BAB I
PENDAHULUAN

1.1 Latar Belakang

UD. Trois Dinamika Sentosa (TDS) adalah perusahaan distribusi yang bergerak di bidang penyediaan peralatan teknik dan alat pertukangan. Kurang lebih lima tahun beroperasi, TDS telah mengandalkan peran sebagai distributor dengan jaringan pemasaran yang stabil. Namun, seiring perkembangan bisnis, perusahaan menyadari pentingnya efisiensi operasional melalui pemanfaatan teknologi. Saat ini, TDS menggunakan *software* akuntansi untuk mengelola aspek keuangan dan manajemen stok barang [1]. *Software* ini digunakan oleh tim admin untuk melakukan pencatatan transaksi pembelian, penjualan, penghitungan keuangan, serta pemantauan stok secara *real-time* [2],[3]. Meskipun *software* akuntansi memberikan kemudahan dalam otomatisasi proses, implementasinya belum sepenuhnya optimal dalam mendukung kebutuhan operasional perusahaan.

Di sisi lain, tim sales TDS masih mengandalkan sistem pencatatan manual melalui buku atau dokumen fisik untuk menangani permintaan konsumen. Metode ini menimbulkan ketidakefisienan karena tim sales sering kali harus berada di luar kantor untuk melakukan pendekatan ke pelanggan. Akibatnya, terjadi keterlambatan dalam penyampaian data permintaan konsumen ke tim admin. Data yang tidak terintegrasi secara *real-time* ini berpotensi menimbulkan kesalahan pencatatan, duplikasi pesanan, atau ketidakakuratan laporan stok.

Penggunaan *software* akuntansi sebagai tulang punggung sistem manajemen TDS juga menghadapi beberapa tantangan. Pertama, antarmuka *software* dinilai rumit oleh tim admin dan kasir, sehingga memerlukan pelatihan khusus dan kesalahan operasional masih sering terjadi. Kedua, biaya langganan tahunan *software* akuntansi sebesar Rp1.520.948 (mengikuti fluktuasi kurs Dollar AS per-8 April 2026) menjadi beban keuangan bagi perusahaan skala menengah. Ketiga, *software* akuntansi memiliki keterbatasan teknis, seperti ketergantungan pada koneksi jaringan yang stabil. Masalah jaringan sering menyebabkan gangguan sinkronisasi data antara komputer admin, kasir, dan *server*, sehingga menghambat

produktivitas. Selain itu, akses *software* akuntansi hanya tersedia melalui komputer, tidak melalui perangkat telepon genggam, sehingga mempersulit tim sales atau manajemen untuk memantau data secara fleksibel [4], [5].

Selain permasalahan di atas, terdapat isu operasional signifikan pada proses pembayaran antara klien dengan UD. Trois Dinamika Sentosa yang belum tertangani dengan maksimal. Sebagai distributor, TDS memang melakukan pembelian barang dari importir atau produsen dengan mekanisme penagihan produsen/ importir kepada distributor (TDS) yang umumnya dapat dilakukan secara tempo, hal ini bukan menjadi fokus permasalahan utama. Namun, untuk penjualan ke klien, proses pemesanan hingga pengiriman sudah berjalan dengan cara tradisional. Klien memesan melalui tim sales, tim sales akan mencatat pesanan dan menyerahkan daftar permintaan klien ke admin/kasir, lalu melakukan input data penjualan, barang dikemas, dan pengiriman biasanya dilakukan oleh tim sales sendiri.

Permasalahan muncul pada tahap penagihan dan pencatatan pembayaran klien. Saat ini metode penagihan dan penerimaan pembayaran masih bersifat manual dan terfragmentasi, ada toko yang membayar tunai langsung ke sales, ada yang membayar melalui transfer bank, dan ada pula proses penagihan daring yang dilakukan secara manual melalui pesan WhatsApp oleh admin/kasir. Karena sistem tidak terintegrasi, admin/kasir perlu melakukan pengecekan secara manual untuk mencocokkan data transfer dengan faktur atau nota penjualan terkait. Selain itu, banyak transaksi yang bersifat pembayaran tempo artinya tidak semua pembayaran lunas saat barang diterima sehingga tim harus memonitor jatuh tempo, mengirim pengingat, dan melakukan rekonsiliasi berkala secara manual. Proses manual ini rentan terhadap keterlambatan pencatatan, kesalahan pencocokan bukti transfer, duplikasi atau kelalaian penagihan, serta memakan waktu dan tenaga yang cukup besar untuk administrasi [6].

Untuk mengatasi permasalahan tersebut, integrasi sistem pembayaran berbasis teknologi keuangan *(fintech)* menjadi opsi yang potensial dan relevan [7]. Dengan menghubungkan sistem informasi penjualan TDS ke vendor *Payment Gateway*, proses penagihan dan penerimaan pembayaran dapat terdokumentasi

secara otomatis, status pembayaran dapat diperbarui secara *real-time*, serta rekonsiliasi antara bukti transfer dan *invoice* dapat dipermudah atau diotomatisasi. *Payment Gateway* juga memungkinkan variasi metode pembayaran (misalnya transfer bank terintegrasi, virtual account, QRIS, kartu kredit/debit, dan notifikasi otomatis), serta fitur pengingat jatuh tempo *(reminder)* dan laporan transaksi yang bisa diakses oleh admin maupun manajemen. Implementasi solusi semacam ini diharapkan dapat mengurangi kesalahan manual, mempercepat proses penagihan, meningkatkan akurasi laporan keuangan, serta membantu *cash flow management* Perusahaan [8].

Berdasarkan uraian di atas, TDS memerlukan solusi sistem yang terintegrasi, lebih hemat biaya, dan mudah diakses tidak hanya untuk pengelolaan stok dan transaksi, tetapi juga untuk proses pembayaran klien. Pengembangan website berbasis *custom* yang terintegrasi dengan fitur-fitur teknologi keuangan (*fintech*) menjadi alternatif utama untuk menggantikan peran *software* akuntansi yang kurang fleksibel dan sistem pencatatan manual tim sales. Website ini diharapkan dapat dikustomisasi sesuai kebutuhan bisnis, seperti mengelola stok barang, transaksi keuangan, pemantauan permintaan konsumen secara *real-time*, serta integrasi otomatis untuk proses penagihan dan rekonsiliasi pembayaran. Fleksibilitas akses melalui komputer maupun perangkat *mobile* juga akan memudahkan tim sales dalam memasukan data langsung dari lapangan, mengurangi ketergantungan pada proses manual, dan mempercepat alur kerja. Dengan demikian, transformasi sistem ini diharapkan mampu meningkatkan akurasi data, efisiensi operasional, serta penghematan biaya jangka panjang bagi TDS khususnya dalam hal pengelolaan piutang dan proses pembayaran klien.

1.2 Rumusan Masalah

1. Bagaimana merancang sistem informasi pembelian dan penjualan di UD. Trois Dinamika Sentosa?
2. Bagaimana merancang sistem informasi keuangan di UD. Trois Dinamika Sentosa?
3. Bagaimana menerapkan sistem informasi pembelian, penjualan, dan keuangan berbasis teknologi keuangan dengan *framework* Laravel?

1.3 Tujuan Penelitian

1. Menganalisis dampak sistem pencatatan manual pada tim sales terhadap efisiensi operasional dan akurasi data di TDS.
2. Mengidentifikasi kelemahan *software* akuntansi dalam mendukung kebutuhan bisnis TDS, khususnya dari aspek teknis, biaya, dan antarmuka pengguna.
3. Merancang dan mengimplementasikan sistem berbasis website yang terintegrasi untuk menggantikan *software* akuntansi dan sistem manual, dengan fitur manajemen stok, penjualan, pembelian, dan keuangan [4], [3].
4. Mengimplementasikan sistem berbasis website yang terintegrasi dengan *framework* Laravel, dengan fitur manajemen stok, penjualan, keuangan, dan integrasi teknologi keuangan (*Payment Gateway*) [8].
5. Mengevaluasi efektivitas sistem website baru dalam meningkatkan aksesibilitas (melalui perangkat komputer dan *mobile*), mengurangi biaya operasional, serta mengintegrasikan alur kerja antara admin, kasir, dan tim sales [2].

1.4 Batasan Masalah

Agar penelitian ini lebih fokus, terarah, dan tidak meluas dari tujuan yang telah ditetapkan, maka penulis memberikan batasan-batasan masalah sebagai berikut:

1. **Ruang Lingkup Fungsionalitas**

   Sistem informasi yang akan dikembangkan mencakup modul-modul utama untuk kebutuhan operasional perusahaan, yaitu:

   - **Manajemen Keuangan:**

     Pencatatan transaksi pembelian, penjualan, pembayaran, penerimaan, integrasi pembayaran klien melalui *Payment Gateway*, perhitungan keuangan dan rekapitulasi asset tetap. Sistem ini mencakup analisis keuangan sederhana seperti *cash flow forecasting* atau laporan laba rugi [2], [6].

   - **Manajemen Penjualan:**

     Pencatatan permintaan dari konsumen yang diinput oleh tim sales, serta pencatatan status pembayaran baik tempo maupun langsung.

- **Manajemen Pembelian:**

  Pencatatan pembelian barang yang diinput oleh tim admin dan kasir, serta permintaan barang pesanan dari tim sales.

- **Manajemen Stok:**

  Pemantauan dan pengelolaan stok barang secara *real-time* yang terintegrasi dengan penjualan dan pembelian [3].

Sistem tidak mencakup modul manajemen sumber daya manusia (HRM) secara lengkap seperti presensi, penilaian kinerja, atau modul untuk pelanggan (misalnya, portal pelanggan untuk melacak pesanan).

2. **Ruang Lingkup Teknologi**

   Pengembangan sistem informasi ini difokuskan menggunakan *framework* Laravel dengan basis data MySQL. Penelitian ini tidak akan melakukan perbandingan antara Laravel dengan *framework* sejenis atau bahasa pemrograman lainnya [4], [9].

3. **Ruang Lingkup Pengguna**

   Pengguna sistem yang akan dirancang terbatas pada pihak internal UD. Trois Dinamika Sentosa, yaitu tim admin sebagai pengelola utama data, tim kasir sebagai pengelola penjualan dan pembelian, dan tim sales sebagai pencatat data permintaan konsumen.

4. **Ruang Lingkup Implementasi**

   Sistem ini dirancang dan diimplementasikan secara khusus untuk studi kasus pada UD. Trois Dinamika Sentosa. Penelitian ini tidak membahas skalabilitas sistem untuk diterapkan di perusahaan lain dengan alur bisnis yang berbeda.

5. **Ruang Lingkup Evaluasi**

   Efektivitas sistem baru akan dievaluasi berdasarkan kemudahan penggunaan *(usability)*, akurasi data, efisiensi alur kerja antara tim admin, kasir dan sales, serta efektivitas integrasi pembayaran melalui *Payment Gateway* [10]. Penelitian ini tidak melakukan analisis mendalam terhadap dampak finansial jangka panjang seperti *Return on Investment (ROI).*

BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi Keuangan

Sistem Informasi Keuangan (SIK) didefinisikan sebagai subsistem krusial dari Sistem Informasi Manajemen yang dirancang secara spesifik untuk mengumpulkan, mencatat, memproses, dan menyimpan data keuangan guna menghasilkan informasi strategis bagi pengambilan keputusan [2]. Dalam konteks operasional, kerangka sistem ini tidak sekadar berfungsi sebagai alat pencatatan kas, melainkan instrumen esensial untuk memecahkan berbagai permasalahan finansial perusahaan dengan melacak seluruh arus transaksi secara terintegrasi [6]. Efektivitas sebuah SIK sangat dipengaruhi oleh tingkat kepatuhan *(compliance)*, sistem pengawasan internal, dan kualitas data yang divalidasi [2]. SIK yang dirancang dengan baik memungkinkan otomatisasi pelaporan yang akurat sehingga dapat dimanfaatkan secara optimal untuk evaluasi bisnis yang sedang berjalan, manajemen anggaran, hingga mendukung kelancaran arus dokumen operasional harian terintegrasi seperti pesanan pembelian *(purchase order)* dan penagihan penjualan *(invoice)*.

Untuk memastikan keandalan informasi yang diproses, fungsionalitas dan kualitas SIK dapat dievaluasi secara komprehensif menggunakan metrik kerangka kerja *PIECES (Performance, Information, Economy, Control, Efficiency, dan Service)* [6]. Penerapan aspek kontrol dan keamanan dalam standar ini sangat krusial, terutama ketika sistem dituntut untuk mengelola basis data yang terstruktur dengan puluhan tabel relasional, serta harus mampu mengatur hierarki pembatasan akses pengguna berdasarkan kewenangan perannya di dalam struktur organisasi [6]. Lebih jauh, kualitas pemrosesan sistem secara langsung memengaruhi kemudahan manajemen dalam menganalisis informasi, meminimalkan kesalahan pencatatan *(human error)*, dan meningkatkan efisiensi waktu [2]. Oleh karena itu, sistem informasi keuangan pada dasarnya adalah fondasi penggerak yang mengonversi data mentah menjadi landasan pertanggungjawaban tata kelola bisnis yang akuntabel, transparan, dan adaptif.

2.2 Manajemen Aset Perusahaan & Manajemen Stok Gudang

Manajemen aset perusahaan merupakan kerangka strategis untuk melacak, memelihara, dan mengoptimalkan siklus hidup aset tetap secara menyeluruh demi efisiensi bisnis [1]. Keandalan sistem manajemen aset sangat bergantung pada akurasi pencatatan penyusutan *(depreciation)* dan proyeksi nilai sisa *(residual value)* pada akhir umur ekonomis suatu aset, seperti contohnya kalkulasi pada siklus hidup 5 tahun untuk armada operasional. Melalui pendekatan *Life Cycle Costing (LCC)*, entitas bisnis dapat mengidentifikasi total biaya yang dikeluarkan sejak akuisisi hingga tahapan penghapusan aset *(scrapping)*. Penerapan pelacakan aset yang tervalidasi di dalam rancangan basis data terstruktur, misalnya melalui pemetaan relasi yang solid, master data dengan transaksi pemeliharaan, sehingga membantu manajemen meminimalkan beban finansial akibat peralatan yang sudah habis masa pakainya. Dengan arsitektur data yang terukur ini, organisasi mampu menjamin transparansi pelaporan dan mengambil keputusan presisi terkait peremajaan armada investasi guna menjaga stabilitas layanan [1].

Sementara itu, manajemen stok gudang memainkan peranan sentral dalam mengendalikan ketersediaan produk dagang untuk memastikan rantai pasok, dengan siklus yang terintegrasi penuh dari penerimaan pesanan pembelian *(purchase order*) hingga eksekusi faktur penjualan *(sales invoice)* [3], [11]. Instrumen paling vital dalam menjaga validitas aliran barang ini adalah prosedur *stock opname*, yakni proses verifikasi fisik rutin yang secara khusus dirancang untuk merekonsiliasi pencatatan sistem digital dengan ketersediaan aktual di lapangan [12]. Integrasi aktivitas stock opname ke dalam sistem manajemen pergudangan terbukti secara ilmiah mampu menekan angka persentase ketidaksesuaian *(discrepancy)* persediaan secara signifikan [3]. Agar operasional pergerakan stok berjalan aman dan akuntabel, alur logistik ini secara teoritis harus diikat dengan kontrol akses berbasis peran *(RBAC/ Role-Based Access Control)* yang ketat. Sehingga pemisahan kewenangan antara entitas pengelola gudang, bagian penjualan, kasir, dan pemilik dapat berjalan transparan serta bebas dari risiko manipulasi data [12].

2.3 Metode Agile dalam Perancangan Sistem Informasi

Metode pengembangan perangkat lunak Agile merupakan fondasi perancangan untuk meminimalkan berbagai kendala rigiditas yang sering ditemui pada pendekatan pengembangan perangkat lunak tradisional. Secara konseptual, Agile berpusat pada kolaborasi intensif antara tim pengembang dan pemangku kepentingan, kemampuan adaptasi yang tinggi terhadap perubahan kebutuhan sistem, serta penerapan siklus pengiriman fungsionalitas secara singkat dan berulang *(iterative development)* [5]. Untuk mengimplementasikan konsep dasar ini, pengembang umumnya memanfaatkan kerangka kerja spesifik seperti Scrum atau Kanban, yang didukung oleh rutinitas harian *(stand-ups)* dan pemetaan kebutuhan pengguna *(user stories)*. Berdasarkan evaluasi operasional, penerapan kombinasi praktik Agile secara sistematis terbukti secara empiris memberikan dampak positif yang signifikan terhadap kualitas komunikasi tim, akurasi penentuan prioritas fitur, serta penyelesaian masalah manajemen kebutuhan rekayasa perangkat lunak [5].

Lebih jauh dalam ekosistem digital yang dinamis, efektivitas metode Agile telah tervalidasi melalui riset berbasis data berskala besar terhadap ribuan repositori proyek perangkat lunak publik [13]. Hasil analisis tersebut menegaskan bahwa proyek rekayasa perangkat lunak yang berorientasi pada prinsip Agile cenderung mendapatkan penilaian keberhasilan dan tingkat penerimaan yang lebih tinggi oleh komunitas pengembang dibandingkan dengan metodologi terencana yang kaku *(plan-driven)*. Karakteristik utama yang membedakan metode ini adalah kemampuannya dalam menjaga ketahanan sistem sambil terus menyerap perubahan spesifikasi *(refactoring)* di tengah berjalannya proyek [13]. Oleh karena itu, pendekatan pengembangan Agile sangat direkomendasikan sebagai landasan teoretis dan metodologi operasional dalam merancang dan membangun sistem informasi berskala bisnis, karena memastikan perangkat lunak yang dihasilkan responsif, fleksibel, dan memberikan nilai fungsionalitas yang terukur.

2.4 PHP

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman berbasis skrip sisi *server (server-side scripting)* yang dirancang secara spesifik untuk

membangun arsitektur sistem informasi web dinamis dan berskala penuh [14]. Sebagai mesin pemroses utama di balik antarmuka aplikasi bisnis, PHP beroperasi dengan menerima interaksi pengguna dari peramban, mengeksekusi logika komputasi operasional, dan merender dokumen akhir secara *real-time* sebelum dikirimkan kembali ke antarmuka klien. Kemampuannya yang tinggi dalam menjembatani pertukaran data kompleks, termasuk memfasilitasi pelacakan informasi bisnis secara paralel pada tingkat backend, menjadikannya fondasi infrastruktur yang sangat andal. Dalam ekosistem rekayasa perangkat lunak modern, penggunaan bahasa pemrograman terstruktur ini terbukti secara empiris mampu menjaga stabilitas *server* dengan kompatibilitas performa yang kuat terhadap berbagai standar teknologi kerangka kerja *(framework)* rekayasa perangkat lunak [14].

Lebih jauh, fungsi krusial PHP secara fundamental bertumpu pada keandalannya dalam mengeksekusi protokol relasional atau antarmuka pemograman *(API)* guna mengelola fungsionalitas *CRUD (Create, Read, Update, Delete)* pada arsitektur pangkalan data *(database)* yang kompleks [4]. Kinerja pemrosesan berbasis skrip *server* ini terbukti tangguh dalam memvalidasi logika batasan hak akses secara aman, contohnya, memisahkan otoritas eksekusi antara staf operasional, tenaga penjual, kasir, dan pemilik entitas sistem sebelum kueri diizinkan untuk mengubah data secara permanen. Ekosistem PHP yang solid bahkan sangat teruji kemampuannya dalam menopang beban lalu lintas kueri pada skema basis data relasional yang masif, termasuk menjaga konsistensi sinkronisasi rekaman yang melibatkan hingga 55 tabel master dan transaksi yang terikat secara terpusat [14], [4]. Melalui manajemen komputasi sisi *server* inilah, arsitektur web berbasis PHP mampu menekan potensi tabrakan data *(data redundancy)* dan memastikan sistem beroperasi secara skalabel, responsif, serta terhindar dari anomali pelaporan finansial maupun logistik harian.

2.5 *Framework* Laravel

Framework Laravel merupakan kerangka kerja pengembangan aplikasi web berbasis PHP yang bersifat sumber terbuka *(open-source)* dan mengadopsi pola arsitektur *Model-View-Controller (MVC)* untuk memisahkan logika bisnis,

manajemen data, dan antarmuka pengguna secara terstruktur [4]. Keunggulan utama Laravel terletak pada sintaksisnya yang ekspresif dan elegan, yang dirancang untuk menyederhanakan tugas-tugas kompleks dalam rekayasa perangkat lunak seperti autentikasi, perutean *(routing)*, pengelolaan sesi, dan pemetaan basis data melalui *Eloquent Object-Relational Mapping (ORM)*. Penggunaan *Eloquent* memungkinkan pengembang untuk berinteraksi dengan *database*, menggunakan sintaksis berorientasi objek yang intuitif, yang sangat efektif dalam mengelola integritas data pada sistem dengan skema tabel yang luas, termasuk mempermudah pemetaan relasi yang kompleks antara 55 tabel master dan transaksi operasional dalam sistem manajemen [4], [14].

Di sisi lain, Laravel menyediakan ekosistem pendukung yang kuat melalui antarmuka baris perintah Artisan *(CLI)* dan sistem *middleware* yang krusial untuk aspek keamanan serta kontrol akses sistem informasi [14]. Implementasi *middleware* dalam Laravel memungkinkan pengembang untuk membangun lapisan validasi keamanan yang berlapis, seperti proteksi terhadap serangan *Cross-Site Request Forgery (CSRF)* dan *SQL injection* secara otomatis. Lebih jauh, fleksibilitas *middleware* ini menjadi fondasi utama dalam menerapkan mekanisme *Role-Based Access Control (RBAC)* yang ketat, guna memastikan bahwa fungsionalitas kritis seperti modul penagihan *(invoice)*, pemesanan pembelian, dan rekonsiliasi stok *(stock opname)* hanya dapat diakses oleh peran pengguna yang sah seperti Admin, Kasir, Sales, atau Pemilik. Dengan dukungan mesin templat Blade yang efisien, Laravel tidak hanya mempercepat proses pengembangan tetapi juga memastikan perangkat lunak yang dihasilkan memiliki kode yang bersih, mudah dipelihara *(maintainable)*, dan memiliki skalabilitas tinggi untuk kebutuhan bisnis jangka panjang [4], [14]

2.6 MySQL

MySQL merupakan salah satu Sistem Manajemen Basis Data Relasional *(RDBMS)* berbasis sumber terbuka *(open-source)* yang paling luas diadopsi dalam arsitektur rekayasa perangkat lunak modern untuk menyimpan, mengelola, dan mengambil data secara terstruktur [9]. Secara teoretis, sistem ini beroperasi dengan merepresentasikan entitas data ke dalam format tabel relasional (baris dan kolom)

yang saling terikat menggunakan parameter kunci utama *(primary key)* dan kunci tamu *(foreign key)*. Dalam implementasi sistem informasi bisnis skala menengah hingga besar, keandalan fungsionalitas MySQL sangat esensial untuk mengamankan integritas struktural dan konsistensi aliran data yang kompleks. Sebagai contoh, mesin ini mampu mengelola arsitektur relasional terpusat yang melibatkan hingga 55 tabel master dan transaksi yang telah dirancang dengan presisi menggunakan visualisasi *Entity Relationship Diagram (ERD)*. Pemanfaatan relasi tabel yang terstandar ini menjamin bahwa seluruh riwayat pencatatan operasional mulai dari manajemen persediaan hingga faktur penjualan terintegrasi secara ketat guna menekan risiko redundansi maupun anomali data [9], [15].

Dari perspektif metrik komputasi, MySQL dirancang untuk mencapai ketahanan dan optimalisasi tinggi *(robustness)* dalam mengeksekusi operasi dasar *CRUD (Create, Read, Update, Delete)* serta menangani beban transaksi konkuren [15]. Kecepatan pemrosesan dan latensi kueri SQL yang rendah pada MySQL menjadikannya fondasi penyimpanan yang ideal untuk dikolaborasikan bersama kerangka kerja sistem sisi *server* yang dinamis, seperti Laravel. Melalui dukungan teknik pengindeksan *(indexing)* dan kontrol konkurensi bawaan, MySQL dapat secara konsisten mempertahankan waktu respons pemrosesan dalam hitungan milidetik meskipun sistem sedang merespons eksekusi manipulasi data paralel dari berbagai peran fungsional dalam sebuah organisasi bisnis [9]. Dengan demikian, MySQL tidak sekadar bertindak sebagai ruang penyimpanan pasif, melainkan sebuah instrumen vital yang mengamankan skalabilitas, reliabilitas pencatatan, dan kelancaran transaksi data *real-time* di seluruh modul aplikasi bisnis.

2.7 Tailwind CSS

Tailwind CSS secara teoretis adalah kerangka kerja *(framework) Cascading Style Sheets (CSS)* berbasis *utility-first* yang merombak paradigma desain antarmuka pengguna web secara fundamental [16]. Alih-alih mengharuskan pengembang untuk menulis kode *style* secara terpisah di dalam berkas *CSS* konvensional, Tailwind menyediakan sekumpulan kelas utilitas berukuran kecil (seperti *flex, text-center, p-4*) yang dapat disisipkan langsung ke dalam markah elemen HTML atau mesin templat. Pendekatan arsitektur ini secara signifikan

mempercepat proses penataan gaya *(styling)*, menekan lonjakan ukuran berkas *CSS*, dan membebaskan pengembang dari konflik penamaan kelas *(class name conflicts)* ketika struktur aplikasi mulai membesar [17]. Evaluasi perbandingan teknologi antarmuka juga membuktikan bahwa Tailwind CSS menawarkan fleksibilitas modifikasi desain yang jauh melebihi kerangka kerja *UI* tradisional, menjadikannya standar baru dalam rekayasa *front-end* modern [17].

Fleksibilitas berbasis utilitas ini menjadi sangat krusial ketika sistem informasi dituntut untuk mengelola ekosistem pangkalan data berskala luas. Dalam skenario pengembangan *server back-office* menggunakan kerangka kerja seperti Laravel, Tailwind CSS sangat ideal untuk menyusun antarmuka administrasi tanpa perlu meninggalkan lingkungan templat Blade. Hal ini memungkinkan visualisasi data yang kompleks termasuk merender rekapitulasi data dari struktur tabel relasional secara dinamis menjadi lebih efisien dan responsif di berbagai ukuran perangkat [16]. Lebih jauh lagi, sifat deklaratif Tailwind memastikan bahwa setiap perubahan tata letak alur kerja bisnis, yang umumnya dipetakan menjadi belasan bentuk rancangan *Activity Diagram*, dapat diimplementasikan ke dalam desain antarmuka *(User Interface)* dengan konsistensi visual dan performa memuat halaman yang optimal di sisi klien

2.8 *Framework Component* Flowbite

Flowbite merupakan pustaka komponen antarmuka pengguna *(UI component library)* mutakhir yang dibangun di atas fondasi kerangka kerja Tailwind CSS [18]. Secara konseptual, jika Tailwind menyediakan blok bangunan kasar *(utility classes)*, maka Flowbite menyajikan susunan komponen interaktif siap pakai yang telah dirakit sempurna seperti menu navigasi, blok *modal pop-up*, hingga tabel data interaktif *(DataTables)*. Pustaka ini dilengkapi dengan skrip JavaScript internal yang dikonfigurasi untuk menangani aksi interaktif tanpa mengharuskan pengembang merancang logika *DOM (Document Object Model)* dari nol. Implementasi Flowbite terbukti secara ilmiah tidak hanya mempertahankan konsistensi gaya visual antarmuka secara presisi di berbagai rupa platform, melainkan juga mempercepat secara drastis siklus pengembangan antarmuka untuk aplikasi pengelola data *(dashboard)* berskala besar.

Dalam konteks keamanan dan pengalaman pengguna sistem bisnis, Flowbite menawarkan elemen navigasi struktural yang sangat cocok untuk mengawal mekanisme pembatasan hak akses atau *Role-Based Access Control (RBAC)*. Komponen seperti sidebar dinamis dan dropdown menu dapat dikondisikan secara logis agar hanya merender fitur tertentu sesuai dengan tingkat otoritas penggunanya, guna menjaga batasan privasi fungsional yang ketat antara entitas Kasir, Staff Sales, dan Pemilik. Selain itu, ketika aplikasi dihadapkan pada tugas untuk menayangkan ribuan baris data inventaris untuk kebutuhan verifikasi *stock opname* atau penagihan penjualan, komponen tabel dari Flowbite memastikan proses pengurutan *(sorting)*, penyaringan *(filtering)*, dan paginasi berjalan sangat mulus [18]. Penggabungan fundamental fungsionalitas ini menjadikan Flowbite instrumen esensial dalam menerjemahkan rancang bangun logika sistem yang rumit menjadi antarmuka visual yang sangat interaktif dan mudah dipahami oleh pengguna operasional.

2.9 Teknologi Keuangan *(FinTech)*

Teknologi Keuangan atau *FinTech* umumnya merupakan hasil dari konvergensi antara layanan keuangan konvensional dengan inovasi teknologi informasi yang bertujuan untuk meningkatkan efisiensi, aksesibilitas, dan transparansi proses finansial [7]. Dalam spektrum yang lebih luas, *fintech* tidak hanya mencakup layanan perbankan digital, tetapi juga mencakup otomasi sistem akuntansi, manajemen aset, dan pemrosesan data keuangan yang terintegrasi secara *real-time* di dalam sistem informasi manajemen perusahaan. Secara empiris, adopsi *fintech* dalam skala operasional bisnis terbukti mampu mentransformasi model bisnis tradisional menjadi ekosistem digital yang lebih adaptif, di mana pengambilan keputusan strategis didorong oleh validitas data transaksi yang akurat dan otomatis [7], [19].

Penerapan *fintech* dalam sebuah sistem manajemen terpadu memainkan peran krusial dalam menyederhanakan alur kerja keuangan yang kompleks, terutama dalam meminimalkan ketergantungan pada proses manual yang rentan terhadap kesalahan manusia *(human error)* [19]. Melalui arsitektur sistem yang solid, teknologi ini memungkinkan pemrosesan ribuan data transaksi keuangan

yang melibatkan puluhan tabel basis data secara sinkron, guna menghasilkan laporan pertanggungjawaban yang transparan bagi pemangku kepentingan. Selain itu, *FinTech* memfasilitasi integrasi berbagai instrumen manajemen, mulai dari pemantauan tagihan penjualan *(sales invoice)* hingga rekonsiliasi pengeluaran operasional armada, yang pada akhirnya meningkatkan skalabilitas dan ketahanan finansial entitas bisnis di tengah persaingan pasar yang dinamis [7], [19].

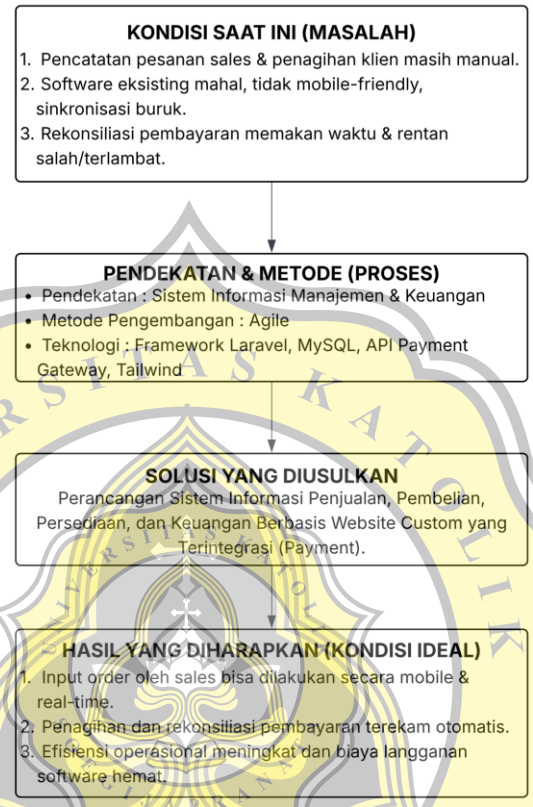
2.10 Gerbang Pembayaran *(Payment Gateway)*

Payment Gateway merupakan infrastruktur teknologi berbasis perangkat lunak yang berfungsi sebagai mediator aman untuk mengotorisasi dan memproses transaksi pembayaran antara aplikasi pedagang *(merchant)* dengan lembaga keuangan atau penyedia layanan pembayaran [8]. Secara teknis, sistem ini bekerja dengan mengenkripsi data sensitif transaksi seperti rincian kartu atau kredensial dompet digital untuk memastikan bahwa informasi keuangan tersebut terlindungi dari ancaman siber selama proses transmisi data. Keberadaan *payment gateway* dalam arsitektur web memungkinkan sistem untuk menerima berbagai metode pembayaran secara otomatis, yang secara langsung meningkatkan efisiensi penagihan dan mengurangi siklus piutang usaha melalui sinkronisasi status pembayaran yang instan [8], [10].

Dalam implementasi rekayasa perangkat lunak, integrasi *payment gateway* harus didukung oleh mekanisme keamanan yang ketat, termasuk penerapan protokol autentikasi ganda dan kontrol akses berbasis peran *(Role-Based Access Control)* untuk mengamankan data transaksi pada tingkat basis data [10]. Penggunaan teknologi ini sangat efektif untuk memvalidasi aliran kas pada modul penjualan secara otomatis, sehingga setiap faktur yang terbayar dapat langsung memperbarui rekaman stok gudang dan laporan laba rugi tanpa intervensi manual yang berulang. Dengan demikian *payment gateway* tidak hanya bertindak sebagai alat penerima pembayaran, tetapi juga sebagai lapisan keamanan krusial yang menjamin integritas transaksi, meningkatkan kepercayaan pengguna, dan mempercepat pertumbuhan ekosistem ekonomi digital secara keseluruhan [8], [10]

BAB III
METODE PENELITIAN

3.1 Kerangka Pikiran



Gambar 3. 1 Kerangka Pikiran

3.2 Desain Penelitian

Penelitian ini menggunakan pendekatan *Research and Development (R&D)* dengan metode Agile untuk mengembangkan sistem informasi keuangan dan aset berbasis web. Desain ini dipilih karena sesuai dengan kebutuhan UD. Trois Dinamika Sentosa yang memerlukan sistem kustom dengan penyesuaian cepat berdasarkan umpan balik pengguna [11]. Tahapan penelitian meliputi:

a) Analisis Kebutuhan:

Analisis kebutuhan akan dilakukan dengan cara wawancara dengan pemilik, admin, kasir, dan tim sales UD. Trois Dinamika Sentosa

b) Perancangan Sistem:

Merancang arsitektur sistem, antarmuka pengguna, dan basis data terintegrasi.

c) Pengembangan:

Implementasi menggunakan *framework* Laravel dengan pengembangan Agile (setiap iterasi berfokus pada fitur spesifik, seperti modul pembelian barang, penjualan barang, pelacakan stok, dan keuangan) [11], [5], [4].

d) Pengujian:

Melakukan pengujian berdasarkan *User Acceptance Test (UAT)* berbasis wawancara kualitatif untuk mengetahui penerimaan pengguna terhadap sistem.

e) Evaluasi:

Mengukur efektivitas sistem melalui parameter akurasi data, waktu proses, dan kepuasan pengguna.

3.3 Jenis Penelitian

Penelitian ini termasuk penelitian terapan dengan pendekatan kualitatif deskriptif. Pendekatan kualitatif digunakan untuk memahami fenomena proses bisnis, menganalisis kebutuhan sistem, dan mengevaluasi penerimaan sistem melalui wawancara dan observasi langsung:

a) Kualitatif:

Analisis kebutuhan pengguna melalui wawancara dengan tim UD. Trois Dinamika Sentosa.

3.4 Populasi dan Sampel

a) Populasi:

Seluruh karyawan UD. Trois Dinamika Sentosa yang terlibat dalam pengelolaan keuangan, aset, pembelian, dan penjualan.

b) Sampel

a. Purposive Sampling:

Penentuan sampel dilakukan dengan teknik *Purposive Sampling* berdasarkan subjek yang berinteraksi langsung dengan sistem, yaitu 4 orang (pemilik, admin, kasir, dan sales).

3.5 Teknik Pengumpulan Data

3.5.1 Wawancara Semi Terstruktur

Mengidentifikasi kebutuhan sistem, kelemahan *software* akuntansi, dan harapan pengguna.

3.5.2 Observasi Partisipatif

Memantau proses manual pengelolaan keuangan dan stok di UD. Trois Dinamika Sentosa untuk mengukur waktu dan kesalahan operasional.

3.5.3 Dokumentasi

Analisis dokumen keuangan (laporan laba rugi dan buku besar) dan catatan stok barang selama 3 bulan terakhir.

3.5.4 Pengujian Sistem

a. Uji fungsionalitas modul (contoh: *input* transaksi penjualan, *input* transaksi pembelian, dan penghitungan keuangan) dengan skenario kasus uji.

3.5.5 Wawancara Evaluasi Sistem *(User Acceptance Test)*

Mengukur tingkat penerimaan, kemudahan penggunaan *(usability)*, dan kepuasan pengguna terhadap sistem yang telah dibangun. Pengumpulan data dilakukan secara kualitatif melalui wawancara mendalam *(in-depth interview)* dan observasi langsung saat pengguna menguji coba sistem.

BAB IV
PEMBAHASAN DAN HASIL PENELITIAN

4.1 Analisis Sistem yang Berjalan

Saat ini, proses bisnis utama di UD. Trois Dinamika Sentosa (TDS) yang meliputi manajemen arus kas, pembelian, dan penjualan telah didukung oleh *software* akuntansi atau SAP. Namun, dalam praktiknya, sistem ini masih dioperasikan berdampingan dengan proses manual di lapangan, sehingga menimbulkan berbagai kendala operasional. Untuk memahami mekanisme kerja perusahaan saat ini, berikut adalah aktor yang terlibat beserta alur sistem yang berjalan.

4.1.1 Alur Prosedur yang Berjalan

Adapun prosedur operasional yang berjalan di UD. Trois Dinamika Sentosa saat ini dapat diuraikan sebagai berikut:

1. **Proses Pemesanan *(Order)***:
   Tim sales melakukan kunjungan rutin ke lokasi klien berdasarkan pembagian rute (misalnya rute timur dan barat). Klien menyampaikan pesanan, kemudian tim sales mencatat pesanan tersebut secara manual di dalam buku catatan pesanan.
2. **Proses Pengadaan Barang *(Purchasing)*:**
   Karena perusahaan menerapkan sistem stok *by order* (meskipun memiliki gudang penyimpanan), daftar pesanan dari buku tim sales diserahkan kepada pemilik. Kemudian pemilik melakukan pemesanan barang kepada *supplier* secara manual (melalui pesan *WhatsApp*).
3. **Proses Pencatatan dan Pengiriman:**
   Setelah barang dari *supplier* tiba di gudang, kasir akan memasukkan data barang masuk *(purchase order)* ke dalam software untuk menambah stok. Selanjutnya, kasir memasukkan pesanan klien dari buku catatan tim sales untuk menerbitkan faktur penjualan *(invoice)*. Setelah itu, barang dikemas dan dikirim

kepada klien melalui pihak ketiga (ekspedisi) atau dititipkan melalui tim sales yang rutenya searah.

4. **Proses Penagihan dan Pembayaran**:

   Pada minggu berikutnya, tim sales akan kembali mengunjungi klien untuk melakukan penagihan berdasarkan *invoice* yang telah diterbitkan dan masa jatuh temponya. Hasil penagihan ini kemudian diserahkan kembali kepada admin untuk dicatat penyelesaiannya di dalam *software SAP*.

4.1.2 Analisis Kelemahan Sistem yang Berjalan

Berdasarkan alur operasional di atas, ditemukan beberapa kelemahan pada *software* akuntansi eksisting dan prosedur manual yang berjalan, antara lain:

1. **Keterlambatan Sinkronisasi Data:**

   Pencatatan manual oleh tim sales mengakibatkan admin kasir harus menunggu tim sales kembali ke kantor untuk memproses pesanan. Hal ini memakan waktu lama, terutama jika pesanan berjumlah besar, yang akan menjadi kendala.

2. **Keterbatasan Manajemen Cabang (*Branching*):**

   Software lama tidak mendukung pemisahan pencatatan (*segregation*) antarsales. Karena setiap sales memiliki target klien dan nota faktur *supplier* yang berbeda-beda, perusahaan terpaksa membuat *file database* terpisah untuk masing-masing sales. Hal ini menghilangkan kemampuan pemilik/ admin untuk melakukan pemantauan data secara global dan terpusat.

3. **Kendala Aksesibilitas dan Jaringan:**

   Software lama hanya dapat diakses melalui komputer desktop di kantor *(on-premises)*. Selain itu, sering terjadi gangguan koneksi di komputer *server* lokal yang menghambat produktivitas admin kasir dan memerlukan waktu pemulihan.

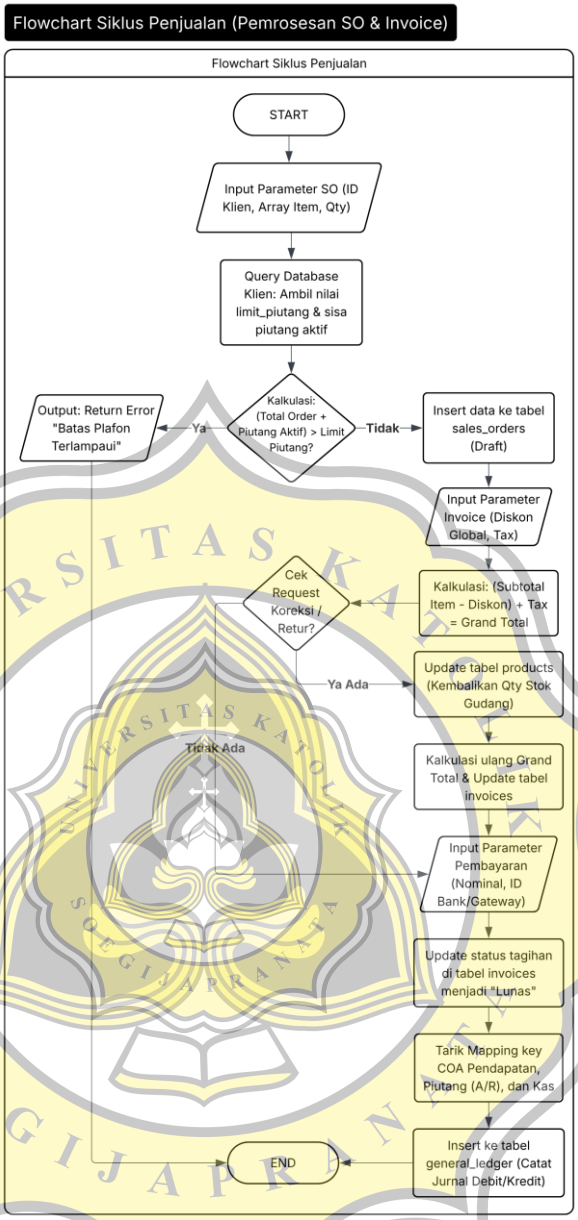
4. **Beban Biaya Fluktuatif:**

Software yang digunakan memerlukan biaya langganan dalam mata uang Dolar AS, sehingga nominal pengeluaran perusahaan menjadi fluktuatif mengikuti nilai tukar kurs.

4.2 Perancangan Sistem

Pada tahap perancangan sistem ini, peneliti memodelkan alur kerja sistem informasi yang diusulkan melalui visualisasi *flowchart*. Perancangan ini bertujuan untuk memberikan gambaran logika secara rinci mengenai interaksi antara pengguna dengan sistem dalam setiap proses bisnis utama di UD. Trois Dinamika Sentosa (TDS).

4.2.1 *Flowchart*

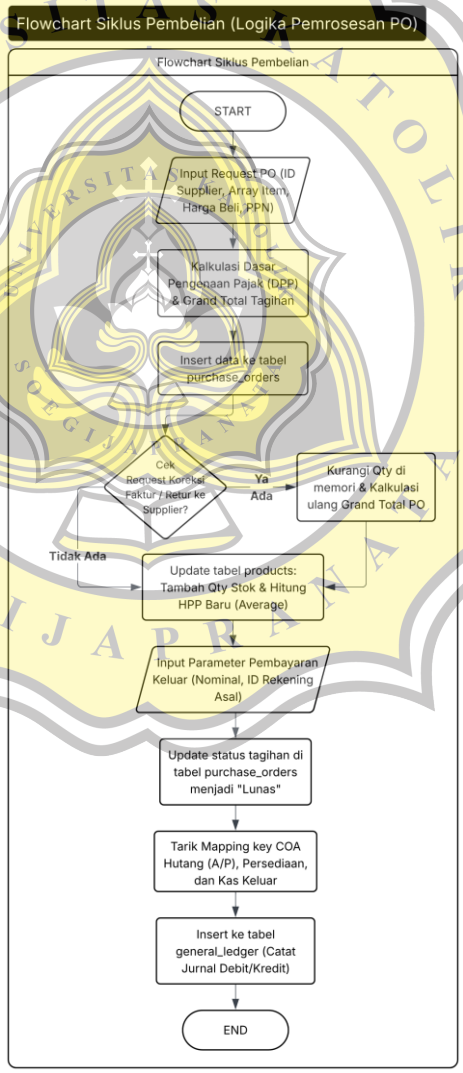


Gambar 4. 1 *Flowchart* Sistem Penjualan *(SO dan Invoice)*

Gambar 4.1 merupakan flowchart yang menggambarkan siklus penjualan terintegrasi, mulai dari pencatatan pesanan hingga penerimaan pembayaran. Proses menggunakan empat jalur *(swimlanes)*: Staf *Sales*, Kasir, Admin/*Owner*, dan Sistem. Alur dimulai ketika staf sales membuka modul *Sales Order (SO)* dan memasukkan detail pesanan klien. Secara otomatis, sistem melakukan validasi terhadap plafon piutang klien. Apabila plafon telah mencapai batas maksimal, sistem akan menampilkan

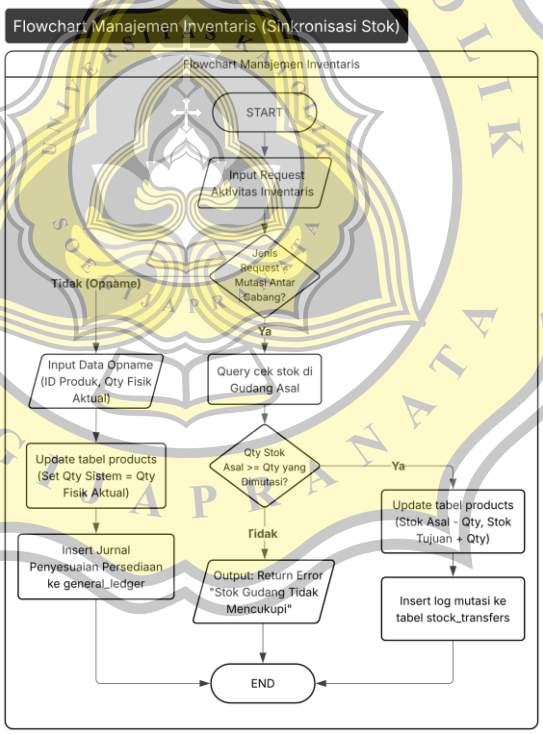
peringatan dan menolak pesanan. Jika aman, kasir akan menarik data dari modul *Sales Order (SO)* tersebut menjadi draf *Sales Invoice* untuk di*input* diskon global dan diterbitkan menjadi nota resmi.

Nota fisik kemudian dibawa oleh staf sales untuk proses penagihan. Jika terdapat ketidaksesuaian barang atau retur, kasir akan melakukan input pada modul *Invoice Adjustment* atau *Sales Return* yang secara otomatis akan mengoreksi stok gudang dan nilai tagihan. Siklus berakhir ketika Admin atau Pemilik melakukan pencatatan pembayaran dengan memilih metode pembayaran dan rekening bank, di mana sistem akan langsung melakukan pemetaan jurnal ke akun Piutang Usaha dan Kas/Bank pada Buku Besar.



Gambar 4. 2 *Flowchart* Sistem Pembelian/ *Purchase Order*

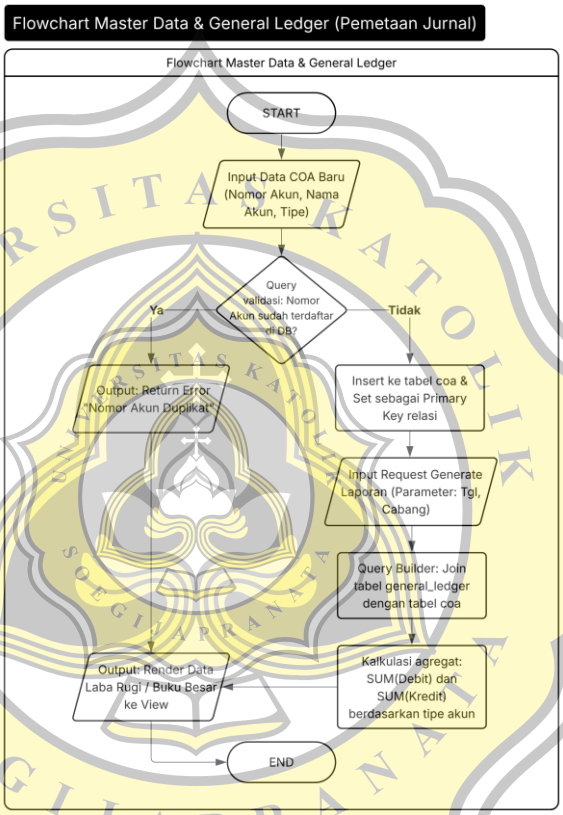
Gambar 4.2 menjelaskan prosedur pengadaan barang yang dimulai dari *owner* melakukan kontak melalui whatsapp ke s*upplier*. Alur sistem dimulai pada jalur kasir saat barang dan faktur fisik tiba di gudang. Kasir memasukkan draf *Purchase Order (PO)* dengan rincian item, diskon bertingkat, PPN, dan DPP sesuai faktur *supplier*. Setelah barang dikonfirmasi masuk, sistem akan memperbarui stok dan menghitung Harga Pokok Penjualan (HPP) menggunakan metode *average*. Terdapat titik keputusan pada Kasir untuk memeriksa kesesuaian faktur, jika ditemukan salah tagih oleh *supplier* atau barang cacat. Kasir melakukan *PO Adjustment* atau *Purchase Return*. Setelah data valid, admin atau *owner* melakukan pencatatan pembayaran. Sistem kemudian menutup siklus dengan memperbarui akun Hutang Dagang dan Persediaan Barang pada modul akuntansi.



Gambar 4. 3 *Flowchart* Sistem Manajemen Inventaris

Gambar 4.3 menunjukkan tata kelola stok yang melibatkan dua aktivitas utama, yaitu *Stock Opname* dan *Stock Transfer*. Proses dimulai dengan keputusan Admin atau *Owner* untuk memilih jenis aktivitas gudang. Jika memilih *Stock Opname*, tugas dialihkan ke jalur kasir untuk melakukan

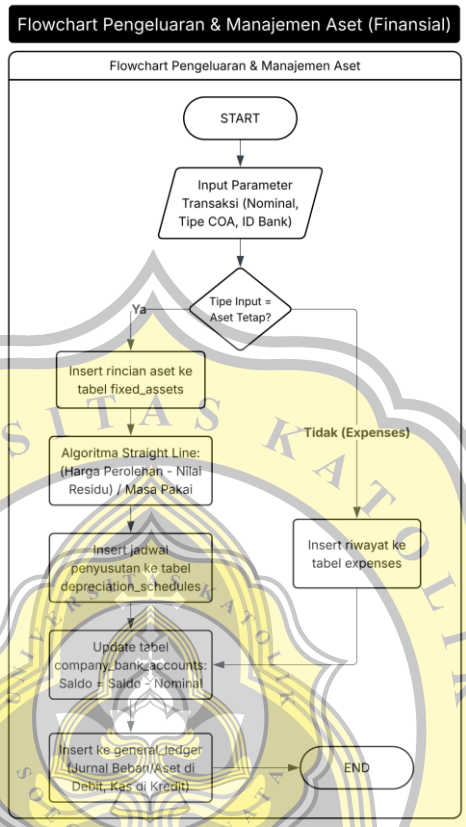
recheck antara fisik barang dan data sistem. Kasir memasukkan angka penyesuaian (positif atau negatif), lalu sistem akan menyelaraskan kuantitas produk serta mencatat jurnal penyesuaian stok. Jika memilih *Stock Transfer*, admin akan memasukkan cabang asal dan tujuan beserta jumlah barang yang dimutasi. Sistem kemudian memperbarui posisi stok di masing-masing gudang cabang secara *real-time* untuk memastikan distribusi barang tetap terpantau secara tersentralisasi.



Gambar 4. 4 *Flowchart* Sistem Informasi Penjualan Berbasis Teknologi Keuangan

Gambar 4.4 merupakan alur pemetaan sentral yang menjadi pondasi sistem. Dimulai oleh *Owner* atau admin dengan melakukan konfigurasi pada modul *Chart of Accounts (COA)* untuk menentukan hierarki akun induk dan sub-akun. Selanjutnya, dilakukan pemetaan pada modul Setting untuk menghubungkan setiap jenis transaksi (seperti penjualan, hpp, atau pajak) ke nomor akun yang tepat. Setelah parameter akun siap, seluruh *data master* seperti produk, cabang, dan klien dimasukkan. Hasil dari seluruh aktivitas transaksional pada modul lain akan bermuara di sini, di mana sistem

melakukan kalkulasi data secara otomatis untuk menyajikan laporan buku besar, laporan laba rugi, dan keuangan yang dapat diakses oleh *Owner* kapan saja.



Gambar 4. 5 *Flowchart* Sistem Informasi Penjualan Berbasis Teknologi Keuangan

Gambar 4.5 menggambarkan pengelolaan finansial internal perusahaan. Admin atau *Owner* dapat memilih antara pencatatan beban dan aset. Pada modul *Expenses*, pengguna memasukkan detail pengeluaran dan melakukan *mapping COA* untuk menentukan kategori beban serta sumber dana bank yang digunakan. Sementara pada modul *Fixed Assets*, pengguna memasukkan data perolehan aset, nilai residu, dan masa pakai. Sistem secara otomatis menerapkan metode penyusutan *Straight Line* dan menjadwalkan jurnal penyusutan bulanan. Kedua proses ini akan langsung memotong saldo rekening bank perusahaan dan tercatat dalam laporan arus kas serta laporan keuangan.

4.2.2 *Use Case Diagram*

Pada Gambar 4.6 dibawah merupakan *Use case Diagram* dari Sistem Informasi Penjualan Berbasis Teknologi Keuangan di UD. Trois Dinamika Sentosa. Diagram ini menggambarkan interaksi fungsional antara pengguna (aktor) dengan sistem yang dibangun, di mana batasan ruang lingkup berfokus pada fungsionalitas Portal admin dan Portal Publik. Terdapat empat aktor utama internal, satu aktor publik, dan dua aktor sistem eksternal yang terlibat.

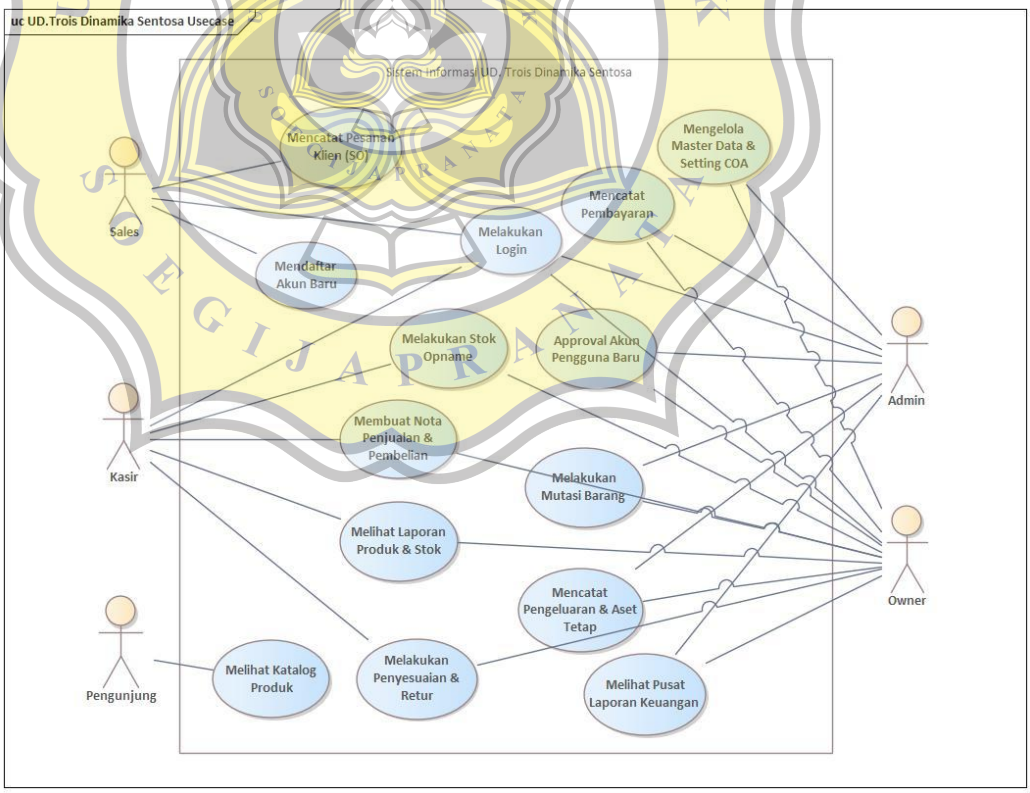
Proses interaksi dimulai dari aktor Pengunjung yang mengakses sistem melalui Portal Publik. Pengunjung memiliki hak akses terbatas yang hanya memungkinkan mereka untuk melakukan *use case* Melihat Katalog Produk tanpa perlu melewati proses autentikasi. Untuk mengakses Portal admin, setiap aktor internal wajib melakukan *use case* Melakukan Login. Sistem menyediakan integrasi *Single Sign-On (SSO)*, sehingga *use case* Login dapat berinteraksi langsung dengan aktor eksternal, yaitu Sistem Google. Bagi pengguna yang baru mendaftar, akun tersebut masuk ke status tunda *(pending)* dan membutuhkan approval dari admin.

Setelah berhasil masuk ke dalam sistem, pembagian hak akses *(Role and Permission)* berlaku sebagai berikut:

1. Staff *Sales*: Merupakan aktor yang bertugas di lapangan. Aktor ini hanya berinteraksi dengan sistem untuk *use case* Mencatat Pesanan Klien (*Sales* Order), yang mana datanya akan diteruskan ke bagian kasir.
2. Kasir: Bertanggung jawab atas sirkulasi produk *(Input/Output)*. Kasir berinteraksi dengan sistem untuk membuat Nota Penjualan, mencatat faktur melalui Nota Pembelian, serta mengelola *use case* Penyesuaian (Adjustment) dan Retur jika terjadi kesalahan. Selain itu, kasir melakukan sinkronisasi data gudang melalui *use case* Stok Opname dan dapat memantau pergerakan barang melalui Laporan Kartu Stok dan Riwayat Produk.

3. Admin: Bertindak seperti tangan kanan pemilik perusahaan dengan otoritas level manajerial. admin mengelola *use case* krusial seperti pengaturan Master Data, Konfigurasi Perusahaan, serta Pemetaan Jurnal Akuntansi (*Chart of Accounts/COA)*. Terkait arus kas, admin adalah satu-satunya entitas (selain *Owner*) yang dapat mengakses *use case* Pencatatan Pembayaran, Pencatatan Pengeluaran Operasional, Manajemen Aset Tetap, dan Mutasi Barang antarcabang. Pada proses pembayaran yang menggunakan metode otomatis, sistem berinteraksi dengan aktor eksternal, Sistem Midtrans, sebagai *Payment Gateway*.

4. *Owner*: Merupakan aktor dengan hak akses tertinggi. Dalam diagram ini, *Owner* dihubungkan dengan relasi generalisasi terhadap admin dan Kasir, yang mengindikasikan bahwa *Owner* memiliki akses menyeluruh ke semua modul transaksional, operasional, hingga memantau Pusat Laporan Keuangan secara komprehensif.

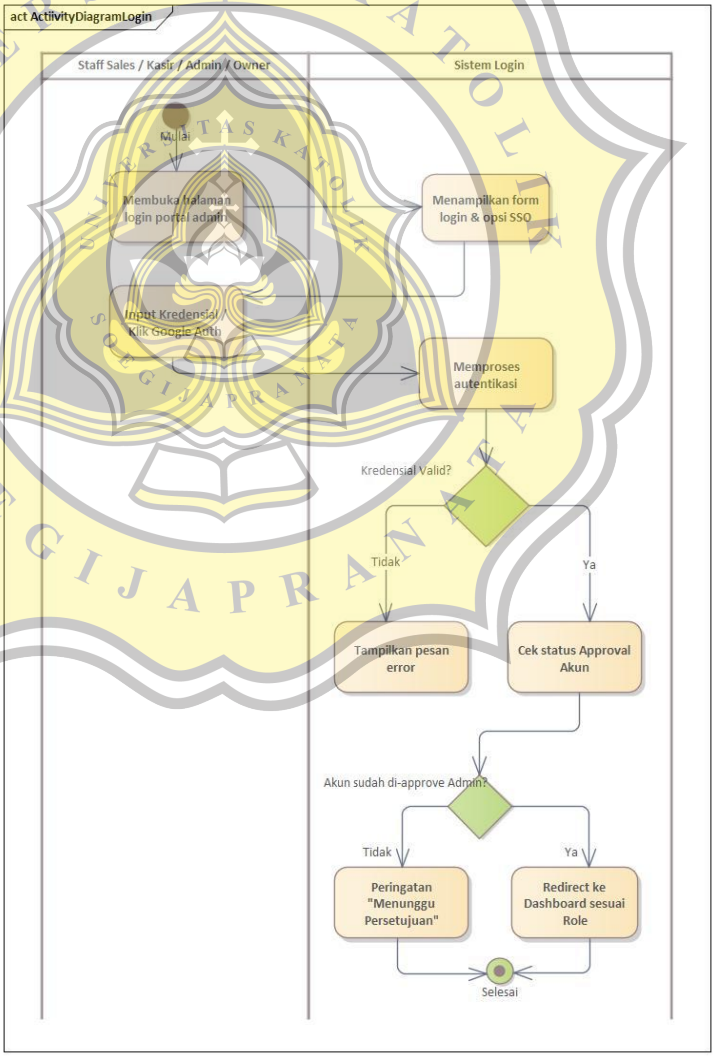


Gambar 4. 6 *Use Case Diagram*

4.2.3 *Activity Diagram*

a. *Activity Diagram* Melakukan Login

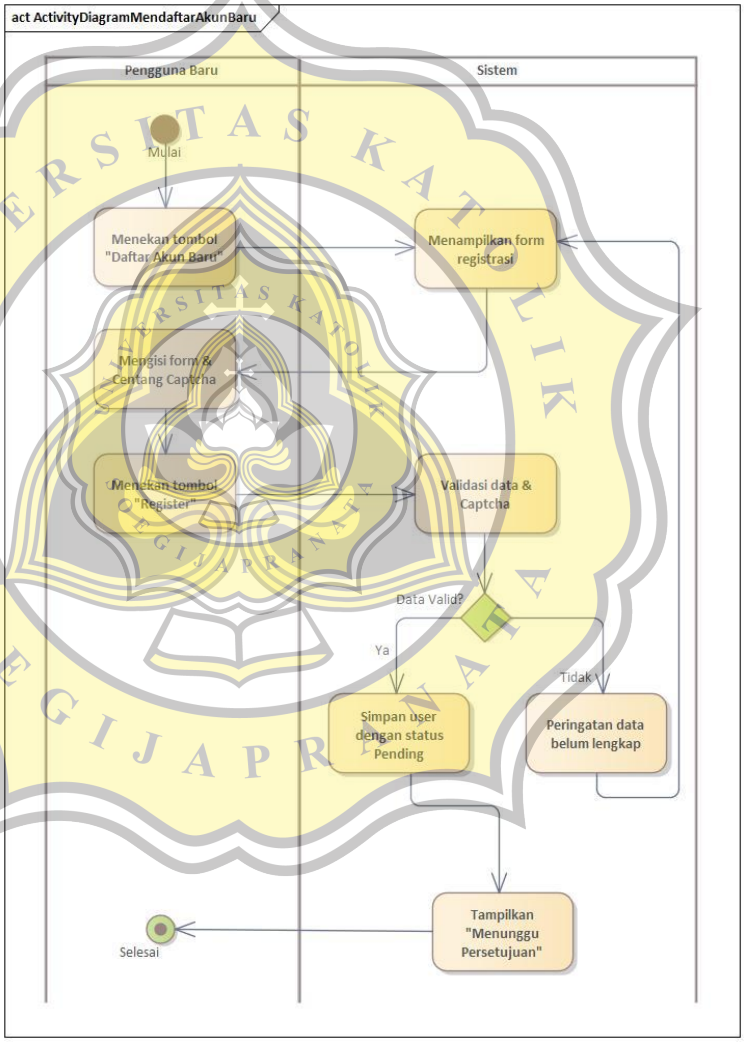
Pada Gambar 4.7 menjelaskan alur kerja proses login. Pengguna membuka halaman login dan memilih metode autentikasi, baik melalui pengisian kredensial secara manual maupun melalui integrasi Single Sign-On (SSO) Google. Sistem memvalidasi data tersebut. Jika kredensial tidak valid, sistem menampilkan pesan galat. Jika valid, sistem memeriksa status approval akun. Pengguna yang telah disetujui akan diarahkan menuju dashboard sesuai dengan hak akses (role) masing-masing, sedangkan akun yang belum disetujui akan menerima peringatan penundaan.



Gambar 4. 7 *Activity Diagram* Melakukan Login

b. *Activity Diagram* Mendaftar Akun Baru

Pada Gambar 4.8 menggambarkan tahapan pendaftaran akun baru oleh pengguna. Proses dimulai dengan mengakses form registrasi dan mengisi data diri beserta verifikasi captcha. Sistem memvalidasi kelengkapan data. Apabila terdapat kekurangan, pengguna diminta untuk melengkapinya. Jika data valid, sistem menyimpan informasi pengguna dengan status pending dan menampilkan pesan bahwa akun sedang menunggu persetujuan dari pihak administrator.

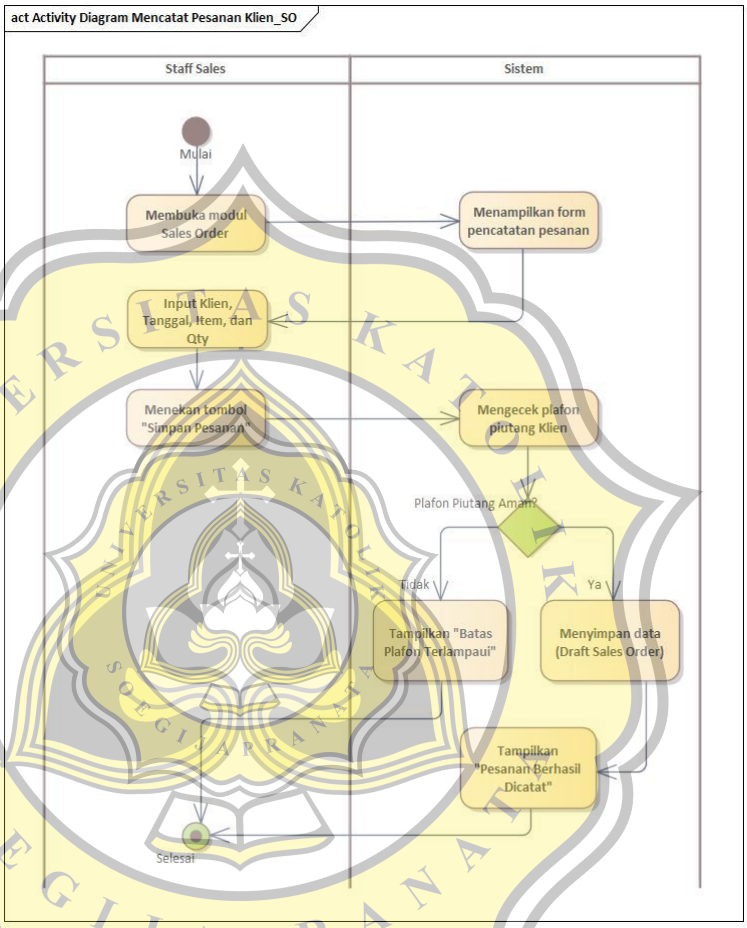


Gambar 4. 8 *Activity Diagram* Mendaftar Akun Baru

c. *Activity Diagram* Mencatat Pesanan Klien *(Sales Order)*

Pada Gambar 4.9 menguraikan prosedur pencatatan pesanan klien oleh Staff Sales. Pengguna menginput detail pesanan ke dalam

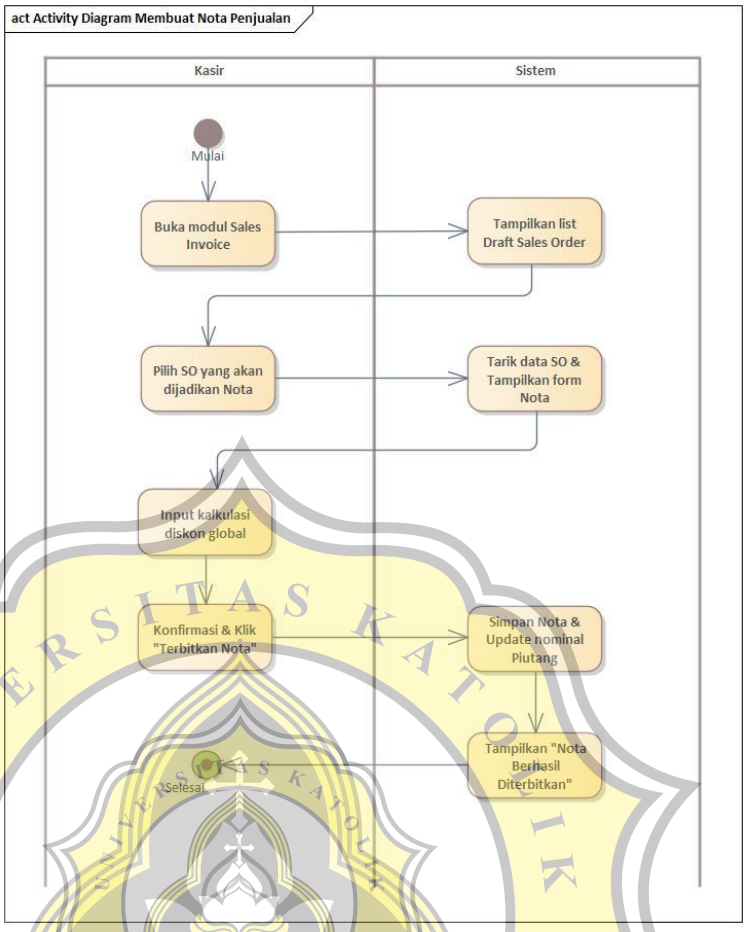
modul Sales Order. Sistem kemudian memvalidasi batas plafon piutang klien terkait. Jika pesanan melebihi batas plafon, sistem akan menolak transaksi dan menampilkan peringatan. Sebaliknya, jika plafon mencukupi, sistem menyimpan data sebagai draft pesanan dan memberikan konfirmasi keberhasilan.



Gambar 4. 9 *Activity Diagram* Mencatat Pesanan Klien (Sales Order)

d. *Activity Diagram* Membuat Nota Penjualan *(Sales Invoice)*

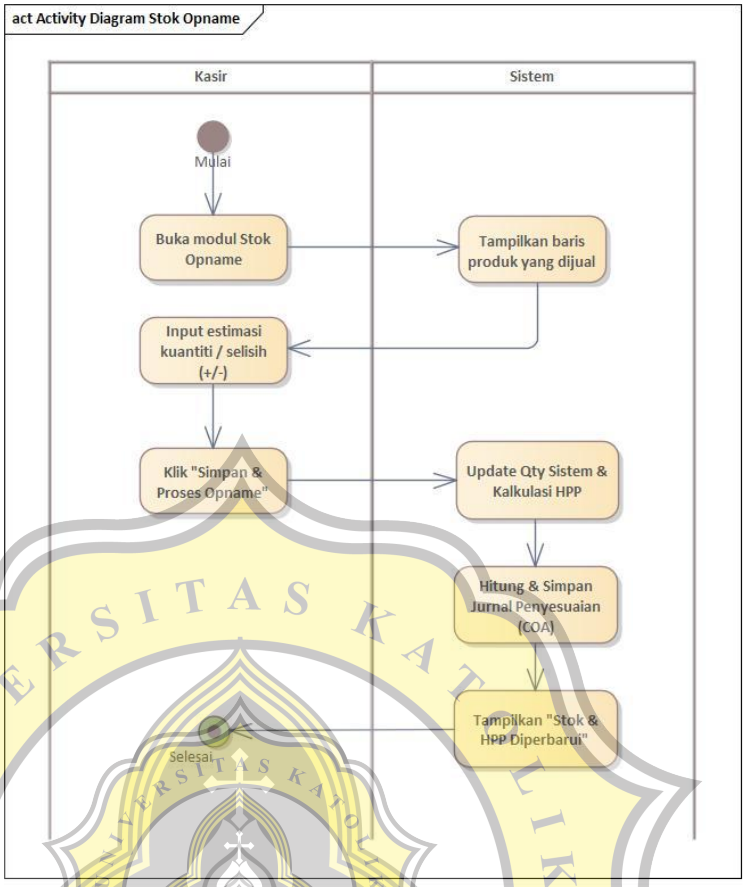
Pada Gambar 4.10 menunjukkan proses penerbitan nota penjualan oleh Kasir. Alur berlanjut dari penarikan data draft Sales Order ke dalam form Sales Invoice. Kasir melakukan penyesuaian akhir, seperti penginputan diskon global, lalu mengonfirmasi penerbitan nota. Sistem secara otomatis menyimpan nota, memperbarui nominal piutang klien, dan menampilkan pesan keberhasilan.



Gambar 4. 10 *Activity Diagram* Membuat Nota Penjualan (Sales Invoice)

e. *Activity Diagram* Melakukan Stok Opname

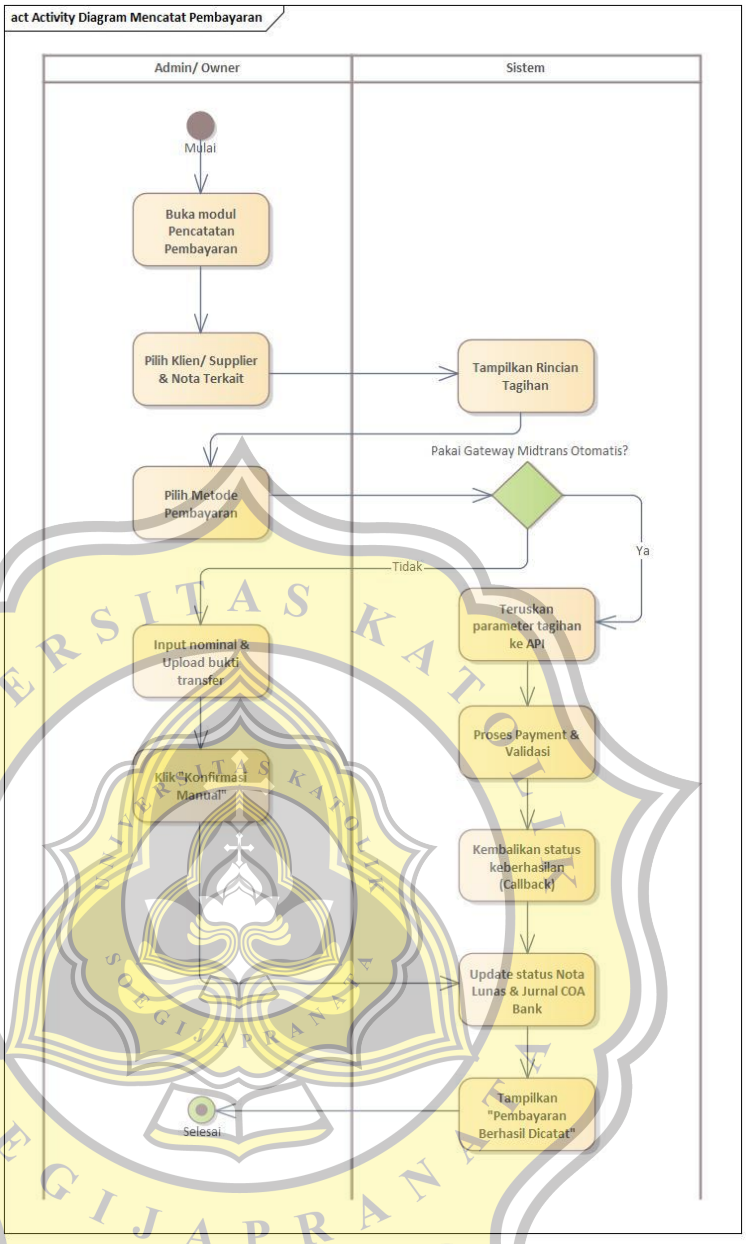
Pada Gambar 4.11 menjelaskan prosedur penyesuaian fisik barang yang dilakukan oleh Kasir. Pengguna memasukkan estimasi kuantitas atau selisih barang pada modul stok opname. Sistem memproses data tersebut untuk memperbarui kuantitas produk, mengalkulasi ulang Harga Pokok Penjualan (HPP), serta mencatat jurnal penyesuaian pada Chart of Accounts (COA) secara otomatis.



Gambar 4. 11 *Activity Diagram* Melakukan Stok Opname

f. *Activity Diagram* Mencatat Pembayaran

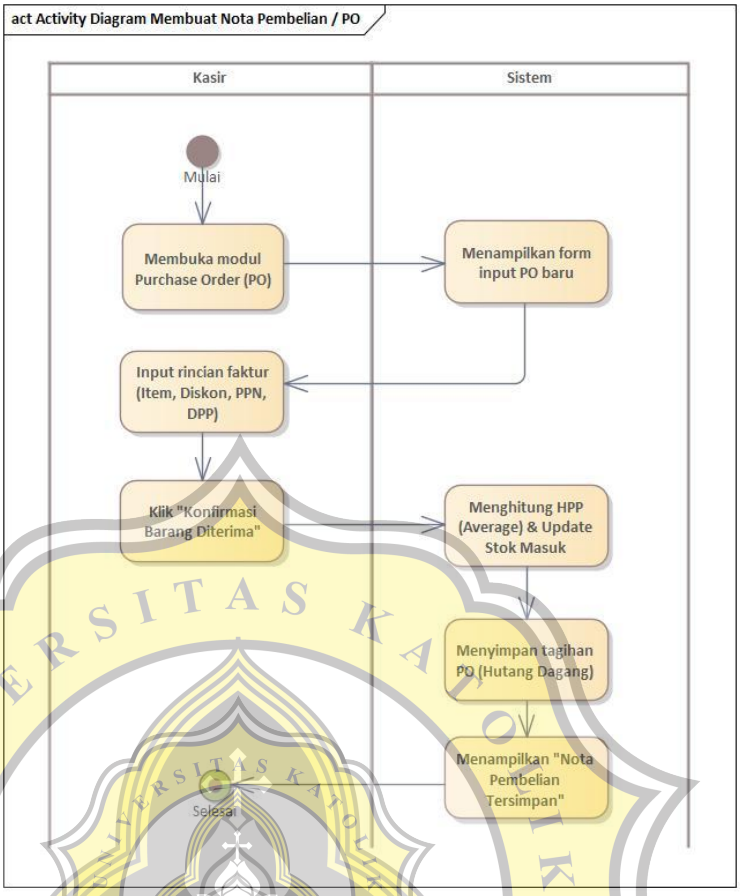
Pada Gambar 4.12 menjabarkan alur pencatatan pembayaran klien atau pemasok oleh admin. Terdapat dua metode penyelesaian: otomatis dan manual. Jika menggunakan payment gateway (Midtrans), sistem meneruskan parameter tagihan ke antarmuka pihak ketiga untuk diproses hingga mengembalikan status keberhasilan. Jika manual, admin menginput nominal dan bukti transfer. Sistem kemudian memperbarui status nota menjadi lunas dan mencatat jurnal pada buku besar.



Gambar 4. 12 *Activity Diagram* Mencatat Pembayaran

g. *Activity Diagram* Membuat Nota Pembelian *(Purchase Order)*

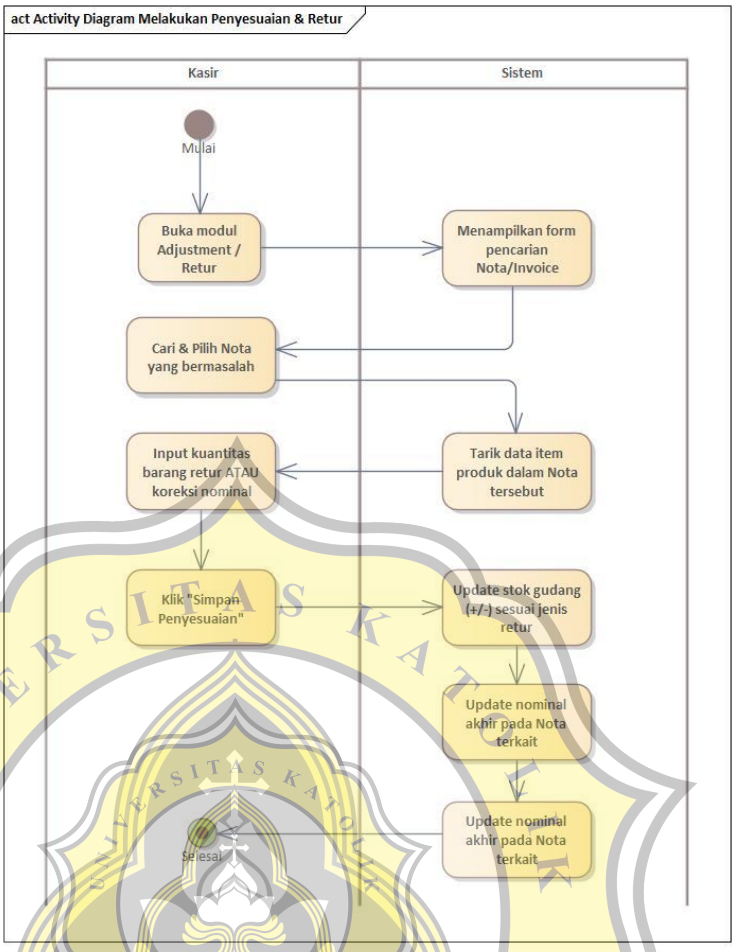
Pada Gambar 4.13 memvisualisasikan pencatatan faktur dari pemasok (supplier) oleh Kasir. Pengguna menginput rincian item, diskon, Pajak Pertambahan Nilai (PPN), dan Dasar Pengenaan Pajak (DPP) ke dalam modul Purchase Order. Setelah barang dikonfirmasi diterima, sistem menghitung HPP menggunakan metode Average, menambah stok gudang, dan menyimpan tagihan sebagai hutang dagang.



Gambar 4. 13 *Activity Diagram* Membuat Nota Pembelian (Purchase Order)

h. *Activity Diagram* Melakukan Penyesuaian & Retur

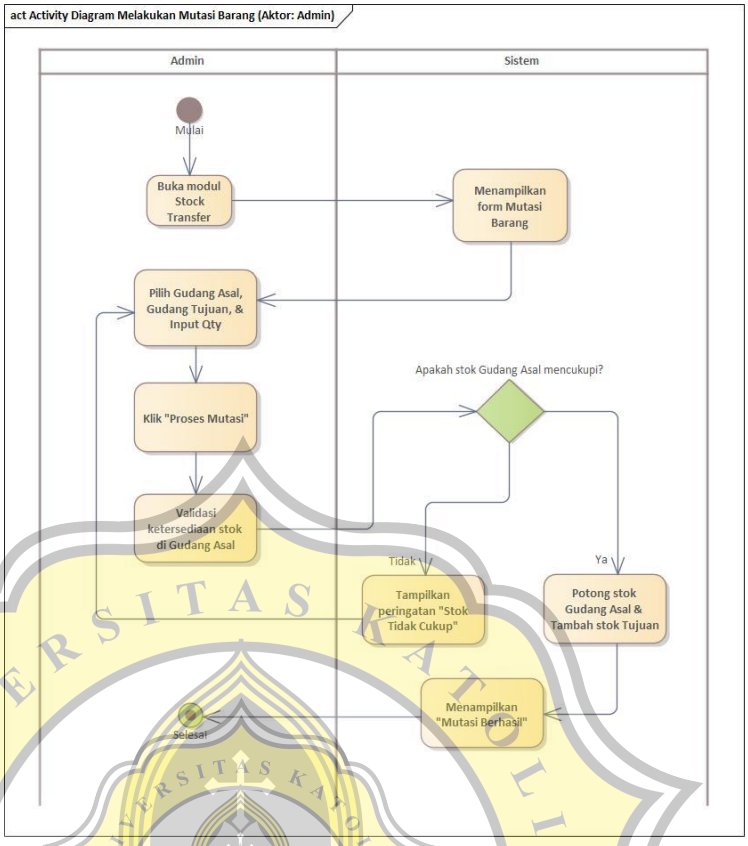
Pada Gambar 4.14 mendeskripsikan langkah-langkah koreksi dan pengembalian barang (retur) oleh Kasir. Pengguna mencari nota yang bermasalah, lalu menginput kuantitas barang retur atau koreksi nominal tagihan. Sistem merespons dengan memperbarui stok gudang, menyesuaikan nominal akhir pada nota, dan membuat jurnal koreksi di buku besar.



Gambar 4. 14 *Activity Diagram* Melakukan Penyesuaian & Retur

i. *Activity Diagram* Melakukan Mutasi Barang *(Stock Transfer)*

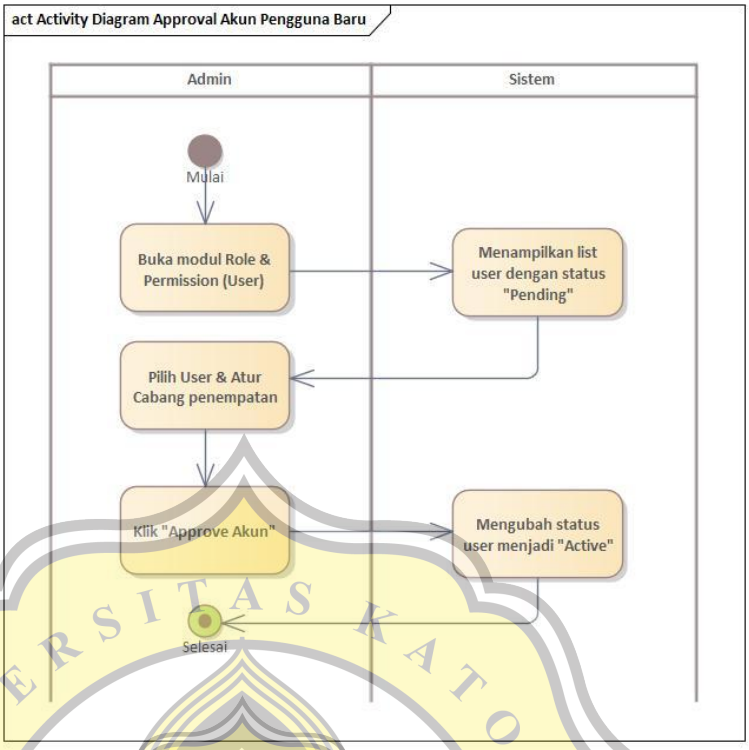
Pada Gambar 4.15 menguraikan alur perpindahan barang antar gudang atau cabang oleh admin. Pengguna memilih gudang asal, gudang tujuan, dan jumlah barang. Sistem memvalidasi ketersediaan stok di gudang asal. Jika tidak mencukupi, sistem menolak proses tersebut. Jika tersedia, sistem memotong stok asal, menambah stok tujuan, dan menyelesaikan mutasi.



Gambar 4. 15 *Activity Diagram* Melakukan Mutasi Barang *(Stock Transfer)*

j. *Activity Diagram Approval* Akun Pengguna Baru

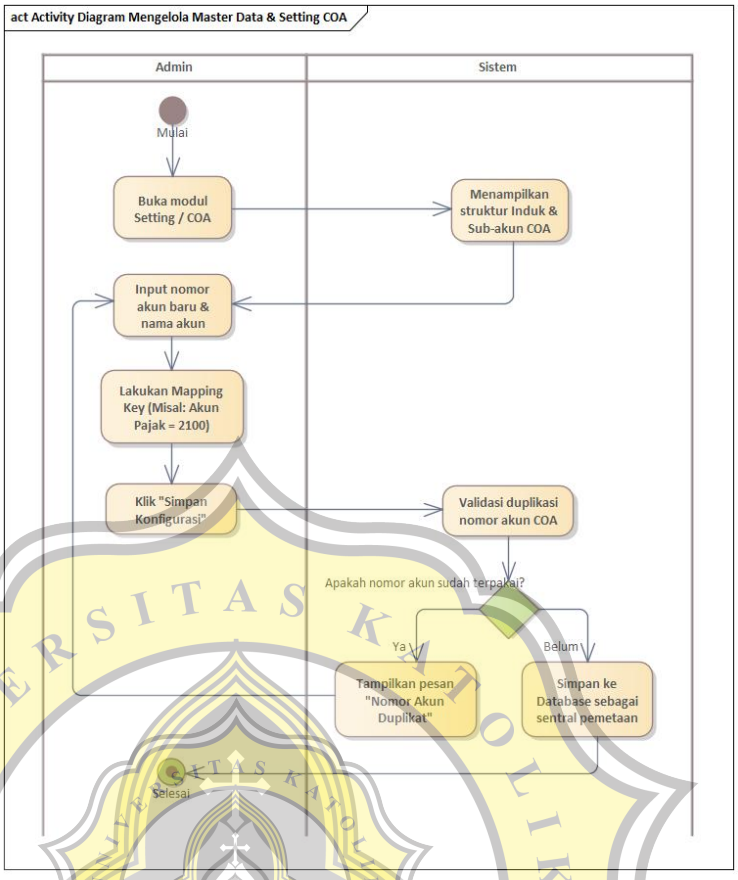
Pada Gambar 4.16 menggambarkan proses persetujuan akun baru oleh admin. admin meninjau daftar pengguna berstatus pending, menentukan penempatan cabang, dan mengonfirmasi persetujuan. Sistem kemudian mengubah status pengguna menjadi aktif, sehingga pengguna dapat mengakses portal sesuai wewenangnya.



Gambar 4. 16 *Activity Diagram Approval* Akun Pengguna Baru

k. *Activity Diagram* Mengelola Master Data & Setting COA

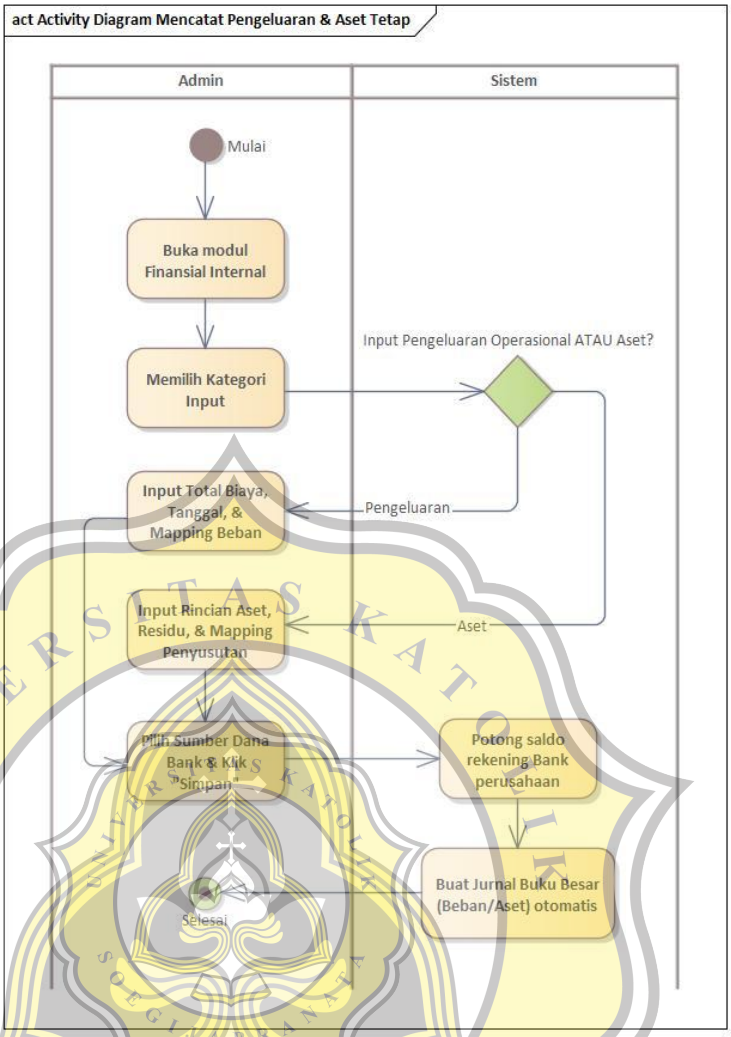
Pada Gambar 4.17 menjelaskan pengaturan pemetaan akuntansi sentral. admin memasukkan nomor akun baru, nama akun, dan menentukan mapping key (seperti akun pajak atau persediaan). Sistem memvalidasi agar tidak terjadi duplikasi nomor akun. Data yang valid akan disimpan sebagai parameter utama penggerak jurnal transaksi di seluruh modul.



Gambar 4. 17 *Activity Diagram* Mengelola Master Data & Setting COA

l. *Activity Diagram* Mencatat Pengeluaran & Aset Tetap

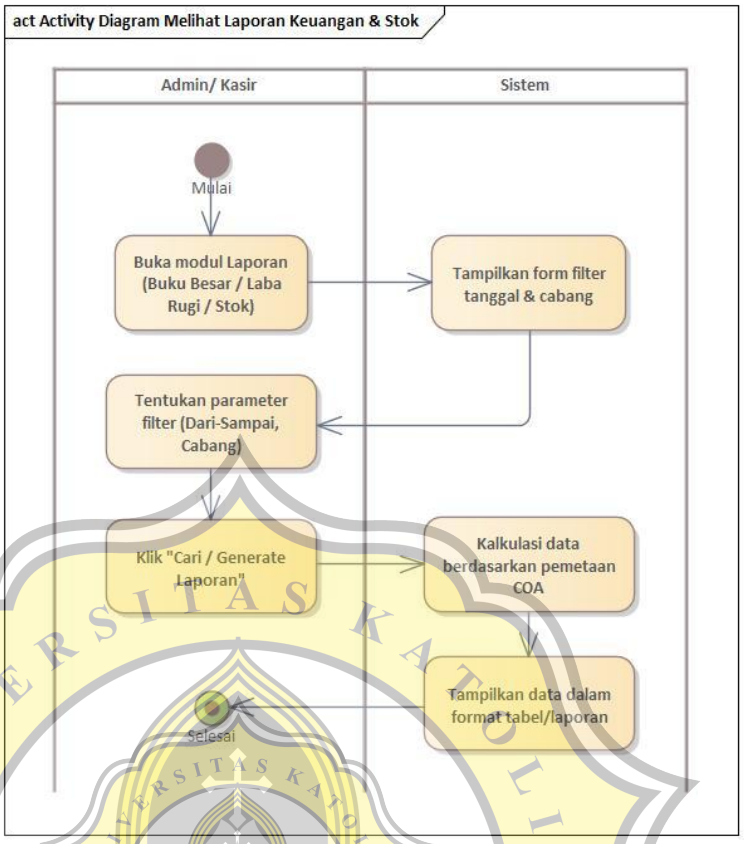
Pada Gambar 4.18 menunjukkan prosedur pencatatan finansial internal. admin mengklasifikasikan input antara beban operasional (expenses) atau aset tetap (fixed assets). Pengguna mengisi nominal, detail pemetaan beban atau penyusutan, dan memilih sumber dana. Sistem selanjutnya memotong saldo bank dan mencatat jurnal terkait di buku besar secara otomatis.



Gambar 4. 18 *Activity Diagram* Mencatat Pengeluaran & Aset Tetap

m. *Activity Diagram* Melihat Laporan Keuangan & Stok

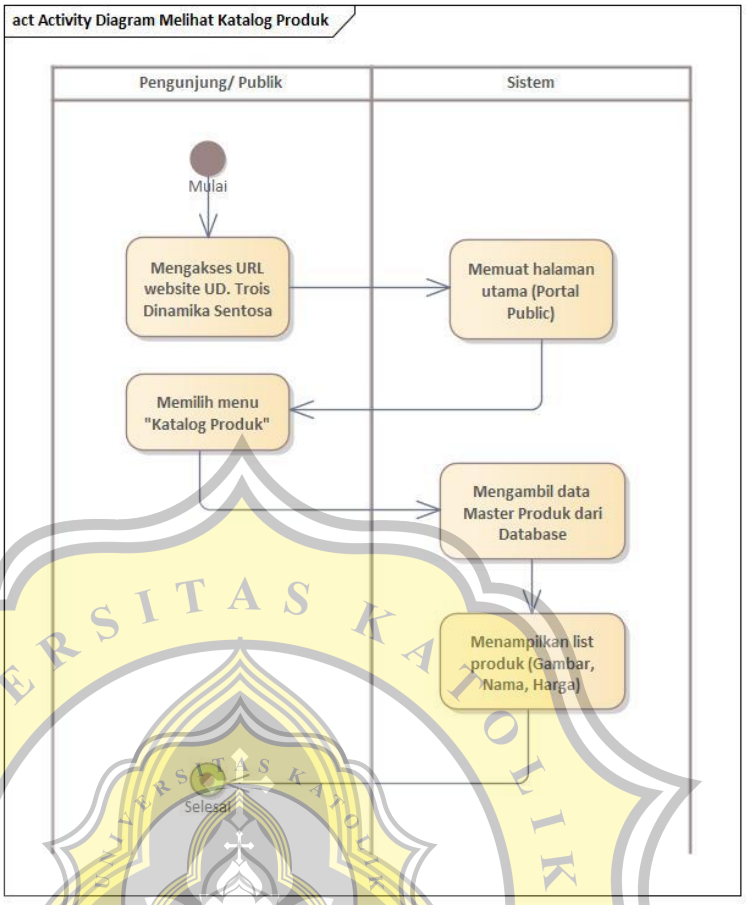
Pada Gambar 4.19 memaparkan alur pembuatan laporan oleh pengguna dengan hak akses manajerial atau operasional. Pengguna menentukan parameter pencarian seperti rentang tanggal dan lokasi cabang. Sistem mengalkulasi data berdasarkan pemetaan COA dan menyajikannya dalam bentuk tabel riwayat produk, kartu stok, maupun format laporan keuangan formal.



Gambar 4. 19 *Activity Diagram* Melihat Laporan Keuangan & Stok

n. *Activity Diagram* Melihat Katalog Produk

Pada Gambar 4.20 menjelaskan aktivitas pengunjung sistem melalui Portal Public. Pengunjung mengakses situs web utama dan menavigasi ke menu katalog produk. Sistem mengambil referensi dari pangkalan data master produk dan menampilkannya kepada pengunjung tanpa memerlukan proses autentikasi.



Gambar 4. 20 *Activity Diagram* Melihat Katalog Produk

4.2.4 *Entity Relationship Diagram* (ERD)

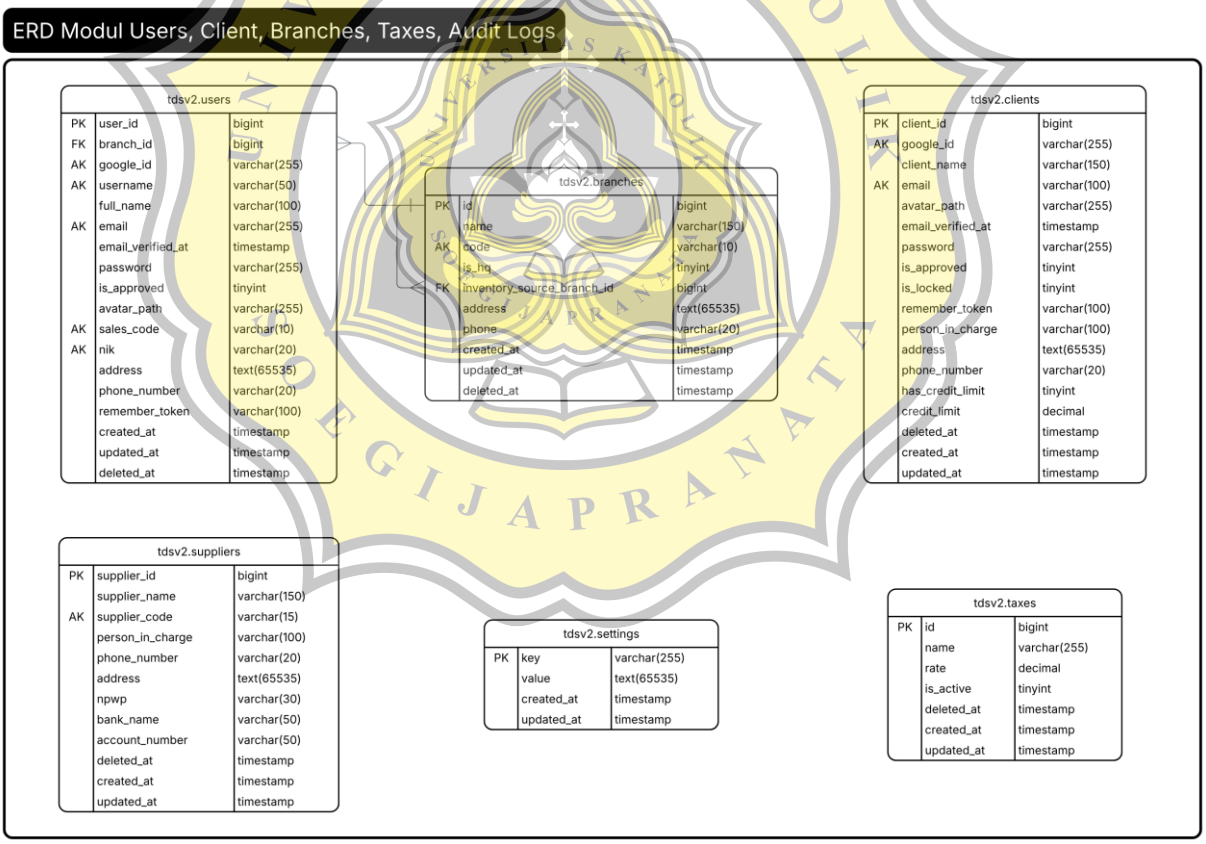
Sistem informasi pada UD. Trois Dinamika Sentosa dirancang menggunakan basis data relasional. Untuk menjaga batasan masalah dan mengoptimalkan pemahaman alur data, *rancangan Entity Relationship Diagram (ERD)* dipresentasikan secara parsial berdasarkan pengelompokan modul operasional. Pada setiap visualisasi *ERD*, tabel utama pada modul terkait direpresentasikan dengan warna biru, sedangkan tabel berwarna kuning merupakan tabel bayangan *(shadow table)*.

Tabel bayangan ini berfungsi sebagai entitas referensi dari modul lain untuk mempertahankan integritas relasi *Foreign Key* tanpa membuat diagram keseluruhan menjadi kehilangan relasi.

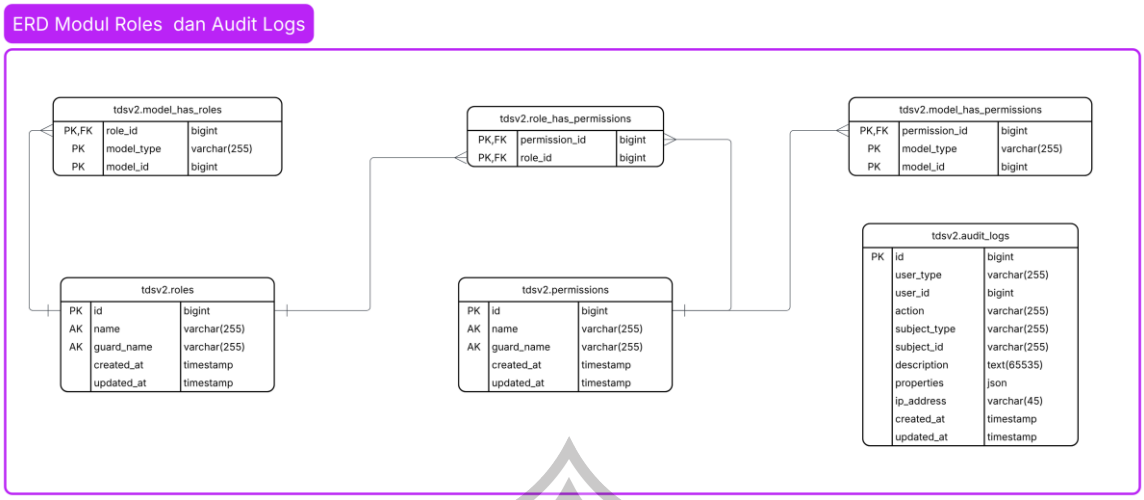
a. ***ERD* Modul Pengguna dan Konfigurasi.**

Subsistem ini merupakan tulang punggung operasional yang mengontrol otorisasi staf dan parameter global dari sistem.

- Tabel *branches* menyimpan informasi wilayah operasional, yang menjadi referensi kewilayahan bagi hampir seluruh transaksi.
- Tabel *users* menampung data autentikasi staf perusahaan.
- Hak akses dikelola menggunakan tabel *roles* dan *permissions*, yang dihubungkan ke entitas pengguna melalui tabel pivot *model_has_roles* dan *role_has_permissions*.
- Tabel settings dan *audit_logs* digunakan untuk menyimpan variabel konfigurasi sistem dan rekaman riwayat aktivitas operasional.



Gambar 4. 21 *ERD* Modul Pengguna dan Konfigurasi

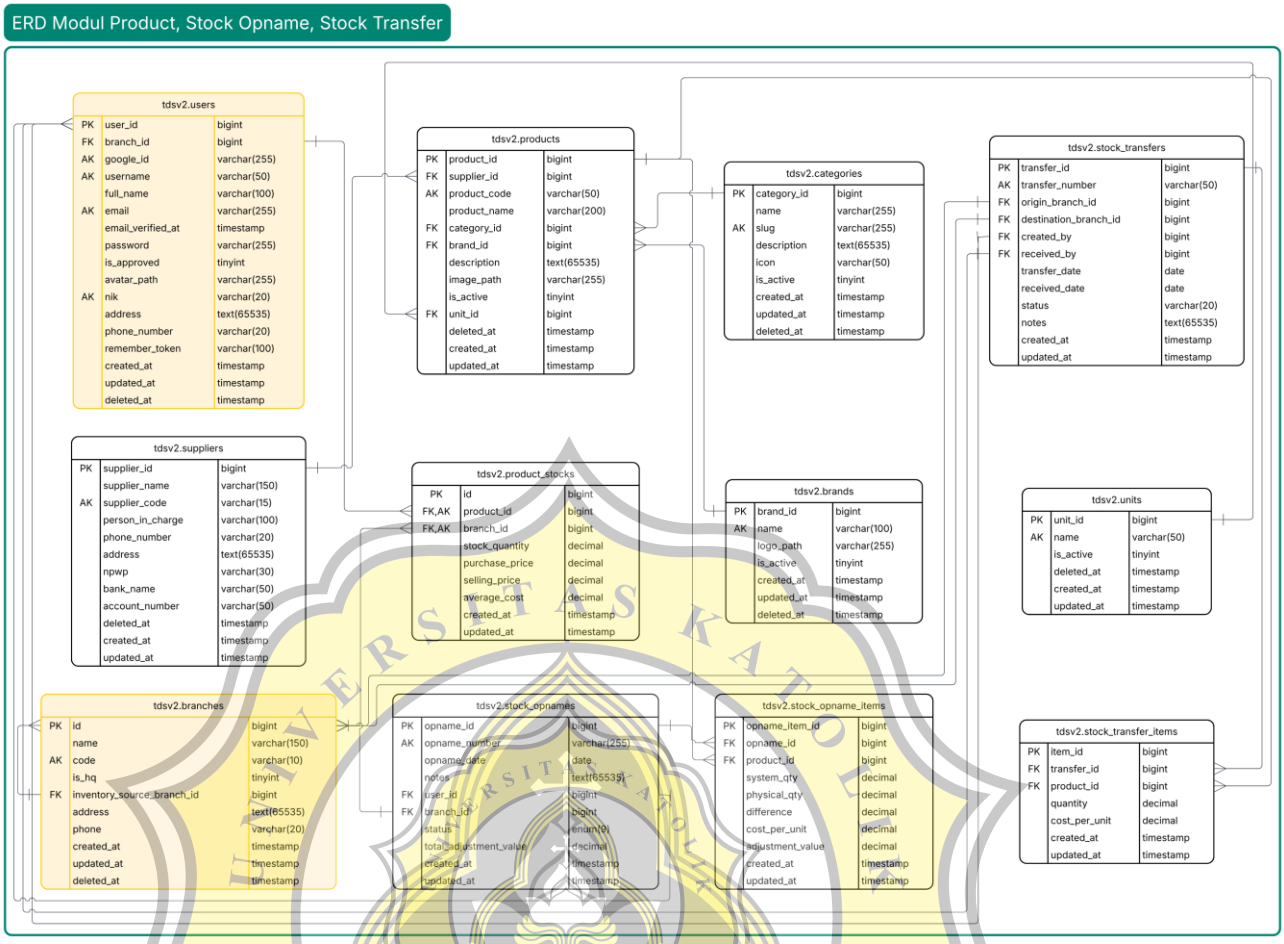


Gambar 4. 22 *ERD* Modul Pengguna dan Konfigurasi

b. ***ERD* Modul Master Produk**

Subsistem ini menggabungkan pengelolaan informasi katalog barang fisik dengan pencatatan pergerakan kuantitas stoknya secara internal.

- Tabel *products* menjadi pusat katalog barang, yang berelasi dengan *categories* untuk klasifikasi, *brands* untuk merk, dan *units* untuk satuan ukur
- Akumulasi kuantitas fisik barang secara spesifik di masing-masing cabang dicatat pada tabel *product_stocks*.
- Perpindahan fisik barang antar cabang direkam pada tabel *stock_transfers* beserta rincian barangnya di *stock_transfer_items*.
- Proses sinkronisasi antara jumlah stok pada sistem dengan fisik di gudang diakomodasi melalui tabel *stock_opnames* dan *stock_opname_items*.
- Modul gabungan ini meminjam tabel *suppliers*, *branches*, dan *users* sebagai tabel bayangan (kuning) untuk mengidentifikasi pihak pemasok barang, lokasi wilayah stok, serta staf yang melakukan mutasi.



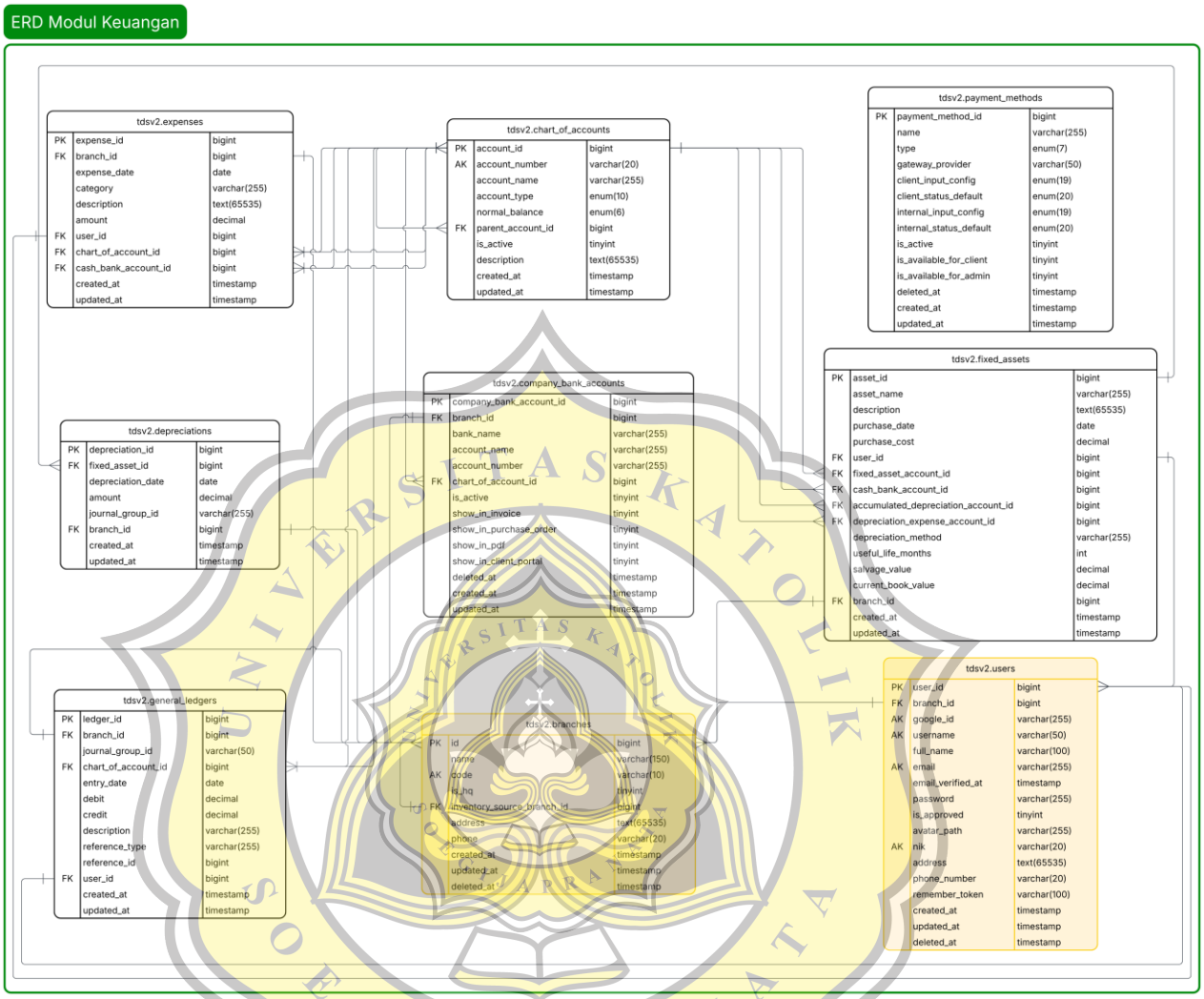
Gambar 4. 23 *ERD* Modul Master Produk dan Pergerakan Stok

c. *ERD* Modul Keuangan

Subsistem ini memetakan parameter finansial yang menjadi acuan dasar dalam pencatatan akuntansi dan laporan keuangan perusahaan.

- Tabel *chart_of_accounts* bertindak sebagai pusat klasifikasi akun keuangan (aset, kewajiban, modal, pendapatan, beban).
- Parameter metode pembayaran, pajak, dan rekening perusahaan diatur melalui tabel *payment_methods*, *taxes*, dan *company_bank_accounts*.
- Pengeluaran operasional perusahaan dicatat pada tabel *expenses*. Sementara itu, inventaris kantor dicatat pada tabel *fixed_assets* yang berelasi dengan tabel *depreciations* untuk perhitungan penyusutan nilai aset.
- Seluruh transaksi finansial yang tervalidasi akan bermuara dan dicatat pada buku besar melalui tabel *general_ledgers*.

- Tabel *users* dan *branches* bertindak sebagai tabel bayangan untuk merekam pelaku dan lokasi transaksi keuangan.



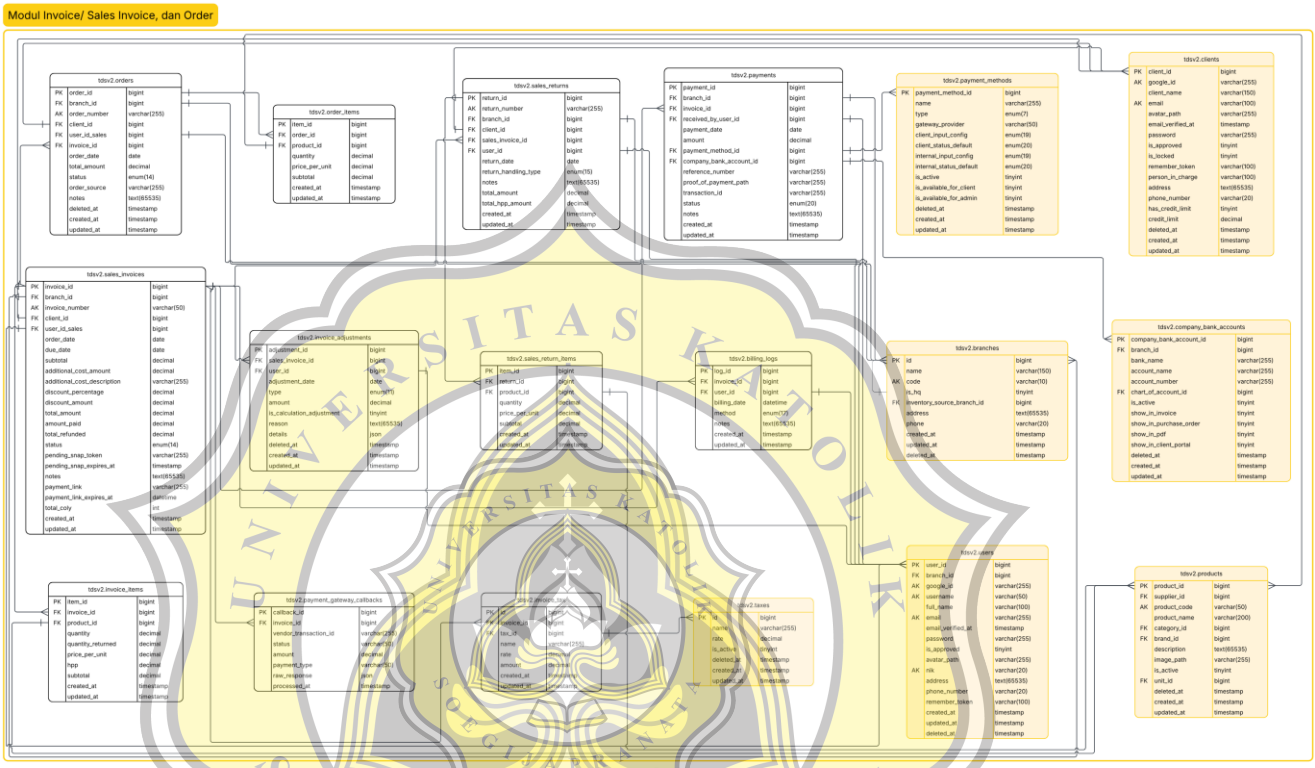
Gambar 4. 24 *ERD* Modul Keuangan

d. *ERD* Modul Transaksi Penjualan *(Sales Cycle)*

- Subsistem ini mendokumentasikan alur barang keluar, mulai dari tahap pemesanan hingga penagihan kepada pelanggan.
- Tahap inisiasi pesanan oleh tenaga penjual dicatat pada tabel *orders* dan detailnya di *order_items*.
- Penerbitan faktur tagihan resmi diproses melalui tabel sales_*invoices* beserta rincian pajaknya di *invoice_tax* dan detail barang di *invoice_items*.
- Penyesuaian nilai tagihan pasca-penerbitan dicatat di *invoice_adjustments*, sedangkan pengembalian fisik barang

dari klien ditangani melalui tabel *sales_returns* dan *sales_return_items.*

- Modul ini menggunakan tabel *clients*, *users*, *branches*, *products*, dan *taxes* sebagai tabel bayangan untuk melengkapi relasi transaksi penjualan.



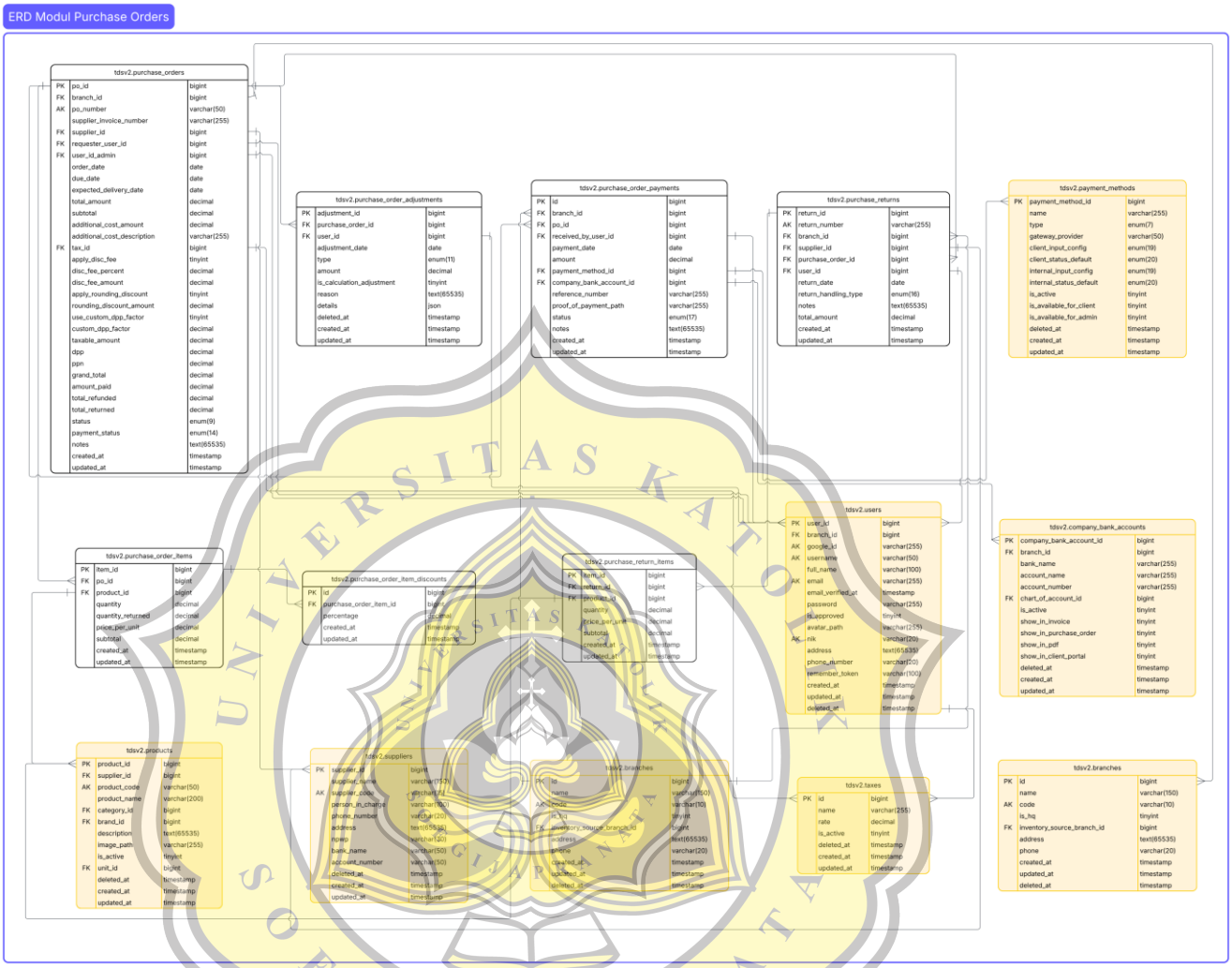
Gambar 4. 25 *ERD* Modul Transaksi Penjualan

e. ***ERD* Modul Transaksi Pembelian *(Purchase Cycle)***

Subsistem ini mengatur proses pengadaan barang dari pihak eksternal untuk menambah persediaan stok gudang.

- Transaksi utama direkam pada tabel *purchase_orders* dan didetailkan pada *purchase_order_items*. Pembelian ini difasilitasi dengan tabel pembantu, yaitu *purchase_order_item_discounts*, untuk mengakomodasi skema potongan harga per item.
- Koreksi nominal nilai pembelian dicatat pada tabel *purchase_order_adjustments*, sedangkan retur barang ke pemasok dicatat melalui tabel *purchase_returns* dan rinciannya di *purchase_return_items*.

- Modul ini meminjam tabel *suppliers*, *users*, *branches*, *products*, dan *taxes* sebagai tabel bayangan.



Gambar 4. 26 *ERD* Modul Transaksi Pembelian

4.3 Implementasi Sistem

4.3.1 Halaman Publik

Modul pertama pada sistem ini adalah Halaman Publik (*Landing Page*) yang bersifat terbuka dan dapat diakses oleh seluruh pengguna tanpa perlu melakukan autentikasi (*login*). Halaman ini berfungsi utama sebagai profil perusahaan (*company profile*) digital dan etalase interaktif untuk UD. Trois Dinamika Sentosa. Melalui halaman ini, pengunjung dapat melihat informasi mengenai identitas perusahaan, keunggulan layanan, visi dan misi, serta menelusuri katalog alat teknik dan suku cadang secara lengkap menggunakan fitur pencarian dan filter kategori. Tampilan keseluruhan antarmuka *landing page* ini dapat dilihat pada Gambar 4.27.

Gambar 4. 27 Halaman Profil Perusahaan

Sebagaimana ditunjukkan pada Gambar 4.28, pada bagian implementasi *Controller* ini, FrontController mengelola tiga fitur utama yaitu visualisasi beranda melalui fungsi index, penyajian profil melalui fungsi *about*, dan manajemen katalog produk melalui fungsi *catalog* yang telah dilengkapi dengan fitur parameter *query* untuk pencarian data secara dinamis

```php
class FrontController extends Controller
{
    public function index(): \Illuminate\View\View
    {
        $settings = \App\Models\Setting::getAllSettings();
        $categories = \App\Models\Category::where('is_active', true)->get();
        $brands = \App\Models\Brand::where('is_active', true)->get();
        $galleries = \App\Models\CompanyGallery::where('is_active', true)->get();
        return view('web.home', compact('settings', 'categories', 'brands', 'galleries'));
    }

    public function about(): View
    {
        $settings = Setting::getAllSettings();
        return view('web.about', compact('settings'));
    }

    public function catalog(Request $request): View
    {
        $settings = Setting::getAllSettings();
        $categories = Category::where('is_active', true)->orderBy('name')->get();
        $query = Product::with(['category', 'unit'])->where('is_active', true);
        if ($request->filled('category')) {
            $query->where('category_id', $request->category);
        }
        if ($request->filled('search')) {
            $query->where('product_name', 'like', '%' . $request->search . '%');
        }
        $products = $query->latest()->paginate(12)->appends($request->query());
        return view('web.catalog', compact('settings', 'categories', 'products'));
    }
}
```

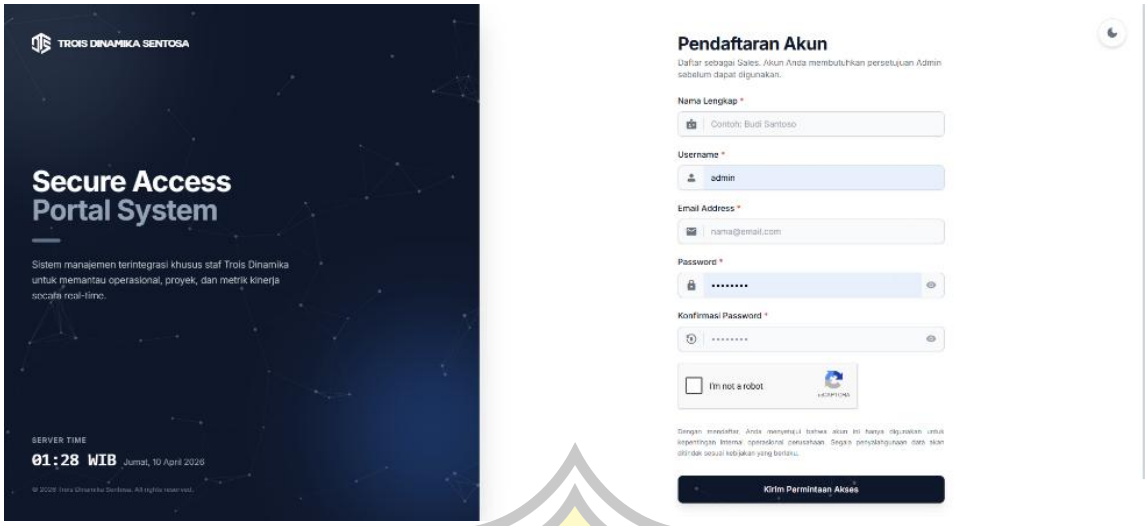
Gambar 4. 28 Kode Program Halaman Profil Perusahaan

4.3.2 Halaman Autentikasi Portal admin

Modul autentikasi ini merupakan gerbang keamanan utama bagi staf sebelum mengakses Admin Portal System. Implementasinya mencakup antarmuka pengguna (*front-end*) dan logika pengendali (*back-end controller*) yang dibagi ke dalam tiga proses utama, yaitu pendaftaran, login, dan pemulihan kata sandi.

a. Tahap Registrasi (Pendaftaran Akun)

Secara antarmuka, halaman pendaftaran memfasilitasi pembuatan akun staf baru melalui pengisian formulir data diri (Nama Lengkap, Username, Email, *Password*) yang dilengkapi dengan sistem keamanan reCAPTCHA. Tampilan antarmuka halaman registrasi dapat dilihat pada Gambar 4.29

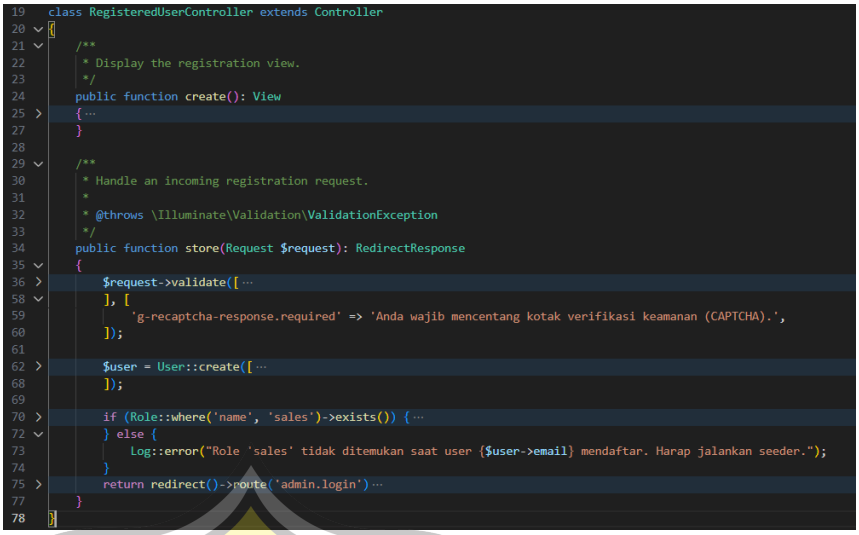


Gambar 4. 29 Halaman Pendaftaran Akun/ Pengguna Baru

Pada sisi *backend*, pendaftaran diakomodasi melalui dua cara:

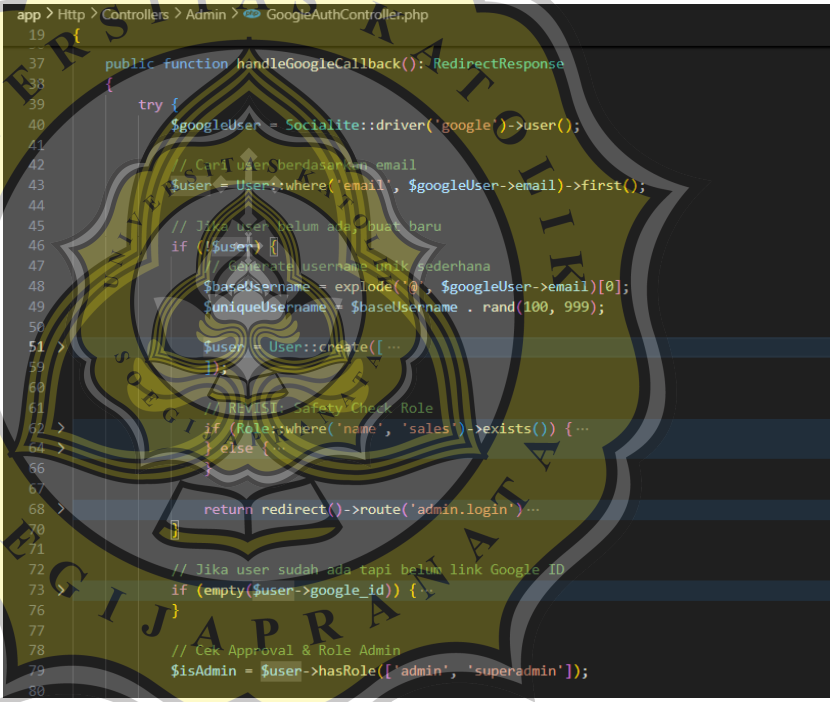
- **Registrasi Manual:**
  Ditangani oleh RegisteredUserController yang memvalidasi masukan formulir. Setelah valid, sistem menyimpan akun baru dengan status *pending (is_approved = false)* dan memberikan hak akses bawaan sebagai sales.
- **Registrasi via Google:**
  Ditangani oleh *GoogleAuthController* yang menangkap respons profil Google pengguna. Sistem akan membuat akun baru secara otomatis dengan kata sandi acak dan menyimpannya dengan status pending.

  Potongan kode untuk logika registrasi ini dapat dilihat pada Gambar 4.30 dan Gambar 4.31.



```
class RegisteredUserController extends Controller
{
    /**
     * Display the registration view.
     */
    public function create(): View
    {…
    }

    /**
     * Handle an incoming registration request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([…
        ], [
            'g-recaptcha-response.required' => 'Anda wajib mencentang kotak verifikasi keamanan (CAPTCHA).',
        ]);

        $user = User::create([…
        ]);

        if (Role::where('name', 'sales')->exists()) {…
        } else {
            Log::error("Role 'sales' tidak ditemukan saat user {$user->email} mendaftar. Harap jalankan seeder.");
        }
        return redirect()->route('admin.login')…
    }
}
```

Gambar 4. 30 Kode Program dari RegisteredUserController



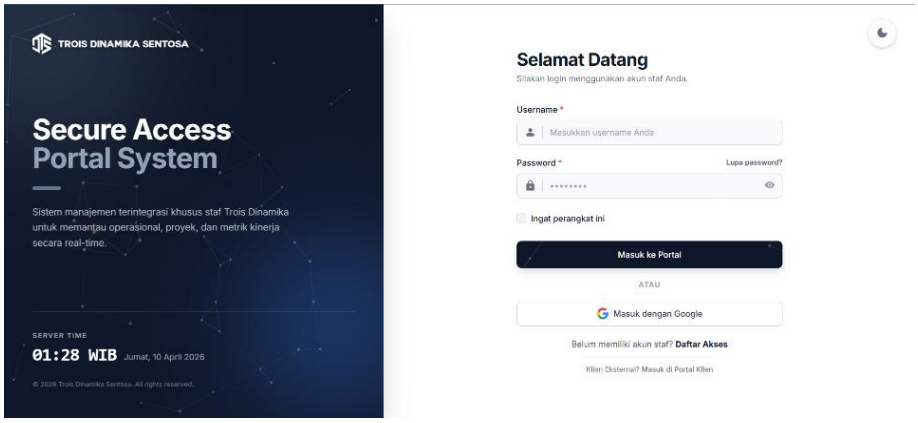
app > Http > Controllers > Admin > GoogleAuthController.php

```
{
    public function handleGoogleCallback(): RedirectResponse
    {
        try {
            $googleUser = Socialite::driver('google')->user();

            // Cari user berdasarkan email
            $user = User::where('email', $googleUser->email)->first();

            // Jika user belum ada, buat baru
            if (!$user) {
                // Generate username unik sederhana
                $baseUsername = explode('@', $googleUser->email)[0];
                $uniqueUsername = $baseUsername . rand(100, 999);

                $user = User::create([…
                ]);

                // REVISI: Safety Check Role
                if (Role::where('name', 'sales')->exists()) {…
                } else {…
                }

                return redirect()->route('admin.login')…
            }

            // Jika user sudah ada tapi belum link Google ID
            if (empty($user->google_id)) {…
            }

            // Cek Approval & Role Admin
            $isAdmin = $user->hasRole(['admin', 'superadmin']);
```

Gambar 4. 31 Program dari GoogleAuthController

b. Tahap Login (Autentikasi Pengguna)

Halaman login menyediakan formulir autentikasi manual serta tombol integrasi masuk menggunakan akun Google (SSO). Tampilan antarmuka halaman login dapat dilihat pada Gambar 4.32.



Gambar 4. 32 Halaman Masuk/ Login Portal admin

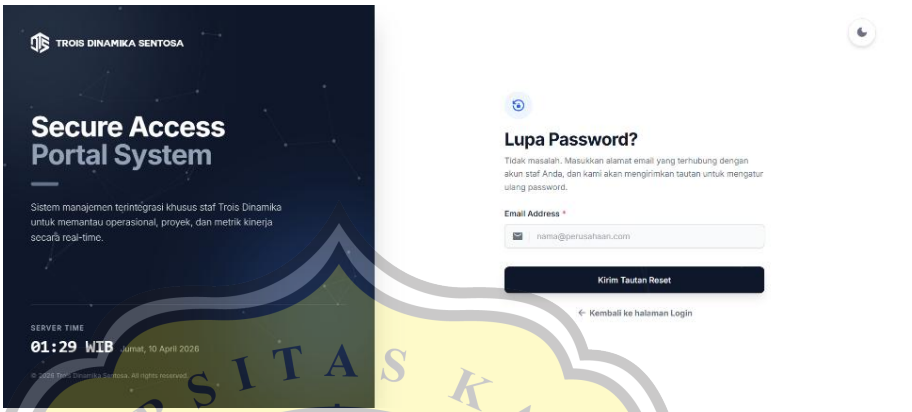
Dari sisi back-end, sistem menerapkan pengecekan otorisasi berlapis. Baik pada login manual maupun via GoogleAuthController, sistem tidak hanya memvalidasi kecocokan kredensial (kecocokan *password* atau validitas akun Google), tetapi juga memeriksa status verifikasi *(is_approved)*. Jika akun belum disetujui oleh admin, sesi login akan ditolak. Jika valid, sesi diterbitkan dan pengguna diarahkan ke dashboard. Potongan kode implementasi autentikasi ini ditunjukkan pada Gambar 4.33.

```
app > Http > Controllers > Admin > GoogleAuthController.php
19   {
37       public function handleGoogleCallback(): RedirectResponse
38       {
39           try {
40               $googleUser = Socialite::driver('google')->user();
41
42               $user = User::where('email', $googleUser->email)->first();
43
44               if (!$user) {…
66               }
67
68               if (empty($user->google_id)) {…
71               }
72
73               $isAdmin = $user->hasRole(['admin', 'superadmin']);
74
75               if (!$user->is_approved && !$isAdmin) {…
78               }
79
80               Auth::login($user, true);
81
82               return redirect()->intended(route('admin.dashboard'));
83
84           } catch (Exception $e) {
85               Log::error('Google login error', [
86                   'message' => $e->getMessage(),
87                   'trace' => $e->getTraceAsString(),
88               ]);
89
90               return redirect()->route('admin.login')…
92           }
93       }
94   }
```

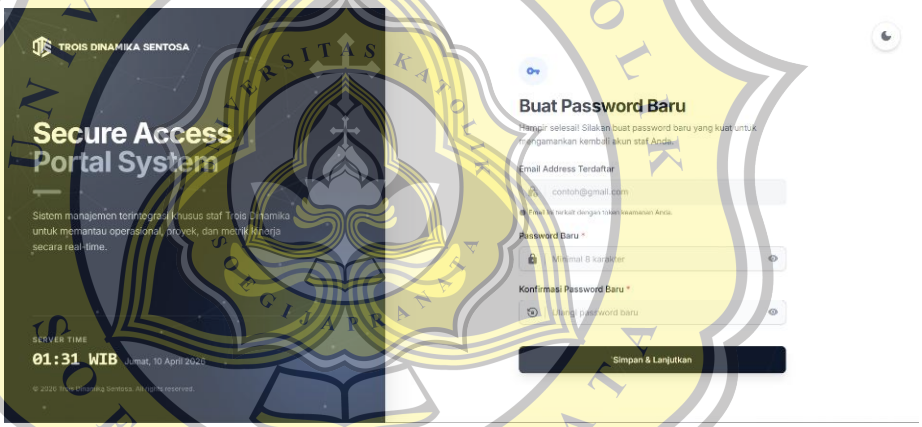
Gambar 4. 33 Kode Program GoogleAuthController

c. Tahap Pemulihan Kata Sandi (Lupa *Password*)

Antarmuka modul ini terdiri dari dua bagian: halaman permintaan tautan *(link)* melalui input email dan halaman pembuatan kata sandi baru. Tampilan antarmuka pemulihan kata sandi dapat dilihat pada Gambar 4.34 dan Gambar 4.35.



Gambar 4. 34 Halaman Lupa *Password*



Gambar 4. 35 Halaman Reset *Password*

Logika back-end pada proses ini ditangani oleh dua controller:

a) **Permintaan Tautan:**

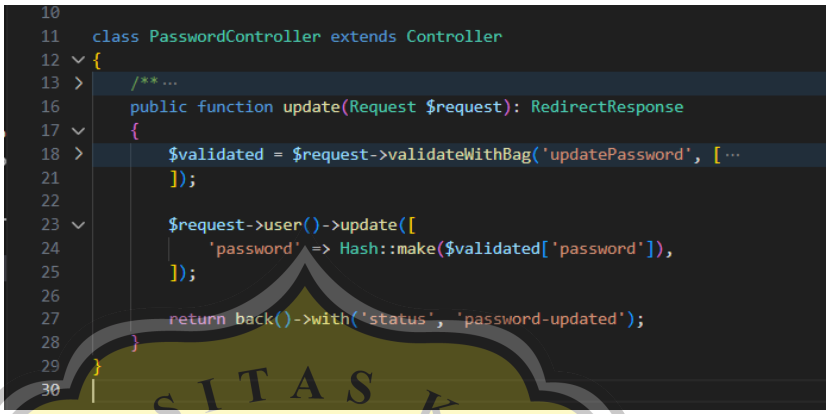
*Password*ResetLinkController memvalidasi keberadaan email di basis data, lalu menginstruksikan layanan *Password* Broker untuk menghasilkan token unik dan mengirimkannya ke email pengguna.

b) **Penyimpanan *Password* Baru:**

Setelah tautan dibuka, New*Password*Controller memvalidasi keabsahan token pada URL. Jika sesuai, sistem akan

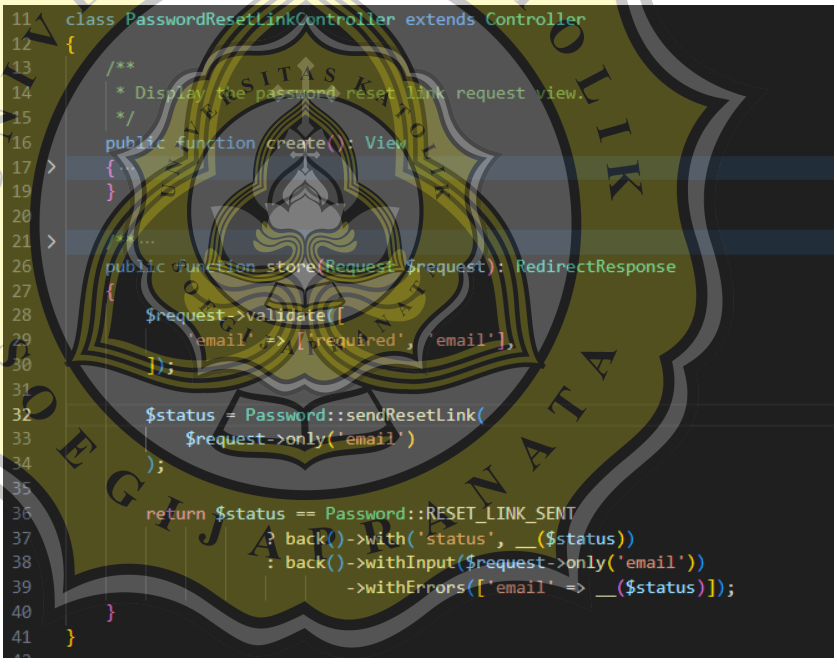
memperbarui kata sandi lama dengan kata sandi baru yang telah dienkripsi menggunakan metode hashing.

Potongan kode untuk fitur pemulihan ini dapat dilihat pada Gambar 4.36 dan Gambar 4.37



```
class PasswordController extends Controller
{
    /** ...
    public function update(Request $request): RedirectResponse
    {
        $validated = $request->validateWithBag('updatePassword', [...
        ]);

        $request->user()->update([
            'password' => Hash::make($validated['password']),
        ]);

        return back()->with('status', 'password-updated');
    }
}
```

Gambar 4. 36 Kode Program *Password*Controller



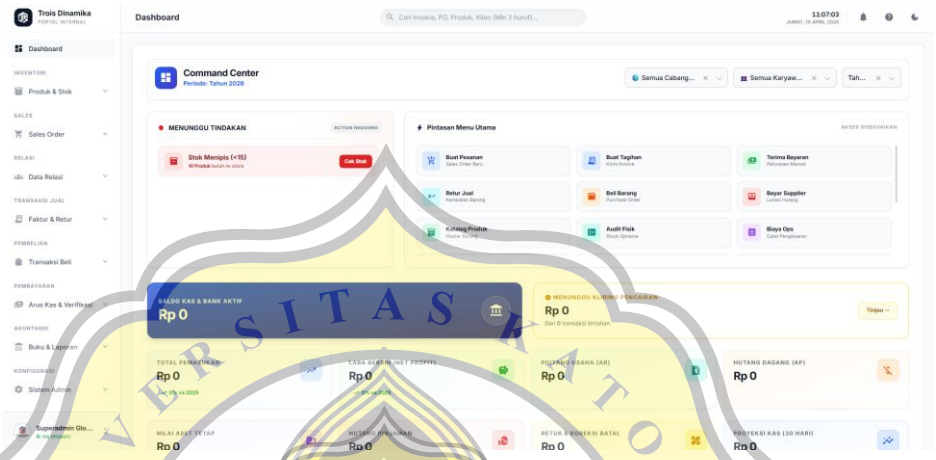
```
class PasswordResetLinkController extends Controller
{
    /**
     * Display the password reset link request view.
     */
    public function create(): View
    {
    }

    /** ...
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'email' => ['required', 'email'],
        ]);

        $status = Password::sendResetLink(
            $request->only('email')
        );

        return $status == Password::RESET_LINK_SENT
                    ? back()->with('status', __($status))
                    : back()->withInput($request->only('email'))
                            ->withErrors(['email' => __($status)]);
    }
}
```

Gambar 4. 37 Kode Program *Password*ResetLinkController

4.3.3 Halaman Utama (*Dashboard)*

Setelah pengguna berhasil melewati proses login, sistem akan langsung mengarahkan mereka ke halaman dashboard utama seperti yang ditunjukkan pada Gambar 4.38. Halaman dashboard ini berfungsi sebagai pusat kendali *(Command Center)* yang memberikan ringkasan aktivitas operasional dan finansial perusahaan secara *real-time*. Pada antarmuka ini,

pengguna dapat melihat panel peringatan untuk tindakan yang memerlukan respons cepat (seperti notifikasi stok menipis), deretan tombol pintasan untuk menu utama, serta berbagai kartu metrik yang menyajikan angka saldo kas, total pemasukan, hingga laba bersih. Di sebelah kiri layar juga terdapat sidebar navigasi untuk mengakses seluruh modul di dalam sistem.



Gambar 4. 38 Halaman Utama/ Dashboard

Semua data dinamis yang muncul di halaman tersebut diproses secara terpusat melalui DashboardController di dalam fungsi index. Potongan kode untuk controller ini dapat dilihat pada Gambar 4.39

Proses pemuatan halaman ini diawali dengan identifikasi hak akses menggunakan fungsi Auth::user(). Sistem akan mengecek apakah pengguna memiliki izin melihat laporan keuangan atau inventori. Jika pengguna yang login adalah level staf, *query* data akan otomatis dibatasi hanya untuk cabang staf tersebut melalui variabel $selectedBranchId. Namun, jika pengguna adalah admin atau superadmin, sistem akan menarik rekapan data dari seluruh cabang.

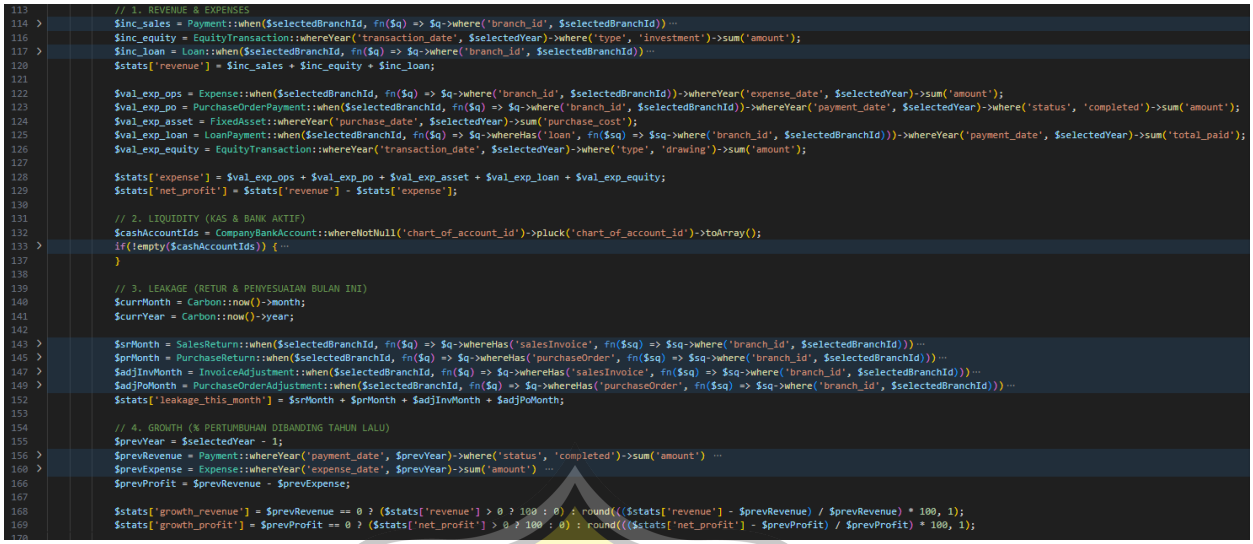
```php
class DashboardController extends Controller implements HasMiddleware
{
    public static function middleware(): array
    {...
    }

    public function index(Request $request): View
    {
        $user = Auth::user();
        $selectedYear = $request->input('year', date('Y'));
        $selectedUserId = $request->input('filter_user_id');
        $selectedBranchId = $request->input('filter_branch_id');

        $isAdmin = $user->hasRole(['admin', 'superadmin']);
        $canViewFinancials = $user->can('view-reports');
        $canViewInventory = $user->can('view-products');

        if (!$isAdmin) {
            $selectedUserId = $user->user_id;
            $selectedBranchId = $user->branch_id;
        }

        $branches = Branch::orderBy('name')->get();
        // Eager load roles untuk Optgroup TomSelect di View
        $allUsers = $isAdmin ? User::with('roles')->orderBy('full_name')->get() : collect([]);
```

Gambar 4. 39 Kode Program Dashboard

Untuk merangkum dan mengirim data ke halaman *view*, *controller* ini mengandalkan beberapa variabel utama. Variabel $stats digunakan untuk menampung hasil kalkulasi agregat keuangan, mulai dari total pendapatan *(revenue)*, pengeluaran operasional *(expense)*, laba bersih *(net_profit)*, hingga tingkat keberhasilan penagihan (*success rate*). Sistem juga menggunakan variabel $forecast untuk memproyeksikan perputaran arus kas selama 30 hari ke depan berdasarkan tanggal jatuh tempo faktur, dapat dilihat pada Gambar 4.40 dan Gambar 4.41

```php
        // --- INIT VARIABLES ---
        $stats = [
            'revenue' => 0, 'expense' => 0, 'net_profit' => 0,
            'growth_revenue' => 0, 'growth_profit' => 0, 'prev_year_label' => $selectedYear - 1,
            'payables' => 0, 'loans' => 0, 'assets' => 0, 'receivables_global' => 0,
            'total_invoices' => 0, 'paid_invoices' => 0, 'total_sales_value' => 0,
            'active_clients' => 0, 'success_rate' => 0, 'receivables_filtered' => 0,
            'sales_breakdown' => [],
            'purchase_breakdown' => [],
            'liquidity_balance' => 0,
            'leakage_this_month' => 0,
        ];

        $forecast = [
            'incoming_30_days' => 0, 'outgoing_30_days' => 0, 'net_forecast' => 0,
            'monthly_target' => 500000000, 'current_monthly_sales' => 0, 'target_percentage' => 0
        ];
```

Gambar 4. 40 Potongan Kode Program Bagian Statistik Dashboard



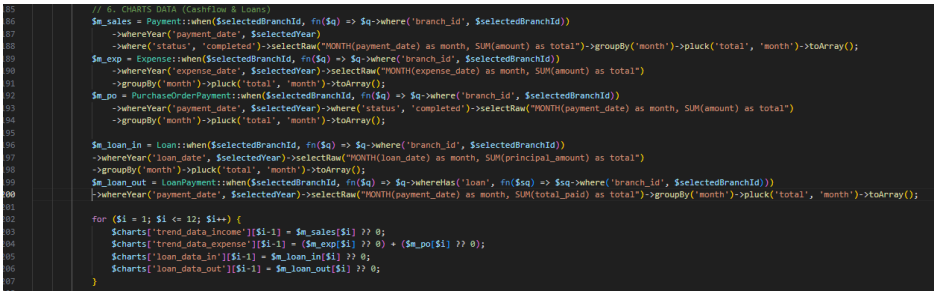
```
// 1. REVENUE & EXPENSES
$inc_sales = Payment::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))···
$inc_equity = EquityTransaction::whereYear('transaction_date', $selectedYear)->where('type', 'investment')->sum('amount');
$inc_loan = Loan::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))···
$stats['revenue'] = $inc_sales + $inc_equity + $inc_loan;

$val_exp_ops = Expense::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))->whereYear('expense_date', $selectedYear)->sum('amount');
$val_exp_po = PurchaseOrderPayment::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))->whereYear('payment_date', $selectedYear)->where('status', 'completed')->sum('amount');
$val_exp_asset = FixedAsset::whereYear('purchase_date', $selectedYear)->sum('purchase_cost');
$val_exp_loan = LoanPayment::when($selectedBranchId, fn($q) => $q->whereHas('loan', fn($sq) => $sq->where('branch_id', $selectedBranchId)))->whereYear('payment_date', $selectedYear)->sum('total_paid');
$val_exp_equity = EquityTransaction::whereYear('transaction_date', $selectedYear)->where('type', 'drawing')->sum('amount');

$stats['expense'] = $val_exp_ops + $val_exp_po + $val_exp_asset + $val_exp_loan + $val_exp_equity;
$stats['net_profit'] = $stats['revenue'] - $stats['expense'];

// 2. LIQUIDITY (KAS & BANK AKTIF)
$cashAccountIds = CompanyBankAccount::whereNotNull('chart_of_account_id')->pluck('chart_of_account_id')->toArray();
if(!empty($cashAccountIds)) {···
}

// 3. LEAKAGE (RETUR & PENYESUAIAN BULAN INI)
$currMonth = Carbon::now()->month;
$currYear = Carbon::now()->year;

$srMonth = SalesReturn::when($selectedBranchId, fn($q) => $q->whereHas('salesInvoice', fn($sq) => $sq->where('branch_id', $selectedBranchId)))···
$prMonth = PurchaseReturn::when($selectedBranchId, fn($q) => $q->whereHas('purchaseOrder', fn($sq) => $sq->where('branch_id', $selectedBranchId)))···
$adjInvMonth = InvoiceAdjustment::when($selectedBranchId, fn($q) => $q->whereHas('salesInvoice', fn($sq) => $sq->where('branch_id', $selectedBranchId)))···
$adjPoMonth = PurchaseOrderAdjustment::when($selectedBranchId, fn($q) => $q->whereHas('purchaseOrder', fn($sq) => $sq->where('branch_id', $selectedBranchId)))···
$stats['leakage_this_month'] = $srMonth + $prMonth + $adjInvMonth + $adjPoMonth;

// 4. GROWTH (% PERTUMBUHAN DIBANDING TAHUN LALU)
$prevYear = $selectedYear - 1;
$prevRevenue = Payment::whereYear('payment_date', $prevYear)->where('status', 'completed')->sum('amount') ···
$prevExpense = Expense::whereYear('expense_date', $prevYear)->sum('amount') ···
$prevProfit = $prevRevenue - $prevExpense;

$stats['growth_revenue'] = $prevRevenue == 0 ? ($stats['revenue'] > 0 ? 100 : 0) : round((($stats['revenue'] - $prevRevenue) / $prevRevenue) * 100, 1);
$stats['growth_profit'] = $prevProfit == 0 ? ($stats['net_profit'] > 0 ? 100 : 0) : round((($stats['net_profit'] - $prevProfit) / $prevProfit) * 100, 1);
```

Gambar 4. 41 Potongan Kode Program dari Bagian Finansial

Agar data lebih mudah dibaca secara visual, kode program menyiapkan variabel $charts yang bertugas menyusun data berformat *array* ke dalam grafik, seperti tren pemasukan bulanan dan umur piutang (*aging schedule*). Selain data statistik, *controller* ini juga menjalankan fungsi peringatan cerdas (*smart detection*). Hal ini diimplementasikan melalui variabel $lowStockProducts yang secara otomatis menyeleksi barang dengan sisa stok di bawah 15 unit, serta variabel $systemAlerts yang akan memunculkan peringatan apabila mendeteksi adanya transaksi bank bulan lalu yang belum direkonsiliasi, dapat dilihat pada Gambar 4.42 dan 4.43

```
$charts = [
    'months' => ['Jan', 'Feb', 'Mar', 'Apr', 'Mei', 'Jun', 'Jul', 'Agu', 'Sep', 'Okt', 'Nov', 'Des'],
    'trend_data_income' => array_fill(0, 12, 0),
    'trend_data_expense' => array_fill(0, 12, 0),
    'expense_composition' => [0, 0, 0, 0, 0],
    'loan_data_in' => array_fill(0, 12, 0),
    'loan_data_out' => array_fill(0, 12, 0),
    'ar_aging' => [0, 0, 0],
    'ap_aging' => [0, 0, 0],
    'top_products_labels' => [],
    'top_products_data' => []
];
```

Gambar 4. 42 Kode Program dari Bagian Penampilan Charts

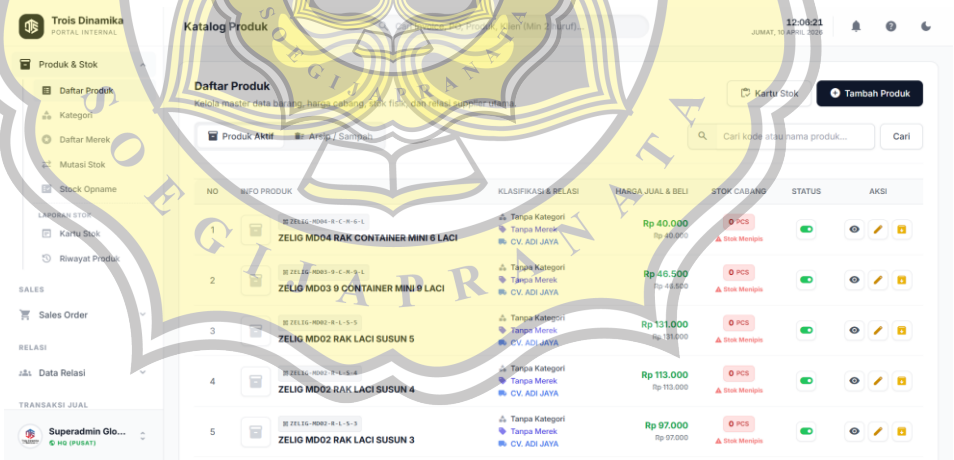


```
// 6. CHARTS DATA (Cashflow & Loans)
$m_sales = Payment::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))
    ->whereYear('payment_date', $selectedYear)
    ->where('status', 'completed')->selectRaw("MONTH(payment_date) as month, SUM(amount) as total")->groupBy('month')->pluck('total', 'month')->toArray();
$m_exp = Expense::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))
    ->whereYear('expense_date', $selectedYear)->selectRaw("MONTH(expense_date) as month, SUM(amount) as total")
    ->groupBy('month')->pluck('total', 'month')->toArray();
$m_po = PurchaseOrderPayment::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))
    ->whereYear('payment_date', $selectedYear)->where('status', 'completed')->selectRaw("MONTH(payment_date) as month, SUM(amount) as total")
    ->groupBy('month')->pluck('total', 'month')->toArray();

$m_loan_in = Loan::when($selectedBranchId, fn($q) => $q->where('branch_id', $selectedBranchId))
->whereYear('loan_date', $selectedYear)->selectRaw("MONTH(loan_date) as month, SUM(principal_amount) as total")
->groupBy('month')->pluck('total', 'month')->toArray();
$m_loan_out = LoanPayment::when($selectedBranchId, fn($q) => $q->whereHas('loan', fn($sq) => $sq->where('branch_id', $selectedBranchId)))
->whereYear('payment_date', $selectedYear)->selectRaw("MONTH(payment_date) as month, SUM(total_paid) as total")->groupBy('month')->pluck('total', 'month')->toArray();

for ($i = 1; $i <= 12; $i++) {
    $charts['trend_data_income'][$i-1] = $m_sales[$i] ?? 0;
    $charts['trend_data_expense'][$i-1] = ($m_exp[$i] ?? 0) + ($m_po[$i] ?? 0);
    $charts['loan_data_in'][$i-1] = $m_loan_in[$i] ?? 0;
    $charts['loan_data_out'][$i-1] = $m_loan_out[$i] ?? 0;
}
```

Gambar 4. 43 Kode Program dari Bagian Penampilan Charts

4.3.4 Modul Manajemen Produk

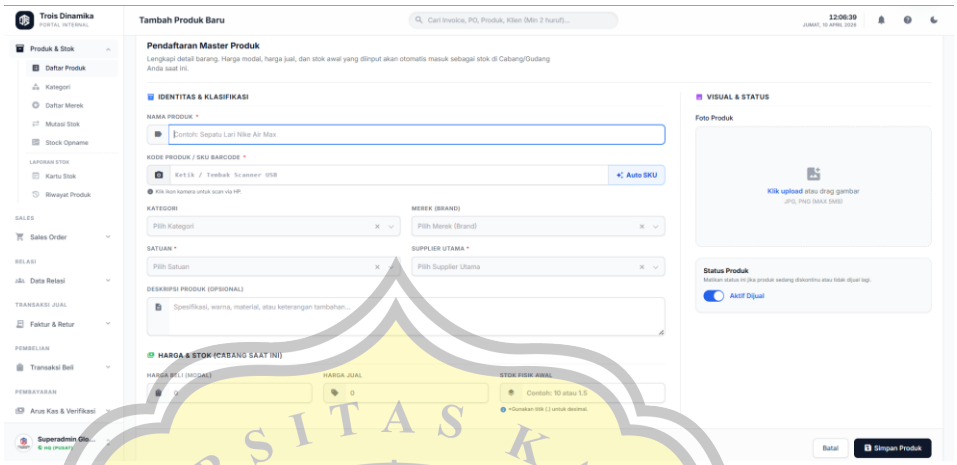
Proses pertama dalam pengelolaan master data barang dilakukan melalui halaman katalog produk. Gambar 4.44 di bawah merupakan tampilan antarmuka utama daftar produk. Pada halaman ini, pengguna dapat melihat seluruh daftar barang yang terdaftar di sistem, melakukan pencarian berdasarkan nama atau kode, melihat ketersediaan stok cabang, dan menggunakan tombol aksi yang tersedia. Namun, karena ini merupakan modul krusial, sistem menerapkan pembatasan hak akses yang ketat. Hanya pengguna dengan *role* atau izin tertentu yang diizinkan untuk melihat, menambah, mengedit, atau mengarsipkan data barang.



Gambar 4. 44 Halaman Utama Modul Manajemen Produk

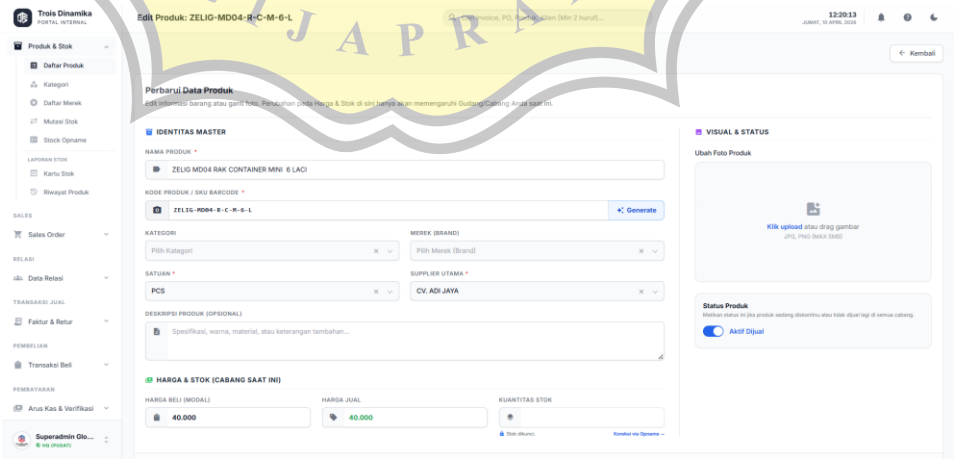
Apabila pengguna memiliki hak akses untuk menambah barang, mereka dapat menekan tombol tambah produk di pojok kanan atas. Nantinya akan muncul halaman formulir pendaftaran barang baru seperti pada Gambar 4.45. Merupakan antarmuka untuk mendaftarkan master produk. Staf dapat melengkapi informasi seperti identitas barang, mengunggah foto,

menentukan kategori, memilih *supplier* utama, serta mengisi harga beli, harga jual, dan stok awal. Terdapat juga tombol bantuan *Auto SKU* agar sistem bisa membuatkan kode *barcode* unik secara otomatis.



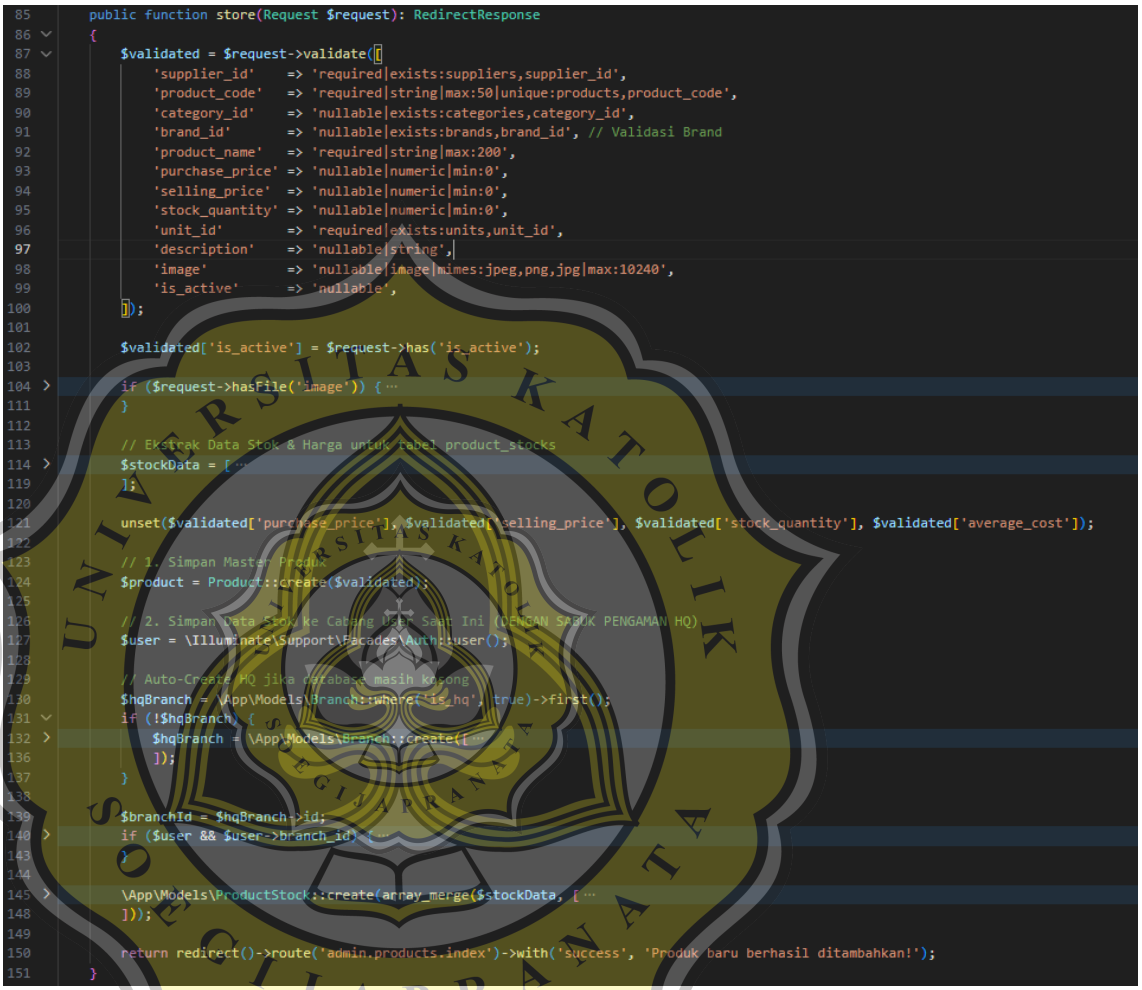
Gambar 4. 45 Form Pendaftaran Produk

Jika di kemudian hari terdapat kesalahan *input* atau penyesuaian data, pengguna dapat memperbaruinya melalui halaman edit produk. Gambar 4.46 di bawah merupakan halaman formulir perbarui data produk. Pengguna dapat mengubah rincian barang di halaman ini. Sesuai dengan batasan akses cabang, sistem memberikan informasi bahwa perubahan pada nominal harga dan kuantitas stok di formulir ini hanya akan berdampak pada lokasi cabang tempat pengguna tersebut sedang bertugas, tanpa merusak data harga di cabang lain.



Gambar 4. 46 Form Edit Produk

Seluruh proses operasional pada halaman katalog barang ini diatur oleh beberapa fungsi yang ada di dalam berkas ProductController, seperti yang ditunjukkan pada Gambar 4.47 dan Gambar 4.48. Dibawah ini merupakan potongan kode penginisialisasian *controller* produk.



```
85  public function store(Request $request): RedirectResponse
86  {
87      $validated = $request->validate([
88          'supplier_id'    => 'required|exists:suppliers,supplier_id',
89          'product_code'   => 'required|string|max:50|unique:products,product_code',
90          'category_id'    => 'nullable|exists:categories,category_id',
91          'brand_id'       => 'nullable|exists:brands,brand_id', // Validasi Brand
92          'product_name'   => 'required|string|max:200',
93          'purchase_price' => 'nullable|numeric|min:0',
94          'selling_price'  => 'nullable|numeric|min:0',
95          'stock_quantity' => 'nullable|numeric|min:0',
96          'unit_id'        => 'required|exists:units,unit_id',
97          'description'    => 'nullable|string',
98          'image'          => 'nullable|image|mimes:jpeg,png,jpg|max:10240',
99          'is_active'      => 'nullable',
100     ]);
101
102     $validated['is_active'] = $request->has('is_active');
103
104     if ($request->hasFile('image')) {
111     }
112
113     // Ekstrak Data Stok & Harga untuk tabel product_stocks
114     $stockData = [
119     ];
120
121     unset($validated['purchase_price'], $validated['selling_price'], $validated['stock_quantity'], $validated['average_cost']);
122
123     // 1. Simpan Master Produk
124     $product = Product::create($validated);
125
126     // 2. Simpan Data Stok ke Cabang User Saat Ini (DENGAN SABUK PENGAMAN HQ)
127     $user = \Illuminate\Support\Facades\Auth::user();
128
129     // Auto-Create HQ jika database masih kosong
130     $hqBranch = \App\Models\Branch::where('is_hq', true)->first();
131     if (!$hqBranch) {
132         $hqBranch = \App\Models\Branch::create([
136         ]);
137     }
138
139     $branchId = $hqBranch->id;
140     if ($user && $user->branch_id) {
143     }
144
145     \App\Models\ProductStock::create(array_merge($stockData, [
148     ]));
149
150     return redirect()->route('admin.products.index')->with('success', 'Produk baru berhasil ditambahkan!');
151 }
```

Gambar 4. 47 Kode Porgram Modul Manajemen Produk

```php
public function show(Product $product): View
{
    $product->load([
        'unit',
        'supplier',
        'category',
        'brand', // Eager Load Brand
        'stockOpnameItems.opname.user',
        'invoiceItems'
    ]);

    // Hitung Statistik (Global / Konsolidasi)
    $totalSold = $product->invoiceItems->sum('quantity');
    $totalRevenue = $product->invoiceItems->sum('subtotal');

    // Hitung Margin Profit berdasarkan Harga dan Stok Cabang Saat ini
    $stockRecord = $product->getCurrentStockRecord();
    $currentSellingPrice = $stockRecord ? $stockRecord->selling_price : 0;
    $currentAvgCost = $stockRecord ? $stockRecord->average_cost : 0;
    $currentStockQty = $stockRecord ? $stockRecord->stock_quantity : 0;

    $marginPerUnit = $currentSellingPrice - $currentAvgCost;
    $marginPercentage = $currentSellingPrice > 0 ? ($marginPerUnit / $currentSellingPrice) * 100 : 0;
    $markupPercentage = $currentAvgCost > 0 ? ($marginPerUnit / $currentAvgCost) * 100 : 100;

    // Potensi Profit dari stok yang ada di cabang ini
    $potentialProfit = $currentStockQty * $marginPerUnit;

    return view('admin.products.show', compact(
        'product',
        'totalSold',
        'totalRevenue',
        'marginPerUnit',
        'marginPercentage',
        'markupPercentage',
        'potentialProfit'
    ));
}
```

Gambar 4. 48 Kode Porgram Modul Manajemen Produk

Pada bagian awal *controller* tersebut, terdapat sistem *middleware* yang bertindak sebagai penjaga gerbang. *Middleware* ini memastikan bahwa aksi seperti melihat (can:view-products), menambah (can:create-products), mengedit (can:edit-products), hingga menghapus barang (can:delete-products) hanya bisa dieksekusi oleh *user* yang memiliki *permission* tersebut.

Beberapa fungsi utama di dalam *controller* ini diantaranya adalah function index untuk memuat daftar barang dengan mekanisme *filter*, function store untuk memvalidasi masukan formulir dan mengompres ukuran gambar secara otomatis, serta function update untuk menyimpan perubahan data. Selain itu, terdapat pengamanan data berlapis pada *function destroy* untuk proses pengarsipan (hapus sementara). Sistem diprogram untuk otomatis menolak permohonan hapus barang apabila sistem mendeteksi bahwa barang tersebut masih memiliki sisa stok fisik yang lebih dari nol, atau jika

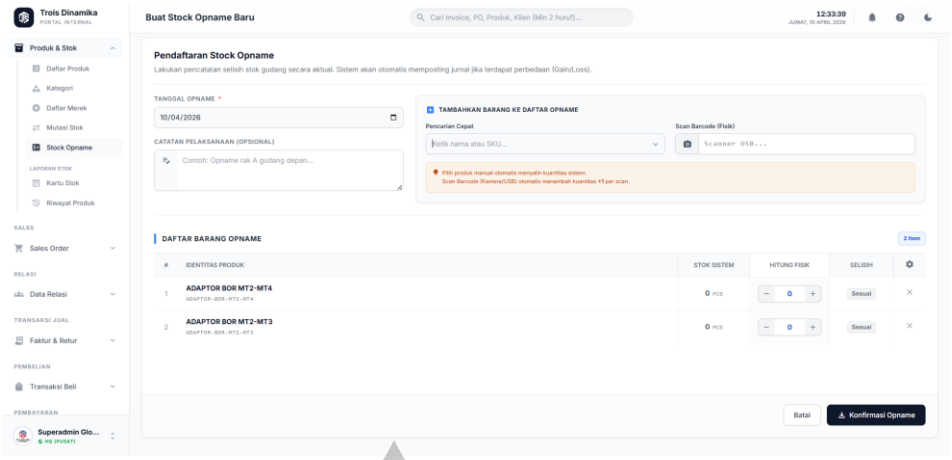
barang tersebut masih tersangkut di dalam transaksi penjualan (*invoice*) dan pembelian (p*urchase order*) yang belum selesai diproses.

4.3.5 Modul Verifikasi Stok Fisik

Modul *Stock Opname* berfungsi untuk menyamakan catatan jumlah barang di dalam sistem dengan kondisi fisik yang sebenarnya di gudang. Tampilan awal modul ini adalah daftar riwayat pencocokan stok yang dapat dilihat pada Gambar 4.49. Melalui halaman ini, petugas gudang dapat mencetak lembar kerja fisik atau menekan tombol untuk memulai proses opname baru.

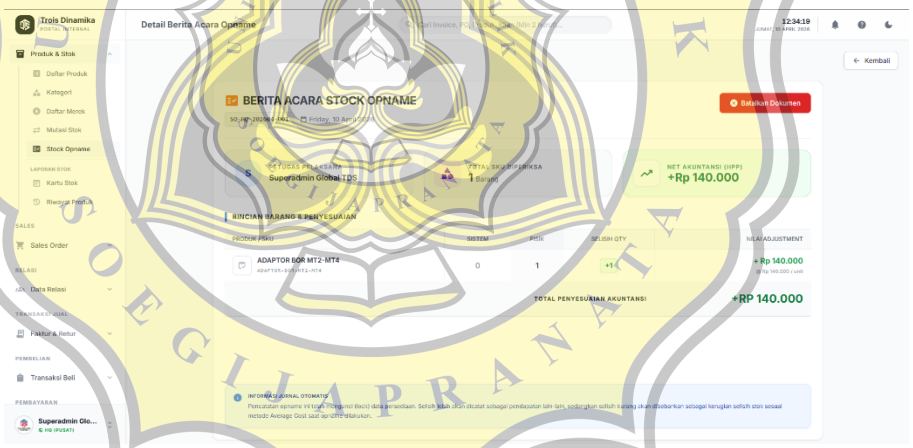
Gambar 4. 49 Halaman Utama Modul Stock Opnname

Apabila pengguna memilih untuk membuat opname baru, sistem akan memuat formulir pendaftaran seperti pada Gambar 4.50. Pada antarmuka ini, proses pendataan dibuat sangat praktis. Petugas cukup memindai barcode fisik menggunakan scanner atau mencari nama produk secara manual, kemudian memasukkan jumlah fisik yang ditemukan di lapangan. Sistem nantinya akan otomatis menghitung selisih kuantitasnya.



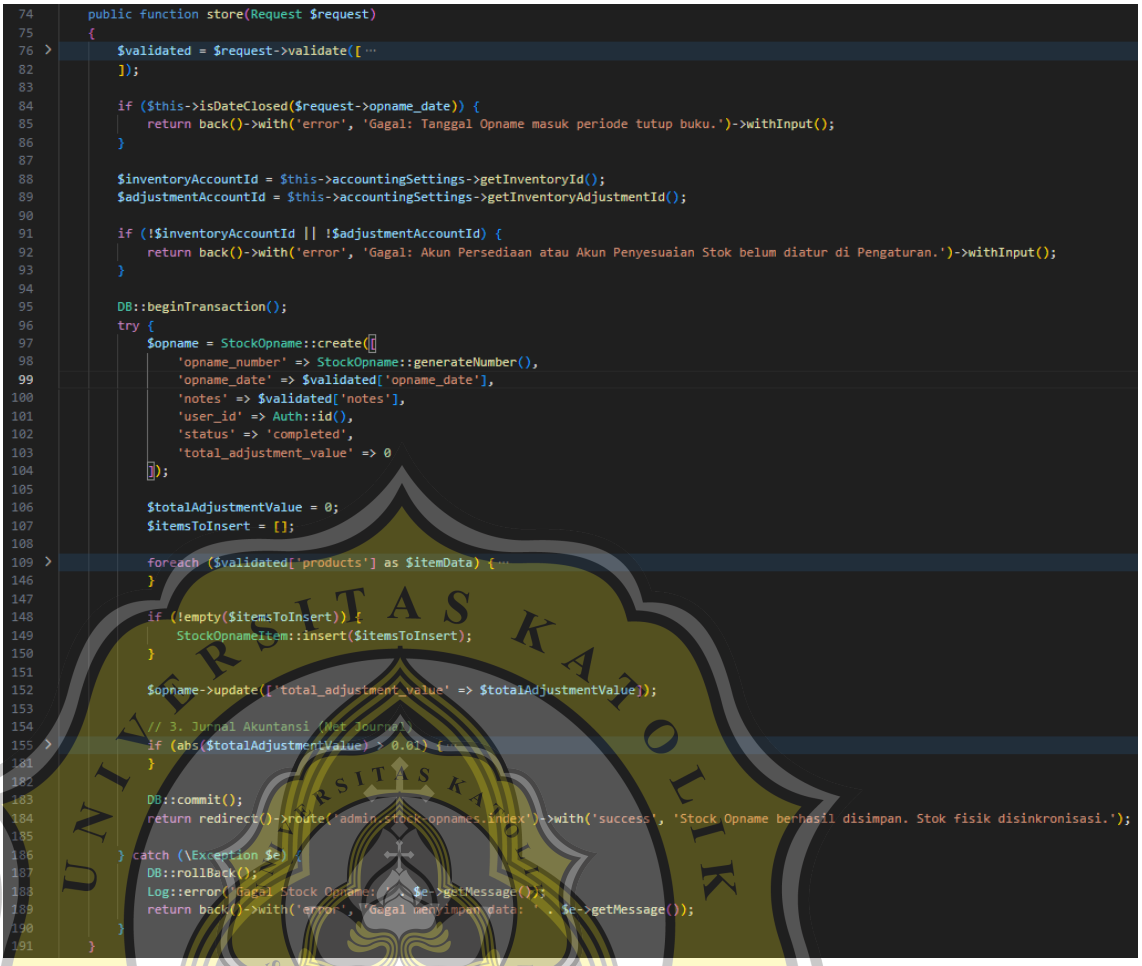
Gambar 4. 50 Form Pembuatan *Stock Opname*

Setelah formulir dikonfirmasi, sistem akan menerbitkan dokumen Berita Acara seperti yang ditunjukkan pada Gambar 4.51. Halaman detail ini merangkum total SKU yang diperiksa sekaligus mengkalkulasi nilai rupiah dari penyesuaian akuntansinya, baik itu berstatus penambahan (*gain*) maupun penyusutan (*loss*).



Gambar 4. 51 Halaman Detail Stock Opname

Seluruh proses pencatatan fisik ini dikendalikan oleh modul StockOpnameController. Potongan kode inisialisasinya dapat dilihat pada Gambar 4.52. Keamanan akses modul ini juga diatur oleh *middleware*, sehingga hanya staf dengan izin khusus yang bisa memanipulasi data stok fisik.



```
74    public function store(Request $request)
75    {
76        $validated = $request->validate([ ···
82        ]);
83
84        if ($this->isDateClosed($request->opname_date)) {
85            return back()->with('error', 'Gagal: Tanggal Opname masuk periode tutup buku.')->withInput();
86        }
87
88        $inventoryAccountId = $this->accountingSettings->getInventoryId();
89        $adjustmentAccountId = $this->accountingSettings->getInventoryAdjustmentId();
90
91        if (!$inventoryAccountId || !$adjustmentAccountId) {
92            return back()->with('error', 'Gagal: Akun Persediaan atau Akun Penyesuaian Stok belum diatur di Pengaturan.')->withInput();
93        }
94
95        DB::beginTransaction();
96        try {
97            $opname = StockOpname::create([
98                'opname_number' => StockOpname::generateNumber(),
99                'opname_date' => $validated['opname_date'],
100               'notes' => $validated['notes'],
101               'user_id' => Auth::id(),
102               'status' => 'completed',
103               'total_adjustment_value' => 0
104           ]);
105
106           $totalAdjustmentValue = 0;
107           $itemsToInsert = [];
108
109           foreach ($validated['products'] as $itemData) { ···
146           }
147
148           if (!empty($itemsToInsert)) {
149               StockOpnameItem::insert($itemsToInsert);
150           }
151
152           $opname->update(['total_adjustment_value' => $totalAdjustmentValue]);
153
154           // 3. Jurnal Akuntansi (Net Journal)
155           if (abs($totalAdjustmentValue) > 0.01) { ···
181           }
182
183           DB::commit();
184           return redirect()->route('admin.stock-opnames.index')->with('success', 'Stock Opname berhasil disimpan. Stok fisik disinkronisasi.');
185
186       } catch (\Exception $e) {
187           DB::rollBack();
188           Log::error('Gagal Stock Opname: ' . $e->getMessage());
189           return back()->with('error', 'Gagal menyimpan data: ' . $e->getMessage());
190       }
191   }
```

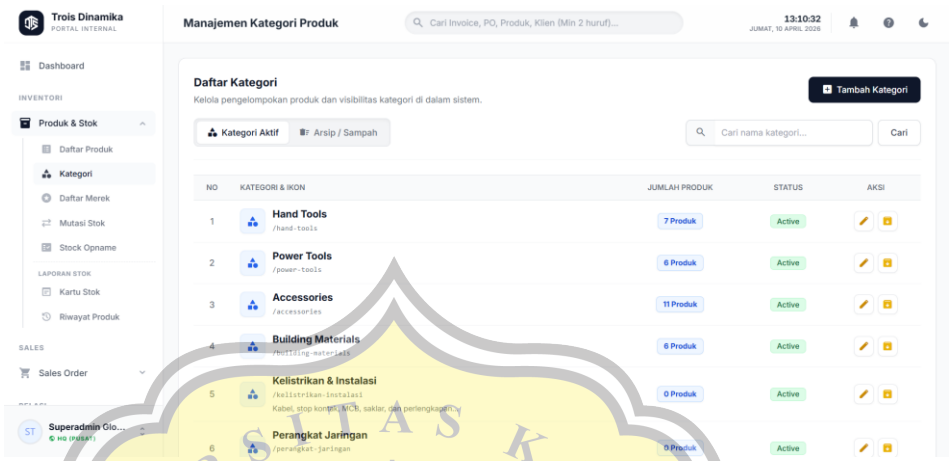
Gambar 4. 52 Kode Program Stock Opname

Terdapat beberapa fungsi krusial di balik modul ini. Misalnya pada f*unction store*, sistem diprogram agar tidak sekadar mengubah angka persediaan di database. Fungsi tersebut secara otomatis memanggil layanan akuntansi (AccountingService) untuk membuat jurnal otomatis berdasarkan nilai HPP (Harga Pokok Penjualan) barang yang berselisih. Selain itu, terdapat fitur mitigasi pada *function destroy*. Jika sebuah dokumen opname dibatalkan, sistem tidak akan menghapus datanya secara permanen, melainkan menerbitkan jurnal pembalik *(reversal)*. Hal ini bertujuan agar siklus audit dan rekam jejak keuangan perusahaan tetap terjaga integritasnya.

4.3.6 Modul Pengaturan Kategori

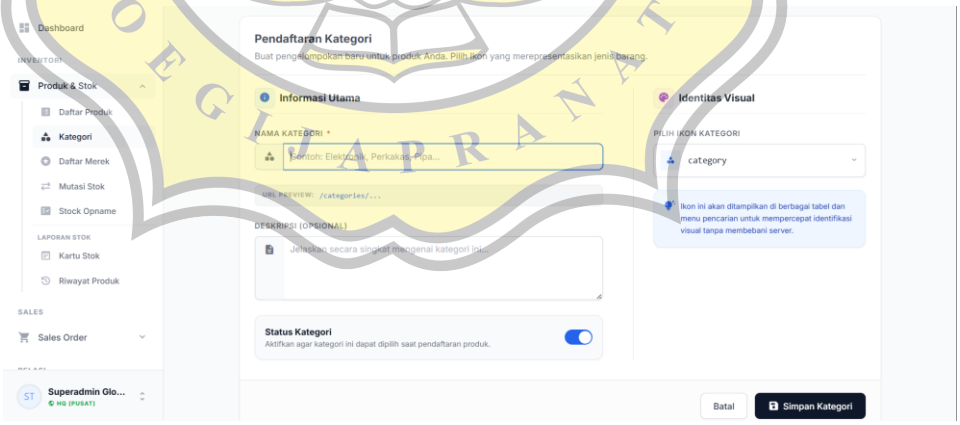
Modul Manajemen Kategori berfungsi sebagai alat untuk mengelompokkan produk agar lebih terstruktur dan mudah dicari. Antarmuka utama modul ini menampilkan tabel daftar kelompok barang

seperti pada Gambar 4.53. Melalui halaman ini, pengguna dapat memantau status kategori yang aktif sekaligus melihat ringkasan jumlah barang yang terikat pada masing-masing kategori tersebut.



Gambar 4. 53 Halaman Utama Manajemen Kategori

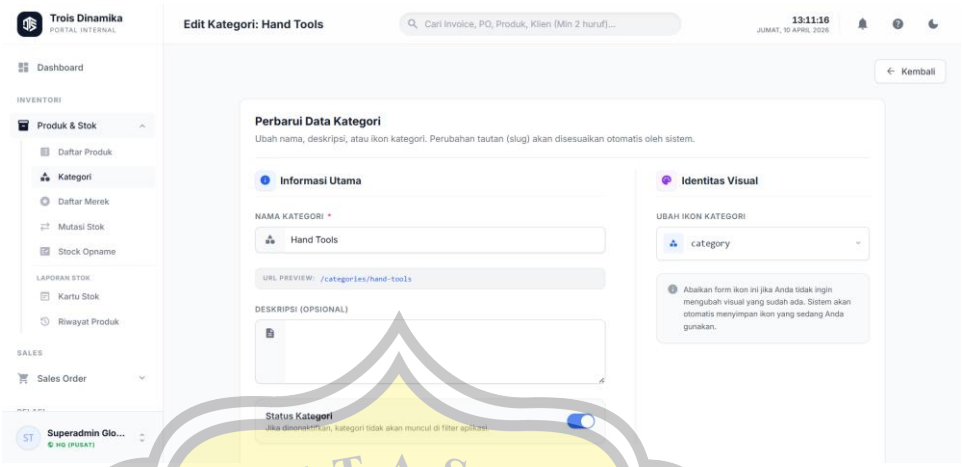
Jika pengguna ingin mendaftarkan kelompok barang baru, sistem akan menampilkan formulir pendaftaran kategori seperti yang ditunjukkan pada Gambar 4.54. Untuk mengoptimalkan kinerja aplikasi dan menghemat ruang penyimpanan *server*, identitas visual pada formulir ini tidak menggunakan sistem unggah gambar fisik. Pengguna cukup memilih ikon dari daftar pustaka (*library*) ikon yang sudah terintegrasi di dalam sistem.



Gambar 4. 54 Halaman Tambah Kategori

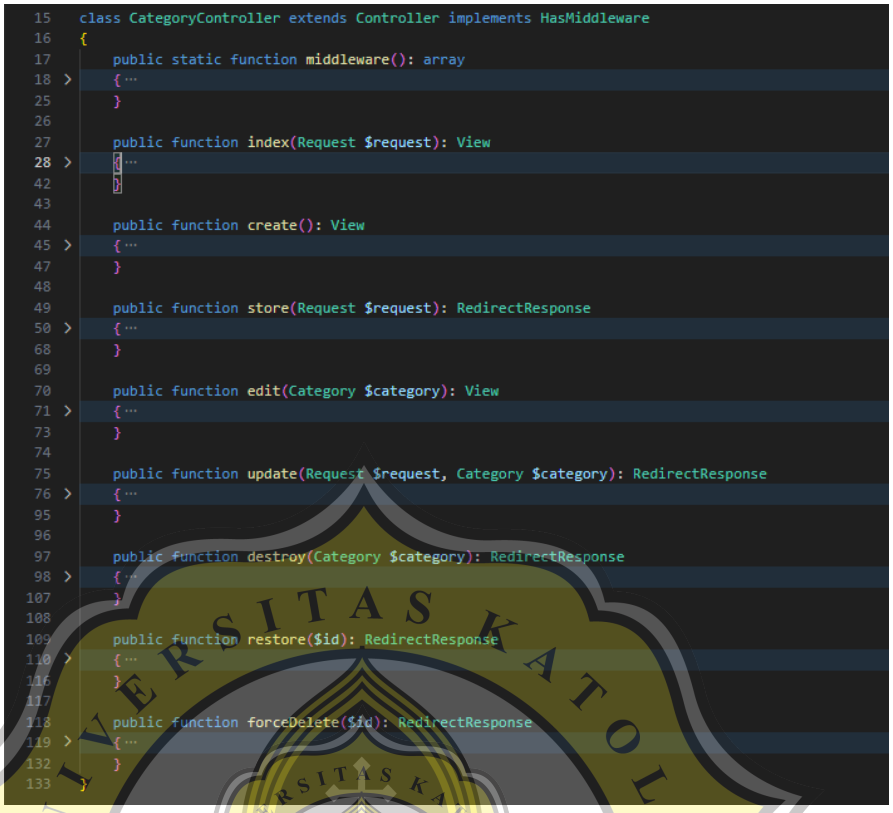
Sementara itu, apabila terdapat kesalahan pengetikan atau pembaruan identitas visual, pengguna dapat menyesuaikannya melalui formulir edit kategori pada Gambar 4.55. Ketika nama kategori diubah, sistem akan

memberikan pratinjau tautan (*URL Preview*) di bawah kolom nama yang menandakan bahwa URL tersebut akan otomatis diperbarui.



Gambar 4. 55 Halaman Edit Kategori

Seluruh alur pemrosesan data tersebut dikendalikan oleh modul CategoryController. Potongan kode inisialisasinya dapat dilihat pada Gambar 4.56. Sama seperti modul master data lainnya, sistem ini menerapkan lapisan keamanan *middleware* di awal kode untuk memverifikasi apakah pengguna yang mengakses memiliki hak (*role*) yang sesuai.

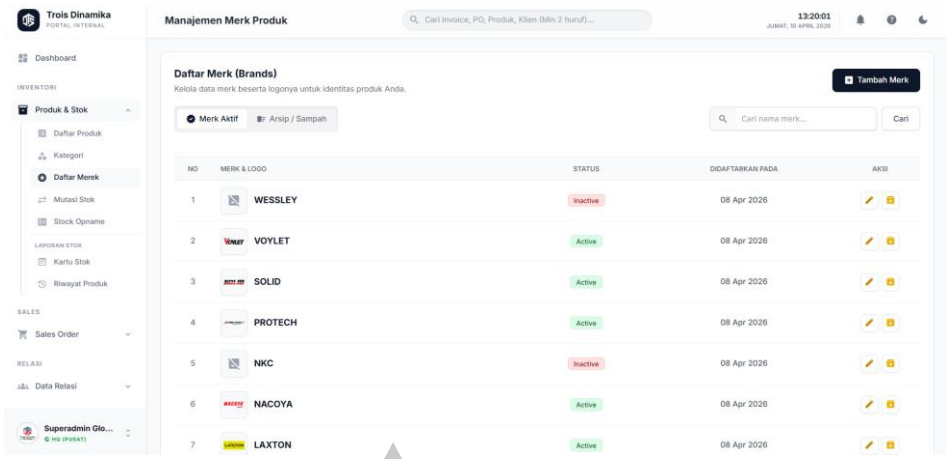


Gambar 4. 56 Kode Program Modul Kategori

Pada proses di balik layar, salah satu mekanisme otomatis yang berjalan adalah pembuatan tautan ramah mesin pencari (*SEO-friendly URL*) menggunakan perintah Str::slug yang disisipkan pada *function store* dan *update*. Selain itu, demi menjaga integritas basis data, sistem menerapkan validasi ketat pada *function destroy*. Apabila seorang pengguna mencoba menghapus sebuah kategori, sistem akan mengecek relasi datanya terlebih dahulu. Jika kategori tersebut masih digunakan oleh sebuah produk sekalipun produk tersebut sedang berada di keranjang sampah (*soft deleted*) maka permohonan hapus akan langsung ditolak untuk menghindari terjadinya data yatim piatu (*orphan data*).

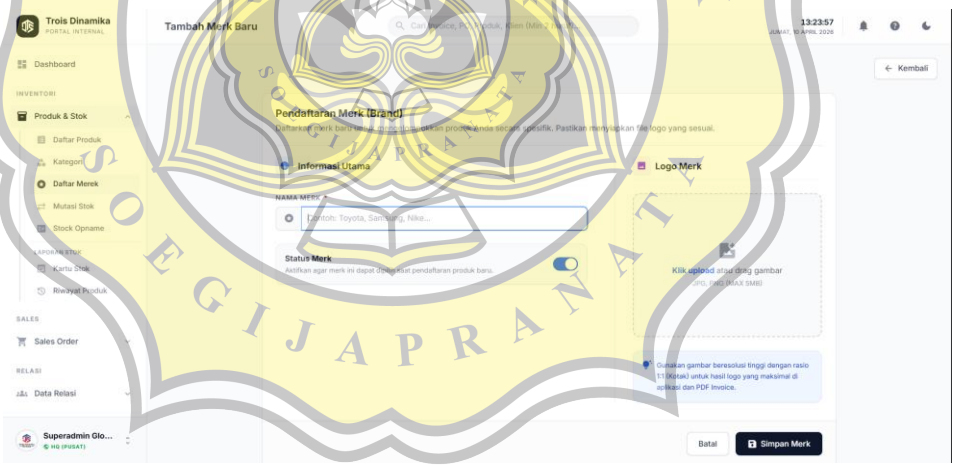
4.3.7 Modul Pengaturan Merk

Modul Manajemen Merk disediakan untuk mengakomodasi pengelompokan produk berdasarkan pabrikan atau prinsipalnya. Pada antarmuka awal modul ini, sistem menampilkan tabel daftar merek lengkap dengan identitas logo, status keaktifan, dan tanggal pendaftaran seperti yang ditunjukkan pada Gambar 4.57.



Gambar 4. 57 Halaman Utama Modul Brand

Untuk mendaftarkan merek baru, staf dapat menekan tombol tambah data yang akan mengarahkan mereka ke formulir pendaftaran pada Gambar 4.58. Di sini, pengguna diminta mengisi nama merek dan mengunggah *file* gambar logo. Terdapat sebuah catatan kecil (*tooltip*) di antarmuka tersebut yang menyarankan pengguna untuk mengunggah gambar dengan rasio kotak (1:1) demi hasil visual yang maksimal.



Gambar 4. 58 Form Tambah Brand

Alur pemrosesan dan penyimpanan data untuk modul ini dikelola secara terpusat pada Modul BrandController, yang sebagian potongan kodenya dapat dilihat pada Gambar 4.59

```php
public function index(Request $request): View
{
    $query = Brand::query();

    // Tampilkan data yang diarsipkan (Soft Deletes)
    if ($request->get('status') === 'trash') {
        $query->onlyTrashed();
    }

    // Fitur Pencarian
    if ($request->filled('search')) {
        $query->where('name', 'like', '%' . $request->search . '%');
    }

    $brands = $query->latest('brand_id')->paginate(10)->appends($request->query());

    return view('admin.brands.index', compact('brands'));
}

public function create(): View
{
    return view('admin.brands.create');
}

public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'name' => 'required|string|max:100|unique:brands,name',
        'logo' => 'nullable|image|mimes:png,jpg,jpeg,svg|max:5120',
        'is_active' => 'nullable|boolean',
    ]);

    $data = [
        'name' => $validated['name'],
        'is_active' => $request->has('is_active'),
    ];

    if ($request->hasFile('logo')) {
        $image = $request->file('logo');
        $filename = 'brands/' . uniqid() . '.' . $image->getClientOriginalExtension();

        $img = Image::read($image);
        $img->scaleDown(800, 800);

        Storage::disk('public')->put($filename, (string) $img->encode());
        $data['logo_path'] = $filename;
    }

    Brand::create($data);

    return redirect()->route('admin.brands.index')->with('success', 'Merk/Brand berhasil ditambahkan.');
}
```

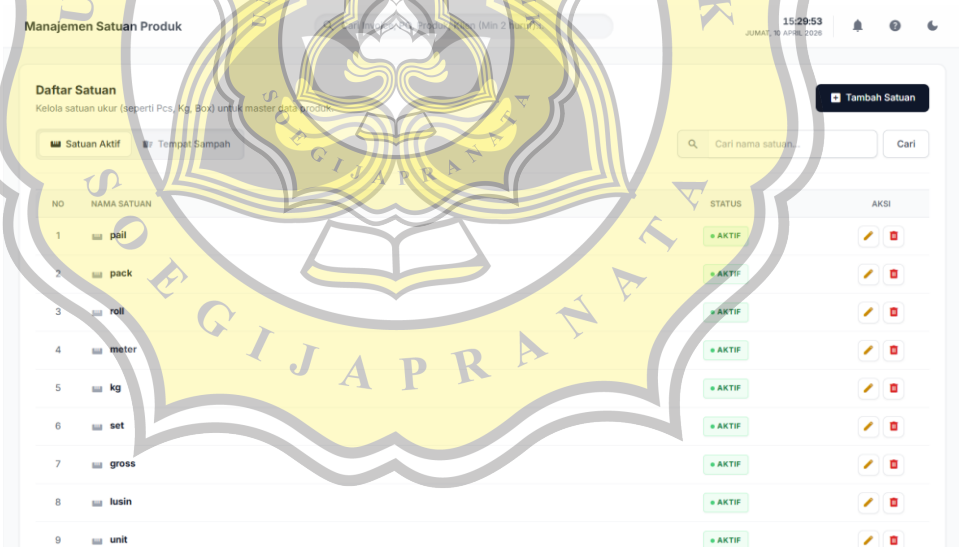
Gambar 4. 59 Kode Program Modul Brand/ Merk

Pada tahap rekam data *(store)* dan pembaruan *(update)*, modul ini tidak sekadar memindahkan berkas gambar mentah yang diunggah pengguna ke dalam *database*. Apabila sistem mendeteksi adanya masukan gambar melalui kondisi if ($request->hasFile('logo')), sistem akan memanggil pustaka *Intervention Image* untuk memproses logo tersebut. Gambar kemudian disesuaikan dimensinya menggunakan instruksi scaleDown(800, 800). Logika pemrograman ini membatasi resolusi maksimal gambar di angka 800 piksel, namun secara otomatis tetap mempertahankan rasio aspek aslinya agar tampilan visual logo tidak *stretched.*

Selain manipulasi dimensi gambar, modul ini juga dibekali dengan pengaturan ruang penyimpanan *server* (*storage management*). Ketika pengguna memperbarui logo dengan gambar yang baru pada *fungsi update*, atau ketika data merek dihapus secara permanen melalui fungsi *forceDelete*, sistem akan mengeksekusi perintah Storage::disk('public')->delete($brand->logo_path). Instruksi ini bertugas mencari dan melenyapkan fail gambar fisik yang lama dari *server* untuk mencegah penumpukan fail sampah yang dapat menghabiskan kapasitas penyimpanan perusahaan.

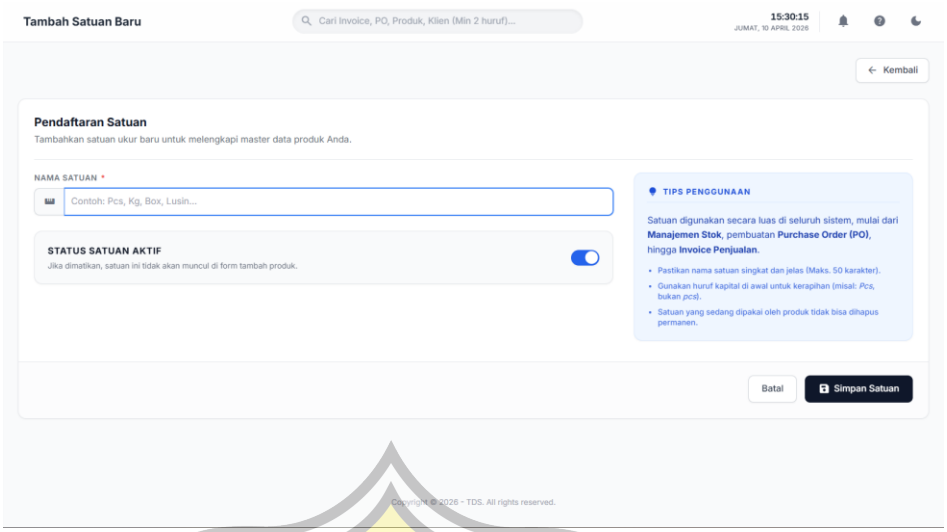
4.3.8 Modul Pengaturan Satuan Barang

Modul Manajemen Satuan difungsikan untuk mendata standar ukuran kuantitas fisik barang, seperti kilogram (Kg), meter, gulung (*roll*), maupun boks. Tampilan utama modul ini memuat tabel daftar satuan yang terdaftar di dalam sistem, seperti yang ditunjukkan pada Gambar 4.60. Pada halaman ini, pengguna dapat mencari nama satuan atau meninjau data yang telah diarsipkan melalui navigasi tab di bagian atas tabel.



Gambar 4. 60 Daftar Unit/ Satuan Barang

Untuk menambahkan standar ukuran yang baru, staf dapat mengakses formulir pendaftaran pada Gambar 4.61. Antarmuka ini dirancang minimalis, di mana pengguna hanya diminta untuk memasukkan nama satuan dan mengatur status aktifnya melalui sebuah tombol sakelar (*toggle*).



Gambar 4. 61 Form Tambah Unit/ Satuan

Seluruh pemrosesan input dari antarmuka tersebut dikelola oleh UnitController. Potongan kode inisialisasinya dapat dilihat pada Gambar 4.62.

```
26      public function index(Request $request): View
27      {
28          $query = Unit::query();
29
30          if ($request->get('status') === 'trash') {
31              $query->onlyTrashed();
32          }
33
34          if ($request->filled('search')) {
35              $query->where('name', 'like', '%' . $request->search . '%');
36          }
37
38          $units = $query->latest('unit_id')->paginate(10)->appends($request->query());
39
40          return view('admin.units.index', compact('units'));
41      }
42
43      public function create(): View
44      {
45          return view('admin.units.create');
46      }
47      Ctrl+L for Agent
48      public function store(Request $request): RedirectResponse
49      {
50          $validated = $request->validate([
51              'name' => 'required|string|max:50|unique:units,name',
52              'is_active' => 'nullable|boolean',
53          ]);
54
55          $validated['is_active'] = $request->has('is_active');
56
57          Unit::create($validated);
58
59          return redirect()->route('admin.units.index')->with('success', 'Satuan baru berhasil ditambahkan.');
60      }
61
```

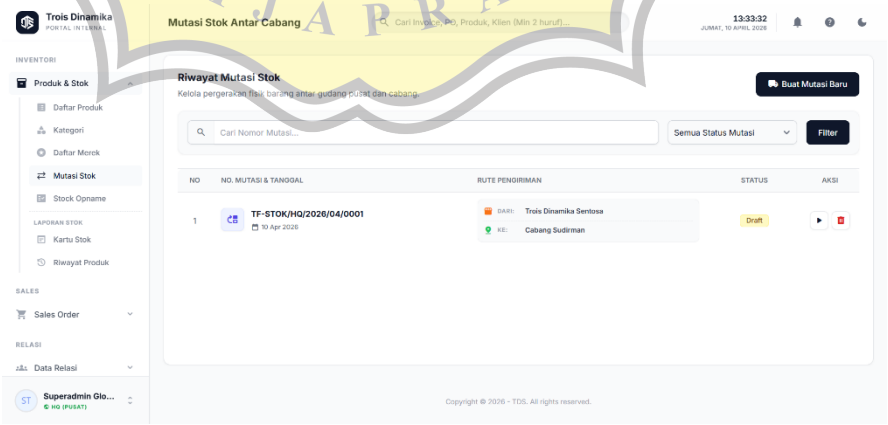
Gambar 4. 62 Kode Program Modul Unit

Pada proses rekam data di *public function store* dan pembaruan di *public function update*, sistem memvalidasi masukan teks menggunakan aturan unique:units,name untuk memastikan tidak ada duplikasi nama satuan di dalam basis data. Selanjutnya, status sakelar dari formulir dievaluasi melalui metode $request->has('is_active'). Metode ini akan mengonversi masukan tersebut menjadi nilai *boolean* (benar atau salah) sebelum akhirnya ditampung ke dalam variabel $validated dan disimpan secara permanen.

Aspek paling krusial dari *controller* ini terletak pada perlindungan integritas datanya. Saat pengguna mencoba memindahkan data ke arsip *melalui public function destroy* atau menghapusnya secara permanen melalui *public function forceDelete*, sistem tidak akan langsung mengeksekusi perintah tersebut. Sistem akan terlebih dahulu menjalankan kueri $unit->products()->withTrashed()->count() > 0. Logika ini bertugas memblokir penghapusan apabila satuan tersebut masih terikat dengan data master produk mana pun, termasuk produk yang posisinya sudah dibuang ke tempat sampah (*soft deleted*). Mekanisme ini diterapkan untuk memastikan detail satuan pada dokumen riwayat transaksi di masa lalu tidak akan mengalami *error* akibat hilangnya data referensi.

4.3.9 Modul Mutasi Barang

Seluruh jejak historis pemindahan barang ini dapat dipantau melalui antarmuka daftar riwayat mutasi pada Gambar 4.63

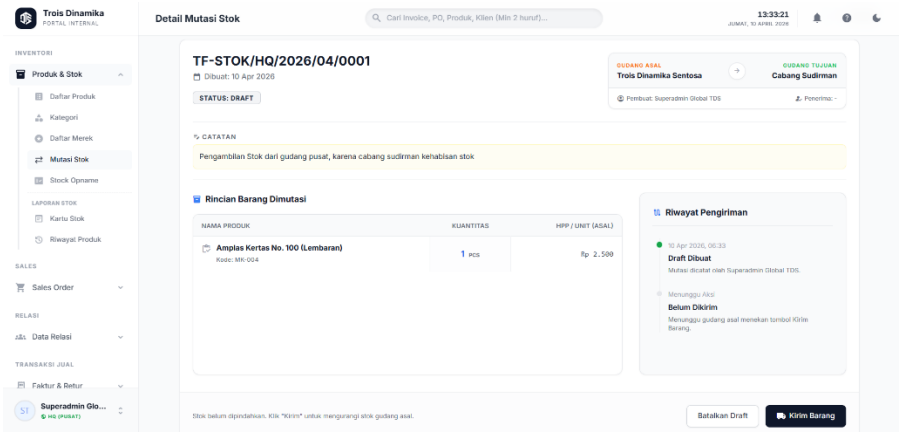


Gambar 4. 63 Halaman Mutasi Stok atau Stock Transfer

Modul Mutasi Stok difungsikan untuk melacak pergerakan fisik barang antara gudang pusat *(HQ)* dan gudang cabang, atau antar sesama gudang cabang. Modul ini dirancang menggunakan alur persetujuan ganda (*dual-step verification*) untuk meminimalisasi selisih pencatatan. Pada halaman Gambar 4.64, pengguna dapat melihat antarmuka formulir pembuatan dokumen mutasi baru. Melalui formulir ini, staf harus menentukan lokasi gudang asal, lokasi gudang penerima, dan mendata produk apa saja yang akan dipindahkan.

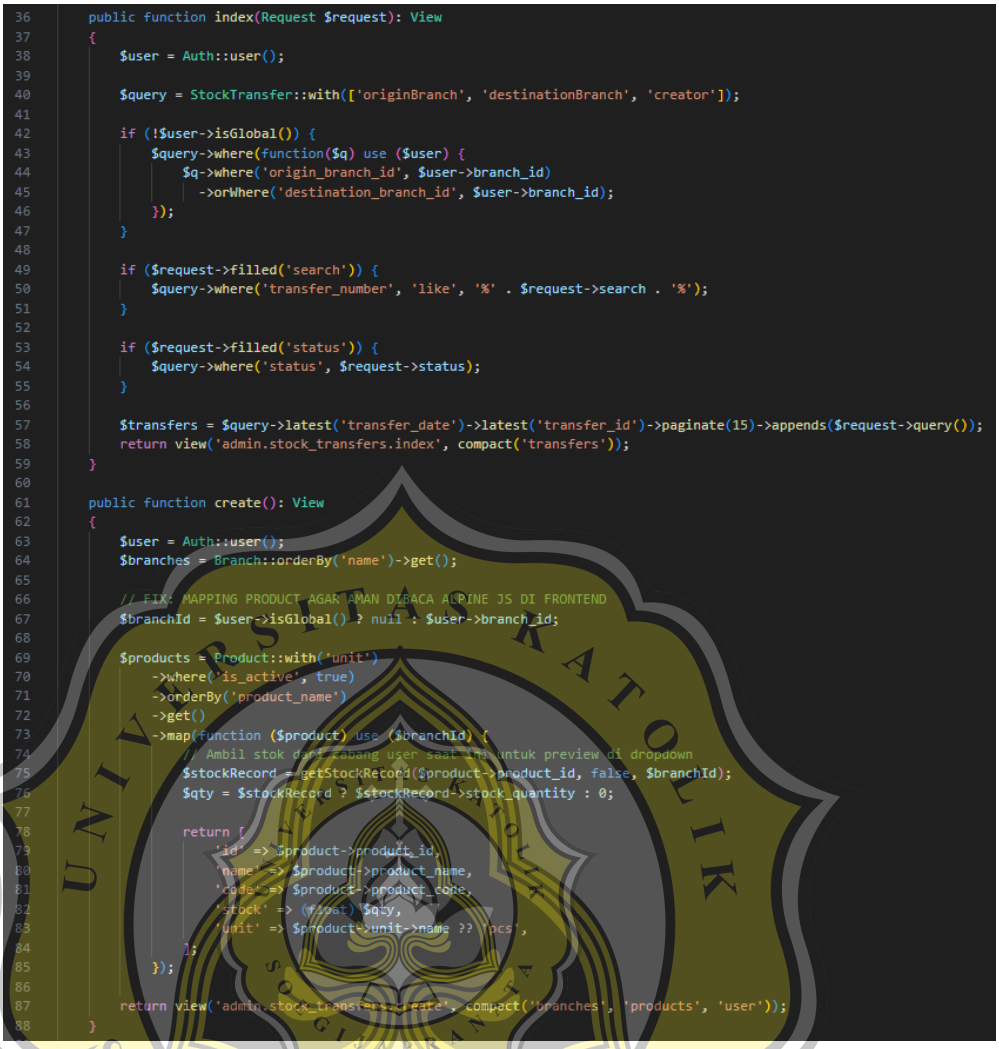
Gambar 4. 64 Form Pembuatan Mutasi Stok

Setelah dokumen mutasi dibuat, statusnya tidak langsung memotong jumlah persediaan, melainkan tersimpan sebagai *draft* seperti yang terlihat pada Gambar 4.65. Pada halaman detail ini, terdapat tombol khusus untuk mengubah status pengiriman. Ketika pihak gudang asal menekan tombol "Kirim Barang", status dokumen akan berubah menjadi sedang di perjalanan (*In Transit*). Pada detik itu juga, sistem akan mengunci dan mengurangi kuantitas stok di gudang asal.



Gambar 4. 65 Detail Mutasi Stok Yang Telah Dilakukan

Di balik layar, pemrosesan pergerakan persediaan ini ditangani oleh berkas pengontrol StockTransferController, yang potongan kodenya dapat dilihat pada Gambar 4.66.



```php
public function index(Request $request): View
{
    $user = Auth::user();

    $query = StockTransfer::with(['originBranch', 'destinationBranch', 'creator']);

    if (!$user->isGlobal()) {
        $query->where(function($q) use ($user) {
            $q->where('origin_branch_id', $user->branch_id)
              ->orWhere('destination_branch_id', $user->branch_id);
        });
    }

    if ($request->filled('search')) {
        $query->where('transfer_number', 'like', '%' . $request->search . '%');
    }

    if ($request->filled('status')) {
        $query->where('status', $request->status);
    }

    $transfers = $query->latest('transfer_date')->latest('transfer_id')->paginate(15)->appends($request->query());
    return view('admin.stock_transfers.index', compact('transfers'));
}

public function create(): View
{
    $user = Auth::user();
    $branches = Branch::orderBy('name')->get();

    // FIX: MAPPING PRODUCT AGAR AMAN DIBACA ALPINE JS DI FRONTEND
    $branchId = $user->isGlobal() ? null : $user->branch_id;

    $products = Product::with('unit')
        ->where('is_active', true)
        ->orderBy('product_name')
        ->get()
        ->map(function ($product) use ($branchId) {
            // Ambil stok dari cabang user saat ini untuk preview di dropdown
            $stockRecord = getStockRecord($product->product_id, false, $branchId);
            $qty = $stockRecord ? $stockRecord->stock_quantity : 0;

            return [
                'id' => $product->product_id,
                'name' => $product->product_name,
                'code' => $product->product_code,
                'stock' => (float) $qty,
                'unit' => $product->unit->name ?? 'pcs',
            ];
        });

    return view('admin.stock_transfers.create', compact('branches', 'products', 'user'));
}
```

Gambar 4. 66 Kode Program dari Modul Stock Transfer

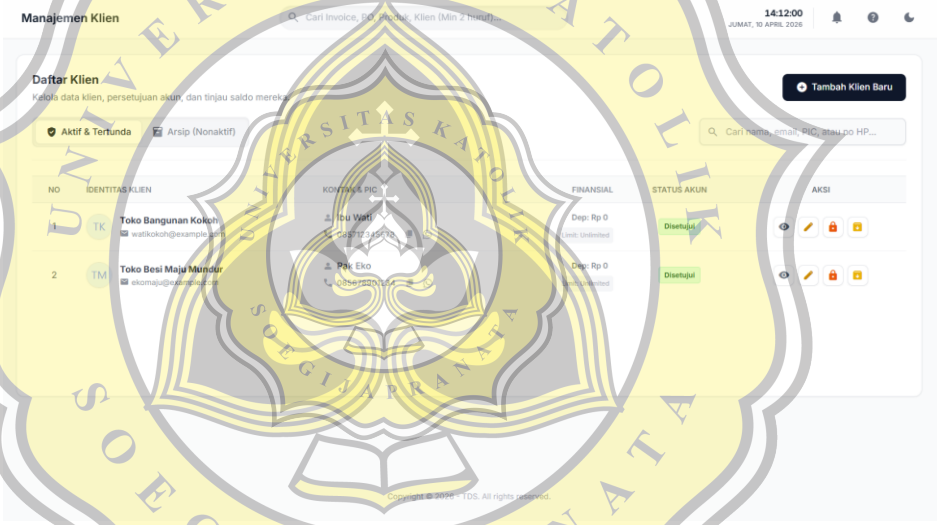
Logika inti dari perpindahan barang ini terletak pada dua fungsi utama, yaitu *function send* dan *function receive*. Pada saat function send dieksekusi, sistem tidak hanya mengurangi jumlah persediaan fisik di gudang pengirim, tetapi juga merekam harga pokok (HPP) barang tersebut pada saat itu.

Selanjutnya, ketika staf di gudang tujuan menerima barang menggunakan *function receive*, sistem akan mengeksekusi perhitungan yang kompleks. Sistem tidak sekadar menambahkan jumlah fisik barang (incomingQty) ke etalase cabang penerima, tetapi juga secara dinamis menghitung ulang nilai *Average Cost* (Biaya Rata-Rata) di gudang tersebut. Rumus matematika yang ditanamkan pada fungsi penerimaan ini akan melebur total nilai aset barang yang lama dengan total nilai aset barang yang baru masuk, lalu

membaginya dengan total kuantitas yang baru. Hal ini memastikan bahwa laporan valuasi aset perusahaan tetap seimbang dan akurat meskipun terjadi perputaran barang antar wilayah.

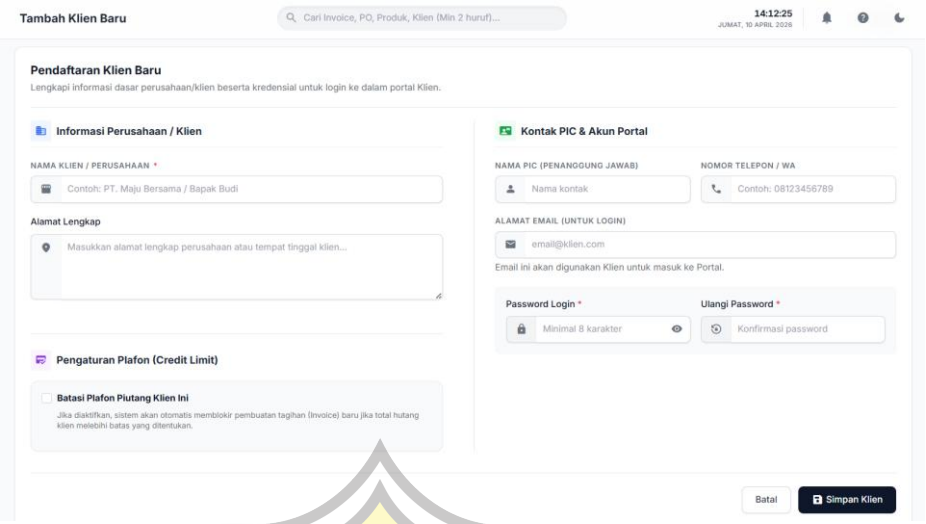
4.3.10 Modul Manajemen Klien

Modul Manajemen Klien bertindak sebagai pusat basis data untuk mengelola informasi perusahaan mitra, kredensial akses portal, hingga pengaturan batas hutang (plafon piutang). Pada Gambar 4.67 di bawah, ditampilkan antarmuka daftar klien yang menyajikan rangkuman informasi esensial. Melalui tabel ini, admin dapat memantau nama entitas klien, detail kontak Penanggung Jawab (PIC), ringkasan finansial, serta status persetujuan akun mereka di dalam sistem.



Gambar 4. 67 Halaman Daftar Klien

Untuk mendaftarkan mitra bisnis yang baru, pengguna dapat mengakses formulir pendaftaran seperti yang diperlihatkan pada Gambar 4.68. Formulir ini tidak hanya meminta identitas dasar perusahaan, tetapi juga terintegrasi dengan pembuatan akun Portal Klien. Oleh karena itu, staf harus memasukkan alamat email dan *password* yang nantinya akan digunakan oleh klien untuk login. Di bagian bawah formulir, terdapat juga sakelar (*toggle*) khusus untuk mengaktifkan fitur batasan plafon piutang.



Gambar 4. 68 Form Tambah Klien

Alur penyimpanan dan manipulasi data klien ini diproses melalui berkas ClientController. Potongan kode inisialisasi dari *controller* tersebut ditunjukkan pada Gambar 4.69.

```
58      public function store(Request $request): RedirectResponse
59      {
60          $validated = $request->validate([
61              'client_name' => 'required|string|max:150',
62              'email' => 'nullable|string|email|max:100|unique:clients,email',
63              'password' => ['required', 'confirmed', Password::defaults()],
64              'person_in_charge' => 'nullable|string|max:100',
65              'address' => 'nullable|string',
66              'phone_number' => 'nullable|string|max:20',
67              'has_credit_limit' => 'nullable|boolean',
68              'credit_limit' => 'nullable|numeric|min:0',
69          ]);
70          $validated['password'] = Hash::make($validated['password']);
71          $validated['is_approved'] = false;
72          $validated['has_credit_limit'] = $request->boolean('has_credit_limit');
73          if (!$validated['has_credit_limit']) {
74              $validated['credit_limit'] = 0;
75          }
76          Client::create($validated);
77          return redirect()->route('admin.clients.index')->with('success', 'Klien baru berhasil ditambahkan.');
78      }
79      public function show(Client $client): View
80      {...
82      }
83      public function getTabContent(Request $request, Client $client)
84      {...
119     }
120     public function edit(Client $client): View
121     {...
123     }
124     public function update(Request $request, Client $client): RedirectResponse
125     {...
152     }
153     public function destroy(Client $client): RedirectResponse
154     {...
157     }
158     public function approve(Client $client): RedirectResponse
159     {
160         $client->update(['is_approved' => true]);
161         return back()->with('success', 'Akun klien ' . $client->client_name . ' telah disetujui.');
162     }
163
164     public function restore(Client $client): RedirectResponse
165     {
166         if ($client->trashed()) {
167             $client->restore();
168             return back()->with('success', 'Akun klien ' . $client->client_name . ' telah dipulihkan.');
169         }
170
171         return back()->with('error', 'Klien tidak terhapus.');
172     }
```

Gambar 4. 69 Kode Program Modul Klien

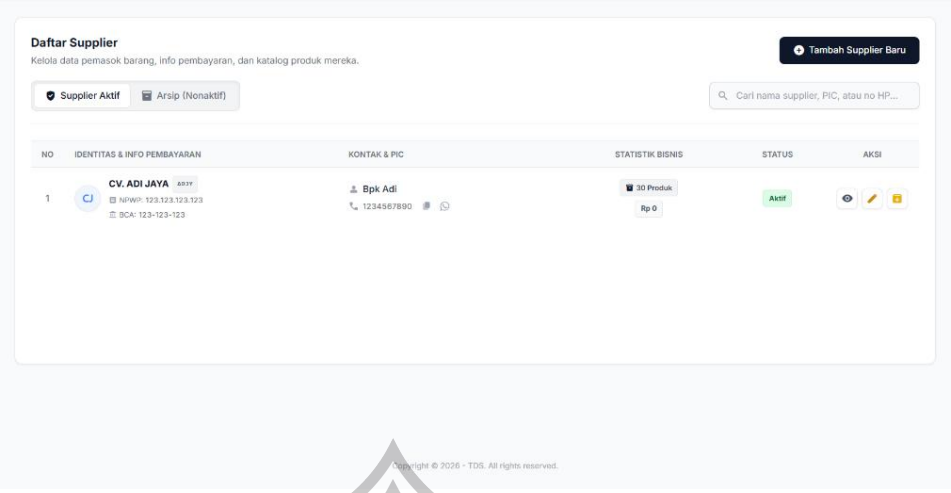
Proses rekam data klien baru ditangani secara spesifik oleh *public function store(Request $request)*. Ketika formulir dikirim, seluruh isian akan divalidasi dan ditampung ke dalam sekumpulan *array* pada variabel $validated. Demi alasan keamanan siber, *password* mentah yang diinputkan oleh staf tidak akan langsung disimpan. Sistem akan memanggil fungsi Hash::make($validated['password']) untuk mengenkripsi kata sandi tersebut menjadi teks acak (*ciphertext*).

Selain itu, untuk mencegah penyalahgunaan akses, sistem secara otomatis menanamkan nilai $validated['is_approved'] = false. Ini berarti setiap klien baru tidak bisa langsung menggunakan portal sebelum akunnya ditinjau ulang. Terkait fitur finansial, sistem akan mengevaluasi masukan pengguna melalui variabel $validated['has_credit_limit']. Jika staf tidak mencentang opsi batasan plafon ($request->boolean('has_credit_limit') bernilai salah), maka sistem akan memaksa nilai $validated['credit_limit'] = 0 untuk memastikan tidak ada limit bayangan yang tersimpan di basis data.

Modul ini juga dibekali beberapa fungsi kontrol akses seperti *public function approve(), public function lock(),* dan *public function unlock().* Fungsi-fungsi ini bertugas untuk mengubah nilai *boolean* is_approved dan is_locked di dalam *database*, yang secara langsung akan mencabut atau memberikan kembali hak akses klien ke dalam portal. Menariknya, untuk mengoptimalkan kinerja halaman saat melihat detail klien, *controller* ini menggunakan public function getTabContent(). Fungsi ini bekerja secara dinamis (*AJAX*) untuk memuat riwayat buku besar ($client->ledgers()) atau riwayat faktur secara terpisah hanya ketika *tab* tersebut diklik oleh pengguna, lengkap dengan parameter halamannya sendiri seperti 'ledger_page' atau 'inv_page'.

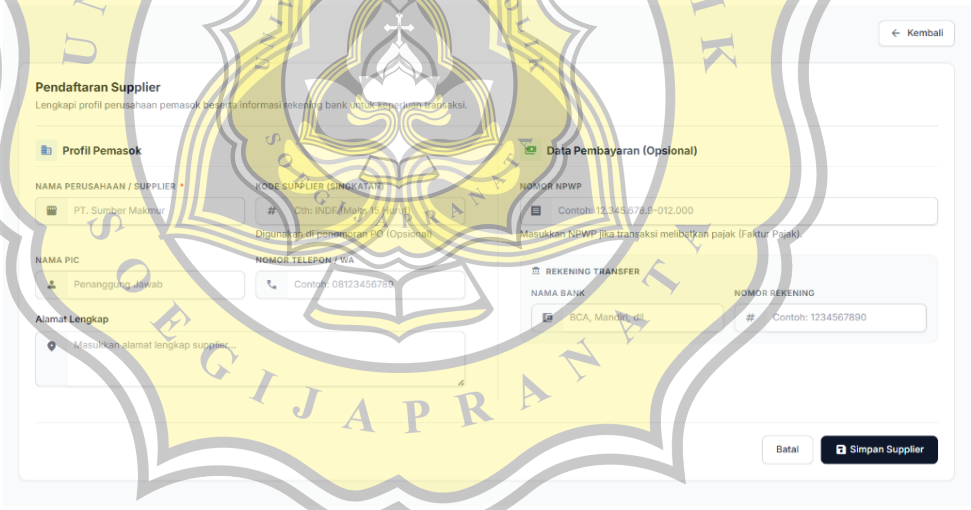
4.3.11 Modul Manajemen Pemasok

Modul Pemasok digunakan untuk mencatat dan mengelola data vendor. Tampilan utama modul ini adalah tabel daftar pemasok aktif yang memuat identitas perusahaan, NPWP, dan jumlah produk yang disuplai, seperti pada Gambar 4.70.



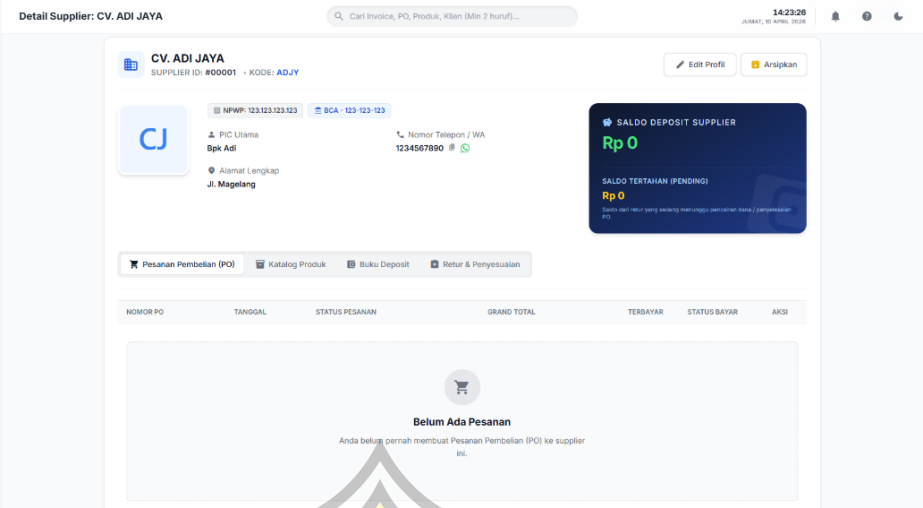
Gambar 4. 70 Halaman Daftar Supplier

Pendaftaran vendor baru dilakukan melalui formulir pada Gambar 4.71. Pengguna perlu mengisi nama perusahaan, penanggung jawab, dan rekening bank. Pada kolom "Kode *Supplier*", pengguna dapat mengisinya secara manual atau mengosongkannya agar sistem membuat kode secara otomatis..



Gambar 4. 71 Halaman Tambah Supplier

Setelah profil pemasok tersimpan, pengguna dapat melihat rekam jejak lengkap vendor tersebut melalui halaman detail pada Gambar 4.72. Halaman ini bertindak sebagai buku besar mini yang mengelompokkan riwayat Pesanan Pembelian *(PO)*, katalog produk, hingga retur ke dalam *tab-tab* yang terpisah.



Gambar 4. 72 Halaman Detail Supplier

Alur penyimpanan dan validasi modul ini diatur di dalam berkas *Supplier*Controller, yang potongan kodenya dapat dilihat pada Gambar 4.73.

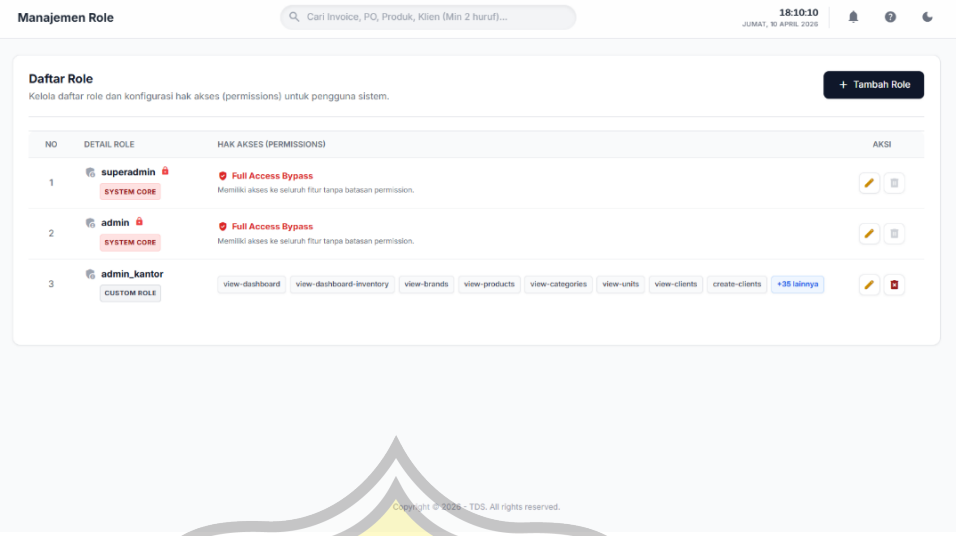
```php
    public function store(Request $request): RedirectResponse
    {
        Gate::authorize('create', Supplier::class);
        if ($request->filled('supplier_code')) {
            $request->merge(['supplier_code' => strtoupper($request->supplier_code)]);
        } else {
            $request->merge(['supplier_code' => $this->generateSupplierCode($request->supplier_name)]);
        }
        $request->validate([
            'supplier_name' => 'required|string|max:150|unique:suppliers,supplier_name',
            'supplier_code' => 'required|string|max:15|unique:suppliers,supplier_code',
            'person_in_charge' => 'nullable|string|max:100',
            'phone_number' => 'nullable|string|max:20',
            'address' => 'nullable|string',
            'npwp' => ['nullable', 'string', 'max:30', Rule::unique('suppliers')],
            'bank_name' => 'nullable|string|max:50',
            'account_number' => 'nullable|string|max:50',
        ]);
        Supplier::create($request->all());
        return redirect()->route('admin.suppliers.index')->with('success', 'Supplier baru berhasil ditambahkan.');
    }
    public function show(Supplier $supplier): View
    {...
    }
    public function getTabContent(Request $request, Supplier $supplier)
    {
        Gate::authorize('view', $supplier);
        $tab = $request->get('tab', 'purchase_orders');
        switch ($tab) {
            case 'purchase_orders':
                $purchaseOrders = $supplier->purchaseOrders()->latest('order_date')->paginate(10, ['*'], 'po_page');
                return view('admin.suppliers.tabs.purchase_orders', compact('purchaseOrders'))->render();
            case 'products':
                $products = $supplier->products()->with(['category', 'unit'])->latest()->paginate(10, ['*'], 'products_page');
                return view('admin.suppliers.tabs.products', compact('products'))->render();
            case 'ledger':...
            case 'returns':
                $returns = \App\Models\PurchaseReturn::where('supplier_id', $supplier->supplier_id)
                            ->with('purchaseOrder')
                            ->latest('return_date')
                            ->paginate(10, ['*'], 'returns_page');
                $adjustments = \App\Models\PurchaseOrderAdjustment::whereHas('purchaseOrder', function($q) use ($supplier) {
                                $q->where('supplier_id', $supplier->supplier_id);
                            })
                            ->with('purchaseOrder')
                            ->latest('adjustment_date')
                            ->paginate(10, ['*'], 'adj_page');
                return view('admin.suppliers.tabs.returns', compact('returns', 'adjustments'))->render();
            default:
                return response()->json(['error' => 'Tab not found'], 404);
        }
    }
```

Gambar 4. 73 Kode Program Modul Supplier

Pada *public function store*, sistem mengevaluasi input kode melalui kondisi if ($request->filled('*supplier*_code')). Jika kosong, sistem memanggil private function generate*Supplier*Code($name). Fungsi ini mengambil maksimal tiga huruf pertama dari setiap kata pada nama vendor yang ditampung dalam variabel $initials. Inisial tersebut digabungkan dengan angka acak dari fungsi rand(100, 999) menjadi variabel $baseCode. Sistem kemudian melakukan perulangan while untuk mengecek ketersediaan kode di basis data. Jika kode tersebut sudah ada, variabel $finalCode akan terus dimodifikasi dengan penambahan nomor urut ($counter) hingga sistem mendapatkan kode yang unik. Pada proses pengarsipan di *public function destroy*, sistem memvalidasi relasi data menggunakan parameter $*supplier*->*Purchase*Orders()->exists(). Jika pemasok tersebut tercatat memiliki riwayat Pesanan Pembelian, sistem akan menolak proses hapus untuk menjaga integritas data transaksi.

4.3.12 Modul Manajemen Peran dan Hak Akses

Modul Manajemen Role berfungsi sebagai pusat kendali otorisasi *(Access Control List)* untuk mengatur kelompok pengguna dan hak aksesnya. Antarmuka daftar role pada Gambar 4.74 menampilkan level akses yang tersedia, di mana peran inti seperti superadmin dan admin ditandai khusus sebagai *"System Core"* yang memiliki akses tak terbatas *(bypass)*.



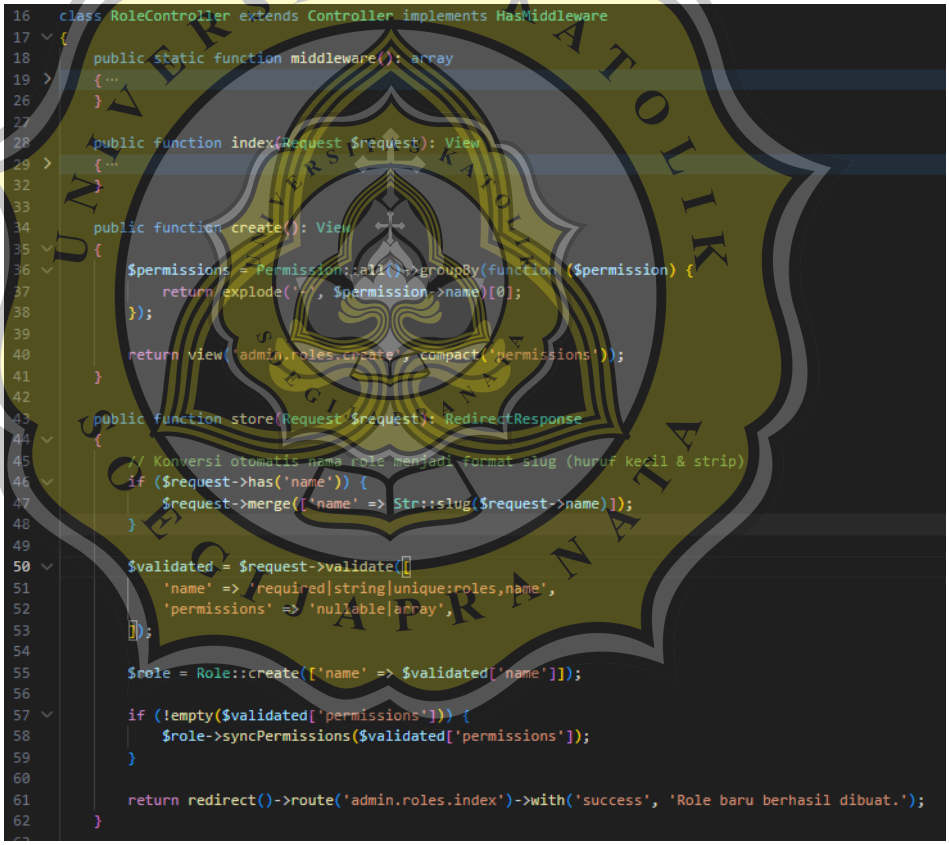
Gambar 4. 74 Halaman Daftar Role dan Hak Akses

Jika administrator ingin membuat kelompok staf baru (misal: "admin Kantor"), mereka dapat menggunakan formulir pada Gambar 4.74. Formulir ini menampilkan puluhan opsi kotak centang *(checkbox) permissions* yang dikelompokkan secara rapi berdasarkan modul (seperti *View*, *Create*, *Edit*, *Delete*) untuk memudahkan konfigurasi batasan tugas setiap karyawan.

Gambar 4. 75 Halaman Tambah Role dan Hak Akses

Pemrosesan data otorisasi ini dieksekusi oleh Modul RoleController, yang potongan logikanya dapat dilihat pada Gambar 4.76. Pada fungsi store dan update, sistem tidak langsung menyimpan nama peran yang diketik pengguna. Sistem mencegat input tersebut dan menjalankannya melalui

fungsi Str::slug($request->name), yang secara otomatis mengubah spasi menjadi tanda strip dan memaksa huruf menjadi kecil (contoh: "admin Gudang" menjadi "admin-gudang") demi konsistensi format database. Selanjutnya, modul ini menggunakan fungsi spesifik dari library Spatie, yaitu $role->syncPermissions($validated['permissions']), untuk membuang hak akses lama dan menggantinya dengan rentetan hak akses baru secara sinkron dalam satu perintah. Untuk mencegah kelumpuhan sistem, fungsi destroy dibekali validasi khusus if ($role->name === 'admin') yang akan langsung memblokir upaya penghapusan role bawaan sistem, serta memblokir penghapusan jika role tersebut masih dipakai oleh staf aktif ($role->users()->count() > 0).



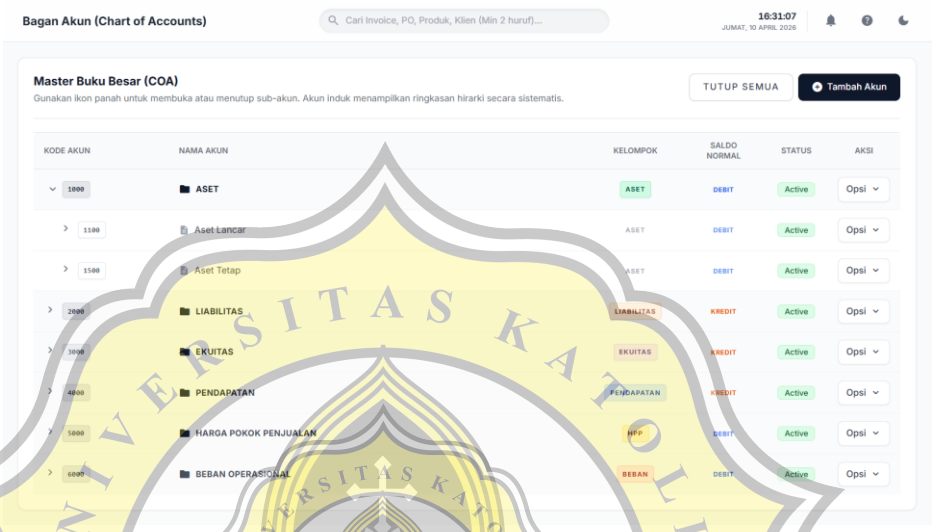
```
class RoleController extends Controller implements HasMiddleware
{
    public static function middleware(): array
    {…
    }

    public function index(Request $request): View
    {…
    }

    public function create(): View
    {
        $permissions = Permission::all()->groupBy(function ($permission) {
            return explode('.', $permission->name)[0];
        });

        return view('admin.roles.create', compact('permissions'));
    }

    public function store(Request $request): RedirectResponse
    {
        // Konversi otomatis nama role menjadi format slug (huruf kecil & strip)
        if ($request->has('name')) {
            $request->merge(['name' => Str::slug($request->name)]);
        }

        $validated = $request->validate([
            'name' => 'required|string|unique:roles,name',
            'permissions' => 'nullable|array',
        ]);

        $role = Role::create(['name' => $validated['name']]);

        if (!empty($validated['permissions'])) {
            $role->syncPermissions($validated['permissions']);
        }

        return redirect()->route('admin.roles.index')->with('success', 'Role baru berhasil dibuat.');
    }
```

Gambar 4. 76 Kode Porgram Modul Peran dan Hak Akses

4.3.13 Modul Manajemen *Chart of Accounts (COA)*

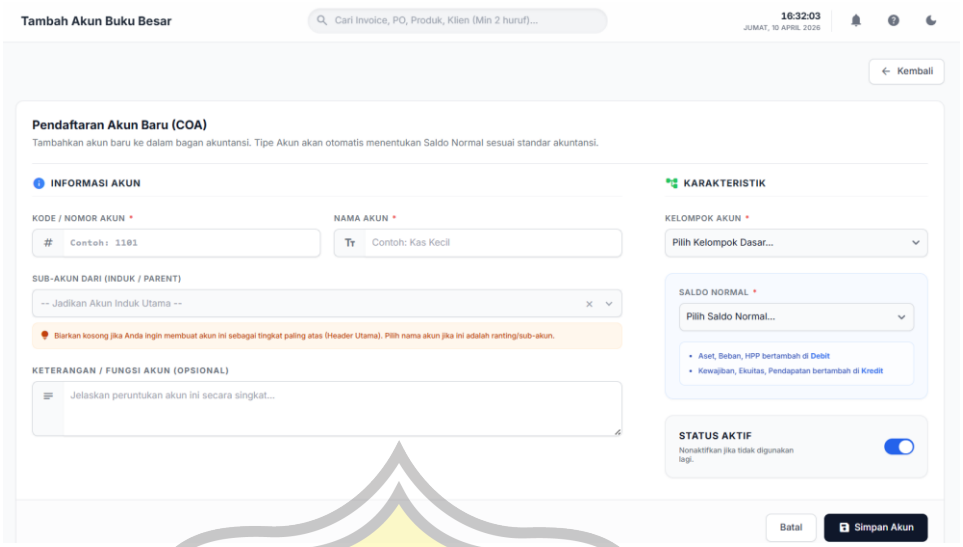
Modul Bagan Akun atau Chart of Accounts (COA) adalah fondasi utama dari sistem akuntansi. Modul ini difungsikan untuk mendaftar dan menyusun kerangka kode akun buku besar yang nantinya akan digunakan

untuk menjurnal seluruh transaksi perusahaan. Tampilan utama modul ini dapat dilihat pada Gambar 4.77. Antarmuka ini menampilkan struktur daftar akun yang tidak sekadar linier, melainkan hierarkis (*nested*). Pengguna dapat mengeklik ikon panah untuk membuka atau menutup sub-akun yang bersarang di bawah akun induk utamanya.



Gambar 4. 77 Halaman Daftar COA

Saat perusahaan membutuhkan klasifikasi pos keuangan yang baru, administrator dapat mendaftarkannya melalui formulir pada Gambar 4.78. Pada antarmuka pendaftaran ini, pengguna diwajibkan untuk menentukan kode akun, memilih kelompok dasar (seperti Aset, Beban, atau Ekuitas), menetapkan aturan saldo normal (Debit/Kredit), dan dapat memilih untuk mengaitkan akun tersebut ke dalam sebuah akun induk (parent account).



Gambar 4. 78 Form Tambah COA

Tingkat kerumitan pemrosesan data untuk menciptakan struktur hierarki antarmuka tersebut dikelola oleh logika di dalam Modul ChartOfAccountController, yang potongan kodenya dapat dilihat pada Gambar 4.79

```
33      public function index(): View
34      {
35          // 1. Ambil semua akun diurutkan berdasarkan nomor akun
36          $allAccounts = ChartOfAccount::orderBy('account_number', 'asc')->get();
37
38          $flattenedAccounts = collect();
39
40          /**...
46          $buildTree = function ($parentId, $depth, $path = []) use (&$buildTree, $allAccounts, &$flattenedAccounts) {
47              $children = $allAccounts->where('parent_account_id', $parentId);
48              foreach ($children as $child) {
49                  $child->depth = $depth;
50                  $child->ancestors = $path;
51                  $child->has_children = $allAccounts->where('parent_account_id', $child->account_id)->isNotEmpty();
52                  $flattenedAccounts->push($child);
53                  $currentPath = array_merge($path, [$child->account_id]);
54                  $buildTree($child->account_id, $depth + 1, $currentPath);
55              }
56          };
57          $buildTree(null, 0);
58          return view('admin.chart_of_accounts.index', ['accounts' => $flattenedAccounts]);
59      }
60      public function create(): View
61      {
62          $parentAccounts = ChartOfAccount::orderBy('account_number', 'asc')->get();
63          $accountTypes = ['Aset', 'Liabilitas', 'Ekuitas', 'Pendapatan', 'HPP', 'Beban'];
64          $normalBalances = ['Debit', 'Kredit'];
65
66          return view('admin.chart_of_accounts.create', compact('parentAccounts', 'accountTypes', 'normalBalances'));
67      }
68      public function store(Request $request): RedirectResponse
69      {
70          $validated = $request->validate([
71              'account_number' => 'required|string|max:20|unique:chart_of_accounts,account_number',
72              'account_name' => 'required|string|max:255',
73              'account_type' => ['required', Rule::in(['Aset', 'Liabilitas', 'Ekuitas', 'Pendapatan', 'HPP', 'Beban'])],
74              'normal_balance' => ['required', Rule::in(['Debit', 'Kredit'])],
75              'parent_account_id' => 'nullable|exists:chart_of_accounts,account_id',
76              'is_active' => 'boolean',
77              'description' => 'nullable|string',
78          ]);
79
80          $validated['is_active'] = $request->has('is_active');
81          $validated['parent_account_id'] = $validated['parent_account_id'] ?? null;
82
83          ChartOfAccount::create($validated);
84
85          return redirect()->route('admin.chart-of-accounts.index')->with('success', 'Akun baru berhasil ditambahkan.');
86      }
```

Gambar 4. 79 Kode Program Modul COA

Keunggulan algoritma pada modul ini terletak di dalam fungsi index. Karena sebuah sub-akun bisa saja memiliki sub-akun lagi di bawahnya tanpa batas kedalaman, sistem tidak dapat menggunakan metode pemanggilan basis data biasa. Oleh karena itu, modul ini menggunakan fungsi rekursif (*recursive function*) bernama $buildTree.

Fungsi ini bekerja dengan cara memanggil dirinya sendiri berulang kali untuk menggali ke bawah ($depth + 1) setiap kali ia menemukan variabel $child yang memiliki kaitan $parentId. Bersamaan dengan itu, kueri $allAccounts->where('parent_account_id', $child->account_id)->isNotEmpty() digunakan untuk menandai apakah sebuah baris akun memiliki "anak" atau tidak, yang nantinya digunakan oleh antarmuka untuk menampilkan atau menyembunyikan ikon panah pembuka (*dropdown*).

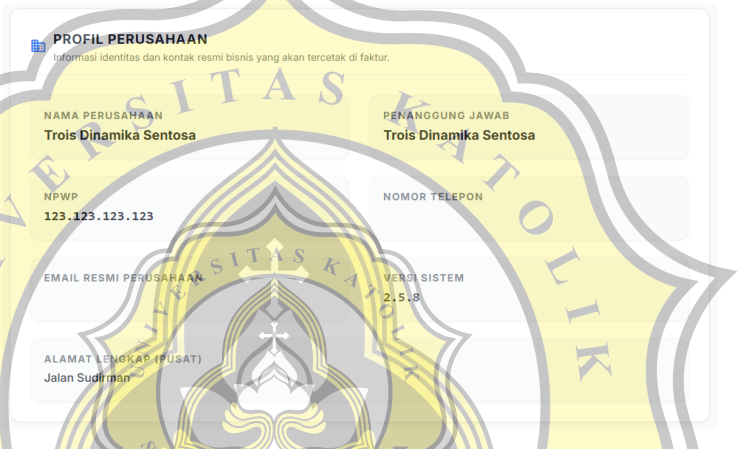
Selain kompleksitas pembacaan data, modul ini juga dilengkapi dengan mekanisme perlindungan penghapusan data *(destroy)* tiga lapis yang sangat ketat. Saat pengguna mencoba menghapus sebuah akun buku besar, sistem akan menahannya dan menjalankan serangkaian validasi. Pertama, sistem mengecek kueri $chartOfAccount->children()->exists(); jika akun tersebut memiliki sub-akun di bawahnya, penghapusan ditolak. Kedua, sistem melakukan kueri ke dalam tabel general_ledgers; jika akun tersebut pernah dipakai mencatat nominal transaksi walau hanya sekali, penghapusan ditolak demi mencegah ketidakseimbangan keuangan.

Terakhir, sistem memvalidasi tabel pengaturan (App\Models\Setting); jika akun tersebut sedang dikunci sebagai akun *default* sistem, penghapusan juga akan ditolak. Jika gagal melewati salah satu dari tiga gerbang validasi ini, pengguna hanya disarankan untuk menonaktifkan akun tersebut melalui sakelar status aktif.

4.3.14 Modul Pengaturan Perusahaan

Keandalan sebuah sistem ERP sangat bergantung pada konfigurasi awal atau pengaturan dasar *(core settings)*. Untuk mewadahi kebutuhan tersebut, dikembangkanlah Modul Pengaturan Sistem yang berfungsi sebagai pusat kendali profil dan integrasi buku besar. Tampilan modul ini

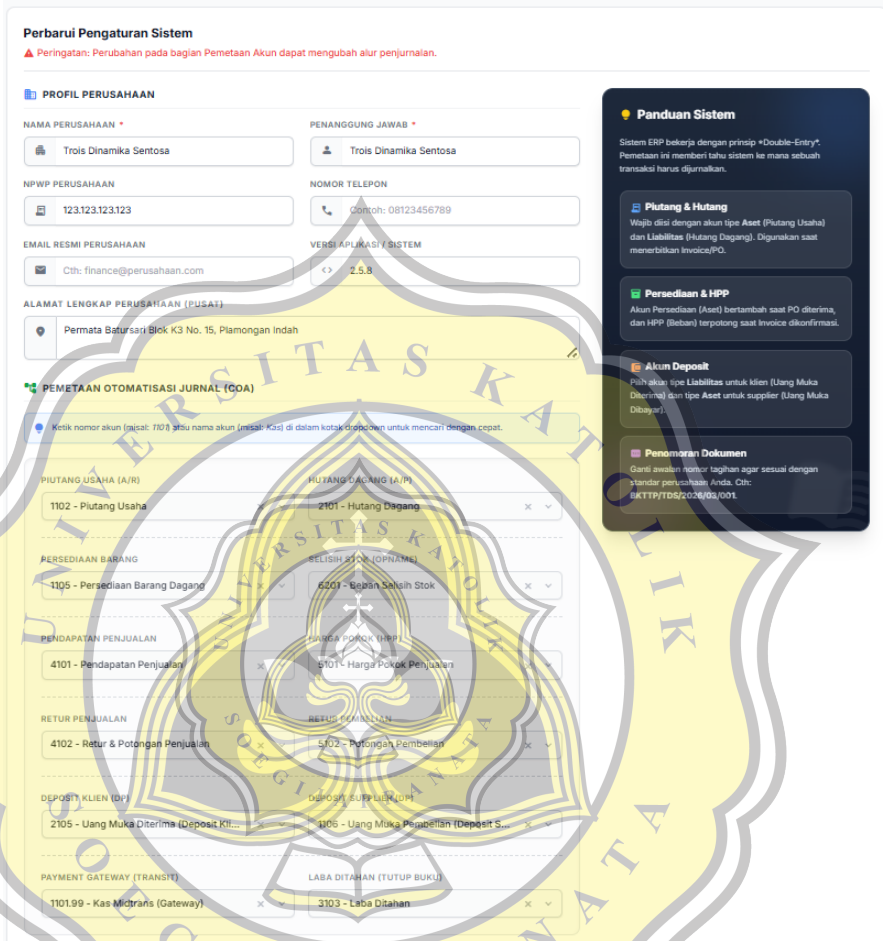
dapat dilihat pada Gambar 4.80, Gambar 4.81, dan Gambar 4.83. Pada antarmuka awal yang bersifat hanya-baca (*read-only*), pengguna dengan hak akses administratif dapat meninjau informasi mendasar. Informasi ini meliputi data profil utama perusahaan (seperti nama entitas, NPWP, alamat), pengaturan identitas awalan (*prefix*) untuk penomoran dokumen, hingga panel Pemetaan Jurnal (*Chart of Accounts Mapping*). Panel pemetaan ini sangat krusial, karena berfungsi mengarahkan pergerakan uang dari setiap transaksi harian (seperti tagihan, retur, atau kasbon) secara otomatis menuju kode akun akuntansi yang tepat.



Gambar 4. 80 Halaman Data Profil Perusahaan

Gambar 4. 81 Pemetaan Sistem Charts of Accounts/COA

Apabila sewaktu-waktu perusahaan perlu menyesuaikan identitas cabang atau melakukan perombakan bagan akun, sistem menyediakan formulir khusus untuk mengubah nilai konfigurasi tersebut. Formulir pengeditan ini diperlihatkan pada Gambar 4.82



Gambar 4. 82 Halaman Edit Pengaturan Profil dan COA

Sistem pembacaan dan penyimpanan pengaturan ini diorkestrasi oleh sebuah kelas pengendali, yaitu SettingController. Potongan kode inisialisasinya dapat dilihat pada Gambar 4.83.

```
public function index(): View
{
    $settings = Setting::getAllSettings();
    $allAccounts = ChartOfAccount::where('is_active', true)
                                 ->orderBy('account_number', 'asc')
                                 ->get();
    return view('admin.settings.index', compact('settings', 'allAccounts'));
}

public function update(Request $request): RedirectResponse
{
    $request->validate([
        'acct_default_ar' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_ap' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_inventory' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_sales_revenue' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_cogs' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_sales_return' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_purchase_return' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_supplier_deposit' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_client_deposit' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_gateway' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_inventory_adjustment' => 'nullable|exists:chart_of_accounts,account_id',
        'acct_default_retained_earnings' => 'nullable|exists:chart_of_accounts,account_id',
        'company_name' => 'required|string|max:255',
        'company_owner' => 'required|string|max:255',
        'company_address' => 'nullable|string',
        'company_city_province' => 'nullable|string|max:255',
        'company_phone' => ['nullable', 'string', 'max:20', 'regex:/^[0-9\+\-\s]+$/'],
        'company_npwp' => ['nullable', 'string', 'max:30', 'regex:/^[0-9\.\-\/]+$/'],
        'company_email' => 'nullable|email|max:255',
        'system_version' => 'nullable|string|max:20',
        'prefix_invoice' => 'nullable|string|max:15',
        'prefix_so' => 'nullable|string|max:15',
        'prefix_po' => 'nullable|string|max:15',
        'hq_code_alias' => 'nullable|string|max:15',
    ], [
        'company_phone.regex' => 'Format nomor telepon tidak valid.',
        'company_npwp.regex' => 'Format NPWP tidak valid (hanya angka, titik, strip).',
    ]);
    foreach ($request->except('_token', '_method') as $key => $value) {
        Setting::updateOrCreate(
            ['key' => $key],
            ['value' => $value ?? '']
        );
    }
    return redirect()->route('admin.settings.index')->with('success', 'Pengaturan berhasil diperbarui.');
}
```

Gambar 4. 83 Kode Program Modul Supplier

Di dalam *controller*, logika utama dititikberatkan pada fungsi pembaruan data, yaitu public function update(Request $request). Sebelum menyentuh basis data, sistem akan menyaring (*filter*) seluruh isian pengguna secara ketat. Sebagai contoh, untuk validasi pemetaan jurnal seperti pada variabel acct_default_inventory, sistem menggunakan aturan exists:chart_of_accounts,account_id. Hal ini berguna untuk mencegah terjadinya penyimpanan ID akun fiktif yang dapat merusak tatanan laporan keuangan di kemudian hari.

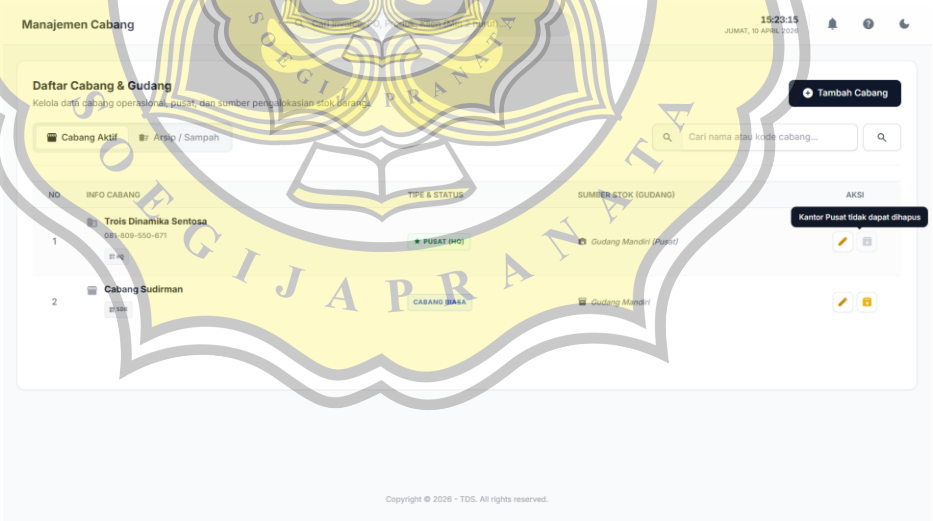
Selain itu, untuk menjaga keabsahan (*validity*) format teks profil perusahaan, sistem mengandalkan regular expression (Regex). Pada variabel 'company_phone', sistem menggunakan instruksi 'regex:/^[0-9\+\-

\s]+$/' agar pengguna hanya bisa memasukkan angka, tanda tambah (+), dan tanda hubung (-).

Konsep validasi serupa juga diterapkan pada pengisian formulir NPWP. Setelah semua masukan dinyatakan absah, fungsi ini menggunakan metode perulangan foreach untuk memecah seluruh isian formulir. Sistem kemudian mengeksekusi perintah Setting::updateOrCreate untuk memperbarui nilai-nilai (*values*) lama atau menciptakan kunci (*keys*) baru secara dinamis di dalam basis data.

4.3.15 Modul Manajemen Cabang

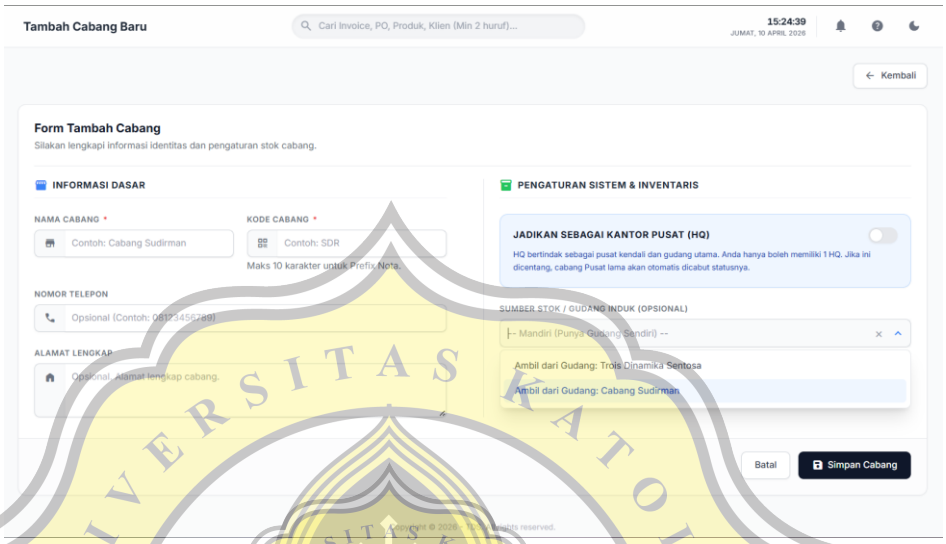
Modul Manajemen Cabang disediakan untuk mengakomodasi kebutuhan tata kelola operasional multi-gudang (*multi-warehouse*). Melalui modul ini, perusahaan dapat mendaftarkan berbagai lokasi cabang dan menentukan rantai pasok dari mana cabang tersebut akan mengambil persediaan barangnya. Pada Gambar 4.84 di bawah, diperlihatkan halaman utama daftar cabang yang menampilkan rincian tipe lokasi dan sumber stok masing-masing cabang yang sedang aktif beroperasi.



Gambar 4. 84 Halaman Daftar Cabang

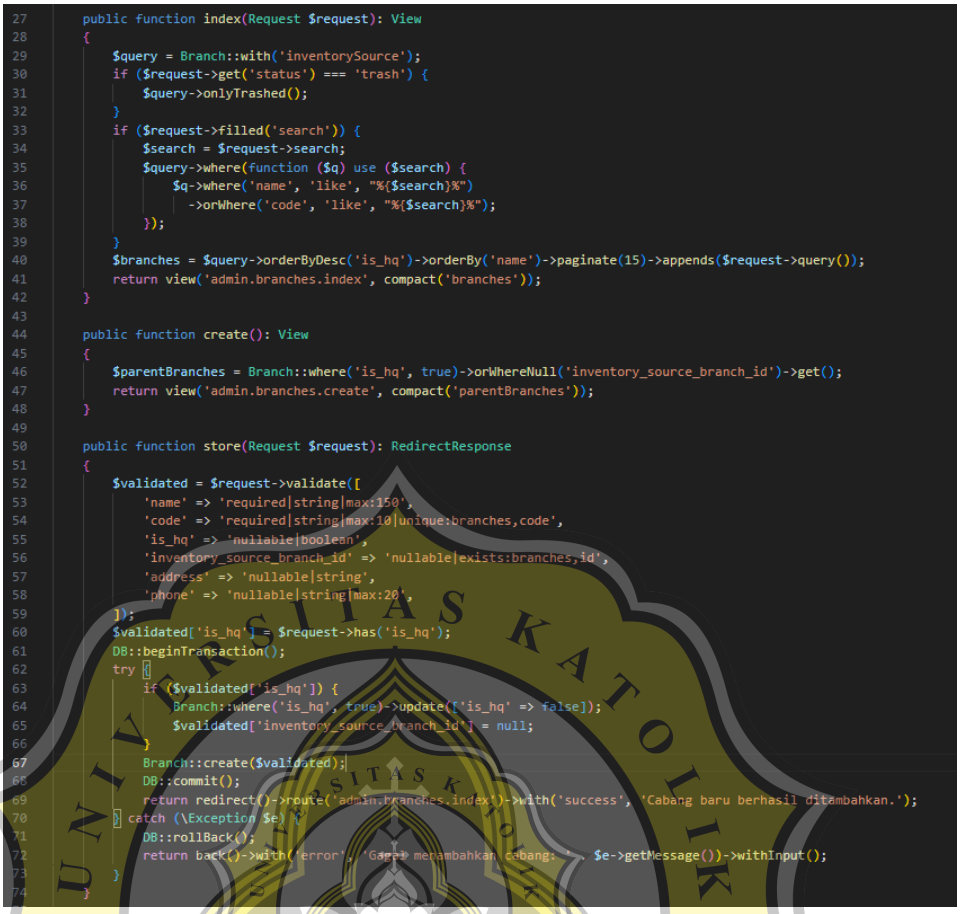
Jika perusahaan memutuskan untuk membuka area operasi baru, administrator dapat mendaftarkan lokasi tersebut melalui formulir penambahan cabang seperti pada Gambar 4.85. Pada formulir ini, admin dapat menetapkan lokasi baru tersebut sebagai cabang biasa, atau

mengaktifkan sakelar (*toggle*) khusus untuk mendeklarasikannya sebagai Kantor Pusat *(HQ)* yang baru. Selain itu, terdapat kolom pilihan "Sumber Stok / Gudang Induk" yang berfungsi untuk mengarahkan dari mana gudang cabang tersebut akan menyuplai barangnya.



Gambar 4. 85 Form Tambah Cabang

Pemrosesan data relasi antar cabang ini dikendalikan oleh berkas BranchController. Potongan kode penginisialisasian untuk berkas ini ditunjukkan pada Gambar 4.86.



```php
public function index(Request $request): View
{
    $query = Branch::with('inventorySource');
    if ($request->get('status') === 'trash') {
        $query->onlyTrashed();
    }
    if ($request->filled('search')) {
        $search = $request->search;
        $query->where(function ($q) use ($search) {
            $q->where('name', 'like', "%{$search}%")
              ->orWhere('code', 'like', "%{$search}%");
        });
    }
    $branches = $query->orderByDesc('is_hq')->orderBy('name')->paginate(15)->appends($request->query());
    return view('admin.branches.index', compact('branches'));
}

public function create(): View
{
    $parentBranches = Branch::where('is_hq', true)->orWhereNull('inventory_source_branch_id')->get();
    return view('admin.branches.create', compact('parentBranches'));
}

public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'name' => 'required|string|max:150',
        'code' => 'required|string|max:10|unique:branches,code',
        'is_hq' => 'nullable|boolean',
        'inventory_source_branch_id' => 'nullable|exists:branches,id',
        'address' => 'nullable|string',
        'phone' => 'nullable|string|max:20',
    ]);
    $validated['is_hq'] = $request->has('is_hq');
    DB::beginTransaction();
    try {
        if ($validated['is_hq']) {
            Branch::where('is_hq', true)->update(['is_hq' => false]);
            $validated['inventory_source_branch_id'] = null;
        }
        Branch::create($validated);
        DB::commit();
        return redirect()->route('admin.branches.index')->with('success', 'Cabang baru berhasil ditambahkan.');
    } catch (\Exception $e) {
        DB::rollBack();
        return back()->with('error', 'Gagal menambahkan cabang: ' . $e->getMessage())->withInput();
    }
}
```

Gambar 4. 86 Kode Program Modul Branch

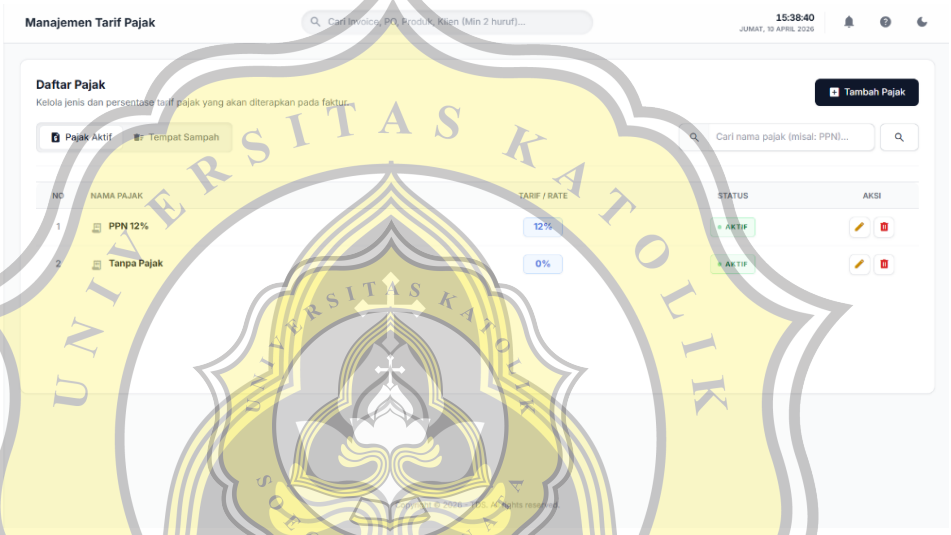
Pada proses pencatatan data di dalam public function store, sistem menerapkan sebuah logika tunggal (*Single Source of Truth*) terhadap status kantor pusat. Ketika pengguna mengirimkan data cabang baru dengan status HQ yang aktif ($validated['is_hq'] bernilai benar), sistem akan memicu perintah Branch::where('is_hq', true)->update(['is_hq' => false]). Instruksi basis data tersebut akan mencabut status HQ dari cabang yang lama terlebih dahulu, sebelum menetapkan cabang yang baru sebagai satu-satunya pusat kendali utama.

Modul ini juga dilengkapi dengan mekanisme perlindungan data bertingkat pada proses penghapusan di dalam *public function destroy*. Untuk mencegah rusaknya operasional sistem, fungsi ini akan langsung menolak perintah hapus apabila cabang yang dipilih merupakan Kantor Pusat ($branch->is_hq). Selain itu, proses hapus juga akan diblokir jika cabang tersebut masih digunakan sebagai gudang induk oleh cabang lain, atau jika

masih terdapat staf (*user*) yang bertugas dan terdaftar secara aktif pada lokasi cabang tersebut.

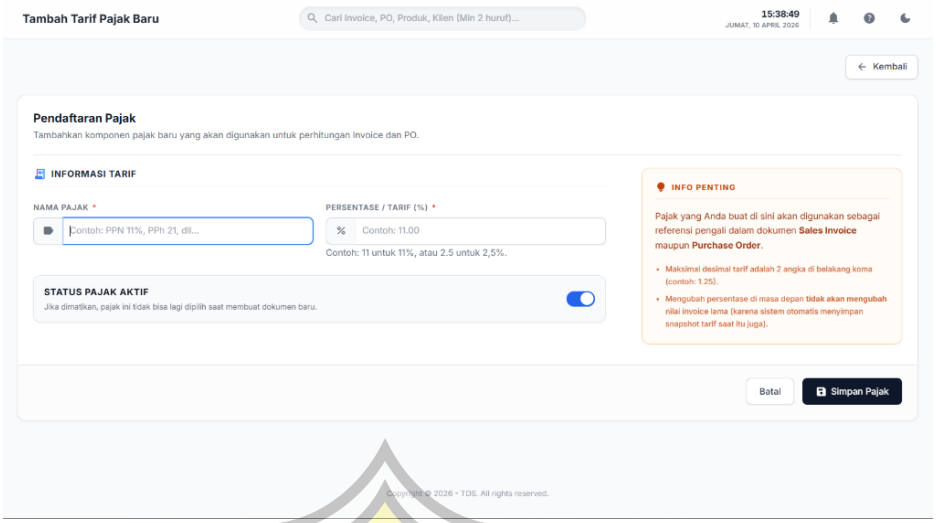
4.3.16 Modul Manajemen Pajak

Modul Manajemen Tarif Pajak dirancang untuk memfasilitasi pendataan persentase komponen pajak yang akan dibebankan pada dokumen transaksi, seperti PPN (Pajak Pertambahan Nilai) maupun PPh. Tampilan utama antarmuka ini berupa daftar pajak aktif dan tarif persentasenya, sebagaimana ditunjukkan pada Gambar 4.87.



Gambar 4. 87 Halaman Daftar Pajak

Untuk menambahkan komponen pajak baru, staf dapat mengakses formulir pendaftaran pada Gambar 4.88. Pada halaman ini, pengguna diminta memasukkan nama referensi pajak dan persentase tarifnya. Antarmuka ini juga memuat informasi bahwa pengubahan nilai persentase di masa mendatang tidak akan merusak kalkulasi pada dokumen faktur lama, karena sistem menggunakan mekanisme penyimpanan rekam jejak nilai (*snapshot*).



Gambar 4. 88 Form Tambah Pajak

Proses pengolahan input dari antarmuka tersebut ditangani oleh berkas TaxController. Potongan kode inisialisasinya dapat dilihat pada Gambar 4.89.

```php
public function index(Request $request): View
{
    $query = Tax::query();
    if ($request->get('status') === 'trash') {
        $query->onlyTrashed();
    }
    if ($request->filled('search')) {
        $query->where('name', 'like', '%' . $request->search . '%');
    }
    $taxes = $query->latest()->paginate(10)->appends($request->query());
    return view('admin.taxes.index', compact('taxes'));
}

public function create(): View
{
    return view('admin.taxes.create');
}

public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'name' => 'required|string|max:255|unique:taxes,name',
        'rate' => 'required|numeric|min:0|max:100',
        'is_active' => 'nullable|boolean',
    ]);
    $validated['is_active'] = $request->has('is_active');
    Tax::create($validated);
    return redirect()->route('admin.taxes.index')->with('success', 'Tarif pajak baru berhasil dibuat.');
}
```

Gambar 4. 89 Kode Porgram Modul Tax/ Pajak

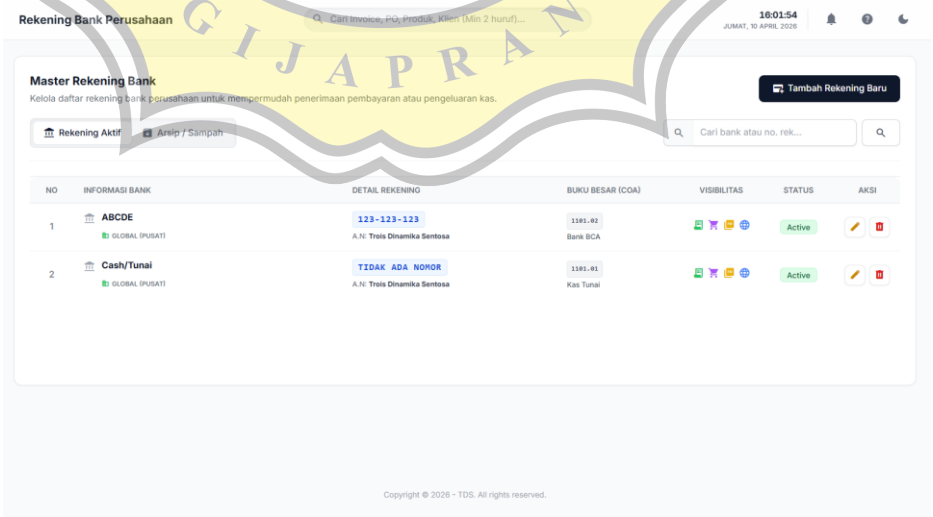
Pada tahap penyimpanan data di fungsi *store* maupun *update*, sistem memvalidasi variabel rate secara ketat agar nilai persentase yang dimasukkan selalu berada pada rentang angka 0 hingga 100. Bersamaan dengan itu, status sakelar pajak ditangkap menggunakan instruksi $request-

>has('is_active') untuk dikonversi menjadi tipe data *boolean* sebelum disimpan.

Terkait penghapusan data, *controller* ini menerapkan dua lapis penanganan. Pada fungsi destroy, sistem mengizinkan proses pengarsipan sementara (*soft delete*) tanpa memberikan blokir peringatan, karena nilai pajak pada faktur sebelumnya sudah terkunci secara independen. Akan tetapi, apabila pengguna menginstruksikan penghapusan permanen melalui fungsi forceDelete, sistem akan mengeksekusi kueri $tax->sales*Invoice*s()->count() > 0. Kueri ini bertugas mengecek relasi basis data; jika pajak tersebut tercatat pernah digunakan dalam satu saja dokumen faktur historis, maka perintah hapus permanen akan langsung ditolak demi menjaga keutuhan data referensi akuntansi.

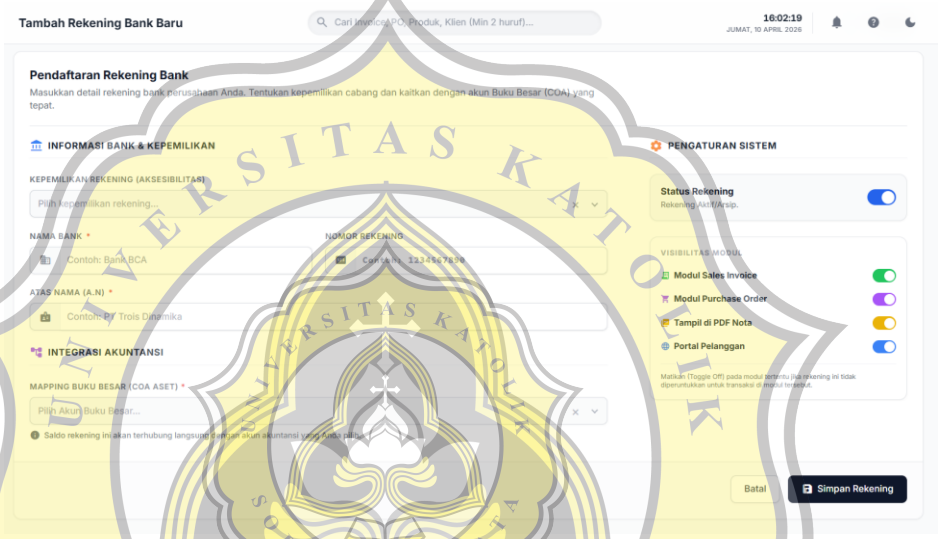
4.3.17 Modul Manajemen Rekening Bank Perusahaan

Modul Rekening Bank Perusahaan dikembangkan untuk mendata saluran penerimaan dan pengeluaran kas yang digunakan oleh entitas bisnis. Antarmuka utama modul ini adalah daftar akun rekening seperti yang disajikan pada Gambar 4.90. Pada halaman ini, staf dapat melihat ringkasan detail rekening, status keaktifan, hingga meninjau indikator visual (ikon) yang menandakan di mana saja rekening tersebut akan ditampilkan dalam sistem.



Gambar 4. 90 Halaman Daftar Rekening Bank Perusahaan

Saat perusahaan membuka saluran kas baru, administrator dapat mendaftarkannya melalui formulir pendaftaran pada Gambar 4.91 dan Gambar 4.92. Berbeda dengan sekadar mencatat nomor rekening, formulir ini dilengkapi dengan integrasi akuntansi dan visibilitas berlapis. Pengguna tidak hanya menentukan identitas bank, tetapi juga harus memetakan saldo rekening tersebut ke sebuah akun Buku Besar (COA), serta memilih pada aplikasi mana saja rekening tersebut boleh muncul (misalnya di modul faktur penjualan, pembelian, PDF nota, atau portal klien).



Gambar 4. 91 Halaman Form Tambah Rekening Bank Perusahaan

Gambar 4. 92 Memilih Kepemilikan Rekening Berdasarkan Cabang

Alur pemrosesan formulir dan penyaringan data untuk antarmuka tersebut dikelola oleh logika inti pada Modul CompanyBankAccountController, yang sebagian potongan kodenya dapat dilihat pada Gambar 4.93.

```php
public function index(Request $request): View
{
    $query = CompanyBankAccount::with(['account', 'branch']);
    $user = Auth::user();
    if (!$user->isGlobal()) {
        $query->where(function($q) use ($user) {
            $q->where('branch_id', $user->branch_id) // Hanya bisa lihat Rekening Cabangnya sendiri
              ->orWhereNull('branch_id');            // Biarkan lain melihat Rekening Pusat (HQ)
        });
    }
    if ($request->get('status') === 'trash') {
        $query->onlyTrashed();
    }
    if ($request->filled('search')) {
        $search = $request->search;
        $query->where(function($q) use ($search) {
            $q->where('bank_name', 'like', "%{$search}%")
              ->orWhere('account_name', 'like', "%{$search}%")
              ->orWhere('account_number', 'like', "%{$search}%");
        });
    }
    $accounts = $query->orderBy('bank_name')->paginate(15)->appends($request->query());
    return view('admin.company_bank_accounts.index', compact('accounts'));
}
public function create(): View
{
    $assetAccounts = ChartOfAccount::where('account_type', 'Aset')
                        ->where('is_active', true)
                        ->orderBy('account_number')
                        ->get();

    $branches = \App\Models\Branch::orderBy('name')->get();
    return view('admin.company_bank_accounts.create', compact('assetAccounts', 'branches'));
}

public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'branch_id' => 'nullable|exists:branches,id',
        'bank_name' => 'required|string|max:255',
        'account_name' => 'required|string|max:255',
        'account_number' => 'nullable|string|max:50',
        'is_active' => 'required|boolean',
        'show_in_invoice' => 'required|boolean',
        'show_in_purchase_order' => 'required|boolean',
        'show_in_client_portal' => 'required|boolean',
        'show_in_pdf' => 'required|boolean',
        'chart_of_account_id' => 'required|exists:chart_of_accounts,account_id',
    ]);
    CompanyBankAccount::create($validated);
    return redirect()->route('admin.company-bank-accounts.index')
                     ->with('success', 'Akun bank baru berhasil ditambahkan.');
}
```

Gambar 4. 93 Kode Program modul Company Bank Account

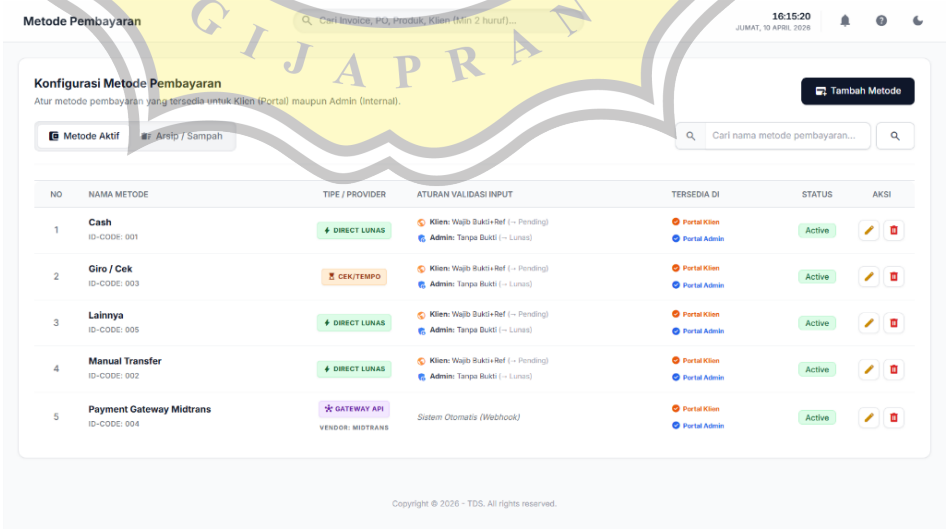
Salah satu fungsi terpenting dari modul ini terletak pada bagaimana sistem membatasi tampilan data di tabel daftar rekening melalui fungsi pembacaan data awal (index). Untuk mengakomodasi struktur multi-cabang perusahaan, modul ini mengevaluasi identitas staf yang sedang *login* menggunakan kueri bersyarat if (!$user->isGlobal()). Jika sistem mendeteksi bahwa staf tersebut bukan berasal dari kantor pusat, kueri pembacaan basis data akan otomatis dibatasi. Sistem hanya akan menampilkan rekening yang kolom branch_id-nya cocok dengan cabang staf tersebut, ditambah dengan rekening berstatus global atau pusat (di mana

kolom branch_id bernilai *null*). Mekanisme ini memastikan staf cabang A tidak bisa melihat atau menggunakan rekening milik cabang B.

Selanjutnya, modul ini menerapkan pengamanan berlapis saat administrator berniat untuk menghapus data rekening secara permanen. Pada fungsi eksekusi penghapusan permanen (forceDelete), sistem tidak akan langsung melenyapkan data. Sistem terlebih dahulu memvalidasi riwayat operasional rekening tersebut dengan menjalankan instruksi pengecekan $account->salesPayments()->exists() dan $account->*Purchase*Payments()->exists(). Logika ini memastikan bahwa apabila sebuah rekening bank pernah digunakan untuk memproses transaksi pembayaran piutang penjualan maupun pembayaran utang pembelian, permohonan hapus permanen akan otomatis diblokir. Hal ini sangat vital untuk mencegah terputusnya laporan rekam jejak kas perusahaan.

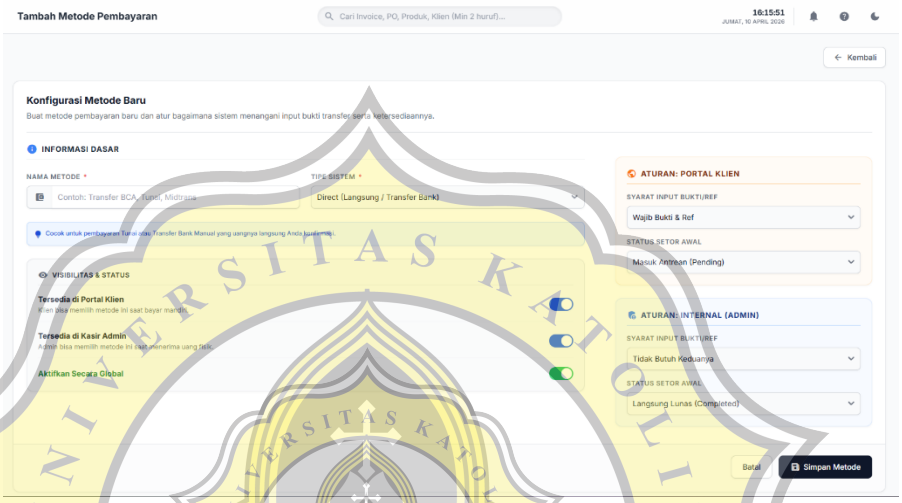
4.3.18 Modul Manajemen Metode Pembayaran

Modul Metode Pembayaran dikembangkan untuk mengatur daftar saluran pembayaran yang dapat digunakan oleh klien di portal eksternal maupun oleh kasir di portal internal. Tampilan awal modul ini menyajikan tabel daftar metode pembayaran yang merangkum nama metode, tipe penyedia (*provider*), aturan validasi unggah bukti transfer, serta status ketersediaannya, seperti yang ditunjukkan pada Gambar 4.94.



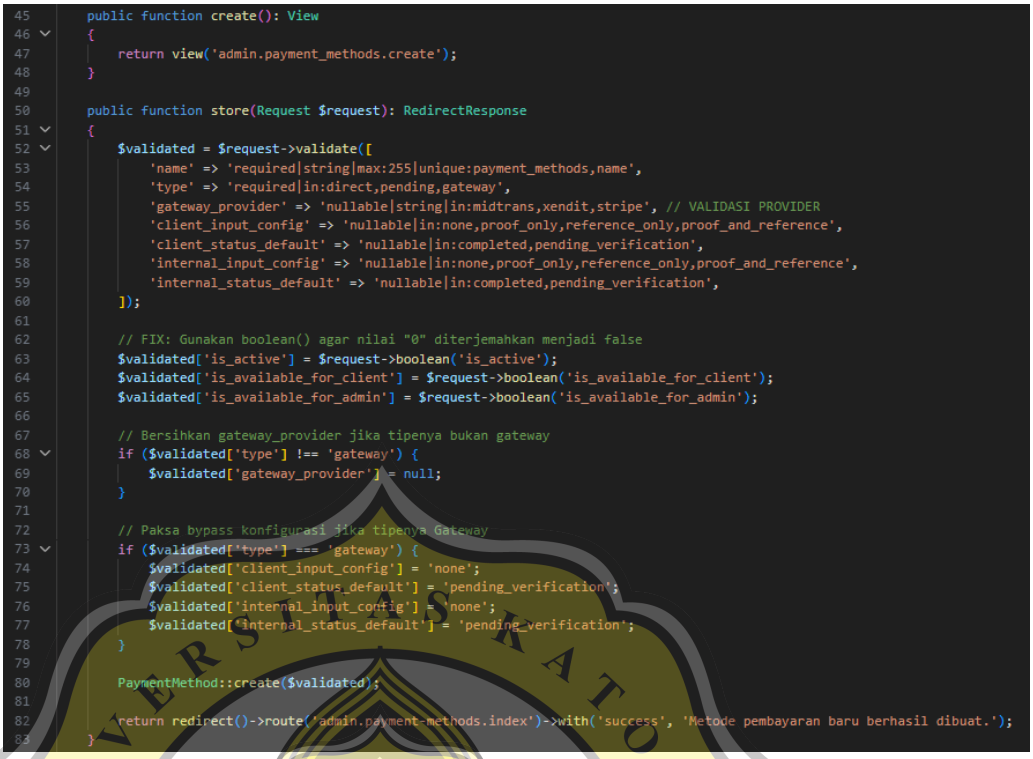
Gambar 4. 94 Halaman Daftar Metode Pembayaran

Untuk merancang alur pembayaran baru, administrator dapat menggunakan formulir konfigurasi pada Gambar 4.95. Melalui antarmuka ini, pengguna menentukan nama metode, tipe sistem (pembayaran langsung, tempo, atau *payment gateway*), serta menetapkan aturan spesifik seperti apakah klien diwajibkan untuk melampirkan bukti dan nomor referensi transfer agar pembayarannya dapat diproses.



Gambar 4. 95 Halaman Tambah Metode Pembayaran

Alur pemrosesan konfigurasi formulir tersebut dikelola oleh logika pada Modul PaymentMethodController, yang potongan kodenya dapat dilihat pada Gambar 4.96.



```php
public function create(): View
{
    return view('admin.payment_methods.create');
}

public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'name' => 'required|string|max:255|unique:payment_methods,name',
        'type' => 'required|in:direct,pending,gateway',
        'gateway_provider' => 'nullable|string|in:midtrans,xendit,stripe', // VALIDASI PROVIDER
        'client_input_config' => 'nullable|in:none,proof_only,reference_only,proof_and_reference',
        'client_status_default' => 'nullable|in:completed,pending_verification',
        'internal_input_config' => 'nullable|in:none,proof_only,reference_only,proof_and_reference',
        'internal_status_default' => 'nullable|in:completed,pending_verification',
    ]);

    // FIX: Gunakan boolean() agar nilai "0" diterjemahkan menjadi false
    $validated['is_active'] = $request->boolean('is_active');
    $validated['is_available_for_client'] = $request->boolean('is_available_for_client');
    $validated['is_available_for_admin'] = $request->boolean('is_available_for_admin');

    // Bersihkan gateway_provider jika tipenya bukan gateway
    if ($validated['type'] !== 'gateway') {
        $validated['gateway_provider'] = null;
    }

    // Paksa bypass konfigurasi jika tipenya Gateway
    if ($validated['type'] === 'gateway') {
        $validated['client_input_config'] = 'none';
        $validated['client_status_default'] = 'pending_verification';
        $validated['internal_input_config'] = 'none';
        $validated['internal_status_default'] = 'pending_verification';
    }

    PaymentMethod::create($validated);

    return redirect()->route('admin.payment-methods.index')->with('success', 'Metode pembayaran baru berhasil dibuat.');
}
```

Gambar 4. 96 Kode Porgram Modul Metode Pembayaran

Pada tahap penerimaan input melalui fungsi store dan update, sistem memproses nilai sakelar (toggle) visibilitas dari antarmuka menggunakan instruksi $request->boolean(). Metode pembacaan ini digunakan untuk memastikan bahwa format masukan pengguna dikonversi secara presisi menjadi tipe data boolean murni sebelum ditampung ke dalam variabel $validated['is_available_for_client'] dan $validated['is_available_for_admin'].

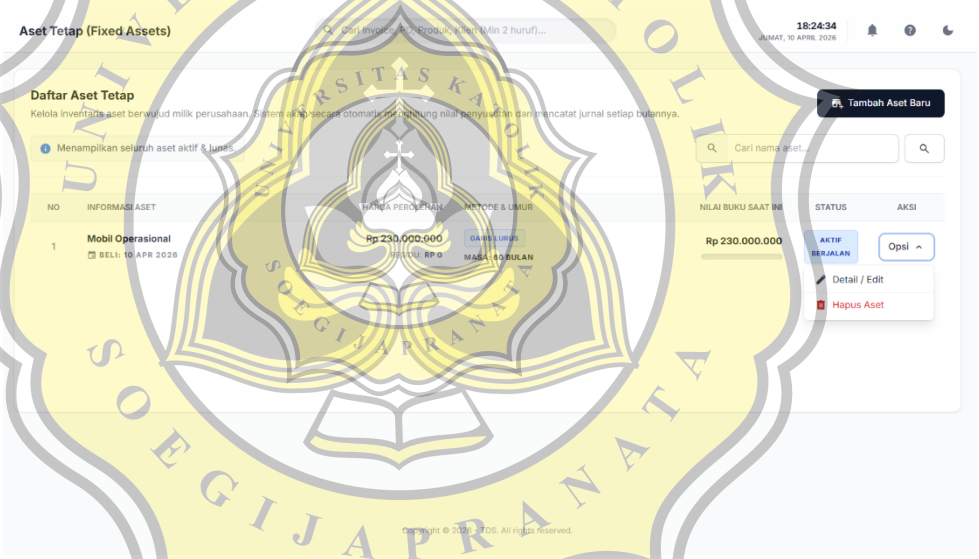
Selanjutnya, modul ini menjalankan evaluasi kondisi untuk menyesuaikan tipe sistem pembayaran. Apabila nilai dari variabel $validated['type'] dideteksi bukan sebagai 'gateway', sistem akan memodifikasi variabel $validated['gateway_provider'] = null untuk membersihkan data penyedia API yang tidak relevan.

Sebaliknya, jika tipe yang dipilih adalah 'gateway' (seperti Midtrans), sistem akan mengeksekusi blok kondisi yang mengabaikan pengaturan manual dari pengguna. Modul akan memaksakan nilai $validated['client_input_config'] = 'none' serta menahan status awal pada $validated['client_status_default']

= 'pending_verification'. Manipulasi variabel ini diterapkan karena metode berbasis *Gateway API* beroperasi secara mandiri; sistem tidak perlu meminta unggahan bukti transfer fisik dari klien, melainkan akan menahan status transaksi hingga mendapatkan respon konfirmasi otomatis (*webhook*) dari *server* pihak ketiga.

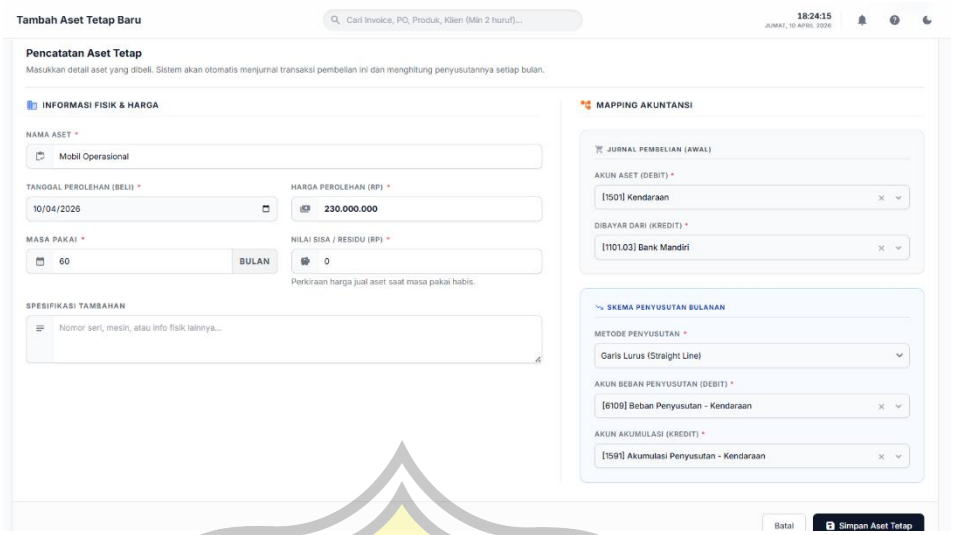
4.3.19 Modul Manajemen Aset Tetap

Modul Aset Tetap difungsikan untuk mendata inventaris aset berwujud milik perusahaan sekaligus mengotomatisasi skema penyusutan nilainya. Antarmuka utama modul ini menampilkan tabel daftar aset yang memuat informasi harga perolehan awal, metode dan umur masa pakai, hingga nilai buku *(book value)* saat ini, seperti yang diperlihatkan pada Gambar 4.97



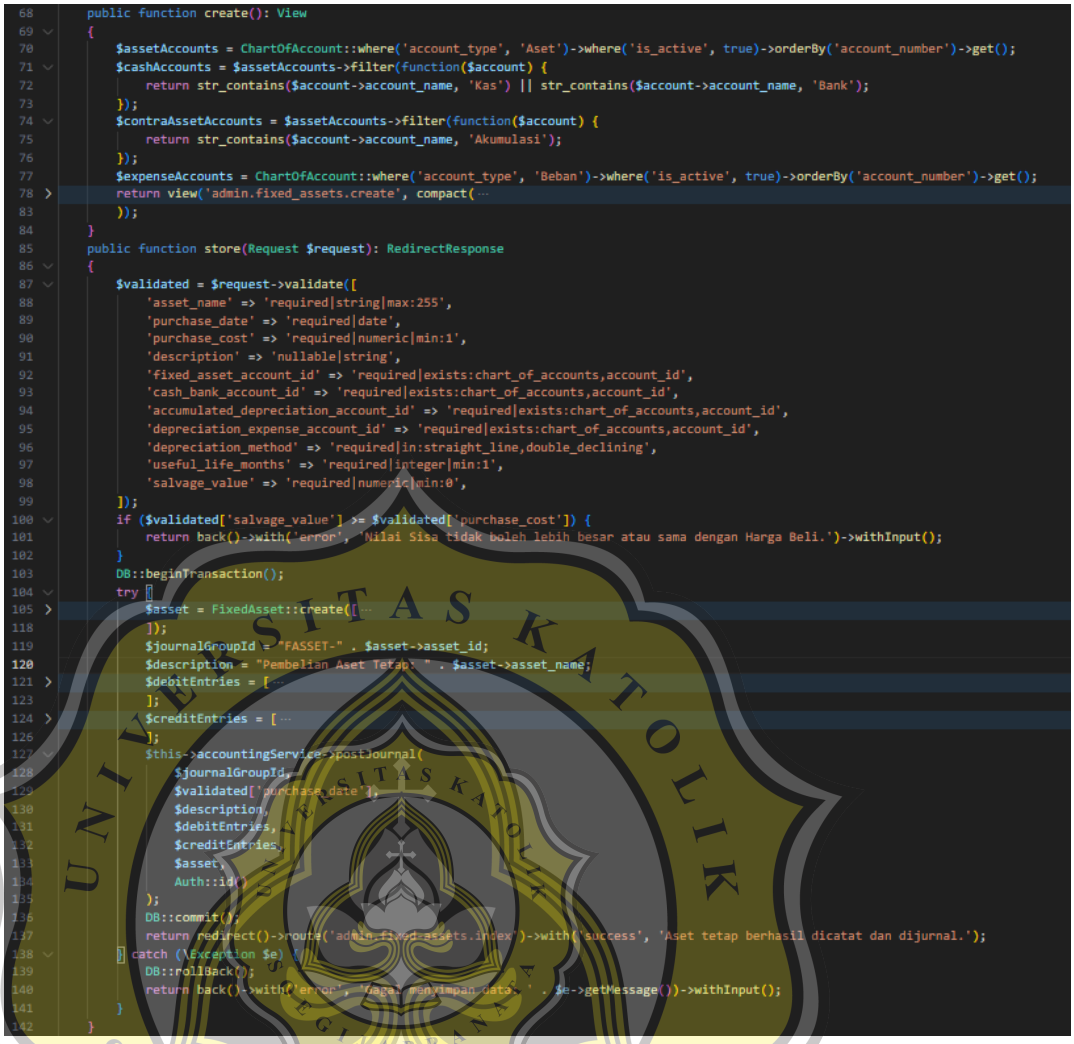
Gambar 4. 97 Halaman Daftar Aset Tetap

Untuk mencatat pengadaan aset baru, administrator menggunakan formulir pada Gambar 4.98 Formulir ini menggabungkan pencatatan spesifikasi fisik barang dengan pemetaan pembukuan. Selain mengisi nominal dan tanggal beli, pengguna diwajibkan untuk menentukan metode penyusutan (misalnya garis lurus) serta mengaitkannya dengan akun Buku Besar untuk alokasi debit aset, kredit pembayaran, hingga akun beban penyusutannya.



Gambar 4. 98 Halaman Tambah Aset Tetap

Alur validasi dan otomatisasi pembukuan dari modul ini dikendalikan oleh Modul FixedAssetController, yang sebagian potongan kodenya ditunjukkan pada Gambar 4.99.



```php
public function create(): View
{
    $assetAccounts = ChartOfAccount::where('account_type', 'Aset')->where('is_active', true)->orderBy('account_number')->get();
    $cashAccounts = $assetAccounts->filter(function($account) {
        return str_contains($account->account_name, 'Kas') || str_contains($account->account_name, 'Bank');
    });
    $contraAssetAccounts = $assetAccounts->filter(function($account) {
        return str_contains($account->account_name, 'Akumulasi');
    });
    $expenseAccounts = ChartOfAccount::where('account_type', 'Beban')->where('is_active', true)->orderBy('account_number')->get();
    return view('admin.fixed_assets.create', compact(···
    ));
}
public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'asset_name' => 'required|string|max:255',
        'purchase_date' => 'required|date',
        'purchase_cost' => 'required|numeric|min:1',
        'description' => 'nullable|string',
        'fixed_asset_account_id' => 'required|exists:chart_of_accounts,account_id',
        'cash_bank_account_id' => 'required|exists:chart_of_accounts,account_id',
        'accumulated_depreciation_account_id' => 'required|exists:chart_of_accounts,account_id',
        'depreciation_expense_account_id' => 'required|exists:chart_of_accounts,account_id',
        'depreciation_method' => 'required|in:straight_line,double_declining',
        'useful_life_months' => 'required|integer|min:1',
        'salvage_value' => 'required|numeric|min:0',
    ]);
    if ($validated['salvage_value'] >= $validated['purchase_cost']) {
        return back()->with('error', 'Nilai Sisa tidak boleh lebih besar atau sama dengan Harga Beli.')->withInput();
    }
    DB::beginTransaction();
    try {
        $asset = FixedAsset::create([···
        ]);
        $journalGroupId = "FASSET-" . $asset->asset_id;
        $description = "Pembelian Aset Tetap: " . $asset->asset_name;
        $debitEntries = [···
        ];
        $creditEntries = [···
        ];
        $this->accountingService->postJournal(
            $journalGroupId,
            $validated['purchase_date'],
            $description,
            $debitEntries,
            $creditEntries,
            $asset,
            Auth::id()
        );
        DB::commit();
        return redirect()->route('admin.fixed_assets.index')->with('success', 'Aset tetap berhasil dicatat dan dijurnal.');
    } catch (\Exception $e) {
        DB::rollBack();
        return back()->with('error', 'Gagal menyimpan data: ' . $e->getMessage())->withInput();
    }
}
```

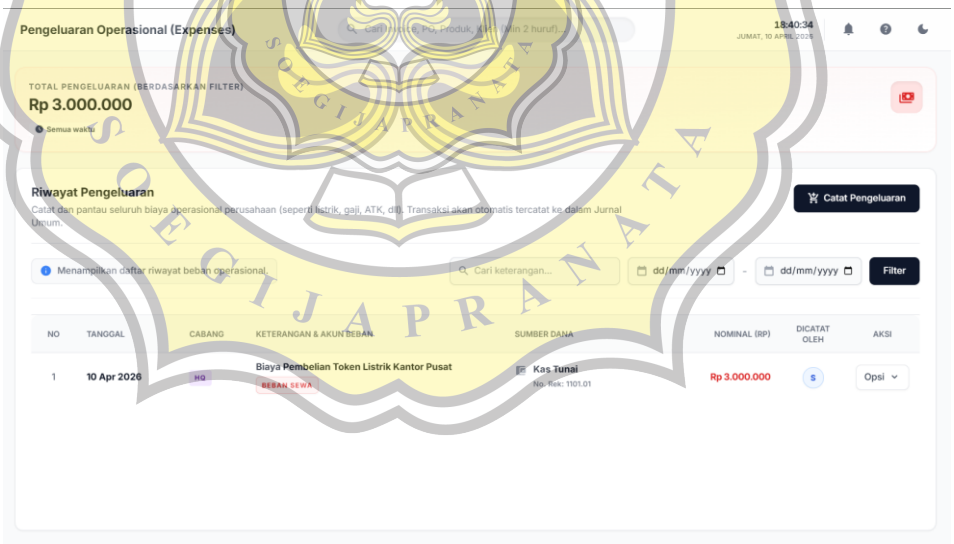
Gambar 4. 99 Kode Program Modul Fixed Assets

Pada tahap awal pemrosesan fungsi *store* dan *update*, sistem menjalankan instruksi kondisional if ($validated['salvage_value'] >= $validated['Purchase_cost']). Logika matematika ini bertugas mencegah kesalahan input dengan memastikan bahwa perkiraan nilai sisa (residu) di akhir masa pakai tidak boleh lebih besar atau sama dengan harga beli awal aset tersebut. Jika validasi ini lolos, sistem tidak hanya menyimpan data ke tabel aset, tetapi juga memanggil perintah $this->accountingService->postJournal(). Instruksi ini secara otomatis menerbitkan entri jurnal untuk mendebit akun aset dan mengkredit akun kas/bank yang dipilih pengguna saat itu juga.

Modul ini juga menerapkan kontrol ketat terhadap histori keuangan pada fungsi update dan destroy. Sebelum data diubah atau dihapus, sistem mengeksekusi kueri $fixedAsset->depreciations()->exists(). Jika kueri ini bernilai benar yang menandakan bahwa aset tersebut sudah memiliki riwayat penyusutan bulanan maka sistem akan memblokir segala bentuk modifikasi. Sebaliknya, apabila aset yang belum menyusut dihapus, sistem tidak sekadar melenyapkan baris data dari tabel. Sistem akan mengonstruksi ulang debitEntries dan creditEntries secara terbalik untuk menerbitkan jurnal pembalik *(reversal journal)*. Hal ini menjamin saldo kas kembali utuh dan nilai aset di buku besar berkurang secara sistematis.

4.3.20 Modul Manajemen Pengeluaran

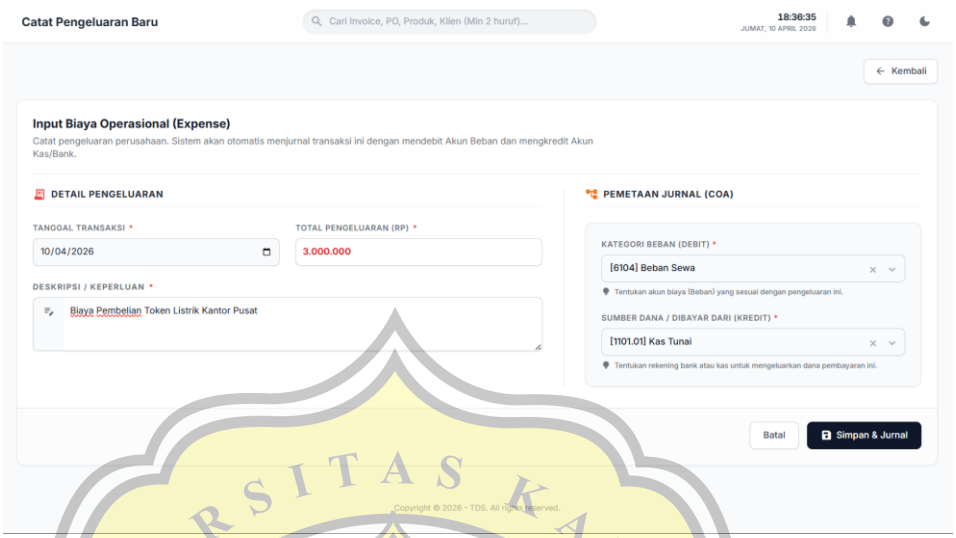
Modul Pengeluaran Operasional dikembangkan untuk mendata berbagai biaya rutin di luar pembelian aset atau persediaan barang dagang, seperti biaya listrik, gaji, dan ATK. Tampilan awal modul ini menyajikan daftar riwayat pembebanan biaya beserta fitur filter waktu dan ringkasan total pengeluaran, seperti pada Gambar 4.100.



Gambar 4.100 Halaman Daftar Pengeluaran

Untuk merekam biaya baru, pengguna mengakses formulir pencatatan pada Gambar 4.101. Pada antarmuka ini, terlihat jelas relasi langsung dengan Modul Bagan Akun (COA). Staf tidak sekadar mengetikkan nominal dan keterangan pengeluaran, tetapi diwajibkan untuk mengalokasikan akun

pemetaan jurnalnya. Pengguna harus memilih akun Kategori Beban untuk pos Debit, serta akun sumber dana (kas/bank) untuk pos Kredit.



Gambar 4. 101 Halaman Tambah Pengeluaran

Alur validasi dan otomatisasi penjurnalan untuk antarmuka tersebut dikelola secara terpusat oleh Modul ExpenseController, yang potongan kode logikanya dapat dilihat pada Gambar 4.102.

```
72      public function create(): View
73      {
74          $expenseAccounts = ChartOfAccount::where('account_type', 'Beban')->where('is_active', true)->orderBy('account_number')->get();
75          $cashAccounts = ChartOfAccount::where('account_type', 'Aset')->where('is_active', true)->orderBy('account_number')->get();
76          return view('admin.expenses.create', compact('expenseAccounts', 'cashAccounts'));
77      }
78      public function store(Request $request): RedirectResponse
79      {
80          $validated = $request->validate([
81              'expense_date' => 'required|date',
82              'amount' => 'required|numeric|min:1',
83              'description' => 'required|string|max:1000',
84              'chart_of_account_id' => 'required|exists:chart_of_accounts,account_id', // Akun Beban (E)
85              'cash_bank_account_id' => 'required|exists:chart_of_accounts,account_id', // Akun Kas/Bank (A)
86          ]);
87          if ($this->isDateClosed($validated['expense_date'])) {
88              return back()->with('error', 'Gagal: Tanggal pengeluaran masuk periode tutup buku.')->withInput();
89          }
90          $expenseAccount = ChartOfAccount::find($validated['chart_of_account_id']);
91          DB::beginTransaction();
92          try {
93              $expense = Expense::create([
94                  'branch_id' => Auth::user()->branch_id,
95                  'expense_date' => $validated['expense_date'],
96                  'amount' => $validated['amount'],
97                  'description' => $validated['description'],
98                  'user_id' => Auth::id(),
99                  'chart_of_account_id' => $validated['chart_of_account_id'],
100                 'cash_bank_account_id' => $validated['cash_bank_account_id'],
101                 'category' => $expenseAccount->account_name,
102             ]);
103
104             // 4. PERSIAPAN JURNAL UMUM
105             $journalGroupId = "EXP-" . $expense->expense_id;
106             $description = "Beban: " . $expense->description;
107             $debitEntries = [
108                 [$validated['chart_of_account_id'], $validated['amount'], "Pembebanan Biaya"]
109             ];
110             $creditEntries = [
111                 [$validated['cash_bank_account_id'], $validated['amount'], "Pembayaran Biaya"]
112             ];
113             $this->accountingService->postJournal(
114                 $journalGroupId,
115                 $validated['expense_date'],
116                 $description,
117                 $debitEntries,
118                 $creditEntries,
119                 $expense,
120                 Auth::id()
121             );
122             DB::commit();
123             return redirect()->route('admin.expenses.index')->with('success', 'Data pengeluaran berhasil disimpan dan dijurnal.');
124         } catch (\Exception $e) {
125             DB::rollBack();
126             return back()->with('error', 'Gagal menyimpan data: ' . $e->getMessage())->withInput();
127         }
128     }
```

Gambar 4. 102 Kode Program Modul Expenses

Pada tahap pembacaan data di fungsi index, sistem menerapkan aturan isolasi data multi-cabang. Menggunakan evaluasi if (!$user->isGlobal()), sistem membatasi kueri agar staf cabang hanya dapat memantau dan menghitung total pengeluaran yang terjadi di lokasi operasional mereka masing-masing (branch_id).

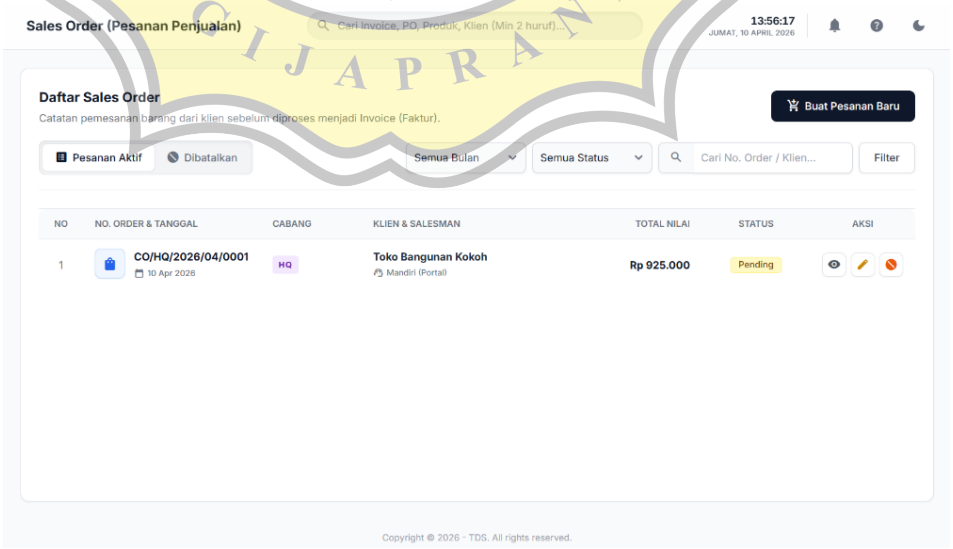
Proses rekam transaksi pada modul ini sangat bergantung pada fungsi store. Sebelum mengeksekusi penyimpanan, sistem memanggil fungsi isDateClosed untuk memastikan bahwa tanggal pengeluaran yang dimasukkan belum melewati periode tutup buku akuntansi. Jika validasi lolos, sistem tidak hanya mencatat informasi ke tabel *Expenses*. Modul ini secara otomatis menyusun struktur data ganda, yaitu $debitEntries (berisi ID akun beban) dan $creditEntries (berisi ID akun kas/bank). Kedua struktur

data tersebut kemudian diteruskan ke fungsi $this->accountingService->postJournal() untuk mencetak jurnal umum (general ledger) secara instan tanpa campur tangan manual.

Terakhir, sistem ini menerapkan perlindungan berlapis pada histori pembukuan. Apabila terdapat pengubahan data melalui fungsi update, sistem akan melacak dan menghapus entri jurnal yang lama di dalam basis data sebelum memublikasikan jurnal yang baru. Lebih jauh lagi, jika sebuah data pengeluaran dihapus melalui fungsi *destroy*, sistem akan menyusun kueri jurnal pembalik *(reversal)*. Nilai kas yang semula dikreditkan akan dimasukkan kembali ke dalam $debitEntries, sehingga saldo kas perusahaan otomatis kembali utuh sebelum dokumen pengeluaran tersebut lenyap dari sistem.

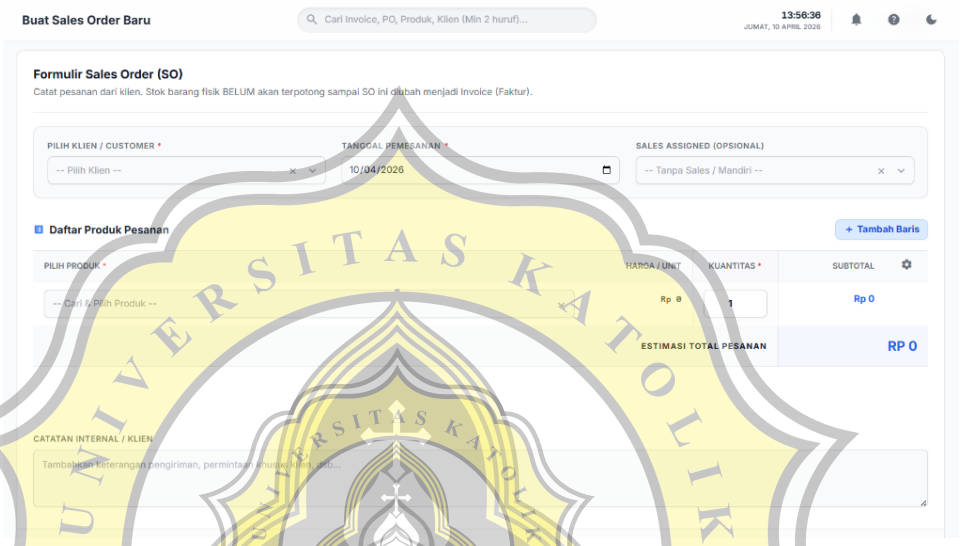
4.3.21 Modul Pencatatan Pesanan

Modul Pesanan Penjualan atau *Sales Order (SO)* dirancang sebagai tahap awal dalam siklus transaksi. Fungsi utamanya adalah untuk mendigitalisasi catatan pesanan dari klien sebelum pesanan tersebut disahkan menjadi dokumen tagihan resmi (*invoice*). Antarmuka utama modul ini adalah daftar riwayat pesanan yang dapat dilihat pada Gambar 4.103. Halaman ini dilengkapi dengan fitur filter berdasarkan bulan dan status untuk memudahkan pencarian dokumen historis.



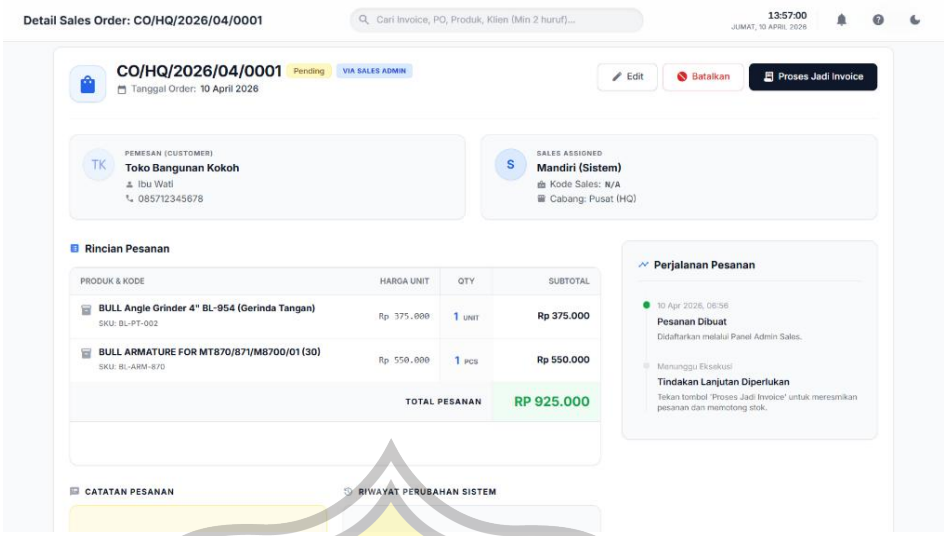
Gambar 4. 103 Halaman Utama *Sales* Order

Apabila ada permintaan barang dari klien, staf dapat menekan tombol "Buat Pesanan Baru" yang akan menampilkan formulir pendaftaran seperti pada Gambar 4.104. Pada halaman ini, staf diwajibkan untuk memilih nama klien, menentukan tanggal pemesanan, dan menunjuk petugas sales yang bertanggung jawab (opsional). Setelah itu, staf dapat memasukkan daftar produk yang dipesan beserta kuantitasnya.



Gambar 4. 104 Form Pencatatan Orderan Klien

Setelah pesanan disimpan, sistem akan menghasilkan dokumen detail pesanan berstatus "*Pending*", seperti yang ditunjukkan pada Gambar 4.105. Pada tahap ini, kuantitas stok fisik barang di dalam gudang belum mengalami pemotongan. Terdapat tombol khusus bertuliskan "Proses Jadi Invoice" yang berfungsi sebagai jembatan untuk meneruskan draf pesanan ini ke tahap penagihan dan pemotongan stok oleh staf akuntansi.



Gambar 4. 105 Halaman Detail Order

Logika penyimpanan data dari formulir tersebut ditangani oleh berkas SalesOrderController, yang potongan kodenya dapat dilihat pada Gambar 4.106.

```
117     public function create(): View
118     {
119         Gate::authorize('create', Order::class);
120         $clients = Client::orderBy('client_name')->get();
121         $products = Product::where('is_active', true)->orderBy('product_name')->get();
122         $salesUsers = User::getAvailableSales();
123 
124         return view('admin.sales_orders.create', compact('clients', 'products', 'salesUsers'));
125     }
126 
127     public function store(Request $request): RedirectResponse
128     {
129         Gate::authorize('create', Order::class);
130         $validated = $request->validate([
131             'client_id' => 'required|exists:clients,client_id',
132             'order_date' => 'required|date',
133             'sales_id' => 'nullable|exists:users,user_id',
134             'notes' => 'nullable|string',
135             'products' => 'required|array|min:1',
136             'products.*.product_id' => 'required|exists:products,product_id',
137             'products.*.quantity' => 'required|numeric|min:0.01',
138         ]);
139         try {
140             DB::beginTransaction();
141             $order = new Order();
142             $order->client_id = $validated['client_id'];
143             $order->order_date = $validated['order_date'];
144             $order->notes = $validated['notes'] ?? null;
145             $order->user_id_sales = $validated['sales_id'] ?? null;
146             $order->order_number = Order::generateOrderNumber($order->user_id_sales);
147             $order->status = 'pending';
148             $order->order_source = 'sales';
149             $order->total_amount = 0;
150             $order->save();
151             $totalAmount = 0;
152             foreach ($validated['products'] as $itemData) {
153                 $product = Product::find($itemData['product_id']);
154                 $price = $product->selling_price;
155                 $subtotal = $itemData['quantity'] * $price;
156                 $totalAmount += $subtotal;
157                 OrderItem::create([
158                     'order_id' => $order->order_id,
159                     'product_id' => $itemData['product_id'],
160                     'quantity' => $itemData['quantity'],
161                     'price_per_unit' => $price,
162                     'subtotal' => $subtotal,
163                 ]);
164             }
165             $order->update(['total_amount' => $totalAmount]);
166             DB::commit();
167             return redirect()
168                 ->route('admin.sales-orders.show', $order->order_id)
169                 ->with('success', 'Pesanan Penjualan berhasil dibuat.');
170         } catch (\Exception $e) {
171             DB::rollBack();
172             return back()
173                 ->with('error', 'Gagal membuat pesanan: ' . $e->getMessage())
174                 ->withInput();
175         }
176     }
```

Gambar 4. 106 Kode Program *Sales* Order

Proses pembuatan dokumen ini sangat bergantung pada keberhasilan function *store*. Pada fungsi tersebut, sistem menggunakan variabel $validated untuk memastikan semua data dari formulir, seperti $validated['client_id'] dan daftar produk ($validated['products']), tidak boleh kosong. Setelah tervalidasi, sistem akan menginstruksikan modul Order::generateOrderNumber($order->user_id_sales) untuk menghasilkan

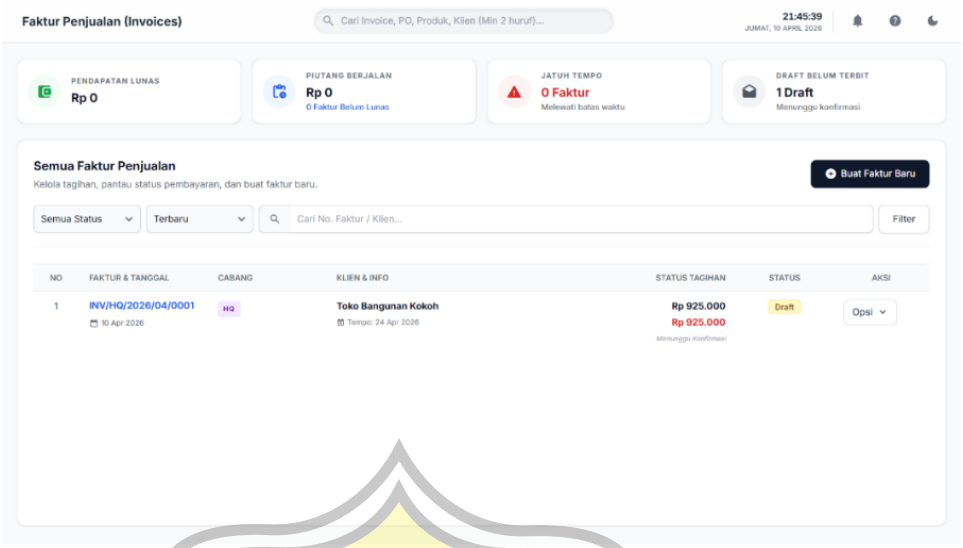
nomor dokumen referensi secara otomatis berdasarkan kode identitas petugas sales tersebut.

Selanjutnya, sistem menggunakan perulangan (*looping*) untuk membedah data produk yang dipesan. Di dalam perulangan tersebut, sistem akan memanggil variabel $product->selling_price dari basis data utama untuk mengunci harga jual barang pada saat pesanan tersebut dibuat. Hal ini dirancang agar $subtotal pesanan tidak akan berubah di masa depan, meskipun harga master produk sewaktu-waktu mengalami kenaikan. Terakhir, variabel $totalAmount akan diakumulasikan dan disimpan bersamaan dengan struktur data pesanan yang baru menggunakan skema transaksi *database* (DB::commit) untuk mencegah terjadinya data yang terputus di tengah jalan.

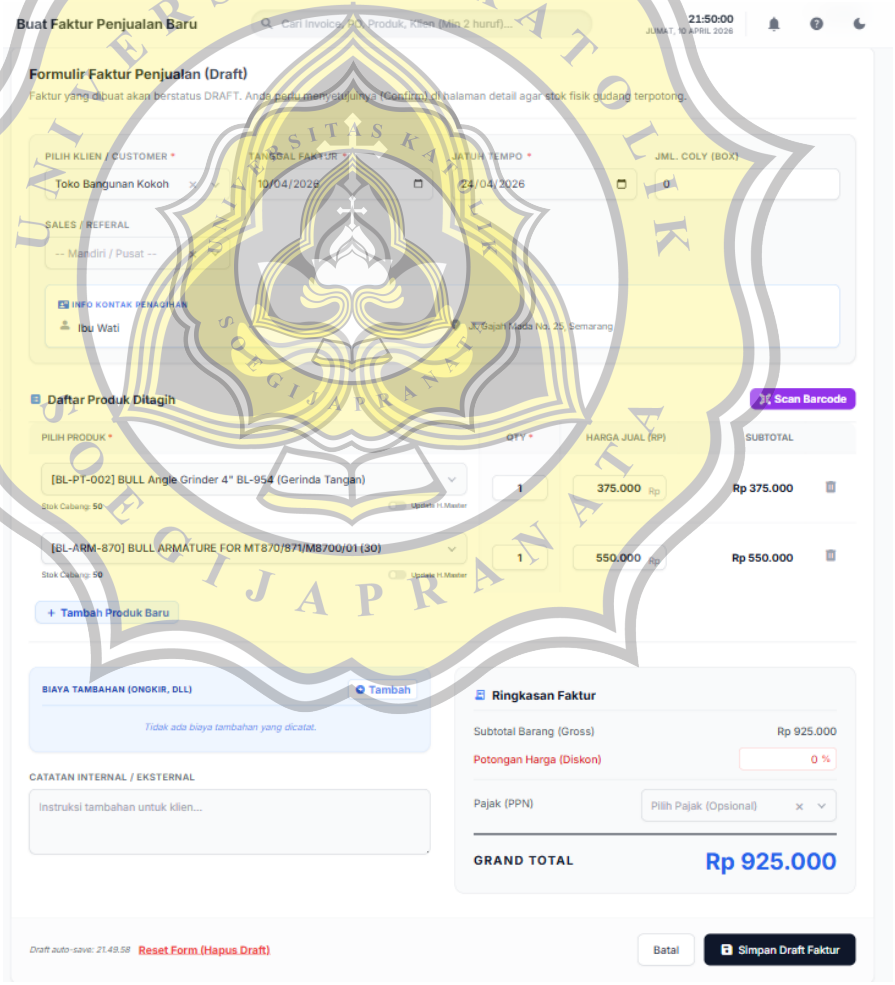
4.3.22 Modul Penerbitan Nota Penjualan

Modul Faktur Penjualan merupakan poros utama dari pencatatan piutang dan pendapatan operasional pada sistem. Tampilan beranda modul ini memuat ringkasan statistik pendapatan lunas, piutang berjalan, serta daftar seluruh faktur yang diterbitkan, seperti yang diperlihatkan pada Gambar 4.107.

Pendaftaran tagihan baru dilakukan melalui formulir pada Gambar 4.108. Selama dokumen baru disimpan, statusnya secara otomatis menjadi *draft*. Pada fase ini, stok fisik gudang belum dipotong dan jurnal akuntansi belum diterbitkan.

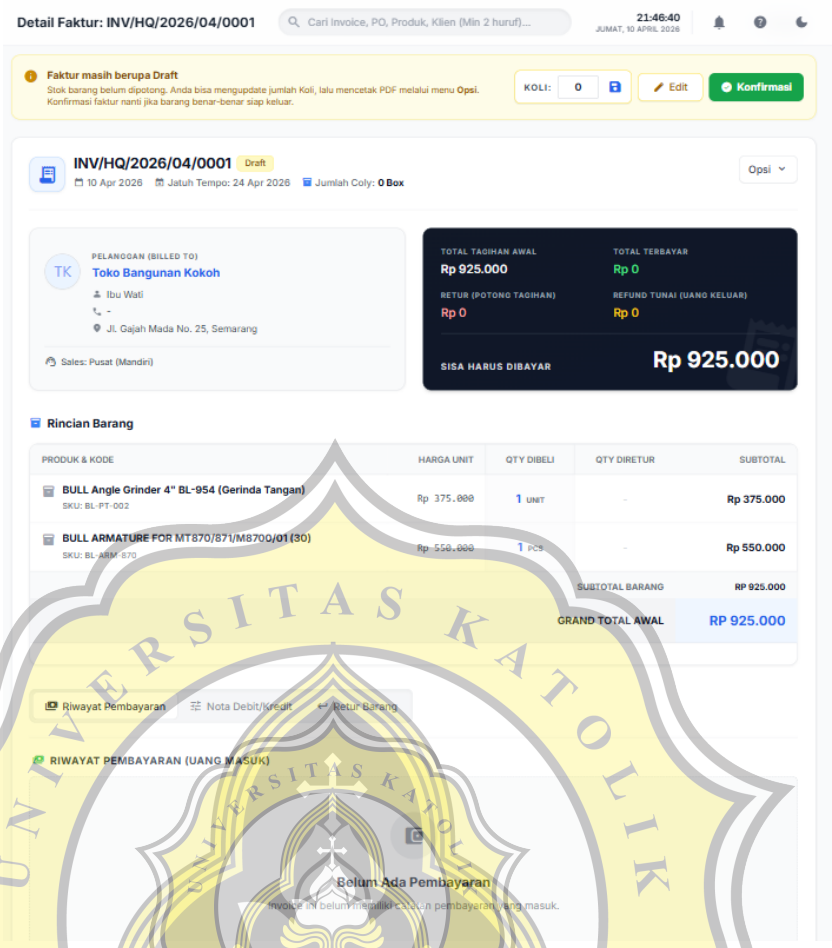


Gambar 4. 107 Halaman Daftar Invoice Penjualan



Gambar 4. 108 Halaman Pembuatan Invoice Penjualan

Pengguna harus meninjau ulang detail tagihan pada halaman Gambar 4.109 dan menekan tombol "Konfirmasi" agar dokumen tersebut sah. Alur konfirmasi ini ditangani oleh Modul SalesInvoiceController pada fungsi *confirm*. Di sinilah sistem menjalankan validasi Plafon Piutang (*Credit Limit*). Sistem mengkalkulasi variabel $piutangLama dari riwayat klien lalu menjumlahkannya dengan nilai $*invoice*->total_amount. Jika variabel $totalPotensiPiutang melebihi batas $client->credit_limit, proses konfirmasi langsung diblokir. Apabila lolos, barulah sistem mengeksekusi instruksi $stockRecord->decrement('stock_quantity', $item->quantity) untuk mengurangi fisik barang dan memanggil fungsi $this->accountingService->postJournal() untuk mencatat pergerakan akun Piutang, Pendapatan, HPP, dan Persediaan.Selama berstatus *Draft*, stok fisik di gudang belum dipotong dan jurnal akuntansi belum diterbitkan. Pengguna harus meninjau ulang detail tagihan pada halaman Gambar 4.109 dan menekan tombol "Konfirmasi" agar dokumen tersebut sah secara sistem.

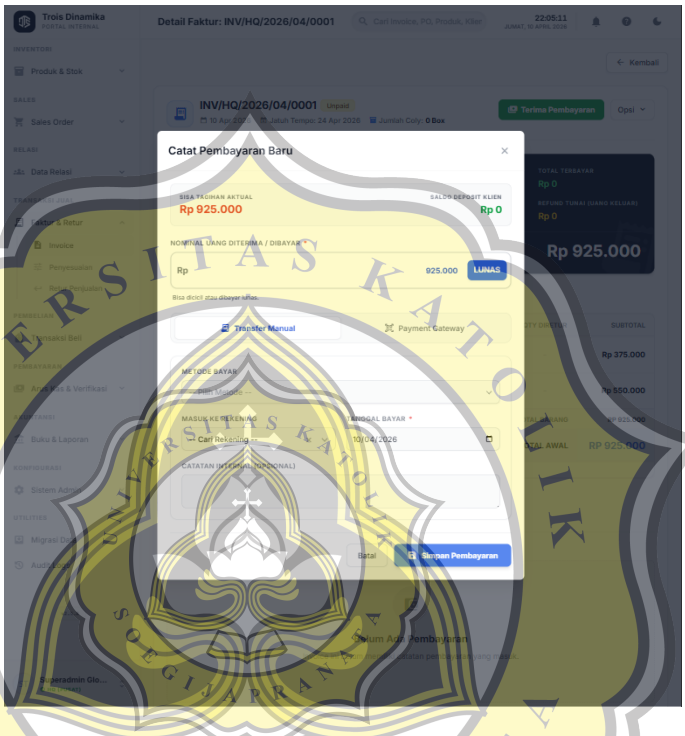


Gambar 4. 109 Halaman Detail Invoice Yang Telah Dibuat

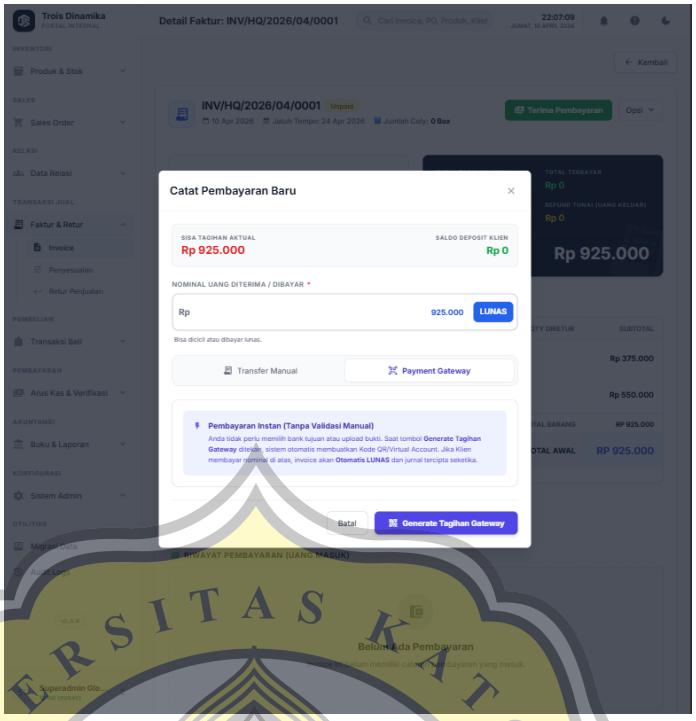
```
    $client = $invoice->client;
    if ($client && $client->has_credit_limit) {…
    }
    $arAccountId = $this->accountingSettings->getAccountsReceivableId();
    $salesRevenueAccountId = $this->accountingSettings->getSalesRevenueId();
    $cogsAccountId = $this->accountingSettings->getCogsId();
    $inventoryAccountId = $this->accountingSettings->getInventoryId();
    if (!$arAccountId || !$salesRevenueAccountId || !$cogsAccountId || !$inventoryAccountId) {
        return back()->with('error', 'Gagal: Akun default (Piutang, Pendapatan, HPP, Persediaan) belum diatur.');
    }
    DB::beginTransaction();
    try {
        $invoice->load('items');
        $totalHpp = 0;
        foreach ($invoice->items as $item) {…
        }
        $invoice->update([…
        ]);
        $journalGroupId = "INV-" . $invoice->invoice_number;
        $description = "Penjualan Invoice #" . $invoice->invoice_number . " (" . $invoice->client->client_name . ")";

        $totalAdditionalCosts = $invoice->additionalCosts()->sum('amount');
        $totalTaxJournal = $invoice->taxes()->sum('invoice_tax.amount');
        $revenueFromProducts = round($invoice->total_amount - $totalAdditionalCosts - $totalTaxJournal, 2);
        $debitEntries = […
        ];
        $creditEntries = […
        ];
        if ($totalAdditionalCosts > 0) {…
        }
        if ($totalTaxJournal > 0) {…
        }
        $this->accountingService->postJournal(…
        );
```

Gambar 4. 110 Kode Program Batas Plafon Piutang Klien

Setelah tagihan berstatus *Unpaid* (Belum Lunas), pengguna dapat mencatat penerimaan dana melalui antarmuka pembayaran pada Gambar 4.110 dan Gambar 4.111. Formulir ini terintegrasi langsung dengan master data pada Modul Metode Pembayaran dan Modul Rekening Bank yang telah dibahas pada sub-bab sebelumnya. Pemrosesan data penerimaan kas ini dieksekusi oleh Modul SalesInvoicePaymentController



Gambar 4. 111 Pop up Pencatatan Pembayaran Manual



Gambar 4. 112 Pop up Pencatatan Pembayaran Gateway

Pada implementasi modul tagihan penjualan *(Sales Invoice)*, sistem menyediakan fitur input pembayaran manual yang dikelola melalui fungsi store. Fungsi ini bertugas memproses dan memvalidasi inputan dari pengguna yang meliputi nominal bayar *(amount)*, tanggal pembayaran, metode pembayaran, bank tujuan, dan bukti transfer fisik, yang bisa dilihat pada Gambar 4.113.

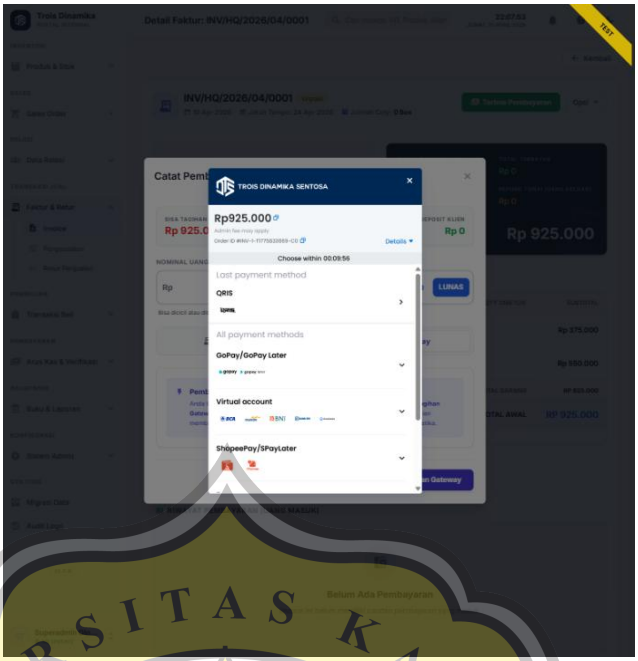
Untuk menjaga integritas pembukuan dan menyesuaikan dengan batasan sistem, algoritma pembayaran menerapkan validasi komparasi yang ketat. Sistem akan mengambil data tagihan yang terkunci (lockForUpdate) untuk mencegah bentrok data *(race condition)*, lalu membandingkan nominal bayar yang diinput dengan sisa tagihan aktual *(remaining balance)*. Sistem dikonfigurasi untuk menolak transaksi (melemparkan Exception) apabila nominal yang diinput melebihi total sisa tagihan. Apabila validasi lolos, sistem akan merekam data transaksi pembayaran tersebut dan secara otomatis memicu metode updatePaymentStatus() untuk memperbarui status faktur penjualan menjadi 'Dibayar Sebagian' (Partial) atau 'Lunas' (Paid). Seluruh rangkaian proses ini dibungkus di dalam Database

Transaction (DB::beginTransaction) untuk memastikan tidak ada data yang tersimpan sebagian apabila terjadi kegagalan pada *server*."

```php
public function store(Request $request, SalesInvoice $invoice): RedirectResponse
{
    $request->merge(['use_credit' => $request->boolean('use_credit')]);

    $inputAmount = $request->input('amount');
    $useCredit = $request->input('use_credit');

    $rules = [...
    ];

    if ($inputAmount > 0 && $request->filled('payment_method_id')) {
        $paymentMethod = PaymentMethod::find($request->input('payment_method_id'));
        $config = $paymentMethod->internal_input_config ?? 'none';

        $rules['proof_of_payment'] = in_array($config, ['proof_only', 'proof_and_reference']) ...
        $rules['reference_number'] = in_array($config, ['reference_only', 'proof_and_reference'])
            ? 'required|string|max:255' : 'nullable|string|max:255';
    }

    $validated = $request->validate($rules);

    if ($this->isDateClosed($request->payment_date)) {...
    }

    DB::beginTransaction();
    try {...
    } catch (\Exception $e) {
        DB::rollBack();
        Log::error('Payment Error: ' . $e->getMessage());
        return back()->with('error', $e->getMessage())->withInput();
    }
}
```

Gambar 4. 113 Kode Program Mekanisme Pencatatan Pembayaran *Sales Invoice*

Sebagai representasi nyata dari implementasi Teknologi Keuangan, sistem ini menyediakan opsi pembayaran otomatis melalui *Payment Gateway*. Ketika metode ini dipilih, sistem mengalihkan alur ke fungsi generateGatewayLink. Sistem menghitung sisa tagihan bersih pada variabel $grossAmount. Kemudian, sistem menyusun array $params yang memuat order_id, gross_amount, serta customer_details. Parameter ini dikirimkan ke antarmuka pemrograman aplikasi *(API)* pihak ketiga untuk mendapatkan token melalui perintah \Midtrans\Snap::getSnapToken($params). Token inilah yang memunculkan jendela pop-up pembayaran instan layaknya *e-commerce*, seperti yang terlihat pada Gambar 4.114.



Gambar 4. 114 Pop up Pembayaran via Midtrans

```php
public function generateGatewayLink(Request $request, SalesInvoice $invoice)
{
    if ($invoice->pending_snap_token && $invoice->pending_snap_expires_at > now()) {
    }

    $request->validate([
        'use_credit' => 'boolean',
        'payment_method_id' => 'required|exists:payment_methods,payment_method_id'
    ]);

    $paymentMethod = PaymentMethod::find($request->payment_method_id);
    $useCredit = $request->boolean('use_credit');

    $sisaTagihan = $invoice->real_remaining_balance;
    $clientBalance = $invoice->client->balance;

    $creditToUse = 0;
    if ($useCredit && $clientBalance > 0) {
    }

    $grossAmount = round($sisaTagihan - $creditToUse);

    if ($grossAmount <= 0) {
    }
    if ($grossAmount < 1000) {
    }

    try {
        $gateway = \App\Services\PaymentGatewayFactory::make($paymentMethod->gateway_provider);
        $response = $gateway->createTransaction($invoice, $grossAmount, $creditToUse);

        return response()->json($response);

    } catch (Exception $e) {
    }
}
```

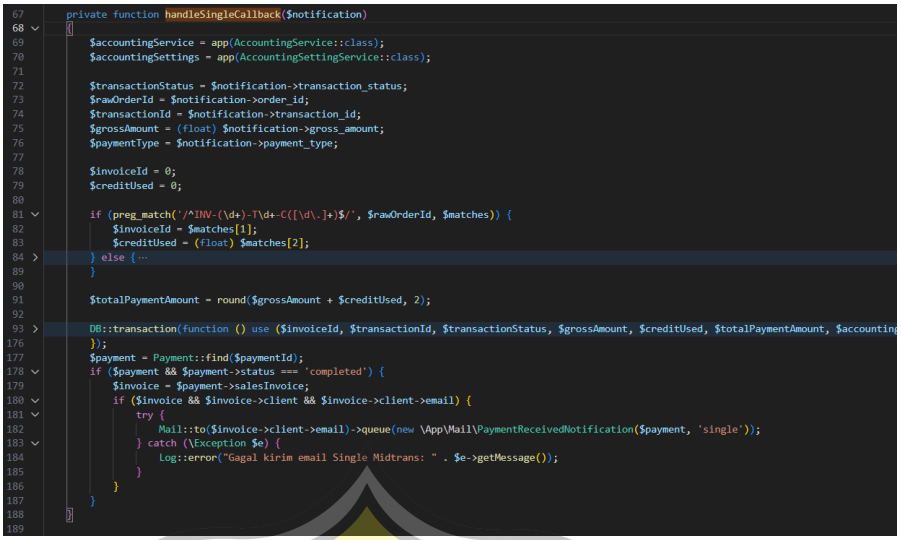
Gambar 4. 115 Kode Program GenerateGatewayLink

Sistem dilengkapi dengan kontrol integritas data yang sangat ketat pada fungsi *cancel* (Pembatalan Tagihan) dan *destroy* (Penghapusan Pembayaran). Apabila dokumen tagihan dibatalkan, sistem memeriksa apakah faktur tersebut memiliki riwayat pembayaran ($*invoice*->amount_paid > 0) atau retur ($*invoice*->*Returns*()->exists()). Jika ada, pembatalan diblokir secara keras untuk mencegah rusaknya pembukuan.

Sebagai kelanjutan dari proses inisiasi pembayaran tersebut, sistem menerapkan arsitektur *webhook* (komunikasi antar-*server*) yang ditangani oleh Modul MidtransController. Ketika klien selesai mentransfer dana, *server* Midtrans akan mengirimkan notifikasi otomatis (*callback*) ke dalam sistem. Pada fungsi callback, sistem tidak serta-merta mempercayai data yang masuk. Untuk menjamin keamanan dari manipulasi peretas, *server* melakukan verifikasi kriptografi dengan mencocokkan variabel $signature yang dienkripsi menggunakan metode SHA512 (openssl_digest) dengan signature_key bawaan Midtrans.

Apabila verifikasi keamanan lolos dan status transaksi pada variabel $transactionStatus bernilai capture atau settlement (lunas), sistem akan mengeksekusi serangkaian otomatisasi. Di dalam satu blok transaksi basis data (DB::transaction), sistem secara otomatis menerbitkan bukti pembayaran (Payment::create), mencatat jurnal akuntansi (*General Ledger*) yang mendebit akun *Gateway* dan mengkredit akun Piutang, serta merubah status faktur menjadi lunas. Sebagai sentuhan akhir dari otomasi ini, *server* akan mendelegasikan tugas pengiriman bukti bayar ke klien melalui surel (email) menggunakan instruksi latar belakang Mail::to($*invoice*->client->email)->queue(...) agar proses pemuatan antarmuka (*loading*) pada sisi pengguna tidak terbebani.

Gambar 4. 116 Kode Program MidtransController bagian function Callback

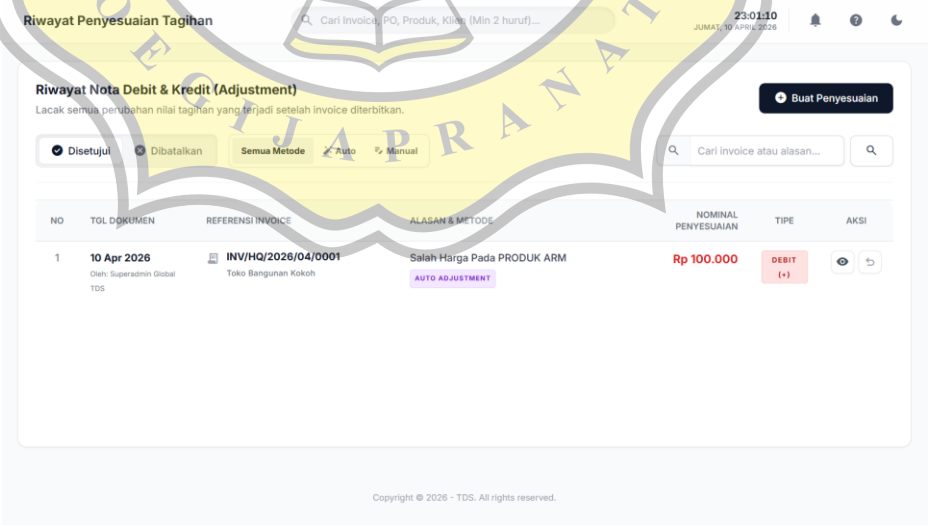


```
private function handleSingleCallback($notification)
{
    $accountingService = app(AccountingService::class);
    $accountingSettings = app(AccountingSettingService::class);

    $transactionStatus = $notification->transaction_status;
    $rawOrderId = $notification->order_id;
    $transactionId = $notification->transaction_id;
    $grossAmount = (float) $notification->gross_amount;
    $paymentType = $notification->payment_type;

    $invoiceId = 0;
    $creditUsed = 0;

    if (preg_match('/^INV-(\d+)-T\d+-C([\d\.]+)$/', $rawOrderId, $matches)) {
        $invoiceId = $matches[1];
        $creditUsed = (float) $matches[2];
    } else { ···
    }

    $totalPaymentAmount = round($grossAmount + $creditUsed, 2);

    DB::transaction(function () use ($invoiceId, $transactionId, $transactionStatus, $grossAmount, $creditUsed, $totalPaymentAmount, $accounting
    });
    $payment = Payment::find($paymentId);
    if ($payment && $payment->status === 'completed') {
        $invoice = $payment->salesInvoice;
        if ($invoice && $invoice->client && $invoice->client->email) {
            try {
                Mail::to($invoice->client->email)->queue(new \App\Mail\PaymentReceivedNotification($payment, 'single'));
            } catch (\Exception $e) {
                Log::error("Gagal kirim email Single Midtrans: " . $e->getMessage());
            }
        }
    }
}
```

Gambar 4. 117 Kode Program Midtrans function handleSingleCallback

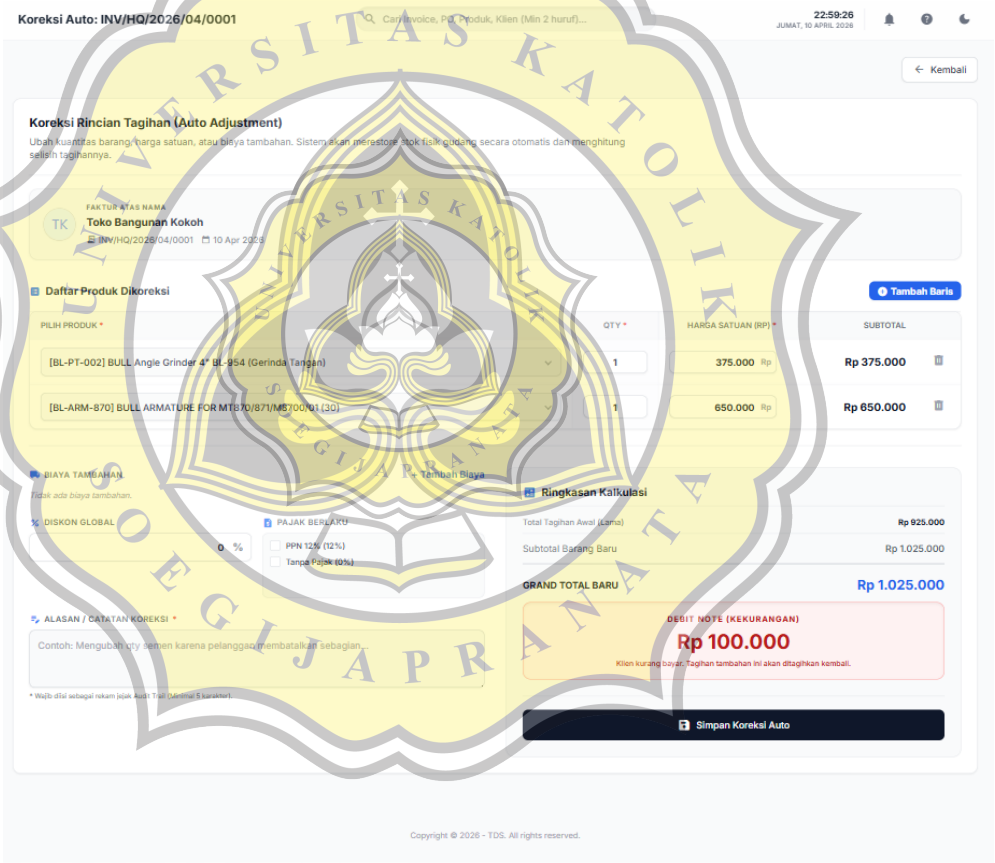
4.3.23 Modul Koreksi/ Penyesuaian Nota Penjualan

Dalam operasional bisnis, sering kali terjadi kesalahan pencatatan harga atau perubahan kuantitas barang setelah dokumen tagihan resmi (*invoice*) diterbitkan dan stok dikurangi. Untuk mengatasi masalah tersebut tanpa harus menghapus dan merusak jejak audit dokumen lama, dikembangkanlah Modul Penyesuaian Tagihan (Koreksi Nota). Riwayat seluruh penyesuaian yang pernah dilakukan dapat dilacak melalui antarmuka pada Gambar 4.118.

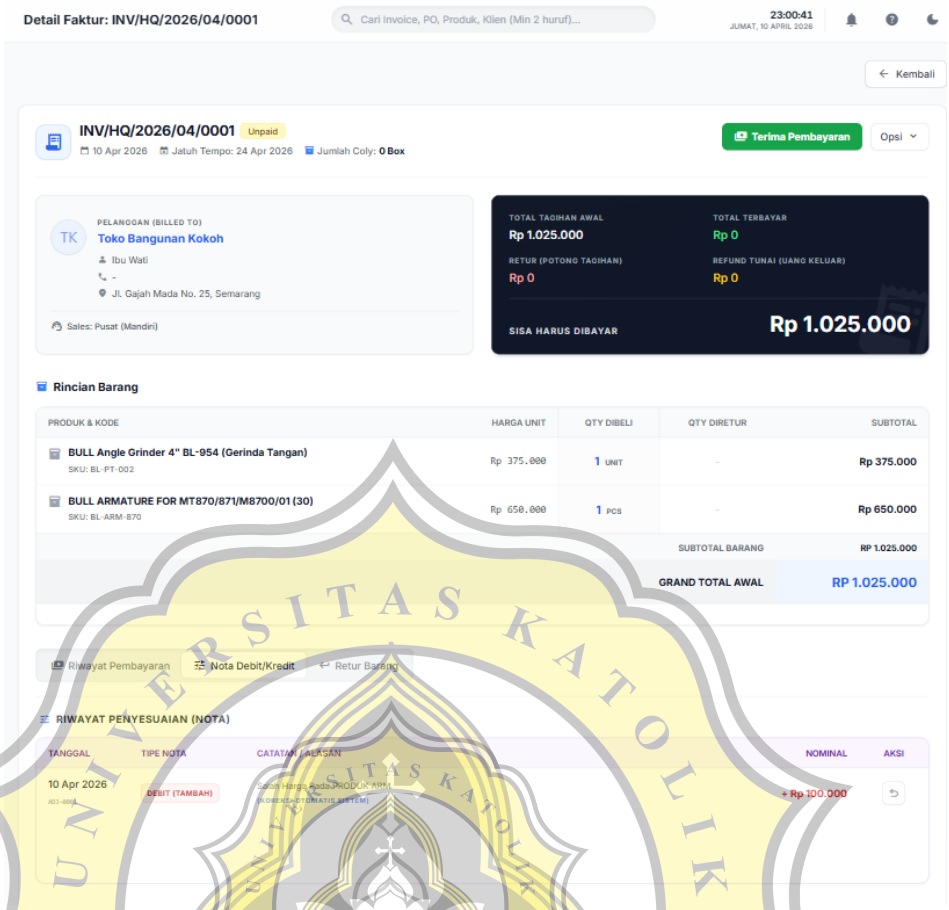


Gambar 4. 118 Halaman Riwayat Koreksi Nota Penjualan

Sistem ini memfasilitasi mode penyesuaian otomatis *(Auto Adjustment)*. Pendekatan ini diwajibkan agar nominal koreksi hutang/piutang selaras secara absolut dengan rincian fisik nota aslinya. Admin tidak sekadar mengetikkan nominal penambahan atau pengurangan, melainkan mengubah langsung kuantitas atau harga satuan produk pada antarmuka penyesuaian yang ditunjukkan pada Gambar 4.119. Sistem kemudian akan mengkalkulasi selisih secara otomatis pada bagian "Ringkasan Kalkulasi" di pojok kanan bawah, lalu menentukan apakah koreksi tersebut berupa Nota Debit (Debit Note / tagihan bertambah) atau Nota Kredit (Credit Note / tagihan berkurang).



Gambar 4. 119 Halaman Buat Koreksi Nota Penjualan

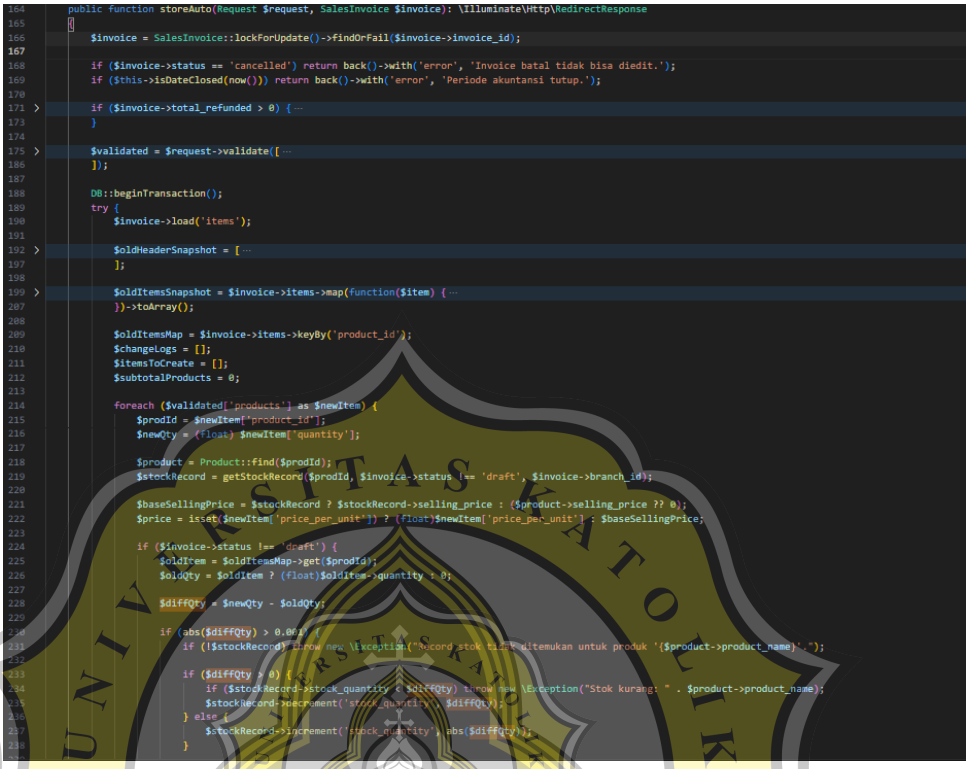


Gambar 4. 120 Detail Nota Setelah Mengalami Koreksi

Keandalan pemrosesan koreksi tingkat lanjut ini ditangani oleh Modul InvoiceAdjustmentController pada fungsi *storeAuto*. Guna mencegah bentrokan manipulasi data secara bersamaan (*race condition*), sistem mengawali proses dengan mengunci baris data faktur menggunakan instruksi $*invoice* = *SalesInvoice*::lockForUpdate()->findOrFail(...).

Setelah terkunci, sistem menyusun variabel $oldItemsSnapshot yang berisi rekam jejak (*snapshot*) kondisi kuantitas dan harga faktur sebelum diubah. Melalui metode perulangan, sistem akan membandingkan kuantitas produk yang baru ($newQty) dengan kuantitas yang lama ($oldQty) menjadi variabel selisih $diffQty. Jika terdeteksi adanya perubahan kuantitas fisik (abs($diffQty) > 0.001), sistem akan langsung mengeksekusi modul pergudangan. Jika selisihnya positif, sistem mengeksekusi $stockRecord->decrement untuk memotong stok gudang lebih lanjut. Sebaliknya, jika selisihnya negatif (barang dikurangi dari nota), sistem memanggil

$stockRecord->increment untuk memulangkan fisik barang kembali ke rak gudang asal.



Gambar 4. 121 Kode Program Bagian Kalkulasi Stok

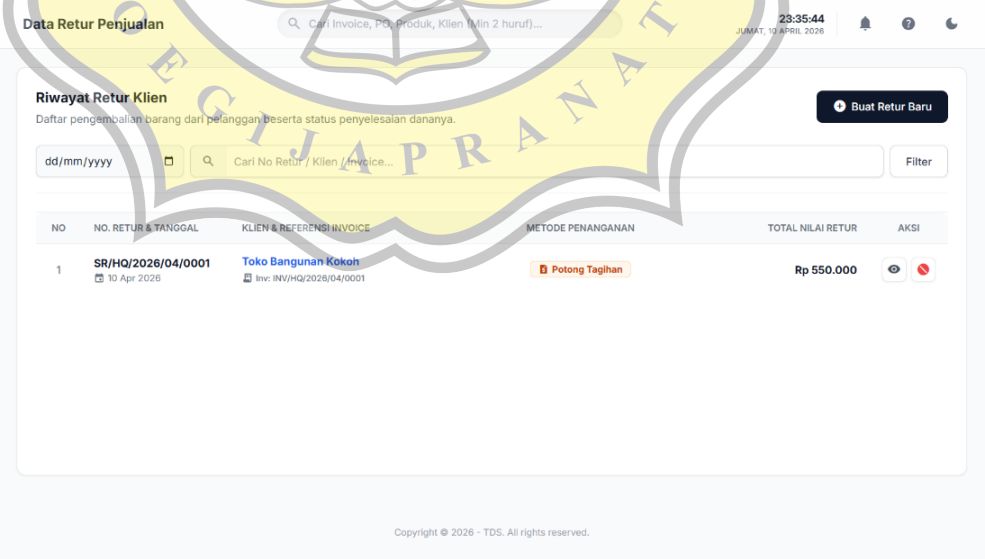
Secara paralel dari sisi akuntansi finansial, sistem mengkalkulasi selisih harga total ($diffAmount). Bergantung pada selisih tersebut, variabel $adjustmentType akan otomatis ditetapkan sebagai 'credit_note' atau 'debit_note'. Seluruh kalkulasi, rekam jejak barang lama ($oldItemsSnapshot), beserta log perubahan ($changeLogs) kemudian dibungkus ke dalam format JSON dan disimpan secara permanen ke dalam tabel *Invoice*Adjustment. Proses ini diakhiri dengan pemanggilan fungsi kustom $this->postAdjustmentJournal(...). Fungsi ini bertugas mempublikasikan jurnal umum yang akan menyeimbangkan ulang akun Piutang Usaha, Pendapatan Penjualan, atau Retur Penjualan secara presisi berdasarkan tipe nota koreksi yang diterbitkan.

```
278         $newTotalAmount = $subtotalAfterDiscount + $totalTaxAmount + $totalAdditionalCosts;
279         $oldTotalAmount = $invoice->total_amount;
280         $diffAmount = $oldTotalAmount - $newTotalAmount;
281
282         if (abs($diffAmount) <= 0.01 && empty($changeLogs)) {…
285         }
286
287         $invoice->items()->delete();
288         $invoice->additionalCosts()->delete();
289
290         if (count($itemsToCreate) > 0) {…
292         }
293         if (!empty($validated['additional_costs'])) {…
297         }
298
299         $invoice->update([…
304         ]);
305
306         if (!empty($validated['taxes'])) $invoice->taxes()->sync($validated['taxes']);
307         else $invoice->taxes()->detach();
308
309         $adjustmentType = $diffAmount > 0 ? 'credit_note' : 'debit_note';
310
311         $adjustment = InvoiceAdjustment::create([
312             'sales_invoice_id' => $invoice->invoice_id,
313             'user_id' => Auth::id(),
314             'adjustment_date' => now(),
315             'type' => $adjustmentType,
316             'amount' => abs($diffAmount),
317             'is_calculation_adjustment' => false,
318             'reason' => $validated['notes'] . (count($changeLogs) > 0 ? " (" . implode(", ", $changeLogs) . ")" : ""),
319             'details' => [
320                 'old_items_snapshot' => $oldItemsSnapshot,
321                 'old_header_snapshot' => $oldHeaderSnapshot,
322                 'change_logs' => $changeLogs
323             ]
324         ]);
325
```

Gambar 4. 122 Kode Program Bagian Kalkulasi Nominal Koreksi

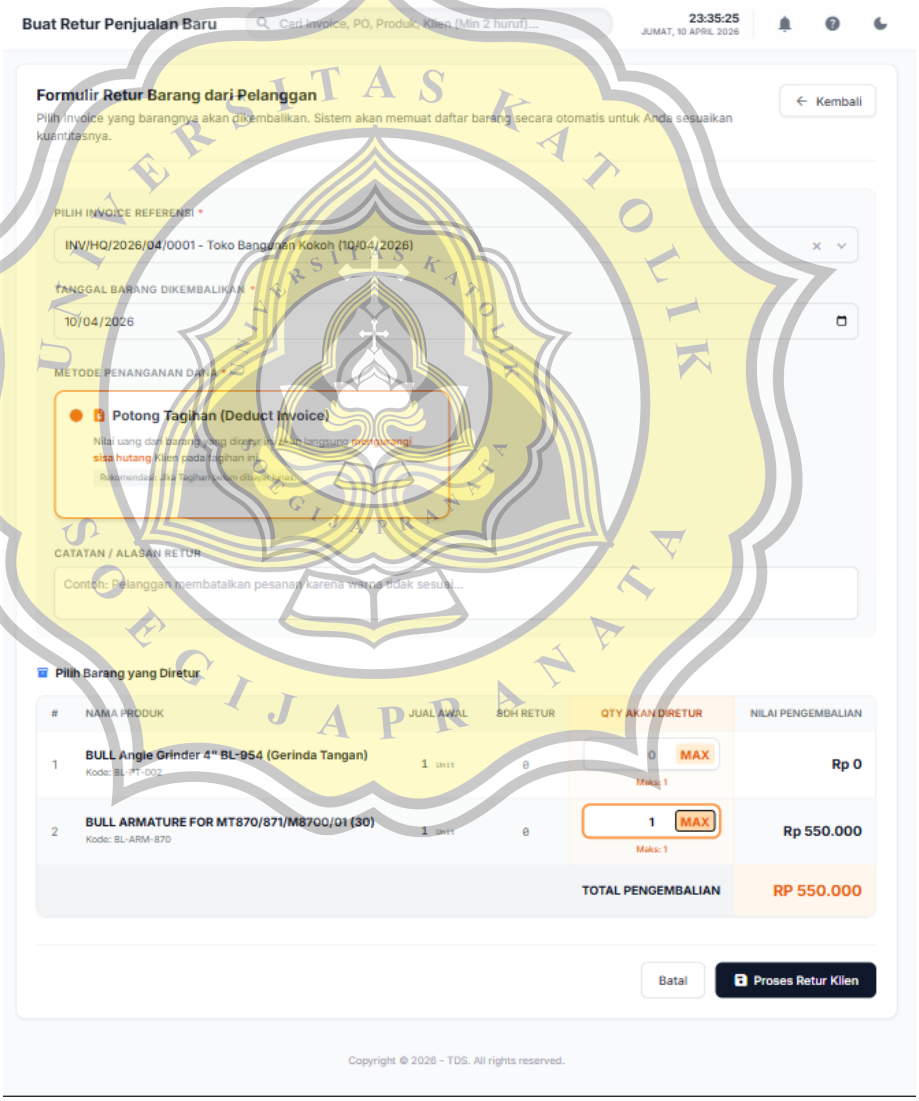
4.3.24 Modul Retur Penjualan

Modul Retur Penjualan difungsikan untuk memproses pengembalian barang dari pelanggan, baik karena kerusakan maupun ketidaksesuaian pesanan. Antarmuka riwayat retur pada Gambar 4.123 menampilkan daftar dokumen pengembalian beserta metode penanganan dana yang disepakati dengan klien.



Gambar 4. 123 Halaman Riwayat Retur Penjualan

Proses pencatatan retur baru dilakukan melalui antarmuka formulir pada Gambar 4.124. Berbeda dengan pembatalan faktur secara penuh, modul ini secara fleksibel memungkinkan terjadinya retur parsial. Sistem akan memuat daftar barang secara otomatis berdasarkan nomor faktur referensi, dan pengguna cukup menginputkan kuantitas barang yang dikembalikan pada kolom "Qty Akan Diretur". Setelah formulir divalidasi, sistem akan mengkalkulasi total nilai retur tersebut dan secara otomatis memotong sisa tagihan (piutang berjalan) pada faktur yang bersangkutan tanpa memerlukan intervensi lanjutan dari Admin.



Gambar 4. 124 Halaman Pembuatan Retur Penjualan

Seluruh jejak aktivitas dan detail pemotongan dokumen retur tersebut selanjutnya dapat ditinjau melalui halaman detail seperti yang ditunjukkan pada Gambar 4.125

Gambar 4. 125 Halaman Detail Retur Penjualan

"Alur pemrosesan data pengembalian fisik barang dan penyesuaian finansial dieksekusi secara terpadu oleh SalesReturnController pada fungsi store. Guna menjaga integritas data di lingkungan multi-cabang, sistem memvalidasi ketersediaan barang melalui parameter lokasi faktur asal (branch_id). Melalui metode perulangan (looping), sistem mengkalkulasi ulang harga bersih barang yang diretur setelah dikurangi diskon global faktur, lalu mengembalikan kuantitas fisik barang tersebut ke dalam gudang. Bersamaan dengan proses tersebut, sistem secara dinamis memperbarui nilai Harga Pokok Penjualan (HPP) rata-rata *(moving average cost)* pada master persediaan.

Tahap krusial selanjutnya adalah sinkronisasi nilai piutang dan akuntansi. Alih-alih menggunakan sistem deposit, sistem dirancang untuk langsung memotong sisa tagihan pada faktur terkait (pemotongan piutang langsung) dengan memperbarui kolom *total_returned* pada tabel *Invoice* dan menyesuaikan status pembayarannya. Sebagai langkah akhir, sistem secara otomatis mempublikasikan jurnal akuntansi double-entry ke dalam buku besar, yang mendebit akun Retur Penjualan dan Persediaan Barang, serta mengkredit akun Piutang Dagang dan Harga Pokok Penjualan (HPP). Rangkaian proses ini dibungkus dalam Database Transaction untuk menjamin data tersimpan secara atomik dan akurat."

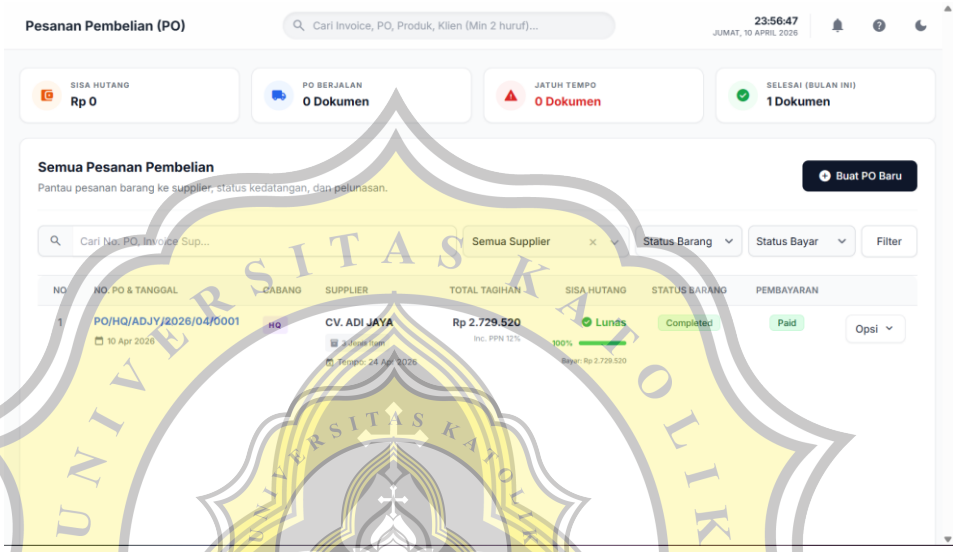
```
144         $totalReturnValue = 0;
145         $totalHppValue = 0;
146         $hasReturnedItems = false;
147         $handlingType = $validated['return_handling_type'];
148
149         $salesReturn = SalesReturn::create([ ···
160         ]);
161
162         foreach ($validated['items'] as $itemData) { ···
221         }
222
223         if (!$hasReturnedItems) { ···
225         }
226
227         $salesReturn->update([ ···
230         ]);
231
232         $journalGroupId = "SAL-RET-" . $salesReturn->return_number;
233         $description = "Retur Penjualan #" . $salesReturn->return_number . " (Inv #" . $invoice->invoice_number . ")";
234
235         $debitEntries = [ ···
238         ];
239
240         $creditEntries = [ ···
243         ];
244
245         $this->accountingService->postJournal( ···
253         );
254
255         if ($handlingType === 'store_as_credit') { ···
290         } else {
291             $totalReturDipotong = $invoice->returns()->where('return_handling_type', 'deduct_invoice')->sum('total_amount');
292             $invoice->update(['total_returned' => $totalReturDipotong]);
293             $this->handleOverpayment($invoice, $salesReturn);
294         }
295
296         $invoice->updatePaymentStatus();
297
298         DB::commit();
299         return redirect()->route('admin.sales-returns.index')
300             ->with('success', 'Retur penjualan berhasil disimpan.');
301
```

Gambar 4. 126 Kode Program SalesReturnController

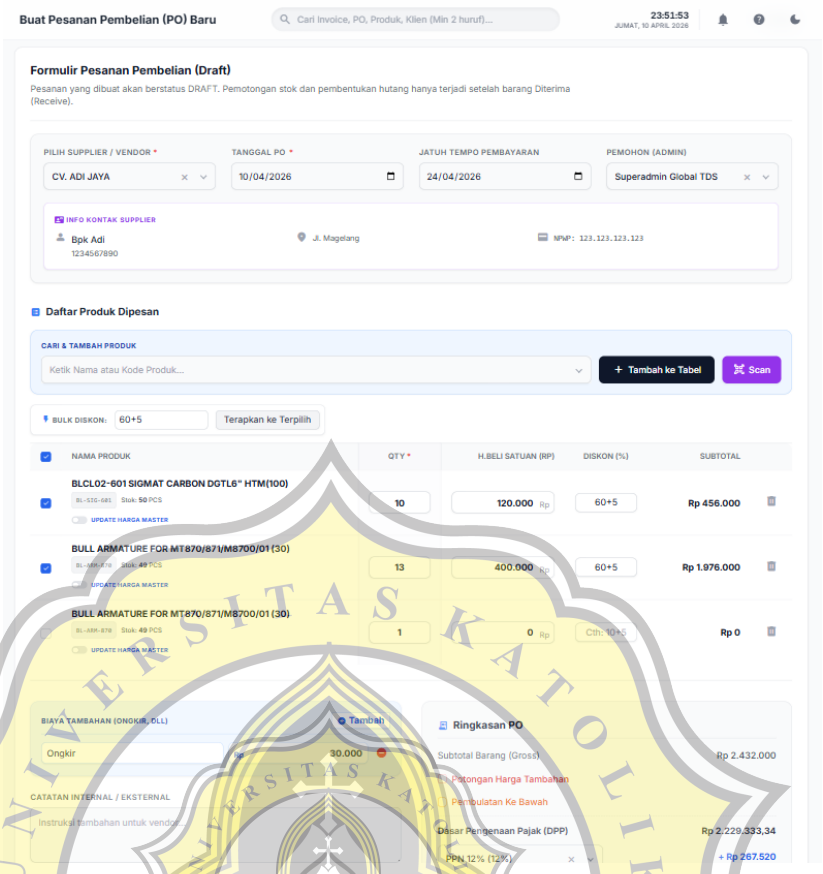
4.3.25 Modul Pembelian Barang

Modul Pesanan Pembelian *(Purchase Order/PO)* berfungsi sebagai pusat kendali pengadaan barang dan pencatatan hutang dagang perusahaan. Halaman beranda modul ini menampilkan daftar dokumen *PO* beserta indikator status barang dan status pelunasannya, seperti pada Gambar 4.127.

Untuk melakukan pengadaan baru, pengguna mengakses formulir pada Gambar 4.128. Formulir ini mencatat identitas *supplier*, daftar barang, serta kalkulasi pemotongan diskon bertingkat dan pajak. Pemrosesan awal formulir ini ditangani oleh Modul PurchaseOrderController pada fungsi *store*, di mana sistem menyimpan dokumen dalam status *Draft* tanpa merubah saldo stok maupun pembuan keuangan.

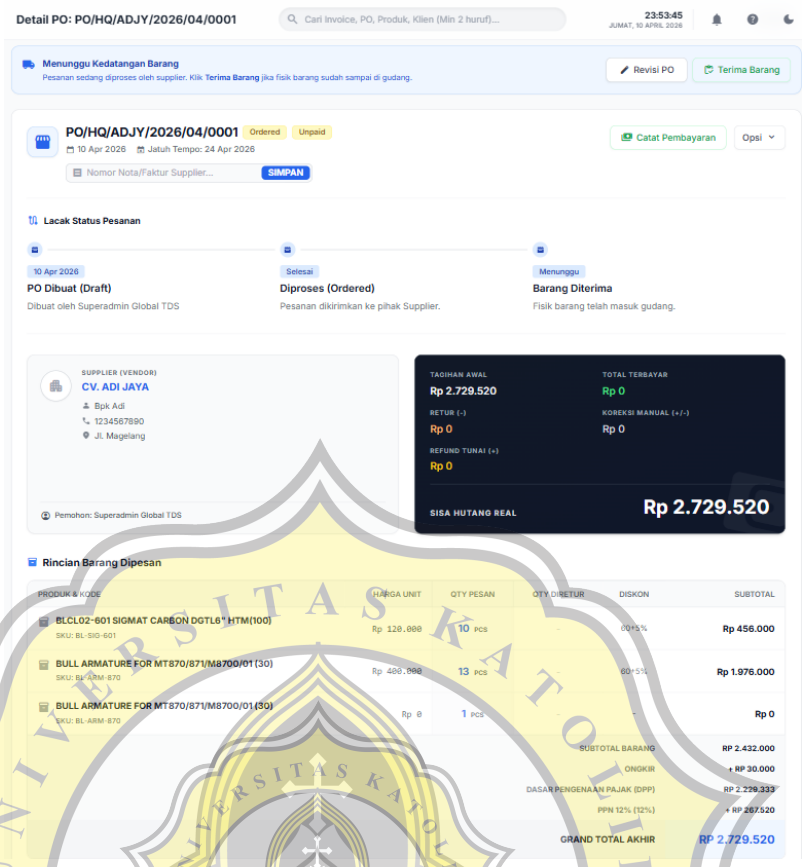


Gambar 4. 127 Halaman Daftar Pesanan Pembelian



Gambar 4. 128 Halaman Buat Pesanan Pembelian Baru

Perubahan fisik dan finansial baru terjadi ketika fisik barang telah sampai di gudang dan pengguna menekan tombol "Terima Barang" pada halaman detail PO (Gambar 4.129). Pada titik ini, sistem mengeksekusi fungsi receive. Sistem melakukan perulangan pada setiap barang yang dipesan dan menjalankan algoritma perhitungan Harga Pokok Penjualan (HPP) menggunakan metode Rata-Rata Tertimbang (*Moving Average*). Sistem meleburkan total nilai aset lama ($totalOldValue) dengan nilai aset barang yang baru masuk ($totalNewValue), lalu membaginya dengan total kuantitas gabungan ($totalNewStock) untuk menghasilkan nilai $newAvgCost yang akurat. Setelah persediaan fisik diperbarui, sistem menerbitkan jurnal akuntansi secara otomatis yang mendebit akun Persediaan Barang dan mengkredit akun Hutang Dagang (AP).



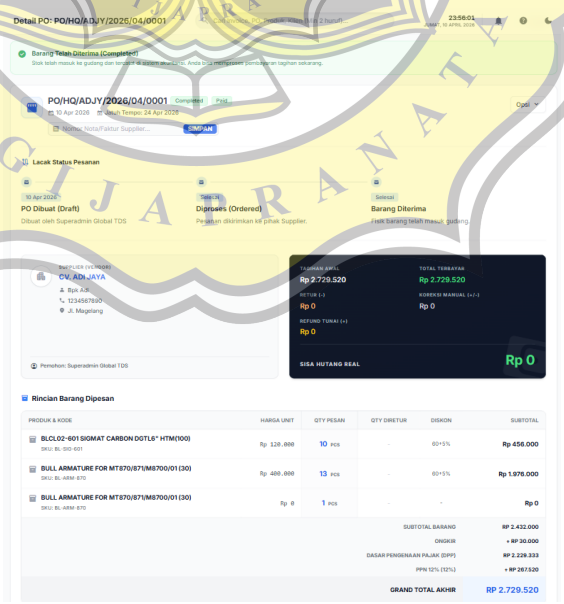
Gambar 4. 129 Halaman Detail Pesanan Pembelian

```
            $totalNewStock = $oldStock + $qtyReceived;
            $newAvgCost = $oldAvgCost;

            if ($totalNewStock > 0) {
                if ($oldStock < 0) {
                    $newAvgCost = $pricePerUnit;
                } else {
                    $totalOldValue = $oldStock * $oldAvgCost;
                    $totalNewValue = $qtyReceived * $pricePerUnit;
                    $newAvgCost = ($totalOldValue + $totalNewValue) / $totalNewStock;
                }
            } else {
                $newAvgCost = $pricePerUnit;
            }

            $stockRecord->stock_quantity = $totalNewStock;
            $stockRecord->average_cost = round($newAvgCost, 4);
            $stockRecord->save();
        }

        $purchaseOrder->status = 'completed';
        $purchaseOrder->save();

        $journalGroupId = "PO-" . $purchaseOrder->po_number;
        $description = "Penerimaan barang PO #" . $purchaseOrder->po_number . " (" . $purchaseOrder->supplier->supplier_name . ")";

        $amount = $purchaseOrder->grand_total;

        $debitEntries = [[$inventoryAccountId, $amount, "Persediaan Masuk PO #" . $purchaseOrder->po_number]];
        $creditEntries = [[$apAccountId, $amount, "Hutang Dagang PO #" . $purchaseOrder->po_number]];

        $this->accountingService->postJournal(
            $journalGroupId,
            $purchaseOrder->order_date,
            $description,
            $debitEntries,
            $creditEntries,
            $purchaseOrder,
            Auth::id()
        );
```

Gambar 4. 130 Kode Program Pembelian Barang

Setelah hutang dagang terbentuk, administrator dapat memproses pelunasan tagihan melalui antarmuka *pop-up* Catat Pembayaran Keluar seperti pada Gambar 4.131 dan Gambar 4.132. Alur transaksi kas ini dikelola secara terpisah oleh Modul PurchaseOrderPaymentController.

Gambar 4. 131 Pop up Pencatatan Pembayaran Pembelian Barang



Gambar 4. 132 Halaman Detail Pembelian Barang Ketika Sudah Lunas

Pemrosesan pelunasan hutang kepada pemasok (supplier) dikelola melalui fungsi store pada modul PurchaseOrderPaymentController. Fungsi ini mengadopsi mekanisme Strict Payment Validation, di mana sistem secara ketat membandingkan nominal uang kas yang diinputkan dengan sisa hutang berjalan *(remaining balance)*. Untuk menjaga integritas data akuntansi dan mencegah terjadinya anomali kelebihan bayar, algoritma sistem akan secara otomatis menolak transaksi dan memunculkan peringatan (Exception) apabila input nominal melebihi sisa tagihan.

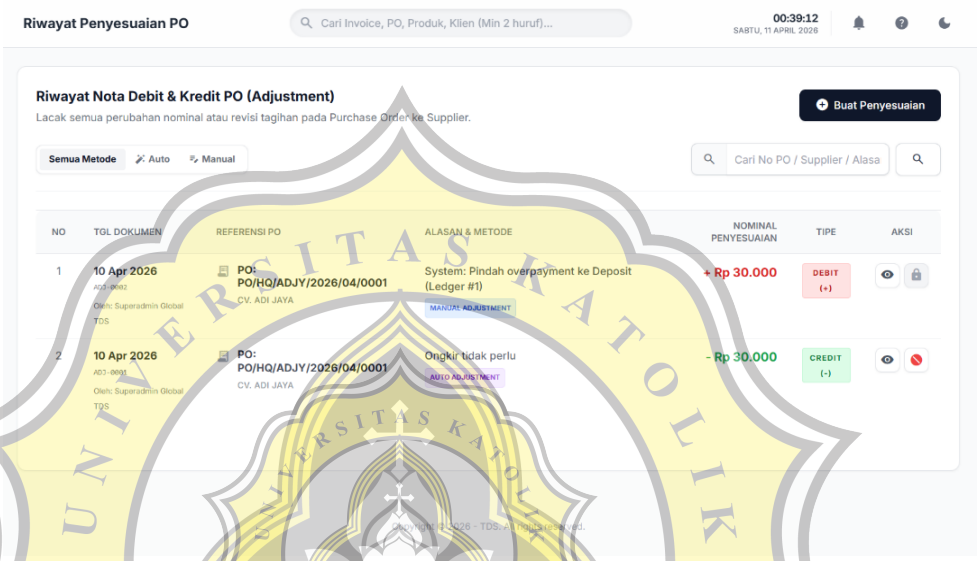
Setelah proses validasi lolos dan data tagihan dikunci melalui metode Pessimistic Locking (lockForUpdate), sistem merekam rincian pembayaran ke dalam basis data beserta dokumen bukti transfer. Sebagai penutup siklus transaksi, sistem mengeksekusi integrasi akuntansi double-entry secara otomatis. Modul akan menerbitkan jurnal yang mendebit akun Hutang Dagang (mengurangi kewajiban) dan mengkredit akun Kas/Bank (mengurangi aset) sesuai dengan rekening sumber dana yang dipilih. Seluruh pemrosesan ini dibungkus menggunakan Database Transaction untuk memastikan setiap tahapan terekam secara utuh atau dibatalkan sepenuhnya jika terjadi kendala pada *server*.

```
public function store(Request $request, PurchaseOrder $purchaseOrder): \Illuminate\Http\RedirectResponse
{
    // 1. Validasi Input Pembayaran
    $validated = $request->validate([
        'amount' => 'required|numeric|min:0.01',
        'payment_date' => 'required|date',
        'proof_of_payment' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
        'reference_number' => 'nullable|string|max:255',
        'notes' => 'nullable|string',
    ]);

    if ($this->isDateClosed($request->payment_date)) {
    }

    DB::beginTransaction();
        $poLocked = PurchaseOrder::lockForUpdate()->find($purchaseOrder->po_id);
        }

        // Ambil pemetaan akun COA
        $apAccountId = $this->accountingSettings->getAccountsPayableId();
        $cashBankAccount = CompanyBankAccount::find($validated['company_bank_account_id']);

        if (!$apAccountId || !$cashBankAccount || !$cashBankAccount->chart_of_account_id) {
        }

        // Proses Upload Bukti Pembayaran
        $proofPath = $request->hasFile('proof_of_payment')
        // 2. Simpan Rekaman Pembayaran
        $payment = $poLocked->payments()->create([
        ]);

        // 3. Posting Jurnal Akuntansi Otomatis
            [[$cashBankAccount->chart_of_account_id, $inputAmount, "Keluar dari Kas/Bank"]], // Kredit
            $payment,
        $poLocked->updatePaymentStatus();

        DB::commit();
        return back()->with('success', 'Pembayaran PO berhasil diproses dan dicatat ke Buku Besar.');

    } catch (\Exception $e) {
        DB::rollBack();
        Log::error('Gagal mencatat pembayaran PO: ' . $e->getMessage());
        return back()->with('error', 'Gagal mencatat pembayaran: ' . $e->getMessage())->withInput();
    }
```

Gambar 4. 133 Kode Program Bagian Pembayaran Pembelian Barang

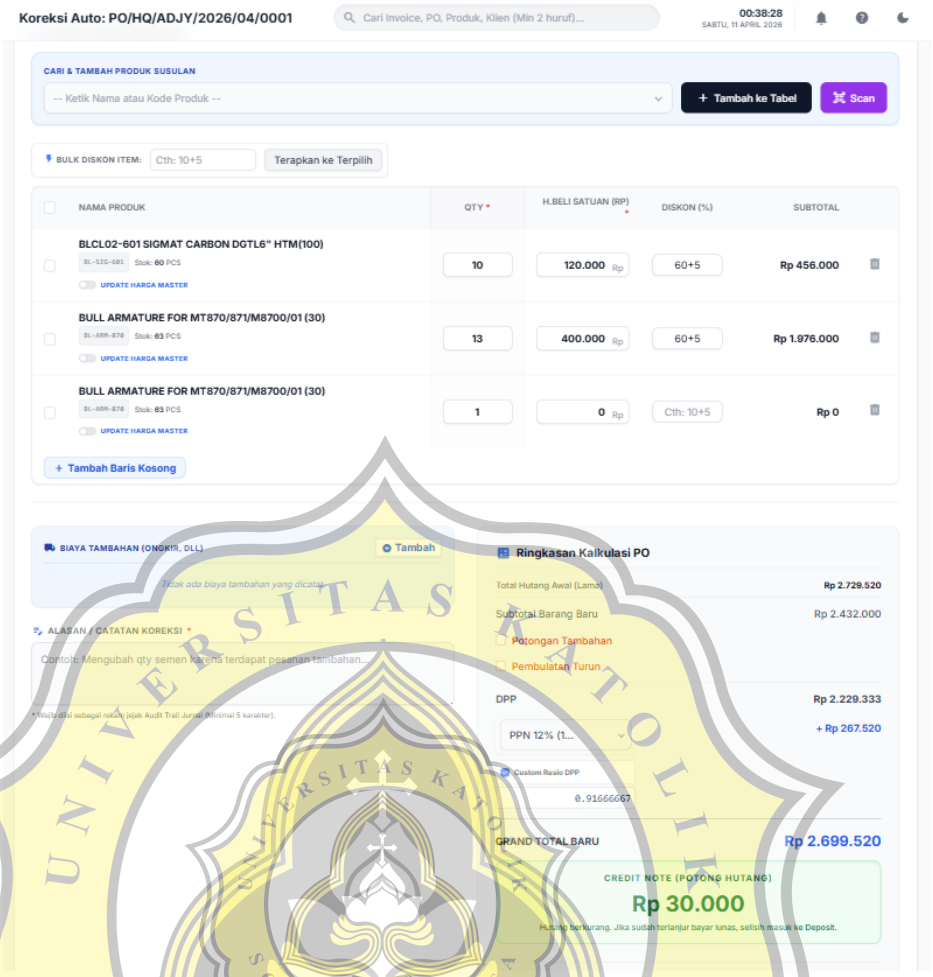
4.3.26 Modul Koreksi/ Penyesuaian Nota Pembelian Barang

Modul Penyesuaian Pesanan Pembelian difungsikan untuk merevisi rincian dokumen *Purchase Order (PO)* yang telah diterbitkan, baik karena adanya kesalahan *input* harga dari *supplier* maupun perubahan kuantitas barang saat penerimaan. Riwayat seluruh revisi dokumen dapat dilacak secara transparan melalui antarmuka pada Gambar 4.134.



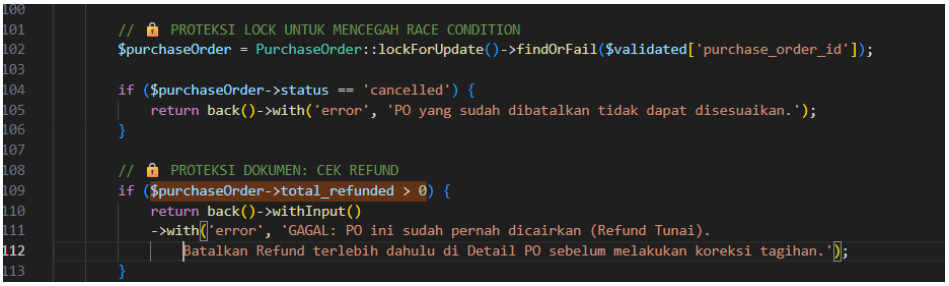
Gambar 4. 134 Halaman Riwayat Koreksi Nota Pembelian

Untuk menjaga integritas data akuntansi, sistem menggunakan mode Koreksi Otomatis (*Auto Adjustment*). Melalui formulir pada Gambar 4.135, administrator tidak memasukkan nominal selisih secara manual, melainkan langsung mengubah kuantitas atau harga satuan pada baris produk yang bersangkutan. Sistem kemudian akan membandingkan total tagihan baru dengan total tagihan lama untuk menerbitkan *Credit Note* (pengurangan hutang) atau *Debit Note* (penambahan hutang). Pada contoh gambar tersebut, sistem mendeteksi adanya pengurangan tagihan sebesar Rp 30.000 (*Credit Note*).



Gambar 4. 135 Form Pembuatan Koreksi Nota Pembelian

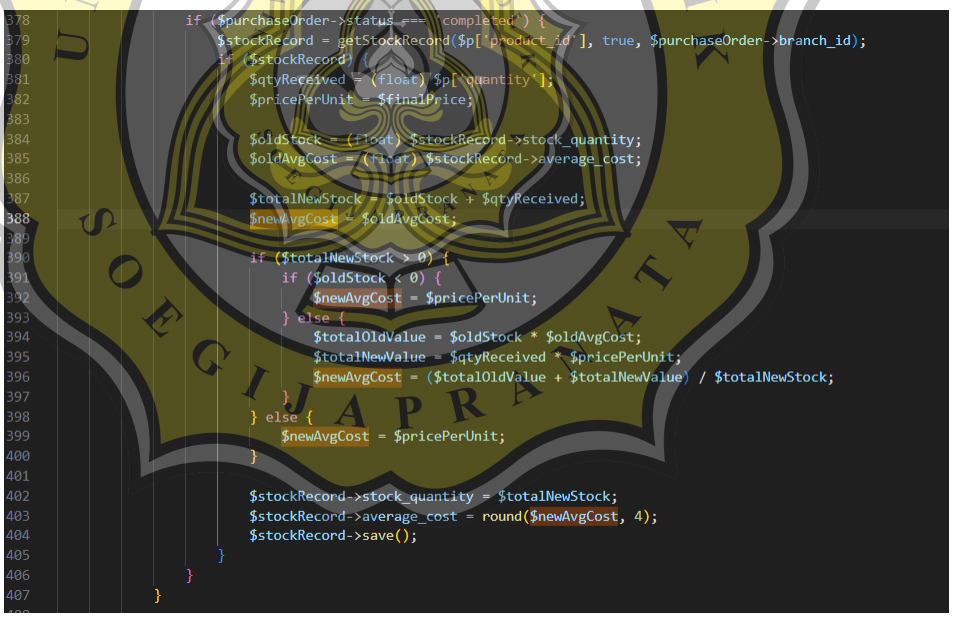
Kompleksitas pemrosesan data koreksi ini dikelola oleh Modul PurchaseOrderAdjustmentController pada fungsi *storeAuto*. Guna mencegah manipulasi data ganda, proses diawali dengan penguncian baris data melalui instruksi $*Purchase*Order = *Purchase*Order::lockForUpdate()->findOrFail(...). Sistem juga menerapkan lapis pelindung dengan memvalidasi variabel $*Purchase*Order->total_refunded > 0; jika PO tersebut sudah pernah melalui proses pencairan dana (*refund*), sistem akan memblokir upaya koreksi untuk mencegah ketidakseimbangan pembukuan kas.



```
        // 🔒 PROTEKSI LOCK UNTUK MENCEGAH RACE CONDITION
        $purchaseOrder = PurchaseOrder::lockForUpdate()->findOrFail($validated['purchase_order_id']);

        if ($purchaseOrder->status == 'cancelled') {
            return back()->with('error', 'PO yang sudah dibatalkan tidak dapat disesuaikan.');
        }

        // 🔒 PROTEKSI DOKUMEN: CEK REFUND
        if ($purchaseOrder->total_refunded > 0) {
            return back()->withInput()
            ->with('error', 'GAGAL: PO ini sudah pernah dicairkan (Refund Tunai).
                Batalkan Refund terlebih dahulu di Detail PO sebelum melakukan koreksi tagihan.');
        }
```

Gambar 4. 136 Kode Program Koreksi Nota Pembelian

Setelah validasi awal lolos, sistem melakukan pembongkaran data secara terstruktur. Jika status PO sudah diterima (*completed*), sistem terlebih dahulu menarik kembali stok fisik dari gudang ($stockRecord->decrement) sesuai kuantitas lama. Selanjutnya, sistem membaca *array* $validated['products'] yang berisi rincian baru. Sistem menghitung ulang Harga Pokok Penjualan (HPP) menggunakan skema rata-rata tertimbang ($newAvgCost), lalu memasukkan kembali stok fisik yang telah direvisi ($stockRecord->stock_quantity = $totalNewStock).



```
            if ($purchaseOrder->status === 'completed') {
                $stockRecord = getStockRecord($p['product_id'], true, $purchaseOrder->branch_id);
                if ($stockRecord) {
                    $qtyReceived = (float) $p['quantity'];
                    $pricePerUnit = $finalPrice;

                    $oldStock = (float) $stockRecord->stock_quantity;
                    $oldAvgCost = (float) $stockRecord->average_cost;

                    $totalNewStock = $oldStock + $qtyReceived;
                    $newAvgCost = $oldAvgCost;

                    if ($totalNewStock > 0) {
                        if ($oldStock < 0) {
                            $newAvgCost = $pricePerUnit;
                        } else {
                            $totalOldValue = $oldStock * $oldAvgCost;
                            $totalNewValue = $qtyReceived * $pricePerUnit;
                            $newAvgCost = ($totalOldValue + $totalNewValue) / $totalNewStock;
                        }
                    } else {
                        $newAvgCost = $pricePerUnit;
                    }

                    $stockRecord->stock_quantity = $totalNewStock;
                    $stockRecord->average_cost = round($newAvgCost, 4);
                    $stockRecord->save();
                }
            }
        }
```

Gambar 4. 137 Kode Program Koreksi Nota Pembelian

Pada tahap akhir proses penyesuaian otomatis, sistem menghitung variabel $diff yang merupakan selisih antara total tagihan awal ($oldTotalAmount) dan total tagihan baru pasca-koreksi ($newTotalAmount). Berdasarkan nilai selisih tersebut, sistem akan menentukan jenis penyesuaian apakah berupa

Nota Kredit (pengurangan hutang) atau Nota Debit (penambahan hutang). Selanjutnya, sistem mengeksekusi instruksi $this->accountingService->postJournal() untuk melakukan sinkronisasi secara *real-time* pada buku besar. Jurnal akuntansi ini secara otomatis menyesuaikan nilai pada akun Persediaan Barang dan akun Hutang Dagang *(AP)* agar selalu akurat dengan kondisi fisik dan finansial terbaru. Rangkaian proses ini ditutup dengan pembaruan status faktur dan pemutakhiran data secara permanen ke dalam basis data.

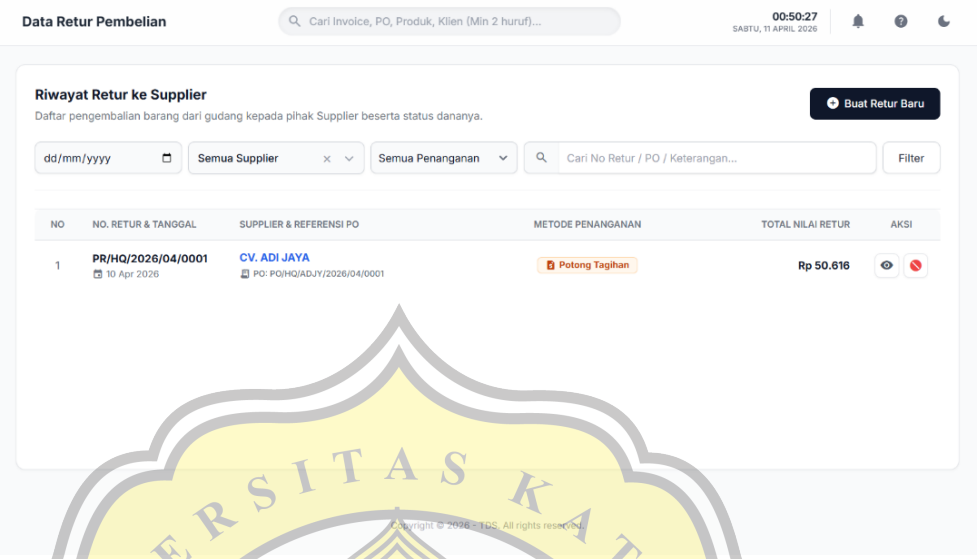
```
307         $newTotalAmount = $calc['grand_total'];
308         $oldTotalAmount = $purchaseOrder->grand_total;
309         $diff = $oldTotalAmount - $newTotalAmount;
310
311         $adjustmentType = $diff > 0 ? 'credit_note' : 'debit_note';
312         $adjustmentAmount = abs($diff);
313
314         $adjustment = PurchaseOrderAdjustment::create([
315             'purchase_order_id' => $purchaseOrder->po_id,
316             'user_id' => Auth::id(),
317             'adjustment_date' => now(),
318             'type' => $adjustmentType,
319             'amount' => $adjustmentAmount,
320             'is_calculation_adjustment' => false,
321             'reason' => $validated['notes'],
322             'details' => ['items' => $newItemsSnapshot, 'calculation' => $calc, 'diff' => $diff]
323         ]);
324
325 >       $purchaseOrder->update([...
342         ]);
343
344         // 4. SYNC ITEMS
345         $purchaseOrder->items()->each(function($item) { $item->discounts()->delete(); $item->delete(); });
346
347 >       foreach ($validated['products'] as $p) {...
397         }
398
399 >       foreach ($productsToUpdatePrice as $prodId => $price) {...
404         }
405
406         // 5. Jurnal Adjustment Auto
407 >       if ($adjustmentAmount > 0.01) {...
425         }
426
427         $purchaseOrder->updatePaymentStatus();
```

Gambar 4. 138 Kode Program Koreksi Nota Pembelian

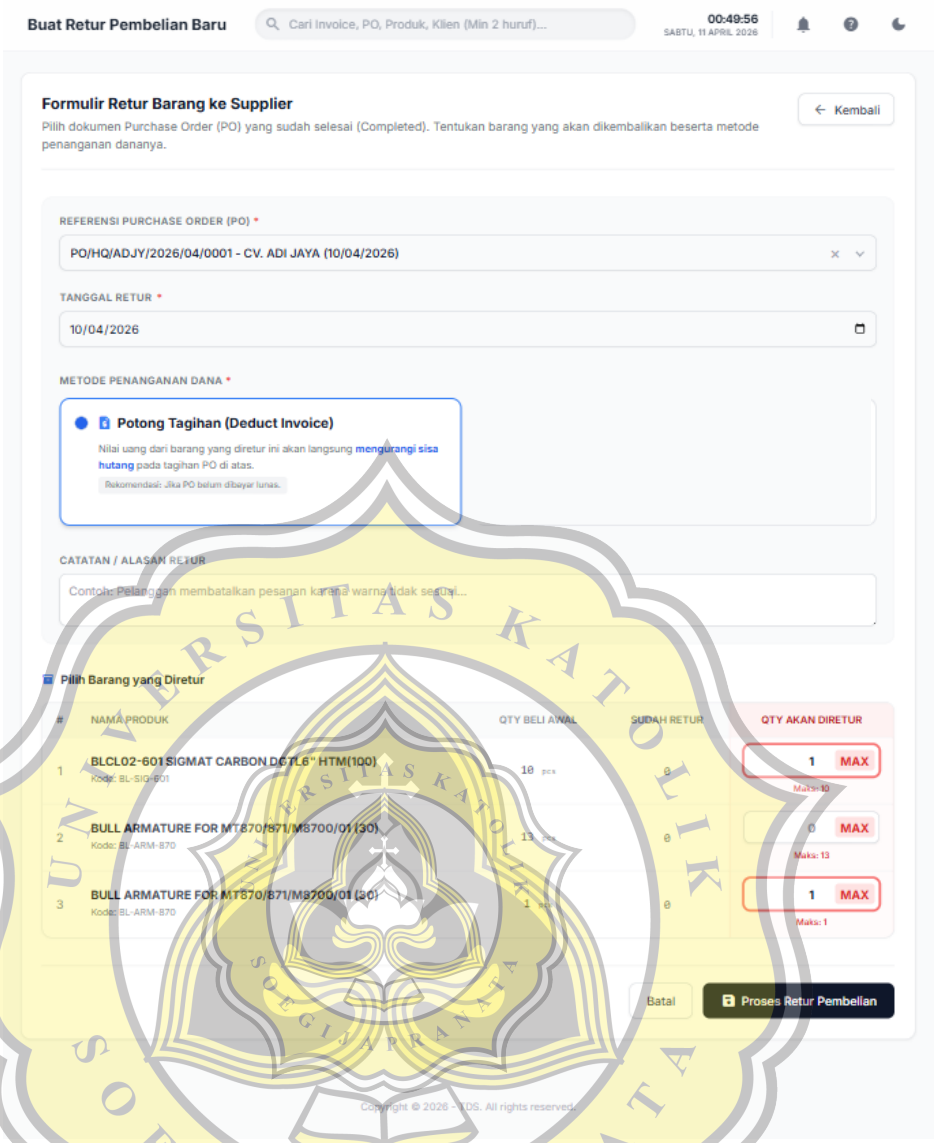
4.3.27 Modul Retur Pembelian

Modul Retur Pembelian difungsikan untuk mendokumentasikan pengembalian fisik persediaan kepada pihak *supplier* beserta penyelesaian kompensasi finansialnya. Antarmuka utama modul ini menampilkan daftar riwayat retur pembelian seperti yang ditunjukkan pada Gambar 4.139. Untuk mengajukan pengembalian barang, pengguna mengakses formulir pada Gambar 4.140. Pada antarmuka ini, sistem secara otomatis memuat daftar rincian barang dari dokumen Pesanan Pembelian *(PO)* yang sudah selesai. Pengguna diwajibkan memasukkan kuantitas yang akan diretur dan

memilih metode penanganan dana kompensasi: "Potong Tagihan" (*Deduct Invoice*) atau "Simpan Sebagai Deposit" (*Store as Credit*).



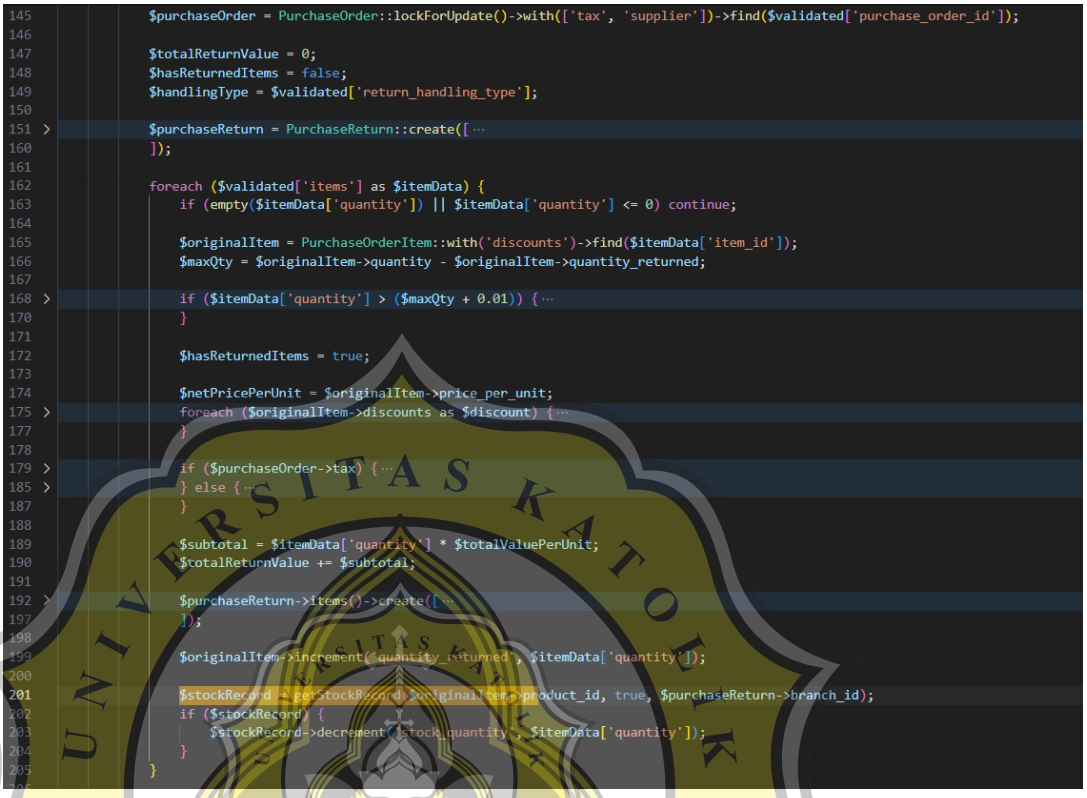
Gambar 4. 139 Halaman Riwayat Retur Pembelian



Gambar 4. 140 Halaman Buat Retur Pembelian Baru

Sinkronisasi antara pengurangan stok fisik dan pencatatan buku besar ini dikelola oleh Modul PurchaseReturnController. Saat pengguna menekan tombol simpan, fungsi store mengeksekusi penguncian transaksi melalui instruksi *Purchase*Order::lockForUpdate() untuk mencegah adanya modifikasi ganda pada dokumen PO secara bersamaan. Sistem kemudian membedah *array* produk yang dikembalikan, mengkalkulasi ulang harga bersihnya dengan memperhitungkan diskon awal ($netPricePerUnit), dan memotong kembali stok fisik dari gudang menggunakan perintah $stockRecord->decrement('stock_quantity', $itemData['quantity']). Setelah nilai total retur ($total*Return*Value) didapatkan, sistem menerbitkan jurnal

akuntansi yang mendebit akun Hutang Dagang *(AP)* dan mengkredit akun Persediaan Barang.



```
        $purchaseOrder = PurchaseOrder::lockForUpdate()->with(['tax', 'supplier'])->find($validated['purchase_order_id']);

        $totalReturnValue = 0;
        $hasReturnedItems = false;
        $handlingType = $validated['return_handling_type'];

        $purchaseReturn = PurchaseReturn::create([ ···
        ]);

        foreach ($validated['items'] as $itemData) {
            if (empty($itemData['quantity']) || $itemData['quantity'] <= 0) continue;

            $originalItem = PurchaseOrderItem::with('discounts')->find($itemData['item_id']);
            $maxQty = $originalItem->quantity - $originalItem->quantity_returned;

            if ($itemData['quantity'] > ($maxQty + 0.01)) {···
            }

            $hasReturnedItems = true;

            $netPricePerUnit = $originalItem->price_per_unit;
            foreach ($originalItem->discounts as $discount) {···
            }

            if ($purchaseOrder->tax) {···
            } else {···
            }

            $subtotal = $itemData['quantity'] * $totalValuePerUnit;
            $totalReturnValue += $subtotal;

            $purchaseReturn->items()->create([···
            ]);

            $originalItem->increment('quantity_returned', $itemData['quantity']);

            $stockRecord = getStockRecord($originalItem->product_id, true, $purchaseReturn->branch_id);
            if ($stockRecord) {
                $stockRecord->decrement('stock_quantity', $itemData['quantity']);
            }
        }
```

Gambar 4. 141 Kode Program Retur Pembelian

```
        $journalGroupId = "PUR-RET-" . $purchaseReturn->return_id;
        $description = "Retur Pembelian #" . $purchaseReturn->return_number . " untuk PO #" . $purchaseOrder->po_number;

        $debitEntries = [[$apAccountId, $totalReturnValue, "Potongan hutang dari retur PO #" . $purchaseOrder->po_number]];
        $creditEntries = [[$inventoryAccountId, $totalReturnValue, "Pengembalian persediaan dari retur #" . $purchaseReturn->return_number]];

        $this->accountingService->postJournal(···
        );

        if ($handlingType === 'store_as_deposit') {···
        } else {···
        }

        DB::commit();
        return redirect()->route('admin.purchase-returns.index')···
    } catch (\Exception $e) {···
    }
}
```

Gambar 4. 142 Kode Program Retur Pembelian

Pada tahap akhir pemrosesan, sistem terprogram untuk langsung memotong total hutang berjalan pada dokumen *Purchase Order (PO)* terkait. Guna merefleksikan perubahan ini secara finansial, modul akan secara otomatis mengeksekusi penerbitan jurnal akuntansi. Jurnal ini bertugas mendebit akun Hutang Dagang (untuk mengurangi kewajiban perusahaan kepada pemasok) dan mengkredit akun Persediaan Barang (sebagai imbas dari

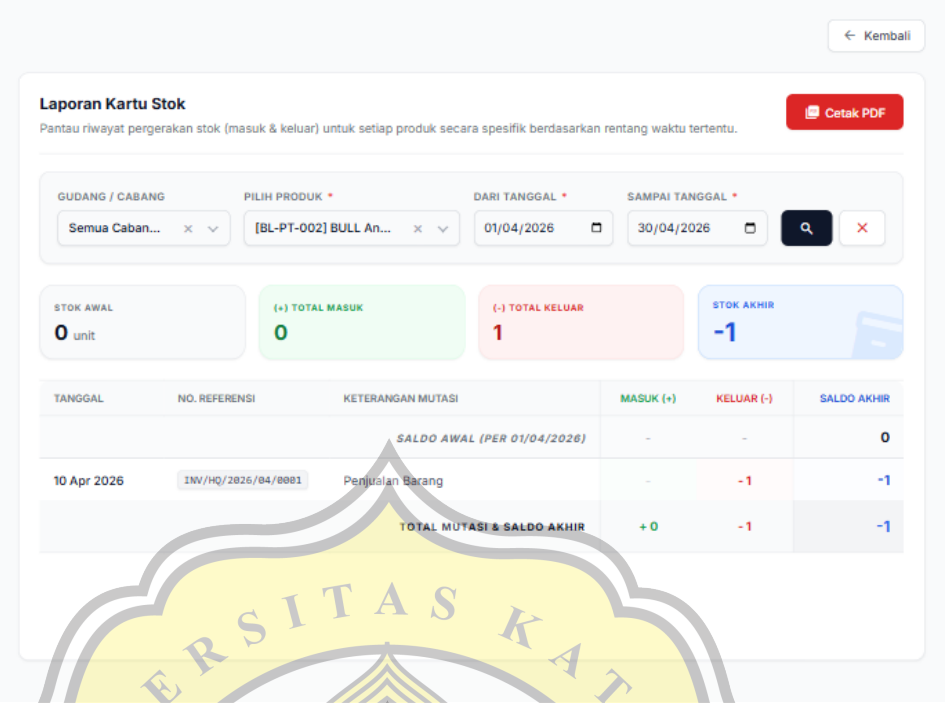
keluarnya fisik barang yang diretur). Setelah penjurnalan berhasil, sistem akan mengalkulasi ulang total akumulasi retur pada dokumen PO dan memutakhirkan status pembayarannya secara otomatis. Detail dari kompensasi pemotongan tagihan akibat retur ini dapat ditinjau ulang oleh pengguna pada antarmuka halaman detail retur (Gambar 4.143).

```php
public function store(Request $request): RedirectResponse
{
    Gate::authorize('create', PurchaseReturn::class);

    // 1. Validasi Input (Tanpa return_handling_type)
    $validated = $request->validate([ ···
    ]);

    if ($this->isDateClosed($request->return_date)) { ···
    }

    // Ambil pemetaan akun COA (Tanpa Akun Deposit)
    $apAccountId = $this->accountingSettings->getAccountsPayableId();
    $purchaseReturnAccountId = $this->accountingSettings->getPurchaseReturnId();
    $inventoryAccountId = $this->accountingSettings->getInventoryId();

    if (!$apAccountId || !$purchaseReturnAccountId || !$inventoryAccountId) { ···
    }

    // Proteksi dokumen PO yang sudah di-refund tunai
    $poCheck = PurchaseOrder::findOrFail($validated['purchase_order_id']);
    if ($poCheck->total_refunded > 0) { ···
    }

    DB::beginTransaction();
    try { ···
    } catch (\Exception $e) {
        DB::rollBack();
        Log::error('Gagal menyimpan retur pembelian: ' . $e->getMessage());
        return back()->with('error', 'Gagal menyimpan retur: ' . $e->getMessage())->withInput();
    }
}
```

Gambar 4. 143 Kode Program Retur Pembelian

4.3.28 Modul Laporan Kartu Stok

Modul Laporan Kartu Stok dikembangkan sebagai alat audit untuk melacak rekam jejak mutasi persediaan barang secara terperinci. Antarmuka utama modul ini, seperti yang disajikan pada Gambar 4.144, menyediakan instrumen penyaringan (*filter*) berdasarkan lokasi cabang operasional, spesifikasi produk, dan rentang waktu. Hasil pencarian disajikan dalam bentuk tabel buku besar persediaan yang merangkum nilai stok awal, akumulasi pergerakan barang masuk dan keluar, hingga posisi stok akhir.



Gambar 4. 144 Halaman Kartu Stok

Proses agregasi data historis ini dikelola oleh Modul StockCardController. Saat pengguna menekan tombol pencarian, sistem pertama-tama menerapkan isolasi data melalui variabel $filterBranchId untuk memastikan bahwa staf cabang hanya dapat menarik laporan mutasi dari gudang mereka sendiri. Setelah itu, sistem mengkalkulasi variabel $openingStock (saldo awal) dengan memanggil fungsi calculateStockMovement. Logika pada fungsi ini bekerja dengan menghitung seluruh riwayat transaksi produk sejak sistem pertama kali mencatat data (direpresentasikan dengan tanggal acuan '01-04-2026') hingga satu hari sebelum rentang tanggal yang dipilih oleh pengguna (Carbon::parse($startDate)->subDay()).

Untuk merangkai baris riwayat mutasi pada periode berjalan, modul ini menerapkan arsitektur pembacaan basis data yang kompleks pada fungsi getTransactions. Sistem menggunakan metode kueri unionAll yang beroperasi secara langsung pada lapisan basis data (DB::table). Metode ini menyatukan dan menyelaraskan lima tabel transaksi yang berbeda secara bersamaan, yaitu: pesanan pembelian (Purchase_order_items), faktur

penjualan (invoice_items), retur penjualan (sales_return_items), retur pembelian (purchase_return_items), dan penyesuaian opname (stock_opname_items).

```php
$user = \Illuminate\Support\Facades\Auth::user();
$filterBranchId = $user->isGlobal() ? $request->input('branch_id') : $user->branch_id;
$branches = Branch::orderBy('name')->get();

// Default tanggal: bulan ini
$startDate = $request->input('start_date', Carbon::now()->startOfMonth()->toDateString());
$endDate = $request->input('end_date', Carbon::now()->endOfMonth()->toDateString());

if ($request->filled('product_id')) {
    $selectedProduct = Product::find($request->product_id);

    if ($selectedProduct) {
        // 1. HITUNG SALDO AWAL (Opening Stock)
        $openingStock = $this->calculateStockMovement($selectedProduct->product_id, '1970-01-01', Carbon::parse($startDate)
        ->subDay()->toDateString(), $filterBranchId);

        // 2. AMBIL TRANSAKSI PERIODE INI
        $transactions = $this->getTransactions($selectedProduct->product_id, $startDate, $endDate, $filterBranchId);

        // 3. HITUNG RUNNING BALANCE
        $currentBalance = $openingStock;
        foreach ($transactions as $trx) {...
        }
        $endingStock = $currentBalance;
    }
}
```

Gambar 4. 145 Kode Program Kartu Stok

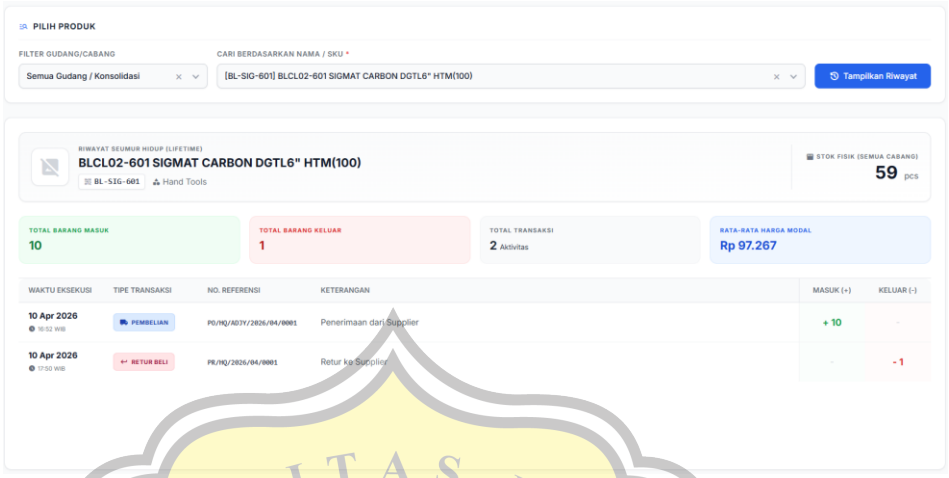
```php
private function getTransactions($productId, $start, $end, $branchId = null)
{
    $purchases = DB::table('purchase_order_items as poi')
        ->join('purchase_orders as po', 'poi.po_id', '=', 'po.po_id')
        ->where('poi.product_id', $productId)
        ->where('po.status', 'completed')
        ->whereBetween('po.order_date', [$start, $end])
        ->when($branchId, fn($q) => $q->where('po.branch_id', $branchId))
        ->selectRaw("'purchase_order' as source, po.order_date as date, po.po_number as reference,
        'Pembelian Barang' as description, poi.quantity as qty, po.created_at");

    $sales = DB::table('invoice_items as ii')...
    $salesReturns = DB::table('sales_return_items as sri')...
    $purchaseReturns = DB::table('purchase_return_items as pri')...
    $opnames = DB::table('stock_opname_items as soi')...
    return $purchases
        ->unionAll($sales)
        ->unionAll($salesReturns)
        ->unionAll($purchaseReturns)
        ->unionAll($opnames)
        ->orderBy('date')
        ->orderBy('created_at')
        ->get();
}
```

Gambar 4. 146 Kode Program Kartu Stok

4.3.29 Modul Laporan Riwayat Produk

Modul Riwayat Produk dikembangkan untuk menyajikan rekam jejak aktivitas seumur hidup (*lifetime*) dari sebuah barang, mulai dari pertama kali dibeli hingga transaksi terakhirnya. Berbeda dengan Kartu Stok yang dibatasi oleh rentang tanggal, antarmuka modul ini pada Gambar 4.147 menampilkan akumulasi total barang masuk, total barang keluar, jumlah

frekuensi transaksi, serta rincian log pergerakannya secara menyeluruh tanpa batasan periode.



Gambar 4. 147 Halaman Riwayat Produk

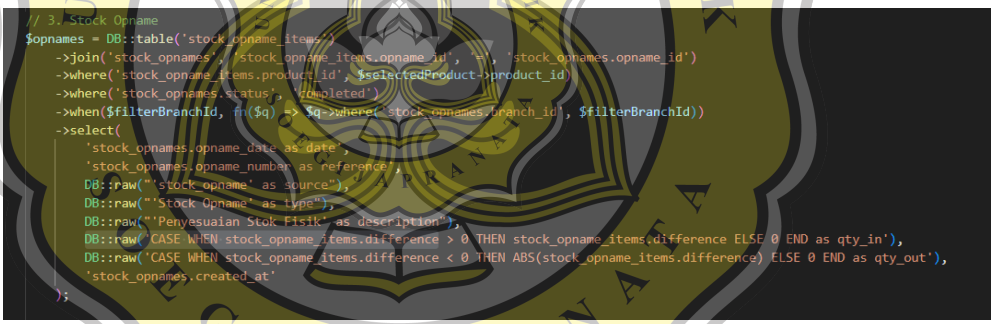
Pemrosesan penarikan data ini dikendalikan oleh Modul ProductHistoryController. Salah satu logika penting pada modul ini terletak pada awal proses inisiasi pencarian data, di mana sistem mengimplementasikan metode kueri Product::withTrashed()->find($request->product_id). Penggunaan metode ini menjamin integritas rekam jejak sistem; meskipun sebuah barang telah dinonaktifkan atau diarsipkan (*soft delete*) dari katalog master oleh Admin, data riwayat transaksinya di masa lalu tidak akan menjadi yatim (*orphan data*) dan tetap dapat ditelusuri secara utuh.

```php
class ProductHistoryController extends Controller implements HasMiddleware
{
    public static function middleware(): array
    {
        return [
            new Middleware('can:view-products', only: ['index']),
        ];
    }

    public function index(Request $request): View
    {
        Gate::authorize('view-reports');

        $products = Product::withTrashed()->orderBy('product_name')->get();
        $selectedProduct = null;
        $history = collect();

        $user = \Illuminate\Support\Facades\Auth::user();
        $filterBranchId = $user->isGlobal() ? $request->input('branch_id') : $user->branch_id;
        $branches = Branch::orderBy('name')->get();

        if ($request->filled('product_id')) {
        }

        return view('admin.reports.product-history.index', compact('products', 'selectedProduct', 'history', 'branches', 'filterBranchId'));
    }
}
```

Gambar 4. 148 Kode Program Riwayat Produk

Untuk menyajikan tabel riwayat yang komprehensif, modul ini kembali memanfaatkan metode unionAll() pada fungsi lapisan basis data (DB::table) untuk menggabungkan lima entitas transaksi yang berbeda. Keunggulan dari algoritma penggabungan pada modul ini adalah implementasi pemetaan kolom dinamis (*Dynamic Column Mapping*).

Pada setiap kueri pembangunnya, sistem secara eksplisit mendeklarasikan kolom qty_in (kuantitas masuk) dan qty_out (kuantitas keluar). Sebagai contoh, pada kueri tabel sales_*invoice*s, sistem menetapkan nilai 0 as qty_in dan nilai asli faktur pada qty_out. Sebaliknya, pada kueri tabel stock_opnames (Penyesuaian Stok), sistem mengimplementasikan logika kondisional tingkat basis data menggunakan instruksi DB::raw('CASE WHEN...'). Apabila selisih opname bernilai positif, angka tersebut akan dimasukkan ke dalam qty_in, sedangkan apabila negatif, fungsi absolut (ABS()) digunakan untuk mengubahnya menjadi bilangan positif yang dialokasikan ke kolom qty_out.



```
// 3. Stock Opname
$opnames = DB::table('stock_opname_items')
    ->join('stock_opnames', 'stock_opname_items.opname_id', '=', 'stock_opnames.opname_id')
    ->where('stock_opname_items.product_id', $selectedProduct->product_id)
    ->where('stock_opnames.status', 'completed')
    ->when($filterBranchId, fn($q) => $q->where('stock_opnames.branch_id', $filterBranchId))
    ->select(
        'stock_opnames.opname_date as date',
        'stock_opnames.opname_number as reference',
        DB::raw("'stock_opname' as source"),
        DB::raw("'Stock Opname' as type"),
        DB::raw("'Penyesuaian Stok Fisik' as description"),
        DB::raw('CASE WHEN stock_opname_items.difference > 0 THEN stock_opname_items.difference ELSE 0 END as qty_in'),
        DB::raw('CASE WHEN stock_opname_items.difference < 0 THEN ABS(stock_opname_items.difference) ELSE 0 END as qty_out'),
        'stock_opnames.created_at'
    );
```

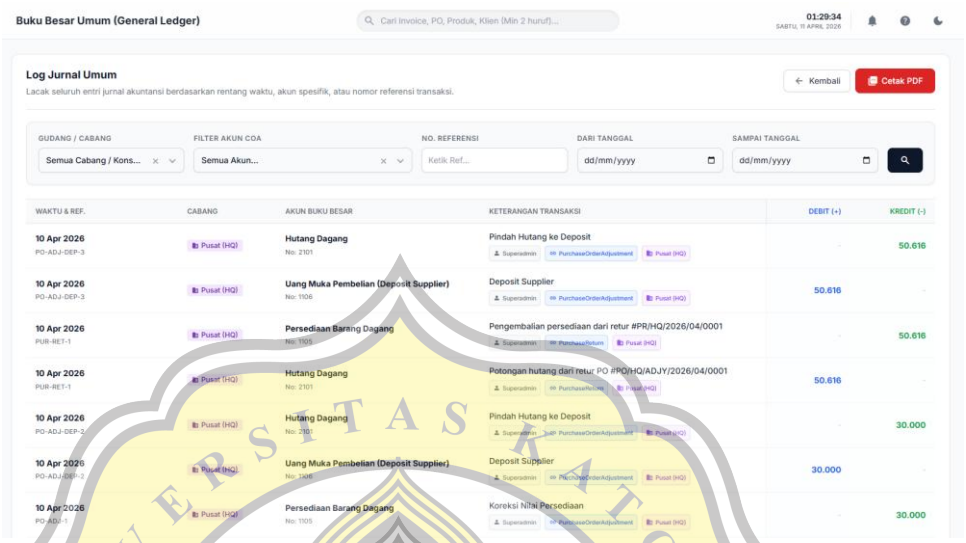
Gambar 4. 149 Kode Program Riwayat Produk

Seluruh hasil kueri gabungan tersebut kemudian dieksekusi, diurutkan berdasarkan tanggal eksekusi sistem (created_at), dan dilempar ke antarmuka pengguna dalam bentuk variabel tunggal $history

4.3.30 Modul Laporan Buku Besar

Modul Buku Besar Umum merupakan pusat pencatatan akuntansi yang melacak seluruh entri jurnal debit dan kredit dari berbagai aktivitas operasional perusahaan. Antarmuka utama modul ini pada Gambar 4.150 dirancang sebagai instrumen audit. Staf akuntansi dapat melakukan pencarian spesifik menggunakan filter rentang tanggal, filter akun Chart of

Account (misalnya hanya melihat arus "Hutang Dagang"), hingga melacak nomor referensi transaksi (*Journal Group ID*) untuk menemukan akar dari suatu entri.



Gambar 4. 150 Halaman Modul Buku Besar

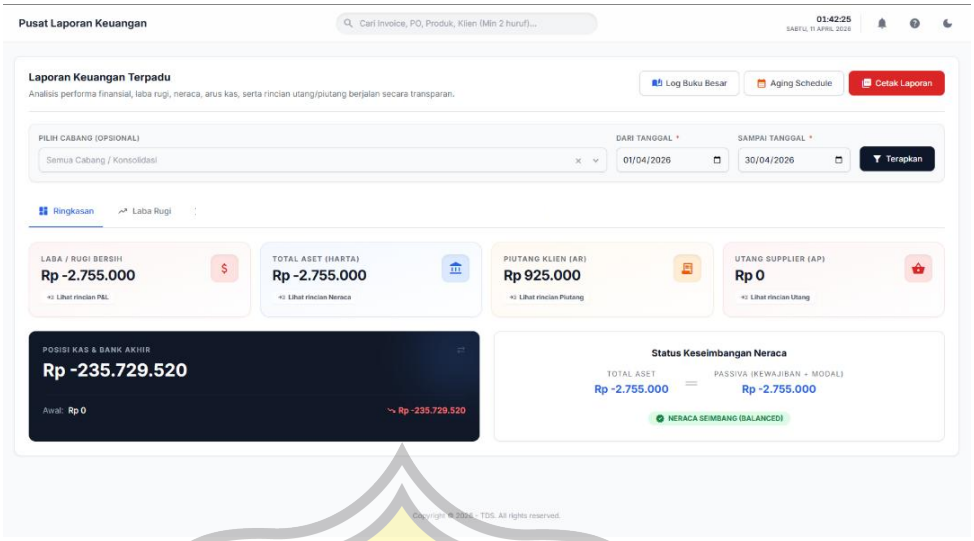
Penarikan data besar-besaran ini ditangani oleh Modul GeneralLedgerController. Pada fungsi index, sistem menggunakan teknik optimasi *Eager Loading* with(['account', 'reference', 'user', 'branch']) saat mengeksekusi kueri ke basis data. Teknik ini sangat krusial untuk mencegah masalah performa *N+1 Query Problem*, di mana sistem hanya perlu melakukan beberapa kueri besar di awal untuk menarik semua data relasional (nama akun, nama pengguna yang mencatat, dan lokasi cabang), alih-alih melakukan kueri berulang-ulang untuk setiap baris jurnal yang ditampilkan.

```php
public function index(Request $request): View
{
    $query = GeneralLedger::with(['account', 'reference', 'user', 'branch'])
                ->orderBy('entry_date', 'desc')
                ->orderBy('ledger_id', 'desc');
    if ($request->filled('start_date')) {
        $query->whereDate('entry_date', '>=', $request->start_date);
    }
    if ($request->filled('end_date')) {
        $query->whereDate('entry_date', '<=', $request->end_date);
    }

    if ($request->filled('account_id')) {
        $query->where('chart_of_account_id', $request->account_id);
    }
    if ($request->filled('journal_group_id')) {
        $query->where('journal_group_id', 'like', '%' . $request->journal_group_id . '%');
    }

    $user = auth()->user();
    if ($user->isGlobal() && $request->filled('branch_id')) {
        $query->where('branch_id', $request->branch_id);
    }

    $journalEntries = $query->paginate(50)->appends($request->query());
    $accounts = ChartOfAccount::where('is_active', true)->orderBy('account_number')->get();

    $branches = [];
    if ($user->isGlobal()) {
        $branches = \App\Models\Branch::orderBy('is_hq', 'desc')->orderBy('name')->get();
    }

    return view('admin.reports.general-ledger.index', compact(
        'journalEntries',
        'accounts',
        'branches'
    ));
}
```

Gambar 4. 151 Kode Program Modul Buku Besar

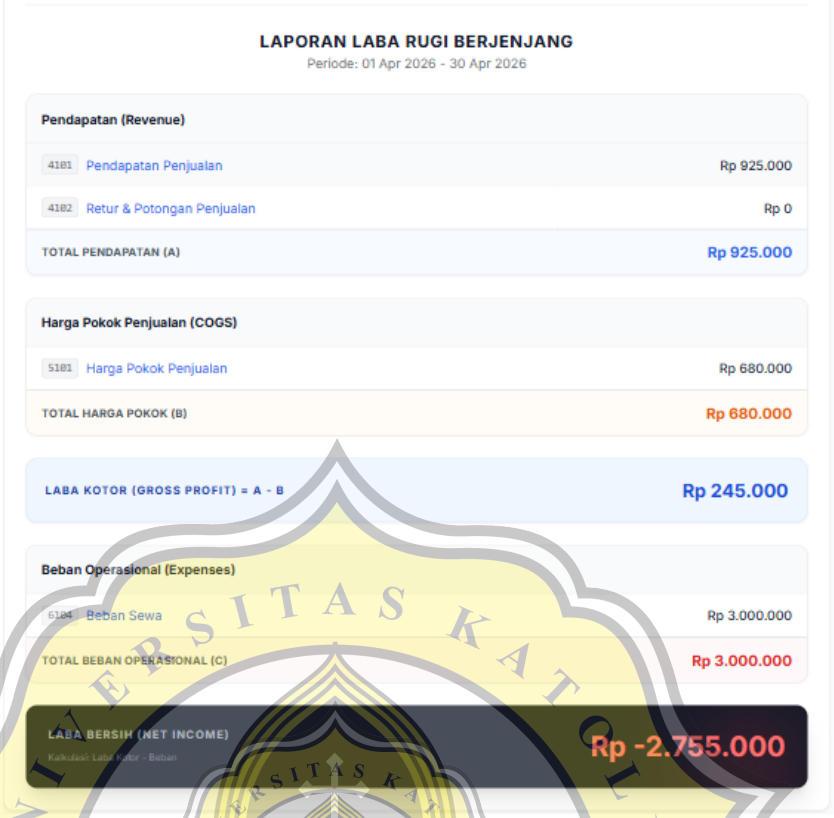
4.3.31 Modul Pusat Laporan Keuangan

Pusat Laporan Keuangan dirancang sebagai dasbor analitik bagi pihak manajemen untuk memantau performa dan kesehatan finansial perusahaan secara terintegrasi. Halaman beranda modul ini menyajikan ringkasan posisi kas aktual, total aset, serta sisa piutang dan utang yang belum terselesaikan, seperti yang terlihat pada Gambar 4.152. Administrator pusat (HQ) dapat memfilter laporan ini untuk melihat kinerja finansial perusahaan secara konsolidasi (keseluruhan cabang) maupun secara individu per cabang operasional.



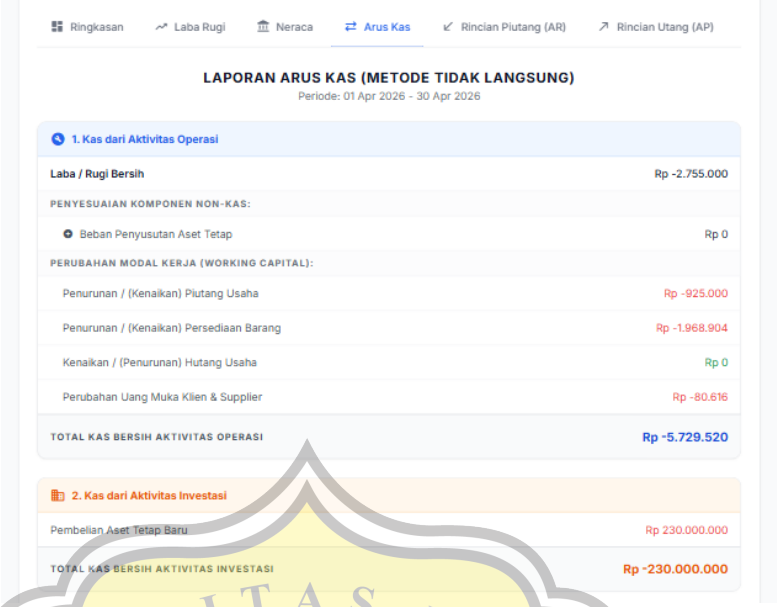
Gambar 4. 152 Halaman Ringkasan Pusat Laporan Keuangan

Modul ini terdiri dari beberapa sub-laporan utama, yang pembentukan datanya dipusatkan pada fungsi getFinancialData di dalam Modul ReportController. Laporan pertama adalah Laporan Laba Rugi (*Income Statement*) yang ditunjukkan pada Gambar 4.153. Untuk membangun laporan ini, sistem menyaring seluruh jurnal (general_ledgers) yang masuk ke dalam tipe akun Pendapatan, HPP, dan Beban, lalu menghitung Laba Bersih berdasarkan selisih agregat Debit dan Kredit dalam rentang periode yang ditentukan.



Gambar 4. 153 Halaman Laporan Laba Rugi

Laporan kedua adalah Laporan Arus Kas (*Cash Flow Statement*) metode tidak langsung, seperti pada Gambar 4.3.129 dan Gambar 4.3.130. Sistem menyusun aliran kas menjadi tiga aktivitas: Operasi, Investasi, dan Pendanaan. Untuk aktivitas Operasi, sistem menghitung komponen *Working Capital* menggunakan *closure function* $getChange. Fungsi ini melacak selisih saldo awal bulan ($previousDateCarbon) dan akhir bulan ($endDateCarbon) pada akun-akun likuid seperti Piutang, Persediaan, dan Hutang. Selisih tersebut kemudian ditambahkan ke Laba Bersih untuk mendapatkan nilai Kas Bersih Aktivitas Operasi.



Gambar 4. 154 Laporan Arus Kas

```
$getChange = function($accountId, $isAsset) use ($previousDateCarbon, $endDateCarbon, $branchId) {
    if (!$accountId) return 0;
    $startBal = GeneralLedger::where('chart_of_account_id', $accountId)
        ->where('entry_date', '<', $previousDateCarbon)
        ->when($branchId, fn($q) => $q->where('branch_id', $branchId))
        ->sum(DB::raw($isAsset ? 'debit - credit' : 'credit - debit'));

    $endBal = GeneralLedger::where('chart_of_account_id', $accountId)
        ->where('entry_date', '<=', $endDateCarbon)
        ->when($branchId, fn($q) => $q->where('branch_id', $branchId))
        ->sum(DB::raw($isAsset ? 'debit - credit' : 'credit - debit'));

    return $isAsset ? ($startBal - $endBal) : ($endBal - $startBal);
};

$cf_change_ar = $getChange($arId, true);
$cf_change_inventory = $getChange($invId, true);
$cf_change_supplier_deposit = $getChange($supplierDepositId, true);
$cf_change_ap = $getChange($apId, false);
$cf_change_client_deposit = $getChange($clientDepositId, false);
```

Gambar 4. 155 Kode Program Laporan Keuangan

4.4 Analisa Data Responden

4.4.1 Profil Responden

Pengujian sistem ini melibatkan 5 responden dari UD. Trois Dinamika Sentosa terdiri dari 1 Pemilik, 1 Admin, 1 Kasir, dan 2 Sales. Seluruh responden sudah terbiasa menggunakan smartphone dan komputer dalam kesehariannya. Sebelumnya, operasional perusahaan sepenuhnya bergantung pada perangkat lunak *(software)* akuntansi (SAP) yang hanya bisa diakses melalui komputer di kantor.

Kendala utama dari sistem lama tersebut adalah antarmuka yang kaku dan tidak mendukung akses mobile. Akibatnya, tim sales di lapangan harus mencatat pesanan dari klien secara manual di buku. Kehadiran sistem ERP baru yang berbasis web ini dinilai sangat tepat karena langsung menjawab kebutuhan mobilitas dan kemudahan akses bagi seluruh peran di perusahaan.

4.4.2 Hasil Wawancara

Pengukuran tingkat penerimaan sistem baru dilakukan melalui wawancara langsung dengan para pengguna akhir. Wawancara difokuskan pada perbandingan antara sistem lama (SAP desktop dan pencatatan manual) dengan sistem yang baru.

Berikut adalah rangkuman hasil wawancara dari perwakilan setiap pengguna:

Tabel 4. 1 Hasil Wawancara Evaluasi Sistem oleh Pengguna UD. Trois Dinamika Sentosa

No.	Pertanyaan	Pemilik	Admin & Kasir	Staff *Sales*
1.	Bagaimana pendapat Anda mengenai tampilan sistem ini dan kemudahan aksesnya melalui smartphone dibandingkan dengan SAP sebelumnya?	Lebih ringkas. Saya bisa memantau transaksi dari HP kapan saja, nggak harus duduk di depan komputer kantor kayak waktu pakai SAP.	Lebih gampang dipahami. Kalau SAP dulu menunya terlalu kompleks, sekarang lebih sederhana namun fungsional.	Lumayan praktis. Dulu di lapangan cuma bisa mencatat manual di nota; sekarang langsung input *draft* tagihan lewat HP.
2.	Seberapa besar dampak otomatisasi (seperti Payment Gateway, Jurnal Otomatis, dan Potong Stok) terhadap pekerjaan Anda?	Bagus, karena stok otomatis terpotong dan jurnal langsung terbentuk. Ini meminimalisir human error.	Memudahkan, apalagi ada payment gateway. Nggak perlu bolak-balik cek mutasi rekening manual buat mastiin klien udah transfer.	Kerjaan jadi lebih cepat. Klien tinggal bayar lewat link *(QRIS/VA)* dan status nota otomatis berubah lunas di sistem.
3.	Apakah fitur pelacakan riwayat (histori produk dan	Sangat transparan. Kalau ada klien yang melakukan retur	Lumayan gampang buat ngurus retur. Riwayatnya tercatat	Ngebantu banget pas ada klien yang mau retur barang. Datanya

	retur) membantu efisiensi kerja?	produknya, jadi pembukuannya nggak berantakan.	semua dan nggak pusing ngitung sisa uang karena sistem udah otomatis.	jelas dan gampang dicari.
4.	Secara keseluruhan, apakah Anda merasa nyaman dan berniat terus menggunakan sistem ini ke depannya?	Nyaman dan mempermudah pengawasan bisnis secara keseluruhan.	Sangat nyaman, bikin kerjaan harian lebih cepat beres. Karena detail informasi muncul lengkap.	Nyaman dipakai. Harapannya *server* sistem ini lancar terus ketika ada banyak orderan.
5.	Bagaimana pendapat Anda mengenai keakuratan data dan laporan pada sistem ini?	Menurut saya, laporannya cukup akurat. Laba rugi dan ringkasan keuangan bisa langsung dilihat secara *real-time* tanpa nunggu akhir bulan.	Sangat ngebantu buat pembukuan. Angkanya otomatis sinkron dengan jurnal.	Sisa hutang sama limit kredit klien selalu diperbarui, jadi kita nggak ragu waktu mau menagih ke toko.
6.	Apakah pembatasan hak akses (*Role & Permission*) membuat pekerjaan Anda terasa lebih terarah dan aman?	Aman, karena saya bisa membatasi apa saja yang boleh dilihat atau diubah sama admin atau sales.	Pas pembagian tugasnya. Fitur batalin nota atau koreksi juga ada lognya, jadi ketahuan siapa yang ngubah.	Lebih simpel karena menu di akun saya cuma fokus pada klien sama pesanan saja. Nggak pusing lihat menu lain.
7.	Berapa lama waktu adaptasi yang Anda butuhkan untuk lancar menggunakan sistem ini dibandingkan dengan SAP?	Butuh sekitar 12 hari buat paham cara baca semua laporan dan tindakan yang dilakukan oleh Admin/*Sales*.	Sekitar 9 hari saja. Alurnya lumayan jelas, berbeda dengan SAP yang lama yang memiliki terlalu banyak langkah.	Kurang lebih 4 hari, karena cara pakainya mirip aplikasi belanja *online* biasa, jadi gampang belajarnya.

Kesimpulan Hasil Wawancara:

Bagi Pemilik, Admin, dan Kasir, implementasi sistem ini memberikan transformasi yang signifikan terhadap tata kelola keuangan dan

pengawasan operasional. Jika sebelumnya pekerjaan terikat secara kaku pada perangkat lunak SAP di komputer desktop kantor, kini pemantauan dan persetujuan dokumen dapat dilakukan secara fleksibel melalui smartphone. Fitur otomatisasi seperti *Payment Gateway*, pemotongan stok, dan pencatatan jurnal akuntansi yang berjalan seketika (*real-time*) terbukti mampu memangkas waktu kerja dan meminimalisir *human error*. Selain itu, modul pelaporan keuangan (Laba Rugi dan Keuangan) yang terintegrasi membuat pembukuan menjadi jauh lebih transparan. Adanya fitur pembatasan hak akses *(Role & Permission)* juga dinilai sangat efektif dalam menjaga keamanan dan integritas data perusahaan. Secara keseluruhan, sistem ini menjadi solusi cerdas yang mempermudah pengawasan bisnis dan pelaporan keuangan harian maupun bulanan.

Bagi Staf Sales, sistem ini berhasil menjadi solusi atas kendala pencatatan pesanan manual di buku yang selama ini membebani operasional di lapangan. Tampilan antarmuka sistem dinilai sangat praktis, modern, dan mudah dipelajari dalam waktu singkat karena menyerupai aplikasi *online* pada umumnya. Proses pembuatan rancangan tagihan (*draft invoice* atau *PO*) kini dapat dieksekusi langsung dari lokasi pelanggan hanya menggunakan *handphone*. Transparansi data secara *real-time* mengenai sisa hutang dan limit kredit pelanggan juga sangat membantu staf sales dalam mengambil keputusan penagihan secara akurat. Ditambah dengan adanya fitur tautan pembayaran instan (QRIS/VA) yang otomatis memperbarui status lunas di sistem, alur koordinasi dengan pihak kasir menjadi jauh lebih ringkas. Secara keseluruhan, Staf Sales merasa sangat nyaman dan terbantu karena sistem baru ini mengeliminasi kekakuan proses lama dan membuat pekerjaan di lapangan menjadi jauh lebih efisien.

BAB V
PENUTUP

5.1 Kesimpulan

Perancangan sistem informasi pembelian dan penjualan dilakukan dengan membangun platform berbasis website yang mensentralisasi seluruh data operasional UD. Trois Dinamika Sentosa. Alur kerja yang sebelumnya kaku pada perangkat lunak lama serta ketergantungan pada buku catatan manual kini digantikan oleh alur digital yang jauh lebih fleksibel. Tim sales lapangan dapat menginput pesanan secara langsung melalui perangkat seluler di lokasi pelanggan, sementara proses pengadaan barang terhubung otomatis dengan pembaruan stok gudang menggunakan perhitungan harga rata-rata yang akurat. Integrasi ini menciptakan sinkronisasi data yang efisien antara tim lapangan, gudang, dan administrasi kantor secara *real-time*.

Sistem informasi keuangan dirancang dengan mengadopsi standar otomatisasi akuntansi yang mencakup manajemen Chart of Accounts (COA) hingga pembentukan Jurnal Umum secara instan. Setiap transaksi dari modul pembelian, penjualan, maupun pengeluaran operasional dipetakan secara otomatis ke dalam buku besar, termasuk penanganan saldo deposit yang tercipta dari kelebihan bayar klien maupun kepada supplier. Perancangan ini menghasilkan pusat laporan keuangan terpadu yang mampu menyajikan data Laba Rugi, Neraca, serta Arus Kas secara otomatis, sehingga memberikan transparansi penuh bagi manajemen dalam memantau kesehatan finansial perusahaan kapan saja tanpa perlu rekapitulasi manual.

Penerapan sistem secara teknis memanfaatkan framework Laravel dan basis data MySQL untuk menciptakan antarmuka yang responsif, aman, dan mudah diakses melalui berbagai perangkat. Implementasi Teknologi Keuangan diwujudkan melalui integrasi *API Payment Gateway* yang mengubah proses penagihan tradisional menjadi pembayaran digital instan melalui QRIS dan Virtual Account. Penggunaan mekanisme *webhook* pada sistem memastikan setiap pembayaran yang masuk terdeteksi secara otomatis, memperbarui status pelunasan nota, dan

menerbitkan jurnal penerimaan kas tanpa campur tangan admin. Penerapan teknologi ini secara nyata mempercepat siklus arus kas perusahaan dan meminimalisir risiko kesalahan manusia dalam rekonsiliasi pembayaran.

5.2 Saran

Berdasarkan hasil penelitian dan implementasi sistem yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan selanjutnya. Pertama, sistem berbasis website ini disarankan untuk dikembangkan lebih lanjut menjadi aplikasi mobile native (berbasis Android dan iOS). Pengembangan ini bertujuan agar operasional tim sales di lapangan menjadi lebih responsif, intuitif, dan tidak lagi bergantung penuh pada akses melalui web browser saja. Bersamaan dengan itu, penambahan fitur *push notification* ke perangkat pengguna juga sangat direkomendasikan. Fitur ini akan mempercepat alur informasi harian, seperti memberikan pengingat jatuh tempo tagihan, notifikasi persetujuan dokumen, hingga konfirmasi masuknya pembayaran secara *real-time* tanpa mengharuskan pengguna membuka aplikasi terlebih dahulu.

Selanjutnya, seiring dengan proyeksi peningkatan volume data transaksi perusahaan di masa mendatang, diperlukan pemeliharaan dan optimasi query basis data, terutama pada modul-modul pelaporan keuangan terpadu. Langkah optimasi tingkat lanjut ini sangat penting dilakukan untuk menjaga kecepatan *loading time* serta mencegah beban *server* yang berlebihan, sehingga performa dan stabilitas sistem tetap terjaga untuk skala penggunaan jangka panjang.