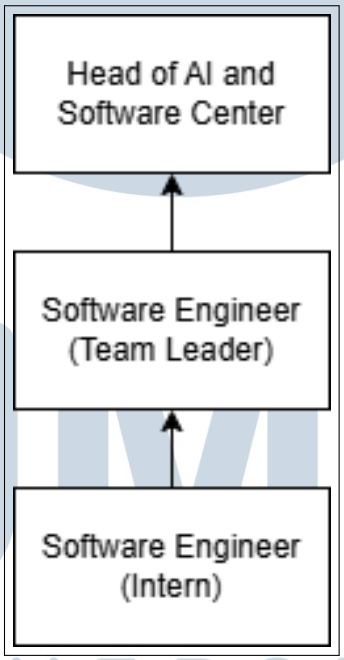
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama masa pelaksanaan program kerja praktik di PT Kalbe Farma Tbk., posisi ditempatkan dalam lingkup kerja tim Software Engineer dengan status Intern di bawah naungan departemen teknologi digital. Dalam struktur organisasi, tugas dan tanggung jawab dilaksanakan dengan pelaporan langsung kepada supervisor selaku pimpinan tim teknis (*team leader*), diilustrasikan pada Gambar 3.1. Penugasan kerja dilakukan melalui koordinasi mingguan yang terstruktur, baik dalam bentuk rapat daring yang dilaksanakan melalui Microsoft Teams atau Discord, maupun komunikasi tertulis melalui *group chat* Whatsapp.



Gambar 3.1. Struktur pelaporan dan pertanggungjawaban

Kegiatan koordinasi tidak terbatas pada lingkup tim internal, tetapi juga melibatkan unit kerja lintas departemen. Salah satu bentuk koordinasi tersebut tercermin melalui pertemuan mingguan yang difokuskan pada proses identifikasi kebutuhan sistem RIMS (*requirement gathering*), yang mempertemukan pihak pengelola proyek (project manager) dengan pengguna akhir dari departemen terkait. Kolaborasi ini menjadi komponen penting dalam memastikan kesesuaian antara kebutuhan bisnis dengan solusi teknologi yang dikembangkan.

Melalui sistem pelaporan, koordinasi, dan kolaborasi yang terstruktur tersebut, alur kerja dapat berjalan secara sistematis dan terintegrasi dengan proses bisnis yang berlaku di lingkungan perusahaan.

3.2 Tugas yang Dilakukan

Selama masa pelaksanaan kerja praktik di PT Kalbe Farma Tbk, kegiatan berfokus pada pembuatan berbagai *Proof of Concept* (PoC) untuk mengevaluasi kelayakan implementasi fitur-fitur teknologi yang direncanakan akan diintegrasikan ke dalam sistem informasi *real-time* perusahaan. Setiap tugas memiliki fokus eksploratif yang bertujuan untuk memastikan bahwa solusi yang diusulkan dapat diimplementasikan secara teknis dan sesuai dengan kebutuhan organisasi. Adapun rincian tugas yang dilakukan adalah sebagai berikut.

1. Pembuatan Editor LaTeX Interaktif dengan Kompilasi dan Visualisasi

   Tugas ini mencakup eksplorasi kemungkinan membangun sebuah editor berbasis web yang mampu menyunting sintaks LaTeX secara langsung (*live editing*), melakukan kompilasi terhadap dokumen tersebut menjadi format PDF, serta menampilkan hasil akhir pada antarmuka pengguna secara *real-time*. Implementasi dilakukan menggunakan pustaka-pustaka sumber terbuka (*open-source packages*) yang tersedia di internet. Pengujian dilakukan untuk memastikan kompatibilitas antarmuka, efisiensi kompilasi, serta kestabilan visualisasi hasil pada peramban modern.

2. Penggabungan Dokumen PDF secara Dinamis

   Dalam konteks pengelolaan dokumen, dilakukan eksplorasi terhadap integrasi fitur penggabungan beberapa berkas PDF menjadi satu dokumen utuh melalui antarmuka web. Penelitian ini mencakup kajian pustaka JavaScript yang memungkinkan proses *merging* PDF dilakukan di sisi server. Pendekatan ini diharapkan dapat meningkatkan efisiensi dalam pengelolaan laporan dan dokumen internal perusahaan, tanpa memerlukan proses *rendering* PDF yang rumit di sisi klien.

3. Integrasi Aplikasi Web dengan Penyimpanan *Cloud* Pihak Ketiga

   Tugas selanjutnya difokuskan pada integrasi sistem informasi perusahaan dengan platform penyimpanan *cloud* dari pihak ketiga. Proses dilakukan melalui pemanfaatan API, yang memungkinkan aplikasi web untuk melakukan operasi

create, *read*, *update*, dan *delete* (CRUD) terhadap dokumen dan folder yang tersimpan di *cloud*. Pengujian dimulai melalui simulasi di Postman, kemudian dilanjutkan dengan implementasi pada aplikasi web menggunakan *Software Development Kit* (SDK) resmi. Penekanan diberikan pada aspek autentikasi, pengelolaan izin akses, serta performa transfer data.

4. Implementasi Sistem *Drag and Drop* dalam Antarmuka Web

   Fitur *Drag and Drop* menjadi komponen penting dalam pembuatan sistem manajemen dokumen yang efisien dan intuitif. Pada tahap awal, implementasi dilakukan dalam konteks *kanban-style board* untuk mendukung representasi tugas atau dokumen dalam alur proses. Selanjutnya, fitur ini dikembangkan menjadi representasi berbasis *treeview* guna menggambarkan struktur hierarki dokumen dan folder. Salah satu tantangan yang diatasi dalam tugas ini adalah pembuatan fitur *cross-tab Drag and Drop*, yang memungkinkan pengguna untuk menyeret elemen dari satu tab peramban ke tab lainnya. Berbagai pustaka *Drag and Drop* diuji, dibandingkan, dan disesuaikan untuk memenuhi kebutuhan fungsional serta pengalaman pengguna yang optimal.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang pada PT Kalbe Farma Tbk diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	*Onboarding* perusahaan dan divisi, mempelajari dasar-dasar Typescript
2	Mempelajari *online course* yang diberikan oleh supervisor
3	Mempelajari *online course* yang diberikan oleh supervisor
4	Mempelajari penggunaan Supabase dan menghubungkan ke aplikasi web
5	Mini Project Part 1 .....
6	Mini Project Part 2 *styling*, mendalami penggunaan PostgreSQL
7	Riset dokumentasi shadcn dan Vuexy
8	Riset perbandingan shadcn dan Vuexy

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
9	Mengimplementasikan shadcn dan Vuexy ke dalam aplikasi
10	Mengimplementasikan shadcn dan Vuexy ke dalam aplikasi, riset penggunaan LaTeX di aplikasi
11	Riset *library package* yang mampu mengelola sintaksis dokumen LaTeX
12	Riset LJ, N-PL, N-L library dan Overleaf API untuk pembuatan file PDF
13	Riset API penyimpanan cloud pihak ketiga dan penggunaan Postman untuk testing API
14	Membuat LaTeX editor di aplikasi web, menambahkan fungsi kompilasi dan unduh file pada aplikasi
15	Membuat tutorial penggunaan Overleaf, menghubungkan aplikasi web dengan cloud API, *merging* PDF, *transfer knowledge* dengan senior
16	*Transfer knowledge* dengan senior, latihan *coding* menggunakan template proyek, *requirement gathering* dengan user
17	*Transfer knowledge* dengan senior, menambahkan fungsi *merging* PDF pada aplikasi, *requirement gathering* dengan user, riset *library package* untuk fungsi *Drag and Drop*, mengimplementasikan *Drag and Drop* dalam aplikasi
18	*Transfer knowledge* dengan senior, *debugging* penggunaan *Drag and Drop*
19	Riset *library package* untuk fungsi *Drag and Drop*, *debugging* & *refactor* kode *Drag and Drop*, *transfer knowledge* dengan senior
20	Mengimplementasikan fungsi *Delete Item* pada *Drag and Drop*
21	*Requirement gathering* dengan user, mengimplementasikan *memoization* pada *Drag and Drop*, riset *library package* yang dapat mengelola perpindahan *item* antar tab peramban

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
22	Mengimplementasikan dan *debugging Drag and Drop* di aplikasi, memperbarui logika implementasi *Drag and Drop*
23	Riset penggunaan dan dokumentasi *Drag and Drop* untuk *treeview*, mengimplementasikan *Drag and Drop* dalam bentuk *tree*
24	*Debugging Drag and Drop* dalam bentuk *Tree*, *requirement gathering* dengan user, memperbarui logika implementasi *treeview*
25	Riset kustomisasi *Drag and Drop*, pengelolaan *item* duplikat dan *copy paste*
26	Riset *Drag and Drop item pop-up preview*, *item copy-paste*
27	Penulisan dokumentasi proses CRUD pada aplikasi
28	Refaktor proses CRUD pada source code, riset *Business Process Model and Notation* (BPMN)

3.3.1 Fitur LaTeX *Live* Editor

Pembuatan fitur LaTeX *Live* Editor merupakan salah satu bagian dari kegiatan kerja praktik yang berorientasi pada pemenuhan kebutuhan pengguna dalam mengelola dan memvisualisasikan dokumen teknis yang berbasis sintaks LaTeX. Fitur ini dirancang untuk memungkinkan pengguna menulis, menyunting, dan mengompilasi dokumen LaTeX secara langsung melalui antarmuka web, dengan hasil akhir berupa tampilan dokumen dalam format PDF yang dapat ditinjau secara *real-time*.

A User Requirement

Berdasarkan diskusi dengan pihak pengguna dari departemen terkait, diperoleh kebutuhan utama yang melatarbelakangi pembuatan fitur ini, yaitu

1. Pengguna dapat menulis sintaks LaTeX dalam editor teks yang tersedia, atau mengunggah file .tex dari perangkat lokal.
2. Sistem dapat mengompilasi dokumen .tex menjadi format PDF.

3. Hasil kompilasi ditampilkan secara langsung pada panel pratinjau di samping editor, tanpa perlu mengunduh file terlebih dahulu.

4. Sistem bersifat *web-based* dan diakses melalui peramban standar perusahaan.

B Riset Pustaka / Dokumentasi

Untuk memenuhi kebutuhan tersebut, dilakukan studi terhadap berbagai pustaka JavaScript dan Node.js yang mendukung penyuntingan, kompilasi, dan pratinjau dokumen LaTeXĖvaluasi dilakukan berdasarkan kapabilitas, stabilitas, dokumentasi, serta kompatibilitas pustaka dengan *framework* yang digunakan (Next.js dan React). Ringkasan hasil evaluasi dapat dilihat pada Tabel 3.2.

Tabel 3.2. Hasil evaluasi riset pustaka LaTeX Live Editor

Pustaka	Kapabilitas	Kelebihan	Kekurangan
K	Merender ekspresi matematika LaTeX sebagai HTML	Ringan, cepat, kompatibel dengan browser	Tidak mendukung kompilasi dokumen penuh, hanya cocok untuk ekspresi matematika
MJ	Merender ekspresi matematika LaTeX	Dukungan luas terhadap sintaks matematis	Performa lebih lambat dibanding K, tidak mendukung output PDF
TLJ	Kompilasi penuh LaTeX ke PDF di sisi klien	Berbasis WASM, kompilasi offline di browser	Ukuran besar ($>$ 100MB), waktu muat lama, tidak stabil di semua peramban
LJ	Mengonversi LaTeX menjadi HTML	Cocok untuk pratinjau langsung	Tidak mendukung kompilasi ke PDF, dukungan pustaka LaTeX terbatas

Tabel 3.2. Hasil evaluasi riset pustaka LaTeX Live Editor (lanjutan)

Pustaka	Kapabilitas	Kelebihan	Kekurangan
R-L	Komponen React untuk merender LaTeX sebagai HTML	Integrasi mudah, cocok untuk tampilan ekspresi	Terbatas pada tampilan statis, tidak ada konversi PDF
R-L-N	Versi lanjutan R-L	Dukungan React modern, ringan	Fungsi terbatas pada ekspresi matematis sederhana
R-A	Konversi LaTeX sederhana dalam React	Integrasi mudah dengan proyek kecil	Tidak mendukung dokumen besar atau konversi PDF
N-L	Mendukung kompilasi LaTeX ke PDF, penambahan mandiri pustaka LaTeX	Dapat dikustomisasi, hasil PDF stabil	Membutuhkan dependensi lingkungan LaTeX di server
N-P	Eksekusi melalui Node.js untuk kompilasi PDF	Output standar LaTeX cocok untuk server	Bergantung pada eksekusi sistem, perlu konfigurasi dan sanitasi file
R-P	Menampilkan file PDF dalam antarmuka React	Mendukung tampilan PDF langsung	Tidak mengonversi LaTeX ke PDF; hanya untuk pratinjau file PDF yang sudah ada

Dari hasil kajian, pendekatan terbaik untuk memenuhi seluruh kebutuhan pengguna adalah menggunakan beberapa pustaka sebagai berikut.

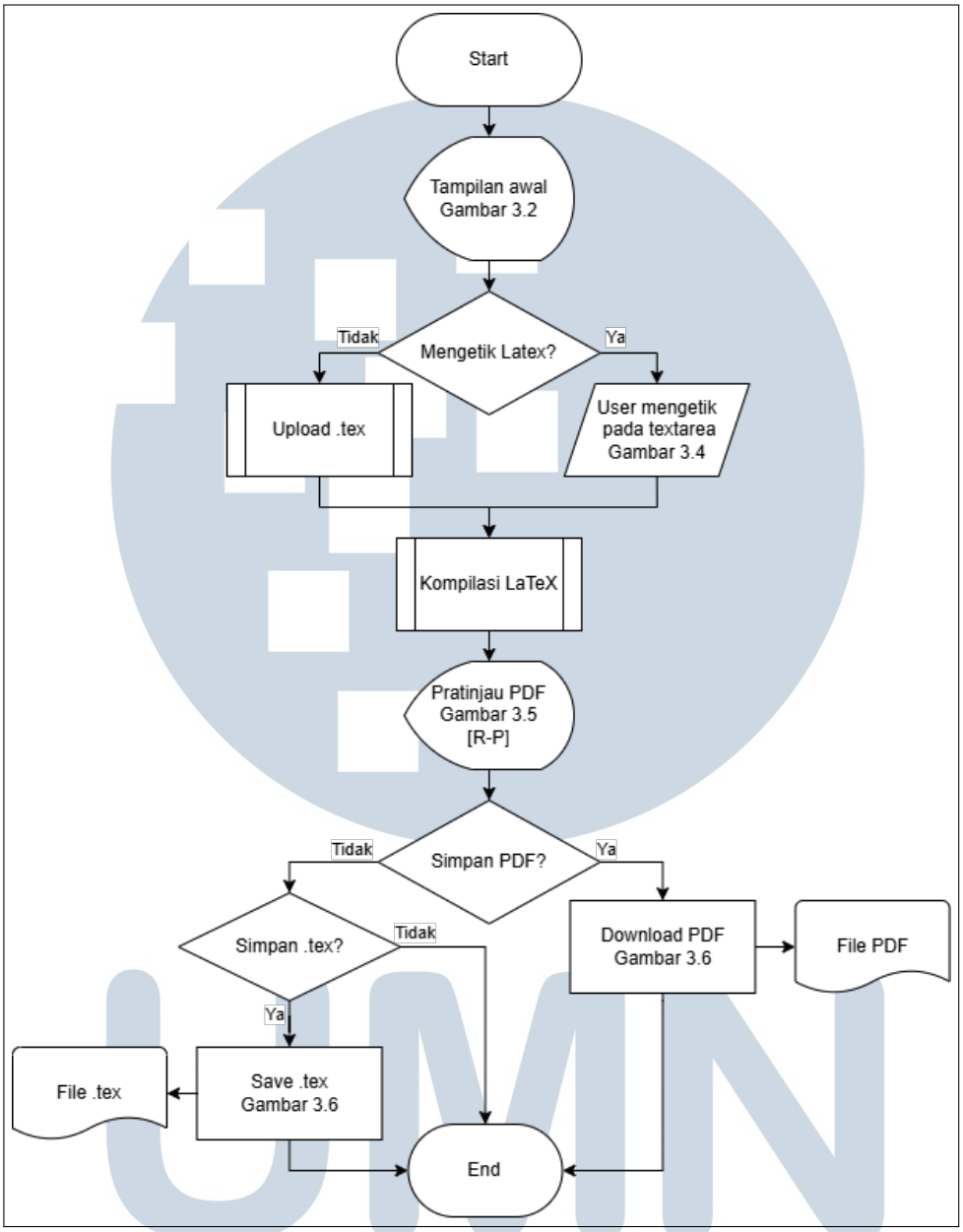
1. Pengeditan dan unggah file LaTeX dilakukan melalui editor teks React dan input file.
2. Kompilasi menggunakan N-L yang dijalankan pada sisi server, untuk menghasilkan file PDF dari input LaTeX. Dengan catatan, LaTeX harus terpasang pada server.
3. Pratinjau PDF menggunakan pustaka R-P untuk menampilkan hasil di halaman

yang sama, tanpa perlu mengunduh dokumen.

Pendekatan ini dinyatakan berhasil memenuhi seluruh kebutuhan yang telah dirumuskan sebelumnya dan dapat dijalankan dalam lingkungan Next.js dengan konfigurasi *server-side rendering* (SSR) yang sesuai.

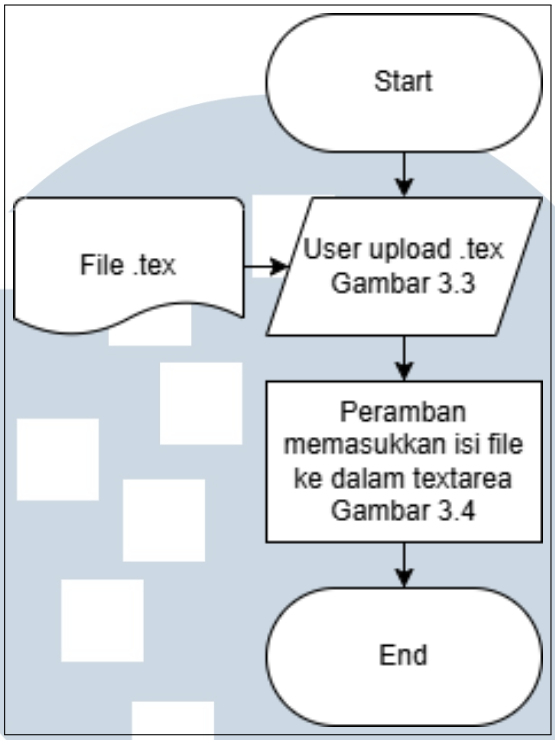
C Implementasi

Proses implementasi fitur LaTeX *Live* Editor dilakukan secara bertahap, dimulai dari integrasi komponen dasar untuk penulisan sintaks LaTeX, lalu dilanjutkan dengan sistem kompilasi, penyimpanan, dan visualisasi hasil kompilasi. Alur fungsional utama fitur ini dijabarkan dalam bentuk *flowchart* pada Gambar 3.2, yang menggambarkan keseluruhan *flowchart* dari interaksi pengguna hingga hasil ditampilkan dalam aplikasi web.

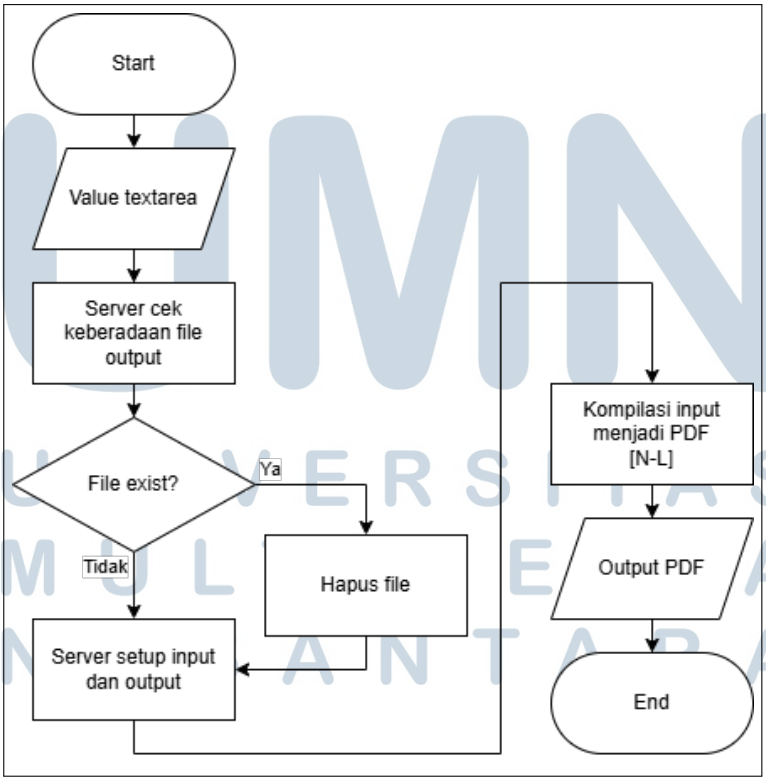


Gambar 3.2. Flowchart fitur LaTeX editor

Salah satu subproses penting dalam alur tersebut adalah proses unggah file .tex dari perangkat pengguna, yang merupakan alternatif dari penulisan langsung di editor. Alur logis dari proses unggah ini divisualisasikan dalam Gambar 3.3. Langkah berikutnya adalah proses kompilasi sintaks LaTeX yang dimasukkan pengguna, baik melalui pengetikan langsung maupun melalui file unggahan. Proses ini diilustrasikan dalam Gambar 3.4.

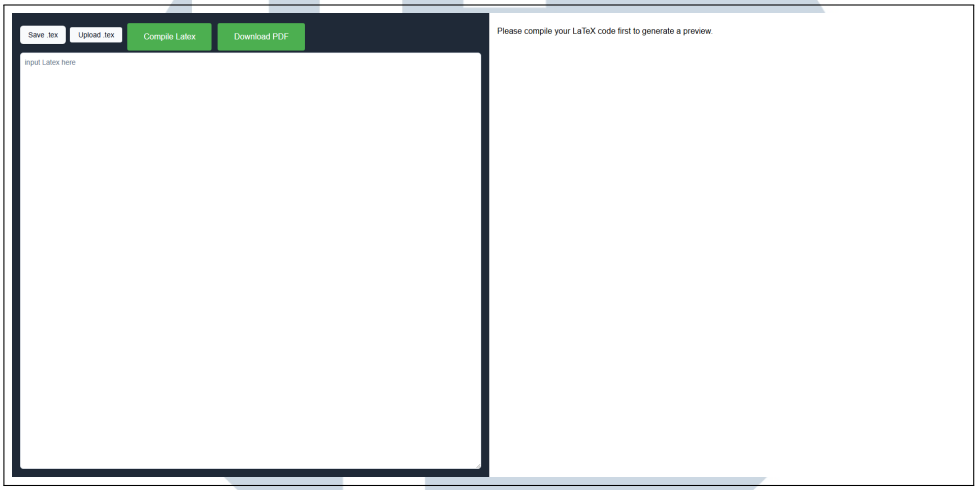


Gambar 3.3. Flowchart subproses Upload .tex



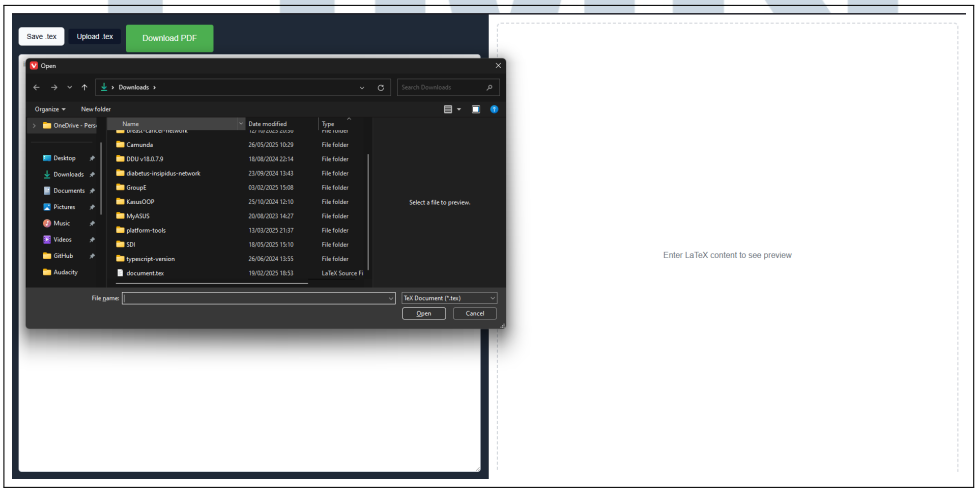
Gambar 3.4. Flowchart subproses Kompilasi LaTeX

Tampilan awal antarmuka LaTeX editor ditunjukkan dalam Gambar 3.5. Dalam tampilan ini, pengguna dapat melihat beberapa elemen utama fitur yang tersedia. Di sisi kiri antarmuka terdapat area editor sintaks LaTeX berupa kotak putih dengan *placeholder input LaTeX here*. Sementara di sisi kanan terdapat modul pratinjau yang akan menampilkan hasil kompilasi setelah proses dijalankan.



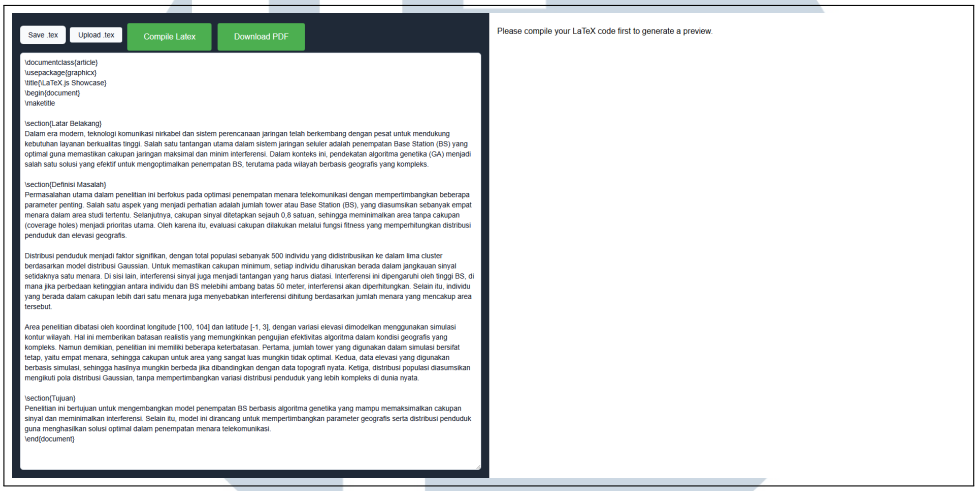
Gambar 3.5. Tampilan awal LaTeX editor

Pada bagian atas antarmuka editor terdapat deretan tombol utama. Salah satunya adalah tombol untuk mengunggah file .tex, yang ketika ditekan akan memunculkan modal unggah file sebagaimana diperlihatkan dalam Gambar 3.6. Proses ini berfungsi untuk menggantikan isi editor dengan konten dari file yang diunggah.



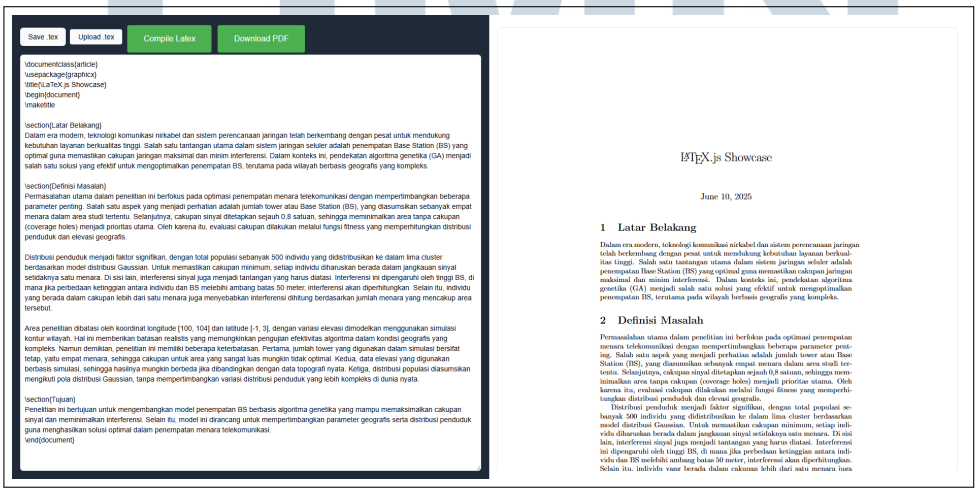
Gambar 3.6. Tampilan upload file .tex

Setelah file diunggah atau sintaks ditulis secara manual, pengguna akan melihat isi editor yang terisi. Keadaan ini ditunjukkan dalam Gambar 3.7, di mana konten LaTeX telah dimuat dan siap untuk dikompilasi. Keberadaan isi dalam editor memungkinkan pengguna melakukan modifikasi langsung sebelum proses kompilasi dilakukan.



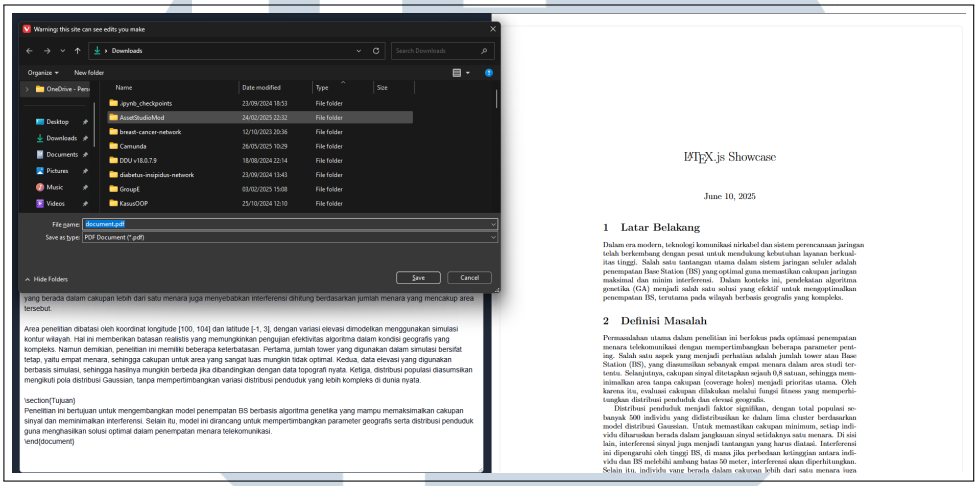
Gambar 3.7. Tampilan editor yang terisi

Setelah pengguna menekan tombol *Compile LaTeX*, sistem akan memproses isi editor di server dan mengembalikan hasil dalam bentuk PDF. Hasil tersebut kemudian ditampilkan pada modul pratinjau seperti terlihat dalam Gambar 3.8, yang menampilkan pemisah halaman dan hasil visual akhir dari dokumen LaTeX yang telah dikompilasi.



Gambar 3.8. Tampilan pratinjau LaTeX

Selain itu, antarmuka juga menyediakan dua tombol tambahan, yaitu tombol untuk menyimpan isi editor sebagai file .tex, dan tombol untuk mengunduh hasil kompilasi dalam format PDF. Ketika salah satu dari tombol tersebut ditekan, sistem akan menampilkan dialog lanjutan seperti ditampilkan dalam Gambar 3.9, yang memperlihatkan opsi penyimpanan lokal.



Gambar 3.9. Tampilan mengunduh PDF / simpan .tex

Dengan penyusunan proses dan tampilan antarmuka yang sistematis, fitur ini memberikan fleksibilitas penuh kepada pengguna untuk menyusun dokumen LaTeX secara efisien melalui antarmuka web yang intuitif. Seluruh fitur telah dirancang untuk mendukung kebutuhan aktual dalam proses penyusunan dokumen teknis berbasis LaTeX.

3.3.2 Fitur Penghubungan Aplikasi Web dengan Penyimpanan *Cloud* Pihak Ketiga

Integrasi sistem informasi berbasis web dengan platform manajemen dokumen eksternal menjadi salah satu kebutuhan utama dalam mendukung proses kerja yang terdigitalisasi dan terstandarisasi. Dalam konteks ini, penyimpanan *cloud* pihak ketiga dipilih sebagai platform penyimpanan dokumen, yang perlu dihubungkan dengan aplikasi web internal untuk mendukung akses data lintas sistem secara efisien dan aman.

A User Requirement

Berdasarkan kebutuhan yang disampaikan oleh pihak pengguna dari unit pengelola dokumen, diperlukan integrasi sistem yang memungkinkan aplikasi web internal untuk melakukan operasi dasar terhadap dokumen yang tersimpan di penyimpanan *cloud* pihak ketiga. Operasi tersebut mencakup:

1. Mengakses (*read*) dan menampilkan daftar file serta struktur folder yang tersimpan di *cloud*.
2. Mengunggah file baru (*create*) ke dalam direktori tertentu di *cloud*.
3. Memodifikasi atau mengganti file yang telah ada (*update*).
4. Menghapus file atau folder sesuai otorisasi yang berlaku (*delete*).

Fungsi-fungsi tersebut perlu dilakukan dengan tetap menjaga integritas data, kontrol akses yang ketat, dan kepatuhan terhadap kebijakan keamanan perusahaan.

B Riset Pustaka / Dokumentasi

Pembuatan dimulai dengan mempelajari terlebih dahulu mengenai penyimpanan *cloud* itu sendiri, apa kegunaan dan fungsi dari penyimpanan *cloud*, kemudian bagaimana penyimpanan *cloud* tersebut terintegrasi dalam ekosistem layanan *cloud-based*. Untuk memahami mekanisme dasar komunikasi antarsistem, dilakukan eksplorasi terhadap API *cloud* pihak ketiga, yaitu antarmuka terstandarisasi yang disediakan untuk mengakses data lintas layanan *cloud-based*, termasuk penyimpanan *cloud*.

Langkah awal dalam proses pembelajaran dilakukan melalui *cloud* API Explorer, yaitu antarmuka daring resmi untuk menguji permintaan (*request*) terhadap *cloud* API. Dalam platform ini, dipelajari cara membuat permintaan dasar seperti *GET*, *PUT*, *CREATE*, *DELETE* untuk beberapa operasi penyimpanan *cloud* seperti memperoleh struktur folder dan daftar file. Tahap ini memberikan pemahaman awal mengenai struktur URL, parameter permintaan, dan format tanggapan (*response*).

Setelah memahami permintaan dasar, eksplorasi dilanjutkan menggunakan aplikasi Postman untuk mencoba permintaan yang lebih kompleks yang akan digunakan saat tahap produksi, seperti pengambilan konten spesifik berdasarkan ID. Selain itu, Postman digunakan untuk menelusuri proses autentikasi OAuth 2.0,

yang diperlukan untuk mengakses API secara resmi dari aplikasi eksternal, serta pengecekan jika ada perubahan pada struktur *response*.

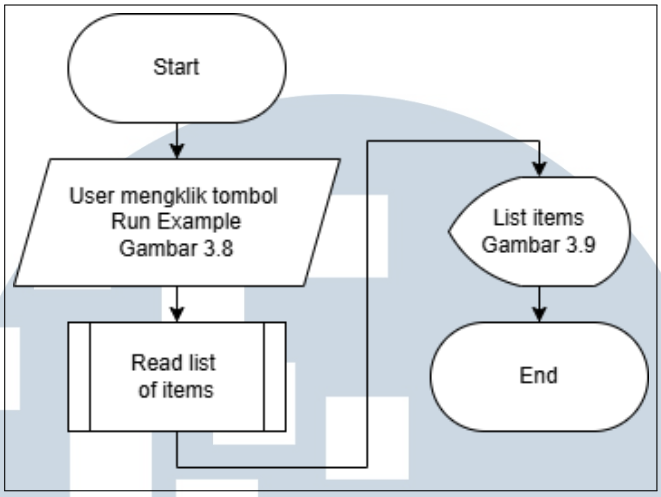
Pencarian selanjutnya diarahkan pada dokumentasi pengembangan aplikasi yang mengakses API *cloud*. Penyedia layanan *cloud* menyediakan berbagai *cloud* SDK sebagai pustaka untuk mendukung pembuatan aplikasi. Dari sekian SDK tersebut, maka terpilih *cloud* Javascript SDK yang memiliki kompatibilitas tertinggi dengan arsitektur proyek. Pustaka ini dirancang untuk menyederhanakan proses autentikasi dan komunikasi dengan *endpoint* API. Beberapa contoh proyek *single-page application* (SPA) juga disediakan secara resmi untuk tujuan pembelajaran. Kemudian, salah satu contoh proyek tersebut diimplementasikan sebagai langkah awal pembuatan fitur, meskipun implementasi *single-page application* tidak dapat memenuhi seluruh kebutuhan pengguna.

Setelah memahami lebih dalam struktur dan alur dari contoh proyek, pendekatan dialihkan ke dokumentasi reguler yang menjelaskan bagaimana melakukan autentikasi dengan server *cloud*, memperoleh *access token*, dan membuat *HTTP request* secara langsung. Pendekatan ini dianggap lebih fleksibel dan sesuai dengan arsitektur *server-side rendering* (SSR) yang digunakan dalam pembuatan sistem.

C Implementasi

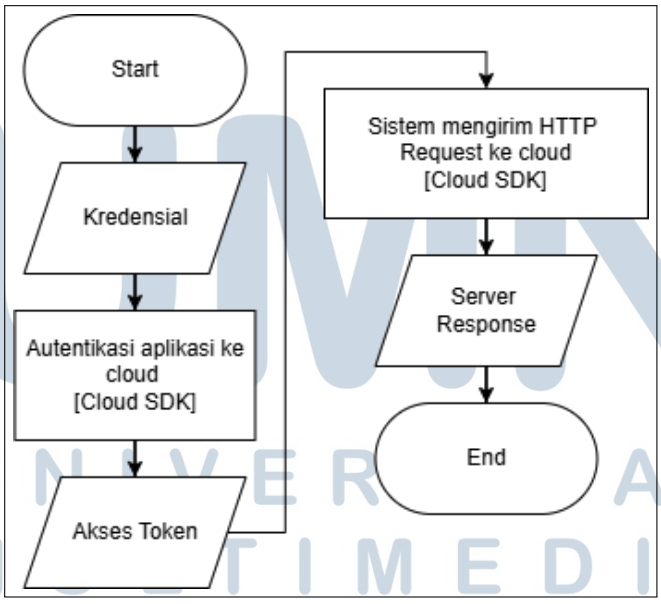
Implementasi dilakukan secara bertahap, dimulai dari proses konfigurasi autentikasi melalui portal layanan *cloud*. Langkah awal ini bertujuan untuk mendaftarkan aplikasi dan memperoleh kredensial resmi sebagai syarat untuk mengakses layanan *cloud* API. Setelah proses autentikasi berhasil dikonfigurasi dan diuji, tahap selanjutnya adalah integrasi modul API ke dalam aplikasi web yang telah dibangun.

Alur utama dari proses integrasi ini dijabarkan dalam Gambar 3.10, yang memperlihatkan bagaimana proses autentikasi dilakukan terlebih dahulu, lalu dilanjutkan dengan permintaan data menggunakan *access token* yang telah diperoleh.



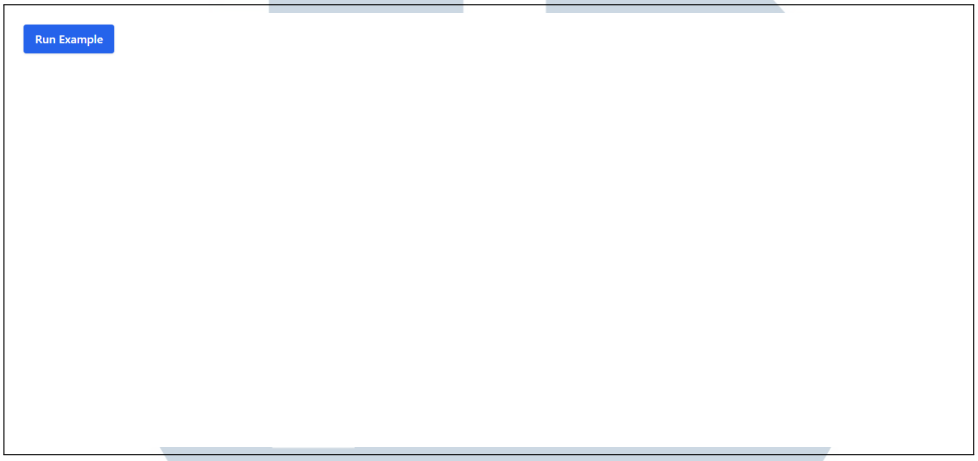
Gambar 3.10. Flowchart penghubungan aplikasi web dengan cloud

Setelah sistem berhasil terhubung, pengguna dapat menampilkan daftar folder dan dokumen yang tersimpan dalam *cloud*. Proses pengambilan daftar file ini dilakukan melalui *endpoint* API dan ditampilkan ke dalam antarmuka aplikasi. Tahapan teknis dari proses ini digambarkan dalam Gambar 3.11, yang menunjukkan subproses pengambilan data dari direktori *cloud*.



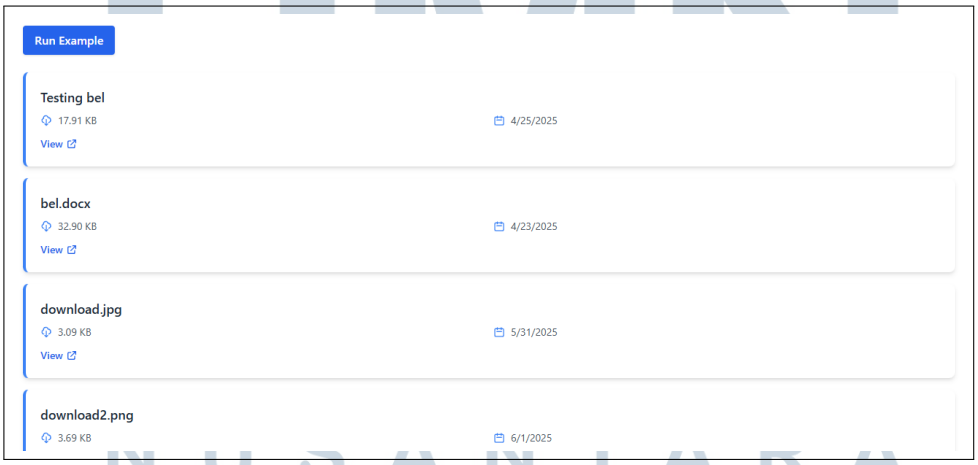
Gambar 3.11. Flowchart subproses Read List of Items

Setelah berhasil mengambil data, sistem akan menampilkan tampilan awal dari fitur integrasi pada antarmuka pengguna. Tampilan ini memuat tombol untuk memicu permintaan data dan area tampilan daftar dokumen. Ilustrasi dari tampilan awal fitur ini ditunjukkan dalam Gambar 3.12.



Gambar 3.12. Tampilan awal fitur penghubungan aplikasi web dengan cloud

Setelah pengguna menekan tombol untuk memuat data, sistem akan menampilkan hasil *response* dari layanan *cloud*, yakni berupa daftar file yang tersedia dalam direktori. Setiap file ditampilkan dengan informasi metadata dasar, seperti nama file, ukuran, dan waktu modifikasi terakhir. Tampilan hasil response ini dapat dilihat pada Gambar 3.13, yang menampilkan daftar file dalam bentuk tabel.



Gambar 3.13. Tampilan hasil response dari cloud

Selain menampilkan daftar file, sistem juga menyediakan kemampuan unggah dokumen ke direktori yang telah ditentukan, pembaruan dokumen

(*overwrite*), serta penghapusan file apabila pengguna memiliki hak akses. Seluruh fungsi tersebut dibangun dengan mempertimbangkan efisiensi, keamanan, serta pengalaman pengguna yang intuitif.

3.3.3 Fitur Penggabungan Dokumen PDF

Fitur penggabungan dokumen PDF (PDF *merging*) dibuat sebagai bagian dari solusi digital untuk mendukung efisiensi kerja dalam penyusunan dokumen administratif dan teknis yang seringkali terdiri dari banyak berkas terpisah. Kebutuhan untuk menggabungkan beberapa dokumen menjadi satu file utuh muncul dari permintaan pengguna internal yang ingin menyusun berkas lengkap secara otomatis tanpa harus menggunakan perangkat lunak pihak ketiga.

A User Requirement

Berdasarkan hasil diskusi kebutuhan sistem, pengguna memerlukan fitur yang memungkinkan penggabungan otomatis beberapa dokumen PDF yang telah disusun dalam satu unit kerja atau binder. Proses ini sebelumnya dilakukan secara manual, memakan waktu, dan rentan terhadap kesalahan urutan dokumen. Fitur yang dimaksud harus mampu melakukan hal berikut:

1. Menggabungkan seluruh dokumen yang terdapat dalam satu binder menjadi satu file PDF secara otomatis.
2. Menjaga urutan dokumen sesuai pengaturan yang telah dilakukan pengguna dalam sistem.
3. Menghasilkan file akhir yang siap diunduh atau digunakan dalam proses selanjutnya.

Fitur ini diharapkan dapat mengurangi ketergantungan terhadap perangkat lunak luar serta meningkatkan keterpaduan proses kerja dalam satu ekosistem aplikasi internal perusahaan.

B Riset Pustaka / Dokumentasi

Dalam tahap eksplorasi teknologi, dilakukan kajian terhadap pustaka JavaScript yang mendukung operasi penggabungan (*merge*) file PDF. Evaluasi dilakukan terhadap empat pustaka utama berdasarkan dokumentasi resmi,

komunitas pengguna, kemudahan integrasi, dan kompatibilitas dengan arsitektur serta eksekusi sisi server.

Tabel 3.3. Hasil evaluasi riset pustaka Penggabungan PDF

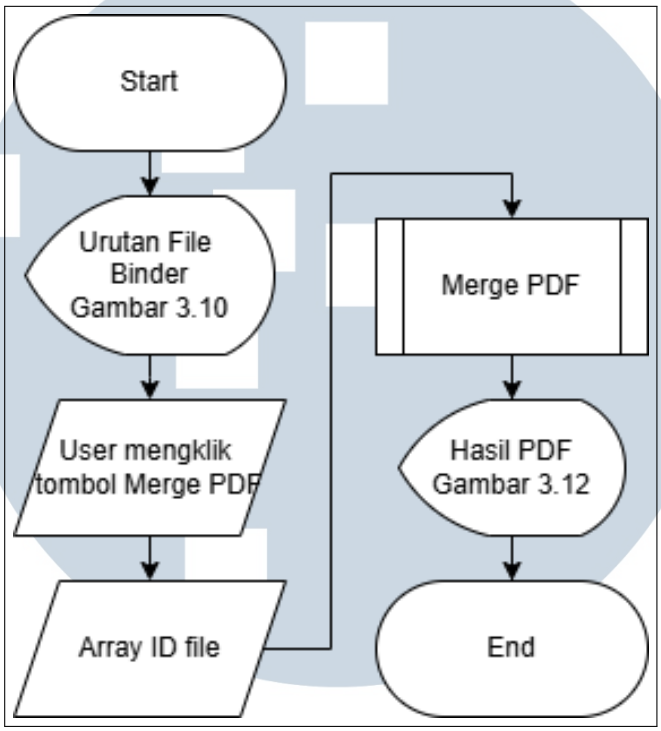
Pustaka	Kapabilitas	Kelebihan	Kekurangan
P-L	Membaca, menulis, dan menggabungkan file PDF	Tidak bergantung pada pustaka eksternal, mendukung manipulasi halaman	Kompleksitas tinggi untuk operasi besar, kurang efisien untuk penggabungan massal
P-M-J	Menggabungkan banyak file PDF secara mudah	Antarmuka sederhana, berbasis Node.js, ringan, cocok untuk *server-side*	Terbatas pada penggabungan, tidak mendukung modifikasi konten
E-P-M	Utilitas sederhana untuk penggabungan file PDF	Mudah digunakan, integrasi cepat	Bergantung pada *child process*, terbatas pada lingkungan sistem tertentu
PJ	Merender PDF di sisi klien	Ideal untuk pratinjau dokumen PDF dalam peramban	Tidak mendukung penggabungan atau modifikasi file PDF

Berdasarkan evaluasi tersebut, P-M-J dipilih sebagai pustaka utama dalam implementasi karena kesederhanaannya, dokumentasi yang memadai, dan dukungan penuh terhadap proses *merging* dokumen PDF dalam lingkungan Node.js. Pustaka ini dinilai paling sesuai dengan kebutuhan fungsional serta arsitektur sistem yang telah dirancang.

C Implementasi

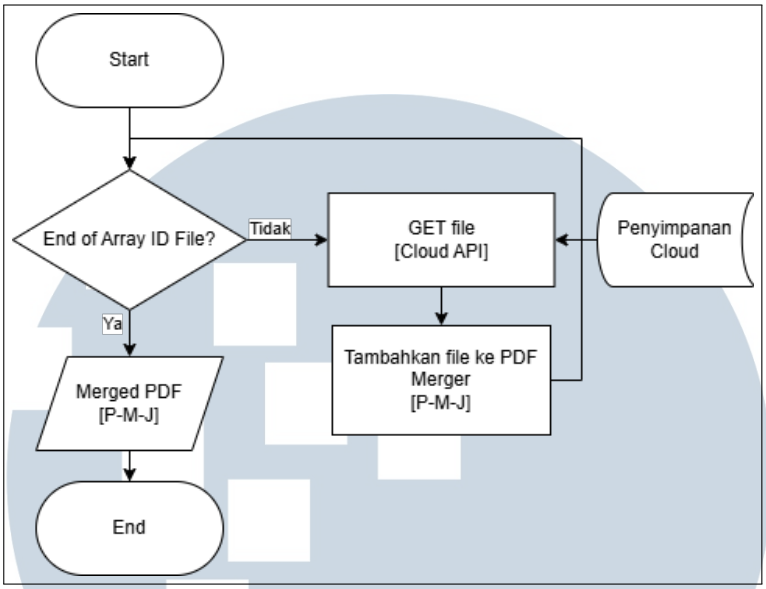
Fitur penggabungan PDF dibuat sebagai bagian dari modul *binder view*, dengan tujuan untuk menyatukan beberapa dokumen PDF ke dalam satu berkas secara otomatis berdasarkan urutan yang telah ditentukan oleh pengguna. Alur proses yang dijalankan sistem dalam menangani permintaan tersebut

divisualisasikan dalam bentuk diagram pada Gambar 3.14, yang menggambarkan komunikasi dari antarmuka pengguna ke server serta tahapan penyusunan dan penyimpanan dokumen hasil akhir.



Gambar 3.14. Flowchart fitur penggabungan PDF

Secara internal, subproses penggabungan PDF ini juga dirancang dalam bentuk diagram alur yang lebih terperinci, yang dijelaskan pada Gambar 3.15. Diagram ini menunjukkan tahapan spesifik, seperti menambahkan file PDF ke PDF *Merger*, proses iterasi berdasarkan urutan file, dan tahap akhir berupa output file PDF gabungan.



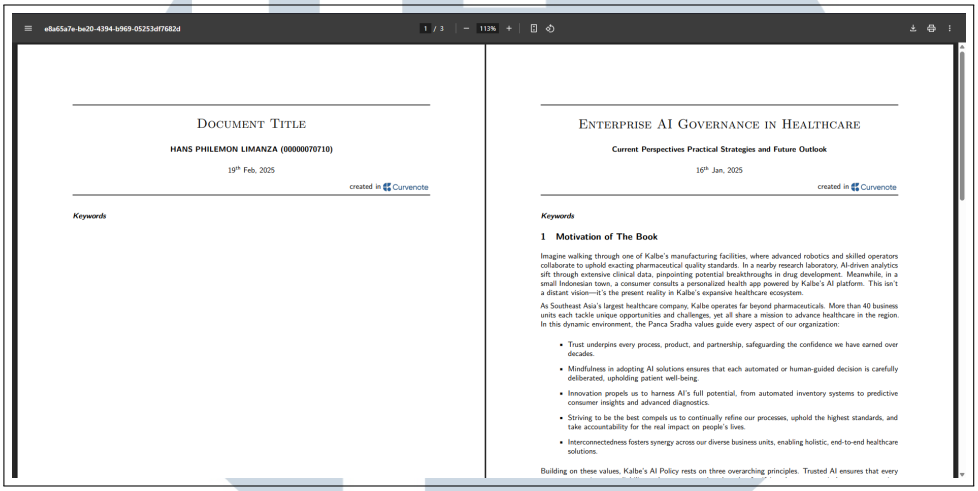
Gambar 3.15. Flowchart subproses Merge PDF

Saat pengguna membuka tampilan binder, sistem akan menampilkan daftar dokumen yang tersedia sebagaimana ditunjukkan pada Gambar 3.16. Di bagian atas antarmuka ini, tersedia tombol *Merge* PDF yang berfungsi sebagai pemicu penggabungan dokumen.

Gambar 3.16. Tampilan awal fitur penggabungan PDF

Setelah pengguna menekan tombol penggabungan, sistem akan memulai proses yang terdiri dari beberapa tahapan, yaitu mengambil ID file PDF yang terdapat dalam binder, lalu menggabungkannya secara berurutan menggunakan pustaka P-M-J. Setelah itu, file hasil penggabungan disimpan sementara di server untuk kemudian dikirimkan kembali ke sisi klien.

Setelah proses penggabungan selesai, sistem akan menampilkan hasilnya kepada pengguna dalam bentuk pratinjau file PDF. Pengguna dapat meninjau dokumen tersebut secara langsung pada antarmuka aplikasi, seperti ditampilkan dalam Gambar 3.17. File yang telah dihasilkan juga dapat diunduh untuk disimpan atau dikirimkan lebih lanjut.



Gambar 3.17. Tampilan hasil penggabungan PDF

Fitur ini sepenuhnya dijalankan di sisi server guna menjamin konsistensi format dokumen dan menjaga efisiensi pemrosesan. Pendekatan ini juga membantu mengurangi beban pemrosesan di sisi klien dan meningkatkan keandalan dalam penanganan berkas berukuran besar.

3.3.4 Fitur Drag and Drop

Pengelolaan dokumen secara visual dalam sistem informasi modern membutuhkan antarmuka interaktif yang mendukung pemindahan elemen secara intuitif. Salah satu fitur yang menjadi kebutuhan adalah *Drag and Drop*, yaitu kemampuan untuk memindahkan dokumen atau unit informasi dari satu lokasi ke lokasi lainnya secara langsung melalui antarmuka grafis. Kebutuhan ini mengalami perluasan, dari awalnya hanya untuk perpindahan antar dua daftar dalam satu tab, hingga mencakup perpindahan antar daftar pada tab peramban berbeda dengan struktur hierarkis (treeview) untuk mendukung pengelolaan folder yang lebih kompleks.

A User Requirement

Kebutuhan awal sistem mencakup pembuatan tampilan dua daftar item (mirip *kanban board*) yang mendukung perpindahan dokumen antar kolom melalui metode *Drag and Drop*. Seiring berjalannya waktu, terdapat perluasan kebutuhan yang mencakup:

1. Dukungan untuk interaksi antar daftar yang terpisah di tab browser berbeda.
2. Representasi elemen dokumen dalam struktur hierarkis (*treeview*).
3. Mekanisme pemindahan file atau folder ke posisi yang tepat dalam struktur.

Fitur ini diharapkan dapat mempercepat pengorganisasian dokumen, memperjelas hierarki, serta mempermudah manipulasi posisi file melalui antarmuka.

B Riset Pustaka / Dokumentasi

Dalam rangka memenuhi kebutuhan tersebut, dilakukan studi pustaka terhadap sejumlah pustaka *Drag and Drop* berbasis JavaScript dan React. Evaluasi pustaka dilakukan berdasarkan kriteria stabilitas, dokumentasi teknis, fleksibilitas fitur, dukungan komunitas, dan kompatibilitas dengan *framework* yang digunakan. Adapun pustaka yang ditinjau meliputi:

Tabel 3.4. Hasil evaluasi riset pustaka mekanisme Drag and Drop

Pustaka	Kapabilitas	Kelebihan	Kekurangan
F/D	Pustaka ringan untuk interaksi sederhana antar elemen	Ringan, dokumentasi sederhana	Tidak mendukung interaksi kompleks, tidak cocok untuk *treeview*
SJS	Mendukung pemindahan dan pengurutan elemen DOM	Stabil dan banyak digunakan	Integrasi terbatas dengan React, perlu *ref* manual

Tabel 3.4. Hasil evaluasi riset pustaka mekanisme Drag and Drop (lanjutan)

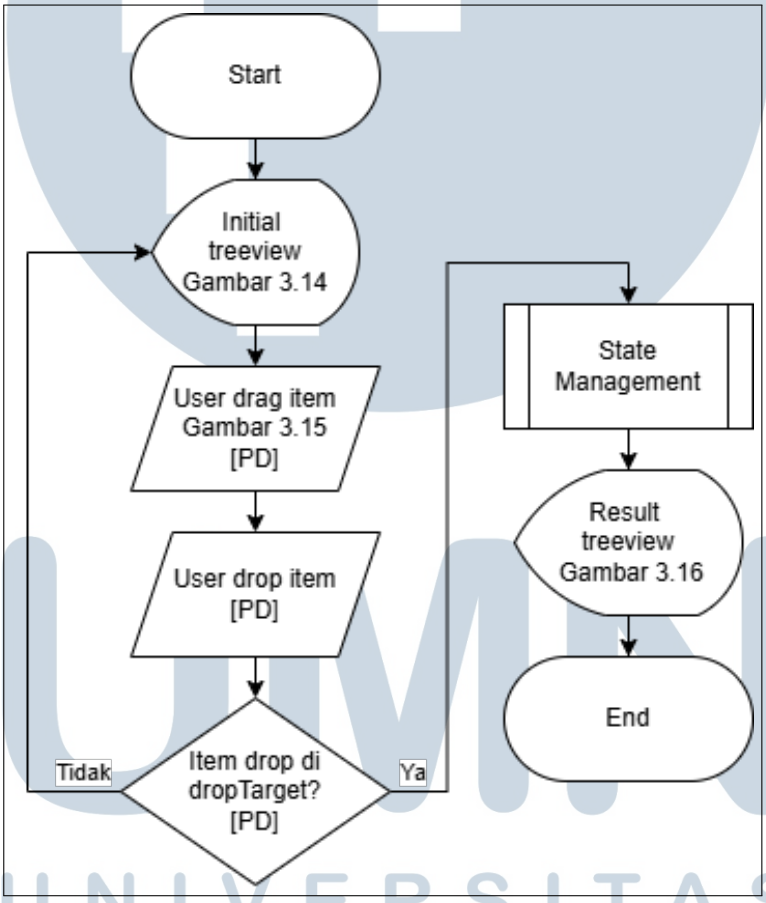
Pustaka	Kapabilitas	Kelebihan	Kekurangan
D-K	Modular, mendukung pengurutan dan struktur hierarkis	Kompatibel dengan React modern. fleksibel	Kompleks untuk implementasi inter-tab dan lintas struktur
R-D	Berdasarkan konsep *drag source* dan *drop target*	Sangat fleksibel, banyak studi kasus	Kurva belajar tinggi, konfigurasi verbose
R-Dr	Fokus pada pemindahan posisi visual elemen	Cocok untuk elemen grafis bebas (*free movement*)	Tidak mendukung pengurutan atau struktur data
H-P/D	*Fork* dari R-B-D, cocok untuk tampilan *kanban*	Mudah diimplementasikan, tampilan mulus	Tidak mendukung *drag* antar tab, terbatas pada list datar (*flat list*)
PD	Mendukung interaksi lintas tab dan struktur hierarkis berbasis *adapter*	Arsitektur modular, cocok untuk *drag* antar tab dan *treeview*	Relatif baru, dokumentasi terbatas, ekosistem belum matang

Pada tahap awal, dipilih pustaka H-P/D karena kemudahan penggunaan dan tampilan antarmuka yang halus. Pustaka ini digunakan untuk membangun antarmuka *kanban board* yang terdiri dari dua kolom daftar dokumen. Namun, setelah muncul kebutuhan untuk melakukan perpindahan lintas tab dan dukungan terhadap struktur hierarkis, pustaka ini dinilai tidak lagi memenuhi kebutuhan.

Sebagai respon terhadap kebutuhan tambahan tersebut, dilakukan migrasi ke pustaka PD. Pustaka ini mengadopsi arsitektur berbasis *Drag and Drop* API yang dimiliki oleh peramban yang memungkinkan pengembangan interaksi *cross-tab*, serta mendukung representasi data dalam bentuk *treeview*. Migrasi dilakukan dengan tetap mempertahankan struktur *kanban* untuk memudahkan proses transisi pengguna, lalu dilanjutkan dengan konversi visualisasi menjadi *treeview*.

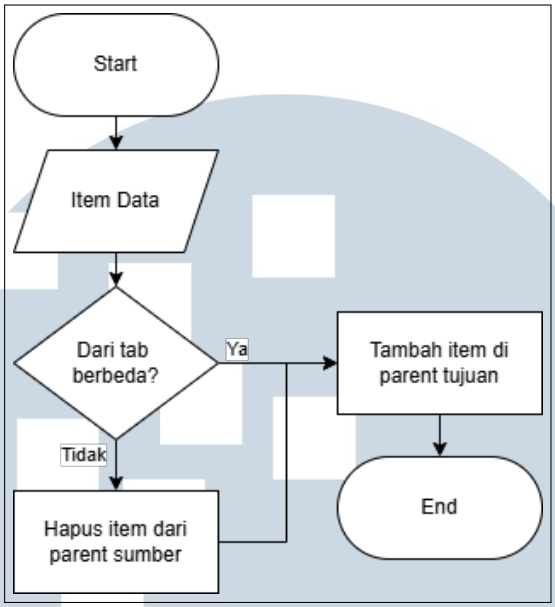
C Implementasi

Fitur *Drag and Drop* dibuat untuk memudahkan pengguna dalam mengelola struktur folder dan file secara intuitif. Fitur ini memungkinkan pemindahan elemen dalam hierarki struktur menggunakan interaksi seret dan lepas (*drag and drop*) langsung melalui antarmuka pengguna. Alur utama dari proses ini dapat dilihat pada Gambar 3.18, yang memperlihatkan bagaimana interaksi pengguna memicu serangkaian proses, mulai dari pendeteksian elemen yang diseret hingga pembaruan posisi dalam struktur data.



Gambar 3.18. Flowchart fitur Drag and Drop

Pada tingkat yang lebih rinci, proses manajemen state yang terjadi setelah aksi *drop* dijelaskan melalui Gambar 3.19. Diagram ini menggambarkan bagaimana pustaka yang digunakan (dalam hal ini PD) menangani perubahan status struktur folder dan file melalui sistem adaptif berbasis *adapter*, yang kemudian dilanjutkan dengan pembaruan data.

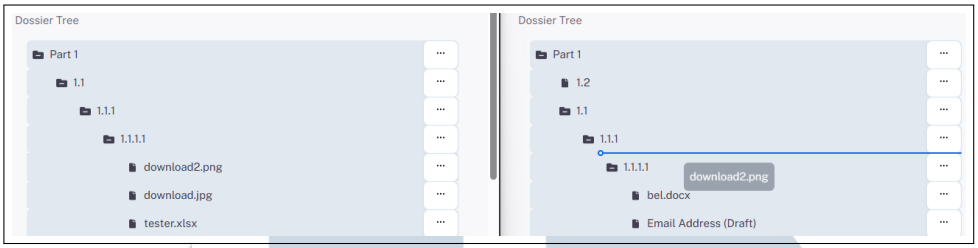


Gambar 3.19. Flowchart subproses State Management

Antarmuka awal fitur ini menampilkan struktur file dalam bentuk *treeview*, dengan folder dan dokumen tersusun secara hierarkis. Pengguna dapat memilih dan menyeret salah satu item ke posisi baru sesuai kebutuhan. Tampilan awal antarmuka ini dapat dilihat pada Gambar 3.20, yang menunjukkan struktur folder sebelum dilakukan interaksi.

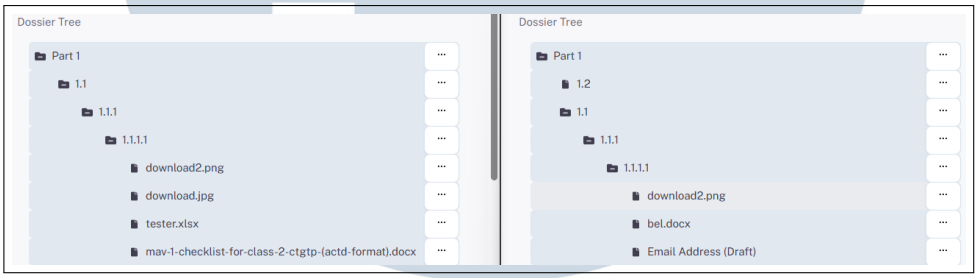
Gambar 3.20. Tampilan awal fitur Drag and Drop

Selama operasi dilakukan, item yang dipilih akan mengikuti pergerakan kursor pengguna hingga ditempatkan pada lokasi target. Proses visual dari interaksi ini terekam dalam Gambar 3.21, yang memperlihatkan status elemen ketika sedang dalam kondisi *dragging* menuju posisi baru.



Gambar 3.21. Tampilan operasi Drag item

Setelah proses *drop* dilakukan, sistem akan memperbarui susunan item secara otomatis dan menampilkan hasil akhir dari posisi baru elemen tersebut. Hasil dari aksi ini divisualisasikan pada Gambar 3.22, yang menunjukkan pergeseran item ke lokasi yang diinginkan dalam struktur *treeview*.



Gambar 3.22. Tampilan hasil operasi Drag and Drop

Fitur ini diimplementasikan untuk mendukung kebutuhan navigasi dan reorganisasi berkas dengan interaksi yang efisien, sekaligus mempertahankan struktur data secara konsisten. Penggunaan pustaka modular berbasis *adapter* memungkinkan fleksibilitas tinggi serta kemudahan adaptasi dengan sistem manajemen dokumen perusahaan yang terus berkembang.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan kerja praktik di lingkungan PT Kalbe Farma Tbk, sejumlah kendala ditemukan baik dari aspek teknis maupun non-teknis. Kendala-kendala tersebut merupakan bagian wajar dari proses pembuatan sistem berbasis teknologi informasi, khususnya dalam tahap Proof of Concept.

1. Kompleksitas Integrasi dengan Layanan Pihak Ketiga

   Integrasi aplikasi web dengan layanan eksternal, seperti sistem penyimpanan dokumen berbasis *cloud* pihak ketiga memerlukan pemahaman mendalam

terhadap dokumentasi API dan skema autentikasi yang digunakan. Beberapa kendala yang dihadapi meliputi konfigurasi perizinan akses, serta keterbatasan dokumentasi kasus penggunaan tertentu.

Solusi: Dilakukan studi dokumentasi resmi dan eksplorasi langsung melalui alat bantu, seperti Postman. Selain itu, pengujian dilakukan secara bertahap pada lingkungan pengembangan untuk memverifikasi alur autentikasi dan akses data.

2. Pengelolaan *State* pada Fitur *Drag and Drop*

   Pembuatan fitur *Drag and Drop* dalam struktur data hierarkis (seperti *treeview*) menimbulkan tantangan dalam pengelolaan *state* yang kompleks. Perpindahan antar elemen dalam konteks folder menyebabkan ketidakstabilan tampilan apabila tidak dikelola dengan benar.

   Solusi: Dilakukan refaktorisasi komponen React menjadi lebih modular dan terpisah sesuai tanggung jawab masing-masing, serta penerapan pustaka manajemen state tambahan guna menjaga konsistensi data. Pengujian dilakukan secara iteratif untuk setiap perubahan logika.