BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang di PT Kalbe Farma Tbk dilakukan pada departemen Corporate Digital Technology dengan posisi sebagai Software Engineer, yang memiliki tanggung jawab dalam merancang serta mengembangkan aplikasi untuk mendukung kebutuhan internal perusahaan. Proses perancangan dan pengembangan aplikasi dilaksanakan melalui koordinasi dan diskusi bersama pihak pengguna dari departemen terkait, guna memastikan bahwa solusi yang dikembangkan sesuai dengan kebutuhan operasional. Pengawasan dan pembimbingan selama program magang dilakukan oleh Bapak Russell Otniel selaku Software Engineer Officer di PT Kalbe Farma Tbk, guna memastikan bahwa seluruh aktivitas serta proses pengembangan aplikasi berjalan dengan baik, terarah, dan sesuai dengan standar yang ditetapkan.

3.2 Tugas yang Dilakukan

Selama mengikuti program magang di PT Kalbe Farma Tbk pada departemen Corporate Digital Technology (CDT), tanggung jawab yang dijalankan meliputi kegiatan riset, perancangan, dan pengembangan aplikasi internal perusahaan. Kegiatan diawali dengan melakukan riset terhadap fitur yang akan dikembangkan melalui pendekatan *proof of concept* (PoC), kemudian dilanjutkan ke tahap implementasi pengembangan aplikasi, serta diakhiri dengan proses pengujian dan uji coba bersama pengguna yang menjadi sasaran penggunaan aplikasi. Berikut merupakan aktivitas yang dilaksanakan sepanjang periode magang di PT Kalbe Farma Tbk.

1. Mendalami teknologi (*tech stack*) yang digunakan dalam pengembangan aplikasi di lingkungan perusahaan.
2. Melakukan riset dan pengembangan untuk identifikasi *library* editor dokumen pihak ketiga yang relevan dan sesuai dengan kebutuhan pengembangan aplikasi.
3. Berpartisipasi dalam diskusi dengan divisi internal Software Engineer,

Project Management, serta dengan pengguna untuk tahap perencanaan aplikasi.

4. Menyusun dokumentasi teknis untuk aplikasi yang telah diselesaikan, guna mendukung proses pemeliharaan dan pengembangan lanjutan.
5. Melakukan *refactor* pada aplikasi untuk menyederhanakan logika, mengurangi duplikasi kode, dan mematuhi standar penulisan kode yang baik.

3.3 Uraian Pelaksanaan Magang

Berikut merupakan uraian kegiatan yang telah dilakukan selama masa magang sebagaimana tercantum pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang Dilakukan Setiap Minggu

Minggu ke-	Pekerjaan yang Dilakukan
1	• Mengikuti sesi orientasi program magang dan pengenalan lingkungan kerja. • Memahami struktur organisasi dan peran departemen Corporate Digital Technology. • Pengenalan terhadap alur kerja tim pengembangan perangkat lunak. • Pemberian akses ke sistem internal dan tools pengembangan. • Studi mandiri serta presentasi materi HTML, CSS, dan JavaScript.
2	• Studi mandiri serta presentasi Git dan TypeScript. • Menyusun materi *AI Clinic* untuk edukasi AI di lingkungan kerja. • Studi proyek mengenai React.js, TypeScript, dan Material UI.

Minggu ke-	Pekerjaan yang Dilakukan
3	• Presentasi proyek React.js, TypeScript, dan Material UI. • Studi proyek terkait Next.js dan PostgreSQL.
4	• Presentasi dan pendalaman Next.js dan PostgreSQL. • Revisi materi presentasi *AI Clinic*. • Studi proyek terkait Next.js dan Supabase.
5	• Presentasi proyek Next.js dan Supabase. • Studi mendalam terkait Next.js dan Supabase. • Pengenalan terhadap penggunaan Docker untuk pembuatan aplikasi.
6	• Pengerjaan proyek kelompok menggunakan Next.js, Supabase, dan Docker. • Pengenalan dan mempelajari proyek *handoff*.
7	• Presentasi proyek hasil pengerjaan kelompok menggunakan Next.js, Supabase, dan Docker. • Pengenalan dan mempelajari proyek *handoff*.
8	• Mempelajari lebih dalam tentang proyek *handoff*. • Melakukan pengetesan aplikasi *handoff* guna menjamin keandalan sistem.

Minggu ke-	Pekerjaan yang Dilakukan
9	• Mempelajari lebih dalam tentang proyek *handoff*. • Melakukan pengetesan aplikasi *handoff* guna menjamin keandalan sistem.
10	• Mempelajari kode dan alur bisnis proyek *handoff*. • Pengenalan terhadap proyek aplikasi *Regulatory Information Management*.
11	• Mempelajari kode dan alur bisnis proyek *handoff*. • Memahami alur bisnis serta mengidentifikasi fitur yang diperlukan dalam pengembangan proyek aplikasi *Regulatory Information Management*. • Melakukan riset dan pengembangan terhadap *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan.
12	• Melakukan riset dan pengembangan terhadap *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*.

Minggu ke-	Pekerjaan yang Dilakukan
13	• Melakukan riset dan pengembangan terhadap *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*.
14	• Melakukan riset dan pengembangan terhadap *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*. • Menyusun dokumentasi teknis untuk fitur CRUD pada proyek *handoff*.
15	• Melakukan penyempurnaan pengembangan terhadap *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*. • Menyusun dokumentasi teknis untuk fitur CRUD pada proyek *handoff*.

Minggu ke-	Pekerjaan yang Dilakukan
16	• Melakukan penyempurnaan pengembangan *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*. • Menyusun dokumentasi teknis untuk fitur CRUD pada proyek *handoff*.
17	• Melakukan penyempurnaan pengembangan *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*. • Menyusun dokumentasi teknis untuk fitur CRUD pada proyek *handoff*.
18	• Melakukan penyempurnaan pengembangan *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Menyusun dokumentasi teknis untuk fitur otentikasi pada proyek *handoff*.

Minggu ke-	Pekerjaan yang Dilakukan
19	• Melakukan penyempurnaan pengembangan *library* editor dokumen yang relevan untuk mendukung kebutuhan fungsional dari fitur aplikasi *Regulatory Information Management* yang akan dikembangkan. • Berpartisipasi dalam pertemuan dengan pengguna untuk membahas dan mengidentifikasi kebutuhan aplikasi dalam proyek *Regulatory Information Management*. • Menyusun dokumentasi teknis untuk fitur otentikasi pada proyek *handoff*.

3.3.1 *User Requirement*

User requirement dalam proyek ini disusun berdasarkan dokumen *User Requirement Specification* (URS) yang dikembangkan melalui proses kolaboratif antar berbagai pihak terkait. Penyusunan URS diawali dengan studi kelayakan (*feasibility study*) untuk memastikan bahwa aplikasi yang dirancang sesuai dengan kebutuhan. URS berfungsi sebagai landasan untuk mengidentifikasi fitur utama yang dibutuhkan, alur kerja sistem, serta batasan dan pengembangan [10]. Berikut merupakan fitur utama yang dibutuhkan dalam pembuatan editor dokumen.

A Pengambilan dan Mengolah Dokumen

Salah satu kebutuhan utama dalam pengembangan editor dokumen adalah kemampuan untuk mengambil dan mengolah dokumen yang tersimpan di media penyimpanan dokumen. Fitur ini memungkinkan pengguna untuk membuka dan mengolah dokumen secara langsung dari repositori penyimpanan tanpa perlu mengunduh file secara manual. Dengan adanya integrasi ini, alur kerja pengguna menjadi lebih efisien dan terpusat.

B Pembaruan dan Penyimpanan Dokumen ke Layanan Cloud

Fitur ini memungkinkan pembaruan dokumen yang tersimpan di layanan *cloud* secara otomatis ketika pengguna melakukan penyimpanan (*save*) melalui aplikasi.

Dengan integrasi ini, setiap perubahan yang dilakukan oleh pengguna akan langsung diperbarui pada dokumen yang sama di layanan *cloud* tanpa perlu tindakan manual tambahan. Mekanisme ini bertujuan untuk menjaga konsistensi data, meminimalkan risiko versi ganda, serta memastikan bahwa dokumen yang diambil dan diakses oleh pengguna lain selalu merupakan versi terbaru yang telah diedit melalui aplikasi.

C Memberikan Komentar pada Dokumen

Fitur komentar pada dokumen memungkinkan pengguna memberikan catatan atau masukan secara langsung pada bagian tertentu dari dokumen. Fitur ini mendukung komunikasi internal tim dalam proses penyusunan dan revisi dokumen, karena pengguna lain dapat membaca, menanggapi, atau menyelesaikan komentar yang diberikan. Hal ini menjadikan proses kolaborasi lebih terstruktur dan efisien.

D *Collaborative Editing*

Fitur *collaborative editing* memungkinkan lebih dari satu pengguna untuk mengedit dokumen secara bersamaan secara *real-time*. Perubahan yang dilakukan oleh masing-masing pengguna akan langsung terlihat oleh pengguna lain, sehingga dapat mendorong efisiensi kerja tim dan mempercepat proses penyusunan dokumen. Fitur ini juga dilengkapi dengan mekanisme penguncian atau penanda editan untuk menghindari konflik saat dokumen diedit secara bersama.

E Pencatatan dan Pemantauan *Version History* pada Dokumen

Fitur ini berfungsi untuk merekam setiap perubahan yang dilakukan terhadap dokumen dari waktu ke waktu. Dengan adanya riwayat versi, pengguna dapat melacak siapa yang melakukan perubahan, kapan perubahan dilakukan, serta melihat atau memulihkan versi sebelumnya apabila diperlukan. Hal ini sangat penting untuk menjamin auditabilitas dan keterlacakan dalam pengelolaan dokumen.

3.3.2 Riset dan Pengembangan

Untuk memenuhi *user requirement*, diperlukan proses riset dan pengembangan teknologi guna memastikan bahwa fitur-fitur yang dirancang dapat berjalan sesuai

dengan kebutuhan pengguna. Fokus utama riset ini adalah mengidentifikasi *library* editor dokumen yang mendukung fungsionalitas lanjutan, seperti pemberian komentar, pencatatan *version history*, dan *collaborative editing* secara *real-time*. Selain itu, dilakukan pula evaluasi terhadap kemungkinan integrasi antara aplikasi yang dikembangkan dengan platform penyimpanan dokumen seperti *cloud*.

Beberapa *library* editor dokumen berbasis web yang dikaji antara lain *library* T, *library* C, *library* TM, *library* Q, *library* C, dan O. Evaluasi dilakukan berdasarkan kriteria fungsionalitas, kelengkapan dokumentasi, dukungan komunitas, kemudahan integrasi, serta fleksibilitas pengembangan (*open-source*). Berdasarkan hasil kajian, *library* O dinilai paling memenuhi kriteria tersebut karena menyediakan fitur lengkap yang mendukung kebutuhan pengguna, seperti *real-time collaborative editing*, pencatatan *version history* dokumen, serta kemampuan integrasi dengan platform penyimpanan dokumen berbasis *cloud* melalui API yang kompatibel. Selain itu, *library* O memiliki dokumentasi yang bersifat *open-source*, sehingga memungkinkan kustomisasi lebih lanjut sesuai kebutuhan internal. Dengan demikian, *library* O dipertimbangkan sebagai kandidat utama dalam tahap implementasi editor dokumen untuk mendukung pengembangan aplikasi RIM.

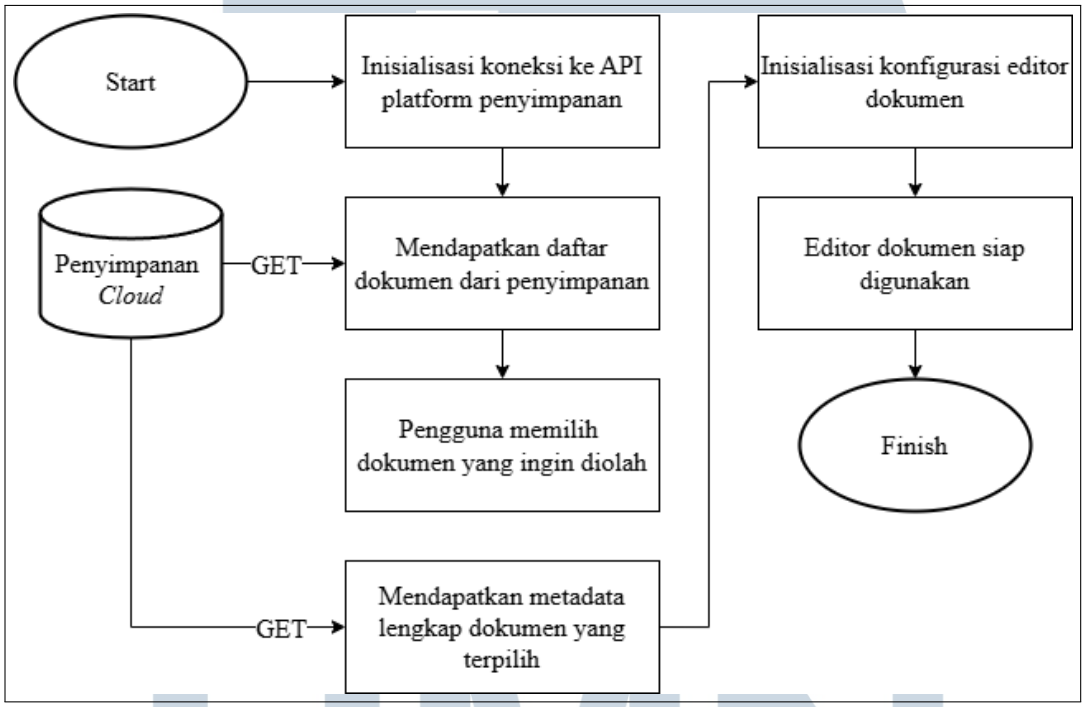
Riset juga mencakup berbagai jenis platform penyimpanan dokumen berbasis *cloud* untuk mengevaluasi kapabilitas integrasi dengan sistem pengelolaan dokumen yang dirancang. Salah satu fokus utama riset adalah menelusuri mekanisme API yang disediakan oleh masing-masing platform, khususnya dalam hal pembaruan otomatis dokumen ketika pengguna melakukan penyimpanan (*save*) melalui aplikasi. Untuk tahap awal, riset dilakukan pada salah satu platform yang menyediakan dukungan API yang cukup matang sebagai dasar implementasi. Hasil dari studi ini menjadi acuan dalam perancangan arsitektur sistem, pemilihan *library*, serta perumusan alur kerja fitur editor dokumen agar sesuai dengan kebutuhan pengelolaan informasi regulasi secara digital dan terintegrasi.

3.3.3 Hasil Riset dan Pengembangan

Hasil riset dan pengembangan dalam proyek ini, merujuk untuk menjawab kebutuhan pengguna sesuai dengan *User Requirement Specification*. Setiap fitur yang dikaji bertujuan untuk mendukung proses pengelolaan dokumen regulasi secara efisien, terintegrasi, dan sesuai standar yang berlaku.

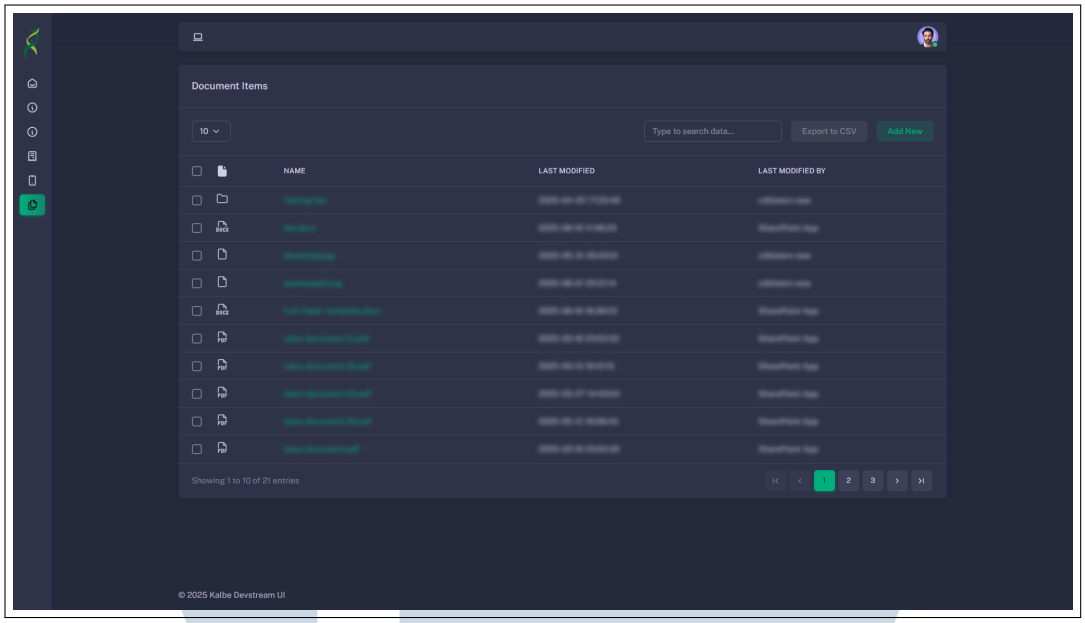
A Pengambilan dan Mengolah Dokumen

Berdasarkan hasil riset, proses pengambilan dan pengolahan dokumen dari platform penyimpanan dokumen berbasis *cloud* ke dalam aplikasi dapat diimplementasikan melalui integrasi dengan API yang disediakan oleh platform tersebut. Berikut merupakan gambar diagram alur untuk proses pengambilan dokumen dari penyimpanan *cloud* ke dalam aplikasi RIM.



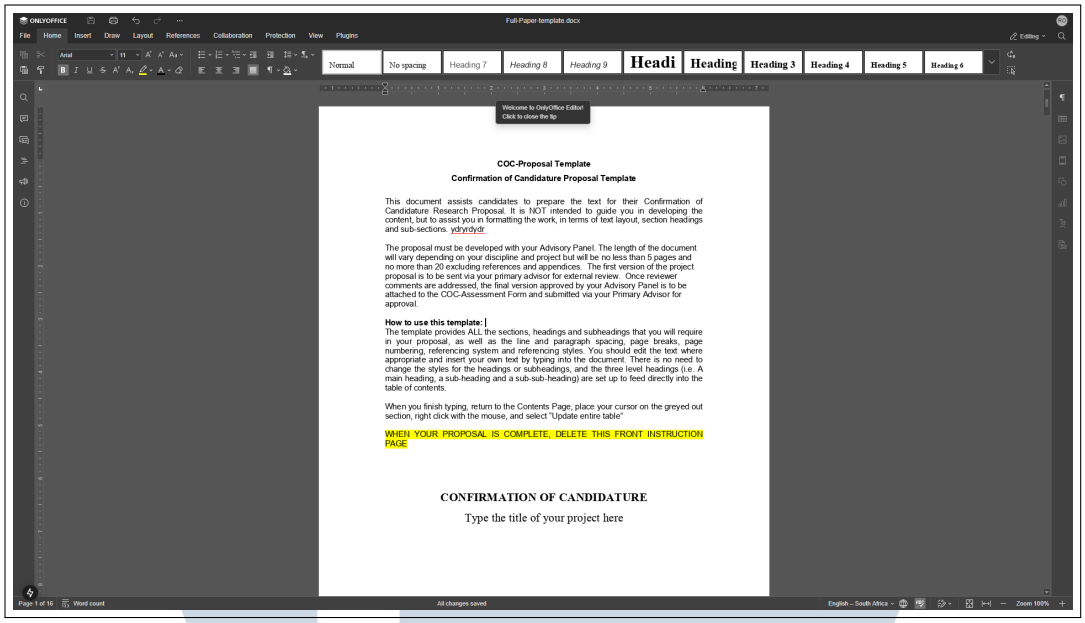
Gambar 3.1. Diagram Alur Pengambilan Dokumen

Gambar 3.1 merepresentasikan alur proses pengambilan dokumen yang dimulai dengan tahap inisialisasi koneksi ke layanan API dari platform penyimpanan dokumen berbasis *cloud*. Koneksi ini bertujuan untuk memperoleh daftar dokumen yang tersimpan pada repositori digital. Agar proses autentikasi dan otorisasi berhasil, sistem memerlukan kredensial khusus yang memberikan izin akses terhadap penyimpanan dokumen yang relevan. Setelah daftar dokumen tersedia dan pengguna memilih salah satu dokumen, sistem akan mengambil metadata lengkap dari dokumen terpilih melalui layanan API. Metadata ini kemudian digunakan untuk mengonfigurasi editor dokumen, sehingga dokumen dapat dimuat dan ditampilkan kepada pengguna untuk keperluan pengolahan lebih lanjut.



Gambar 3.2. Tampilan Antarmuka Daftar Dokumen

Gambar 3.2 menampilkan antarmuka pengguna yang menjadi visualisasi dari hasil proses pengambilan data yang telah diuraikan sebelumnya. Antarmuka ini menyajikan daftar dokumen yang berhasil diperoleh dari repositori digital dalam format tabel yang terstruktur. Setiap baris pada tabel merepresentasikan satu dokumen dan menampilkan berbagai metadata, seperti nama dokumen, stempel waktu modifikasi terakhir, serta informasi pengguna atau aplikasi yang terakhir mengubahnya.



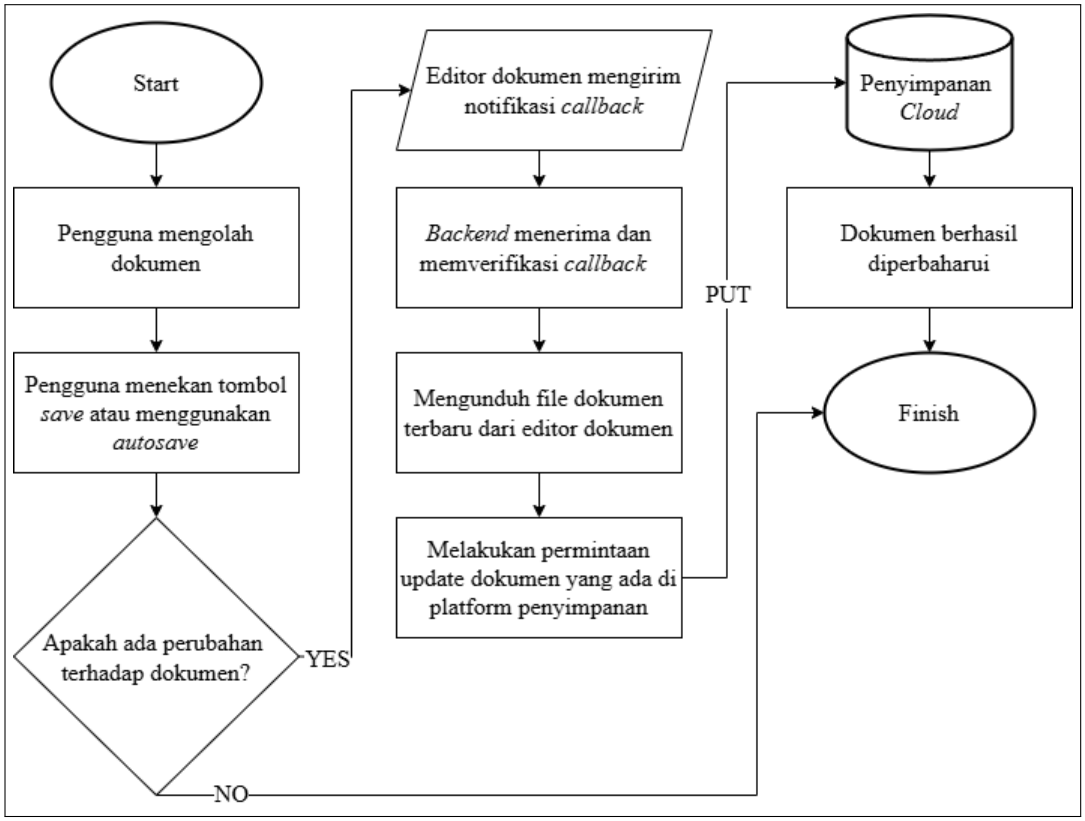
Gambar 3.3. Tampilan Antarmuka Editor Dokumen

Gambar 3.3 menampilkan antarmuka editor dokumen yang berhasil memuat dokumen yang dipilih oleh pengguna pada tahap sebelumnya. Tampilan ini merupakan hasil akhir dari proses pengambilan dokumen, di mana metadata yang diperoleh digunakan untuk mengonfigurasi dan menginisialisasi komponen editor secara akurat. Antarmuka ini menyediakan lingkungan pengolahan dokumen yang fungsional dan interaktif, dilengkapi dengan berbagai perangkat (*toolbar*) untuk pemformatan teks, pengaturan tata letak, serta fitur penyuntingan lainnya. Dengan demikian, pengguna dapat langsung memulai proses pengolahan dokumen, baik untuk meninjau maupun memodifikasi konten, secara langsung di dalam aplikasi.

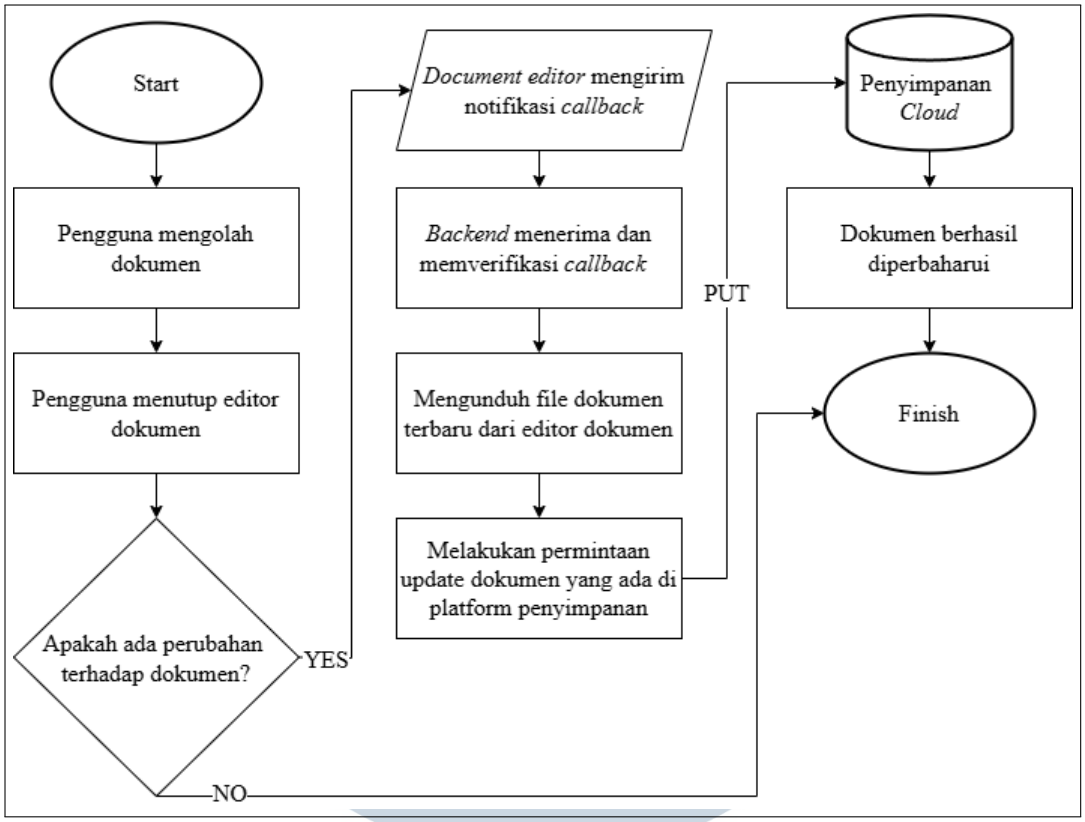
B Pembaruan dan Penyimpanan Dokumen

Proses penyimpanan dokumen pada editor dokumen dapat dilakukan melalui dua mekanisme utama. Pertama, pengguna dapat menyimpan perubahan secara manual melalui tombol *save* atau mengandalkan fitur *autosave* yang berjalan secara berkala. Kedua, dokumen secara otomatis disimpan ketika pengguna keluar dari sesi pengeditan. Setelah proses penyimpanan dilakukan oleh editor dokumen, sistem akan memperbarui dokumen yang tersimpan di platform penyimpanan dokumen berbasis *cloud*. Kedua metode ini memastikan bahwa seluruh perubahan yang dilakukan oleh pengguna dapat tersimpan dan diperbarui secara konsisten pada sistem penyimpanan. Berikut merupakan gambar diagram alur dari kedua metode

penyimpanan.



Gambar 3.4. Diagram Alur Proses Pembaruan Dokumen Melalui Tombol *Save* atau *Autosave*

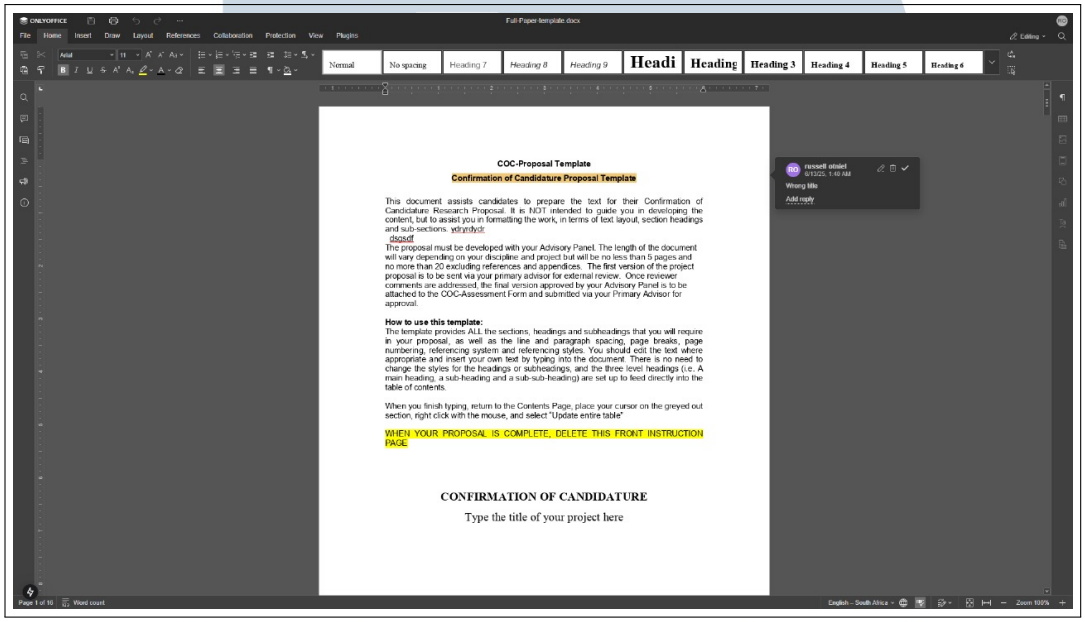


Gambar 3.5. Diagram Alur Proses Pembaruan Dokumen Ketika Pengguna Menutup Editor Dokumen

Gambar 3.4 dan 3.5 menggambarkan alur proses pembaruan dokumen dari editor dokumen ke platform penyimpanan dokumen berbasis *cloud* melalui dua metode penyimpanan yang tersedia. Proses ini dimulai ketika pengguna melakukan perubahan terhadap isi dokumen, kemudian memicu proses penyimpanan, baik secara manual melalui tombol *save*, secara otomatis melalui fitur *autosave*, maupun saat pengguna keluar dari sesi pengeditan. Selanjutnya, sistem akan melakukan pemeriksaan untuk memastikan bahwa perubahan benar-benar terjadi pada konten dokumen. Jika tidak ditemukan perubahan, proses akan dihentikan untuk menghindari pembaruan yang tidak diperlukan. Namun, apabila perubahan terdeteksi, editor dokumen akan mengirimkan notifikasi berupa *callback* ke layanan *backend*. *Backend* kemudian memverifikasi notifikasi tersebut dan mengambil URL dokumen yang telah diperbarui. Setelah itu, sistem akan melakukan permintaan pembaruan ke platform penyimpanan dengan mengunggah versi terbaru dokumen untuk menggantikan versi sebelumnya. Seluruh proses ini memastikan bahwa setiap modifikasi yang dilakukan oleh pengguna tersimpan secara konsisten dan sinkron dengan sistem penyimpanan dokumen.

C Memberikan Komentar pada Dokumen

Fitur komentar pada dokumen dapat diaktifkan langsung melalui konfigurasi di dalam komponen editor dokumen tanpa memerlukan integrasi tambahan dengan layanan eksternal. Seluruh fungsi terkait komentar, seperti menambah, membalas, dan menyelesaikan komentar, telah disediakan sebagai bagian dari fitur bawaan editor, sehingga implementasinya cukup dilakukan di sisi *frontend*.



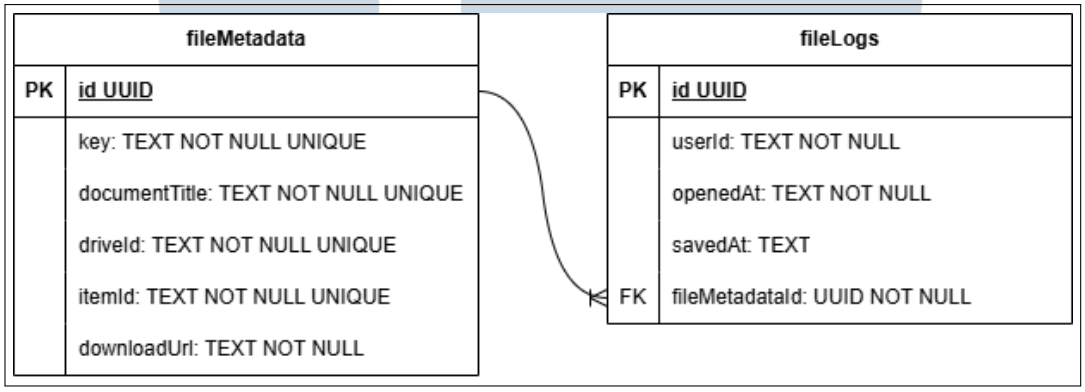
Gambar 3.6. Tampilan Antarmuka dari Implementasi Komentar pada Dokumen

Gambar 3.6 mendemonstrasikan implementasi fitur komentar dalam antarmuka editor. Ketika pengguna menandai suatu bagian teks, sebuah kotak dialog komentar akan muncul secara kontekstual di sisi kanan dokumen. Kotak dialog ini menampilkan identitas pengguna yang memberikan komentar, stempel waktu, serta kolom input untuk menuliskan tanggapan. Selain itu, terdapat pula fungsionalitas untuk membalas dan menyelesaikan komentar dengan menekan tombol centang. Teks yang telah diberi komentar akan ditandai dengan sorotan berwarna untuk memberikan indikator visual yang jelas kepada pengguna lain bahwa pada bagian tersebut terdapat diskusi atau catatan yang perlu diperhatikan.

D *Collaborative Editing*

Berdasarkan hasil riset, fitur *collaborative editing* pada editor dokumen dapat diimplementasikan dengan memanfaatkan mekanisme sinkronisasi yang

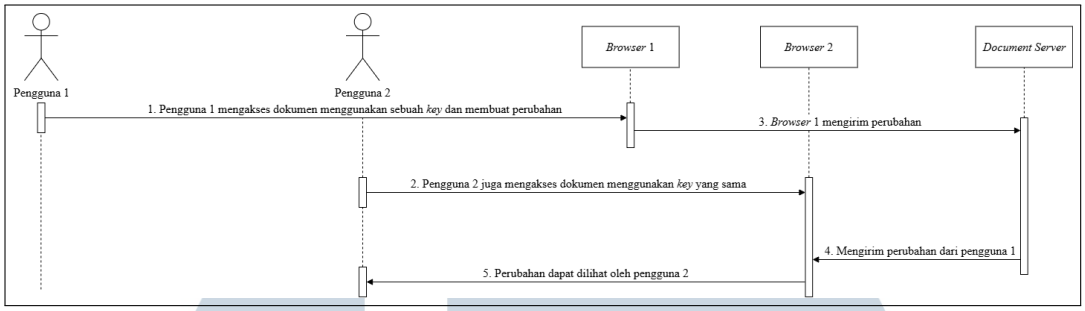
telah disediakan oleh komponen editor dokumen. Mekanisme sinkronisasi memungkinkan beberapa pengguna untuk melakukan pengeditan secara bersamaan pada dokumen yang sama. Setiap perubahan yang diberikan oleh pengguna, akan dikirimkan secara langsung ke *document server* dan didistribusikan kembali ke pengguna lain secara *real-time*. Untuk mendukung proses *collaborative editing*, setiap dokumen harus memiliki *key* yang unik sebagai identifikasi khusus. *Key* ini berfungsi untuk membedakan antara satu dokumen dengan yang lain dan perlu disimpan di dalam *database* bersama dengan metadata terkait. Berikut merupakan *entity relationship diagram* (ERD) yang digunakan untuk menyimpan informasi *key* dan metadata dokumen.



Gambar 3.7. *Entity Relationship Diagram* (ERD) untuk Menyimpan Informasi *Key* dan Metadata Dokumen

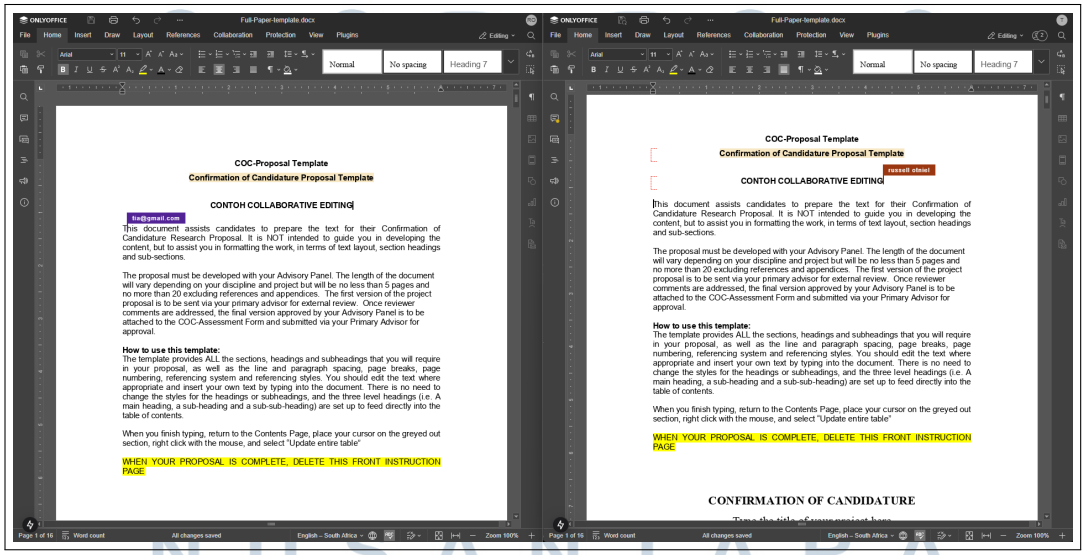
Gambar 3.7 menunjukkan ERD yang terdiri dari dua tabel yang saling berelasi, yaitu fileMetadata dan fileLogs. Tabel fileMetadata berfungsi sebagai penyimpanan informasi utama dokumen, termasuk *key* yang digunakan untuk mendukung fitur *collaborative editing*. Kolom seperti documentTitle, driveId, itemId, dan downloadUrl merepresentasikan identitas serta lokasi dokumen yang tersimpan di platform penyimpanan dokumen berbasis *cloud*. Tabel ini memiliki relasi *one-to-many* dengan tabel fileLogs, yang berperan dalam mencatat riwayat aktivitas pengolahan dokumen secara terperinci.

Untuk mengelola proses *collaborative editing*, struktur penyimpanan data yang terdapat pada gambar 3.7 menjadi landasan penting dalam pencatatan dan identifikasi dokumen. Setelah informasi dokumen dan *key* tersimpan, sistem dapat menjalankan mekanisme sinkronisasi yang memungkinkan kolaborasi secara *real-time* antar pengguna. Berikut merupakan diagram urutan dari mekanisme sinkronisasi selama sesi *collaborative editing* berlangsung.



Gambar 3.8. Diagram Urutan Proses Sinkronisasi *collaborative editing*

Gambar 3.8 mengilustrasikan alur proses sinkronisasi *collaborative editing* secara *real-time*. Proses ini dimulai ketika Pengguna Satu mengakses dan melakukan perubahan pada sebuah dokumen melalui Browser Satu dengan menggunakan *key* unik yang mengidentifikasi dokumen tersebut. Secara bersamaan, Pengguna Dua juga bergabung dalam sesi pengeditan dengan mengakses dokumen yang sama melalui Browser Dua, menggunakan *key* yang identik. Setiap perubahan yang dilakukan oleh Pengguna Satu akan ditangkap oleh Browser Satu dan dikirimkan ke *document server* untuk diproses. *Document server* kemudian berperan sebagai koordinator yang mendistribusikan data perubahan tersebut kepada seluruh pengguna lain yang terhubung dalam sesi, sehingga setiap pengguna menerima pembaruan secara *real-time*.



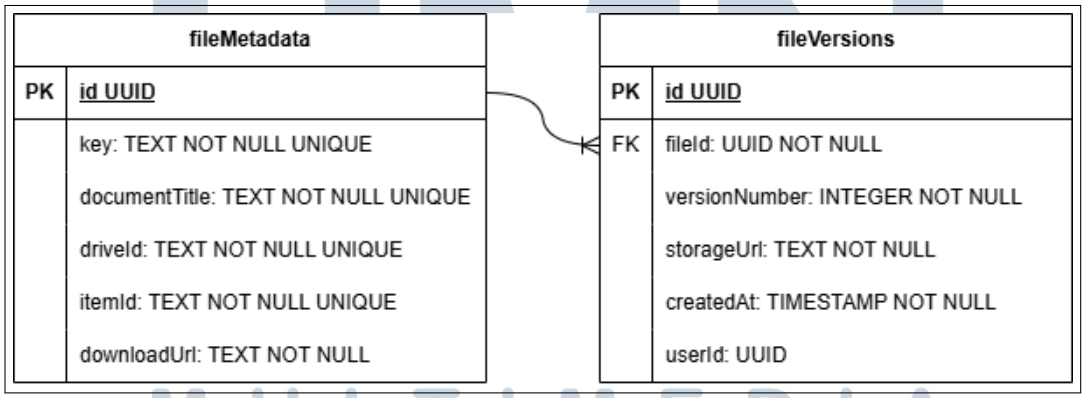
Gambar 3.9. Tampilan Antarmuka Ketika Melakukan *Collaborative Editing*

Gambar 3.9 memperlihatkan tampilan antarmuka pengguna selama sesi *collaborative editing* berlangsung. Visualisasi ini menampilkan dua *window*

yang merepresentasikan dua pengguna berbeda yang sedang mengakses dan mengedit dokumen yang sama secara bersamaan. Sistem menyediakan indikator visual untuk menunjukkan keberadaan dan aktivitas masing-masing pengguna, seperti penanda kursor serta teks yang sedang disorot, yang ditampilkan dalam warna berbeda dan dilengkapi dengan label identitas pengguna. Ketika salah satu pengguna melakukan perubahan seperti memasukkan teks, modifikasi tersebut akan secara otomatis disinkronkan dan ditampilkan secara *real-time* pada jendela pengguna lainnya. Mekanisme umpan balik visual ini berperan penting dalam memastikan pengalaman kolaborasi yang konsisten dan interaktif.

E Pencatatan dan Pemantauan *Version History* pada Dokumen

Fitur pencatatan dan pemantauan *version history* dapat diimplementasikan dengan merekam setiap perubahan yang terjadi pada dokumen secara otomatis, termasuk informasi seperti waktu modifikasi, identitas pengguna yang melakukan perubahan, serta URL penyimpanan dokumen di layanan cloud. Mekanisme ini memungkinkan sistem menyimpan versi-versi terdahulu dari dokumen sebagai bagian dari proses pelacakan dan audit. Untuk mendukung fitur ini, diperlukan penyimpanan informasi terkait di dalam *database*. Berikut merupakan *entity relationship diagram* (ERD) yang digunakan untuk merepresentasikan struktur data dalam pencatatan *version history*.

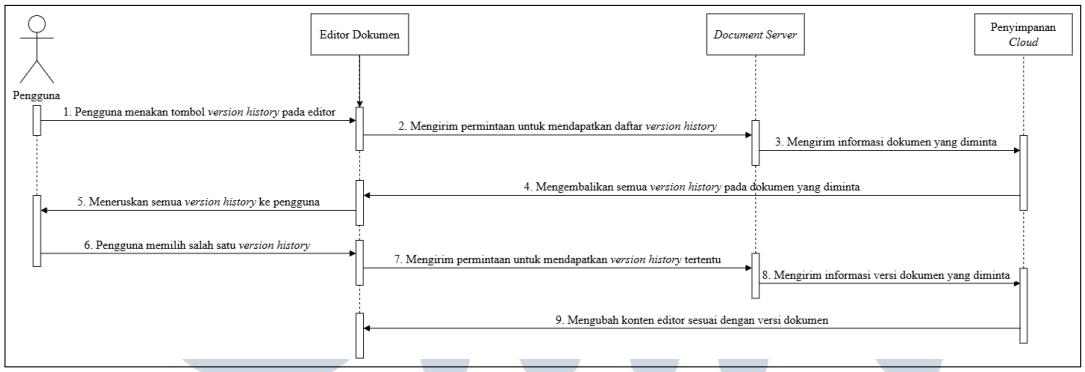


Gambar 3.10. *Entity Relationship Diagram* (ERD) untuk Menyimpan Informasi *Version History* pada Dokumen

Gambar 3.10 menunjukkan ERD yang dirancang untuk mendukung fitur *version history*, terdiri dari dua tabel yang saling berelasi, yaitu fileMetadata dan fileVersions. Tabel fileMetadata berfungsi sebagai induk yang menyimpan

informasi utama dan unik dari setiap dokumen, seperti key dan documentTitle. Tabel ini memiliki relasi *one-to-many* dengan tabel fileVersions, yang secara spesifik mencatat setiap versi historis dari sebuah dokumen. Melalui *foreign key* fileId, setiap data dalam fileVersions dapat dihubungkan kembali ke dokumen induknya. Kolom versionNumber digunakan untuk mengidentifikasi urutan versi, storageUrl menyimpan lokasi penyimpanan unik untuk setiap versi dokumen, sementara createdAt dan userId mencatat stempel waktu pembuatan versi dan identitas pengguna yang melakukan perubahan. Struktur ini memungkinkan sistem untuk melacak dan mengelola seluruh riwayat modifikasi dokumen secara terperinci dan terstruktur.

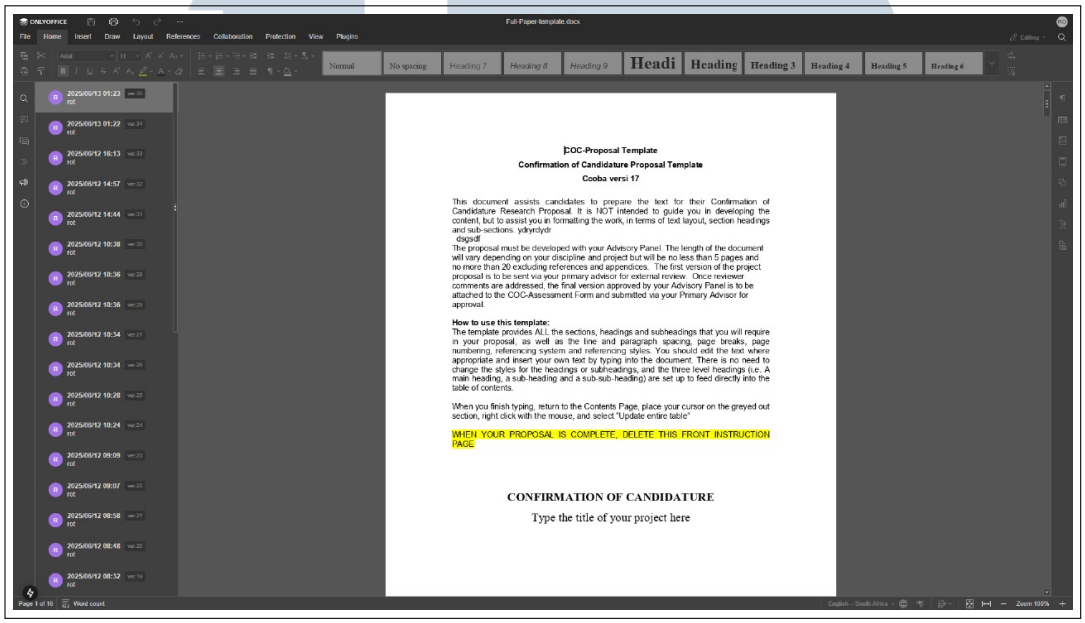
Setelah struktur *database* dirancang untuk mendukung pencatatan *version history*, diperlukan pemahaman lebih lanjut mengenai alur komunikasi antar komponen sistem saat pengguna melakukan permintaan terhadap riwayat versi suatu dokumen. Proses ini mencakup interaksi antara antarmuka pengguna, layanan backend, dan penyimpanan dokumen di layanan penyimpanan berbasis *cloud*. Untuk menggambarkan bagaimana permintaan *version history* diproses, berikut merupakan diagram urutan yang memvisualisasikan tahapan-tahapan dalam alur komunikasi tersebut.



Gambar 3.11. Diagram Urutan Proses Permintaan *Version History* pada Dokumen

Gambar 3.11 mengilustrasikan alur proses permintaan *version history* pada dokumen. Proses ini diawali ketika pengguna menekan tombol *version history* pada antarmuka editor dokumen. Sebagai respons, editor akan mengirimkan permintaan kepada *document server* untuk memperoleh daftar versi dokumen yang tersedia. *Document server* kemudian melakukan koordinasi dengan layanan penyimpanan *cloud* guna mengambil daftar *version history* yang relevan dan mengirimkannya kembali ke editor dokumen. Setelah daftar versi diterima dan ditampilkan, pengguna dapat memilih salah satu versi yang diinginkan. Pilihan ini akan memicu

pengiriman permintaan lanjutan dari editor dokumen ke *document server* untuk memperoleh detail versi yang dipilih. Selanjutnya, *document server* mengambil informasi versi tersebut dari layanan penyimpanan *cloud* dan mengirimkannya ke editor. Pada tahap akhir, editor dokumen akan memuat dan menampilkan konten dari versi yang dipilih, sehingga memungkinkan pengguna untuk meninjau atau membandingkan perubahan yang telah terjadi sebelumnya.



Gambar 3.12. Tampilan Antarmuka Daftar *Version History*

Gambar 3.12 menampilkan antarmuka pengguna yang menjadi gambaran dari hasil permintaan *version history* pada dokumen. Antarmuka ini menyajikan daftar *version history* pada dokumen yang berhasil diperoleh dari *database*. Setiap baris pada panel sisi kiri merepresentasikan satu versi historis dokumen dan menampilkan berbagai metadata, seperti nomor versi, stempel waktu kapan versi tersebut dibuat, serta informasi pengguna yang menyimpannya. Ketika salah satu versi dari daftar tersebut dipilih oleh pengguna, konten utama editor di sisi kanan akan secara dinamis diperbarui untuk menampilkan isi dokumen sesuai dengan versi yang terpilih, sehingga memungkinkan pengguna untuk melakukan perbandingan antar-versi secara visual. Namun, fitur untuk menyoroti perbedaan perubahan antar-versi secara langsung maupun melakukan pemulihan (*restore*) ke versi sebelumnya belum dapat diimplementasikan pada tahap ini, dan masih memerlukan riset serta pengembangan lebih lanjut untuk memastikan fungsionalitas tersebut berjalan dengan andal dan efisien.

3.4 Kendala dan Solusi yang Ditemukan

Salah satu kendala utama yang dihadapi selama proses implementasi fitur editor dokumen adalah terbatasnya informasi yang tersedia pada dokumentasi resmi *library* editor dokumen yang digunakan. Dokumentasi tersebut umumnya lebih berfokus pada skenario penggunaan tertentu yang mengandalkan layanan pihak ketiga melalui pendekatan *hosting*, dan kurang memberikan panduan teknis untuk proses instalasi dan konfigurasi secara mandiri dalam lingkungan yang dikontainerisasi. Hal ini menimbulkan kesulitan dalam memahami alur integrasi yang diperlukan, terutama ketika penerapan dilakukan di luar lingkungan standar yang diberikan oleh pengembang *library* tersebut.

Solusi yang diterapkan untuk mengatasi kendala tersebut adalah dengan melakukan eksplorasi mandiri secara bertahap terhadap proses konfigurasi dan integrasi *library* editor dokumen di dalam lingkungan pengembangan yang digunakan. Proses ini melibatkan pengujian konfigurasi secara iteratif, penelusuran pesan kesalahan yang muncul, serta penyesuaian teknis berdasarkan hasil pengamatan langsung. Meskipun memerlukan waktu dan upaya tambahan, pendekatan ini memungkinkan pemahaman yang lebih dalam terhadap cara kerja *library* serta menghasilkan solusi yang sesuai dengan kebutuhan dan konteks sistem yang dibangun.