KERJA PRAKTIK - EF234701

Perancangan dan Pengembangan Modul Frontend untuk Aplikasi DroneMEQ, Portal Smart Indicator Economy, serta Aplikasi Enterprise Resource Planning

Laboratorium Manajemen Cerdas Informasi
Kampus ITS, Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
Periode: 6 Januari 2025 – 6 Juni 2025

Oleh:

Muhammad Wafi Zaki Hanif 5025221039

Pembimbing Jurusan
Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.

Pembimbing Lapangan
Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

KERJA PRAKTIK - EF234701

Perancangan dan Pengembangan Modul Frontend untuk Aplikasi DroneMEQ, Portal Smart Indicator Economy, serta Aplikasi Enterprise Resource Planning

Laboratorium Manajemen Cerdas Informasi

Kampus ITS, Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111

Periode: 6 Januari 2025 – 6 Juni 2025

Oleh:

Muhammad Wafi Zaki Hanif 5025221039

Pembimbing Jurusan

Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.

Pembimbing Lapangan

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

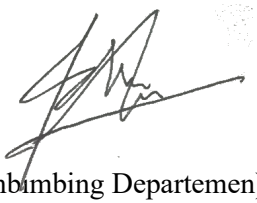
Perancangan dan Pengembangan Modul Frontend untuk Aplikasi DroneMEQ, Portal Smart Indicator Economy, serta Aplikasi Enterprise Resource Planning

Oleh:

Muhammad Wafi Zaki Hanif 5025221039

Disetujui oleh Pembimbing Kerja Praktik:

1. Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.
   NIP. 198412102014042003



(Pembimbing Departemen)

2. Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.
   NIP. 1994201912088

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perancangan dan Pengembangan Modul Frontend untuk Aplikasi DroneMEQ, Portal Smart Indicator Economy, serta Aplikasi Enterprise Resource Planning

Nama Mahasiswa	: Muhammad Wafi Zaki Hanif
NRP	: 5025221039
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.
Pembimbing Lapangan	: Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.

ABSTRAK

Laporan kerja praktik ini membahas proses perancangan, pengembangan, serta integrasi fitur frontend dari tiga sistem utama, yaitu DroneMEQ, Sistem Enterprise Resource Planning (ERP), dan Portal Smart Indicator Economy. Kegiatan kerja praktik difokuskan pada peningkatan fungsionalitas antarmuka pengguna (UI) dan integrasi sistem berbasis autentikasi serta manajemen akses. Seluruh pengembangan frontend dilakukan menggunakan Next.js dan TypeScript, melanjutkan ekosistem yang telah dibangun oleh pengembang sebelumnya.

Pada sistem DroneMEQ, penulis melakukan peningkatan fungsionalitas dengan mengubah antarmuka modul Pelanggaran dari berbasis kartu menjadi tabel untuk meningkatkan densitas informasi, menerapkan pendekatan dinamis pada integrasi backend pada modul kelola asset, serta mengimplementasikan sistem translasi multibahasa (Indonesia dan inggris). Pada sistem ERP, penulis mengimplementasikan sistem otorisasi berbasis peran yang secara dinamis menyesuaikan tampilan komponen antarmuka sesuai hak akses pengguna. Penulis juga menerapkan integrasi autentikasi melalui portal untuk aplikasi ERP. Sementara itu, Portal Smart Indicator Economy dibangun dari awal,

mencakup fitur autentikasi, dashboard, serta modul terpusat untuk manajemen pengguna, peran, dan hak akses yang terintegrasi dengan aplikasi ERP.

Kegiatan ini bertujuan untuk mengembangkan antarmuka sistem yang fungsional, terintegrasi lintas aplikasi, serta menerapkan praktik terbaik dalam pengembangan perangkat lunak berskala industri.

Hasil pengujian membuktikan bahwa pengembangan yang dilakukan berhasil meningkatkan kualitas dan fungsionalitas sistem secara signifikan. Pengujian manual menunjukkan 100% fungsionalitas berjalan sesuai skenario tanpa galat dan tidak ditemukan adanya kebocoran kredensial, yang merupakan peningkatan substansial dari kondisi sebelumnya yang memiliki bug dan integrasi statis. Lebih lanjut, evaluasi kualitas teknis menggunakan Google Lighthouse menunjukkan hasil yang sangat memuaskan. Dengan skor rata-rata Kinerja 92, Aksesibilitas 83, Best Practices 95, dan SEO 97, hasil ini secara kuantitatif membuktikan bahwa antarmuka yang baru tidak hanya fungsional, tetapi juga dibangun dengan standar modern yang tinggi, sehingga lebih efisien dan andal dibandingkan sistem sebelumnya..

Kata Kunci: DroneMEQ, ERP, Portal Smart Indicator Economy

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan kerja praktik yang menjadi salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS dengan tepat waktu. Pengalaman kerja praktik di Laboratorium Manajemen Cerdas Informasi merupakan salah satu kesempatan yang penuh dengan pembelajaran dan pengalaman baru di bidang perangkat lunak, khususnya dalam perancangan dan pembangunan *backend*.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap dengan dibuatnya buku laporan ini dapat memenuhi kewajiban penulis sebagai syarat lulus kerja praktik dan bisa bermanfaat bagi banyak orang serta berkontribusi terhadap perkembangan teknologi informasi di masa depan

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu selama proses kerja praktuk dan penyusun laporan kerja praktik baik secara langsung. Ucapan terima kasih penulis sampaikan kepada:

1. Kedua orang tua penulis, atas doa dan dukungan hingga kini kepada penulis.
2. Ibu Dr. Kelly Rossa Sungkono, S.Kom., M.Kom selaku pembimbing lapangan selama kerja praktik berlangsung.
3. Mas Kurnia. Selaku senior di Teknik Informatika yang selalu memberikan arahan dan pembelajaran baru saat proses kerja praktik.
4. Teman-teman penulis yang senantiasa membantu dan memberikan semangat ketika penulis melaksanakan kerja praktik.

Akhir kata, penulis mengucapkan terima kasih yang tak terhingga kepada semua yang telah mendukung, baik secara langsung maupun tidak langsung, dalam perjalanan ini. Semoga laporan kerja praktik ini tidak hanya mencerminkan pengalaman dan pembelajaran yang penulis peroleh, tetapi juga dapat menjadi sumber inspirasi dan pengetahuan bagi mereka yang membacanya.

Surabaya, 30 Mei 2025
Muhammad Wafi Zaki Hanif

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam era transformasi digital saat ini, pengembangan aplikasi yang memiliki antarmuka pengguna (*frontend)* yang responsif, intuitif, dan fungsional menjadi komponen kunci dalam menunjang efisiensi dan kenyamanan operasional sistem berbasis web [1]. Perusahaan dan institusi semakin bergantung pada berbagai aplikasi digital untuk mendukung kegiatan manajerial, operasional, dan pemantauan data secara real-time.

Salah satu sistem penting yang terus berkembang adalah Enterprise Resource Planning (ERP), yang menyatukan berbagai fungsi bisnis seperti inventaris, pembelian, penjualan, dan manajemen aset ke dalam satu platform yang terintegrasi [2]. Keberhasilan penerapan ERP tidak hanya ditentukan oleh kekuatan logika backend, namun juga sangat bergantung pada desain dan implementasi antarmuka pengguna yang mampu menyajikan data dan fitur secara efisien dan mudah dipahami [3].

Di sisi lain, sistem DroneMEQ (Drone Marine Environment Quality) yang berfokus pada pemantauan kualitas lingkungan laut dengan dukungan teknologi drone dan perangkat IoT, membutuhkan antarmuka frontend yang dapat menampilkan data secara real-time dan interaktif [4]. Antarmuka ini berperan penting dalam membantu pengguna memantau dan mengambil keputusan berbasis data lingkungan yang diperoleh dari lapangan.

Selain itu, sistem Smart Indicator Economy merupakan platform digital yang dirancang untuk menyajikan indikator-indikator ekonomi secara terpadu dan informatif. Di dalamnya terdapat portal utama yang berfungsi sebagai pintu masuk dan penghubung antar berbagai aplikasi turunan dalam ekosistem tersebut. Portal ini memiliki peran strategis dalam menyatukan berbagai sumber informasi, serta menyajikannya dalam format

visual yang mudah dipahami oleh pengguna dari berbagai kalangan [5].

Berdasarkan latar belakang tersebut, proyek kerja praktik ini difokuskan pada perancangan dan pengembangan modul frontend pada tiga sistem utama: (1) Aplikasi DroneMEQ, (2) Sistem Smart Indicator Economy khususnya pada bagian portal sebagai penghubung aplikasi-aplikasi turunannya, dan (3) Modul ERP yang mencakup fitur Inventory, Purchasing, Selling, dan Asset. Proyek ini mencakup proses identifikasi kekurangan pada sistem yang telah ada, pengembangan tampilan antarmuka baru, serta penyempurnaan fitur dengan tujuan meningkatkan pengalaman pengguna dan efisiensi penggunaan aplikasi.

Dengan dilaksanakannya proyek ini, diharapkan sistem-sistem tersebut dapat berjalan secara optimal dari sisi antarmuka pengguna, serta mampu memenuhi kebutuhan informasi, visualisasi data, dan pengelolaan operasional yang efektif di tengah kompleksitas tantangan digital saat ini [6].

1.2. Tujuan

Tujuan dari kegiatan kerja praktik ini adalah sebagai berikut:

1. Mengaplikasikan pengetahuan dan keterampilan yang sudah diperoleh di Teknik Informatika, khususnya di dalam bidang pengembangan perangkat lunak dalam perbaikan aplikasi DroneMEQ, perancangan portal Smart Indicator Economy serta integrasi aplikasi ERP dengan backend pada modul manajemen asset, manajemen penjualan, manajemen pembelian dan manajemen inventaris
2. Memperoleh pengalaman praktik dalam pengembangan frontend modular yang mendukung arsitektur microservices dan integrasi dengan backend melalui API.
3. Mengasah kemampuan problem solving, kolaborasi tim, dan manajemen proyek.

4. Memenuhi salah satu syarat akademis untuk mata kuliah Kerja Praktik sebagai bagian dari kurikulum Teknik Informatika di Institut Teknologi Sepuluh Nopember.

1.3. Manfaat

Manfaat yang diperoleh dari kerja praktik ini antara lain:

1. Menyediakan antarmuka yang intuitif, efisien dan terstruktur untuk mendukung aksesibilitas dan produktivitas pengguna pada sistem DroneMEQ dan ERP.
2. Mengembangkan portal utama Smart Indicator Economy sebagai gerbang penghubung antara aplikasi turunan pada sistem.
3. Penulis mendapatkan pengalaman lebih mendalam mengenai pengembangan perangkat lunak.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan sistem otorisasi berbasis peran pada antarmuka pengguna modul ERP agar dapat secara dinamis menyesuaikan fungsionalitas sesuai dengan hak akses pengguna?
2. Bagaimana merancang portal utama yang mampu mengintegrasikan dan mengarahkan pengguna ke berbagai aplikasi dalam ekosistem Smart Indicator Economy secara konsisten dan terpusat?
3. Bagaimana mengembangkan tampilan frontend untuk sistem DroneMEQ agar mampu menampilkan data pemantauan lingkungan laut secara real-time dengan user experience yang optimal?
4. Bagaimana proses pengembangan frontend dilakukan agar selaras dengan prinsip integrasi backend melalui API, serta kompatibel dengan arsitektur microservices?

5. Bagaimana hasil evaluasi fungsionalitas, performa, dan aksesibilitas dari antarmuka pengguna yang telah dikembangkan pada aplikasi DroneMEQ, sistem ERP, dan Portal Smart Indicator Economy?

1.5. Lokasi dan Waktu Kerja Praktik

Kegiatan kerja praktik dilaksanakan secara *remote*. Adapun kerja praktik dilaksanakan pada periode tanggal 6 Januari 2025 hingga 6 Juni 2025

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan kerja praktik meliputi :

1.6.1. Perumusan Masalah

Tahap awal dimulai dengan identifikasi kebutuhan dari sistem-sistem yang akan dikerjakan.

Pada aplikasi DroneMEQ, dilakukan peninjauan terhadap beberapa modul yang telah berjalan untuk menemukan kekurangan pada sisi antarmuka pengguna. Selanjutnya, dilakukan perbaikan-perbaikan minor pada tampilan dan fungsionalitas sesuai dengan kebutuhan tim pengembang.

Untuk Portal Smart Indicator Economy, karena sistem ini dirancang dari awal, maka dilakukan perancangan penuh mulai dari struktur tampilan, layout halaman, navigasi, hingga implementasi desain UI/UX yang konsisten sebagai portal penghubung antara berbagai aplikasi ekonomi indikator.

Pada aplikasi ERP, fokus kerja praktik berada pada tahap integrasi frontend dengan backend yang telah tersedia, khususnya untuk modul Inventory, Purchasing, Selling, dan Asset. Pekerjaan ini mencakup konsumsi API, penanganan data, dan penyajian antarmuka agar sesuai dengan alur bisnis dari masing-masing modul.

1.6.2. Studi Literatur

Studi literatur dilakukan untuk memperkuat pemahaman teknis, khususnya pada:

1. Pengembangan frontend modern menggunakan Next.js dan Tailwind CSS.
2. Prinsip desain UI/UX untuk aplikasi enterprise dan dashboard.
3. Teknik konsumsi dan integrasi REST API.
4. Standar struktur folder, reusable component, dan penulisan kode frontend agar maintainable dalam tim.

1.6.3. Analisis dan Perancangan Sistem

Tahap ini berfokus pada perancangan komponen dan struktur antarmuka pengguna. Untuk portal Smart Indicator Economy, perancangan dilakukan dari nol dengan pendekatan modular menggunakan Next.js dan Tailwind CSS, disertai pertimbangan responsivitas dan aksesibilitas. Sedangkan pada DroneMEQ, analisis dilakukan terhadap komponen yang sudah ada untuk diidentifikasi kekurangan atau ketidaksesuaian UI yang perlu diperbaiki. Pada sistem ERP, dilakukan analisis terhadap kebutuhan tampilan untuk menyajikan data dari API backend. Tidak dilakukan perancangan dari awal, tetapi lebih kepada penyusunan struktur frontend yang sesuai dengan data yang dikonsumsi.

1.6.4. Implementasi Sistem

Implementasi dilakukan dengan membangun atau memperbarui komponen frontend berdasarkan desain yang telah ditentukan.

Teknologi utama yang digunakan:

1. *Next.js* untuk struktur dan *routing* aplikasi *frontend*
2. *Tailwind CSS* untuk *styling* antarmuka yang fleksibel dan efisien
3. *Axios* atau *React Query* untuk konsumsi *API* dari *backend*

4. Setiap fitur diuji integrasinya dengan *backend* serta disesuaikan tampilannya untuk memastikan *user experience* yang optimal.

1.6.5. Pengujian dan Evaluasi

Setelah implementasi selesai, dilakukan pengujian fungsionalitas tampilan dan integrasi data dari API. Pengujian dilakukan baik secara manual maupun dengan feedback dari tim pengembang lain.

Evaluasi dilakukan untuk menyesuaikan kembali jika ada tampilan yang tidak sesuai, bug pada integrasi API, atau kebutuhan tambahan dari sisi frontend. Hasil evaluasi menjadi acuan untuk iterasi perbaikan.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan. Namun, ada saran untuk pengembangan kedepannya, termasuk area yang perlu peningkatan berdasarkan pengalaman kerja praktik dari penulis.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum mengenai Laboratorium Manajemen Cerdas Informasi, meliputi profil laboratorium dan lokasi laboratorium.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Rincian Kerja Praktik

Bab ini berisi rincian kegiatan yang dilakukan selama periode kerja praktik.

1.7.5. Bab V Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.6. Bab VI Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.7. Bab VII Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.8. Bab VIII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran terhadap aplikasi dan sistem yang dikembangkan selama proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil Laboratorium Manajemen Cerdas Informasi

Laboratorium Manajemen Cerdas Informasi merupakan salah satu laboratorium di Teknik Informatika ITS yang berfokus pada analisis, sintesis, dan evaluasi proses bisnis serta sistem informasi dalam lingkungan Enterprise. Laboratorium ini mengembangkan keahlian dalam rekayasa pengetahuan untuk diaplikasikan dalam pengembangan sistem cerdas, sekaligus melakukan investigasi, pengujian, serta evaluasi kematangan dan kepatutan prosedur standar serta tata kelola teknologi informasi.

Selain itu, laboratorium ini juga meneliti dan mengembangkan metode tata kelola proyek dan sumber daya manusia, guna mendukung efektivitas pengelolaan teknologi dalam organisasi. Dalam bidang data, laboratorium ini berperan dalam perancangan dan implementasi solusi basis data terdistribusi serta eksplorasi teknologi Big Data untuk menghasilkan sistem informasi yang lebih efisien dan adaptif di era digital.

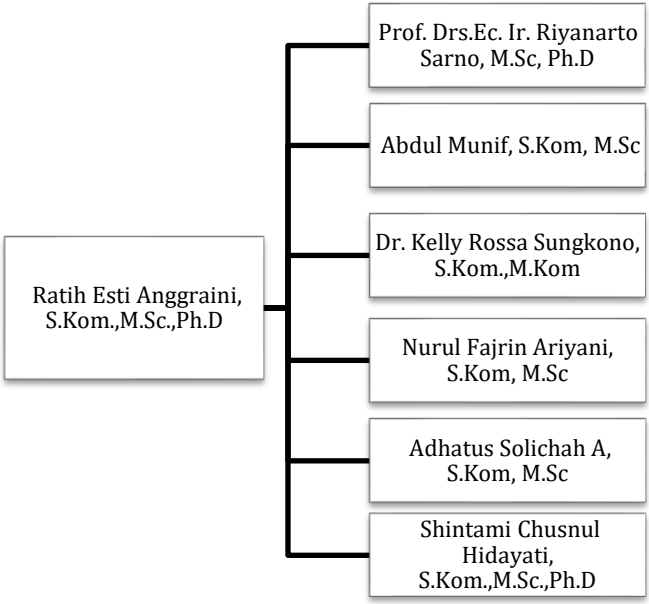
2.2. Lokasi

Laboratorium Manajemen Cerdas Informasi berlokasi di Kampus ITS, Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111.

2.3. Struktur Organisasi

Struktur organisasi di Lab Manajemen Cerdas Informasi Departemen Teknik Informatika dirancang secara sederhana namun efektif untuk mendukung kegiatan akademik dan riset di bidang manajemen informasi dan teknologi cerdas. Lab ini dipimpin oleh seorang Kepala Laboratorium yang bertanggung

jawab atas koordinasi seluruh aktivitas laboratorium, termasuk pengembangan program kerja, pengelolaan sumber daya, serta pembinaan terhadap anggota lab. Seluruh anggota laboratorium terdiri dari dosen-dosen di lingkungan Departemen Teknik Informatika yang memiliki minat dan keahlian dalam bidang terkait. Kolaborasi antara kepala lab dan anggota lab ini membentuk struktur kerja yang solid, mendukung terciptanya suasana akademik yang produktif serta mendorong kontribusi aktif dalam pengajaran, penelitian, dan pengabdian kepada masyaraka



Gambar 2.1 Bagan Dosen Lab MCI Teknik Informatika ITS

Struktur organisasi laboratorium disajikan pada Gambar 2.1 dengan Kepala Laboratorium pada posisi sebelah kiri, sementara anggota laboratorium tersusun secara vertikal di sebelah kanan. Setiap anggota terhubung secara langsung dengan

Kepala 8 Laboratorium, mencerminkan hubungan koordinasi yang bersifat tunggal dan hierarkis dalam satu tingkat.

2.4. Proyek Aktif

Laboratorium Cerdas Informasi saat ini memiliki sejumlah proyek aktif yang sedang berada dalam tahap pengembangan antara lain.

1. **ITS Smart Farming**

   ITS Smart Farming adalah aplikasi yang berfokus di bidang agrikultur dengan mengintegrasikan teknologi IoT untuk monitoring secara real time terkait kualitas.

2. **Brains**

   Brains merupakan sistem yang berfokus pada bidang kesehatan otak dengan mengintegrasikan teknologi IoT dan Machine Learning untuk mendukung praktik pembedahan.

3. **DroneMEQ**

   DroneMEQ atau Drone Marine Environment Quality adalah aplikasi manajemen dan monitoring berbasis IoT yang membantu dalam konservasi lingkungan khususnya lingkungan laut.

4. **Lecsens**

   Lecsens adalah layanan SaaS yang terintegrasi dengan IoT untuk monitoring kualitas air.

5. **ERP SCM CRM SRM**

   ERP, SCM, CRM, dan SRM adalah layanan SaaS manajemen terintegrasi yang mendukung proses bisnis utama dalam organisasi.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

Bab ini membahas berbagai pendekatan dan teknologi yang digunakan dalam pengembangan frontend pada aplikasi DroneMEQ, Portal Smart Indicator Economy, serta integrasi antarmuka pada sistem ERP. Untuk aplikasi DroneMEQ, perbaikan frontend dilakukan dengan menyesuaikan modul-modul yang telah ada agar lebih sesuai dengan standar tampilan dan fungsionalitas terkini. Sedangkan pada Portal Smart Indicator Economy, pengembangan dilakukan dari awal, termasuk perancangan tampilan, navigasi, dan antarmuka pengguna secara menyeluruh. Untuk sistem ERP, integrasi frontend dilakukan terhadap modul Inventory, Purchasing, Selling, dan Asset, dengan fokus pada penyajian data dari backend melalui konsumsi API.

Seluruh pengembangan frontend dilakukan menggunakan framework Next.js dan pustaka CSS utility-first Tailwind CSS. Kedua teknologi ini mendukung pendekatan komponen modular, yang mempermudah kolaborasi tim dan perawatan sistem jangka panjang. Selain itu, proses pengembangan frontend mengikuti prinsip-prinsip integrasi dengan microservices di sisi backend, sehingga komunikasi antar modul tetap konsisten dan skalabel.

Dalam praktiknya, untuk menjaga alur kerja yang terstruktur selama rentang waktu pengembangan lima bulan, tim menerapkan pendekatan koordinasi rutin. Rapat koordinasi diadakan secara berkala dua kali seminggu (setiap hari Selasa dan Kamis) untuk membahas kemajuan, mengidentifikasi kendala, dan menyelaraskan tugas antar anggota tim. Proses deployment dan integrasi selanjutnya dilakukan secara otomatis melalui pipeline CI/CD menggunakan Jenkins dan Docker, meskipun penulis lebih fokus pada sinkronisasi frontend dengan layanan backend yang telah tersedia.

Penjelasan lebih lanjut mengenai teknologi, pendekatan, serta metodologi yang digunakan akan dibahas pada subbab-subbab berikut.

3.1. Single Sign-On (SSO)

Single Sign-On (SSO) adalah sistem otentikasi yang memungkinkan pengguna untuk masuk sekali dengan satu set kredensial (seperti nama pengguna dan kata sandi) dan kemudian mendapatkan akses ke beberapa aplikasi atau sistem tanpa perlu melakukan login berulang kali [7].

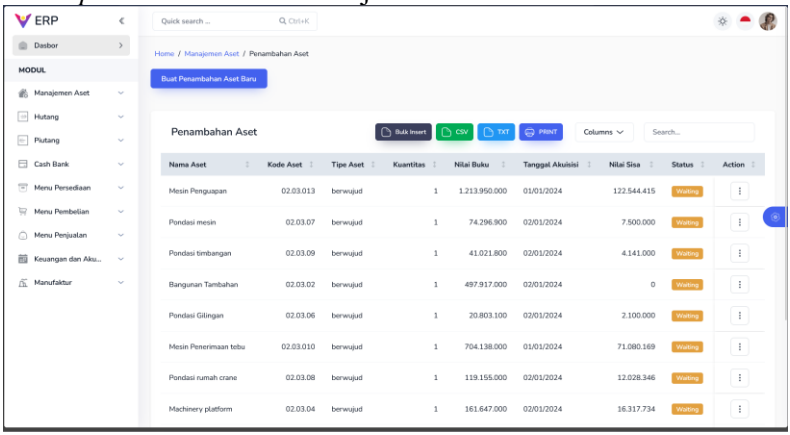
SSO bekerja dengan menyimpan dan mengelola informasi otentikasi secara terpusat. Ketika pengguna sudah diautentikasi di satu sistem, mereka tidak perlu melakukan otentikasi lagi untuk mengakses sistem lain yang terhubung.

3.2. Enterprise Resource Planning (ERP)

Enterprise Resource Planning (ERP) adalah sistem informasi terintegrasi yang menghubungkan berbagai fungsi bisnis dalam sebuah organisasi. Menurut Monk dan Wagner (2013), ERP merupakan perangkat lunak inti yang digunakan perusahaan untuk mengintegrasikan dan mengkoordinasikan informasi di setiap area bisnis [2].

ERP mengkonsolidasikan semua operasi bisnis ke dalam satu sistem komputer yang terpadu, memungkinkan data mengalir secara lancar antar departemen. Sistem ini biasanya mencakup modul seperti keuangan, rantai pasokan, hubungan pelanggan, sumber daya manusia, dan inventaris. Sebagai contoh, Gambar 3.2

adalah website sistem ERP yang sedang dikembangkan oleh tim *developer* Laboratorium Manajemen Cerdas dan Informasi ITS.

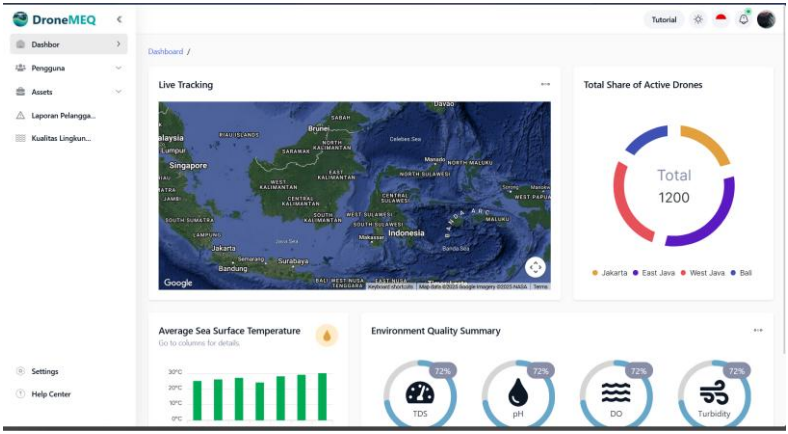


Gambar 3.1 Tampilan dashboard website ERP

3.3. DroneMEQ (*Drone Marine Environment Quality*)

DroneMEQ adalah platform teknologi komprehensif yang dirancang khusus untuk memfasilitasi dan meningkatkan efektivitas observasi lingkungan pesisir dan laut. Sistem ini mengintegrasikan teknologi drone, IoT, dan sistem manajemen data berbasis cloud untuk membantu surveyor dan peneliti dalam pengumpulan data dan pemantauan kondisi ekosistem secara efisien dan akurat [4]. Sebagai contoh, pada Gambar 3.3 adalah tampilan dashboard website DroneMEQ yang sedang

dikembangkan oleh tim *developer* Laboratorium Manajemen Cerdas dan Informasi ITS.

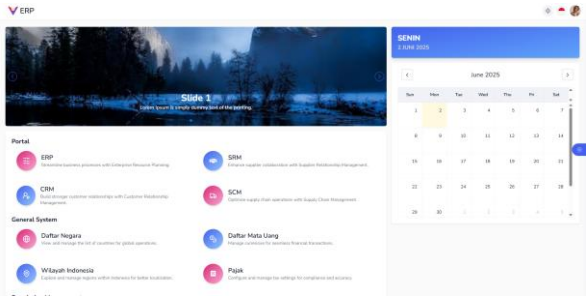


Gambar 3.2 Tampilan dashboard website DroneMEQ

3.4. Portal Smart Indicator Economy

Portal Smart Indicator Economy merupakan aplikasi web yang dirancang sebagai gerbang utama untuk mengakses seluruh aplikasi yang tergabung dalam ekosistem tersebut. Portal ini berfungsi sebagai pusat integrasi dan autentikasi, memungkinkan pengguna untuk login satu kali melalui SSO [4], sehingga meningkatkan efisiensi dan keamanan akses.

Portal juga mengelola hak akses lintas aplikasi dan menyediakan navigasi terintegrasi ke modul-modul yang tersedia dalam sistem [5]. Sebagai contoh, pada Gambar 3.4 adalah tampilan dashboard website Portal Smart Indicator Economy yang sedang dikembangkan oleh tim *developer* Laboratorium Manajemen Cerdas dan Informasi ITS.



Gambar 3.3 Tampilan dashboard website Potal Smart Indicator Economy

3.5. Typescript

TypeScript adalah bahasa pemrograman yang dikembangkan oleh Microsoft sebagai superset dari JavaScript, menambahkan fitur static typing. TypeScript membantu meningkatkan skalabilitas, mengurangi bug, dan mempercepat pengembangan melalui integrasi alat bantu seperti autocompletion dan type inference [8].

Bahasa ini mendukung fitur modern JavaScript serta menambahkan fitur seperti interface, generic types, dan compile-time checking.

3.6. Next.js

Next.js adalah framework React yang dikembangkan oleh Vercel, dirancang untuk mendukung Server-Side Rendering (SSR), Static Site Generation (SSG), dan Client-Side Rendering (CSR) dalam satu proyek. Kemampuan ini membuat Next.js cocok untuk aplikasi modern yang membutuhkan performa tinggi dan SEO-friendly [9]. Kelebihan lain termasuk optimisasi gambar otomatis, code splitting, dan prefetching halaman.

3.7. Tailwind CSS

Tailwind CSS adalah framework CSS utility-first yang memungkinkan pengembang membuat antarmuka pengguna dengan cepat dan fleksibel langsung dalam markup HTML atau JSX. Alih-alih menyediakan komponen siap pakai, Tailwind menyediakan kelas-kelas utility yang dapat dikombinasikan secara bebas untuk membentuk desain [13].

3.8. Google Lighthouse

Google Lighthouse adalah sebuah tool open-source otomatis yang dikembangkan oleh Google untuk membantu pengembang meningkatkan kualitas halaman web secara keseluruhan. Lighthouse menjalankan serangkaian audit komprehensif terhadap sebuah halaman web, kemudian menghasilkan laporan terperinci yang memberikan skor serta masukan untuk perbaikan [14].
Lighthouse menganalisis halaman web dalam beberapa kategori utama:

1. ***Performance*** **(Performa)**
   Mengukur seberapa cepat halaman dimuat dan menjadi interaktif bagi pengguna. Metrik utama yang diukur antara lain Largest Contentful Paint (LCP) yang mengukur persepsi kecepatan muat, Total Blocking Time (TBT) yang mengukur responsivitas, dan Cumulative Layout Shift (CLS) yang mengukur stabilitas visual.
2. ***Accessibility*** **(Aksesibilitas)**
   Memeriksa apakah halaman dapat diakses dan digunakan oleh semua orang, termasuk pengguna dengan disabilitas. Audit ini mencakup pengecekan kontras warna, label pada elemen formulir, dan struktur HTML yang semantik.
3. ***Best Practices*** **(Praktik Terbaik)**
   Mengevaluasi apakah halaman mengikuti praktik terbaik dalam pengembangan web modern, seperti penggunaan HTTPS,

tidak adanya error pada konsol browser, dan keamanan pustaka JavaScript.

4. *Search Engine Optimization* (SEO)

Menganalisis sejauh mana halaman dioptimalkan untuk dapat ditemukan oleh mesin pencari. Audit ini mencakup pengecekan dasar seperti keberadaan tag <title> dan meta description.

Google Lighthouse dapat diakses dengan mudah melalui *Google Chrome DevTools*, sebagai ekstensi *browser*, atau dijalankan melalui *command line*. Hasil audit tidak hanya berupa skor, tetapi juga memberikan rekomendasi dan langkah-langkah konkret yang dapat diambil untuk memperbaiki setiap masalah yang ditemukan. Hal ini menjadikannya alat yang sangat penting dalam siklus pengembangan untuk mengidentifikasi dan mengatasi isu kualitas secara proaktif.

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1. Lingkup Pekerjaan

Lingkup kerja praktik ini berfokus pada pengembangan dan penyempurnaan antarmuka pengguna (frontend) dari tiga sistem utama yang saling terintegrasi, yaitu DroneMEQ, Portal Smart Indicator Economy, dan *Enterprise Resource Planning* (ERP). Pada aplikasi DroneMEQ, penulis bertanggung jawab atas penyempurnaan berbagai fitur penting seperti manajemen pelanggaran, manajemen aset, pemilihan bahasa, serta halaman autentikasi yang mencakup fitur *login,* registrasi dan lupa kata sandi. Kemudian penulis juga melakukan integrasi sistem *cookies* untuk mendukung pengelolaan sesi pengguna secara aman dan efisien.

Pada sistem Portal Smart Indicator Economy, penulis merancang dan membangun sistem frontend dari awal. Pengerjaan dimulai dari halaman autentikasi yang mencakup fungsionalitas registrasi, *login*, dan lupa kata sandi. Setelah itu, penulis merancang halaman dashboard utama, fitur mengelola profil pribadi, fitur melihat informasi umum, serta mengakses aplikasi pada ekosistem Smart Indicator Economy yang terintegrasi. Untuk kebutuhan administrasi, penulis juga membangun modul khusus yang mencakup fitur untuk mengelola pengguna, mengelola peran dan mengelola hak akses.

Sementara itu, pada aplikasi ERP, fokus pekerjaan penulis adalah pada integrasi dan penerapan permission authorization pada beberapa modul utama yaitu manajemen aset, manajemen inventaris, pembelian, dan penjualan. Implementasi ini memastikan bahwa hak akses pengguna disesuaikan berdasarkan

peran yang dimiliki, sejalan dengan prinsip keamanan dan manajemen akses berbasis peran dalam sistem enterprise.

4.2. Analisis Sistem

Bagian ini membahas analisis sistem berdasarkan ruang lingkup kerja praktik yang dilakukan penulis. Fokus utama berada pada tiga sistem terintegrasi, yaitu aplikasi DroneMEQ, Portal Smart Indicator Economy, dan sistem ERP. Analisis dilakukan terhadap kebutuhan fungsional, non-fungsional, serta kebutuhan spesifik proyek yang disesuaikan dengan kontribusi penulis dalam pengembangan antarmuka pengguna (frontend) dari masing-masing sistem.

4.2.1. Definisi Umum Aplikasi

4.2.1.1. DroneMEQ

DroneMEQ merupakan aplikasi berbasis web yang digunakan oleh lembaga konservasi untuk memantau dan mengelola data lingkungan, khususnya melalui perangkat drone yang ditanamkan di wilayah pesisir.

4.2.1.2. Portal Smart Indicator Economy

Portal Smart Indicator Economy merupakan aplikasi yang berfungsi sebagai gerbang utama menuju berbagai aplikasi yang terintegrasi. Portal ini terhubung langsung dengan aplikasi yang termasuk dalam ekosistem Smart Indicator Economy dan bertindak sebagai pusat kendali akses terhadap aplikasi turunan.

4.2.1.3. *Enterprise Resource Planning* (ERP)

Aplikasi *Enterprise Resource Planning* (ERP) adalah sistem informasi terintegrasi yang dirancang untuk menghubungkan dan mengelola seluruh proses bisnis inti dalam sebuah organisasi. Aplikasi ini mengkonsolidasikan berbagai operasi ke dalam satu platform terpadu, yang memungkinkan data mengalir secara lancar antar departemen.

4.2.2. Analisis Kebutuhan Fungsional

4.2.2.1 DroneMEQ

1. Kebutuhan Fungsional Pengguna Umum

a. Melakukan *Login*.

Sistem harus menyediakan fungsionalitas autentikasi dasar yang memungkinkan pengguna untuk masuk ke aplikasi dengan kredensial yang valid.

b. Melakukan Registrasi.

Kebutuhan fungsional ini dibutuhkan sebagai fungsionalitas autentikasi dasar yang memungkinkan pengguna untuk mendaftarkan akun baru ke dalam sistem.

c. Melakukan Pemulihan Akun.

Sistem harus menyediakan fungsionalitas autentikasi dasar yang membantu pengguna untuk memulihkan akun, guna menghindari masalah kehilangan akun.

2. Kebutuhan Fungsional Pengguna Terdaftar

a. Mengelola Data Pelanggaran.

Pengguna yang telah terautentikasi dapat melakukan operasi *Create, Read, Update dan Delete* (CRUD) terhadap data laporan pelanggaran di wilayah laut.

b. Mengelola Data Aset.

Pengguna yang telah terautentikasi dapat melakukan operasi *Create, Read, Update* dan *Delete* (CRUD) terhadap data aset.

c. Memilih Bahasa.

Aplikasi harus menyediakan fitur translasi multibahasa (Indonesia dan Inggris) yang memungkinkan pengguna mengubah bahasa antarmuka sesuai preferensi.

4.2.2.2 Portal Smart Indicator Economy

1. Kebutuhan Fungsional untuk Pengguna Umum

a. Melakukan *Login*.

Sistem harus menyediakan fungsionalitas autentikasi dasar yang memungkinkan pengguna untuk masuk ke aplikasi dengan kredensial yang valid.

b. Melakukan Registrasi.

Kebutuhan fungsional ini dibutuhkan sebagai fungsionalitas autentikasi dasar yang memungkinkan pengguna untuk mendaftarkan akun baru ke dalam sistem.

c. Melakukan Pemulihan Akun

Sistem harus menyediakan fungsionalitas autentikasi dasar yang membantu pengguna untuk memulihkan akun, guna menghindari masalah kehilangan akun.

2. Kebutuhan Fungsional untuk Pengguna Terdaftar

a. Mengakses Dashboard

Setelah login, pengguna dapat mengakses halaman dashboard utama yang berfungsi sebagai pusat navigasi dan menampilkan informasi ringkas.

b. Mengelola Profil Pribadi

Pengguna dapat melihat dan memperbarui informasi personal mereka, seperti nama, email, serta melihat riwayat aktivitas di dalam sistem.

c. Melihat Informasi Umum

Pengguna memiliki akses hanya-baca (read-only) ke data referensi sistem, seperti daftar negara, mata uang, pajak, dan wilayah.

d. Mengakses Aplikasi Terintegrasi

Sistem memungkinkan pengguna untuk bernavigasi ke aplikasi turunan (seperti ERP) yang hak aksesnya telah diberikan, tanpa perlu login ulang (Single Sign-On).

3. Kebutuhan Fungsional untuk Super Admin

a. Melakukan Fungsi Seluruh Pengguna

Super Admin adalah seorang pengguna dengan hak akses tertinggi. Oleh karena itu, *Super Admin* juga bisa melakukan seluruh fungsi yang bisa dilakukan oleh pengguna.

b. Mengelola Hak Akses Portal

Super Admin dapat melakukan operasi *Create, Read, Update* dan *Delete* (CRUD) kepada hak akses setiap peran (*role*) pada aplikasi portal. Pada fungsi ini, *Super Admin* dapat melakukan operasi tersebut kepada :

- Hak Akses (*Permission).*
- Peran (*Role).*
- Pengguna (*User).*

c. Mengelola Hak Akses Aplikasi

Super Admin dapat melakukan operasi *Create, Read, Update* dan *Delete* (CRUD) pada hak akses setiap peran (*role)* yang terdaftar pada aplikasi turunan ekosistem Smart Indicator Economy (sementara hanya ERP).

4.2.2.3 *Enterprise Resource Planning* (ERP)

1. Kebutuhan Fungsional Pengguna

a. Melakukan *Login*.

Aplikasi ERP harus dapat memverifikasi sesi login pengguna. Sistem akan menampilkan peringatan dan mengembalikan pengguna ke halaman *login* pada portal jika belum memiliki kredensial yang valid. Kemudian untuk memenuhi konsep *Single Sign-On* dan memvalidasi kredensial pengguna, sistem akan melewati halaman ini kepada pengguna yang terautentikasi dari portal.

b. Mengelola Data Aset

Pengguna dapat melakukan operasi *Create, Read, Update* dan *Delete* (CRUD) pada data aset sesuai dengan hak akses yang diberikan.

c. Mengelola Data Inventaris

Pengguna dapat melakukan operasi Create, Read, Update dan Delete (CRUD) pada data inventaris sesuai dengan hak akses yang diberikan.

d. Mengelola Data Pembelian

Pengguna dapat melakukan operasi Create, Read, Update dan Delete (CRUD) pada data pembelian sesuai dengan hak akses yang diberikan.

e. **Mengelola Data Penjualan**

Pengguna dapat melakukan operasi Create, Read, Update dan Delete (CRUD) pada data penjualan sesuai dengan hak akses yang diberikan.

4.2.3. Analisis Kebutuhan Non Fungsional

1. Keamanan

Penggunaan sistem otentikasi dan otorisasi berbasis peran; pengelolaan sesi menggunakan *cookie* untuk menjaga keamanan akses.

2. Modularitas

Antarmuka frontend dibangun modular agar mudah dikembangkan dan dikelola.

3. Skalabilitas

Desain UI mendukung penambahan modul dan integrasi ke sistem baru di masa mendatang.

4. Integrasi

Aplikasi dirancang untuk dapat berkomunikasi secara efisien melalui API *backend* terpusat.

4.2.4. Analisis Kebutuhan Spesifik Proyek

4.2.4.1 Metodologi Pengembangan

Pengembangan sistem ini menerapkan pendekatan pengembangan terstruktur untuk memastikan proses kerja berjalan kolaboratif dan iteratif. Alur kerja yang penulis ikuti mencakup beberapa praktik utama:

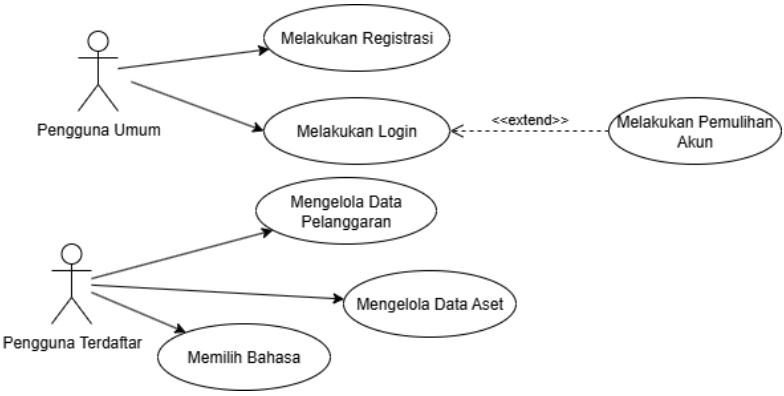
- **Perencanaan Tugas:** Sebelum memulai siklus kerja, tim menentukan daftar tugas (backlog) yang akan dikerjakan dan melakukan estimasi waktu pengerjaan.
- **Koordinasi Rutin:** Dilakukan secara berkala setiap hari Selasa dan Kamis untuk melaporkan progres, membahas kendala, menyelaraskan langkah selanjutnya dan mendemonstrasikan hasil pekerjaan.
- **Pengelolaan Tugas:** Seluruh tugas dari perencanaan hingga selesai dikelola secara transparan menggunakan papan *Trello*, diurutkan berdasarkan prioritas.

4.2.5. Diagram Use Case

Untuk memvisualisasikan interaksi antara pengguna (aktor) dengan fungsionalitas utama sistem, dibuatlah diagram use case. Diagram ini menggambarkan fungsi-fungsi yang tersedia pada aplikasi DroneMEQ, Portal Smart Indicator Economy, dan aplikasi *Enterprise Resource Planning* (ERP) berdasarkan kebutuhan fungsional yang telah dianalisis sebelumnya.

1. Diagram Use Case Aplikasi DroneMEQ

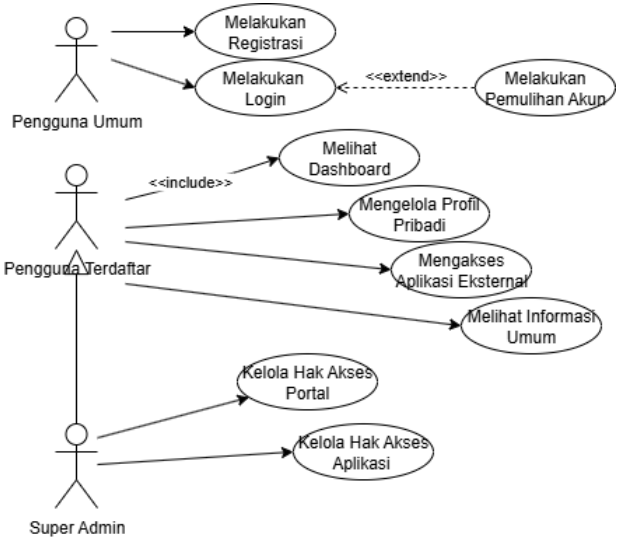
Pada Gambar 4.1, diberikan gambar yang menampilkan diagram use case pada aplikasi DroneMEQ sesuai dengan analisis kebutuhan fungsional.



Gambar 4.1 Use Case Diagram Aplikasi DroneMEQ

2. Diagram Use Case Portal Smart Indicator Economy

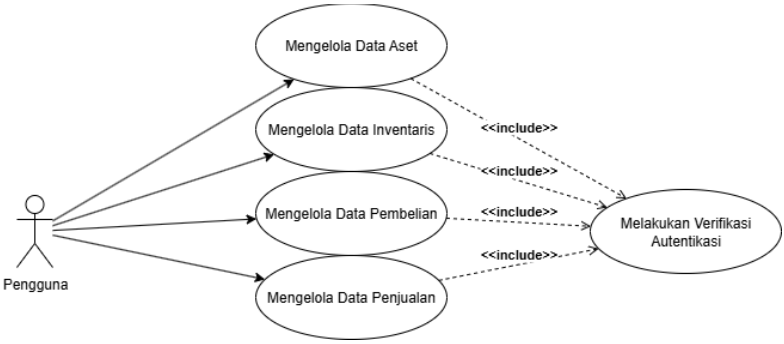
Pada Gambar 4.2, diberikan gambar yang menampilkan diagram use case pada aplikasi Portal Smart Indicator Economy sesuai dengan analisis kebutuhan fungsional.



Gambar 4.2 Diagram Use Case Aplikasi Portal Smart Indicator Economy

3. Diagram Use Case Aplikasi *Enterprise Resource Planning* (ERP)

Pada Gambar 4.3, diberikan gambar yang menampilkan diagram use case pada aplikasi *Enterprise Resource Planning* (ERP) sesuai dengan analisis kebutuhan fungsional.



Gambar 4.3 Diagram Use Case Aplikasi *Enterprise Resource Planning* (ERP)

4.3. Perancangan Antarmuka Pengguna

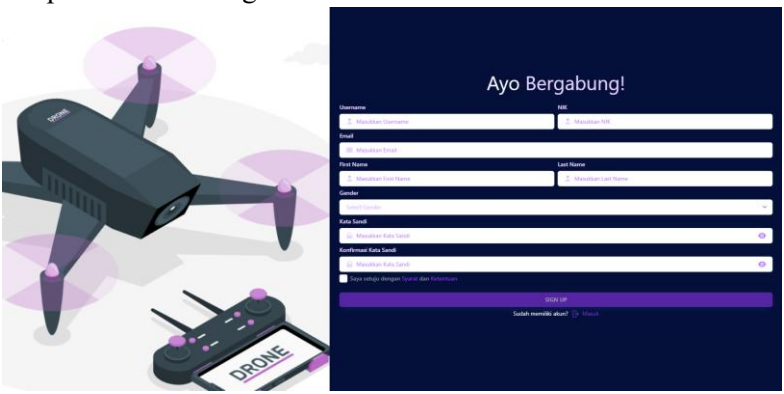
Sub-bab ini menyajikan perancangan antarmuka pengguna untuk fungsionalitas-fungsionalitas utama yang telah dianalisis sebelumnya. Perancangan ini berfungsi sebagai cetak biru visual yang memandu proses implementasi sistem. Desain antarmuka difokuskan pada kemudahan penggunaan (usability) dan alur interaksi yang intuitif. Berikut adalah rancangan antarmuka untuk setiap aplikasi yang dikembangkan.

4.3.1. Rancangan Antarmuka Aplikasi DroneMEQ

Perancangan pada aplikasi DroneMEQ tidak mencakup pembuatan antarmuka visual dari awal. Desain yang sudah ada dinilai telah memadai, kecuali pada halaman kelola pelanggaran sehingga fokus perancangan adalah pada perbaikan *bug* dan integrasi dengan *backend*.

1. Antarmuka Halaman Registrasi

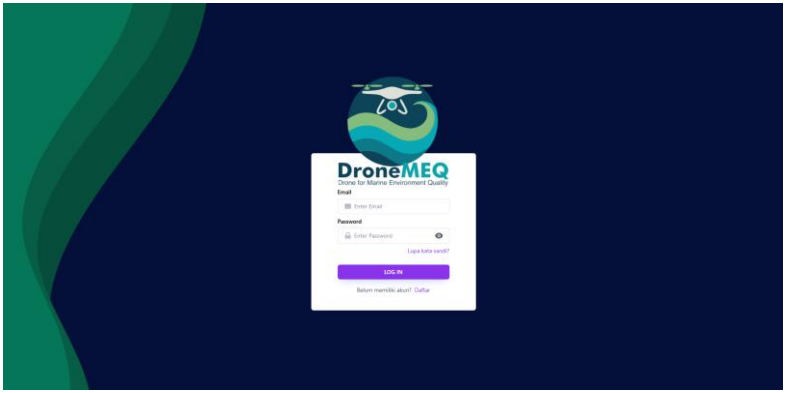
Pada antarmuka registrasi, penulis hanya melakukan perbaikan dari segi kode dimana kondisi awal aplikasi masih melakukan *hardcode* pada integrasi *backend* yang kemudian diubah oleh penulis menjadi dinamis. Berikut pada Gambar 4.4 diberikan tampilan halaman registrasi.



Gambar 4.4 Halaman Registrasi DroneMEQ

2. Antarmuka Halaman *Login*

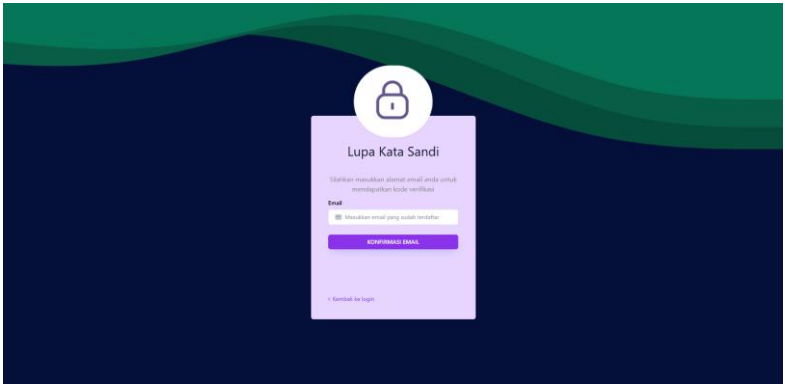
Pada antarmuka *login,* hanya dilakukan perubahan dari segi implementasi kode integrasi *backend* yang awalnya masih *hardcode* menjadi dinamis. Pada Gambar 4.5 ditampilkan gambar halaman *Login* DroneMEQ.



Gambar 4.5 Halaman *Login* DroneMEQ

3. Antarmuka Halaman Pemulihan Akun

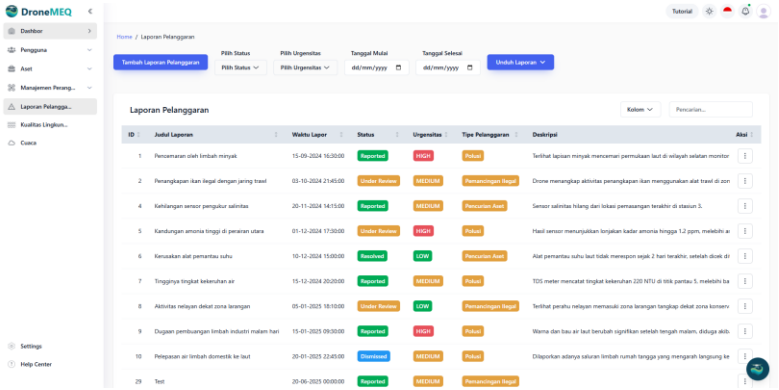
Pada antarmuka pemulihan akun, penulis tidak melakukan perubahan dari segi visual. Namun penulis melakukan perbaikan dari segi kode dimana kondisi awal aplikasi masih *hardcode* pada integrasi *backend* yang kemudian diubah oleh penulis menjadi dinamis. Hal ini dilakukan dengan tujuan membedakan *environment* pada *local* dan *server*. Berikut pada Gambar 4.6 diberikan tampilan halaman pemulihan akun.



Gambar 4.6 Halaman Pemulihan Akun DroneMEQ

4. Antarmuka Halaman Kelola Data Pelanggaran

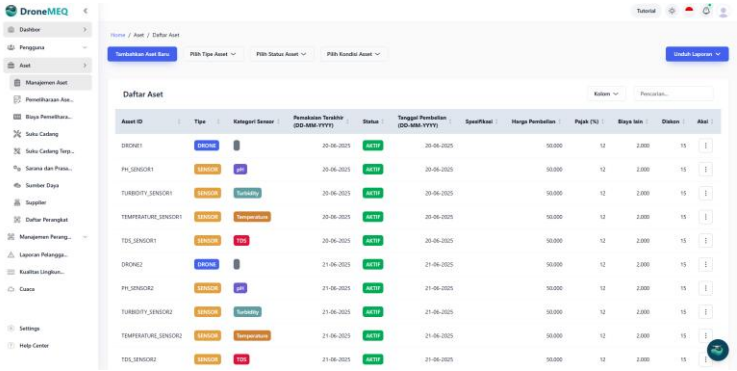
Pada antarmuka ini, penulis ditugaskan untuk merubah tampilan antarmuka yang awalnya berbentuk *card* menjadi bentuk tabel. Perubahan ini dilakukan dengan tujuan melihatkan sebanyak mungkin informasi pelanggaran dalam satu halaman dibandingkan dengan *card* yang memuat lebih sedikit informasi. Selain perubahan visual, dilakukan juga perbaikan pada fungsionalitas yang ada dengan menambahkan fungsi filter agar pengguna dapat menyaring data secara spesifik. Pada Gambar 4.7, ditampilkan rancangan antarmuka yang baru.



Gambar 4.7 Rancangan Antarmuka Halaman Kelola Data Pelanggaran

5. Antarmuka Halaman Kelola Data Aset

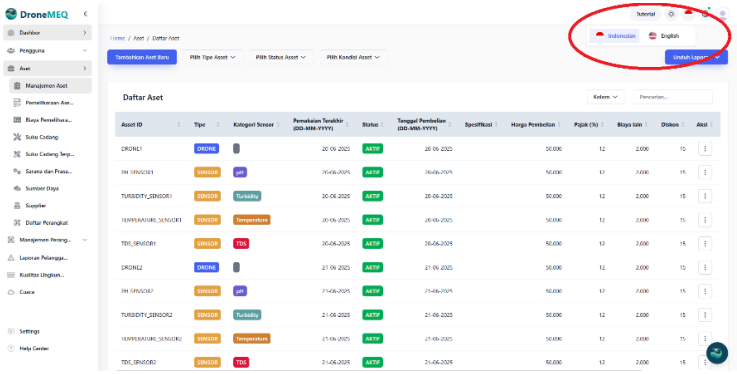
Pada antarmuka kelola data aset, penulis tidak melakukan perubahan secara visual. Namun penulis melakukan perbaikan pada *bug* yang membuat aplikasi tidak terhubung dengan *backend.* Berikut pada Gambar 4.8 ditampilkan gambar rancangan halaman kelola data aset yang akan dijadikan objek perancangan integrasi.



Gambar 4.8 Rancangan Halaman Kelola Data Aset

6. Antarmuka Komponen Pemilihan Bahasa

Pada aplikasi, antarmuka komponen pemilihan bahasa sudah terimplementasi pada aplikasi. Namun, hanya sebagai tampilan saja dan belum bisa digunakan. Oleh karena itu, penulis melakukan impementasi pada komponen pemilihan bahasa. Akan tetapi, tidak ada perubahan visual yang terjadi. Pada Gambar 4.9 yang dilingkari berwarna merah ditampilkan tampilan rancangan komponen pemilihan bahasa.



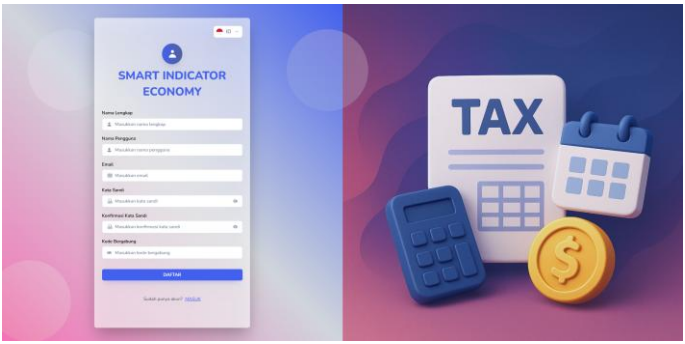
Gambar 4.9 Rancangan Antarmuka Komponen Pemilihan Bahasa

4.3.2. Rancangan Antarmuka Aplikasi Portal Smart Indicator Economy

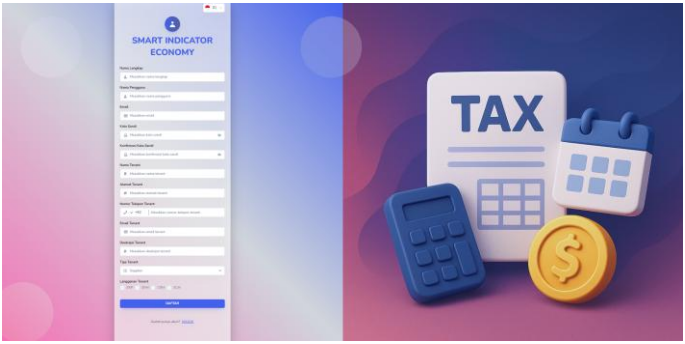
Karena aplikasi Portal Smart Indicator Economy dibangun dari awal, seluruh antarmuka dirancang untuk menciptakan pengalaman pengguna yang modern dan mudah digunakan.

1. Antarmuka Halaman Registrasi

Rancangan ini menggunakan formulir yang bersih dengan instruksi yang jelas untuk meminimalkan kebingungan pengguna saat mendaftar ke dalam sistem. Berikut tampilan rancangan antarmuka registrasi sebagai pengguna disajikan pada Gambar 4.10 dan registrasi sebagai *tenant* disajikan pada Gambar 4.11.



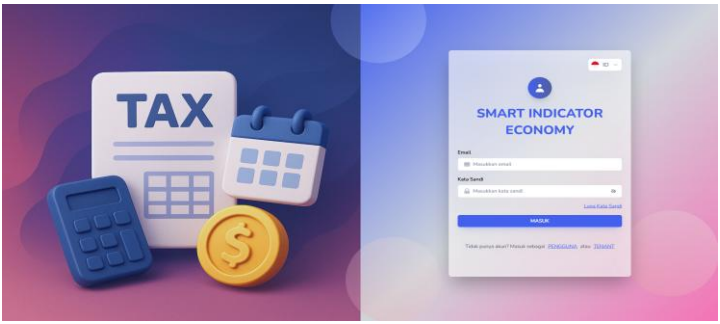
Gambar 4.10 Rancangan Antarmuka Registrasi sebagai Pengguna



Gambar 4.11 Rancangan Antarmuka Registrasi sebagai Tenant

2. Antarmuka Halaman *Login*

Desain untuk fitur *login* difokuskan pada alur yang sederhana. Rancangan ini menggunakan formulir yang bersih dengan instruksi yang jelas untuk meminimalkan kebingungan pengguna saat masuk ke dalam sistem. Kemudian untuk mendukung alur *Single Sign-On* antar aplikasi, *cookies* akan digunakan sebagai mekanisme penyimpanan sesi. Berikut pada Gambar 4.12 adalah tampilan rancangan antarmuka *login.*



Gambar 4.12 Rancangan Antarmuka Login

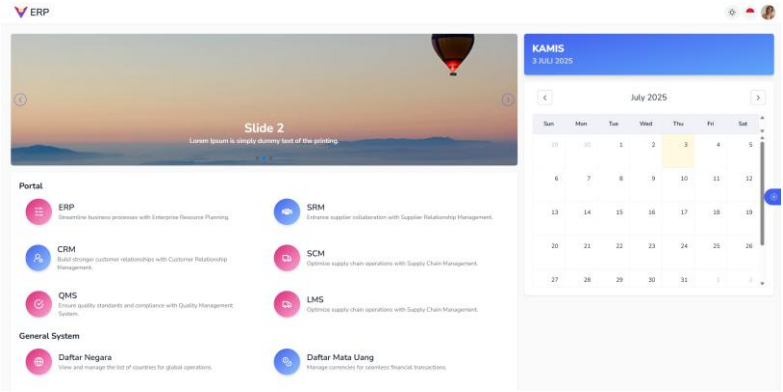
3. Antarmuka Halaman Pemulihan Akun

Rancangan ini menggunakan formulir yang bersih dengan instruksi yang jelas untuk meminimalkan kebingungan pengguna saat ingin melakukan pemulihan akun. Pada Gambar 4.13, diberikan tampilan rancangan halaman pemulihan akun.

Gambar 4.13 Tampilan Rancangan Antarmuka Pemulihan Akun

4. Antarmuka Halaman *Dashboard*

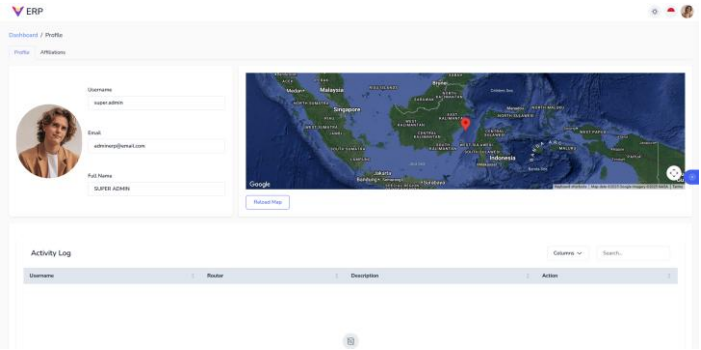
Halaman *Dashboard* dirancang sebagai pusat navigasi utama setelah pengguna berhasil login. Tata letaknya terdiri dari beberapa komponen kunci yaitu *sliding banner* untuk pengumuman, kalender interaktif, dan menu navigasi berbasis kartu untuk mengakses setiap modul yang tersedia, seperti yang ditunjukkan pada rancangan di Gambar 4.14.



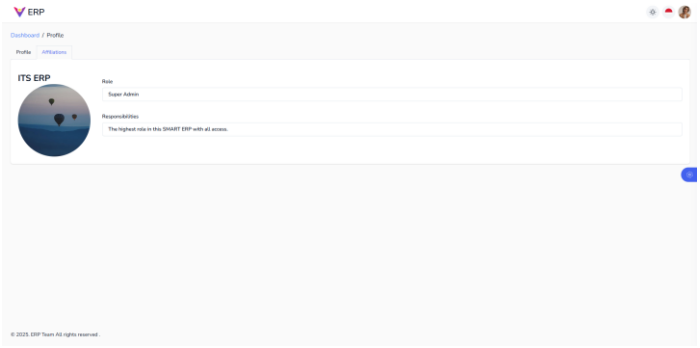
Gambar 4.14 Rancangan Antarmuka Dashboard

5. Antarmuka Halaman Kelola Profil

Pada halaman profil, penulis memecah halaman menjadi 2 *tab* berbeda. Pada *tab* profil penulis memecah menjadi 3 *card* berbeda yaitu *profile card* yang menunjukkan identitas pribadi, *location card* yang menunjukkan lokasi pengguna, serta *activity log card* yang menunjukkan riwayat kegiatan pengguna. Kemudian pada *tab affiliations* memiliki tujuan untuk menunjukkan tenant yang berafiiliasi dengan pengguna. Pada Gambar 4.15 menunjukkan rancangan *tab profile* dan pada Gambar 4.16 untuk *tab affiliations.*



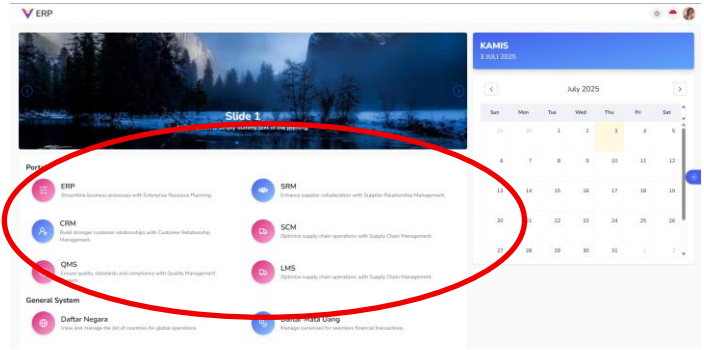
Gambar 4.15 Rancangan Halaman Profil *Tab Profile*



Gambar 4.16 Rancangan Halaman Profil *Tab Affiliations*

6. Antarmuka Akses Aplikasi Eksternal

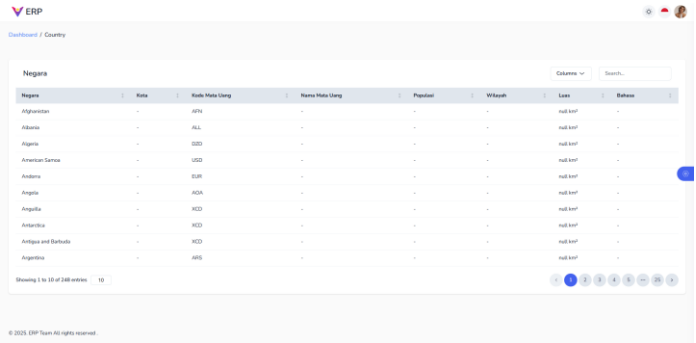
Rancangan antarmuka untuk mengakses aplikasi eksternal diwujudkan dalam bentuk menu navigasi berbasis kartu terletak di halaman Dashboard. Setiap kartu secara visual merepresentasikan satu aplikasi turunan, seperti ERP, CRM, atau SCM, lengkap dengan nama dan deskripsi singkatnya. Desain ini bertujuan untuk menyediakan titik akses yang terpusat dan intuitif, memungkinkan pengguna berpindah antar aplikasi dalam ekosistem dengan mudah dan mendukung alur *Single Sign-On* (SSO). Rancangan antarmuka akses aplikasi eksternal ditunjukkan pada Gambar 4.17.



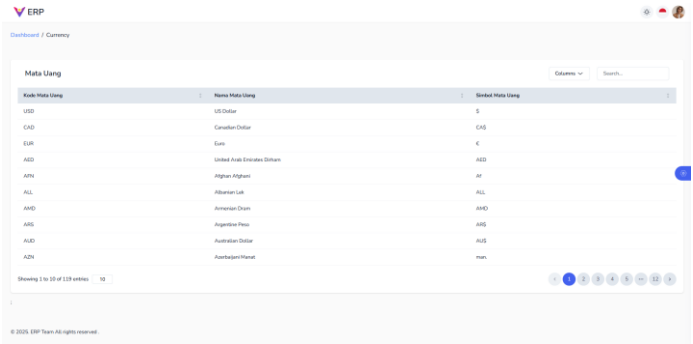
Gambar 4.17 Rancangan Antarmuka Akses Aplikasi Eksternal

7. Antarmuka Halaman Informasi Umum

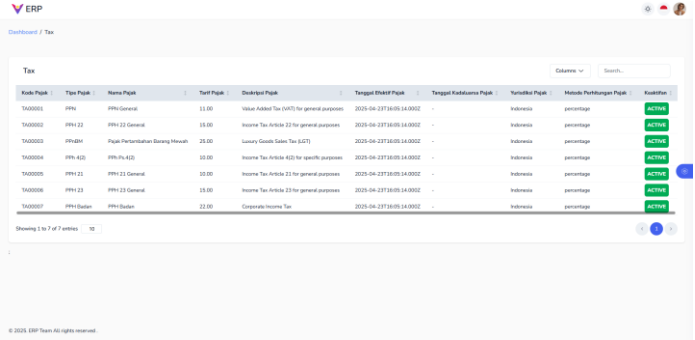
Antarmuka untuk menyajikan informasi umum sistem dirancang untuk memberikan akses read-only kepada pengguna terhadap data referensi. Desain utamanya menggunakan pola tabel data yang bersih dan terstruktur. Setiap tabel dilengkapi dengan fungsionalitas esensial seperti pencarian dan untuk memudahkan pengguna menelusuri data dalam jumlah besar. Informasi yang disajikan mencakup data master seperti Daftar Negara, Mata Uang, Pajak, dan Wilayah Indonesia. Pada Gambar 4.18 sampai dengan Gambar 4.21, diberikan gambar untuk setiap rancangan antarmuka informasi umum.



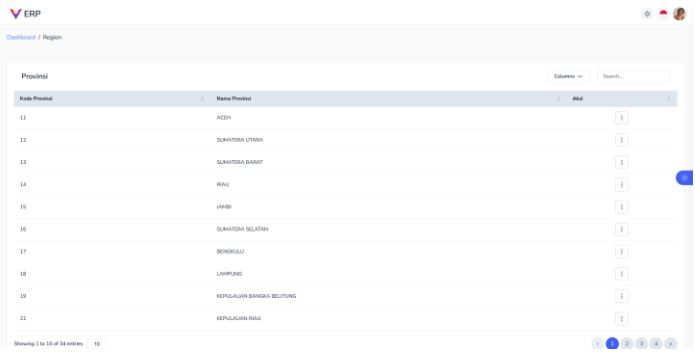
Gambar 4.18 Rancangan Halaman Informasi Umum Daftar Negara



Gambar 4.19 Rancangan Halaman Informasi Umum Mata Uang



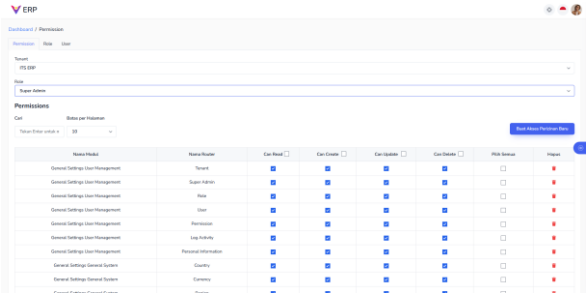
Gambar 4.20 Rancangan Halaman Informasi Umum Pajak



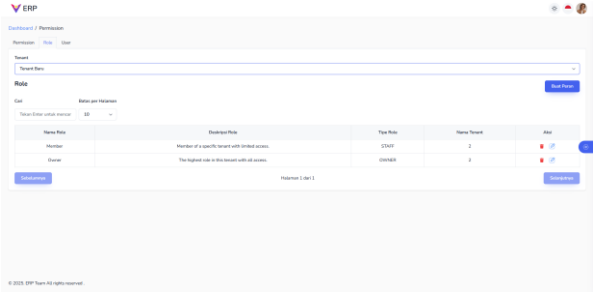
Gambar 4.21 Rancangan Halaman Informasi Umum Wilayah Indonesia

8. Antarmuka Halaman Kelola Hak Akses Portal

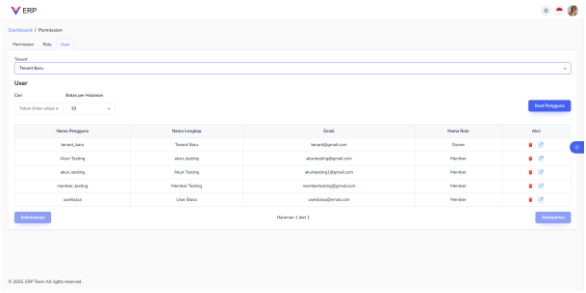
Untuk fungsionalitas pengelolaan hak akses yang diakses oleh Super Admin, antarmuka dirancang sebagai sebuah modul administrasi yang aman. Halaman ini dibagi menjadi 3 *tab* utama yaitu *tab* hak akses (*permission), tab* peran (*role)* dan *tab* pengguna (*user).* Inti dari desain setiap *tab* adalah sebuah tabel data interaktif yang menampilkan daftar seluruh hak akses atau peran atau pengguna yang terdaftar di sistem. Rancangan ini mendukung operasi CRUD secara penuh, yang diwujudkan melalui tombol-tombol aksi yang jelas. Untuk menunjang skalabilitas, desain ini juga mencakup fitur pencarian dan filter untuk memudahkan admin menemukan dan mengelola akun pengguna secara efisien. Berikut ditampilkan rancangan setiap *tab* pada halaman pada Gambar 4.22 sampai dengan 4.24.



Gambar 4.22 Rancangan Halaman Kelola Pengguna Portal Tab Profile



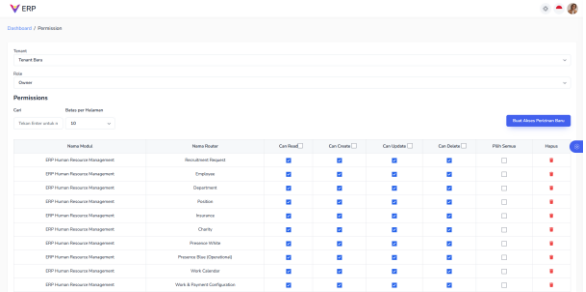
Gambar 4.23 Rancangan Halaman Kelola Pengguna Portal *Tab Role*



Gambar 4.24 Rancangan Halaman Kelola Pengguna Portal *Tab User*

9. Antarmuka Halaman Kelola Hak Akses Aplikasi

Antarmuka pengelolaan hak akses (*permission management*) dirancang untuk menjadi fondasi dari sistem *Role-Based Access Control* (RBAC). Desainnya berpusat pada sebuah tabel yang menyajikan daftar semua hak akses yang telah didefinisikan untuk aplikasi ekosistem Smart Indicator Economy. Setiap baris pada tabel dirancang untuk menampilkan detail hak akses dan menyediakan aksi untuk mengubah. Selain itu, terdapat fitur untuk membuat hak akses baru. Salah satu elemen kunci dalam rancangan ini adalah adanya filter berdasarkan *tenant* dan peran (*role)*, yang memastikan bahwa pengelolaan hak akses bersifat kontekstual dan aman untuk setiap organisasi. Berikut tampilan rancangan halaman kelola hak akses aplikasi pada Gambar 4.25.



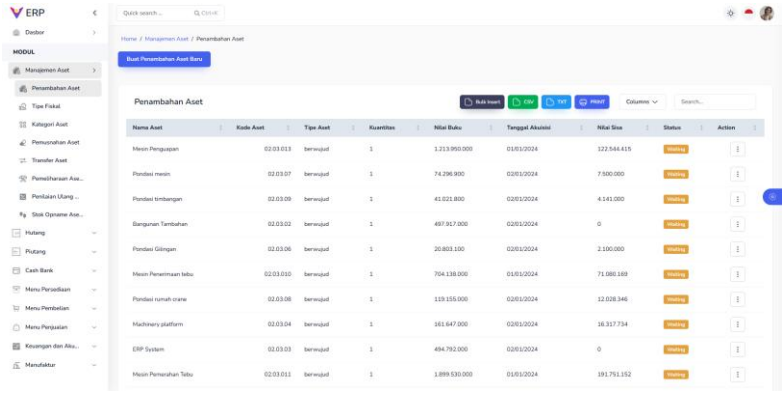
Gambar 4.25 Rancangan Halaman Hak Akses Aplikasi

4.3.3. Rancangan Antarmuka Aplikasi *Enterprise Resource Planning* (ERP)

Berbeda dengan Portal dan DroneMEQ, perancangan pada aplikasi ERP tidak mencakup pembuatan antarmuka visual dari awal kecuali pada halaman verifikasi autentikasi. Desain yang sudah ada dinilai telah memadai, sehingga fokus perancangan adalah pada aspek integrasi dan otorisasi data.

1. Antarmuka Kelola Data Aset

Rancangan pada bagian ini terpusat pada tabel yang berisi data aset dengan setiap komponen interaktif (seperti tombol tambah, ubah dan hapus) pada modul aset menyesuaikan hak akses yang telah diatur di Portal. Tujuannya adalah antarmuka dapat secara dinamis menampilkan tombol aksi berdasarkan hak akses peran pengguna yang sedang login, tanpa mengubah tata letak visual halaman. Gambar 4.26 menunjukkan contoh antarmuka yang menjadi objek dari perancangan integrasi ini.

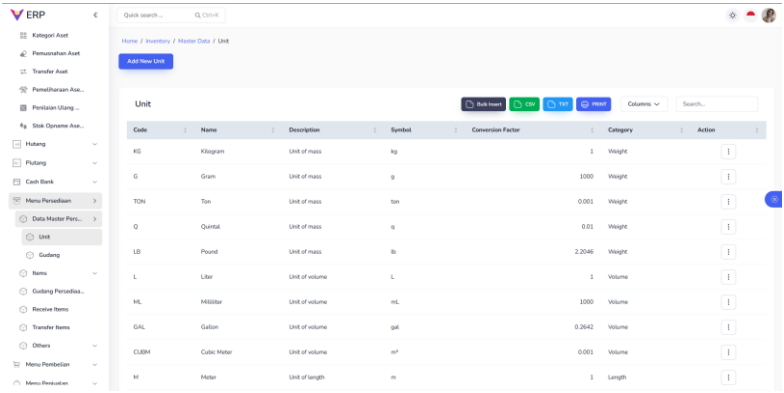


Gambar 4.26 Rancangan Antarmuka Kelola Data Aset

2. Antarmuka Kelola Data Inventaris

Pada Halaman ini, rancangan terpusat pada tabel yang berisi data inventaris dengan setiap komponen interaktif (seperti tombol

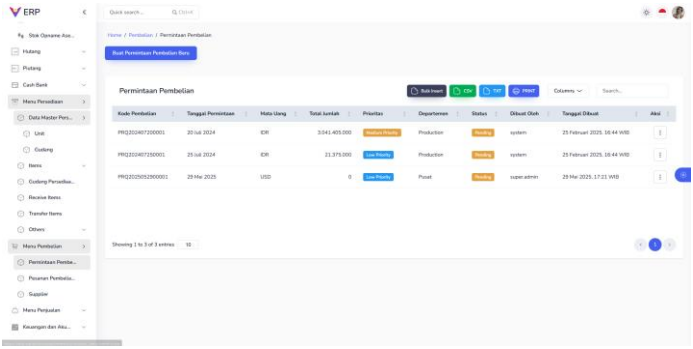
tambah, ubah dan hapus) pada modul inventaris menyesuaikan hak akses yang telah diatur di Portal. Tujuannya adalah agar antarmuka dapat secara dinamis menampilkan tombol aksi berdasarkan hak akses peran pengguna yang sedang login, tanpa mengubah tata letak visual halaman. Gambar 4.27 menunjukkan contoh antarmuka yang menjadi objek dari perancangan integrasi ini.



Gambar 4.27 Rancangan Antarmuka Kelola Data Inventaris

3. Antarmuka Kelola Pembelian

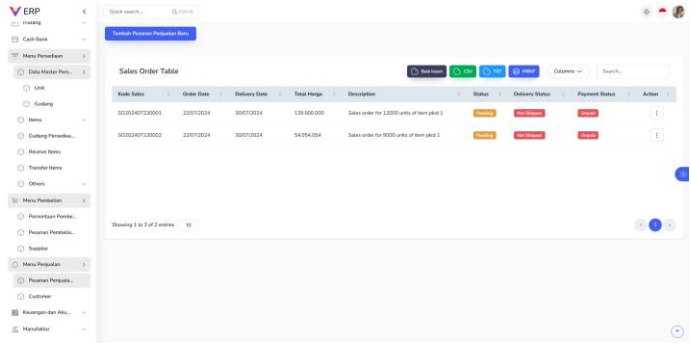
Rancangan pada bagian ini terpusat pada tabel yang berisi data pembelian dengan setiap komponen interaktif (seperti tombol tambah, ubah dan hapus) pada modul pembelian menyesuaikan hak akses yang telah diatur di Portal. Tujuannya adalah agar antarmuka dapat secara dinamis menampilkan tombol aksi berdasarkan hak akses peran pengguna yang sedang login, tanpa mengubah tata letak visual halaman. Gambar 4.28 menunjukkan contoh antarmuka yang menjadi objek dari perancangan integrasi ini.



Gambar 4.28 Rancangan Antarmuka Kelola Data Pembelian

4. Antarmuka Kelola Data Penjualan

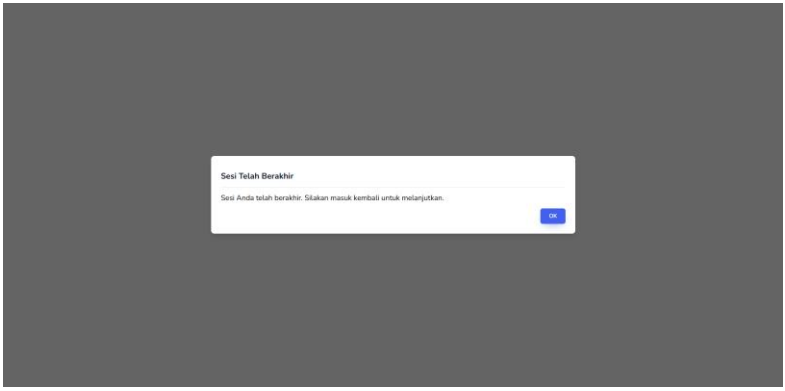
Pada bagian ini, rancangan terpusat pada tabel yang berisi data penjualan dengan setiap komponen interaktif (seperti tombol tambah, ubah dan hapus) pada modul penjualan menyesuaikan hak akses yang telah diatur di Portal. Tujuannya adalah agar antarmuka dapat secara dinamis menampilkan tombol aksi berdasarkan hak akses peran pengguna yang sedang login, tanpa mengubah tata letak visual halaman. Gambar 4.29 menunjukkan contoh antarmuka yang menjadi objek dari perancangan integrasi ini.



Gambar 4.29 Rancangan Antarmuka Kelola Data Penjualan

5. Antarmuka Verifikasi Autentikasi

Pada halaman ini memiliki tujuan untuk memberikan peringatan kepada pengguna yang belum terautentikasi. Halaman akan menampilkan modal yang memberikan pesan bahwa pengguna belum terautentikasi dan harus melakukan *login* terlebih dahulu. Berikut pada Gambar 4.30 diberikan rancangan tampilannya.



Gambar 4.30 Rancangan Antarmuka Halaman Verifikasi Autentikasi

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari hasil pada tahap analisa dan perancangan terhadap aplikasi DroneMEQ, aplikasi ERP, dan aplikasi Portal Smart Indicator Economy. Rincian Implementasi dituliskan pada Tabel 5.1 yang dibagi menjadi tiga bagian yakni implementasi aplikasi DroneMEQ, implementasi aplikasi ERP, dan implementasi Portal Smart Indicator Economy.

Tabel 5.1 Rincian Implementasi

Scope	Aktivitas
DroneMEQ	• Melakukan Registrasi • Melakukan *Login* • Melakukan Pemulihan Akun • Mengelola Data Pelanggaran • Mengelola Data Aset • Memilih Bahasa
Portal Smart Indicator Economy	• Melakukan Registrasi • Melakukan *Login* • Melakukan Pemulihan Akun • Melihat Dashboard • Mengelola Profil Pribadi • Mengakses Aplikasi Eksternal • Melihat Informasi Umum • Mengelola Hak Akses Portal • Mengelola Hak Akses Aplikasi
Enterprise Resource Planning (ERP)	• Mengelola Data Aset • Mengelola Data Inventaris • Mengelola Data Pembelian • Mengelola Data Penjualan • Melakukan Verifikasi Autentikasi

5.1. Implementasi Aplikasi DroneMEQ

5.1.1. Melakukan Registrasi

Fitur registrasi memungkinkan pengguna baru untuk membuat akun. Implementasinya dimulai dengan formulir pendaftaran yang divalidasi di sisi klien untuk memastikan semua data yang diperlukan telah terisi dengan benar. Setelah pengguna mengirimkan formulir, data akan dikirim ke *endpoint API POST* /api/auth/register. Sistem akan menangani respons dari *API*. Jika pendaftaran berhasil, pengguna akan diarahkan ke halaman login. Jika terjadi galat, seperti email yang sudah terdaftar, pesan kesalahan yang sesuai akan muncul. Berikut ditampilkan kode implementasi pada Gambar 5.1

```
1.  const submitForm = async (e:
    React.FormEvent<HTMLFormElement>) => {
2.      e.preventDefault();
3.      setIsLoading(true);
4.
5.      // Simple validation to check if passwords
    match
6.      if (formData.password !==
    formData.confirmPassword) {
7.        Swal.fire('Error!', 'Passwords do not
    match', 'error');
8.        return;
9.      }
10.
11.     try {
12.       const response = await Register({
13.         nik: formData.nik,
14.         username: formData.username,
15.         email: formData.email,
16.         password: formData.password,
17.         confirm_pwd: formData.confirmPassword,
18.         first_name: formData.firstName,
19.         last_name: formData.lastName,
20.         gender: formData.gender,
21.         role: formData.role,
22.         office: formData.office,
```

```
23.       });
24.
25.       // Accessing the token from the response
26.       const token = response.data.token;
27.
28.       Cookies.set('regisToken', token);
29.
30.       // Handle success response
31.       Swal.fire('Success!', 'Registration
    successful!', 'success');
32.       router.push('/auth/verify-account');
33.     } catch (error) {
34.       setIsLoading(false);
35.
36.       if (axios.isAxiosError(error) &&
    error?.response?.data) {
37.         const errorMessage =
38.           error.response.data.error ?? 'An
    unexpected error occurred';
39.
40.         Swal.fire('Error!', errorMessage,
    'error');
41.       } else {
42.         Swal.fire('Error!', 'An unexpected error
    occurred', 'error');
43.       }
44.     }
45.   };
```

Gambar 5.1 Potongan Kode Implementasi Registrasi

5.1.2. Melakukan *Login*

Fitur *login* dirancang untuk mengautentikasi pengguna menggunakan kredensial yang telah terdaftar. Proses implementasinya dimulai pengguna memasukkan email dan kata sandi pada formulir yang tersedia. Ketika tombol "Masuk" ditekan, sebuah fungsi akan dipanggil untuk mengirimkan data tersebut ke *endpoint API POST* /api/auth/login. Jika respons dari *API* berhasil, token autentikasi yang mengandung kredensial pengguna akan disimpan pada *cookies* dan pengguna akan diarahkan ke halaman

dashboard. *Cookies* dirancang agar menggunakan *domain* ”*.erplabiim.com”* karena ekosistem ini dibangun dari *domain* utama tersebut. Hal ini dilakukan agar seluruh aplikasi dengan *domain* tersebut dapat mengakses kredensial yang sama. Jika gagal, sebuah notifikasi akan ditampilkan kepada pengguna. Pada Gambar 5.2 ditunjukkan kode implementasi *login.*

```
1.  // sambungan kode…
2.    try {
3.        const response = await Login({
4.          email_or_username: email,
5.          password: password,
6.        });
7.
8.        const token = response.data.token;
9.        Cookies.set('authToken', token);
10.
11.       router.push('/dashboard');
12.     } catch (error: unknown) {
13.       setIsLoading(false); // Reset loading state
14.
15.       if (axios.isAxiosError(error) &&
    error?.response?.data) {
16.         const errorMessage =
    error.response.data.error;
17.
18.         // Handle specific error messages
19.         if (errorMessage.includes('not
    registered')) {
20.           setError('email', { message: 'Email is
    not registered.' });
21.         } else if (errorMessage.includes('wrong
    password')) {
22.           setError('password', { message:
    'Incorrect password.' });
23.         } else if (errorMessage.includes('blocked
    account')) {
24.           Swal.fire('Error!', 'Your account is
    blocked.', 'error');
25.         } else {
```

```
26.           Swal.fire('Error!', 'An unexpected
    error occurred.', 'error');
27.         }
28.       } else {
29.         Swal.fire('Error!', 'Server connection
    problem or timeout.', 'error');
30.       }
31.     }
32.   };
```

Gambar 5.2 Potongan Kode Implementasi *Login*

5.1.3. Melakukan Pemulihan Akun

Implementasi fitur ini melibatkan beberapa langkah untuk memastikan keamanan. Pertama, pengguna memasukkan alamat email mereka, yang kemudian dikirimkan ke *endpoint API POST* /api/auth/send-reset-password untuk memicu pengiriman tautan reset ke email tersebut. Pengguna kemudian akan membuka tautan tersebut, yang mengarah ke halaman khusus untuk memasukkan kata sandi baru. Setelah kata sandi baru dikonfirmasi, data akan dikirimkan ke *endpoint API POST* /api/auth/reset-password bersama dengan *token reset* yang unik untuk melakukan pembaruan. Pada Gambar 5.3 diberikan potongan kode implementasi mengirim email untuk mendapatkan tautan, dan Gambar 5.4 adalah potongan kode untuk pembaruan password.

```
1.  //lanjutan kode..
2.      try {
3.        await SendResetPassword({
4.          email: email,
5.        });
6.
7.        // Menampilkan pesan sukses
8.        Swal.fire(
9.          'Success!',
10.         'Reset password link has been sent to
    your email.',
11.         'success',
12.       );
13.       setIsLoading(false);
```

```
14.     } catch (error) {
15.       setIsLoading(false);
16.
17.       if (axios.isAxiosError(error) &&
    error?.response?.data?.message) {
18.         const errorMessage =
    error.response.data.message;
19.
20.         if (errorMessage.includes('not
    registered')) {
21.           setError('email', { message: 'Email is
    not registered.' });
22.         } else {
23.           Swal.fire('Error!', 'An unexpected
    error occurred.', 'error');
24.         }
25.       } else {
26.         Swal.fire('Error!', 'Server connection
    problem or timeout.', 'error');
27.       }
28.     }
29.   };
30.
```

Gambar 5.3 Potongan Kode Mengirim Email untuk Mendapatkan Tautan Pemulihan Akun

```
1.  //lanjutan kode..
2.      try {
3.        const response = await resetPassword({
4.          token: token,
5.          new_password: new_password,
6.          confirm_pwd: confirm_pwd,
7.        });
8.
9.        if (response.status) {
10.         handleSuccess(router);
11.       } else {
12.         handleError(null, response);
13.       }
14.     } catch (error) {
15.       handleError(error);
16.     } finally {
17.       setIsLoading(false);
18.     }
19.   };
20.
```

Gambar 5.4 Potongan Kode Pembaruan Password

5.1.4. Mengelola Data Pelanggaran

Sesuai dengan perancangan di Bab 4 penulis melakukan implementasi untuk merubah tampilan menjadi tabel dan menambahkan filter. Penulis menggunakan komponen *RenderDataTable* dengan *input* data dari hasil pemanggilan *endpoint API GET* /violations/filter dan menambahkan komponen *MultiValueDropdown* untuk mengimplementasi filter. Filter yang ditangkap dikirim ke *endpoint* yang sama dalam bentuk *parameter (*/violations/filter?${parameter}). Berikut tampilan potongan kodenya pada Gambar 5.5.

```
1.  // lanjutan kode..
2.
3.    const {
4.      data: violationFilter,
5.      isLoading,
6.      // error,
7.    } = useGetViolationsbyFilter(
```

```
      { ...filters, page, per_page: limit },
    );

  const updateFilters = (newFilters:
Partial<typeof filters>) => {
    setFilters(prev =>
({ ...prev, ...newFilters }));
    setPage(1); // Reset to first page when
filters change
  };

  // implementasi kode…


          <MultivalueDropdown
            options={ViolationStatusFiter}
            placeholder={t('choose') + ' ' +
t('status')}
            selectedValues={filters.status || []}
            onChange={values =>
              updateFilters({ status:
values.length ? values : undefined })
            }
          />
        </div>
 // implementasi kode..

      <div className='relative flex h-full flex-
col gap-5'>
        <RenderDataTable
          title={t('violations')}
          data={violationFilter?.data}
          columns={cols}
          isLoading={isLoading}
          detailPath='/violations'
          action='RD'
          refetch={() => {
            // Refetch data when needed
          }}
          deleteFunc={deleteViolations}
          pagination={violationFilter?.pagination
} // Pass pagination to ViolationTable
```

```
43.           setPage={handlePageChange} // Pass
    setPage to RenderDataTable
44.           setLimit={handlePageSizeChange}
45.         />
46.       </div>
47.     </div>
48.   );
49. };
50.
```

Gambar 5.5 Potongan Kode Implementasi Tampilan Data Pelanggaran dan *Filter* Pelanggaran

5.1.5. Mengelola Data Aset

Proses implementasi melibatkan penyesuaian pada kode pemanggilan API, memastikan *endpoint* yang dituju sudah benar. Perubahan *endpoint* cukup dilakukan dengan merubah *environment API* yang masih belum disesuaikan dengan *endpoint* terbaru dari *backend.* Berikut potongan kode ditampilkan pada Gambar 5.6.

```
1.  # lanjutan kode..
2.
3.  NEXT_PUBLIC_API_DEV_FIXED_ASSET=https://dronemeq-
    api.erplabiim.com/as/api/
4.
5.  #lanjutan kode...
```

Gambar 5.6 Potongan Kode Perbaikan Kelola Data Aset

5.1.6. Memilih Bahasa

Implementasi dilakukan dengan menggunakan *library* i18next.Sebuah fungsi ditambahkan pada komponen *dropdown* tersebut. Ketika pengguna memilih bahasa (Indonesia atau Inggris), fungsi ini akan mengubah status bahasa pada aplikasi dan secara dinamis memuat file terjemahan yang sesuai. Hasilnya, seluruh teks pada antarmuka aplikasi kini dapat berubah secara dinamis sesuai dengan bahasa yang dipilih pengguna, tanpa perlu

memuat ulang halaman. Berikut potongan kode ditampilkan pada Gambar 5.7.

```
1.   <Dropdown
2.           offset={[0, 8]}
3.           placement={`${isRtl ? 'bottom-start' :
    'bottom-end'}`}
4.           btnClassName={getDropdownClasses()}
5.           button={
6.             i18n.language && (
7.               <Image
8.                 className='h-5 w-5 rounded-full
    object-cover'
9.                 src={`/assets/images/flags/${i18n.l
    anguage.toUpperCase()}.svg`}
10.                alt='flag'
11.                width={20}
12.                height={20}
13.              />
14.            )
15.          }
16.       >
17.          <ul className='text-dark dark:text-white-
    dark dark:text-white-light/90 grid w-[280px]
    grid-cols-2 gap-2 !px-2 font-semibold'>
18.            {themeConfig.languageList.map((item:
    any) => (
19.              <li key={item.code}>
20.                <button
21.                  type='button'
22.                  className={`hover:text-primary
    flex w-full ${
23.                    i18n.language === item.code
24.                      ? 'bg-primary/10 text-
    primary'
25.                      : ''
26.                  }`}
27.                  onClick={() => {
28.                    i18n.changeLanguage(item.code);
29.                    onLanguageChange(item.code);
30.                    window.location.reload();
31.                  }}
32.                >
```

```
33.                 <Image
34.                   src={`/assets/images/flags/${it
    em.code.toUpperCase()}.svg`}
35.                   alt='flag'
36.                   className='h-5 w-5 rounded-full
    object-cover'
37.                   width={20}
38.                   height={20}
39.                 />
40.                 <span className='ltr:ml-3 rtl:mr-
    3'>{item.name}</span>
41.               </button>
42.             </li>
43.           ))}
44.         </ul>
45.       </Dropdown>
```

Gambar 5.7 Potongan Kode Komponen Pemilihan Bahasa

5.2. Implementasi Aplikasi Portal Smart Indicator Economy

5.2.1. Melakukan Registrasi

Fitur registrasi memungkinkan pengguna baru untuk membuat akun sebagai pengguna atau sebagai *tenant*. Implementasinya dimulai dengan formulir pendaftaran yang divalidasi di sisi klien untuk memastikan semua data yang diperlukan telah terisi dengan benar. Setelah pengguna mengirimkan formulir, data akan dikirim ke *endpoint API POST* /api/auth/signup untuk registrasi sebagai pengguna dan /api/auth/tenant-signup. Sistem akan menangani respons dari *API*. Jika pendaftaran berhasil, pengguna akan diarahkan ke halaman selanjutnya. Jika terjadi galat, seperti email yang sudah terdaftar, pesan kesalahan yang sesuai akan muncul. Berikut ditampilkan kode implementasi registrasi sebagai pengguna untuk contoh pada Gambar 5.8

```
1.  const ComponentsAuthSignUpForm = () => {
2.    const { mutateAsync: SignUp } = useSignUp();
3.          const       [form,       setForm]       =
    useState(signUpInitialState);
4.              const           [passwordValidations,
    setPasswordValidations] = useState({
5.      minLength: false,
6.      hasUpperCase: false,
7.      hasNumber: false,
8.      hasSpecialChar: false,
9.    });
10.
```

```
11.   const handleOnChange = (value: string, key: string) => {
12.     setForm({ ...form, [key]: value });
13.
14.     if (key === 'password') {
15.       validatePassword(value);
16.     }
17.   };
18.
19.   const validatePassword = (password: string) => {
20.     setPasswordValidations({
21.       minLength: password.length >= 8,
22.       hasUpperCase: /[A-Z]/.test(password),
23.       hasNumber: /\d/.test(password),
24.                                       hasSpecialChar: /[!@#$%^&*()]/.test(password),
25.     });
26.   };
27.
28.   const triggerSignUp = async () => {
29.     // Validasi form
30.     if (form.email === '' || form.password === '' || form.confirmPassword === '') {
31.         Swal.fire('Error!', 'Please fill all the fields', 'error');
32.       return;
33.     }
34.
35.     if (form.password !== form.confirmPassword) {
36.      Swal.fire('Error!', 'Passwords do not match', 'error');
37.       return;
38.     }
39.
40.     try {
41.       // SignUp API call
```

```
42.       await SignUp({
43.         fullname: form.fullname,
44.         username: form.username,
45.         email: form.email,
46.         password: form.password,
47.         confirmPassword: form.confirmPassword,
48.         joinCode: form.joinCode,
49.       });
50.
51.             Swal.fire('Success!',   'Registration
    successful', 'success')
52.         .then(() => {
53.           window.location.href = '/auth/login';
54.         });
55.     } catch (error) {
56.        Swal.fire('Error!', 'Registration failed',
    'error');
57.     }
58.   };
59.
60.   return (
61.     <form onSubmit={(e) => e.preventDefault()}>
62.       {/* ...potongan kode styling... */}
63.
64.       {/* Email Input */}
65.       <input
66.         type='email'
67.         placeholder='Enter email'
68.                                 onChange={e       =>
    handleOnChange(e.target.value, 'email')}
69.       />
70.
71.       {/* Password Input dengan Validasi */}
72.       <input
73.         type='password'
74.         placeholder='Enter password'
```

```
75.                                 onChange={e       =>
    handleOnChange(e.target.value, 'password')}
76.         onFocus={() => setIsPasswordFocused(true)}
77.       />
78.
79.       {/* Password Validation Display */}
80.       {isPasswordFocused && (
81.         <div className='password-validation'>
82.           <ul>
83.                                                 <li
    className={passwordValidations.minLength        ?
    'valid' : 'invalid'}>
84.               Minimal 8 karakter
85.             </li>
86.                                                 <li
    className={passwordValidations.hasUpperCase     ?
    'valid' : 'invalid'}>
87.               Huruf kapital
88.             </li>
89.                                                 <li
    className={passwordValidations.hasNumber        ?
    'valid' : 'invalid'}>
90.               Angka
91.             </li>
92.                                                 <li
    className={passwordValidations.hasSpecialChar   ?
    'valid' : 'invalid'}>
93.               Karakter spesial
94.             </li>
95.           </ul>
96.         </div>
97.       )}
98.
99.       {/* Confirm Password */}
100.             <input
101.               type='password'
102.               placeholder='Confirm password'
```

```
103.                                onChange={e     =>
    handleOnChange(e.target.value,
    'confirmPassword')}
104.             />
105.
106.                {/*  ...form  lainnya  (fullname,
    username, joinCode)... */}
107.
108.                                              <button
    onClick={triggerSignUp}>Register</button>
109.           </form>
110.         );
111.       };
112.
```

Gambar 5.8 Potongan Kode Implementasi Registrasi sebagai Pengguna

5.2.2. Melakukan *Login*

Fitur *login* dirancang untuk mengautentikasi pengguna menggunakan kredensial yang telah terdaftar. Proses implementasinya dimulai pengguna memasukkan email dan kata sandi pada formulir yang tersedia. Ketika tombol "Masuk" ditekan, sebuah fungsi akan dipanggil untuk mengirimkan data tersebut ke *endpoint API POST* /api/auth/login. Jika respons dari *API* berhasil, token autentikasi akan disimpan pada *cookies* dan pengguna akan diarahkan ke halaman dashboard. Jika gagal, sebuah notifikasi akan ditampilkan kepada pengguna. Pada Gambar 5.9 ditunjukkan kode implementasi *login.*

```
1.  const ComponentsAuthLoginForm = () => {
2.    const { mutateAsync: logIn } = useLogIn();
3.    const dispatch = useDispatch();
4.          const       [form,       setForm]       =
    useState(logInInitialState);
5.      const   [showPassword,   setShowPassword]   =
    useState(false);
```

```
6.
7.    const handleOnChange = (value: string, key: string) => {
8.      setForm({ ...form, [key]: value });
9.    };
10.
11.   const triggerLogIn = async () => {
12.     // Validasi form
13.     if (form.email === '' || form.password === '') {
14.       Swal.fire('Error!', 'Please fill all the fields', 'error');
15.       return;
16.     }
17.
18.     try {
19.       // Proses login
20.       const logInRes = await logIn({
21.         email: form.email,
22.         password: form.password,
23.       });
24.
25.       // Set cookie token
26.       const isLocalhost = window.location.hostname === 'localhost';
27.       const cookieOptions = isLocalhost
28.         ? {}
29.         : { domain: '.erplabiim.com' };
30.
31.                         Cookies.set('login_token', logInRes.data.token, cookieOptions);
32.
33.       // Buka modal untuk pilih tenant
34.       dispatch(setModalForm(true));
35.       setTenantData(logInRes.data.tenant);
36.
37.     } catch (error) {
38.           Swal.fire('Error!',   'Login   failed', 'error');
39.     }
40.   };
41.
```

```
  return (
    <form onSubmit={(e) => e.preventDefault()}>
      {/* Email Input */}
      <input
        type='email'
        placeholder='Enter email'
                                onChange={e       =>
handleOnChange(e.target.value, 'email')}
      />

      {/* Password Input */}
      <input
        type={showPassword ? 'text' : 'password'}
        placeholder='Enter password'
                                onChange={e       =>
handleOnChange(e.target.value, 'password')}
      />

      {/* Login Button */}
                          <button       type='submit'
onClick={triggerLogIn}>
        Login
      </button>
    </form>
  );
};


```

Gambar 5.9 Potongan Kode Melakukan Login

5.2.3. Melakukan Pemulihan Akun

Implementasi fitur ini melibatkan beberapa langkah untuk memastikan keamanan. Pertama, pengguna memasukkan alamat email mereka, yang kemudian dikirimkan ke *endpoint API POST* /api/auth/send-reset-password untuk memicu pengiriman tautan reset ke email tersebut. Pengguna kemudian akan membuka tautan tersebut, yang mengarah ke halaman khusus untuk memasukkan kata sandi baru. Setelah kata sandi baru dikonfirmasi, data akan dikirimkan ke *endpoint API POST*

/api/auth/reset-password bersama dengan *token reset* yang unik untuk melakukan pembaruan. Pada Gambar 5.10 diberikan potongan kode implementasi mengirim email untuk mendapatkan tautan, dan Gambar 5.11 adalah potongan kode untuk pembaruan password.

```
1.  const ForgotPasswordComponent = () => {
2.    const { mutateAsync: SendEmail } =
    useSendLinkResetPass();
3.    const [form, setForm] = useState({ email:
    '' });
4.
5.    const handleOnChange = (value: string, key:
    string) => {
6.      setForm({ ...form, [key]: value });
7.    };
8.
9.    const triggerSend = async () => {
10.     // Validasi email
11.     if (form.email === '') {
12.       Swal.fire('Error!', 'Please fill email
    field', 'error');
13.       return;
14.     }
15.
16.     try {
17.       // Send reset link API call
18.       await SendEmail({
19.         email: form.email,
20.       });
21.
22.       // Success notification
23.       Swal.fire({
24.         title: 'Success',
25.         text: 'Password reset link sent to your
    email',
26.         icon: 'success',
27.       }).then(() => {
28.         window.location.href = '/auth/login';
29.       });
30.
31.     } catch (error) {
```

```
32.       Swal.fire('Error!', 'Failed to send reset
    link', 'error');
33.     }
34.   };
35.
36.   return (
37.     <form onSubmit={(e) => e.preventDefault()}>
38.       {/* ...potongan kode styling... */}
39.
40.       <h1>Forgot Password</h1>
41.
42.       {/* Email Input */}
43.       <input
44.         type='email'
45.         placeholder='Enter your email'
46.         onChange={e =>
    handleOnChange(e.target.value, 'email')}
47.       />
48.
49.       <button onClick={triggerSend}>
50.         Send Reset Link
51.       </button>
52.
53.       {/* ...potongan kode... */}
54.     </form>
55.   );
56. };
57.
```

Gambar 5.10 Potongan Kode Pengiriman Email

```
1.  const UpdatePasswordComponent:
    React.FC<UpdatePasswordComponentProps> = ({
2.    token,
3.  }) => {
4.    const { mutateAsync: resetPassword } =
    useResetPassword();
5.    const [form, setForm] = useState({
6.      password: '',
7.      confirmPassword: '',
8.      token: token,
9.    });
10.
```

```
  const handleOnChange = (value: string, key: string) => {
    setForm({ ...form, [key]: value });
  };

  const triggerSend = async () => {
    // Validasi form
    if (form.password === '' || form.confirmPassword === '') {
      Swal.fire('Error!', 'Please fill all the fields', 'error');
      return;
    }

    try {
      // Reset password API call
      await resetPassword({
        password: form.password,
        confirmPassword: form.confirmPassword,
        token: token || '',
      });

      Swal.fire({
        title: 'Success',
        text: 'Password updated successfully',
        icon: 'success',
      }).then(() => {
        window.location.href = '/auth/login';
      });

    } catch (error) {
      Swal.fire('Error!', 'Failed to update password', 'error');
    }
  };

  return (
    <form onSubmit={(e) => e.preventDefault()}>
      {/* ...potongan kode styling... */}

      <h1>Reset Password</h1>

```

```
49.       {/* Password Input */}
50.       <input
51.         type='password'
52.         placeholder='Enter new password'
53.         onChange={e =>
    handleOnChange(e.target.value, 'password')}
54.       />
55.
56.       {/* Confirm Password */}
57.       <input
58.         type='password'
59.         placeholder='Confirm new password'
60.         onChange={e =>
    handleOnChange(e.target.value, 'confirmPassword')}
61.       />
62.
63.       <button onClick={triggerSend}>
64.         Update Password
65.       </button>
66.
67.       {/* ...potongan kode... */}
68.     </form>
69.   );
70. };
71.
72.
```

Gambar 5.11 Potongan Kode Pembaruan Kata Sandi

5.2.4. Melihat Dashboard

Implementasi dashboard dilakukan dengan membuat satu komponen utama yaitu *ComponentsDashboardPortal.* Komponen ini memiliki layout grid yang terbagi menjadi 2 kolom utama. Kolom pertama memiliki lebar 2/3 layar yang diisi dengan *Carousel* dan *menu* utama. Kemudian kolom kedua memiliki lebar sisanya (1/3) yang diisi dengan kalender. Pada menu utama portal , data diambil dari *file utils helper* dan *direnderer* menggunakan fungsi *renderPortalSection*() untuk menghindari duplikasi kode. Komponen memiliki berbagai fitur utama berupa:

1. ***Dynamic Menu Rendering*** - *Render* berbagai kategori menu dari data array.
2. ***Responsive Grid Layout*** - *Grid* 3 kolom dengan *sidebar calendar.*
3. ***Interactive Cards*** - *Hover effects* dan *click handler* untuk navigasi.
4. **Conditional Rendering** - *Filter* menu berdasarkan *availability.*
5. ***Icon Support*** - *Support* untuk *string image* URL atau *React component icons.*

Berikut diberikan potongan kode untuk dashboard pada Gambar 5.12.

```
1.  const ComponentsDashboardPortal = () => {
2.    const handlePanelClick = (link: string) => {
3.      window.location.href = link;
4.    };
5.
6.    const renderPortalSection = (title: string,
    items: any[]) => (
7.      <div className='flex w-full flex-col'>
8.        <h1 className='text-xl font-
    bold'>{title}</h1>
9.        <div className='grid w-full gap-4 xl:grid-
    cols-2'>
10.         {items.map((item, index) => {
11.           const Icon = item.image;
12.           return (
13.             <div
14.               key={index}
15.               onClick={() =>
    handlePanelClick(item.link)}
16.               className='flex cursor-pointer
    items-center gap-4 rounded-lg p-4 hover:scale-105
    hover:shadow-lg'
17.             >
18.               <div className='flex h-16 w-full
    flex-row items-center'>
19.                 {/* Icon/Image Display */}
```

```
                {typeof item.image === 'string' ? (
                  <img src={item.image} alt={item.name} className='h-16 w-16 rounded-full' />
                ) : (
                  <div className={`flex h-16 w-16 items-center justify-center rounded-full ${item.color}`}>
                    <Icon />
                  </div>
                )}

                {/* Content */}
                <div className='ml-4 flex flex-col'>
                  <h2 className='text-xl font-semibold'>{item.name}</h2>
                  <p className='text-gray-500'>{item.desc}</p>
                </div>
              </div>
            </div>
          );
        })}
      </div>
    </div>
  );

  return (
    <div className='grid h-full w-full gap-4 md:grid-cols-3'>
      {/* Main Content */}
      <div className='flex w-full flex-col items-center gap-4 md:col-span-2'>
        <ComponentsCarouselLoop />

        <div className='panel flex flex-col gap-4'>
          {renderPortalSection('Portal', portal)}
          {renderPortalSection('General System', generalSystem.filter(item => item.availability))}
```

```
50.           {renderPortalSection('Permission
    Management', portalManagement)}
51.           {renderPortalSection('ERP Management',
    erpManagement)}
52.         </div>
53.       </div>
54.
55.       {/* Sidebar */}
56.       <div className='col-span-1 flex w-full
    flex-col gap-4'>
57.         <TodayPanelComponent />
58.         <ViewOnlyCalendar />
59.       </div>
60.     </div>
61.   );
62. };
63.
```

Gambar 5.12 Potongan Kode Implementasi *Dashboard*

5.2.5. Mengelola Profil Pribadi

Penulis membuat komponen utama dengan struktur tab navigation menggunakan *@headlessui/react Tab component*. *Tab* pertama (*profile)* berisi data profil yang diambil dari *useProfileStore* yang disimpan menggunakan *library Zustand, activity log* yang diambil dari *useGetPersonalLog hook*, dan lokasi pengguna yang memanfaatkan *GoogleMap API.* Kemudian pada *tab* selanjutnya *(affiliations),* penulis membuat komponen hanya dengan menggunakan *form* untuk memperlihatkan informasi afiliasi.

```
1.  const ProfilePageComponent = () => {
2.    const { data, refetch, isLoading } =
    useGetPersonalLog();
3.    const { profile } = useProfileStore();
4.
5.    useEffect(() => {
6.      if (!data && isLoading) {
7.        refetch();
```

```
8.      }
9.    });
10.
11.   if (!data || !profile) {
12.     return <Skeleton className='h-full w-full'
    />;
13.   }
14.
15.   return (
16.     <div className='gap-4'>
17.       {/* ...potongan kode breadcrumb... */}
18.
19.       <Tab.Group>
20.         <Tab.List>
21.           <Tab>Profile</Tab>
22.           <Tab>Affiliations</Tab>
23.         </Tab.List>
24.
25.         <Tab.Panels>
26.           {/* Profile Tab */}
27.           <Tab.Panel>
28.             <div className='flex flex-col gap-4'>
29.               <form>
30.                 <div className='grid grid-cols-3
    gap-4'>
31.                   {/* Profile Info Panel */}
32.                   <div className='panel col-span-
    1'>
33.                     {/* Profile Image */}
34.                     <div className='relative h-
    [200px] w-[200px] rounded-full overflow-clip'>
35.                       <img
36.                         src={profile.image_url ||
    '/assets/images/user-profile.jpeg'}
37.                         alt='Profile'
38.                         className='object-cover
    w-full h-full'
39.                       />
40.                     </div>
41.
42.                     {/* Profile Fields */}
```

```
                    <div className='flex flex-col
    gap-4'>
                      <div>
                        <label>Username</label>
                        <input
                          type='text'
                          value={profile.username
    }
                          disabled
                          className='form-input'
                        />
                      </div>

                      <div>
                        <label>Email</label>
                        <input
                          type='text'
                          value={profile.email}
                          disabled
                          className='form-input'
                        />
                      </div>

                      <div>
                        <label>Full Name</label>
                        <input
                          type='text'
                          value={profile.fullname
    }
                          disabled
                          className='form-input'
                        />
                      </div>
                    </div>
                  </div>

                  {/* Google Map Panel */}
                  <div className='panel col-span-
    2'>
                    <GoogleMapPartComponent />
                  </div>
                </div>
```

```
81.                </form>
82.
83.                {/* Activity Log Panel */}
84.                <div className='panel'>
85.                  <ActivityLogTable
    data={data.rows} />
86.                </div>
87.              </div>
88.            </Tab.Panel>
89.
90.            {/* Affiliations Tab */}
91.            <Tab.Panel>
92.              <AffiliationComponent
93.                affiliations={profile.affiliations}
94.                responsibilities={profile.responsib
    ilities}
95.              />
96.            </Tab.Panel>
97.          </Tab.Panels>
98.        </Tab.Group>
99.      </div>
100.         );
101.       };
102.
```

Gambar 5.13 Potongan Kode Halaman Profil

5.2.6. Mengakses Aplikasi Eksternal

Pada bagian ini, penulis mengimplementasi dengan memanfaatkan *map.* Penulis mengambil data aplikasi dari *helper* kemudian melakukan *map* untuk menampilkan seluruh opsi aplikasi. Setiap data aplikasi sudah memiliki nama, gambar, deskripsi dan *link*. Penulis cukup menampilkannya saja untuk membuat fitur ini menjadi fungsional. Berikut diberikan potongan kode yang mengimplementasi akses aplikasi eksternal pada Gambar 5.14.

```
1.  {portal.map((item, index) => {
2.                  const Icon = item.image;
3.                  return (
4.                    <div
```

```
                    key={index}
                                        onClick={() => handlePanelClick(item.link)}
                  className=' flex cursor-pointer items-center justify-between gap-4 rounded-lg p-4 transition-all duration-500 hover:scale-105 hover:shadow-lg'
                >
                  <div className='relative flex h-16 w-full flex-row items-center'>
                    {typeof item.image === 'string' ? (
                      <Image
                        src={item.image}
                        alt={item.name}
                        width={64}
                        height={64}
                        className='flex rounded-full object-cover'
                      />
                    ) : (
                      <div
                        className={`flex h-16 w-16 items-center justify-center rounded-full bg-gradient-to-br ${item.color} text-2xl text-white drop-shadow-md`}
                      >
                        <Icon />
                      </div>
                    )}
                    <div className='ml-4 flex flex-col justify-center '>
                      <h2 className='text-xl font-semibold'>{item.name}</h2>
                      <p className='text-gray-500'>{item.desc}</p>
                    </div>
                  </div>
                </div>
              );
            })}

```

Gambar 5.14 Potongan Kode Akses Aplikasi Eksternal

5.2.7. Melihat Informasi Umum

Implementasi untuk fitur ini dilakukan sesuai dengan rancangan di Bab 4, yaitu menyajikan data referensi dalam bentuk tabel. Empat halaman terpisah dikembangkan untuk menampilkan data Daftar Negara, Mata Uang, Pajak, dan Wilayah Indonesia. Setiap halaman memanggil *endpoint API* yang sesuai untuk mengambil data. Data yang diterima kemudian ditampilkan menggunakan komponen tabel *RenderDataTable* yang telah dilengkapi dengan fungsionalitas pencarian dan paginasi untuk memudahkan navigasi pengguna. Karena memiliki pola yang sama, diberikan contoh potongan kode satu halaman untuk Daftar Negara pada Gambar 5.15.

```
1.   <div className='flex h-full w-full flex-col gap-
   4'>
2.       <div className='mb-4 flex flex-row gap-2
   text-base'>
3.         <Link
4.           href='/'
5.           className=' text-blue-500 hover:text-
   blue-700 hover:underline'
6.         >
7.           Dashboard
8.         </Link>
9.         <p>/</p>
10.        <p>Country</p>
11.      </div>
12.      <RenderDataTable
13.        title={t('country')}
14.        data={combinedData}
15.        columns={cols}
16.      />
17.    </div>
18.
```

Gambar 5.15 Potongan Kode Halaman Informasi Umum Daftar Negara

5.2.8. Kelola Hak Akses Portal

Fitur ini diimplementasikan sebagai sebuah komponen utama yang memiliki tiga tab utama, yaitu *tab permission, tab role* dan *tab user.* Pada komponen utama penulis memanfaatkan *library tab* dari *headlessui* dan menggunakan *conditional rendering* untuk implementasi pengguna berdasarkan hak akses. Pada Gambar 5.16 diberikan potongan kode untuk komponen utama.

```
1.   <div>
2.        {/* ...potongan kode breadcrumb... */}
3.
4.        <Tab.Group>
5.          <Tab.List onClick={handleResetId}>
6.            {/* Permission Tab */}
7.            <Tab>Permission</Tab>
8.
9.            {/* Role Tab - Conditional */}
10.                {getCanReadAllByRouter('Role')  &&
    <Tab>Role</Tab>}
11.
12.           {/* User Tab - Conditional */}
13.                {getCanReadAllByRouter('User')  &&
    <Tab>User</Tab>}
14.         </Tab.List>
15.
16.         <Tab.Panels>
17.           {/* Permission Panel */}
18.           {getCanReadAllByRouter('Permission') &&
    (
19.             <Tab.Panel>
20.                                    <TenantDropdown
    setTenantId={setTenantId}
    setIdForTarget={setRoleId} />
21.                        <RoleDropdown  id={tenantId}
    setRoleId={setRoleId} />
22.                <PermissionTable id={roleId} />
23.             </Tab.Panel>
24.           )}
25.
26.           {/* Role Panel */}
27.           {getCanReadAllByRouter('Role') && (
```

```
28.             <Tab.Panel>
29.               <RoleComponent tenant={data} />
30.             </Tab.Panel>
31.           )}
32.
33.           {/* User Panel */}
34.           {getCanReadAllByRouter('User') && (
35.             <Tab.Panel>
36.                                 <UserTenantDropdown
    setTenantId={setTenantIdForUser} />
37.               <UserTable id={tenantIdForUser} />
38.             </Tab.Panel>
39.           )}
40.         </Tab.Panels>
41.       </Tab.Group>
42.     </div>
```

Gambar 5.16 Potongan Kode Komponen Utama Hak Akses Portal

Pada *tab permission,* penulis menggunakan komponen *PermissionTable* dengan fitur *inline editing* untuk mengatur *permission* berdasarkan *role* dan *tenant*. Data diambil dari *API useGetPermissionByRole* dan ditampilkan dalam bentuk tabel dengan *checkbox* untuk setiap *permission*. Berikut pada Gambar 5.17 diberikan potongan kode untuk *tab permission.*

```
1.  const handleCheckboxChange = (pkid: number,
    field: string, value: boolean) => {
2.      setEditedData(prev => {
3.        const updated = prev.map(item =>
4.          item.pkid === pkid ? { ...item, [field]:
    value } : item
5.        );
6.        setIsDirty(true);
7.        return updated;
8.      });
9.    };
10.
11.   // Bulk toggle column
12.   const toggleColumn = (field: string, value:
    boolean) => {
```

```
13.     const updated = editedData.map(row =>
    ({ ...row, [field]: value }));
14.     setEditedData(updated);
15.     setIsDirty(true);
16.   };
17.
18.   // Save changes
19.   const handleSave = async () => {
20.     const changedRows = editedData.filter(edited
    => {
21.       const original = data.data.find(o => o.pkid
    === edited.pkid);
22.       return !original ||
23.         original.can_read_all !==
    edited.can_read_all ||
24.         original.can_create !== edited.can_create
    ||
25.         original.can_update !== edited.can_update
    ||
26.         original.can_delete !==
    edited.can_delete;
27.     });
28.
29.     if (changedRows.length === 1) {
30.       await updatePermission(changedRows[0]);
31.     } else {
32.       await bulkUpdatePermission({ permissions:
    changedRows });
33.     }
34.
35.     await refetch();
36.     setIsDirty(false);
37.   };
38.
39.   // Delete permission
40.   const handleDeleteRow = (pkid: number) => {
41.     Swal.fire({
42.       title: 'Are you sure?',
43.       icon: 'warning',
44.       showCancelButton: true,
45.     }).then(async result => {
46.       if (result.isConfirmed) {
```

```
47.         await deletePermission(pkid);
48.         refetch();
49.       }
50.     });
51.   };
52.
53.   return (
54.     <div>
55.       {/* ...potongan kode search & filters...
    */}
56.
57.       <table className='w-full border-collapse
    border'>
58.         <thead>
59.           <tr>
60.             <th>Module</th>
61.             <th>Router</th>
62.             <th>
63.               Can Read
64.               <input type='checkbox' onChange={e
    => toggleColumn('can_read_all',
    e.target.checked)} />
65.             </th>
66.             <th>
67.               Can Create
68.               <input type='checkbox' onChange={e
    => toggleColumn('can_create', e.target.checked)}
    />
69.             </th>
70.             <th>
71.               Can Update
72.               <input type='checkbox' onChange={e
    => toggleColumn('can_update', e.target.checked)}
    />
73.             </th>
74.             <th>
75.               Can Delete
76.               <input type='checkbox' onChange={e
    => toggleColumn('can_delete', e.target.checked)}
    />
77.             </th>
78.             <th>Select All</th>
```

```
79.             <th>Delete</th>
80.           </tr>
81.         </thead>
82.         <tbody>
83.           {editedData?.map(item => (
84.             <tr key={item.pkid}>
85.               <td>{item.Router.module_name}</td>
86.               <td>{item.Router.router_name}</td>
87.               <td>
88.                 <input
89.                   type='checkbox'
90.                   checked={item.can_read_all}
91.                   onChange={e =>
    handleCheckboxChange(item.pkid, 'can_read_all',
    e.target.checked)}
92.                 />
93.               </td>
94.               {/* ...checkbox lainnya... */}
95.               <td>
96.                 <button onClick={() =>
    handleDeleteRow(item.pkid)}>Delete</button>
97.               </td>
98.             </tr>
99.           ))}
100.              </tbody>
101.            </table>
102.
103.            {/* Save button - hanya muncul jika
    ada perubahan */}
104.            {isDirty && (
105.              <button onClick={handleSave}>Save
    Changes</button>
106.            )}
107.
108.            {/* ...potongan kode pagination...
    */}
109.          </div>
110.        );
111.      };
112.
```

Gambar 5.17 Potongan Kode *Tab Permission*

Pada *tab role,* penulis menggunakan komponen *RoleTable* dengan fitur CRUD *operations* lengkap untuk manajemen *role*. Tabel dilengkapi dengan *search*, *pagination*, dan *permission-based action buttons*. Berikut diberikan potongan kode pada Gambar 5.18

```
1.  const RoleTable = ({ data, isLoading, refetch,
    meta, setPage, setLimit, setSearch }:
    RoleTableProps) => {
2.    const { mutateAsync: deleteRole } =
    useSoftDeleteRole();
3.    const [tempSearch, setTempSearch] =
    useState('');
4.    const [modalOpen, setModalOpen] =
    useState(false);
5.    const [editModalOpen, setEditModalOpen] =
    useState(false);
6.    const [selectedRole, setSelectedRole] =
    useState<Role | null>(null);
7.
8.    const handleSearch = (search: string) => {
9.      setSearch(search);
10.     setPage(1); // Reset ke halaman pertama saat
    search
11.   };
12.
13.   const handleEdit = (role: Role) => {
14.     setSelectedRole(role);
15.     setEditModalOpen(true);
16.   };
17.
18.   const handleDelete = (pkid: number) => {
19.     Swal.fire({
20.       title: 'Are you sure to delete?',
21.       icon: 'warning',
22.       showCancelButton: true,
23.       confirmButtonText: 'Yes, Delete it!',
24.     }).then(async result => {
25.       if (result.isConfirmed) {
26.         try {
27.           await deleteRole(pkid);
28.           Swal.fire('Deleted!', 'Role has been
    deleted.', 'success');
```

```
29.           refetch();
30.         } catch (error) {
31.           Swal.fire('Error!', 'Something went
    wrong', 'error');
32.         }
33.       }
34.     });
35.   };
36.
37.   if (isLoading) {
38.     return <Skeleton className='h-full w-full'
    />;
39.   }
40.
41.   return (
42.     <div className='w-full'>
43.       {/* Header dengan tombol Create */}
44.       <div className='flex justify-between items-
    center mb-4'>
45.         <h2 className='text-xl font-bold'>Role
    Management</h2>
46.         {getCanCreateAllByRouter('Role') && (
47.           <button className='btn btn-primary'
    onClick={() => setModalOpen(true)}>
48.             Create Role
49.           </button>
50.         )}
51.       </div>
52.
53.       {/* Search & Limit Controls */}
54.       <div className='flex gap-4 mb-4'>
55.         <div>
56.           <label>Search</label>
57.           <input
58.             type='text'
59.             placeholder='Press enter to search'
60.             value={tempSearch}
61.             onChange={e =>
    setTempSearch(e.target.value)}
62.             onKeyDown={e => e.key === 'Enter' &&
    handleSearch(tempSearch)}
```

```
            onBlur={() =>
handleSearch(tempSearch)}
          />
        </div>
        <div>
          <label>Limit per page</label>
          <select value={meta?.limit} onChange={e
=> setLimit(Number(e.target.value))}>
            {[5, 10, 20, 50].map(size => (
              <option key={size}
value={size}>{size}</option>
            ))}
          </select>
        </div>
      </div>

      {/* Data Table */}
      <table className='w-full border-collapse
border'>
        <thead>
          <tr>
            <th>Role Name</th>
            <th>Description</th>
            <th>Type</th>
            <th>Tenant</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {data?.map((item: Role) => (
            <tr key={item.pkid}>
              <td>{item.name}</td>
              <td>{item.description}</td>
              <td>{item.type}</td>
              <td>{item.tenant_id}</td>
              <td>
                <div className='flex gap-2'>
                  {getCanUpdateAllByRouter('Role'
) && (
                    <button onClick={() =>
handleEdit(item)}>Edit</button>
                  )}
```

```
99.                     {getCanDeleteAllByRouter('Role'
    ) && (
100.                             <button onClick={() =>
    handleDelete(item.pkid)}>Delete</button>
101.                           )}
102.                         </div>
103.                       </td>
104.                     </tr>
105.                   ))}
106.                 </tbody>
107.               </table>
108.
109.               {/* Pagination */}
110.               <div className='flex justify-between
    items-center mt-4'>
111.                 <button
112.                   disabled={meta.page === 1}
113.                   onClick={() => setPage(meta.page
    - 1)}
114.                 >
115.                   Previous
116.                 </button>
117.                 <span>Page {meta.page} of
    {meta.maxPage}</span>
118.                 <button
119.                   disabled={meta.page ===
    meta.maxPage}
120.                   onClick={() => setPage(meta.page
    + 1)}
121.                 >
122.                   Next
123.                 </button>
124.               </div>
125.
126.               {/* Modals */}
127.               <CreateRoleModal modal={modalOpen}
    setModal={setModalOpen} onSuccess={refetch} />
128.               <UpdateRoleModal
129.                 modal={editModalOpen}
130.                 setModal={setEditModalOpen}
131.                 roleData={selectedRole}
132.                 onSuccess={refetch}
```

```
133.             />
134.           </div>
135.         );
136.       };
137.
```

Gambar 5.18 Potongan Kode *Tab Role*

Pada *tab user,* penulis menggunakan komponen *UserTable* dengan fitur *tenant-based user management*. Tabel menampilkan *user* yang terkait dengan *tenant* tertentu (berdasarkan *id prop*) dan menyediakan operasi CRUD dengan *permission control*. Berikut diberikan potongan kode pada Gambar 5.19.

```
1.  const UserTable = ({ id }: { id: number }) => {
2.    const { mutateAsync: removeUser } =
    useRemoveFromTenant();
3.    const [page, setPage] = useState(1);
4.    const [limit, setLimit] = useState(10);
5.    const [search, setSearch] = useState('');
6.    const [tempSearch, setTempSearch] =
    useState('');
7.    const [modalOpen, setModalOpen] =
    useState(false);
8.    const [editModalOpen, setEditModalOpen] =
    useState(false);
9.    const [selectedUser, setSelectedUser] =
    useState<User | null>(null);
10.
11.   const { data, isLoading, refetch } =
    useGetAllUser(page, limit, search, id);
12.
13.   const handleSearch = (search: string) => {
14.     setSearch(search);
15.     setPage(1); // Reset to first page when
    searching
16.   };
17.
18.   const handleEdit = (user: User) => {
19.     setSelectedUser(user);
20.     setEditModalOpen(true);
21.   };
```

```
22.
23.   const handleDelete = (pkid: number) => {
24.     Swal.fire({
25.       title: 'Are you sure to delete?',
26.       text: 'You will not be able to revert
    this!',
27.       icon: 'warning',
28.       showCancelButton: true,
29.       confirmButtonText: 'Yes, Delete it!',
30.     }).then(async result => {
31.       if (result.isConfirmed) {
32.         try {
33.           await removeUser(pkid);
34.           Swal.fire('Deleted!', 'User has been
    removed from tenant.', 'success');
35.           refetch();
36.         } catch (error) {
37.           Swal.fire('Error!', 'Something went
    wrong', 'error');
38.         }
39.       }
40.     });
41.   };
42.
43.   useEffect(() => {
44.     if (id) refetch();
45.   }, [id, page, limit, search]);
46.
47.   if (isLoading) {
48.     return <Skeleton className='h-full w-full'
    />;
49.   }
50.
51.   return (
52.     <div className='w-full'>
53.       {/* Header dengan tombol Create */}
54.       <div className='flex justify-between items-
    center mb-4'>
55.         <h2 className='text-xl font-bold'>User
    Management</h2>
56.         {getCanCreateAllByRouter('User') && (
```

```
57.           <button className='btn btn-primary'
    onClick={() => setModalOpen(true)}>
58.             Create User
59.           </button>
60.         )}
61.       </div>
62.
63.       {/* Search & Limit Controls */}
64.       <div className='flex gap-4 mb-4'>
65.         <div>
66.           <label>Search</label>
67.           <input
68.             type='text'
69.             placeholder='Press enter to search'
70.             value={tempSearch}
71.             onChange={e =>
    setTempSearch(e.target.value)}
72.             onKeyDown={e => e.key === 'Enter' &&
    handleSearch(tempSearch)}
73.             onBlur={() =>
    handleSearch(tempSearch)}
74.           />
75.         </div>
76.         <div>
77.           <label>Limit per page</label>
78.           <select
79.             value={limit}
80.             onChange={e => {
81.               setLimit(Number(e.target.value));
82.               setPage(1);
83.             }}
84.           >
85.             {[5, 10, 20, 50].map(size => (
86.               <option key={size}
    value={size}>{size}</option>
87.             ))}
88.           </select>
89.         </div>
90.       </div>
91.
92.       {/* User Table */}
```

```
93.       <table className='w-full border-collapse
    border'>
94.         <thead>
95.           <tr>
96.             <th>Username</th>
97.             <th>Full Name</th>
98.             <th>Email</th>
99.             <th>Role</th>
100.                <th>Actions</th>
101.              </tr>
102.            </thead>
103.            <tbody>
104.              {data?.data?.map((user: User) =>
    (
105.                <tr key={user.pkid}>
106.                  <td>{user.username}</td>
107.                  <td>{user.fullname}</td>
108.                  <td>{user.email}</td>
109.                  <td>{user.responsibilities[0
    ]?.name || 'No Role'}</td>
110.                  <td>
111.                    <div className='flex gap-
    2'>
112.                      {getCanUpdateAllByRouter
    ('User') && (
113.                        <button onClick={() =>
    handleEdit(user)}>Edit</button>
114.                      )}
115.                      {getCanDeleteAllByRouter
    ('User') && (
116.                        <button onClick={() =>
    handleDelete(user.pkid)}>Remove</button>
117.                      )}
118.                    </div>
119.                  </td>
120.                </tr>
121.              ))}
122.            </tbody>
123.          </table>
124.
125.          {/* Pagination */}
```

```
126.             <div className='flex justify-between
    items-center mt-4'>
127.               <button
128.                 disabled={page === 1}
129.                 onClick={() => setPage(prev =>
    prev - 1)}
130.               >
131.                 Previous
132.               </button>
133.               <span>Page {data?.meta.page} of
    {data?.meta.maxPage}</span>
134.               <button
135.                 disabled={page ===
    data?.meta.maxPage}
136.                 onClick={() => setPage(prev =>
    prev + 1)}
137.               >
138.                 Next
139.               </button>
140.             </div>
141.
142.             {/* Modals */}
143.             <CreateUserModal
144.               modal={modalOpen}
145.               setModal={setModalOpen}
146.               onSuccess={refetch}
147.             />
148.             <UpdateUserModal
149.               modal={editModalOpen}
150.               setModal={setEditModalOpen}
151.               userData={selectedUser}
152.               onSuccess={refetch}
153.             />
154.           </div>
155.         );
156.       };
```

Gambar 5.19 Potongan Kode *Tab User*

5.2.9. Kelola Hak Akses Aplikasi

Implementasi halaman ini berfokus pada komponen *ERPPermissionTable*. Fitur *filter* berdasarkan *tenant* dan peran (*role*) menjadi kunci utama. Ketika *admin* memilih *tenant* dan peran dari *dropdown*, sebuah pemanggilan *API* akan dilakukan untuk mengambil data hak akses yang spesifik untuk konteks tersebut.

Tabel hak akses kemudian ditampilkan, di mana setiap perubahan (misalnya, mencentang atau menghapus centang pada sebuah izin) akan secara otomatis memicu fungsi untuk mengirim pembaruan ke *backend* melalui API. Hal ini memastikan bahwa sistem *Role-Based Access Control* (RBAC) dapat dikelola secara efisien dan terpusat dari portal. Berikut diberikan potongan kode pada Gambar 5.20.

```
1.  const ERPPermissionTable: React.FC<{ id?: number }> = ({ id }) => {
2.    // ...state declarations (page, limit, search, editedData, isDirty)...
3.
4.     const { data, isLoading, refetch } = useGetERPPermissionByRoleId(id || 0, page, search, limit);
5.     const { mutateAsync: updatePermission } = useUpdateERPPermission();
6.    const { mutateAsync: bulkUpdatePermission } = useBulkUpdateERPPermission();
7.
8.   // Sync editedData dengan original data saat data berubah
9.    useEffect(() => {
10.     if (id !== 0 && data?.data) {
11.       setEditedData(data.data);
12.     } else {
13.       setEditedData([]);
14.     }
15.   }, [id, data]);
16.
```

```
  // Bulk toggle semua row dalam satu kolom
   const toggleColumn = (field: string, value: boolean) => {
       const updated = editedData.map(row => ({ ...row, [field]: value }));
    setEditedData(updated);
    setIsDirty(!isEqual(updated, data.data));
  };

  // Handle individual checkbox change
   const handleCheckboxChange = (pkid: number, field: string, value: boolean) => {
    setEditedData(prev => {
      const updated = prev.map(item =>
        item.pkid === pkid ? { ...item, [field]: value } : item
      );
      setIsDirty(!isEqual(updated, data.data));
      return updated;
    });
  };

  // Save changes (single or bulk update)
  const handleSave = async () => {
     const changedRows = editedData.filter(edited => {
      const original = data.data.find(o => o.pkid === edited.pkid);
      return !original ||
                      original.can_read_all    !== edited.can_read_all ||
        original.can_create !== edited.can_create ||
        original.can_update !== edited.can_update ||
       original.can_delete !== edited.can_delete;
    });

    if (changedRows.length === 0) return;

    if (changedRows.length === 1) {
      await updatePermission(changedRows[0]);
```

```
    } else {
        await bulkUpdatePermission({ permissions: changedRows });
    }

    await refetch();
    setIsDirty(false);
  };

    //   ...handleDeleteRow   dengan   SweetAlert confirmation...
  // ...handleSearch function...

  if (isLoading) return <Skeleton />;

  return (
    <div>
      <h2>ERP Permissions</h2>

      {/* Search & Limit controls */}
      <div>
        <input
          placeholder="Search..."
          value={tempSearch}
                                  onChange={e     => setTempSearch(e.target.value)}
            onKeyDown={e => e.key === 'Enter' && handleSearch(tempSearch)}
        />
           <select  value={limit}  onChange={e  => setLimit(Number(e.target.value))}>
          {/* Options 5,10,20,50 */}
        </select>
                      <button    onClick={()    => setIsModalOpen(true)}>Create Permission</button>
      </div>

      {/* Permission Table */}
      <table>
        <thead>
          <tr>
            <th>Module</th>
```

```
86.             <th>Router</th>
87.             <th>
88.               Can Read
89.                <input type="checkbox" onChange={e
    =>                    toggleColumn('can_read_all',
    e.target.checked)} />
90.             </th>
91.             <th>
92.               Can Create
93.                <input type="checkbox" onChange={e
    =>  toggleColumn('can_create',  e.target.checked)}
    />
94.             </th>
95.             <th>
96.               Can Update
97.                <input type="checkbox" onChange={e
    =>  toggleColumn('can_update',  e.target.checked)}
    />
98.             </th>
99.             <th>
100.                      Can Delete
101.                           <input  type="checkbox"
    onChange={e     =>     toggleColumn('can_delete',
    e.target.checked)} />
102.                    </th>
103.                    <th>Select All</th>
104.                    <th>Delete</th>
105.                  </tr>
106.               </thead>
107.               <tbody>
108.                 {editedData?.map(item => (
109.                   <tr key={item.pkid}>
110.                     <td>{item.Router.module_name
    }</td>
111.                     <td>{item.Router.router_name
    }</td>
112.                     <td>
113.                       <input
114.                         type="checkbox"
115.                         checked={item.can_read_a
    ll}
```

```
116.                                    onChange={e  =>
    handleCheckboxChange(item.pkid,    'can_read_all',
    e.target.checked)}
117.                          />
118.                        </td>
119.                          {/* ...similar checkboxes
    untuk can_create, can_update, can_delete... */}
120.                        <td>
121.                         {/* Select All Row checkbox
    */}
122.                         <input
123.                           type="checkbox"
124.                           onChange={e => {
125.                                  const  newValue  =
    e.target.checked;
126.                                   const  updated  =
    editedData.map(row =>
127.                                row.pkid === item.pkid
128.                                          ? { ...row,
    can_read_all:   newValue,   can_create:   newValue,
    can_update: newValue, can_delete: newValue }
129.                                      : row
130.                                );
131.                               setEditedData(updated);
132.                                setIsDirty(!isEqual(up
    dated, data));
133.                           }}
134.                         />
135.                        </td>
136.                        <td>
137.                             <button onClick={() =>
    handleDeleteRow(item.pkid)}>Delete</button>
138.                        </td>
139.                      </tr>
140.                  ))}
141.                </tbody>
142.              </table>
143.
144.              {/* Pagination controls */}
145.              <div>
```

```
146.                 <button disabled={page === 1}
    onClick={()    =>    setPage(prev    =>    prev    -
    1)}>Previous</button>
147.                 <span>Page {data?.meta.page} of
    {data?.meta.maxPage}</span>
148.                <button disabled={page === maxPage}
    onClick={()    =>    setPage(prev    =>    prev    +
    1)}>Next</button>
149.             </div>
150.
151.             {/* Save button - hanya muncul jika
    ada perubahan */}
152.             {isDirty && (
153.                <button onClick={handleSave}>Save
    Changes</button>
154.             )}
155.
156.             {/* Create Permission Modal */}
157.             {/* <ERPCreatePermissionModal /> */}
158.           </div>
159.         );
160.       };
```

Gambar 5.20 Potongan Kode Hak Akses Aplikasi

5.3. Implementasi Aplikasi *Enterprise Resource Planning (ERP)*

5.3.1 Mengelola Data Aset

Implementasi pada fitur ini berfokus pada penerapan hak akses dinamis. Saat pengguna pertama kali masuk ke sistem ERP, sistem terlebih dahulu memanggil *API* untuk mengambil data izin (*permission*) pengguna. Data izin ini (misalnya *can_create, can_update, can_delete*) kemudian disimpan secara lokal memanfaatkan *library Zustand*. Komponen tombol interaktif seperti "Tambah Aset", "Ubah", dan "Hapus" pada antarmuka diimplementasikan dengan logika *conditional rendering*, di mana

tombol hanya akan bisa dipencet jika data izin menyatakan pengguna memiliki hak akses yang sesuai. Berikut diberikan potongan kode implementasi pada Gambar 5.21

```
1.  const ComponentsAssetCategory = () => {
2.    const pathname = usePathname();
3.    const dispatch = useDispatch();
4.
5.    const modalForm = useSelector(
6.                  (state:       IRootState)       =>
    state.themeConfig.modalForm,
7.    );
8.    const modalEdit = useSelector(
9.                  (state:       IRootState)       =>
    state.themeConfig.modalEdit,
10.   );
11.   const { data: listCategory, isLoading, refetch }
    = useGetAllAssetCategory();
12.
13.   const handleSetModal = (isOpen: boolean) => {
14.     dispatch(setModalForm(isOpen));
15.   };
16.   const handleSetModalEdit = (isOpen: boolean) =>
    {
17.     dispatch(setModalEdit(isOpen));
18.   };
19.   return (
20.     <div className='space-y-5'>
21.           <CreateBreadCrumb   pathname={pathname}
    key={1} />
22.       <button
23.         type='button'
24.         className='btn btn-primary'
25.                                   onClick={()       =>
    dispatch(setModalForm(true))}
26.          disabled={!getCanCreateAllByRouter('Asset
    Category')}
27.       >
28.         Add New Category
29.       </button>
30.       <ModalCategoryAsset
31.         modal={modalForm}
```

```
        modalEdit={modalEdit}
        setModal={handleSetModal}
        setModalEdit={handleSetModalEdit}
        refetch={refetch}
      />
       <div className='relative flex h-full flex-
col gap-5'>
        <CategoryTable
          data={listCategory}
          isLoading={isLoading}
          refetch={refetch}
           canDelete={getCanDeleteAllByRouter('Ass
et Category')}
           canUpdate={getCanUpdateAllByRouter('Ass
et Category')}
        />
      </div>
    </div>
  );
};
```

Gambar 5.21 Potongan Kode Implementasi Mengelola Data Aset

5.3.2 Mengelola Data Inventaris

Implementasi pada fitur ini sama seperti implementasi pada mengelola data aset, yaitu berfokus pada penerapan hak akses dinamis. Saat pengguna pertama kali masuk ke sistem ERP, sistem terlebih dahulu memanggil API untuk mengambil data izin (permission) pengguna. Data izin ini (misalnya can_create, can_update, can_delete) kemudian disimpan secara lokal memanfaatkan library Zustand. Komponen tombol interaktif seperti "Tambah Aset", "Ubah", dan "Hapus" pada antarmuka diimplementasikan dengan logika conditional rendering, di mana tombol hanya akan bisa dipencet jika data izin menyatakan

pengguna memiliki hak akses yang sesuai. Berikut diberikan potongan kode implementasi pada Gambar 5.22

```
1.  const ComponentsUnit = () => {
2.    const pathname = usePathname();
3.    const dispatch = useDispatch();
4.    const modalForm = useSelector(
5.                  (state:       IRootState)       =>
    state.themeConfig.modalForm,
6.    );
7.    const modalEdit = useSelector(
8.                  (state:       IRootState)       =>
    state.themeConfig.modalEdit,
9.    );
10.   const { data: listCategory, isLoading, refetch }
    = useGetAllUnit();
11.   const handleSetModal = (isOpen: boolean) => {
12.     dispatch(setModalForm(isOpen));
13.   };
14.   const handleSetModalEdit = (isOpen: boolean) =>
    {
15.     dispatch(setModalEdit(isOpen));
16.   };
17.
18.   return (
19.     <div className='space-y-5'>
20.           <CreateBreadCrumb   pathname={pathname}
    key={1} />
21.       <button
22.         type='button'
23.         className='btn btn-primary'
24.                                 onClick={()       =>
    dispatch(setModalForm(true))}
25.          disabled={!getCanCreateAllByRouter('Unit'
    )}
26.       >
27.         Add New Unit
28.       </button>
29.       <ModalUnit
30.         modal={modalForm}
31.         modalEdit={modalEdit}
32.         setModal={handleSetModal}
33.         setModalEdit={handleSetModalEdit}
```

```
34.         refetch={refetch}
35.       />
36.        <div className='relative flex h-full flex-
    col gap-5 sm:h-[calc(100vh_-_150px)]'>
37.         <UnitTable
38.           data={listCategory}
39.           isLoading={isLoading}
40.           refetch={refetch}
41.            canDelete={getCanDeleteAllByRouter('Uni
    t')}
42.            canUpdate={getCanUpdateAllByRouter('Uni
    t')}
43.         />
44.       </div>
45.     </div>
46.   );
47. };
```

Gambar 5.22 Potongan Kode Implementasi Mengelola Data Inventaris

5.3.3 Mengelola Data Pembelian

Implementasi pada fitur ini sama seperti implementasi pada mengelola data aset dan inventaris sebelumnya, yaitu berfokus pada penerapan hak akses dinamis. Saat pengguna pertama kali masuk ke sistem ERP, sistem terlebih dahulu memanggil API untuk mengambil data izin (permission) pengguna. Data izin ini (misalnya can_create, can_update, can_delete) kemudian disimpan secara lokal memanfaatkan library Zustand. Komponen tombol interaktif seperti "Tambah Aset", "Ubah", dan "Hapus" pada antarmuka diimplementasikan dengan logika conditional rendering, di mana tombol hanya akan bisa dipencet jika data izin menyatakan pengguna memiliki hak akses yang sesuai. Berikut diberikan potongan kode implementasi pada Gambar 5.23.

```
1.  const ComponentsPurchasingOrder = () => {
2.    const { t } = getTranslation();
3.    const pathname = usePathname();
```

```
4.    const dispatch = useDispatch();
5.
6.    const modalForm = useSelector(
7.      (state:             IRootState)             =>
    state.themeConfig.modalForm,
8.    );
9.    const modalEdit = useSelector(
10.     (state:             IRootState)             =>
    state.themeConfig.modalEdit,
11.   );
12.
13.   const {
14.     data: listPurchaseOrder,
15.     isLoading,
16.     refetch,
17.   } = useGetAllPurchaseOrderHeaderOnly();
18.   const       {       generateCSV       }       =
    useGenerateCSVPurchaseOrder();
19.
20.   const handleSetModal = (isOpen: boolean) => {
21.     dispatch(setModalForm(isOpen));
22.   };
23.   const handleSetModelEdit = (isOpen: boolean) =>
    {
24.     dispatch(setModalEdit(isOpen));
25.   };
26.
27.   return (
28.     <div className='space-y-5'>
29.       <CreateBreadCrumb       pathname={pathname}
    key={1} />
30.       <button
31.         type='button'
32.         className='btn btn-primary'
33.         onClick={()                             =>
    dispatch(setModalForm(true))}
34.
    disabled={!getCanCreateAllByRouter('Purchase
    Order')}
35.       >
36.         {t('create')}       {t('purchase_order')}
    {t('new')}
```

```
      </button>
      <ModalPurchaseOrder
        modal={modalForm}
        modalEdit={modalEdit}
        setModal={handleSetModal}
        setModalEdit={handleSetModelEdit}
        refetch={refetch}
      />
      <div className='relative flex h-full flex-col sm:h-[calc(100vh_-_150px)]'>
        <PurchaseOrderTable
          data={listPurchaseOrder}
          isLoading={isLoading}
          refetch={refetch}
          exportCSV={generateCSV}
          canUpdate={getCanUpdateAllByRouter('Purchase Order')}
          canDelete={getCanDeleteAllByRouter('Purchase Order')}
        />
      </div>
    </div>
  );
};
```

Gambar 5.23 Potongan Kode Implementasi Mengelola Data Pembelian

5.3.4 Mengelola Data Penjualan

Implementasi pada fitur ini sama seperti implementasi pada mengelola data aset, inventaris dan pembelian sebelumnya, yaitu berfokus pada penerapan hak akses dinamis. Saat pengguna pertama kali masuk ke sistem ERP, sistem terlebih dahulu memanggil API untuk mengambil data izin (permission) pengguna. Data izin ini (misalnya can_create, can_update, can_delete) kemudian disimpan secara lokal memanfaatkan library Zustand. Komponen tombol interaktif seperti "Tambah Aset", "Ubah", dan "Hapus" pada antarmuka diimplementasikan dengan

logika conditional rendering, di mana tombol hanya akan bisa dipencet jika data izin menyatakan pengguna memiliki hak akses yang sesuai. Berikut diberikan potongan kode implementasi pada Gambar 5.24.

```
1.  const ComponentsSalesOrder = () => {
2.    const { t } = getTranslation();
3.    const pathname = usePathname();
4.    const dispatch = useDispatch();
5.    const modalForm = useSelector(
6.                  (state:       IRootState)       =>
    state.themeConfig.modalForm,
7.    );
8.    const modalEdit = useSelector(
9.                  (state:       IRootState)       =>
    state.themeConfig.modalEdit,
10.   );
11.   const {
12.     data: listSalesOrder,
13.     isLoading,
14.     refetch,
15.   } = useGetAllSalesOrderOnlyHeader();
16.   const handleSetModal = (isOpen: boolean) => {
17.     dispatch(setModalForm(isOpen));
18.   };
19.
20.   const handleSetModalEdit = (isOpen: boolean) =>
    {
21.     dispatch(setModalEdit(isOpen));
22.   };
23.   return (
24.     <div className='space-y-5'>
25.           <CreateBreadCrumb   pathname={pathname}
    key={1} />
26.       <button
27.         type='button'
28.         className='btn btn-primary'
29.                               onClick={()       =>
    dispatch(setModalForm(true))}
30.          disabled={!getCanCreateAllByRouter('Sales
    Order')}
31.       >
```

```
32.         {t('add')} {t('sales_order')} {t('new')}
33.       </button>
34.       <ModalSalesOrder
35.         modal={modalForm}
36.         modalEdit={modalEdit}
37.         setModal={handleSetModal}
38.         setModalEdit={handleSetModalEdit}
39.         refetch={refetch}
40.       />
41.        <div className='relative flex h-full flex-
    col sm:h-[calc(100vh_-_150px)]'>
42.         <SalesOrderTable
43.           data={listSalesOrder}
44.           isLoading={isLoading}
45.           refetch={refetch}
46.            canUpdate={getCanUpdateAllByRouter('Sal
    es Order')}
47.            canDelete={getCanDeleteAllByRouter('Sal
    es Order')}
48.         />
49.       </div>
50.     </div>
51.   );
52. };
```

Gambar 5.24 Potongan Kode Implementasi Mengelola Data Penjualan

5.3.5 Melakukan Verifikasi Autentikasi

Fitur ini diimplementasikan dengan menciptakan mekanisme pengecekan status autentikasi setiap kali pengguna mencoba mengakses halaman di aplikasi ERP. Pengguna memanfaatkan library *useEffect* sehingga sistem akan otomatis memeriksa keberadaan dan validitas *token* autentikasi yang tersimpan di *cookies* ketika halaman pertama kali *dirender.* Jika *token* tidak valid atau tidak ada, sistem akan mencegah halaman utama ditampilkan. Sebagai gantinya, sebuah komponen modal yang dibuat dengan memanfaatkan *headlessui* akan muncul untuk memberikan pesan bahwa sesi pengguna telah berakhir dan

mengarahkannya untuk melakukan login ulang melalui portal. Namun, jika ditemukan kredensial yang valid, maka sistem akan otomatis memindahkan pengguna ke *dashboard* untuk masuk ke aplikasi. Berikut pada gambar 5.25 potongan kode implementasi verifikasi autentikasi.

```
1.  const LoadingPageComponent = () => {
2.    const { data, isLoading, refetch } = useGetMe();
3.       const    {    setProfile,    profile    }    =
    useProfileStore();
4.    const { setPermissions } = usePermissionStore();
5.
6.       const     [modalOpen,     setModalOpen]     =
    useState(false);
7.
8.    const token = Cookies.get('access_token');
9.
10.   const handleSetModal = (isOpen: boolean) => {
11.     setModalOpen(isOpen);
12.   };
13.
14.   useEffect(() => {
15.     if (!token) {
16.       handleSetModal(true);
17.     }
18.   }, [token]);
19.
20.   useEffect(() => {
21.     if (!isLoading && !data) {
22.       refetch();
23.     }
24.     if (data) {
25.       setProfile(data);
26.     }
27.   }, [data, isLoading, refetch, setProfile]);
28.
29.   const {
30.     data: permission,
31.     isLoading: permissionLoading,
32.     refetch: permissionRefetch,
```

```
33.                         }                         =
    useGetPermissionByRoleId(profile?.responsibilitie
    s?.[0]?.pkid);
34.
35.   useEffect(() => {
36.     if (!permissionLoading && !permission) {
37.       permissionRefetch();
38.     }
39.     if (permission) {
40.       setPermissions(permission);
41.     }
42.       },     [permission,     permissionLoading,
    permissionRefetch, setPermissions]);
43.
44.   useEffect(() => {
45.     if (!isLoading && !permissionLoading && data
    && permission) {
46.       window.location.href = '/dashboard';
47.     }
48.     },   [isLoading,   permissionLoading,   data,
    permission]);
49.
50.   return (
51.     <div className='flex min-h-screen w-full flex-
    col items-center justify-center space-y-4'>
52.       <Loading />
53.         {isLoading  &&  <p  className='text-gray-
    700'>Data is loading...</p>}
54.       {permissionLoading && (
55.         <p className='text-gray-700'>Permission is
    loading...</p>
56.       )}
57.       <UnauthorizedModal modal={modalOpen} />
58.     </div>
59.   );
60. };
```

Gambar 5.25 Implementasi Kode Verifikasi Autentikasi

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Aplikasi DroneMEQ, Aplikasi ERP, dan Aplikasi Portal Smart Indicator Economy. Pengujian dilakukan untuk memastikan fungsionalitas dan kesesuaian hasil implementasi sistem dengan analisis dan perancangan sistem.

6.1 Tujuan Pengujian

Pengujian dilakukan terhadap Aplikasi DroneMEQ, Aplikasi ERP, dan Portal Smart Indicator Economy guna menguji kemampuan sistem dalam menjalankan fungsionalitas sesuai dengan kebutuhan yang telah dirancang. Pengujian ini bertujuan untuk memastikan bahwa seluruh fitur bekerja dengan baik, data dapat diproses dan ditampilkan secara akurat, serta sistem mampu menangani berbagai skenario penggunaan secara konsisten. Selain itu, pengujian juga dilakukan untuk mengidentifikasi potensi bug, mengevaluasi performa antarmuka pengguna, serta menilai aspek keamanan dasar sebelum sistem diimplementasikan pada lingkungan produksi.

6.2 Kriteria Pengujian

Kriteria pengujian pada Aplikasi DroneMEQ, ERP, dan Portal Smart Indicator Economy ditetapkan untuk menilai kualitas sistem dari berbagai aspek. Pengujian dilakukan secara manual untuk fungsionalitas dan stabilitas, serta menggunakan alat bantu otomatisasi Google Lighthouse untuk mengukur aspek teknis lainnya. Adapun kriteria yang digunakan dalam pengujian adalah sebagai berikut:

1. **Fungsionalitas:** Menilai apakah setiap fitur utama pada halaman bekerja sesuai dengan kebutuhan yang telah

dianalisis. Pengujian dilakukan secara manual dengan mengikuti skenario penggunaan.

2. **Keamanan Dasar:** Memastikan tidak ada informasi sensitif (seperti *token* atau kredensial) yang disimpan di tempat yang tidak aman atau terekspos di konsol *browser*.
3. **Stabilitas Interaksi:** Menilai apakah fitur tetap berjalan konsisten ketika digunakan secara berulang atau dalam urutan yang tidak terduga.
4. **Kinerja (Performance):** Mengukur seberapa cepat halaman dimuat dan menjadi interaktif bagi pengguna. Skor diukur menggunakan Google Lighthouse, dengan target skor di atas 80 dianggap baik.
5. **Aksesibilitas (Accessibility):** Mengukur apakah halaman dapat diakses dan digunakan oleh semua orang, termasuk pengguna dengan disabilitas. Skor diukur menggunakan Google Lighthouse, dengan target skor di atas 80.
6. **Praktik Terbaik (Best Practices):** Mengevaluasi apakah halaman mengikuti praktik pengembangan web modern, seperti penggunaan HTTPS dan keamanan pustaka JavaScript. Skor diukur menggunakan Google Lighthouse.
7. ***Search Engine Optimization* (SEO):** Menganalisis sejauh mana halaman dioptimalkan agar dapat ditemukan oleh mesin pencari. Skor diukur menggunakan Google Lighthouse.

6.3 Hasil Pengujian

Evaluasi dilakukan terhadap setiap halaman yang telah dikembangkan berdasarkan kriteria pada subbab 6.2. Pengujian fungsionalitas, keamanan dasar, dan stabilitas interaksi dilakukan secara manual, sedangkan pengujian kinerja, aksesibilitas, *best practices*, dan *Search Engine Optimization* (SEO) menggunakan

alat bantu *Google Lighthouse*. Berikut diberikan hasil pengujian untuk setiap halaman.

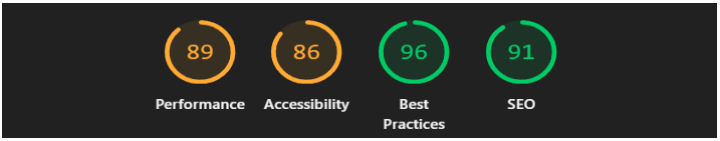
6.3.1 Aplikasi DroneMEQ

1. Halaman Registrasi

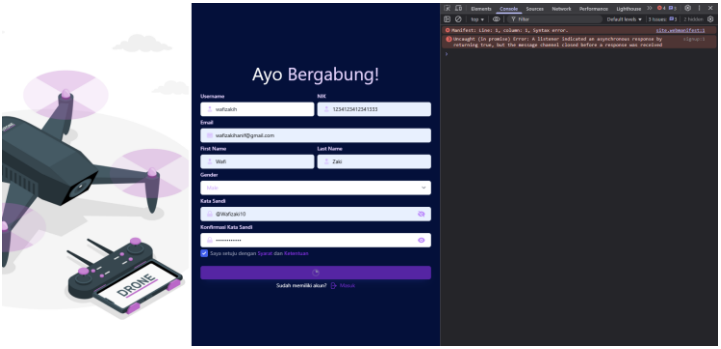
Secara fungsional, fitur registrasi telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (89), Best Practices (96), *SEO* (92) dan Aksesibilitas (86). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.1. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan *registrasi*.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.2.



Gambar 6.4 Hasil Audit *Google Lighthouse* Halaman Registrasi DroneMEQ



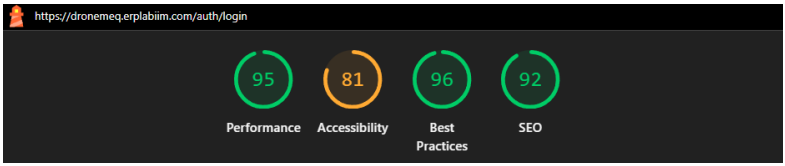
Gambar 6.5 Hasil Pengujian Manual Halaman Registrasi DroneMEQ

2. Halaman *Login*

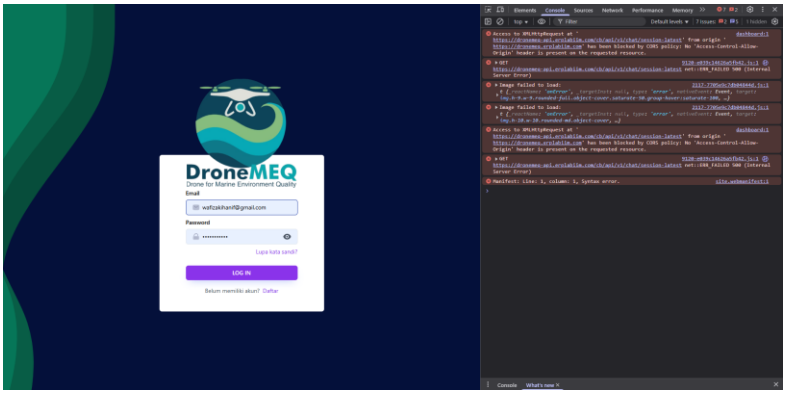
Secara fungsional, fitur Login telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (95), Best Practices (96), *SEO* (92) dan Aksesibilitas (81). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.3. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan *login*.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut, serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.4.



Gambar 6.6 Hasil Audit *Google Lighthouse* pada Halaman *Login* DroneMEQ



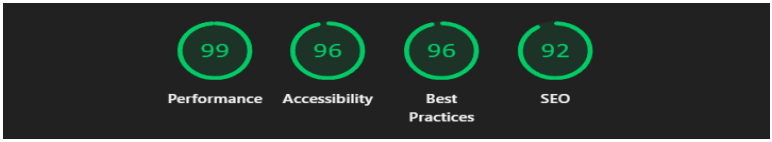
Gambar 6.7 Hasil Pengujian Manual pada Halaman *Login* DroneMEQ

3. Halaman Pemulihan Akun Drone

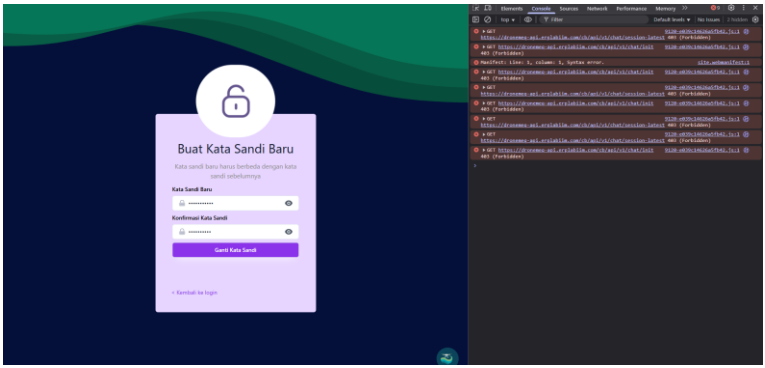
Secara fungsional, fitur pemulihan akun telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (99), Best Practices (96), *SEO* (92) dan Aksesibilitas (96). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.5. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan pemulihan akun.
- Pemeriksaan kredensial pada konsol *browser*.

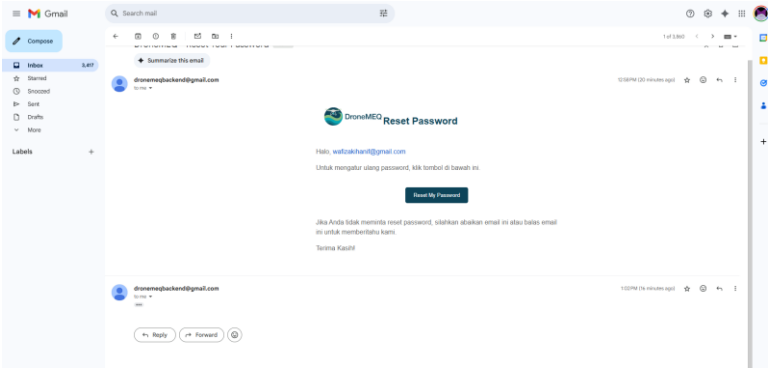
Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.6 dan bukti tautan pada *email* ditampilkan pada Gambar 6.7.



Gambar 6.*8* Hasil Audit Pengujian *Google Lighthouse* Halaman Pemulihan Akun DroneMEQ



Gambar 6.9 Hasil Pengujian Manual Halaman Pemulihan Akun DroneMEQ



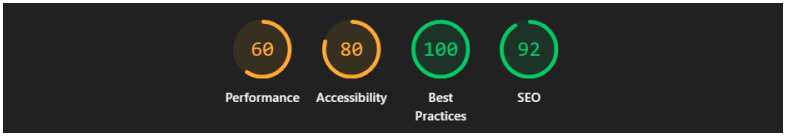
Gambar 6.10 Bukti Tautan Email Pemulihan Akun DroneMEQ

4. Halaman Pelanggaran DroneMEQ

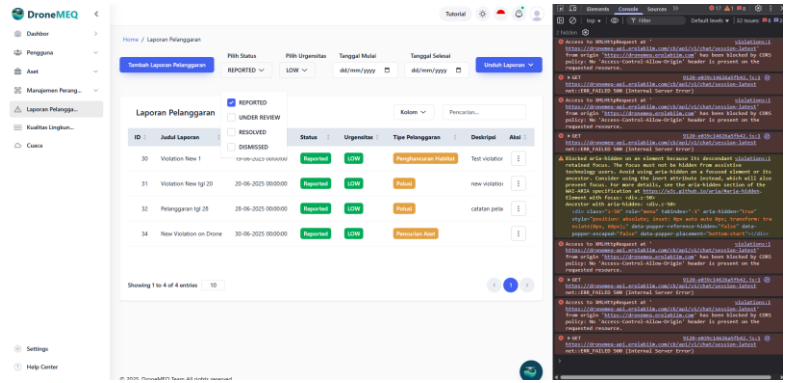
Secara fungsional, fitur *read* dan *filter* pada kelola data pelanggaran telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (60), Best Practices (100), *SEO* (92) dan Aksesibilitas (80). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.8. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji membuka halaman aset
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut. Dapat dilihat tampilan data menggunakan tabel, serta data berhasil disaring melalui fitur *filter* serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.9.



Gambar 6.11 Hasil Audit *Google Lighthouse* Halaman Pelanggaran DroneMEQ



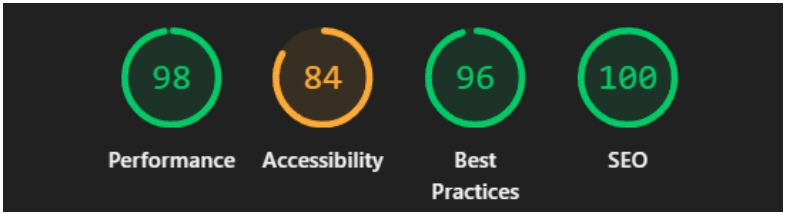
Gambar 6.12 Hasil Pengujian Manual Halaman Pelanggaran DroneMEQ

5. Halaman Aset DroneMEQ

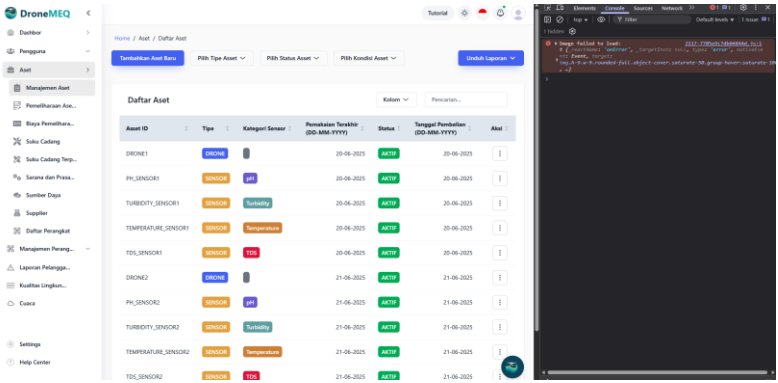
Secara fungsional, fitur *filter* pada kelola data aset telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (98), Best Practices (96), *SEO* (100) dan Aksesibilitas (84). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.10. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji memasukkan *filter* pada data.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut, serta tidak ada kebocoran kredensialpada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.11.



Gambar 6.13 Bukti Hasil *Audit Google Lighthouse* Halaman Aset DroneMEQ



Gambar 6.*14* Bukti Gambar Halaman Aset DroneMEQ

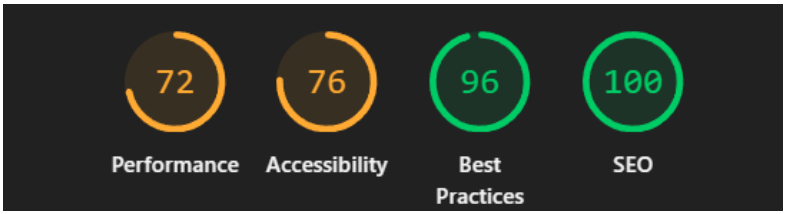
6. Komponen Pemilihan Bahasa

Secara fungsional, fitur pemilihan bahasa telah berhasil. Karena komponen bahasa terdapat pada seluruh halaman aplikasi DroneMEQ, maka pengujian akan dilakukan pada halaman *dashboard.* Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (72), Best Practices (96), *SEO* (100) dan Aksesibilitas (76). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.12. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

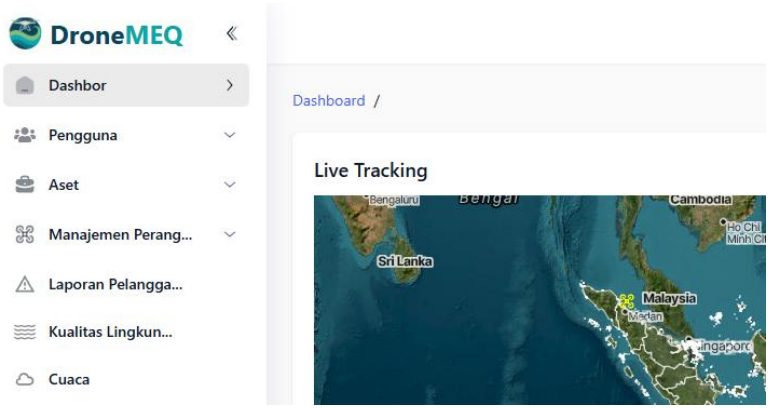
- Penguji melakukan registrasi sebagai pengguna.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut serta tidak ada kredensial yang bocor

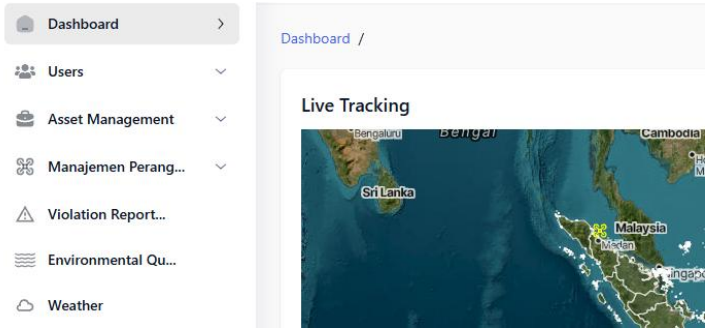
pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.13.



Gambar 6.15 Hasil Audit *Google Lighthouse Dashboard* DroneMEQ



Gambar 6.16 *Sidebar Dashboard* saat Menggunakan Bahasa Indonesia



Gambar 6.17 *Sidebar Dashboard* saat Menggunakan Bahasa Inggris

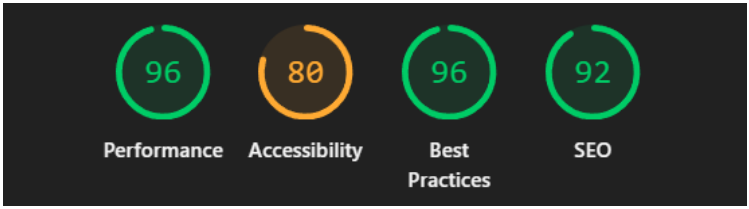
6.3.2 Aplikasi Portal Smart Indicator Economy

1. Halaman Registrasi Pengguna

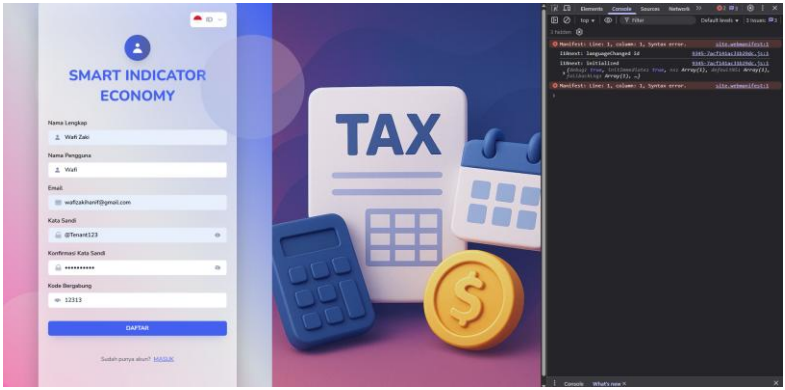
Secara fungsional, fitur registrasi sebagai pengguna telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (96), Best Practices (96), *SEO* (92) dan Aksesibilitas (82). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.15. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan registrasi sebagai pengguna.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.16.



Gambar 6.18 Hasil *Audit Google Lighthouse* Halaman Registrasi Pengguna



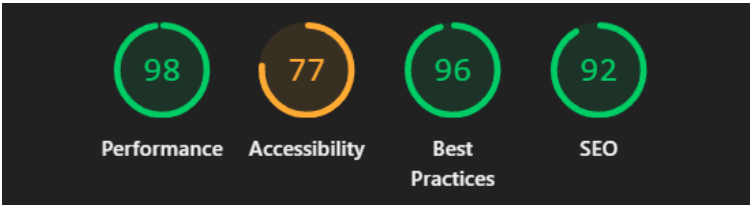
Gambar 6.19 Hasil Pengujian Manual Halaman Registrasi Pengguna

2. Halaman Registrasi *Tenant*

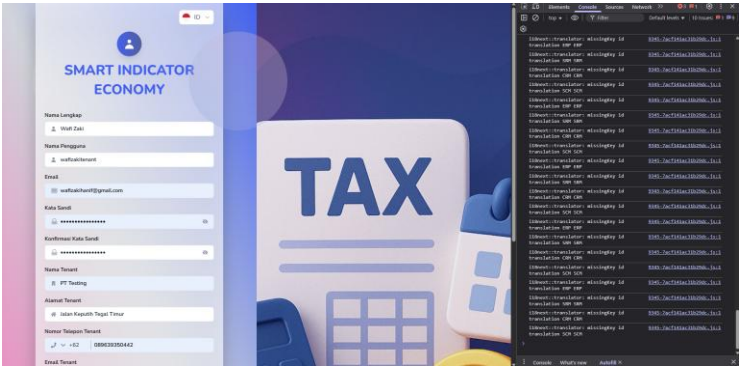
Secara fungsional, fitur registrasi sebagai *tenant* telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (96), Best Practices (96), *SEO* (92) dan Aksesibilitas (82). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.17. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan registrasi sebagai pengguna.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian, halaman ini dinyatakan berhasil melalui skenario tersebut serta tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.18.



Gambar 6.*20* Hasil Audit *Google Lighthouse* pada Halaman Registrasi *Tenant*



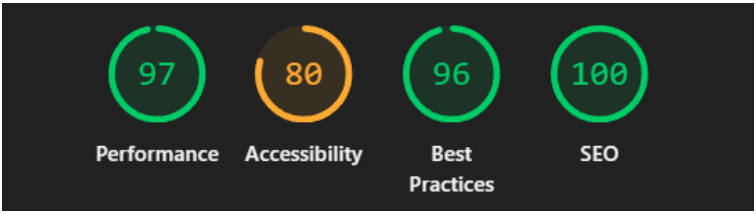
Gambar 6.21 Hasil Pengujian Manual pada Halaman Registrasi *Tenant*

3. Halaman *Login*

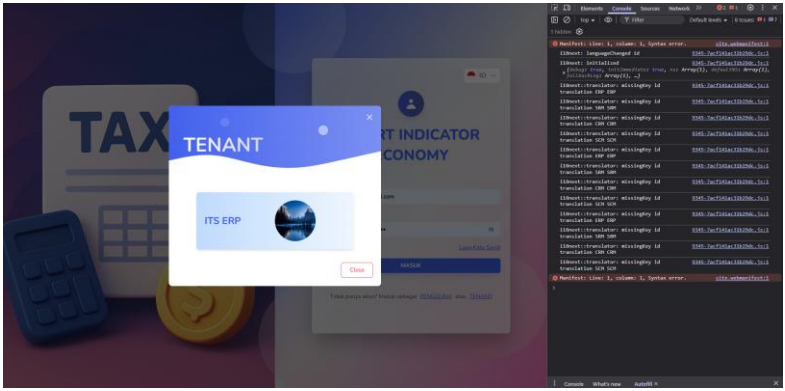
Secara fungsional, fitur *Login* telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (97), Best Practices (96), *SEO* (100) dan Aksesibilitas (80). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.19. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji melakukan login.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.20.



Gambar 6.22 Hasil Audit *Google Lighthouse* Halaman *Login*



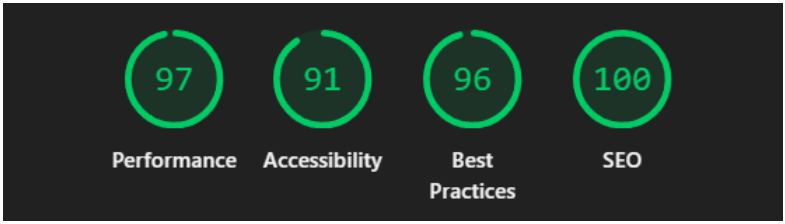
Gambar 6.23 Halaman Pengujian Manual Halaman *Login* Portal

4. Halaman Pemulihan Akun

Secara fungsional, fitur pemulihan akun telah berhasil. Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (97), Best Practices (96), *SEO* (100) dan Aksesibilitas (91). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.21. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

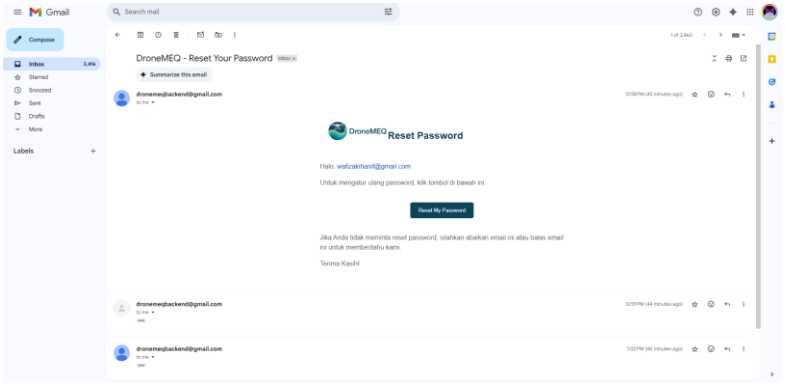
- Penguji melakukan pemulihan akun.
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.21 serta bukti tautan pemulihan akun pada *email* pada Gambar 6.23.



Gambar 6.24 Hasil Audit *Google Lighthouse* Halaman Pemulihan Akun

Gambar 6.25 Hasil Pengujian Manual Halaman Pemulihan Akun



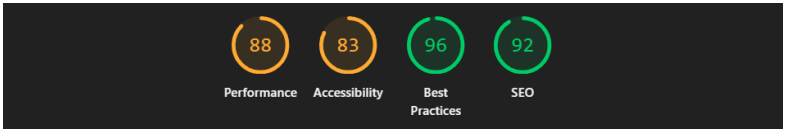
Gambar 6.26 Bukti Tautan *Email* Pemulihan Akun

5. Halaman Dashboard Portal

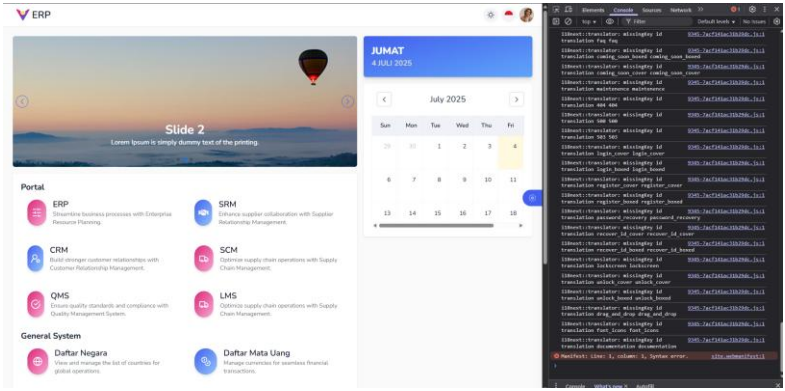
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (88), Best Practices (96), *SEO* (92) dan Aksesibilitas (83). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.24. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman *dashboard.*
- Penguji mencoba masuk ke dalam aplikasi ERP
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.25



Gambar 6.27 Hasil Audit *Google Lighthouse* Halaman *Dashboard* Portal



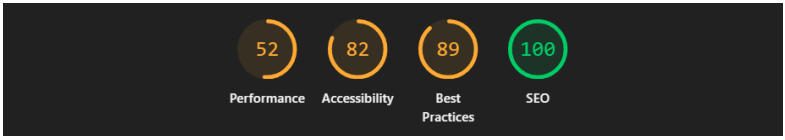
Gambar 6.28 Hasil Pengujian Manual Halaman *Dashboard* Portal

6. Halaman Profile Portal

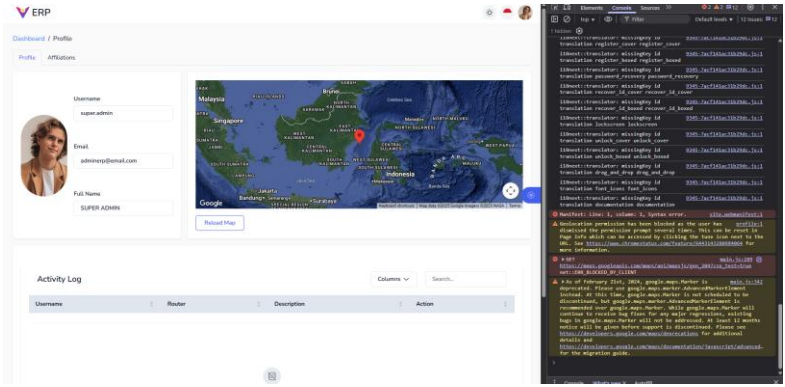
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (52), Best Practices (89), *SEO* (100) dan Aksesibilitas (82). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.26. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman profil.
- Penguji berpindah-pindah tab pada halaman profil.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.27.



Gambar 6.29 Hasil Audit *Google Lighthouse* Halaman Profil



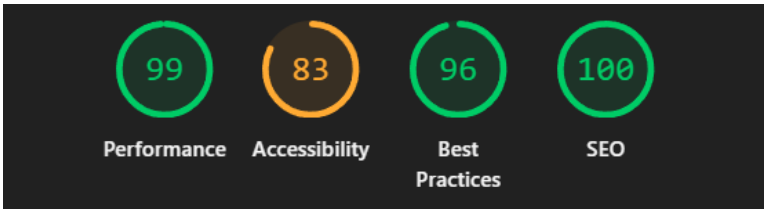
Gambar 6.30 Bukti Pengujian Manual Halaman Profil

7. Halaman Informasi Umum Daftar Negara

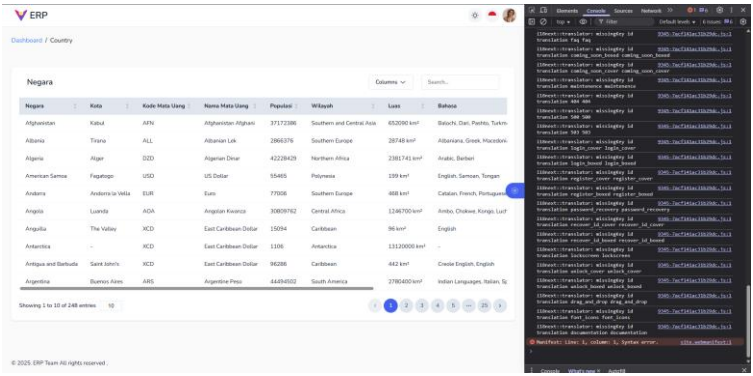
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (99), Best Practices (96), *SEO* (100) dan Aksesibilitas (83). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.28. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman informasi umum daftar negara.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.29.



Gambar 6.31 Hasil Audit Google Lighthouse Halaman Informasi Umum Daftar Negara



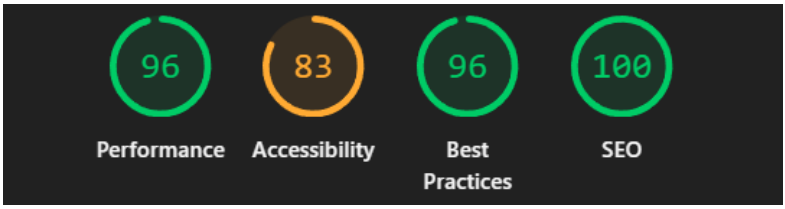
Gambar 6.32 Bukti Pengujian Halaman Informasi Umum Daftar Negara

8. Halaman General System Currency

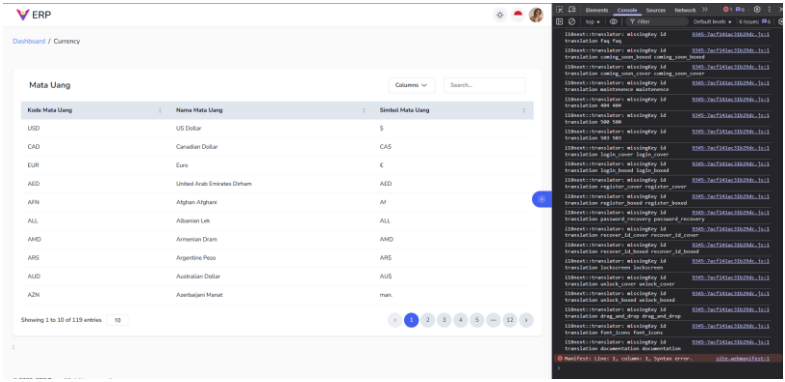
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (99), Best Practices (96), *SEO* (100) dan Aksesibilitas (83). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.30. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman informasi umum mata uang.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.31.



Gambar 6.33 Hasil Audit Google Ligththouse Halaman Informasi Umum Mata Uang



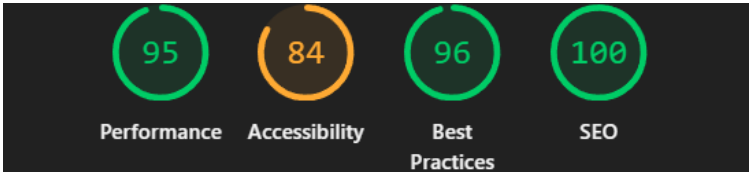
Gambar 6.34 Bukti Pengujian halaman Informasi Umum Mata Uang

9. Halaman *General System Region*

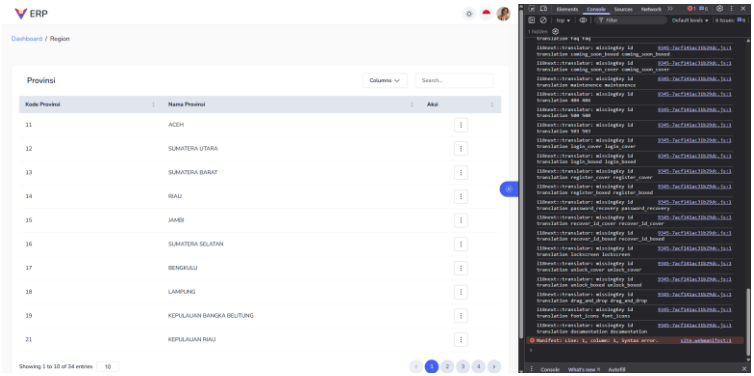
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (95), Best Practices (96), *SEO* (100) dan Aksesibilitas (84). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.32. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman informasi umum wilayah Indonesia.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.33.



Gambar 6.35 Hasil Audit Google Ligthhouse Halaman Informasi Umum Wilayah Indonesia



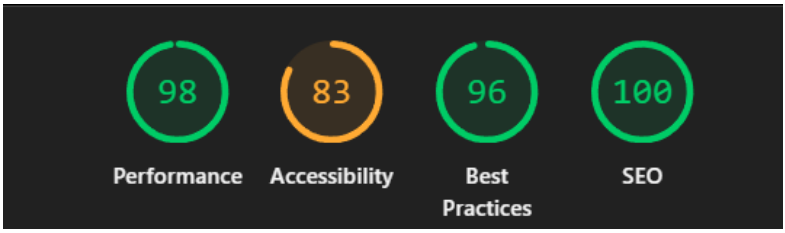
Gambar 6.36 Bukti Pengujian halaman Informasi Umum Wilayah Indonesia

10. Halaman General System Tax

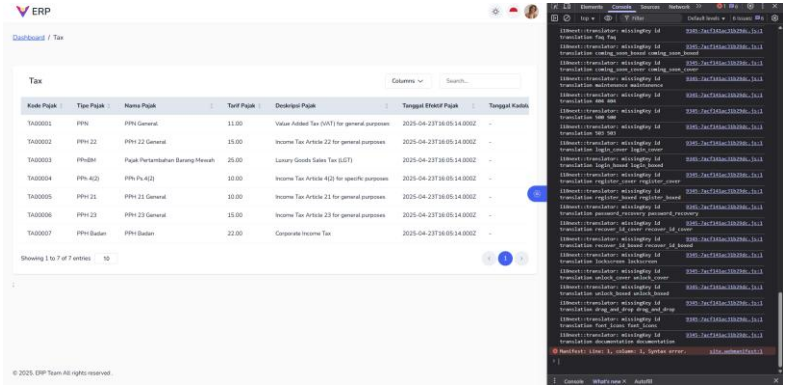
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (98), Best Practices (96), *SEO* (100) dan Aksesibilitas (83). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.34. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji mengakses halaman informasi umum pajak.
- Pemeriksaan kredensial pada konsol browser.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.35.



Gambar 6.37 Hasil Audit Google Ligththouse Halaman Informasi Umum Pajak



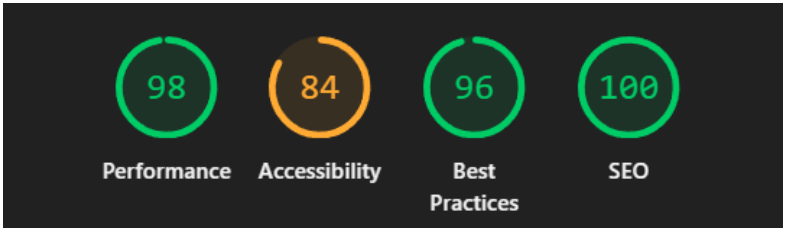
Gambar 6.38 Bukti Pengujian halaman Informasi Umum Pajak

11. Halaman Manajemen Akses Portal

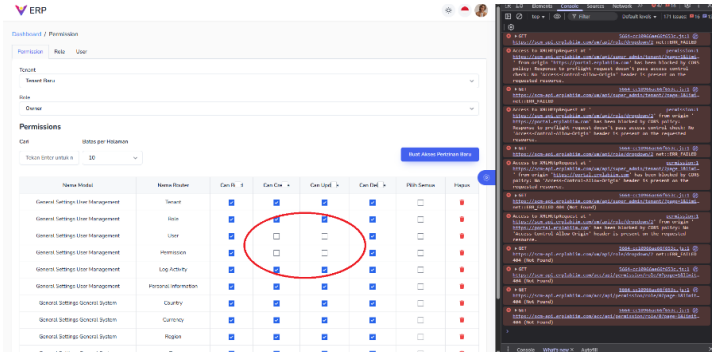
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (98), Best Practices (96), *SEO* (100) dan Aksesibilitas (84). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.36. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji masuk ke tab permission.
- Penguji masuk tenant "Tenant Baru" dan role "Owner".
- Penguji mematikan akses create dan update pada Router User dan Permission.
- Penguji masuk ke tab role.
- Penguji membuat role baru bernama "KROCO" pada tenant "Tenant Baru".
- Penguji masuk ke tab user.
- Penguji merubah role pengguna "akun_testing" menjadi "KROCO".
- Pemeriksaan kredensial pada konsol browser.

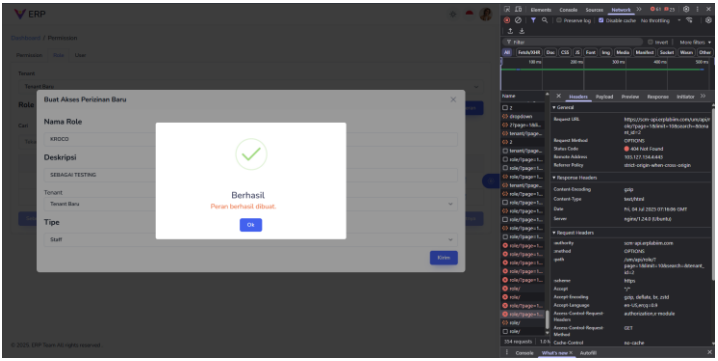
Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.37 sampai dengan 6.40.



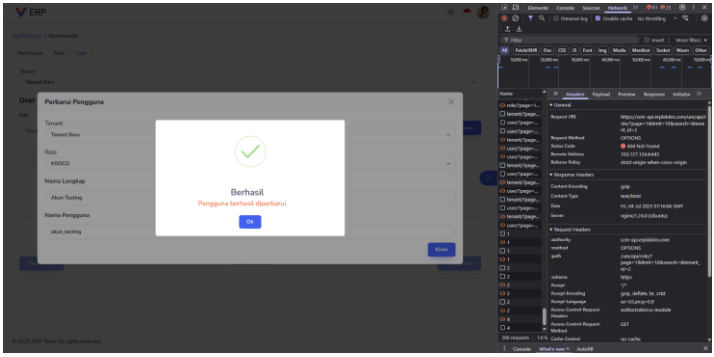
Gambar 6.39 Hasil Audit *Google Lighthouse* Pada Halaman Manajemen Akses Portal



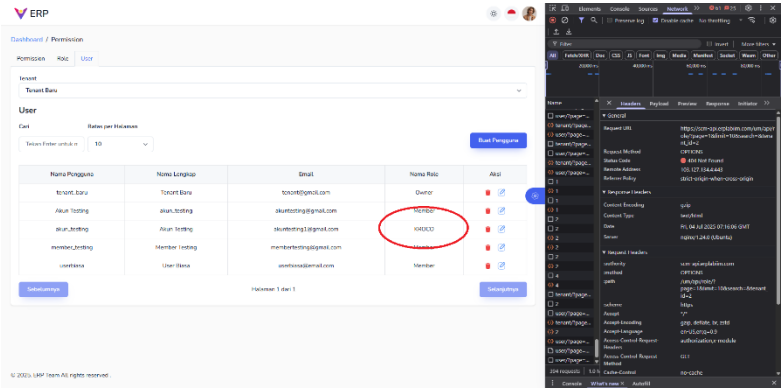
Gambar 6.40 Bukti Perubahan Hak Akses Portal kepada *Owner* Tenant Baru



Gambar 6.41 Berhasil Membuat *Role* Baru dengan Nama “KROCO”



Gambar 6.42 Bukti Berhasil Mengubah Role ”akun_testing” menjadi “KROCO”



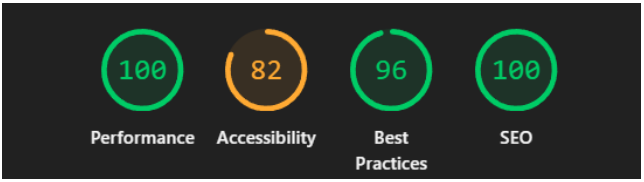
Gambar 6.43 Tampilan Role "akun_testing" Berubah menjadi "KROCO"

12. Halaman Manajemen Akses Aplikasi

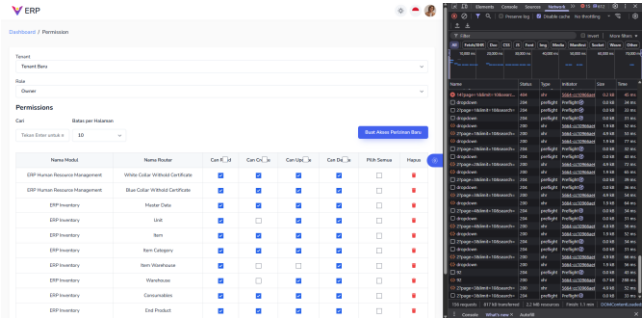
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (100), Best Practices (96), *SEO* (100) dan Aksesibilitas (82). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.41. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji masuk *tenant* "Tenant Baru" dan *role "Owner"*.
- Pada Modul *Inventory* penguji mematikan akses *create* pada *router User,* mematikan akses *create* dan *update* pada *router Item Warehouse,* serta mematikan akses *create* pada *router Warehouse.*
- Pemeriksaan kredensial pada konsol *browser*.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.42.



Gambar 6.44 Hasil Audit Google Lighthouse Halaman Manajemen Akses Aplikasi



Gambar 6.45 Bukti Hasil Pengujian pada Halaman Manajemen Akses Aplikasi

6.3.3 Aplikasi Enterprise Resource Planning (ERP)

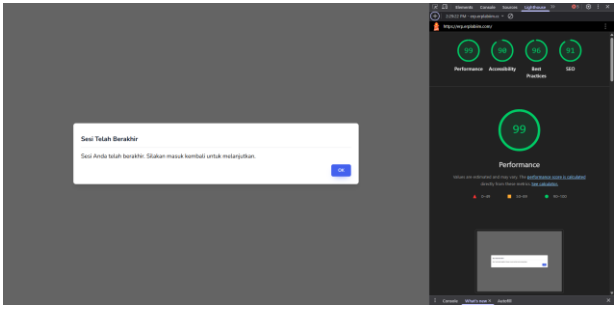
1. Verifikasi Autentikasi

Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (99), Best Practices (96), *SEO* (91) dan Aksesibilitas (90). Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

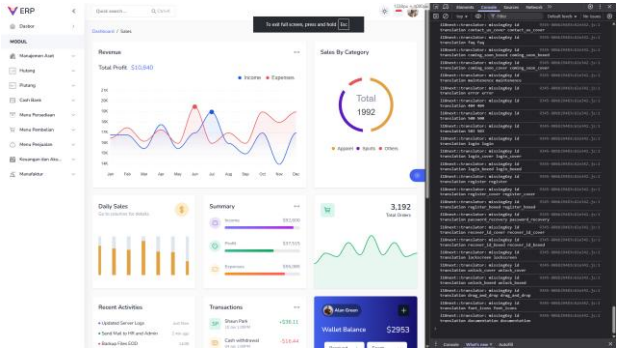
- Penguji berusaha membobol masuk tanpa kredensial.
- Penguji berusaha masuk melalui portal.
- Pemeriksaan kredensial pada konsol *browser.*

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada terjadi kebocoran kredensial. Berikut bukti hasil pengujian serta hasil

audit *Google Ligthhouse* ditampilkan pada Gambar 6.43, dan bukti penguji masuk melalui portal pada Gambar 6.44.



Gambar 6.46 Hasil Pengujian Halaman Verifikasi Autentikasi ERP



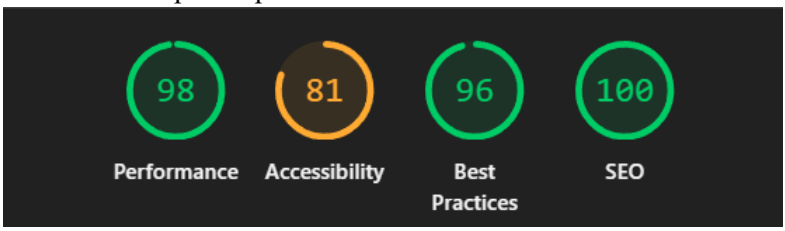
Gambar 6.47 Hasil Pengujian Masuk ke dalam Sistem ERP Melalui Portal

2. Halaman Kelola Aset ERP

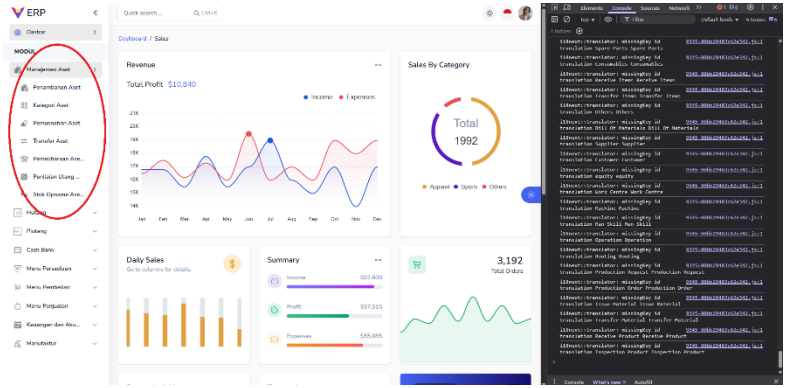
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (98), Best Practices (96), *SEO* (100) dan Aksesibilitas (81). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.45. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji tidak diberikan akses apapun pada modul aset submodul “tipe fiskal”.
- Pemeriksaan kebocoran kredensial pada konsol *browser.*

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Penguji tidak diberikan akses kepada submodul "tipe fiskal". Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.46.



Gambar 6.48 Hasil Audit *Google Lighthouse* Halaman Kelolan Aset ERP



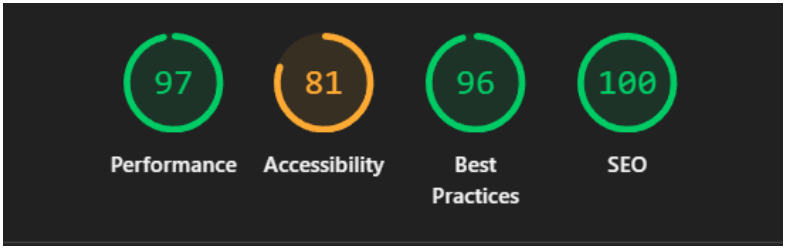
Gambar 6.49 Bukti Hasil Pengujian Halaman Kelola Aset ERP

3. Halaman Purchasing ERP

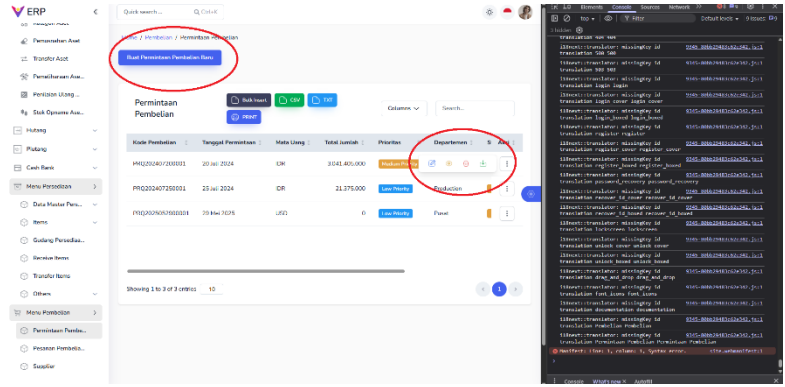
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (97), Best Practices (96), *SEO* (100) dan Aksesibilitas (81). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.47. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji diberikan akses penuh pada modul *purchasing*.
- Pemeriksaan kebocoran kredensial pada konsol *browser*.

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Penguji diberikan akses penuh sehingga seluruh operasi bisa dilakukan oleh penguji. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.48.



Gambar 6.50 Hasil Audit *Google Lighthouse* Halaman *Purchasing*



Gambar 6.51 Bukti Pengujian Manual Halaman *Purchasing*

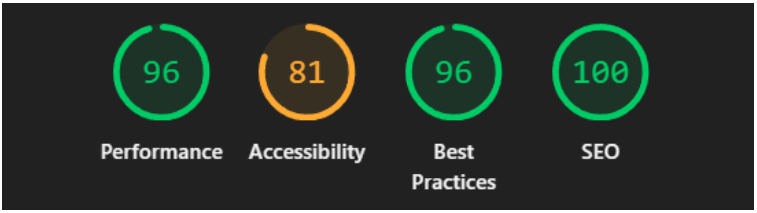
4. **Halaman Inventory (ERP)**

Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (96), Best Practices (96), *SEO* (100) dan Aksesibilitas (81). Bukti hasil

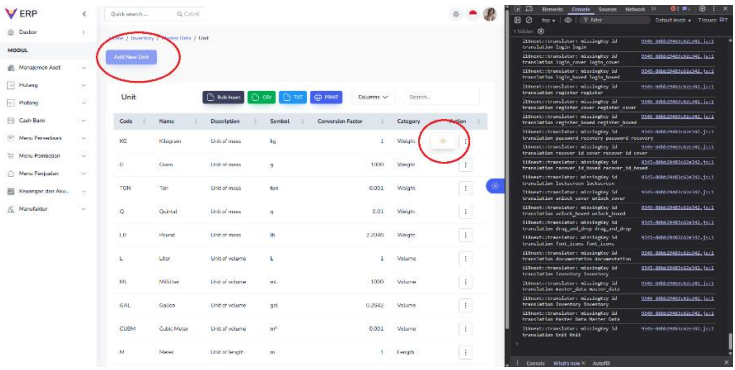
audit dari *Google Lighthouse* ditampilkan pada Gambar 6.49. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji hanya diberikan akses *read*.
- Penguji hanya bisa melakukan *read* tanpa bisa melakukan apa apa.
- Pemeriksaan kebocoran kredensial melalui konsol *browser.*

Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Penguji tidak dapat melakukan operasi *create, update* atau *delete* sesuai dengan konfigurasi yang diberikan. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.50.



Gambar 6.52 Hasil Audit *Google Lighthouse* Halaman Kelolan Aset ERP



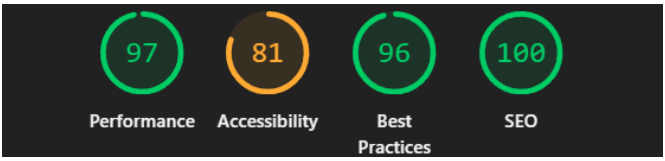
Gambar 6.53 Bukti Hasil Pengujian Halaman Kelola Aset ERP

5. Halaman Selling (ERP)

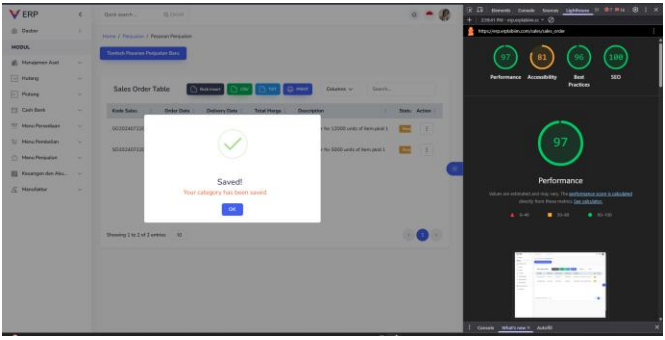
Hasil audit *Google Lighthouse* menunjukkan skor Kinerja (96), Best Practices (96), *SEO* (100) dan Aksesibilitas (81). Bukti hasil audit dari *Google Lighthouse* ditampilkan pada Gambar 6.51. Kemudian untuk pengujian manual dilakukan dengan skenario sebagai berikut:

- Penguji diberikan akses penuh pada modul *purchasing*.
- Penguji membuat data baru.
- Penguji menghapus data tersebut.
- Pemeriksaan kebocoran kredensial melalui konsol *browser*.

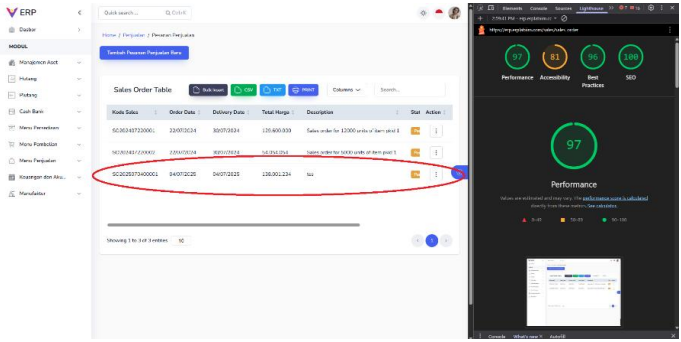
Setelah dilakukan pengujian tidak ada masalah yang muncul sehingga halaman ini dinyatakan berhasil melalui skenario tersebut. Kemudian dari segi keamanan, tidak ada kredensial yang bocor pada konsol. Berikut bukti hasil pengujian serta konsol *browser* ditampilkan pada Gambar 6.52 sampai dengan Gambar 6.54.



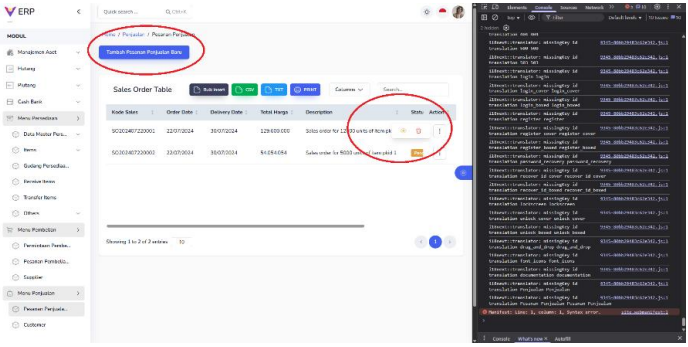
Gambar 6.54 Hasil Audit *Google Lighthouse* pada Modul *Selling*



Gambar 6.55 Bukti Notifikasi Berhasil Membuat Data Baru



Gambar 6.56 Bukti Data Baru Berhasil Dibuat



Gambar 6.57 Bukti Data Berhasil Dihapus dan Bukti Akses Penuh Penguji

6.4 Evaluasi Pengujian

Sub-bab ini akan membahas dan menganalisis temuan-temuan kunci dari hasil pengujian yang telah diringkas pada Tabel 6.2. Analisis dilakukan secara tematik berdasarkan kriteria pengujian untuk mendapatkan gambaran menyeluruh mengenai kualitas sistem

Tabel 6.2 Hasil Evaluasi Pengujian

Nama Halaman	Fungsionalitas	Kinerja (Skor)	Aksesibilitas (Skor)	Best Practices (Skor)	SEO (Skor)	Keamanan Dasar	Stabilitas Interaksi
Aplikasi DroneMEQ							
Halaman Registrasi	Berhasil Melewati Skenario	89	86	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Login	Berhasil Melewati Skenario	95	81	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Pemulihan Akun	Berhasil Melewati Skenario	99	96	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Pelanggaran	Berhasil Melewati Skenario	60	80	100	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Aset	Berhasil Melewati Skenario	98	84	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Komponen Pemilihan Bahasa (Halaman Dashboard)	Berhasil Melewati Skenario	72	76	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario

Portal Smart Indicator Economy							
Registrasi Pengguna	Berhasil Melewati Skenario	96	82	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Registrasi Tenant	Berhasil Melewati Skenario	96	82	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Login	Berhasil Melewati Skenario	97	80	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Pemulihan Akun	Berhasil Melewati Skenario	97	91	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Dashboard	Berhasil Melewati Skenario	88	83	96	92	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Profil	Berhasil Melewati Skenario	52	82	89	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Info Umum: Daftar Negara	Berhasil Melewati Skenario	99	83	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario

Info Umum: Mata Uang	Berhasil Melewati Skenario	99	83	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Info Umum: Wilayah	Berhasil Melewati Skenario	95	84	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Info Umum: Pajak	Berhasil Melewati Skenario	98	83	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Manajemen Akses Portal	Berhasil Melewati Skenario	98	84	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Manajemen Akses Aplikasi	Berhasil Melewati Skenario	100	82	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Aplikasi *Enterprise Resource Planning* (ERP)							
Verifikasi Autentikasi	Berhasil Melewati Skenario	99	90	96	91	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Kelola Aset	Berhasil Melewati Skenario	98	81	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario

Halaman Purchasing	Berhasil Melewati Skenario	97	81	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Inventory	Berhasil Melewati Skenario	96	81	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario
Halaman Selling	Berhasil Melewati Skenario	96	81	96	100	Tidak ada Kebocoran	Berhasil Melewati Skenario

1. Keberhasilan Fungsional, Keamanan, dan Stabilitas

Berdasarkan hasil pengujian manual, dapat disimpulkan bahwa seluruh fungsionalitas utama pada ketiga aplikasi (DroneMEQ, Portal Smart Indicator Economy, dan Enterprise Resource Planning) telah berhasil diimplementasikan sesuai dengan perancangan. Setiap skenario pengujian, mulai dari autentikasi hingga operasi CRUD, berjalan dengan sukses. Selain itu, semua halaman dinyatakan Aman dan Stabil, menandakan tidak adanya kebocoran data sensitif di sisi klien serta ketahanan sistem yang baik terhadap interaksi pengguna secara berulang. Skor pada kategori Best Practices dan SEO secara konsisten sangat tinggi, dengan sebagian besar halaman mencapai skor 96 hingga 100. Hal ini menunjukkan bahwa fondasi teknis aplikasi dibangun dengan mengikuti standar pengembangan web modern, termasuk penerapan HTTPS, struktur HTML yang baik, dan optimasi dasar untuk mesin pencari.

2. Kualitas Kode dan Praktik Terbaik

Skor pada kategori Best Practices dan SEO secara konsisten sangat tinggi, dengan sebagian besar halaman mencapai skor 96

hingga 100. Hal ini menunjukkan bahwa fondasi teknis aplikasi dibangun dengan mengikuti standar pengembangan web modern, termasuk penerapan HTTPS, struktur HTML yang baik, dan optimasi dasar untuk mesin pencari.

3. Analisis Kinerja (Performance)

Kategori Kinerja menunjukkan hasil yang paling bervariasi dan memberikan wawasan penting.

- **Halaman Performa Tinggi**

Halaman-halaman dengan interaksi sederhana seperti formulir autentikasi (Registrasi, Login) dan halaman yang menampilkan data statis (Info Umum) secara konsisten mencapai skor di atas 95. Ini menunjukkan bahwa untuk alur kerja yang ringan, aplikasi sangat responsif.

- **Halaman yang Memerlukan Optimasi**

Tantangan performa yang signifikan teridentifikasi pada halaman dengan kompleksitas tinggi. Halaman Profil Portal (skor 52) menjadi yang terendah, kemungkinan besar disebabkan oleh pemuatan komponen berat seperti peta interaktif dan log aktivitas. Selain itu, Halaman Pelanggaran DroneMEQ (skor 60) juga menunjukkan waktu muat yang perlu ditingkatkan, yang dapat diatribusikan pada proses rendering data tabel yang besar.

Dari data ini, terlihat korelasi kuat antara kompleksitas halaman dan skor Kinerja. Halaman yang kaya akan data menjadi kandidat utama untuk optimasi di masa mendatang.

4. Tinjauan Aksesibilitas

Secara umum, aplikasi memiliki tingkat aksesibilitas yang baik, dengan sebagian besar halaman mendapatkan skor di atas 80. Ini menandakan bahwa fondasi untuk pengalaman pengguna yang inklusif sudah ada. Meskipun demikian, beberapa skor yang lebih rendah, seperti pada Komponen Pemilihan Bahasa (skor 76), menunjukkan masih ada ruang untuk perbaikan minor.

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan proses kerja praktik yang telah dilaksanakan dalam pengembangan frontend untuk aplikasi DroneMEQ, sistem ERP, dan Portal Smart Indicator Economy, diperoleh kesimpulan yang menjawab rumusan masalah sebagai berikut:

1. Sistem otorisasi berbasis peran pada antarmuka ERP berhasil diimplementasikan melalui pendekatan integrasi dinamis dengan Portal utama. Implementasinya dilakukan dengan memanggil *API* untuk mengambil hak akses pengguna, menyimpannya secara lokal menggunakan *state management* (*Zustand*), dan kemudian menerapkan logika *conditional rendering* untuk menampilkan komponen interaktif seperti tombol "Tambah" atau "Hapus" hanya jika pengguna memiliki izin yang sesuai.
2. Portal utama berhasil dirancang sebagai gerbang terpusat yang efektif dengan menerapkan *dashboard* sebagai pusat navigasi dan sistem autentikasi terpadu (*Single Sign-On*). *Dashboard* dirancang dengan memanfaatkan *map* untuk menampilkan seluruh opsi aplikasi yang tersedia. Informasi aplikasi yang ditampilkan mencakup nama, deskripsi, gambar serta *link* sehingga pengguna dapat mengakses hanya dengan sekali *klik.* Kemudian sistem autentikasi *Single Sign-On* dapat dicapai dengan menggunakan kredensial yang sama untuk setiap aplikasi. Kredensial tersebut disimpan didalam *cookies* kemudian dikonfigurasi untuk menggunakan *domain* "*.erplabiim.com"* karena ekosistem *Smart Indicator*

Economy dibangun dari domain tersebut. Dengan begitu, hanya dengan melakukan *login sekali* saja, pengguna dapat mengakses seluruh aplikasi dalam ekosistem.

3. Pengembangan tampilan frontend untuk DroneMEQ berhasil mengoptimalkan pengalaman pengguna dalam memantau data lingkungan laut secara real-time. Ini dicapai dengan merancang dasbor interaktif yang mampu menyajikan data visual, seperti peta dan grafik, secara jelas dan responsif. Desain ini memudahkan pengguna dalam menganalisis informasi dan memantau perubahan data secara langsung.
4. Proses pengembangan frontend berhasil diselaraskan dengan arsitektur microservices di sisi backend melalui konsumsi data yang konsisten via REST API. Pendekatan ini memastikan setiap komponen frontend bersifat independen (decoupled) namun tetap dapat terintegrasi dengan layanan backend, sehingga mendukung skalabilitas, kolaborasi tim yang paralel, dan kemudahan perawatan sistem jangka panjang.
5. Hasil evaluasi menunjukkan bahwa secara fungsionalitas, 100% fitur utama pada aplikasi DroneMEQ, ERP, dan Portal Smart Indicator Economy berhasil diimplementasikan sesuai skenario yang mencakup pengujian serta pemeriksaan keamanan sehingga terbukti aman dan stabil. Dari sisi performa, aplikasi ERP menunjukkan hasil yang paling konsisten dan tinggi dengan skor rata-rata 97.2, diikuti oleh Portal Smart Indicator Economy yang juga sangat baik pada halaman-halaman utamanya (rata-rata 92.9), kemudian pada aplikasi DroneMEQ didapatkan skor performa dengan rata rata 85,5. Namun, terdapat *outlier* signifikan yang

memerlukan optimasi, terutama pada Halaman Profil Portal (skor 52) dan Halaman Pelanggaran DroneMEQ (skor 60), di mana kompleksitas data dan komponen menjadi tantangan utama. Untuk aspek aksesibilitas, sistem secara umum menunjukkan kualitas yang baik dengan sebagian besar halaman mencapai skor di atas 80, seperti pada aplikasi DroneMEQ dengan rata rata 83.83, Portal dengan rata-rata 83.2, dan ERP dengan rata-rata 82.8 yang membuktikan fondasi yang kuat untuk pengalaman pengguna yang inklusif.

7.2 Saran

Berdasarkan hasil pengujian performa dan aksesibilitas, terdapat beberapa aspek yang masih dapat ditingkatkan untuk menyempurnakan kualitas aplikasi DroneMEQ, ERP, dan Portal Smart Indicator Economy. Saran-saran berikut disampaikan sebagai masukan untuk pengembangan lebih lanjut:

- Optimalkan elemen Largest Contentful Paint (LCP) yang memakan waktu hingga 2.000 ms, guna mempercepat persepsi kecepatan halaman.
- Aktifkan kompresi teks (gzip/brotli) untuk mengurangi ukuran transfer data, dengan estimasi penghematan hingga 61 KiB.
- Preconnect ke origin eksternal lebih awal untuk mengurangi latensi awal, dengan estimasi penghematan waktu sekitar 80 ms.
- Kurangi pekerjaan main-thread, yang tercatat mencapai 2,0 detik dan menyebabkan interaksi terasa lambat.
- Hilangkan resource render-blocking, serta kurangi JavaScript dan CSS yang tidak terpakai, yang secara total mencapai ratusan KiB dan memperlambat proses render.

- Minify JavaScript untuk mengurangi ukuran file hingga 45 KiB.
- Hindari penggunaan legacy JavaScript pada browser modern dan minimalkan payload jaringan, yang tercatat hingga 12.890 KiB.
- Kurangi ukuran DOM yang berlebihan (1.385 elemen) dan long tasks pada main-thread untuk memperbaiki kelancaran aplikasi.
- Tambahkan label yang dapat diakses pada tombol dan elemen <select> untuk memperjelas fungsi kontrol antarmuka.
- Perbaiki rasio kontras warna antara teks dan latar belakang agar lebih mudah dibaca oleh pengguna dengan gangguan penglihatan.
- Tambahkan atribut [lang] pada elemen <html> untuk mendukung internasionalisasi dan lokalisasi konten secara lebih baik.
- Pastikan struktur list menggunakan elemen <li> secara konsisten sesuai standar semantik HTML.

Dengan melakukan perbaikan pada aspek performa dan aksesibilitas tersebut, diharapkan aplikasi dapat berjalan lebih cepat, lebih ringan, serta lebih inklusif bagi seluruh pengguna, termasuk mereka yang menggunakan teknologi bantu. Selain itu, optimasi ini juga akan mendukung implementasi sistem dalam lingkungan produksi dengan kualitas dan pengalaman pengguna yang lebih baik.

DAFTAR PUSTAKA

[1] J. Nielssen, 10 Usability Heuristics for User Interface Design, Nielsen Norman Group., 1995.

[2] E. Monk and B. Wagner, Concepts in Enterprise Resource Planning., Cengage Learning, 2012.

[3] S. Krug, Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, New Riders, 2014.

[4] C. Zhang, Y. Zhao, R.-Z. Li, J.-P. Li, S.-W. Wang and W.-X. Wang, A review on Internet of Things (IoT) applications in marine environment monitoring, Marine Pollution Bulletin, 165, 2021.

[5] OECD, Measuring the Digital Transformation: A Roadmap for the Future, OECD Publishing, 2020.

[6] P. Morville and L. Rosenfeld, Information Architecture for the World Wide Web, O'Reilly Media, 2006.

[7] Microsoft, Single sign-on (SSO) documentation, https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/auth-ssodefault, 2023.

[8] Microsoft, TypeScript Handbook, 2024.

[9] Vercel, Next.js Documentation, 2024.

[10] S. Newman, Building Microservices: Designing Fine-Grained Systems (2nd ed.), O'Reilly Media, 2021.

[11] K. Schwaber and J. Sutherland, The Scrum Guide™, https://scrumguides.org, 2020.

[12] K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.

[13] Tailwind Labs, Tailwind CSS Documentation, 2024.

[14] Google, Lighthouse: audit, performance, and reporting tool, 2025.

BIODATA PENULIS I

Nama	: Muhammad Wafi Zaki Hanif
Tempat, Tanggal Lahir	:Bontang, 10 Januari 2004
Jenis Kelamin	: Laki-Laki
Telepon	: +6289639350442
Email	: 5025221039@student.its.ac.id

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2022
Semester	: 6 (Enam)