekosistem ekstensi, kemudahan integrasi, dan dukungan terhadap kolaborasi real-time. Solusi yang dipilih merupakan framework penyunting teks modern berbasis arsitektur modular yang menawarkan pengalaman pengembangan yang baik sekaligus tingkat kustomisasi tinggi. Framework ini juga mendukung integrasi dengan teknologi CRDT untuk kolaborasi yang menjadi salah satu pertimbangan utama.

Alternatif lainnya mencakup pustaka-pustaka dengan fleksibilitas serupa namun memiliki tingkat abstraksi yang lebih rendah, sehingga memerlukan usaha tambahan untuk membangun fitur tingkat lanjut. Sementara itu, ada pula pustaka lain yang bersifat agnostik terhadap framework dan memiliki fitur kaya untuk konten interaktif, tetapi tidak menyediakan dukungan kolaborasi secara langsung, sehingga tidak diprioritaskan.

Untuk kebutuhan anotasi gambar, digunakan pustaka grafis berbasis kanvas yang memungkinkan manipulasi elemen visual secara interaktif. Pustaka ini dipilih karena ringan, mendukung rendering real-time, serta memiliki ekosistem yang mendukung berbagai fitur seperti zoom, drag, serta penambahan elemen bebas. Alternatif lainnya juga sempat ditinjau, namun memiliki kompleksitas lebih tinggi dalam hal integrasi dan performa rendering.

Untuk kebutuhan visualisasi data, dipilih pustaka grafik yang ringan, interaktif, dan memiliki dukungan luas untuk berbagai jenis grafik seperti garis, batang, dan area. Pustaka ini memungkinkan integrasi yang mulus dengan antarmuka pengguna dan memiliki dokumentasi yang baik. Alternatif pustaka lainnya sempat dipertimbangkan, namun dinilai kurang fleksibel dalam konteks kebutuhan spesifik proyek.

Untuk kebutuhan, dipilih pustaka yang dapat me-render HTML tanpa konteks browser dan mengubahnya menjadi format seperti PDF. Pustaka ini dipilih karena ringan dan memiliki banyak fitur yang mendukung pengambilan snapshot konten suatu halaman dengan kontrol yang tinggi. Alternatif pustaka lainnya sempat dipertimbangkan yang lebih *straightforward* dalam ekosistem React, namun

beberapa elemen UI yang seharusnya tidak di-render seperti tanda keberadaan user ikut dalam hasil akhir rendering.

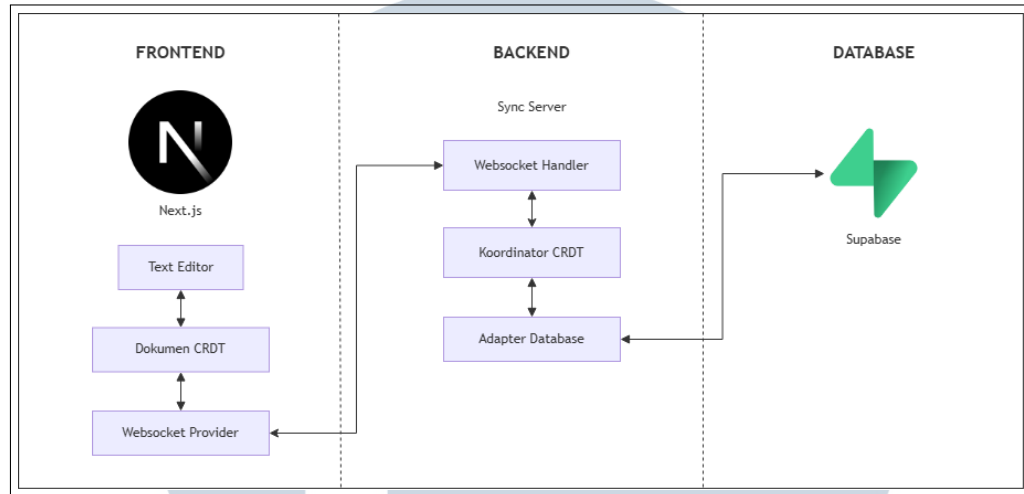
Dari sisi kolaborasi real-time, backend CRDT open-source dipilih sebagai dasar sinkronisasi dokumen. Beberapa pendekatan implementasi dieksplorasi untuk menghubungkan text editor dengan server CRDT:

1. Provider layanan kolaborasi bawaan: Solusi ini menawarkan integrasi tercepat dan paling sederhana, namun menyimpan data dokumen pada layanan pihak ketiga berbayar, yang tidak sejalan dengan kebutuhan penyimpanan internal dan keamanan data perusahaan.
2. Provider komunitas berbasis open-source: Beberapa inisiatif open-source yang menawarkan integrasi langsung dengan infrastruktur backend perusahaan telah diuji, namun sebagian besar masih berada dalam tahap eksperimental, tidak stabil, dan belum mendukung kebutuhan produksi.
3. Provider open-source yang lebih terbukti: Solusi ini memerlukan konfigurasi server tambahan dan penyesuaian untuk menyimpan data secara lokal, namun dinilai lebih ringan, fleksibel, dan scalable dalam jangka panjang. Pendekatan ini akhirnya dipilih karena memungkinkan pengendalian penuh atas proses sinkronisasi dan penyimpanan dokumen ke sistem internal perusahaan.

Melalui riset ini, diambil keputusan implementasi yang mempertimbangkan keseimbangan antara developer experience, performa sistem, keamanan data, dan potensi skalabilitas untuk pengembangan fitur kolaborasi lebih lanjut di masa mendatang.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A

B.1 Arsitektur Sistem



Gambar 3.10. Diagram Arsitektur Sistem Text Editor Kolaboratif

Arsitektur sistem ini terdiri dari tiga komponen utama: sisi client (frontend), sisi server (backend), dan database yang masing-masing memiliki peran spesifik dalam memastikan kolaborasi berlangsung secara sinkron, efisien, dan konsisten. Gambar 3.10 menunjukkan sebuah component diagram yang menjelaskan komponen-komponen utama dalam sistem ini dan hubungan antar komponennya.

B.1.1 Arsitektur Client

Pada sisi client, terdapat beberapa komponen utama yang membentuk ekosistem kolaborasi real-time:

1. Text Editor: UI utama pengguna dalam melakukan penyuntingan dokumen. Editor ini dikonfigurasi dengan sejumlah ekstensi yang mendukung fitur-fitur kolaboratif dan penyuntingan teks kaya.
2. Dokumen CRDT: Berfungsi sebagai struktur data replikatif tanpa konflik yang menyimpan isi dokumen secara lokal dan mendukung sinkronisasi otomatis dengan server serta klien lain.
3. WebSocket Provider: Berperan sebagai penghubung komunikasi dua arah antara klien dan server menggunakan protokol WebSocket, sehingga perubahan yang dilakukan pengguna dapat langsung disebarkan secara real-time.

4. Keberadaan & Posisi Cursor: Modul yang memantau dan menyebarkan informasi kehadiran pengguna lain dalam dokumen, termasuk posisi kursor dan area seleksi teks, guna mendukung kolaborasi yang lebih intuitif.

Ketika pengguna melakukan perubahan pada dokumen, perubahan tersebut diterapkan ke dalam dokumen CRDT lokal, kemudian dikirimkan ke server melalui WebSocket. Informasi kehadiran pengguna juga dikirimkan secara paralel, sehingga pengguna lain dapat melihat siapa yang sedang aktif dan di bagian mana mereka sedang bekerja.

B.1.2 Arsitektur Server

Pada sisi server, terdapat lapisan sinkronisasi yang bertanggung jawab atas koordinasi sesi kolaborasi real-time:

1. WebSocket Handler: Menangani koneksi WebSocket yang masuk dari klien, termasuk autentikasi, manajemen sesi, serta transmisi data.
2. Koordinator CRDT: Bertugas menyatukan perubahan dari berbagai klien, menjaga konsistensi dokumen secara global, serta meneruskan perubahan ke semua peserta kolaborasi yang relevan.
3. Adapter Database: Mengelola interaksi dengan basis data untuk menyimpan dan mengambil salinan dokumen secara periodik atau saat sesi berakhir.

Server dirancang agar mampu menangani perubahan dokumen secara efisien melalui pendekatan berbasis delta, sehingga hanya perubahan yang relevan yang dikirimkan, bukan keseluruhan isi dokumen. Selain itu, server juga mengimplementasikan sistem event handling untuk memantau peristiwa penting, seperti koneksi dan pemutusan klien, pemuatan dokumen, serta modifikasi dokumen.

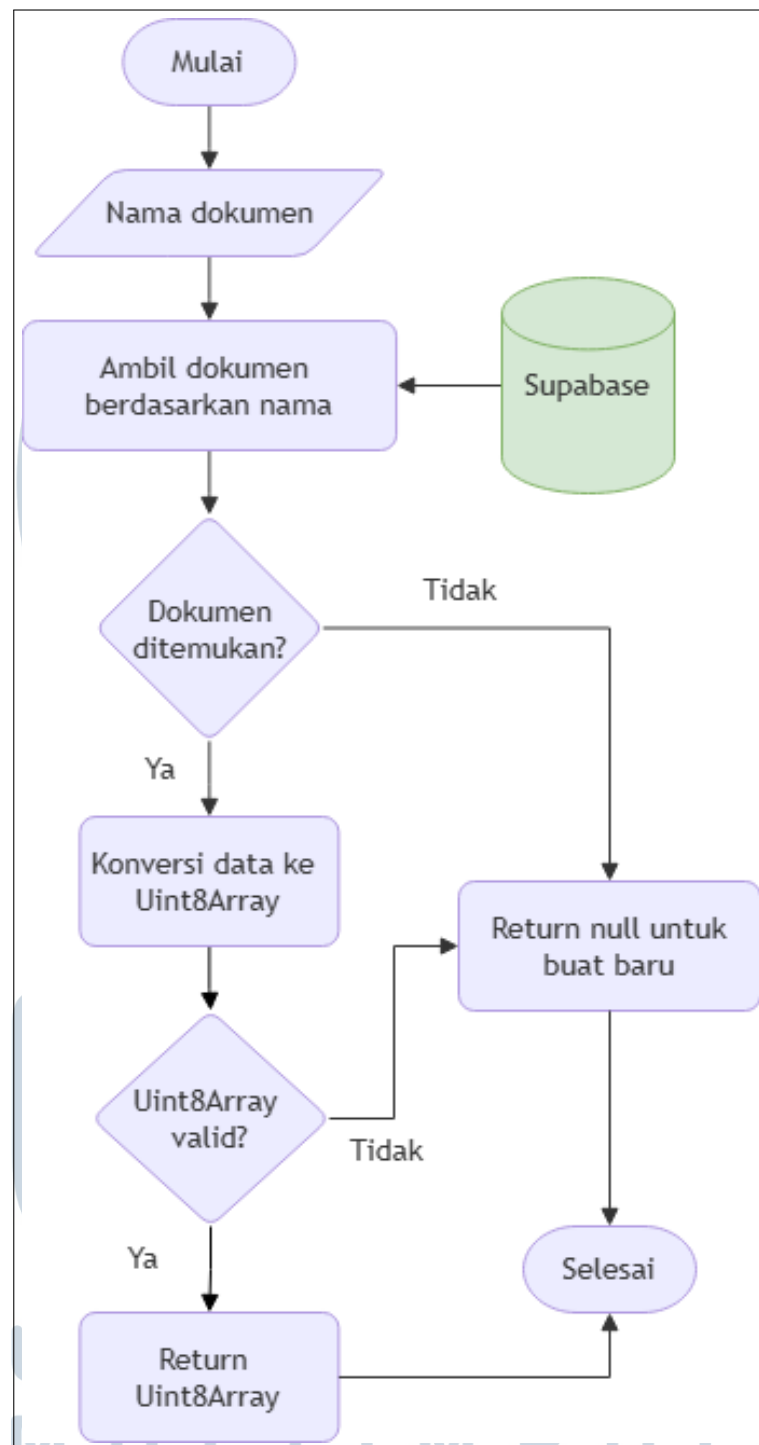
B.1.3 Integrasi Database

Sebagai lapisan penyimpanan permanen, sistem ini menggunakan layanan database yang mendukung komunikasi melalui API. Dokumen yang sedang dikolaborasikan disimpan dalam bentuk terenkripsi atau dalam format terstruktur khusus, dan diambil kembali saat dibutuhkan (misalnya ketika sesi kolaborasi baru dimulai).

Konten dokumen pun disimpan dalam bentuk data biner yang kemudian dikodekan ke dalam format base64. Penyimpanan ini dilakukan dengan chunking untuk menangani dokumen besar secara efisien. Proses penyimpanan mencakup validasi status dokumen, transformasi format data (misalnya ke format biner atau base64), serta operasi upsert untuk efisiensi.

Dalam komunikasinya dengan server sinkronisasi, terdapat dua lifecycle hooks yang berkaitan secara langsung dengan database, yakni onLoad yang terjadi setelah koneksi antara client dengan server berhasil dijalin, serta onChange setiap kali terjadi update dari client.



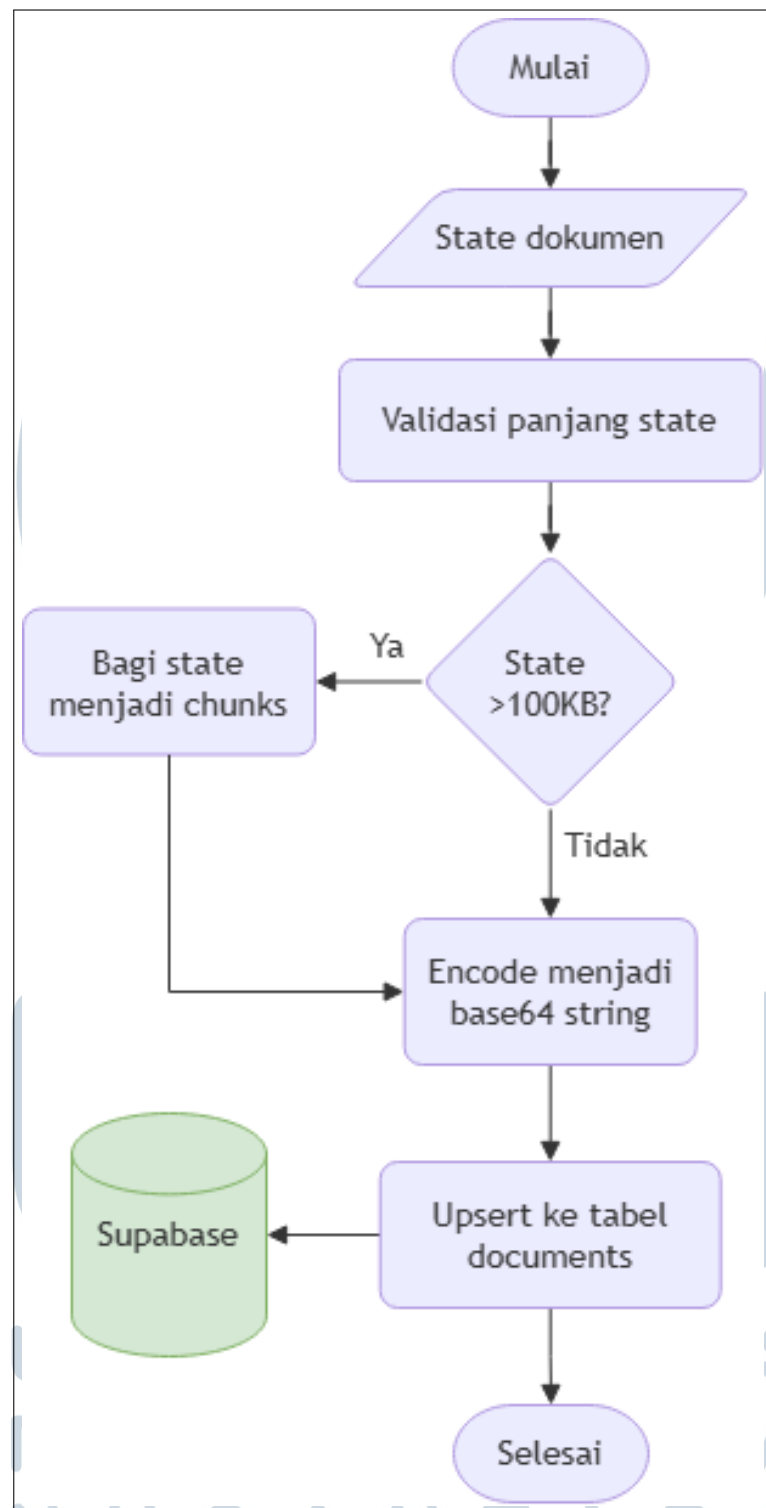


Gambar 3.11. Flowchart Lifecycle Hook onLoad Untuk Mengambil Dokumen

Gambar 3.11 menunjukkan alur hook `onLoad` setelah koneksi terjalin untuk mengambil dokumen berdasarkan nama dokumen yang ingin diambil. Berikut adalah rincian alur tersebut:

1. Sistem mengecek apakah dokumen dengan nama tertentu tersedia di Supabase.
2. Kemudian dilakukan percabangan:
 - (a) Jika dokumen tidak ditemukan, maka fungsi akan mengembalikan nilai `null`. Hal ini akan memicu sistem untuk membuat dokumen baru.
 - (b) Jika dokumen ditemukan, sistem akan mencoba mengonversi data dari format yang disimpan (misal array, string base64) menjadi bentuk `Uint8Array`.
3. Setelah data berhasil dikonversi, dilakukan validasi terhadap isi dari `Uint8Array`:
 - (a) Jika valid (`length > 0`), maka `Uint8Array` dikembalikan sebagai hasil `fetch`.
 - (b) Jika tidak valid atau kosong, maka sistem mengembalikan `null`.





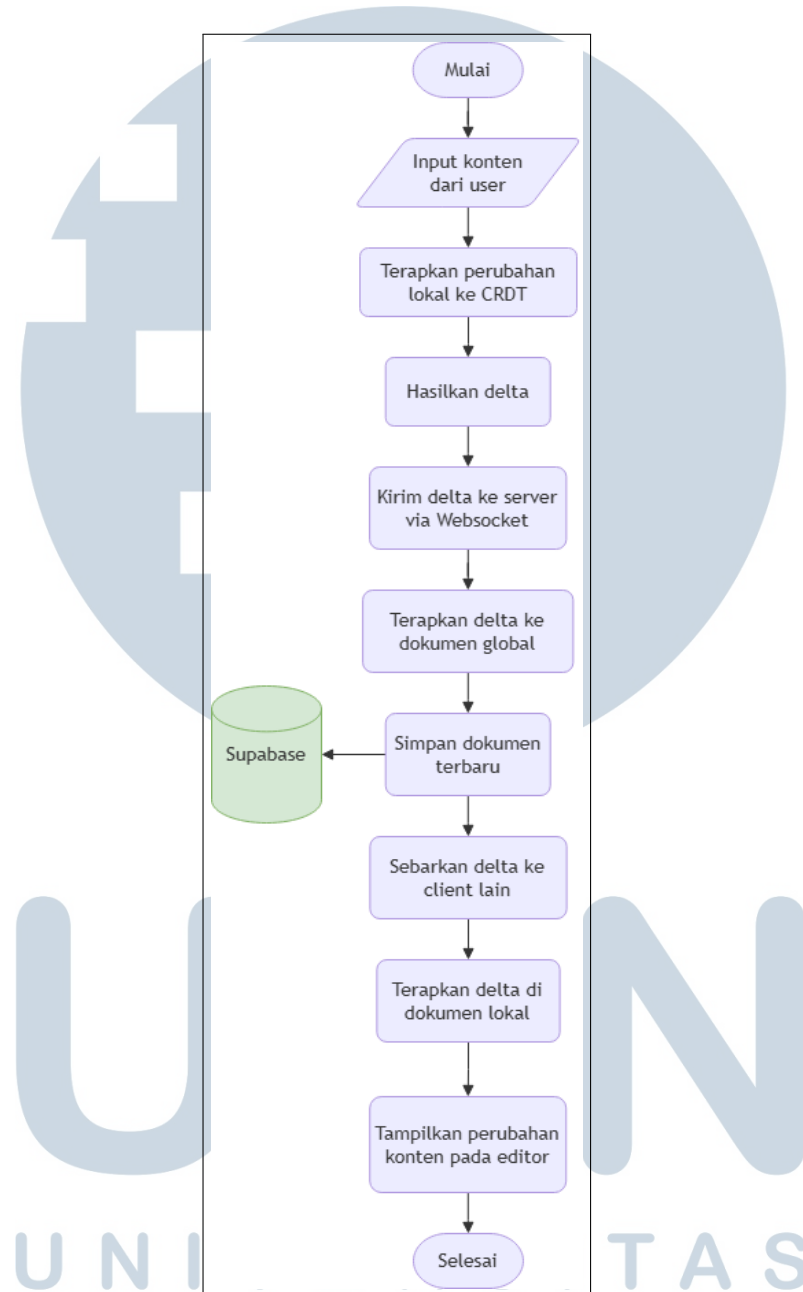
Gambar 3.12. Flowchart Lifecycle Hook onChange Untuk Update Dokumen

Gambar 3.12 menunjukkan alur hook onChange setiap kali ada perubahan dari client. Berikut adalah rincian alur tersebut:

1. Sistem memvalidasi panjang dari `state` dokumen yang akan disimpan.
2. Percabangan dilakukan berdasarkan ukuran data:
 - (a) Jika ukuran `state` lebih dari 100KB, sistem akan memproses data dalam bentuk chunk berukuran 8KB, kemudian melakukan encoding ke dalam format `base64` per chunk.
 - (b) Jika ukuran lebih kecil dari 100KB, sistem langsung mengonversi seluruh data menjadi string `base64` tanpa pembagian.
3. Hasil encoding digabungkan menjadi satu string `base64`.
4. Sistem melakukan `upsert` ke database untuk menyimpan data tersebut.
5. Setelah penyimpanan, dilakukan pengecekan terhadap hasil operasi:
 - (a) Jika tidak terjadi error, sistem mencatat bahwa penyimpanan berhasil.
 - (b) Jika terjadi error, maka error tersebut dicatat di log untuk dianalisis.



B.2 Alur Kolaborasi Real-Time



Gambar 3.13. Flowchart Sinkronisasi dalam Interaksi dengan Text Editor

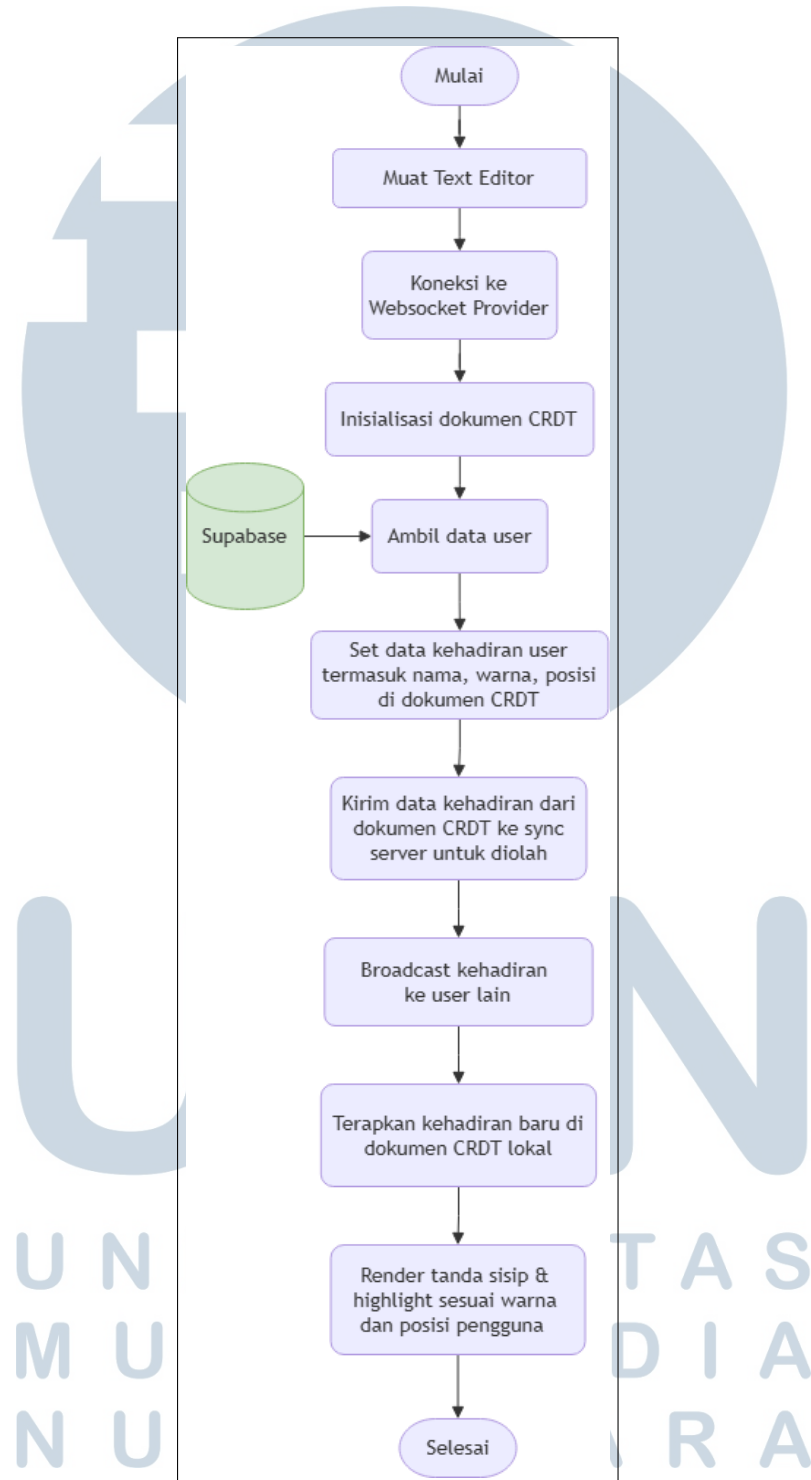
Diagram alur pada Gambar 3.13 menggambarkan proses sinkronisasi data secara real-time pada sistem text editor kolaboratif. Proses ini memastikan setiap perubahan yang dilakukan oleh pengguna akan segera diterapkan secara lokal, dikirimkan ke server, dan disebar ke pengguna lain melalui komunikasi dua arah berbasis WebSocket. Mekanisme ini didukung oleh struktur data CRDT yang

menjamin konsistensi dokumen di seluruh klien. Berikut adalah langkah-langkah rinci dari alur tersebut:

1. Pengguna mengetik karakter pada text editor.
2. Editor langsung menerapkan perubahan tersebut pada struktur data CRDT di sisi client.
3. Perubahan tersebut dikemas dalam bentuk delta (hanya bagian yang berubah).
4. Delta kemudian dikirim ke server melalui koneksi WebSocket.
5. Server bertindak sebagai koordinator yang menerapkan delta ke dokumen global.
6. Dokumen yang telah diperbarui disimpan ke database.
7. Server juga menyebarkan delta tersebut ke semua client lain yang sedang terhubung.
8. Setiap client yang menerima delta akan menerapkannya ke CRDT lokal masing-masing.
9. Text editor memperbarui tampilan UI berdasarkan perubahan yang diterima.



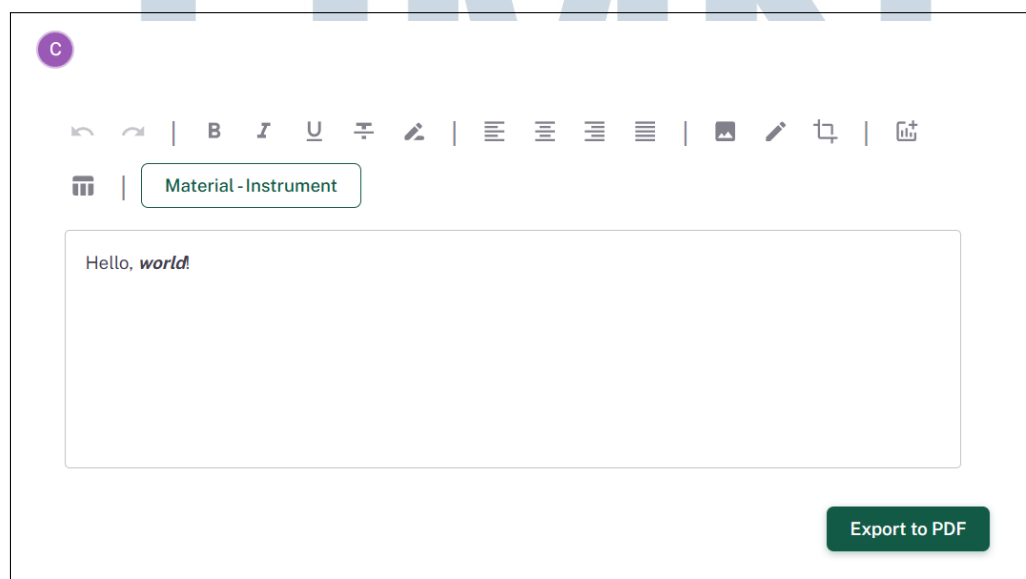
B.3 Fitur Kehadiran di Text Editor



Gambar 3.14. Flowchart Pelacakan Kehadiran Pengguna di Text Editor

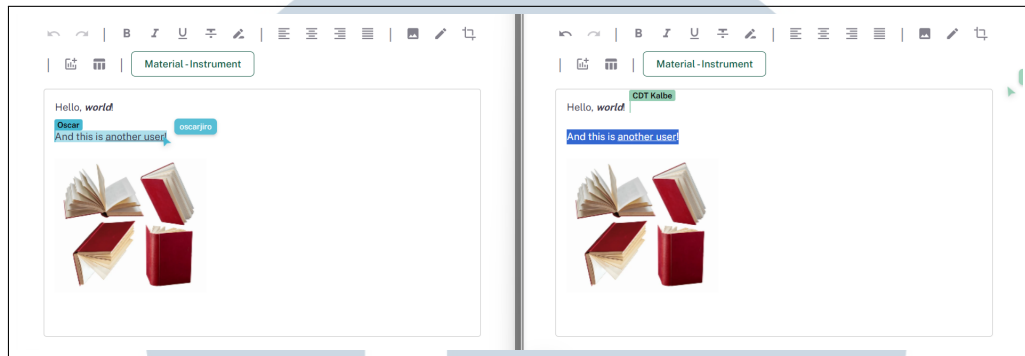
Text editor ini dilengkapi fitur kehadiran user yang menampilkan siapa saja yang sedang online, serta penanda posisi kursor dan seleksi teks dari pengguna lain secara real-time. Dalam mencapai hal ini, Gambar 3.14 menjelaskan alur dari awal kemunculan user hingga rendering fitur kehadiran ini yang dirincikan sebagai berikut:

1. Saat editor dimuat, pengguna menginisialisasi koneksi ke Provider.
2. Dokumen CRDT dibuat untuk sinkronisasi konten.
3. Sistem mengambil data user seperti nama dan warna dari layanan autentikasi.
4. Data tersebut ditetapkan sebagai data kehadiran pada Provider yang dilengkapi dengan posisi dan informasi lainnya setelah ditetapkan. Setiap item berisi informasi seperti nama dan warna unik untuk identifikasi visual.
5. Provider mengirimkan informasi kehadiran ke server sinkronisasi.
6. Server kemudian menyebarkan perubahan awareness ke semua klien yang terhubung pada dokumen yang sama.
7. Klien lain menerima event perubahan, termasuk posisi kursor, seleksi teks, serta atribut visual seperti warna.
8. UI masing-masing user merender tanda sisip, seleksi teks, dan label nama berdasarkan informasi user lain yang diterima.



Gambar 3.15. Preview Text Editor Kolaboratif dengan Indikator Kehadiran User

Gambar 3.15 menunjukkan tampilan awal text editor yang baru dihadiri satu pengguna. Presence tracker di atas text editor menunjukkan hanya satu user yang online yakni diri sendiri dan text editor menampilkan konten seperti biasanya.



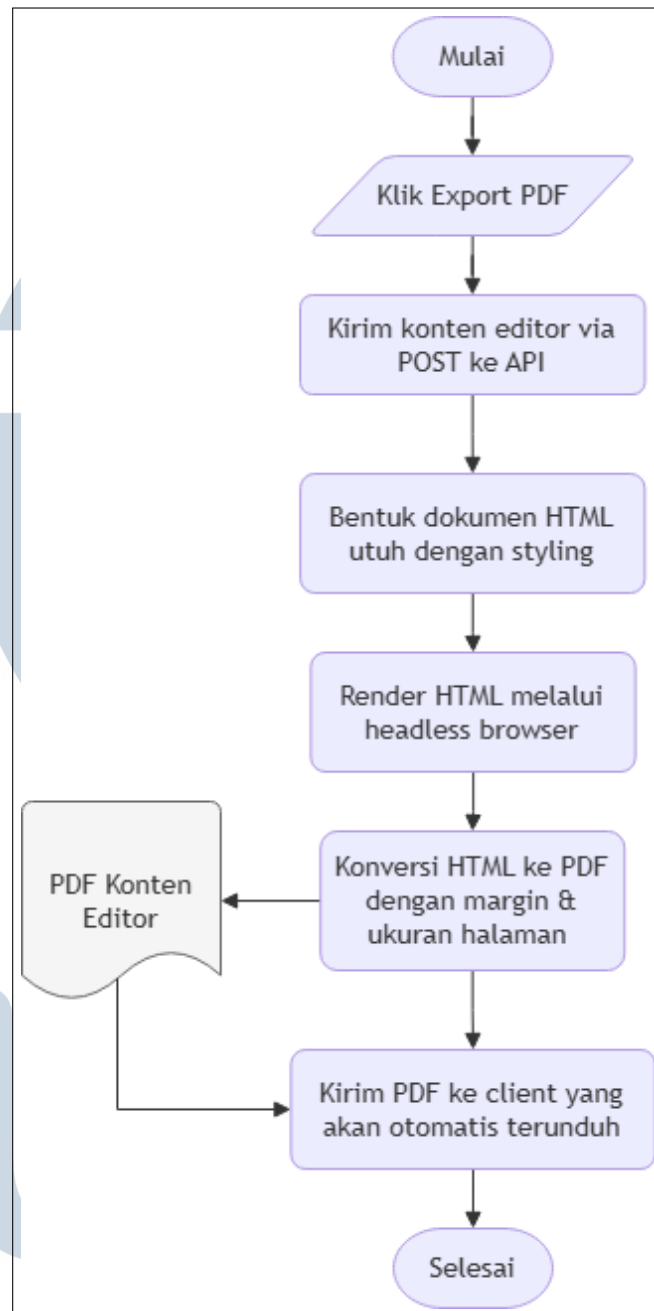
Gambar 3.16. Simulasi Kolaborasi Multi-User dengan Tanda Sisipan Warna dan Konten Kompleks

Gambar 3.16 menunjukkan simulasi ketika lebih ada satu user yang menyunting text editor. Text editor menunjukkan tanda sisipan berwarna dan seleksi teks berwarna yang ditampilkan untuk setiap pengguna, membantu menunjukkan posisi dan aksi pengguna lain secara efektif. Text editor juga terlihat mendukung konten kompleks seperti gambar dan grafik secara optimal.

B.4 Fitur Export Konten ke PDF

Dalam pengembangan text editor ini, dibutuhkan fitur tambahan untuk mendukung proses dokumentasi dan distribusi hasil akhir, yakni fitur export dokumen ke format PDF secara otomatis dari konten editor. Dengan fitur ini, user dapat mengunduh dokumen dengan tampilan yang terstruktur dan konsisten, termasuk elemen seperti gambar, tabel, label, dan grafik.

Fitur export ini dibangun pada sisi server, memanfaatkan API route yang dirancang khusus untuk menerima konten editor dalam bentuk HTML dan CSS, sehingga mengurangi beban di client. API ini kemudian memproses konten HTML menggunakan browser headless (tanpa konteks browser) untuk me-render dan mengubahnya ke dalam format PDF.

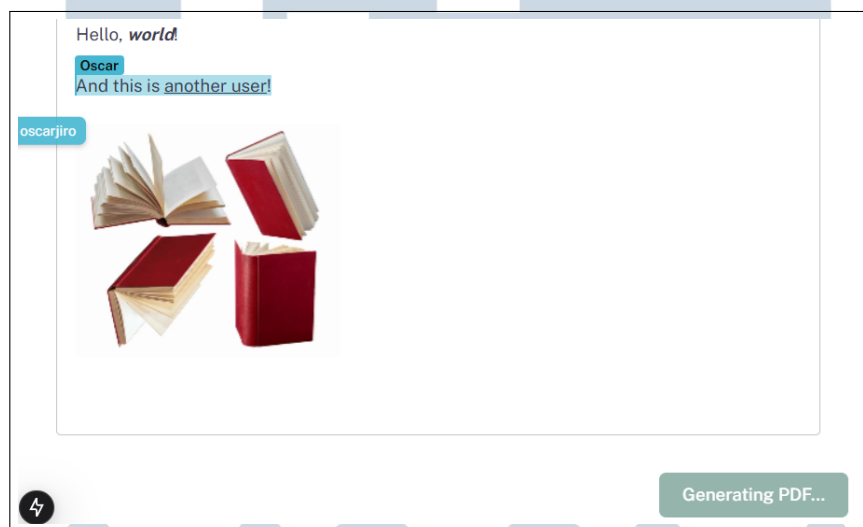


Gambar 3.17. Alur Proses Export Konten Text Editor ke PDF

Berikut ini alur teknis dari proses export PDF sesuai Gambar 3.17:

1. User mengklik tombol "Export to PDF" pada text editor.
2. Client mengirim konten HTML dan CSS pada text editor saat itu melalui permintaan POST ke API backend.

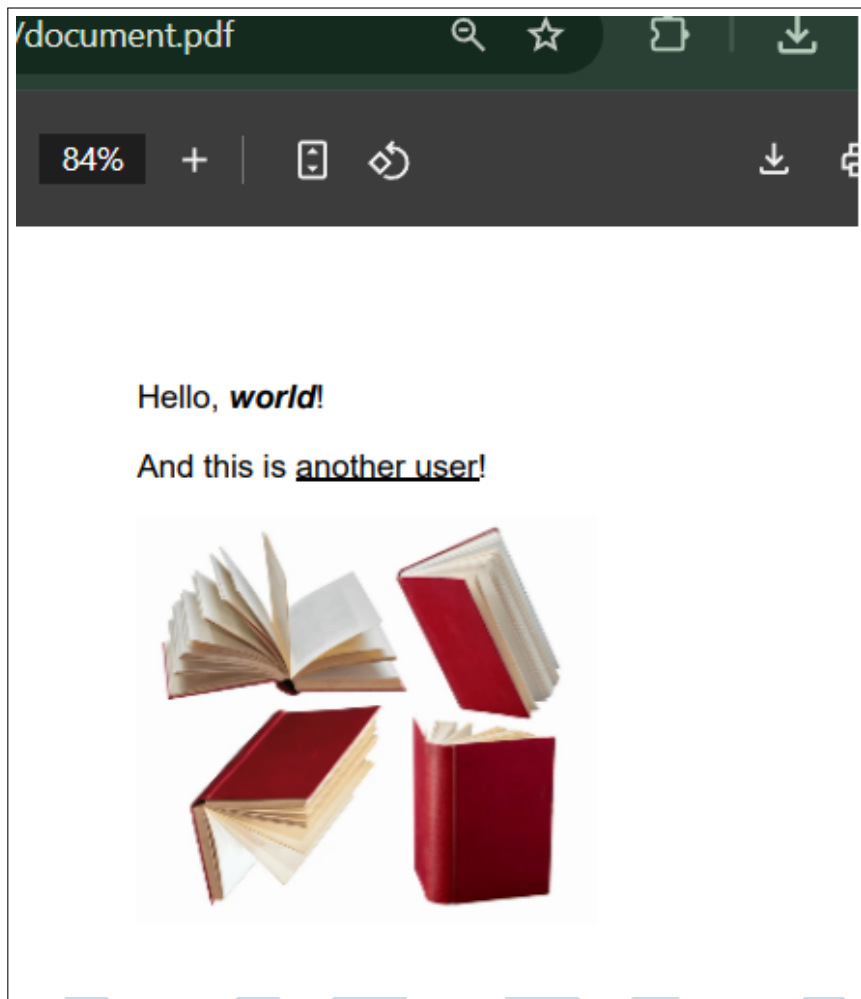
3. API route menerima POST request, membentuk dokumen HTML utuh dengan gaya yang disesuaikan.
4. Browser headless diluncurkan di server untuk me-render dokumen HTML tersebut (vector-based representation, tidak dirasterisasi).
5. Setelah proses render selesai, dokumen dikonversi menjadi PDF dengan pengaturan ukuran halaman dan margin tertentu.
6. PDF dikembalikan ke client dan diunduh secara otomatis oleh user.



Gambar 3.18. Tampilan Text Editor Dalam Progress Export PDF

Gambar 3.18 menunjukkan tampilan text editor dan tombol "Export to PDF" yang sedang loading setelah diklik user. Selama belum mendapat respons dari permintaan POST, tombol dinonaktifkan dan bertuliskan "Generating PDF..." untuk memberikan user visual feedback.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A



Gambar 3.19. Hasil Export PDF dari Konten Text Editor

Gambar 3.19 menunjukkan hasil export konten ke file PDF yang sudah berhasil dan terunduh. Terlihat formatting dari konten editor sebenarnya konsisten dan dipertahankan. Kebutuhan formatting lebih lanjut dari hasil export PDF (seperti template laporan) belum ditentukan dalam fase POC ini, tetapi dapat diintegrasikan dengan lancar di API route tersebut.

B.5 Keunggulan Sistem

Sistem ini dirancang untuk mendukung:

1. Skalabilitas: Tidak ada titik pusat konflik. Setiap client bekerja secara independen dengan tetap menjaga konsistensi.

2. Performa Tinggi: Update yang dikirim hanya berupa delta, bukan keseluruhan dokumen.
3. Reliabilitas: Operasi offline tetap tersimpan, dan akan disinkronkan saat koneksi kembali normal.

B.6 Kesimpulan

POC rich text editor dengan kolaborasi real-time ini berhasil diimplementasikan dan memenuhi kebutuhan user untuk kolaborasi dengan sinkronisasi konten kompleks tanpa konflik, persistensi konten yang konsisten, dan fitur export PDF yang fungsional. POC ini dinyatakan layak untuk dilanjutkan ke tahap pengembangan berikutnya untuk penyempurnaan dan integrasi ke sistem utama.

Text editor ini mengimplementasikan arsitektur yang memungkinkan sinkronisasi instan dan konflik minimal saat banyak pengguna mengedit dokumen yang sama. Dengan pemisahan tanggung jawab antara client, server, dan database, sistem ini menawarkan skalabilitas dan reliabilitas tinggi.

3.4 Kendala dan Solusi yang Ditemukan

Selama proses pembuatan sistem fitur kolaborasi real-time, berbagai tantangan teknis berhasil diidentifikasi dan diatasi secara bertahap melalui pendekatan iteratif dan evaluasi menyeluruh. Berikut ini adalah rangkuman kendala utama yang dihadapi serta solusi yang diterapkan:

3.4.1 Sinkronisasi State Awal Modul Kolaboratif

Pada iterasi pertama modul kolaborasi real-time, penggunaan WebSocket memungkinkan pelacakan kehadiran pengguna serta posisi kursor secara langsung. Namun, karena tidak ada penyimpanan state awal di database, pengguna baru yang bergabung tidak dapat langsung melihat isi terkini hingga ada aktivitas dari pengguna lain. Pendekatan awal yang ingin meminimalisasi akses dan penyimpanan ke database berujung menimbulkan kompleksitas yang tidak diperlukan serta tidak menjamin konsistensi state modul.

Solusi: Strategi ini digantikan dengan sistem langganan perubahan database. Ketika halaman dimuat, state awal diambil dari database, sementara WebSocket tetap digunakan untuk pelacakan waktu nyata. Perubahan disimpan secara berkala menggunakan teknik debounce untuk menjaga performa.

3.4.2 Sinkronisasi Database Real-Time

Pada iterasi kedua modul kolaborasi real-time, kendala terjadi ketika data seperti *page lock* dan kolom tidak tersinkron secara waktu nyata. Setelah debugging secara ekstensif, ditemukan akibat keterbatasan oleh karena pemberlakuan filtering Supabase dan kebijakan Row-Level Security (RLS) yang menghambat deteksi perubahan, terutama pada operasi `DELETE`.

Solusi: Sistem langganan diperbarui dengan menghilangkan filter bawaan dan menggantinya dengan filtering manual. Konfigurasi RLS dan replikasi juga disesuaikan agar tetap aman namun mampu menangani perubahan termasuk penghapusan data.

3.4.3 Integrasi Kolaborasi Real-Time pada Text Editor

Kesulitan muncul ketika mencoba mengintegrasikan penyimpanan kolaboratif ke Supabase karena keterbatasan kompatibilitas provider yang tersedia, di mana kebanyakan masih bersifat eksperimental dan tidak stabil.

Solusi: Implementasi dilakukan menggunakan teknologi open-source yang disesuaikan agar dapat terhubung langsung ke Supabase. Dengan pendekatan ini, sistem tetap mendukung kolaborasi real-time sambil mempertahankan performa dan skalabilitas.

3.4.4 Sinkronisasi Backend Text Editor Kolaboratif

Masalah terjadi ketika client berhasil terhubung ke server namun tidak ada data yang tersinkron, dan tidak terdapat log error untuk membantu proses debugging.

Solusi: Setelah investigasi ekstensif, ditemukan bahwa versi provider tidak sesuai dengan versi server. Pembaruan versi dilakukan dan backend kemudian dikustomisasi agar dapat menyimpan konten secara berkala ke database.

3.4.5 Penyimpanan Konten Text Editor

Konten kompleks seperti gambar, grafik, dan tabel mengalami kegagalan saat disimpan karena proses serialisasi dan deserialisasi yang tidak konsisten, menghasilkan error seperti `Unexpected end of data`.

Solusi: Format penyimpanan diubah dari bentuk array menjadi string berbasis encoding yang lebih stabil. Fungsi pembacaan data diperluas untuk menangani format lama dan baru. Validasi input serta penanganan error ditingkatkan guna mencegah kerusakan data dan memudahkan debugging.

