BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Program magang dilaksanakan di LPPM UMN pada divisi *Apps Dev*, yang berfokus pada pengembangan aplikasi dan website. Dalam proyek ini, peran yang dijalankan adalah sebagai *Fullstack Developer*, dengan tugas utama melakukan upgrade terhadap website Koperasi Wiyata Mandala.

Proses pengerjaan proyek menerapkan sistem *Work From Home* (WFH) dengan jadwal kerja tetap dari hari Senin hingga Jumat, pukul 09.00 pagi hingga 17.00 sore. Meskipun dilakukan secara remote, seluruh aktivitas kerja tetap dipantau melalui komunikasi yang terstruktur dan koordinasi yang efektif. Selain itu, dilakukan pertemuan rutin setiap dua hari sekali, baik secara daring maupun luring, untuk membahas perkembangan proyek dan memastikan penyelesaian tugas sesuai dengan rencana yang telah ditetapkan.

Struktur organisasi dalam proyek ini berada di bawah pengawasan langsung *General Manager Community Engagement & Government Relations*, yang bertanggung jawab memberikan arahan serta melakukan evaluasi terhadap perkembangan proyek. Adapun alur koordinasi yang diterapkan dalam proyek ini adalah sebagai berikut:

- *General Manager Community Engagement & Government Relations*: Mengawasi jalannya proyek, memberikan arahan strategis, serta melakukan evaluasi berkala terhadap progres upgrade website.
- Tim *App Developers*: Bertanggung jawab atas seluruh aspek pengembangan aplikasi dan website, termasuk dalam implementasi UI/UX dan fitur, serta pengujian sistem. Peran dalam tim ini difokuskan pada pengembangan tampilan antarmuka website Koperasi Wiyata Mandala, dengan memastikan pengalaman pengguna yang optimal serta integrasi yang sesuai dengan sistem yang telah dirancang.

3.2 Tugas yang Dilakukan

Selama masa kegiatan magang, dilakukan pengembangan sistem informasi berbasis web yang difokuskan pada implementasi fitur perhitungan Sisa Hasil

Usaha (SHU) serta pelaporan keuangan bulanan koperasi. Sistem ini dirancang untuk mencatat dan mengelola seluruh data keuangan koperasi, termasuk pemasukan, pengeluaran, simpanan, pinjaman, dan aktivitas operasional lainnya. Proses pengembangan mencakup perancangan antarmuka pengguna, logika bisnis, serta integrasi antara komponen-komponen sistem agar mampu berjalan secara fungsional dan efisien.

Dalam proses pembangunan sistem, digunakan teknologi basis data relasional untuk mengelola dan menyimpan informasi keuangan secara terstruktur. Perancangan skema data dilakukan dengan mempertimbangkan kebutuhan transaksi keuangan dan pelaporan bulanan koperasi, termasuk penyusunan tabel, pembuatan relasi antar entitas, serta penyusunan kueri untuk perhitungan dan penyajian data secara akurat. Sistem juga dilengkapi dengan mekanisme untuk mendokumentasikan aktivitas keuangan koperasi dalam bentuk laporan bulanan yang mencakup keseluruhan aktivitas keuangan, tidak hanya terbatas pada anggota.

Selain pengembangan fitur utama, kegiatan magang juga mencakup proses pemeliharaan dan perbaikan sistem secara menyeluruh. Tugas ini meliputi identifikasi dan penyelesaian berbagai kesalahan teknis yang ditemukan selama proses pengujian maupun penggunaan sistem, baik pada sisi antarmuka pengguna maupun pengolahan data di sisi *server*. Dengan adanya proses *debugging* dan pengujian ini, sistem dapat berfungsi dengan lebih stabil, akurat, dan mendukung kebutuhan pengelolaan serta pelaporan keuangan koperasi secara optimal.

3.3 Uraian Pelaksanaan Magang

Dalam proses pengembangan sistem informasi koperasi, digunakan sejumlah alat dan teknologi modern untuk mendukung pembangunan sistem secara menyeluruh, mencakup sisi antarmuka pengguna (*frontend*) maupun sisi logika dan pengelolaan data (*backend*). Pada sisi backend, pengembangan dilakukan dengan menggunakan *Node.js* sebagai lingkungan eksekusi server, *Express* sebagai framework untuk pengelolaan permintaan HTTP dan *middleware*, serta *Sequelize* sebagai pustaka *Object-Relational Mapping* (ORM) untuk mempermudah integrasi antara model aplikasi dan basis data. Penyimpanan dan pengolahan data dilakukan melalui *PostgreSQL*, yang merupakan sistem manajemen basis data relasional dengan dukungan struktur tabel yang kompleks.

Pada sisi frontend, antarmuka pengguna dikembangkan menggunakan *React*, yang memungkinkan pembaruan tampilan secara dinamis sesuai dengan

perubahan data. Proses desain antarmuka didukung oleh *Tailwind CSS*, yaitu *framework* CSS berbasis *utility-first* yang mempercepat pembuatan tampilan yang responsif dan konsisten di berbagai perangkat. Seluruh alat tersebut digunakan secara terintegrasi dalam pengembangan sistem, mulai dari perancangan antarmuka hingga pengelolaan data dan logika aplikasi.

Fokus pengembangan diarahkan pada dua modul utama, yaitu sistem perhitungan dan distribusi Sisa Hasil Usaha (SHU) serta laporan keuangan bulanan koperasi. Modul SHU dirancang untuk menghitung pembagian berdasarkan data transaksi keuangan anggota, sedangkan laporan keuangan mencakup seluruh catatan pemasukan dan pengeluaran koperasi dalam periode bulanan. Selain pengembangan fitur tersebut, dilakukan pula proses *debugging* untuk memastikan sistem berjalan dengan stabil, akurat, dan sesuai dengan kebutuhan operasional koperasi.

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Rangkuman Pekerjaan Mingguan

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Memahami alur proyek, mempelajari struktur kode *frontend* dan *backend*, serta *database* yang digunakan.
2	Menerapkan desain UI/UX awal dan melakukan perbaikan tampilan pada *dashboard* admin.
3	Menyempurnakan tampilan admin serta mempelajari dan menguji fitur manajemen akun.
4	Melakukan *debugging* dan pengujian sistem pada fitur *Account Management* serta validasi penyimpanan data.
5	Melakukan perencanaan dan diskusi awal mengenai alur kerja serta kebutuhan sistem SHU.
6	Menerapkan desain UI/UX untuk halaman SHU serta membangun logika awal perhitungan SHU.
7	Mengembangkan model *database*, *controller*, dan *routing* untuk proses perhitungan dan input SHU.
8	Melanjutkan pengembangan dan *debugging* sistem SHU serta memperbaiki penyimpanan dan tampilan data.
	Lanjut pada halaman berikutnya

Tabel 3.1. Rangkuman Pekerjaan Mingguan (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
9	Merancang dan memulai pembuatan tampilan awal untuk sistem laporan keuangan bulanan.
10	Melakukan perombakan desain antarmuka dan penyempurnaan *layout* pada laporan keuangan.
11	Mengembangkan *backend* laporan keuangan, termasuk *model* dan *controller* untuk pengolahan data bulanan.
12	Mengintegrasikan sistem input manual dan membuat fitur riwayat edit data laporan keuangan.
13	Menyempurnakan UI laporan keuangan dan memisahkan tampilan data keuangan anggota dan transaksi manual.
14	Mengimplementasikan fitur ekspor laporan keuangan bulanan ke dalam format PDF.
15	Melakukan *debugging* akhir dan memastikan sinkronisasi serta penyimpanan data laporan keuangan berjalan dengan baik.

Sebelum dilakukan pengembangan lebih lanjut, sistem informasi koperasi telah memiliki versi awal yang masih sangat terbatas, khususnya dalam aspek pengelolaan keuangan. Pada tahap tersebut, dua modul utama, yaitu modul perhitungan dan distribusi Sisa Hasil Usaha (SHU) serta modul laporan keuangan bulanan, belum tersedia dalam bentuk yang fungsional. Modul SHU belum diimplementasikan sama sekali, baik dari sisi antarmuka pengguna, logika perhitungan, maupun pengelolaan data. Sementara itu, modul laporan keuangan hanya hadir dalam bentuk kerangka visual statis yang belum terhubung dengan sumber data atau sistem *backend*, sehingga tidak dapat menampilkan informasi keuangan koperasi secara dinamis dan aktual.

Berdasarkan hasil evaluasi terhadap versi awal sistem, tidak ditemukan fitur pendukung utama seperti input persentase pembagian SHU, perhitungan otomatis berdasarkan data transaksi anggota, ataupun distribusi hasil. Pada sisi laporan keuangan, belum tersedia fungsi pencatatan transaksi manual, penyimpanan historis, maupun mekanisme ekspor data. Seluruh elemen tersebut masih belum terintegrasi dengan baik dalam sistem, sehingga tidak mendukung kebutuhan pelaporan dan pengelolaan keuangan koperasi secara utuh.

Selama kegiatan magang, seluruh pekerjaan yang dilakukan difokuskan pada pengembangan dua modul utama tersebut. Modul SHU dikembangkan dari awal, mencakup perancangan antarmuka, pembuatan model dan controller backend, serta penerapan logika perhitungan berdasarkan data simpanan dan pinjaman anggota. Ditambahkan pula fitur input manual untuk persentase pembagian SHU, integrasi dengan data pengajuan anggota, serta tampilan hasil distribusi secara dinamis. Untuk modul laporan keuangan, pengembangan meliputi penyusunan ulang tampilan, pengelolaan data pemasukan dan pengeluaran secara manual, riwayat perubahan data (*history edit*), pemisahan tampilan antara data anggota dan transaksi manual, serta fitur ekspor laporan bulanan ke dalam format PDF.

Seluruh penambahan dan peningkatan fitur pada kedua modul dilakukan secara terstruktur dan bertahap, mencakup integrasi antarmuka pengguna dengan backend, pengujian fungsi, dan proses debugging. Hasil pengembangan diharapkan dapat memberikan dukungan penuh terhadap kebutuhan koperasi dalam melakukan perhitungan SHU dan menyusun laporan keuangan secara akurat, efisien, dan sesuai dengan sistem yang terkomputerisasi. Dokumentasi visual sistem sebelum pengembangan dilampirkan sebagai acuan evaluasi dan pembanding terhadap versi akhir yang telah dikembangkan secara menyeluruh.

Proses pengembangan sistem informasi koperasi dijabarkan secara terstruktur berdasarkan masing-masing modul yang dikerjakan. Setiap modul dijelaskan secara menyeluruh, meliputi penjabaran *sitemap*, *flowchart*, perancangan *database*, potongan kode (*code snippet*), serta dokumentasi tampilan antarmuka hasil pengembangan. Penyusunan ini bertujuan untuk memberikan gambaran menyeluruh mengenai proses perancangan dan implementasi teknis dari tiap fitur yang dikembangkan dalam sistem.

Modul utama yang menjadi fokus pengembangan adalah modul laporan keuangan bulanan, yang didesain untuk mencatat, mengelola, dan menampilkan data keuangan koperasi secara periodik. Selain itu, dikembangkan pula modul perhitungan dan distribusi SHU, yang mendukung proses pembagian hasil usaha berdasarkan data keuangan tahunan anggota. Di luar dua modul utama tersebut, terdapat pula sejumlah kontribusi tambahan dalam pengembangan sistem, seperti perbaikan fungsi, penguatan validasi, serta penyempurnaan antarmuka pengguna, yang turut dijelaskan dalam bagian tersendiri.

Dalam setiap pembahasan modul, *sitemap* digunakan untuk menggambarkan navigasi antar halaman, sementara *flowchart* menjelaskan alur proses sistem, mulai dari input data hingga penyimpanan dan keluaran.

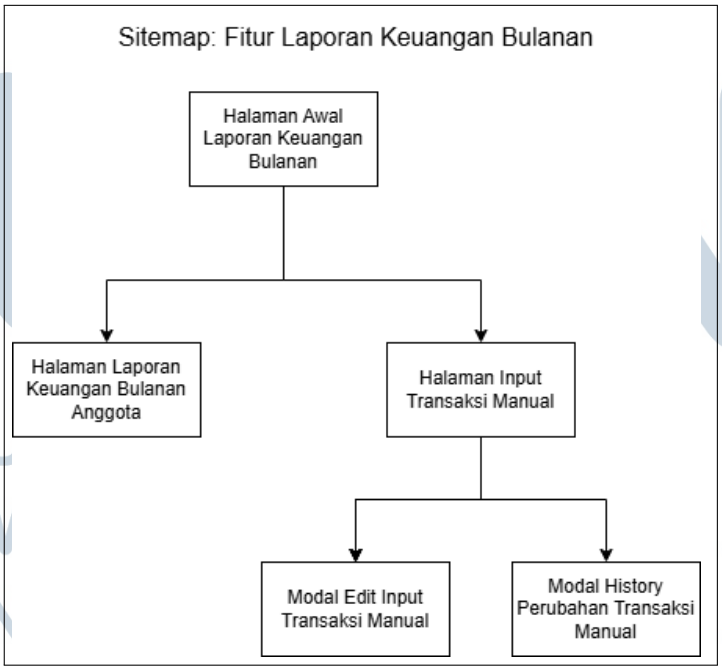
Perancangan *database* disusun untuk memenuhi kebutuhan relasi dan integrasi data, sedangkan potongan kode ditampilkan untuk memperjelas logika fungsional yang dibangun. Tangkapan layar tampilan akhir juga disertakan guna memberikan visualisasi hasil pengembangan secara konkret. Pendekatan ini memastikan seluruh proses dokumentasi sistem bersifat sistematis, transparan, dan terukur.

3.3.1 Modul Laporan Keuangan

A Sitemap

Setelah pengguna dengan peran Pengurus memilih opsi "Laporan Keuangan" dari *dropdown* di *navbar*, sistem akan menampilkan halaman awal laporan keuangan bulanan. Dari halaman ini, pengguna dapat mengakses dua bagian utama, yaitu laporan keuangan bulanan anggota dan input transaksi manual.

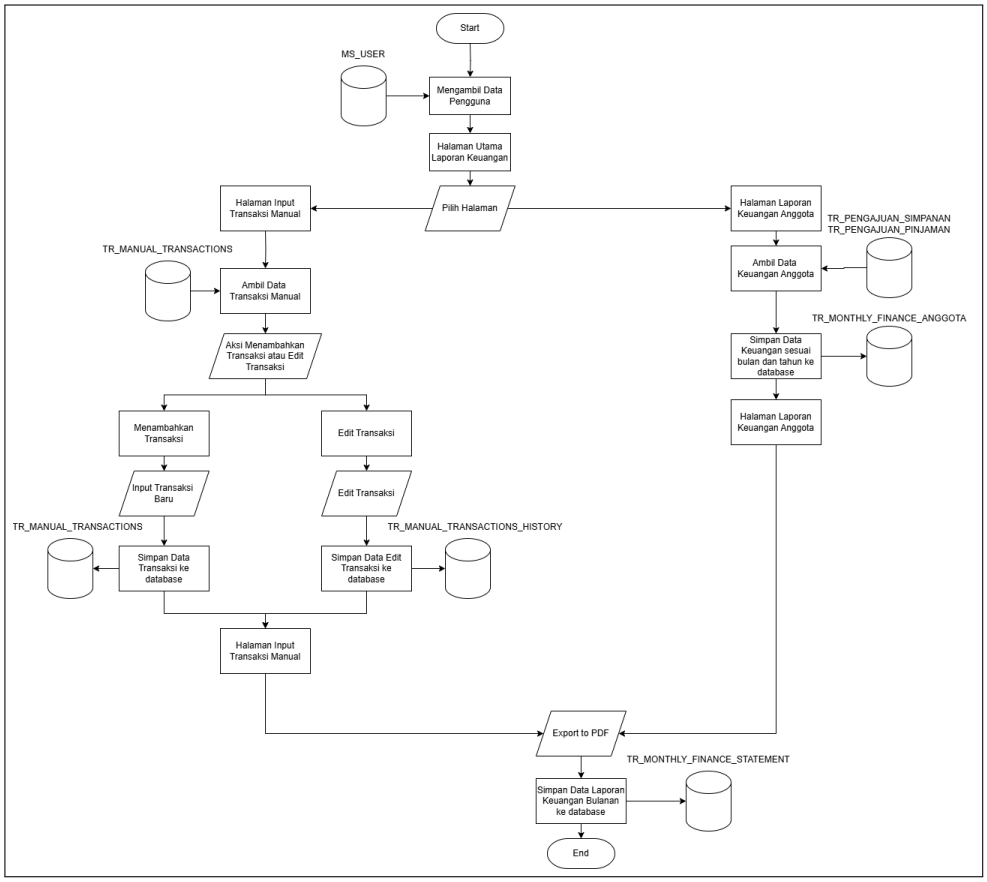
Jika pengguna masuk ke halaman input transaksi manual, tersedia dua interaksi lanjutan, yaitu membuka modal edit transaksi manual untuk memperbarui data, serta modal riwayat perubahan untuk melihat catatan histori edit transaksi. Seluruh alur ini membentuk struktur navigasi sesuai dengan kebutuhan pengelolaan laporan keuangan bulanan oleh pengguna "Pengurus" dalam sistem.



Gambar 3.1. Sitemap Modul Laporan Keuangan Bulanan

B Flowchart

Proses modul laporan keuangan hanya dapat diakses oleh pengguna dengan peran Pengurus, melalui menu "Laporan Keuangan" pada *dropdown* di *navbar*. Setelah menu dipilih, pengguna diarahkan ke halaman utama yang terdiri dari dua bagian, yaitu Laporan Anggota dan Input Transaksi Manual. Pada bagian Laporan Anggota, sistem mengambil data dari tabel *MS_USER*, *TR_PENGAJUAN_SIMPANAN*, dan *TR_PENGAJUAN_PINJAMAN* untuk menghitung arus kas masing-masing anggota. Saat halaman ini diakses, hasil perhitungan otomatis disimpan ke dalam tabel *TR_MONTHLY_FINANCE_ANGGOTA* sesuai dengan bulan dan tahun yang sedang difilter. Pada bagian Input Transaksi Manual, pengguna mencatat transaksi pemasukan dan pengeluaran yang akan disimpan ke dalam tabel *TR_MANUAL_TRANSACTIONS*. Jika transaksi tersebut diedit, riwayat perubahan akan dicatat ke dalam tabel *TR_MANUAL_TRANSACTIONS_HISTORY*. Saat pengguna melakukan Export ke PDF, data dari kedua bagian laporan akan disimpan ke dalam tabel *TR_MONTHLY_FINANCE_STATEMENT* sebagai dokumentasi laporan keuangan gabungan untuk periode yang dipilih.



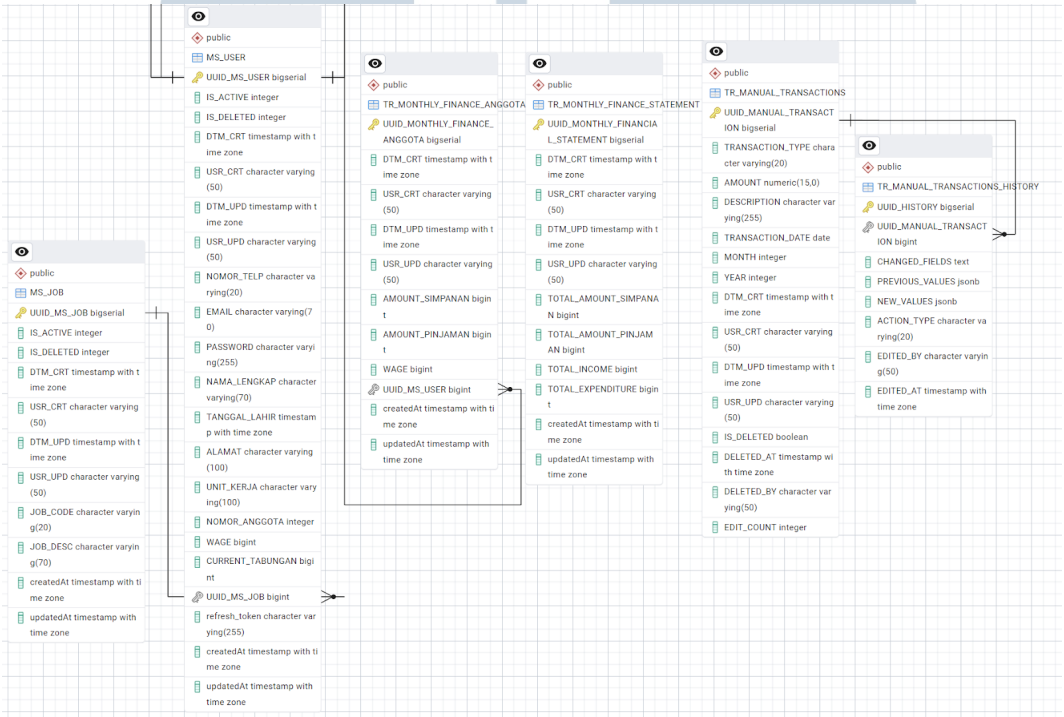
Gambar 3.2. Flowchart Modul Laporan Keuangan Bulanan

C Database

C.1 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) pada sistem ini menggambarkan struktur relasi antar tabel yang membentuk dasar dari modul laporan keuangan dan fitur pendukungnya. Tabel pusat *MS_USER* memiliki relasi langsung ke *MS_JOB* melalui *foreign key UUID_MS_JOB* untuk menetapkan peran atau hak akses pengguna. Tabel *MS_USER* juga berelasi dengan *TR_MONTHLY_FINANCE_ANGGOTA*, *TR_MANUAL_TRANSACTIONS*, dan *MS_SHU_ANGGOTA*, yang masing-masing menggunakan *foreign key UUID_MS_USER* untuk mengaitkan catatan transaksi dan data keuangan dengan identitas pengguna. Tabel *TR_MANUAL_TRANSACTIONS* memiliki relasi *one-to-many* dengan *TR_MANUAL_TRANSACTIONS_HISTORY* melalui *foreign key UUID_MANUAL_TRANSACTION*, yang digunakan

untuk menyimpan riwayat perubahan setiap transaksi manual. Sementara itu, tabel *TR_MONTHLY_FINANCE_STATEMENT* tidak memiliki relasi eksplisit dalam skema ERD, namun digunakan untuk menyimpan hasil agregasi data laporan keuangan bulanan dari berbagai tabel lain dalam sistem. Beberapa tabel seperti *TR_MANUAL_TRANSACTIONS_HISTORY* dan *TR_MONTHLY_FINANCE_STATEMENT* memang tidak terhubung langsung secara struktural melalui *foreign key*, namun tetap berperan penting dalam proses pengelolaan data di tingkat aplikasi.



Gambar 3.3. ERD Modul Laporan Keuangan

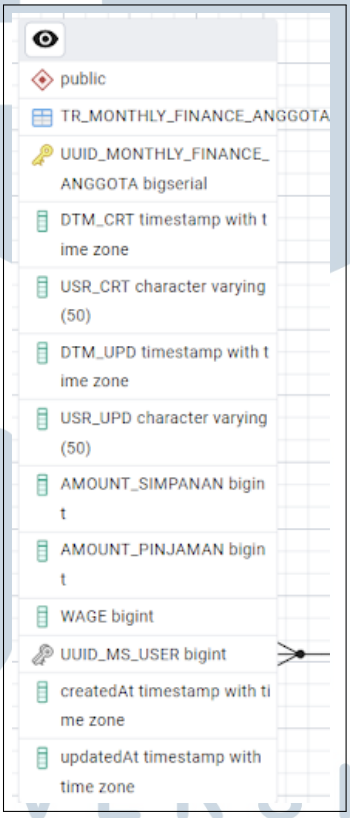
C.2 Model TR_MONTHLY_FINANCE_ANGGOTA

Tabel *TR_MONTHLY_FINANCE_ANGGOTA* digunakan untuk mencatat data keuangan bulanan masing-masing anggota koperasi, yang diperoleh berdasarkan hasil perhitungan otomatis dari sistem. Setiap entri dalam tabel ini dikaitkan dengan pengguna melalui *UUID_MS_USER* sebagai *foreign key*, yang menghubungkan informasi keuangan dengan data identitas pengguna pada tabel *MS_USER*.

Tabel ini menyimpan nilai-nilai penting yang mencerminkan kondisi finansial anggota dalam satu bulan tertentu, seperti jumlah simpanan

(*AMOUNT_SIMPANAN*), jumlah pinjaman (*AMOUNT_PINJAMAN*), dan upah atau pendapatan (*WAGE*). Kolom-kolom tersebut disimpan dalam tipe data numerik untuk memastikan akurasi perhitungan dan agregasi laporan keuangan.

Sebagai bagian dari sistem pencatatan transaksional, tabel ini juga dilengkapi dengan atribut audit seperti *DTM_CRT, USR_CRT, DTM_UPD*, dan *USR_UPD*, serta *createdAt* dan *updatedAt* yang digunakan untuk mencatat waktu penciptaan dan pembaruan data secara otomatis. Tabel *TR_MONTHLY_FINANCE_ANGGOTA* merupakan komponen penting dalam proses pembuatan laporan keuangan bulanan, khususnya untuk mendukung fitur filter berdasarkan bulan dan tahun, serta sebagai dasar dalam fungsi ekspor data laporan ke dalam *format PDF*.



Gambar 3.4. Model Tabel *TR_MONTHLY_FINANCE_ANGGOTA*

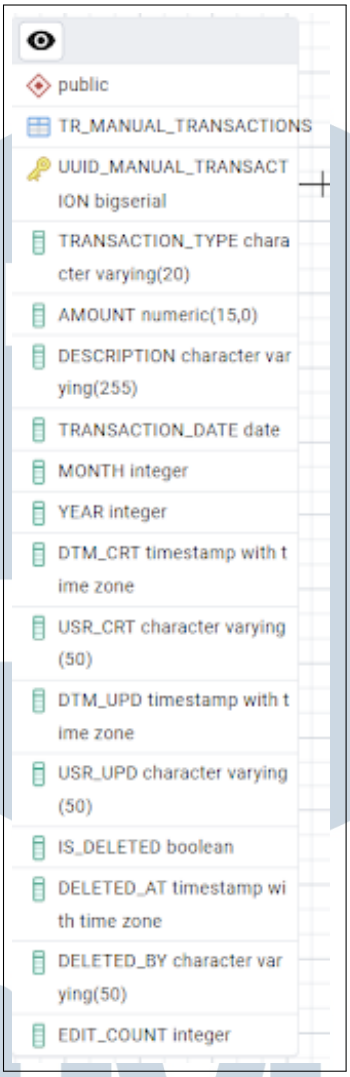
C.3 Model TR_MANUAL_TRANSACTIONS

Tabel *TR_MANUAL_TRANSACTIONS* berfungsi sebagai tempat pencatatan transaksi keuangan manual yang dilakukan di luar sistem otomatis simpan pinjam, seperti pemasukan atau pengeluaran tambahan koperasi. Setiap entri diidentifikasi

secara unik melalui kolom *UUID_MANUAL_TRANSACTION* sebagai *primary key*. Informasi utama yang disimpan meliputi jenis transaksi (*TRANSACTION_TYPE*), nilai transaksi (*AMOUNT*), deskripsi transaksi (*DESCRIPTION*), serta tanggal pelaksanaan transaksi melalui *TRANSACTION_DATE*, dengan pelengkap berupa *MONTH* dan *YEAR* untuk pengelompokan laporan berdasarkan periode.

Tabel ini juga dilengkapi dengan atribut audit yang mencatat waktu serta identitas pengguna yang membuat dan memperbarui data. Kolom *USR_CRT, USR_UPD,* dan *DELETED_BY* masing-masing berisi nama pengguna yang diambil dari tabel *MS_USER* untuk menunjukkan siapa yang membuat, memperbarui, atau menghapus data transaksi. Kolom *DTM_CRT, DTM_UPD*, dan *DELETED_AT* menyimpan informasi tanggal dan waktu yang mencerminkan waktu kejadian dari setiap perubahan yang terjadi, dan seluruh kolom timestamp ini secara konsisten menggunakan nilai waktu aktual saat aksi dilakukan.

Sebagai bagian dari kontrol data, tabel ini juga memiliki kolom *IS_DELETED* untuk menandai penghapusan logis, serta *EDIT_COUNT* yang mencatat jumlah perubahan yang telah dilakukan terhadap transaksi tersebut. Dengan struktur ini, *TR_MANUAL_TRANSACTIONS* menyediakan pencatatan transaksi manual yang transparan dan terverifikasi, sekaligus mendukung integritas laporan keuangan koperasi secara keseluruhan.



Gambar 3.5. Model Tabel *TR_MANUAL_TRANSACTIONS*

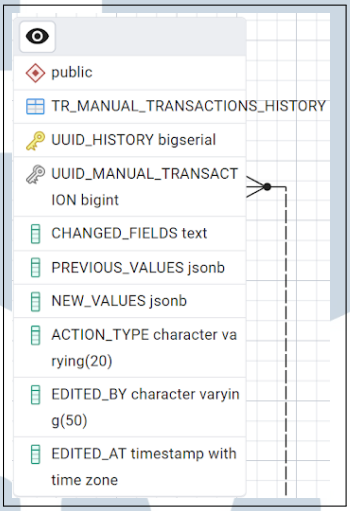
C.4 Model TR_MANUAL_TRANSACTIONS_HISTORY

Tabel *TR_MANUAL_TRANSACTIONS_HISTORY* berfungsi sebagai log histori untuk mencatat setiap perubahan yang terjadi pada data transaksi manual dalam tabel *TR_MANUAL_TRANSACTIONS*. Setiap entri diidentifikasi secara unik melalui *UUID_HISTORY*, dan dikaitkan dengan transaksi asal melalui *foreign key UUID_MANUAL_TRANSACTION*, sehingga setiap riwayat dapat dilacak secara tepat berdasarkan transaksi yang dimodifikasi.

Informasi yang dicatat mencakup kolom *CHANGED_FIELDS*, yang menyimpan daftar atribut yang mengalami perubahan, serta *PREVIOUS_VALUES* dan *NEW_VALUES* dalam format *JSONB* untuk menunjukkan nilai sebelum dan

sesudah perubahan. Jenis tindakan dicatat melalui kolom *ACTION_TYPE*, yang dapat mencerminkan tipe modifikasi seperti *update* atau *soft delete*.

Selain data perubahan, tabel ini juga mencatat siapa yang melakukan pengeditan melalui kolom *EDITED_BY*, yang nilainya merupakan nama pengguna yang diambil dari tabel *MS_USER*. Waktu terjadinya perubahan dicatat secara otomatis dalam kolom *EDITED_AT*, yang menggunakan *timestamp* sesuai dengan saat aksi dilakukan. Dengan struktur ini, *TR_MANUAL_TRANSACTIONS_HISTORY* menyediakan jejak audit yang rinci dan akurat atas seluruh aktivitas pengubahan data transaksi manual, serta memperkuat aspek transparansi dan akuntabilitas dalam sistem keuangan koperasi.



Gambar 3.6. Model Tabel *TR_MANUAL_TRANSACTIONS_HISTORY*

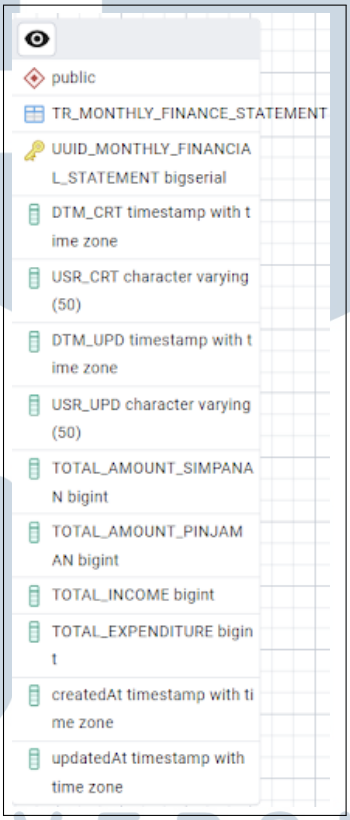
C.5 Model TR_MONTHLY_FINANCE_STATEMENT

Tabel *TR_MONTHLY_FINANCE_STATEMENT* digunakan sebagai media pencatatan hasil akhir dari laporan keuangan bulanan koperasi secara agregat. Setiap entri dalam tabel ini menyimpan rekapitulasi data keuangan untuk satu periode tertentu, yang mencakup total keseluruhan dari simpanan anggota, pinjaman anggota, pemasukan manual, dan pengeluaran manual selama bulan berjalan.

Informasi utama yang dicatat meliputi *TOTAL_AMOUNT_SIMPANAN, TOTAL_AMOUNT_PINJAMAN, TOTAL_INCOME,* dan *TOTAL_EXPENDITURE*, yang masing-masing mewakili jumlah kumulatif berdasarkan hasil perhitungan dari tabel-tabel pendukung seperti *TR_MONTHLY_FINANCE_ANGGOTA* dan

TR_MANUAL_TRANSACTIONS. Data ini secara otomatis dihasilkan dan dicatat ke dalam sistem saat pengguna melakukan ekspor laporan keuangan dalam *format PDF.*

Selain data finansial, tabel ini juga menyimpan informasi terkait pencatatan aktivitas, seperti *DTM_CRT, USR_CRT, DTM_UPD*, dan *USR_UPD*. Nilai *USR_CRT* secara khusus diisi dengan nama pengguna yang melakukan proses ekspor, yang diambil dari tabel *MS_USER* melalui referensi terhadap pengguna yang sedang login. Tabel ini berperan sebagai arsip historis dari laporan keuangan bulanan koperasi yang telah difinalisasi, sehingga dapat dijadikan acuan administratif maupun dokumen pelaporan resmi.



Gambar 3.7. Model Tabel *TR_MONTHLY_FINANCE_STATEMENT*

D Hasil Development

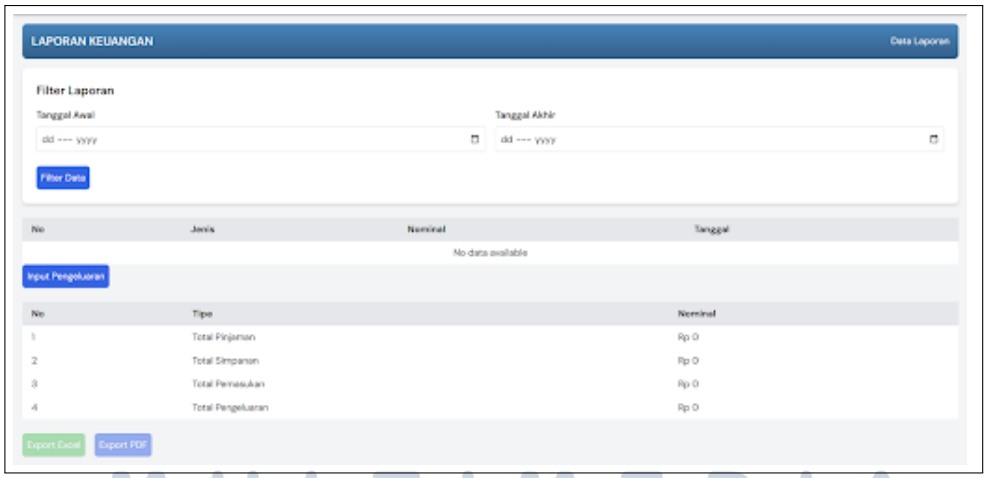
Modul laporan keuangan bulanan pada sistem informasi koperasi berhasil dikembangkan sesuai dengan kebutuhan fungsional pengguna dan perancangan sistem yang telah ditetapkan sebelumnya. Pengembangan modul ini ditujukan untuk mempermudah proses pencatatan, pengelolaan, serta penyajian data

keuangan koperasi secara sistematis dan terintegrasi, khususnya bagi pengguna dengan peran Pengurus.

Melalui implementasi berdasarkan struktur *sitemap*, *flowchart*, dan relasi antar tabel pada basis data, sistem kini mendukung dua komponen utama dalam pengelolaan laporan keuangan, yaitu laporan keuangan anggota dan input transaksi manual. Kedua bagian tersebut dapat diakses dalam satu halaman utama yang dilengkapi dengan fitur filter berdasarkan bulan dan tahun, serta kemampuan untuk melakukan ekspor laporan dalam *format PDF*.

Sebelum proses pengembangan dilakukan, sistem telah memiliki tampilan awal yang masih terbatas baik dari sisi desain maupun fungsionalitas. Fitur-fitur penting seperti pencatatan transaksi manual, pengelompokan laporan per periode, ekspor laporan, serta pencatatan histori edit transaksi belum tersedia atau belum berfungsi secara optimal. Tampilan halaman yang ada bersifat statis dan belum terhubung dengan logika pemrosesan data maupun penyimpanan aktual ke basis data.

Sebagai gambaran awal, Gambar 3.8 berikut menunjukkan tampilan halaman laporan keuangan sebelum dilakukan proses pengembangan. Visual ini menjadi dasar evaluasi untuk membangun sistem yang lebih fungsional, interaktif, dan sesuai dengan kebutuhan pengguna koperasi secara nyata.



Gambar 3.8. Halaman Laporan Keuangan sebelum Pengembangan

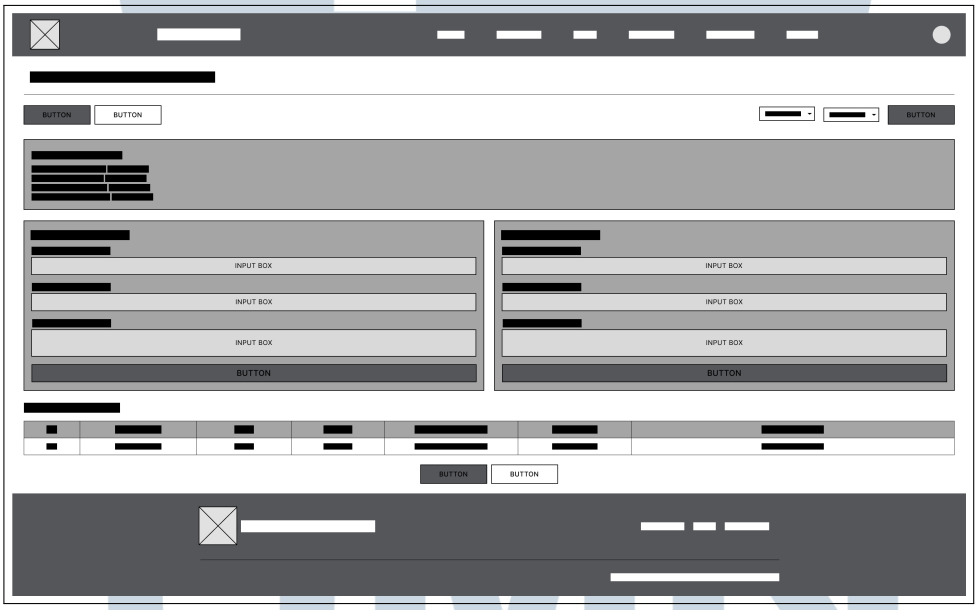
D.1 Tampilan Halaman Transaksi Manual

Halaman ini ditampilkan ketika pengguna memilih tab “Transaksi Manual” dalam modul Laporan Keuangan. Halaman ini dirancang untuk mencatat transaksi

keuangan yang tidak tercakup dalam proses simpan pinjam otomatis, seperti pemasukan dan pengeluaran manual.

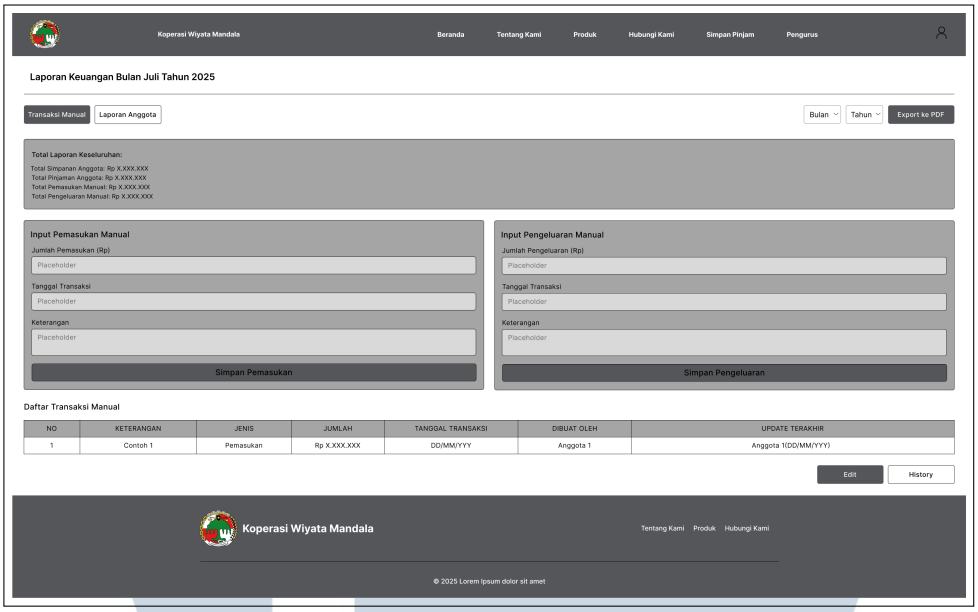
Sebelum implementasi halaman ini dilakukan, disusun terlebih dahulu *wireframe* dalam dua tahap fidelitas: *low-fidelity* dan *medium-fidelity*. *Wireframe* ini berfungsi sebagai acuan awal untuk memastikan *layout*, elemen, dan alur kerja halaman telah sesuai dengan kebutuhan sebelum masuk ke tahap pengembangan UI final.

Gambar 3.9 menunjukkan struktur dasar dari halaman, seperti posisi tab navigasi, formulir pemasukan dan pengeluaran, serta tabel daftar transaksi. Elemen desain masih bersifat blok dasar tanpa detail visual, difokuskan pada penyusunan tata letak dan alur informasi.



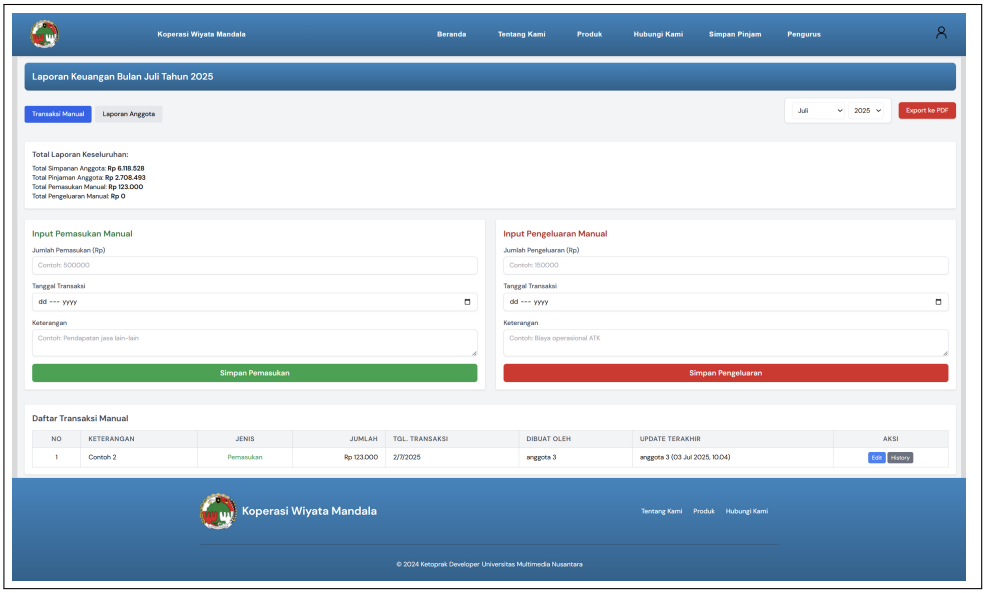
Gambar 3.9. *Wireframe Low-Fidelity* Halaman Transaksi Manual

Gambar 3.10 memperlihatkan versi yang lebih mendetail dari tampilan sebelumnya, dengan penambahan teks label, tombol, dan struktur antarmuka yang menyerupai UI akhir. *Medium-fidelity wireframe* ini digunakan untuk validasi desain sebelum implementasi teknis dimulai.



Gambar 3.10. *Wireframe Medium-Fidelity* Halaman Transaksi Manual

Gambar 3.11 berikut menunjukkan tampilan halaman transaksi manual yang telah dikembangkan untuk mendukung pencatatan keuangan koperasi secara fleksibel dan transparan.



Gambar 3.11. Halaman Transaksi Manual setelah Pengembangan

Tampilan dibagi menjadi dua bagian input: form Input Pemasukan Manual di sebelah kiri dan Input Pengeluaran Manual di sebelah kanan. Masing-masing form memuat kolom untuk jumlah transaksi, tanggal pelaksanaan, serta keterangan

transaksi. Setelah form diisi, pengguna dapat menyimpan data dengan menekan tombol "Simpan Pemasukan" atau "Simpan Pengeluaran", yang akan menyimpan entri ke dalam tabel *TR_MANUAL_TRANSACTIONS*.

Di bagian bawah, terdapat tabel Daftar Transaksi Manual yang menampilkan seluruh transaksi manual yang telah tercatat dalam sistem. Tabel ini memuat informasi penting seperti keterangan transaksi, jenis (pemasukan atau pengeluaran), jumlah nominal, tanggal transaksi, serta identitas pengguna yang membuat dan memperbarui data. Seluruh data yang ditampilkan pada tabel ini merupakan hasil input pengguna melalui form transaksi manual, dan diperbarui secara dinamis berdasarkan aktivitas pencatatan dan pengeditan yang dilakukan.

Pada kolom aksi, tersedia dua tombol interaktif, yaitu *Edit* dan *History*. Tombol *Edit* digunakan untuk membuka modal pop-up yang memungkinkan pengguna memperbarui informasi transaksi secara langsung, termasuk perubahan pada jumlah, tanggal, jenis, dan keterangan. Sementara itu, tombol *History* digunakan untuk menampilkan riwayat perubahan yang pernah terjadi pada transaksi tersebut, dengan data yang diambil dari tabel *TR_MANUAL_TRANSACTIONS_HISTORY*. Tombol *History* hanya akan muncul apabila transaksi telah mengalami proses pengeditan sebelumnya, sehingga sistem hanya menampilkan log perubahan pada data yang relevan.

Sama seperti pada halaman Laporan Anggota, di bagian kanan atas halaman ini juga tersedia tombol "Export ke PDF", yang digunakan untuk menghasilkan dokumen laporan keuangan lengkap berdasarkan data yang sedang ditampilkan. Keberadaan tombol ekspor di posisi yang konsisten pada kedua halaman mempermudah pengguna dalam melakukan dokumentasi laporan keuangan bulanan.

D.2 Tampilan Modal Edit Transaksi Manual

Modal ini ditampilkan ketika pengguna menekan tombol *Edit* pada salah satu entri transaksi manual dalam tabel daftar transaksi. Fitur ini memberikan fleksibilitas bagi pengguna, khususnya dengan peran *Pengurus*, untuk memperbarui atau mengoreksi informasi transaksi yang telah tersimpan sebelumnya dalam sistem.

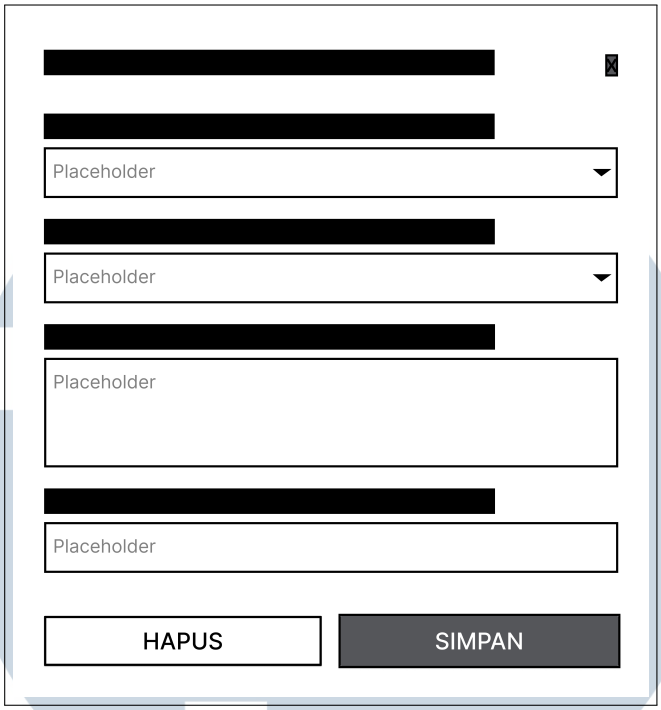
Secara desain, modal ini dibangun untuk memprioritaskan kejelasan dan efisiensi penggunaan. Komponen *form* yang tersedia mencakup empat bagian utama yang disusun secara vertikal dan mudah diakses. Pertama, terdapat *dropdown*

Jenis Transaksi yang memungkinkan pengguna memilih antara dua jenis transaksi, yaitu *Pemasukan* atau *Pengeluaran*. Kedua, terdapat *input field* untuk mengisi Jumlah (Rp), yaitu nilai transaksi dalam satuan mata uang rupiah. Ketiga, tersedia kolom Keterangan dalam bentuk area teks terbuka yang dapat digunakan untuk mencatat detail atau penjelasan tambahan terkait transaksi. Terakhir, pengguna juga dapat menentukan waktu pelaksanaan transaksi melalui komponen *input* Tanggal Transaksi yang dilengkapi format kalender.

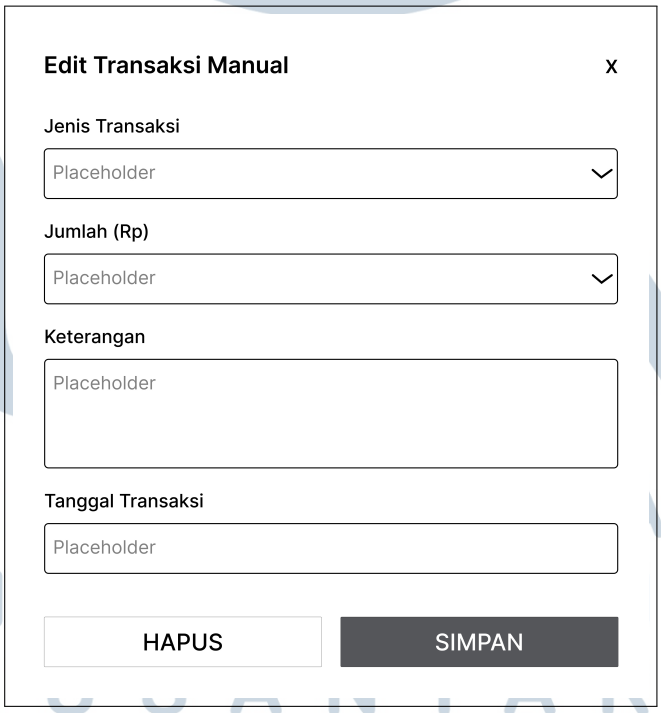
Pada bagian bawah modal, disediakan dua tombol aksi dengan fungsionalitas yang berbeda. Tombol "Hapus" menggunakan gaya *outline* dan digunakan untuk menghapus transaksi secara *soft delete* tanpa menghilangkan data secara permanen dari sistem. Sementara itu, tombol "Simpan" berfungsi sebagai tombol utama (primer) untuk menyimpan seluruh perubahan data yang dilakukan pengguna. Setiap perubahan yang dikonfirmasi melalui tombol ini akan dicatat secara otomatis ke dalam tabel *TR_MANUAL_TRANSACTIONS_HISTORY*, sebagai bagian dari sistem pencatatan riwayat perubahan transaksi yang mendukung aspek akuntabilitas dan audit data.

Untuk mendukung proses desain dan pengembangan antarmuka ini, dua jenis *wireframe* telah dibuat, yaitu *low-fidelity* dan *medium-fidelity*. Gambar 3.12 dan Gambar 3.13 berikut menampilkan proses evolusi desain dari sketsa awal ke tampilan prototipe yang lebih mendekati hasil akhir.

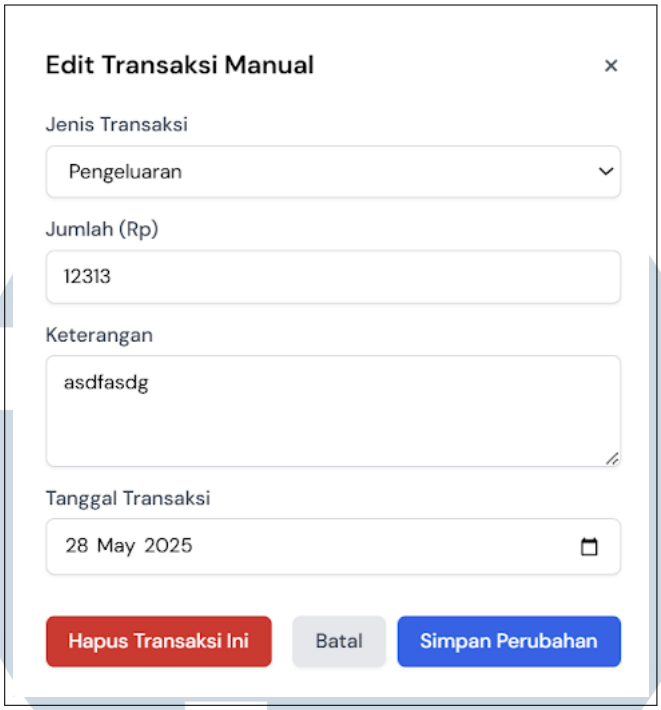
Melalui tahapan ini, struktur dan elemen-elemen penting dari modal berhasil divalidasi dan disesuaikan dengan kebutuhan pengguna, sebelum akhirnya diimplementasikan ke dalam sistem secara fungsional sebagaimana ditampilkan pada Gambar 3.14.



Gambar 3.12. *Wireframe Low-Fidelity* Modal Edit Transaksi Manual



Gambar 3.13. *Wireframe Medium-Fidelity* Modal Edit Transaksi Manual



Gambar 3.14. Tampilan Final Modal Edit Transaksi Manual

D.3 Tampilan Modal History Edit Transaksi Manual

Modal *History Edit* ditampilkan ketika pengguna menekan tombol *History* pada salah satu entri transaksi manual dalam tabel daftar transaksi. Modal ini digunakan untuk menampilkan riwayat perubahan (*log*) yang pernah dilakukan terhadap satu entri transaksi tertentu, guna mendukung transparansi dan akuntabilitas dalam pencatatan data keuangan koperasi.

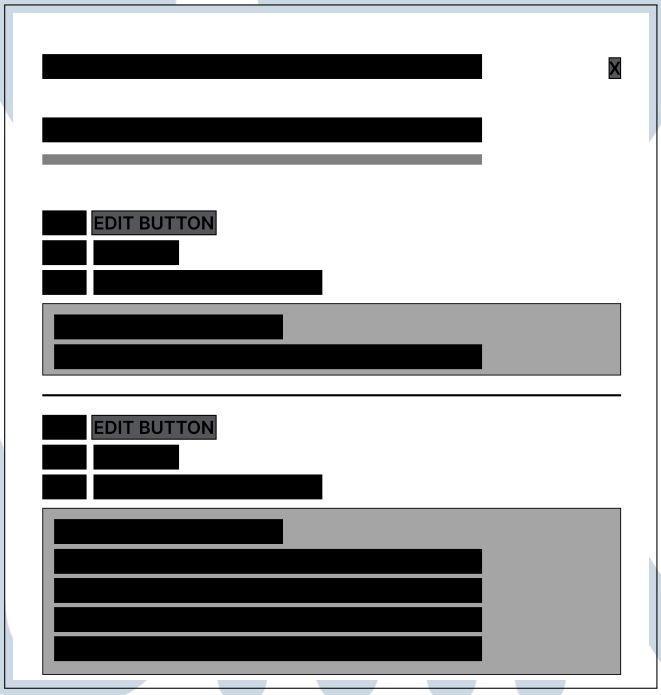
Bagian atas modal menampilkan informasi identitas pencatatan awal, berupa nama pengguna yang pertama kali membuat data, serta tanggal pembuatan data tersebut. Di bawahnya, sistem menampilkan catatan histori perubahan secara kronologis, dimulai dari log yang paling baru. Setiap log memuat tiga informasi utama, yaitu jenis aksi yang dilakukan (misalnya *EDIT*), identitas pengguna yang melakukan perubahan, serta tanggal dan waktu perubahan tersebut dilakukan.

Detail perubahan ditampilkan secara terstruktur dalam kotak berwarna abu-abu. Informasi yang disajikan mencakup nilai sebelumnya dan nilai baru dari setiap atribut yang mengalami perubahan, seperti jenis transaksi (*TRANSACTION_TYPE*) yang berubah dari ”pemasukan” menjadi ”pengeluaran”, jumlah nominal transaksi (*AMOUNT*) yang diperbarui dari nilai lama ke nilai baru, deskripsi transaksi (`DESCRIPTION`) yang diubah dari teks sebelumnya ke teks terbaru, serta tanggal

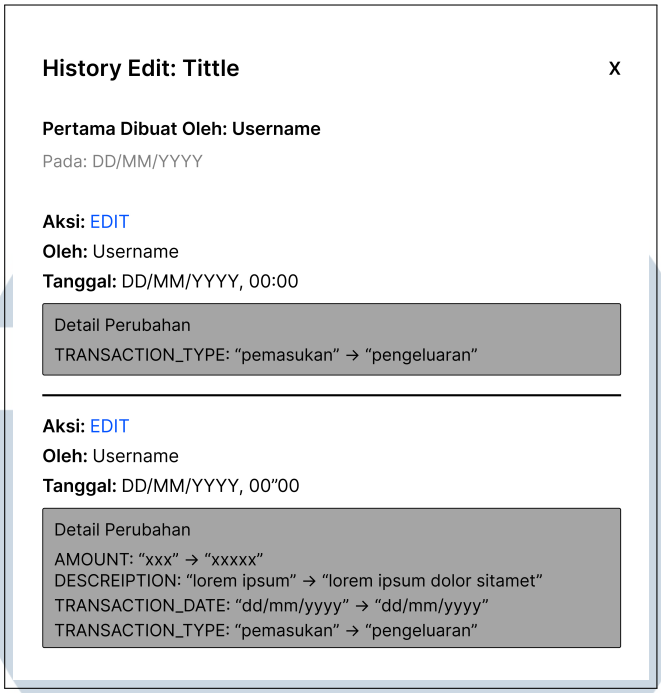
transaksi (*TRANSACTION_DATE*) yang dimutakhirkan sesuai input pengguna.

Seluruh data log ini diambil dari tabel *TR_MANUAL_TRANSACTIONS_HISTORY*, yang secara otomatis mencatat setiap perubahan melalui objek *JSON* berisi *previous values* dan *new values*. Tampilan ini memberikan kejelasan penuh terhadap proses modifikasi data, sehingga pengguna dengan peran Pengurus dapat mengetahui siapa yang mengubah data, kapan perubahan dilakukan, dan bagian mana saja yang diubah. Fitur ini menjadi bagian integral dari sistem audit trail yang diterapkan dalam pengelolaan data transaksi manual koperasi.

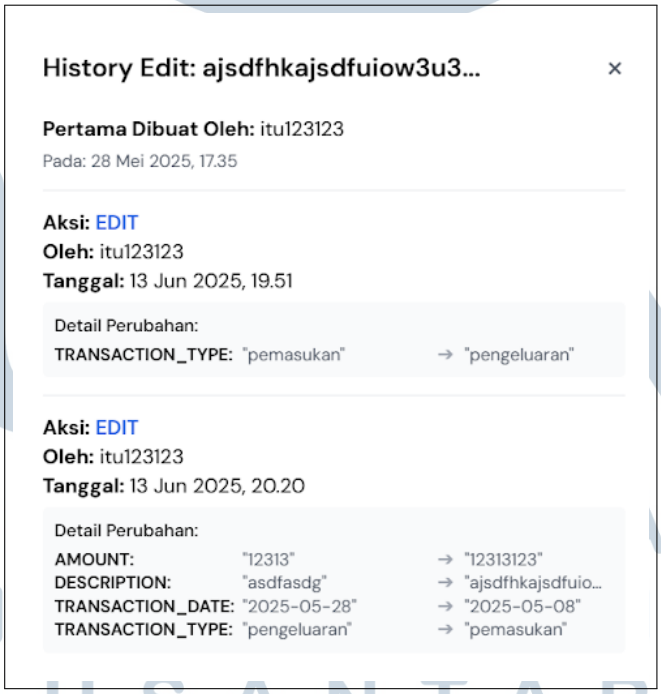
Gambar 3.15, 3.16, dan 3.17 berikut menunjukkan desain tampilan modal ini dari tahap awal perancangan hingga implementasi akhir.



Gambar 3.15. *Wireframe Low-Fidelity* Modal History Edit



Gambar 3.16. *Wireframe Medium-Fidelity* Modal History Edit



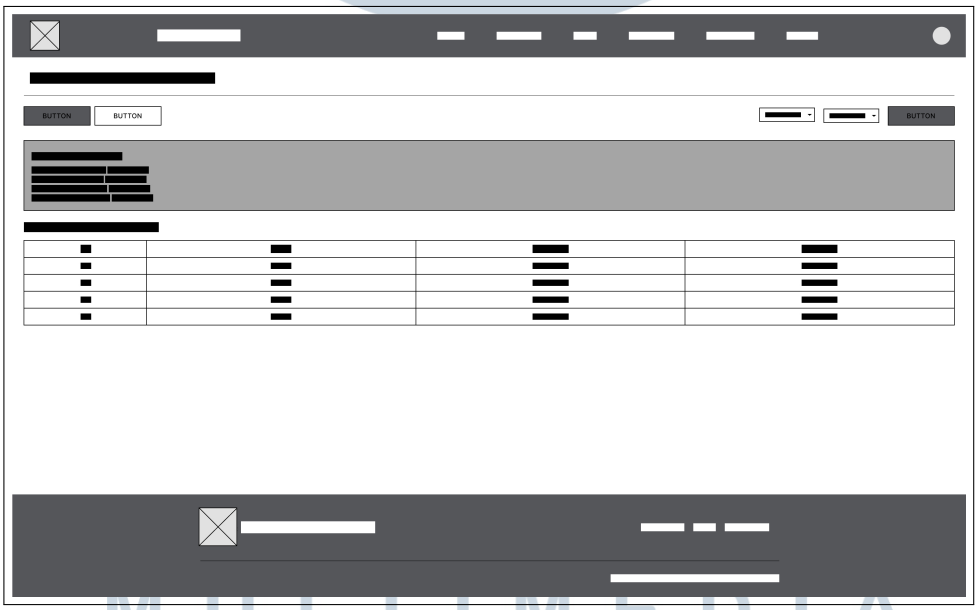
Gambar 3.17. Tampilan Akhir Modal History Edit Transaksi Manual

D.4 Tampilan Halaman Laporan Keuangan Anggota

Halaman ini akan ditampilkan secara otomatis ketika pengguna dengan peran sebagai Pengurus mengakses modul Laporan Keuangan dan memilih tab "Laporan Anggota". Halaman ini dirancang khusus untuk menyajikan data keuangan individual setiap anggota koperasi secara transparan dan terstruktur, sehingga memudahkan pengurus dalam melakukan pemantauan dan evaluasi terhadap aktivitas simpan pinjam anggota selama periode tertentu.

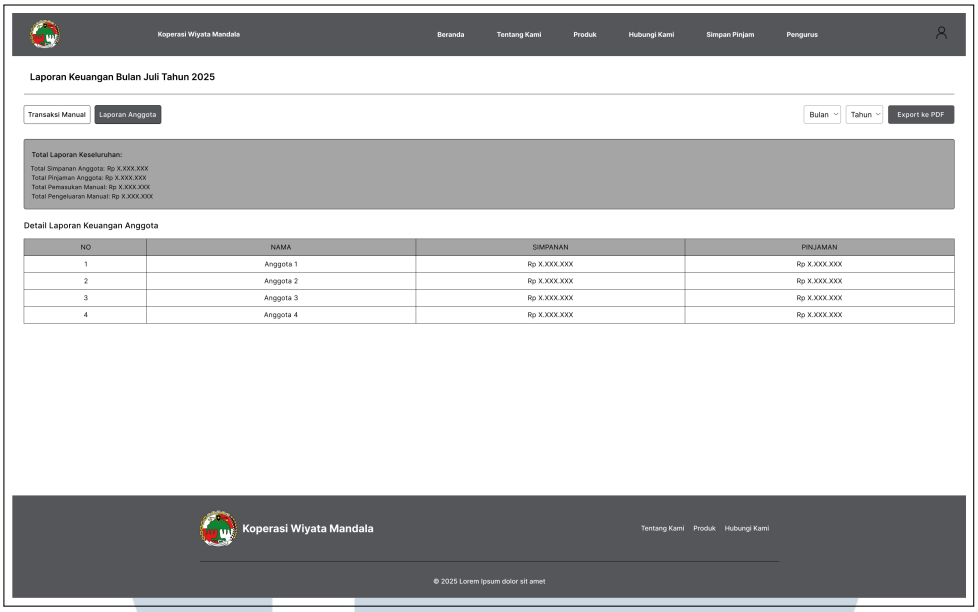
Sebelum tampilan ini diimplementasikan secara penuh, dirancang terlebih dahulu dua versi *wireframe*: *low-fidelity* dan *medium-fidelity*, sebagai langkah awal dalam proses desain antarmuka. *Wireframe* ini membantu tim pengembang dan *stakeholder* memahami struktur halaman, letak elemen, dan alur interaksi sebelum dikembangkan secara teknis.

Gambar 3.18 menggambarkan rancangan awal halaman, dengan struktur navigasi, ringkasan laporan, dan tabel anggota yang disusun dalam *layout* sederhana. Pada tahap ini, elemen visual masih dibentuk dengan blok dan garis, yang difokuskan pada susunan logis informasi dan keterbacaan.



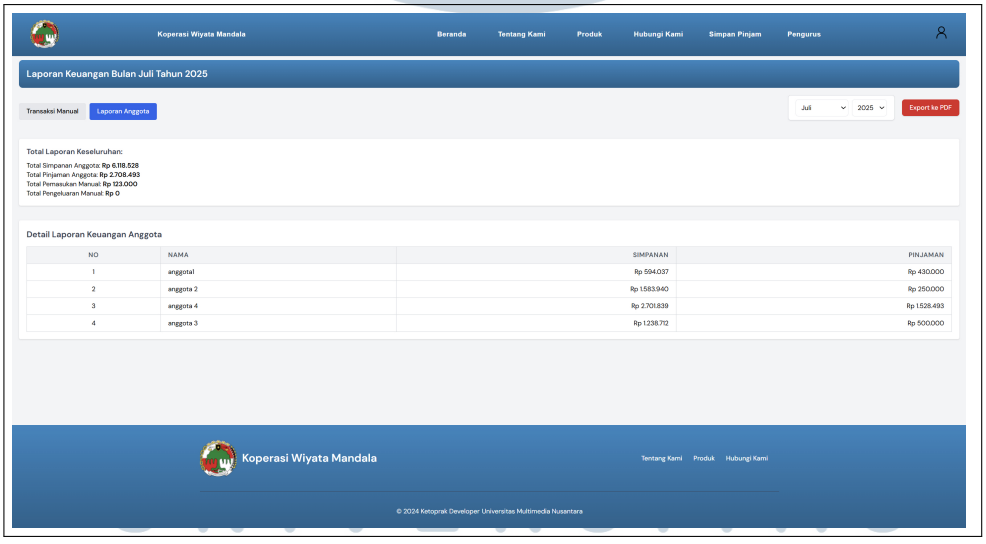
Gambar 3.18. *Wireframe Low-Fidelity* Halaman Laporan Anggota

Gambar 3.19 menampilkan versi *wireframe* dengan detail visual yang lebih kaya, termasuk label teks, tombol interaktif, dan struktur tabel yang menyerupai tampilan akhir. Tahap ini digunakan untuk validasi visual sebelum dilakukan pengembangan antarmuka pengguna secara penuh.



Gambar 3.19. *Wireframe Medium-Fidelity* Halaman Laporan Anggota

Gambar 3.20 berikut merupakan tampilan halaman laporan keuangan anggota setelah proses pengembangan selesai dilakukan.



Gambar 3.20. Halaman Laporan Keuangan Anggota setelah Pengembangan

Tampilan halaman ini terdiri atas dua bagian utama. Pada bagian atas, terdapat informasi ringkasan total laporan keseluruhan yang mencakup: total simpanan anggota, total pinjaman anggota, total pemasukan manual, dan total pengeluaran manual. Nilai-nilai ini ditampilkan dalam format angka yang telah diformat sesuai mata uang dan diperoleh secara otomatis dari hasil kalkulasi data

simpanan, pinjaman, serta transaksi manual yang tercatat di sistem.

Di bagian kanan atas halaman terdapat elemen interaktif berupa *dropdown* filter bulan dan tahun, yang memungkinkan pengguna menampilkan laporan keuangan berdasarkan periode tertentu. Tepat di sebelah elemen filter, terdapat tombol "Export ke PDF" yang digunakan untuk mengekspor laporan bulanan ke dalam dokumen digital. Tombol ini hadir secara konsisten di seluruh halaman dalam modul laporan keuangan, termasuk pada halaman Laporan Anggota dan Transaksi Manual.

Bagian bawah halaman menampilkan tabel Detail Laporan Keuangan Anggota, yang memuat data simpanan dan pinjaman per anggota koperasi. Informasi ini diperoleh secara otomatis dari hasil perhitungan terhadap data transaksi simpanan dan pinjaman yang tersimpan dalam tabel *TR_PENGAJUAN_SIMPANAN* dan *TR_PENGAJUAN_PINJAMAN*. Tabel ini memuat kolom nomor urut, nama anggota, jumlah simpanan, serta jumlah pinjaman dalam periode yang dipilih.

D.5 Sistem Export PDF

Modul laporan keuangan juga dilengkapi dengan fitur ekspor ke PDF, yang berfungsi untuk menyimpan hasil laporan keuangan bulanan dalam bentuk dokumen digital. Fitur ini dapat diakses oleh pengguna dengan peran Pengurus dari kedua halaman utama, yaitu Laporan Anggota dan Transaksi Manual. Tombol "Export ke PDF" ditempatkan secara konsisten di bagian kanan atas halaman, berdampingan dengan filter bulan dan tahun, sehingga memudahkan pengguna dalam melakukan dokumentasi laporan berdasarkan periode tertentu.

Pada sisi implementasi, proses ekspor ke PDF dikembangkan menggunakan pustaka pihak ketiga *jspdf* dan *jspdf-autotable*, yang memungkinkan pembangkitan dokumen PDF secara dinamis langsung dari data yang ditampilkan di antarmuka pengguna. Dengan kombinasi kedua pustaka tersebut, sistem mampu menghasilkan dokumen PDF dengan struktur tabel yang rapi, lengkap dengan judul, informasi periode laporan, serta total agregat keuangan.

Ketika tombol ekspor ditekan, sistem akan menggabungkan data dari dua sumber utama, yaitu:

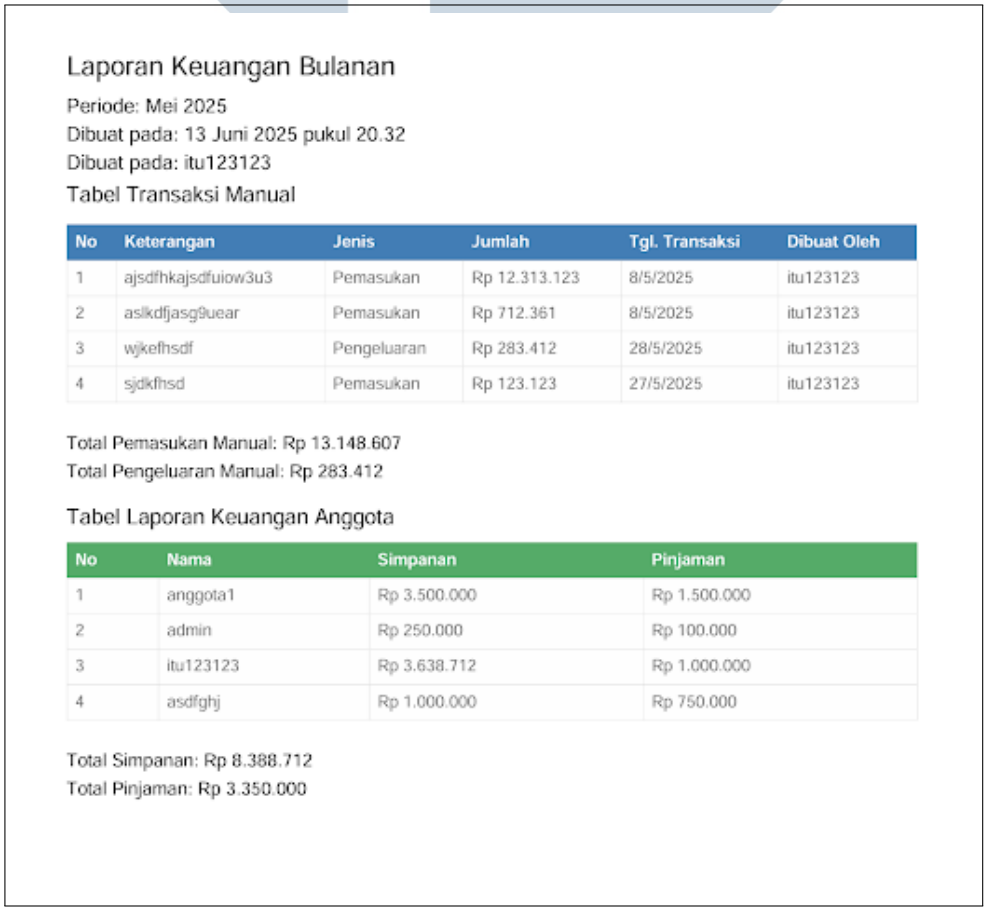
- Laporan keuangan anggota, yang dihasilkan secara otomatis dari data simpanan dan pinjaman anggota dalam tabel *TR_MONTHLY_FINANCE_ANGGOTA*, dan

- Transaksi manual, yang tercatat dalam tabel *TR_MANUAL_TRANSACTIONS*.

Hasil penggabungan kedua data tersebut kemudian dirangkum dan disimpan ke dalam tabel *TR_MONTHLY_FINANCE_STATEMENT*. Tabel ini berfungsi sebagai arsip laporan bulanan final, yang mencakup nilai total simpanan, total pinjaman, total pemasukan manual, dan total pengeluaran manual. Selain itu, sistem juga mencatat informasi pengguna yang melakukan ekspor beserta waktu pelaksanaannya, melalui kolom *USR_CRT* dan *DTM_CRT*.

Dengan adanya fitur ekspor ini, sistem mampu menghasilkan *output* dokumen yang siap digunakan sebagai bukti administratif atau pelaporan resmi koperasi, serta mendukung aspek dokumentasi jangka panjang secara sistematis dan terdigitalisasi.

Gambar 3.21 berikut memperlihatkan hasil akhir dari proses *export* laporan keuangan ke dalam format PDF, yang dihasilkan secara otomatis oleh sistem berdasarkan data periode yang dipilih.



Laporan Keuangan Bulanan

Periode: Mei 2025
Dibuat pada: 13 Juni 2025 pukul 20.32
Dibuat pada: itu123123

Tabel Transaksi Manual

No	Keterangan	Jenis	Jumlah	Tgl. Transaksi	Dibuat Oleh
1	ajsdfhkajsdfuiow3u3	Pemasukan	Rp 12.313.123	8/5/2025	itu123123
2	aslkdfjasg9uear	Pemasukan	Rp 712.361	8/5/2025	itu123123
3	wjkefhsdf	Pengeluaran	Rp 283.412	28/5/2025	itu123123
4	sjdkfhsd	Pemasukan	Rp 123.123	27/5/2025	itu123123

Total Pemasukan Manual: Rp 13.148.607
Total Pengeluaran Manual: Rp 283.412

Tabel Laporan Keuangan Anggota

No	Nama	Simpanan	Pinjaman
1	anggota1	Rp 3.500.000	Rp 1.500.000
2	admin	Rp 250.000	Rp 100.000
3	itu123123	Rp 3.638.712	Rp 1.000.000
4	asdfghj	Rp 1.000.000	Rp 750.000

Total Simpanan: Rp 8.388.712
Total Pinjaman: Rp 3.350.000

Gambar 3.21. Tampilan PDF setelah di-export

D.6 Automasi Refresh Token untuk Sinkronisasi Data Identitas Pengguna

Untuk menjaga agar informasi pengguna yang tersimpan dalam *JWT* (*JSON Web Token*) tetap terbawa dan tercatat secara valid dalam setiap proses transaksi, sistem ini menggunakan konfigurasi khusus bernama *axiosInterceptor*. Konfigurasi ini berfungsi untuk menyisipkan token secara otomatis ke setiap permintaan *HTTP* serta memperbarui token ketika token akses telah kedaluwarsa. Mekanisme ini sangat penting untuk memastikan bahwa data identitas pengguna, seperti nama lengkap dari tabel *MS_USER*, dapat dicatat dengan benar ke dalam kolom seperti *USR_CRT* atau *USR_UPD* di berbagai tabel lain, termasuk *TR_MANUAL_TRANSACTIONS* dan *TR_MANUAL_TRANSACTIONS_HISTORY*.

Potongan kode 3.1 menunjukkan pembuatan instance khusus bernama *axiosInterceptor* dan konfigurasi *interceptor* pada bagian permintaan (*request*). Setiap permintaan akan secara otomatis menyertakan token akses ke dalam *header Authorization*, sehingga server dapat memverifikasi identitas pengguna berdasarkan *JWT* yang dibawa.

```
axiosInterceptor.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('accessToken');
    if (token) {
      config.headers['Authorization'] = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);
```

Kode 3.1: Potongan kode untuk Inisialisasi axiosInterceptor dan Penyisipan Token Otomatis

Kode 3.2 digunakan untuk mengatur permintaan yang tertunda selama proses *refresh token* sedang berlangsung. Hal ini penting untuk mencegah permintaan berulang ke *endpoint refresh token* dan memastikan semua permintaan menunggu *token* baru yang sama.

```
let isRefreshing = false;
let failedQueue = [];

const processQueue = (error, token = null) => {
  failedQueue.forEach(prom => {
```

```
    if (error) {
      prom.reject(error);
    } else {
      prom.resolve(token);
    }
  });
  failedQueue = [];
};
```

Kode 3.2: Potongan kode untuk penanganan antrian saat proses *Refresh Token* sedang berlangsung

Potongan kode 3.3 berfungsi untuk mendeteksi apakah *token JWT* yang dibawa pengguna telah kedaluwarsa (biasanya melalui status "*TOKEN_EXPIRED*"). Jika *token* masih valid, respons akan langsung diteruskan. Namun jika kadaluarsa, maka akan dicek apakah proses *refresh* sedang berjalan. Jika iya, permintaan akan dimasukkan ke dalam antrian *failedQueue* dan dieksekusi setelah *token* baru tersedia.

```
axiosInterceptor.interceptors.response.use(
  (response) => {
    return response;
  },
  async (error) => {
    const originalRequest = error.config;

    if (error.response && error.response.data && error.response.
    data.code === "TOKEN_EXPIRED" && !originalRequest._retry) {

      if (isRefreshing) {
        return new Promise(function (resolve, reject) {
          failedQueue.push({ resolve, reject });
        })
          .then(token => {
            originalRequest.headers['Authorization'] = 'Bearer ' +
    token;
            return axiosInterceptor(originalRequest);
          })
          .catch(err => {
            return Promise.reject(err);
          });
      }

      originalRequest._retry = true;
```

```
24        isRefreshing = true;
```

Kode 3.3: Potongan kode untuk Deteksi Token Kadaluarsa dan Penanganan Antrian

Potongan kode 3.4 menangani proses *refresh token* secara langsung. Sistem mengirim permintaan ke *endpoint* /token, lalu menyimpan *access token* baru ke *localStorage*. Jika proses berhasil, permintaan awal akan dijalankan ulang. Namun, jika gagal, sistem akan membersihkan data sesi dan mengarahkan pengguna kembali ke halaman login.

```
1  try {
2          const response = await axios.get('http://localhost:5000/
       token', {
3            withCredentials: true
4          });
5
6          const newAccessToken = response.data.accessToken;
7          localStorage.setItem('accessToken', newAccessToken);
8
9          if (axiosInterceptor.defaults.headers.common) {
10             axiosInterceptor.defaults.headers.common['
       Authorization'] = 'Bearer ' + newAccessToken;
11         }
12         originalRequest.headers['Authorization'] = 'Bearer ' +
       newAccessToken;
13
14         processQueue(null, newAccessToken);
15
16         return axiosInterceptor(originalRequest);
17
18       } catch (refreshError) {
19         processQueue(refreshError, null);
20         localStorage.removeItem('accessToken');
21         localStorage.removeItem('UUID_MS_JOB');
22
23         alert("Sesi Anda telah berakhir atau refresh token gagal.
       Silakan login kembali.");
24         window.location.href = '/login';
25
26         return Promise.reject(refreshError);
27       } finally {
28         isRefreshing = false;
29       }
```

Kode 3.4: Potongan kode untuk Refresh Token, Simpan Token Baru, dan Penanganan Gagal

Kode *axiosInterceptor* yang diimplementasikan dalam sistem ini berfungsi sebagai solusi otomatis untuk menjaga sesi autentikasi pengguna tetap aktif tanpa perlu melakukan login ulang secara manual. Fungsinya sangat penting dalam konteks pengelolaan data koperasi berbasis web, di mana setiap transaksi atau aktivitas pengguna membutuhkan validasi identitas melalui *JWT* (*JSON Web Token*).

Interceptor ini bekerja dalam dua arah utama:

1. Pada saat request dikirim, token akses yang disimpan secara lokal (*localStorage*) secara otomatis disisipkan ke dalam *header Authorization*. Hal ini memastikan setiap permintaan ke *server* membawa informasi identitas pengguna, yang diperlukan untuk proses audit seperti pencatatan *USR_CRT* dan *USR_UPD* pada berbagai tabel transaksi.

2. Pada saat menerima *response*, sistem akan mendeteksi apakah *token* telah kadaluarsa (*TOKEN_EXPIRED*). Jika ya, maka sistem akan secara otomatis mengirim permintaan *refresh token* ke *server*. Bila berhasil, token yang baru disimpan kembali dan permintaan asli akan diulang secara mulus. Jika gagal, maka token lama akan dihapus, dan pengguna akan diarahkan ke halaman login.

Selain itu, sistem juga menangani kondisi ketika beberapa permintaan terjadi saat proses *refresh* sedang berlangsung. Permintaan tersebut dimasukkan ke dalam antrian (*queue*) dan diproses ulang setelah *token* diperbarui, sehingga mencegah tabrakan permintaan yang tidak sinkron.

Secara keseluruhan, mekanisme ini meningkatkan keamanan, keterhubungan data pengguna, dan kenyamanan penggunaan sistem koperasi digital, terutama untuk fitur-fitur penting seperti pencatatan transaksi manual, penyimpanan data laporan keuangan, dan manajemen identitas pengguna.

D.7 Logika Backend Laporan Keuangan

Sistem laporan keuangan bulanan koperasi terdiri atas dua komponen utama, yaitu laporan keuangan per anggota dan laporan transaksi manual. Masing-masing komponen memiliki fungsi *backend* khusus yang bertugas melakukan perhitungan, pencatatan, pembaruan, serta penyimpanan data ke dalam tabel yang relevan. Seluruh logika dijalankan melalui *controller* berbasis *REST API* yang dihubungkan dengan model *Sequelize*.

Perhitungan simpanan dan pinjaman bulanan setiap anggota dilakukan melalui fungsi *getMonthlyFinanceOverview*. Fungsi ini akan menghimpun data dari tabel *TR_PENGAJUAN_SIMPANAN* dan *TR_PENGAJUAN_PINJAMAN* yang telah berstatus Disetujui, lalu menjumlahkan nilai nominalnya berdasarkan filter bulan dan tahun. Selanjutnya, hasil perhitungan ini disimpan ke dalam tabel *TR_MONTHLY_FINANCE_ANGGOTA* menggunakan metode *findOrCreate*, yang secara otomatis akan memperbarui data jika sudah pernah dibuat sebelumnya.

```
const [financeRecord, created] = await TrMonthlyFinanceAnggota.findOrCreate({
     where: { UUID_MS_USER: user.UUID_MS_USER, MONTH: month, YEAR: year },
     defaults: {
       AMOUNT_SIMPANAN: totalSimpanan,
       AMOUNT_PINJAMAN: totalPinjaman,
       USR_CRT: 'SYSTEM_GET_OVERVIEW',
       USR_UPD: 'SYSTEM_GET_OVERVIEW',
       DTM_UPD: new Date(),
     }
   });

   if (!created) {
     await financeRecord.update({
       AMOUNT_SIMPANAN: totalSimpanan,
       AMOUNT_PINJAMAN: totalPinjaman,
       USR_UPD: 'SYSTEM_GET_OVERVIEW_UPDATE',
       DTM_UPD: new Date(),
     });
   }
```

Kode 3.5: Logika penyimpanan otomatis laporan bulanan per anggota

Modul transaksi manual memungkinkan pengguna (pengurus) untuk mencatat pemasukan atau pengeluaran yang tidak tercakup dalam sistem otomatis. Data transaksi ini disimpan ke tabel *TR_MANUAL_TRANSACTIONS*. Fungsi *addManualTransaction* menangani validasi input serta konversi data numerik dan tanggal sebelum penyimpanan.

```
const newTransaction = await TrManualTransactions.create({
     TRANSACTION_TYPE, AMOUNT: parseFloat(AMOUNT), DESCRIPTION,
     TRANSACTION_DATE: actualTransactionDate, MONTH: parseInt(MONTH), YEAR: parseInt(YEAR),
     USR_CRT: determinedUsrCrt, DTM_CRT: currentDate,
```

```
      USR_UPD: determinedUsrCrt, DTM_UPD: currentDate,
      IS_DELETED: false, EDIT_COUNT: 0
    });
```

Kode 3.6: Penyimpanan transaksi manual ke dalam *database*

Salah satu fitur penting dalam pengelolaan transaksi manual adalah pencatatan riwayat perubahan data. Setiap kali data transaksi diedit, sistem akan menyimpan jejak perubahan ke tabel *TR_MANUAL_TRANSACTIONS_HISTORY*. Fungsi *updateManualTransaction* membandingkan data lama dan data baru, mencatat kolom yang berubah, lalu menyimpannya sebagai *log*.

```
if (changedFieldsList.length > 0) {
        await TrManualTransactionsHistory.create({
            UUID_MANUAL_TRANSACTION: id,
            CHANGED_FIELDS: changedFieldsList.join(', '),
            PREVIOUS_VALUES: previousValues,
            NEW_VALUES: newValues,
            ACTION_TYPE: 'EDIT',
            EDITED_BY: currentUser,
        }, { transaction: t });
    }
```

Kode 3.7: Penyimpanan riwayat perubahan transaksi manual

Setiap kali pengguna melakukan ekspor laporan keuangan dalam bentuk PDF, sistem akan menyimpan data rekapitulasi ke dalam tabel *TR_MONTHLY_FINANCIAL_STATEMENT*. Fungsi *createMonthlyStatement* akan mencatat total simpanan, pinjaman, pemasukan, dan pengeluaran untuk bulan dan tahun tertentu.

```
const result = await TrMonthlyFinancialStatement.create({
      MONTH: req.body.MONTH,
      YEAR: req.body.YEAR,
      TOTAL_AMOUNT_SIMPANAN: req.body.TOTAL_SIMPANAN,
      TOTAL_AMOUNT_PINJAMAN: req.body.TOTAL_PINJAMAN,
      TOTAL_INCOME: req.body.TOTAL_PEMASUKAN_MANUAL,
      TOTAL_EXPENDITURE: req.body.TOTAL_PENGELUARAN_MANUAL,
      USR_CRT: currentUser
    });
```

Kode 3.8: Penyimpanan rekap laporan keuangan ke dalam tabel *TR_MONTHLY_FINANCE_STATEMENT*

Seluruh proses *backend* pada modul laporan keuangan dirancang untuk bekerja secara dinamis dan terintegrasi, memastikan bahwa data yang tercatat selalu

konsisten dan dapat ditelusuri secara historis. Dengan pendekatan ini, sistem mendukung pencatatan kredibel dan transparan yang penting bagi pengelolaan koperasi secara digital.

D.8 Automasi Penamaan Export File Laporan Keuangan

Salah satu fitur penting yang dikembangkan dalam modul laporan keuangan adalah sistem otomatisasi penamaan file PDF yang diekspor setiap bulannya. Fitur ini dirancang untuk menciptakan format penamaan yang konsisten, informatif, dan unik untuk setiap file ekspor, sekaligus menangani kasus ketika pengguna melakukan ekspor lebih dari sekali dalam periode yang sama.

Sistem ini bekerja dengan cara mengecek terlebih dahulu berapa kali laporan untuk bulan dan tahun tertentu telah diekspor sebelumnya, menggunakan *endpoint backend* pada potongan kode 3.9

```
router.get('/monthly-finance-statement/count', async (req, res) =>
    {
  try {
    const { month, year } = req.query;

    const count = await TrMonthlyFinancialStatement.count({
      where: {
        MONTH: parseInt(month),
        YEAR: parseInt(year),
      },
    });

    res.json({ count });
  } catch (error) {
    console.error("Gagal menghitung jumlah ekspor:", error);
    res.status(500).json({ message: "Internal server error" });
  }
});
```

Kode 3.9: *Endpoint Backend* untuk Mendapatkan Jumlah Ekspor Laporan

Pada sisi *frontend*, data jumlah ekspor sebelumnya (*previousExportCount*) ini diambil dan digunakan untuk menyusun nama file berdasarkan bulan dan tahun. Jika *previousExportCount* bernilai lebih dari nol, maka sistem secara otomatis akan menambahkan label “Revisi” dengan nomor urutan ekspor.

```
const { data: countRes } = await axiosInterceptor.get(
```

```
        `/monthly-finance-statement/count?month=${month}&year=${year}`
      );
      const previousExportCount = countRes.count || 0;

      let fileName = `Laporan Keuangan ${months[month - 1]} ${year}`;
      if (previousExportCount > 0) {
        fileName += ` Revisi ${previousExportCount}`;
      }
      fileName += `.pdf`;

      doc.save(fileName);
```

Kode 3.10: Logika Penamaan File Ekspor di *Frontend*

Contoh hasil nama file yang akan dihasilkan oleh sistem meliputi:

- Laporan Keuangan Mei 2025.pdf – untuk ekspor pertama kali.
- Laporan Keuangan Mei 2025 Revisi 1.pdf – untuk ekspor kedua.
- Laporan Keuangan Mei 2025 Revisi 2.pdf – untuk ekspor ketiga, dan seterusnya.

Dengan cara ini, sistem tidak hanya menghindari konflik penamaan file saat diunduh oleh pengguna, tetapi juga mempermudah pelacakan riwayat ekspor dan audit laporan keuangan. Penamaan yang otomatis dan informatif ini menjadi bagian kecil namun penting dalam memastikan sistem tetap profesional dan terorganisasi dengan baik.

3.3.2 Modul Sisa Hasil Usaha (SHU)

A Sitemap

Setelah pengguna dengan peran Pengurus berhasil login, akan muncul dropdown khusus pada navbar yang berisi opsi "SHU". Ketika opsi ini diklik, pengguna diarahkan ke halaman utama modul SHU yang menampilkan data keuangan anggota serta persentase pembagian SHU (jika sudah diinput). Terdapat dropdown filter tahun dan tombol "Input Persentase SHU" pada halaman ini.

Saat tombol tersebut diklik, pengguna diarahkan ke halaman input SHU, yang menampilkan kembali total SHU dan tabel data anggota dengan kolom untuk

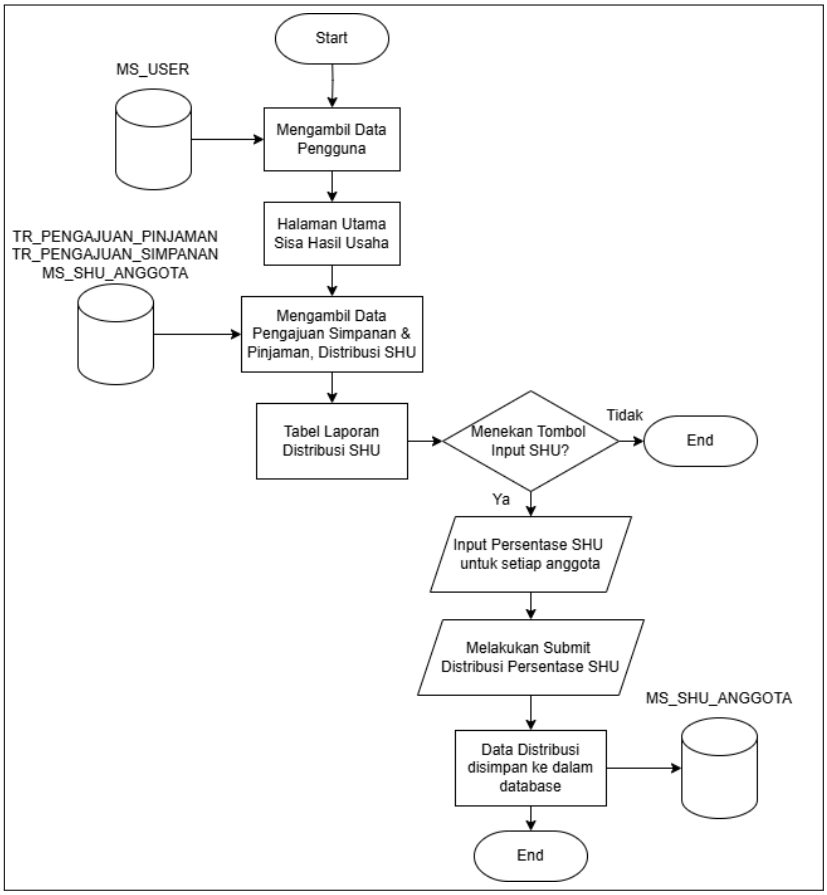
mengisi persentase pembagian SHU secara manual. Semua fitur ini hanya tersedia untuk pengguna dengan peran Pengurus.

Gambar 3.22. Sitemap Modul Sisa Hasil Usaha (SHU)

B Flowchart

Proses perhitungan dan distribusi SHU dimulai ketika pengguna dengan peran Pengurus mengakses halaman utama modul SHU melalui navigasi. Sistem akan mengambil dan menggabungkan data dari tabel *MS_USER*, *TR_PENGAJUAN_SIMPANAN*, dan *TR_PENGAJUAN_PINJAMAN* untuk menampilkan informasi total SHU koperasi, beserta tabel rekapitulasi simpanan dan pinjaman anggota selama satu tahun.

Dari halaman utama ini, pengurus dapat mengklik tombol Input SHU untuk diarahkan ke halaman pengisian persentase. Setelah mengisi persentase pembagian SHU untuk masing-masing anggota, data yang telah ditetapkan akan disimpan ke dalam tabel *MS_SHU_ANGGOTA*. Seluruh proses ini hanya dapat dilakukan oleh pengguna dengan role “Pengurus”.



Gambar 3.23. Flowchart Modul Sisa Hasil Usaha (SHU)

C Database

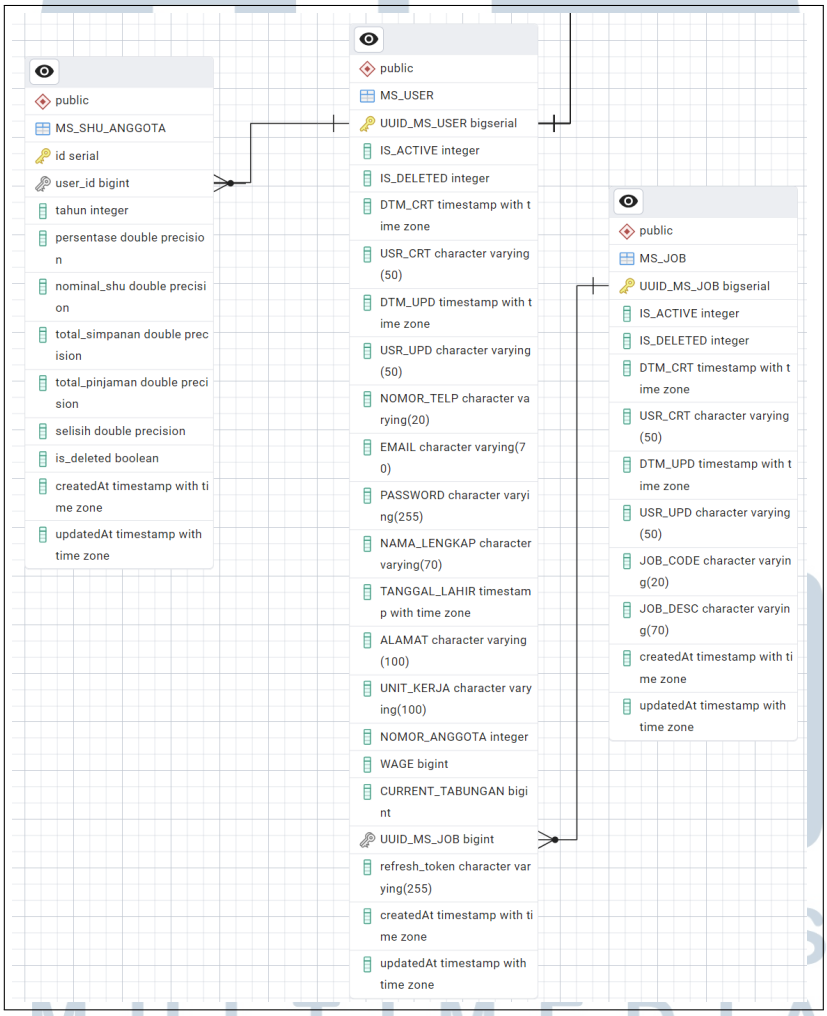
C.1 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) pada sistem ini menggambarkan struktur relasi antar tabel yang membentuk dasar dari modul keanggotaan dan pembagian Sisa Hasil Usaha (SHU) koperasi. Tabel pusat *MS_USER* menyimpan data pengguna atau anggota koperasi, termasuk informasi pribadi seperti nama lengkap, alamat, tanggal lahir, serta data keuangan seperti gaji dan tabungan saat ini. Tabel ini memiliki relasi langsung ke *MS_JOB* melalui *foreign key UUID_MS_JOB*, yang digunakan untuk menetapkan jenis pekerjaan atau peran dari masing-masing pengguna.

Selanjutnya, tabel *MS_SHU_ANGGOTA* berelasi dengan *MS_USER* melalui *foreign key user_id*, yang merujuk ke *UUID_MS_USER*. Tabel ini menyimpan data perhitungan pembagian SHU per anggota tiap tahunnya, termasuk persentase,

nominal SHU, total simpanan, total pinjaman, dan selisihnya. Relasi ini memungkinkan sistem mencatat riwayat pembagian SHU berdasarkan identitas unik setiap anggota.

Walaupun ERD ini hanya menampilkan tiga tabel inti (*MS_USER, MS_JOB*, dan *MS_SHU_ANGGOTA*), struktur relasionalnya cukup menggambarkan keterkaitan antara data pengguna, jenis pekerjaan, dan hasil pembagian SHU. Setiap elemen dirancang untuk mendukung keakuratan pengolahan data keuangan dan pelaporan distribusi SHU tahunan.



Gambar 3.24. ERD Modul Sisa Hasil Usaha (SHU)

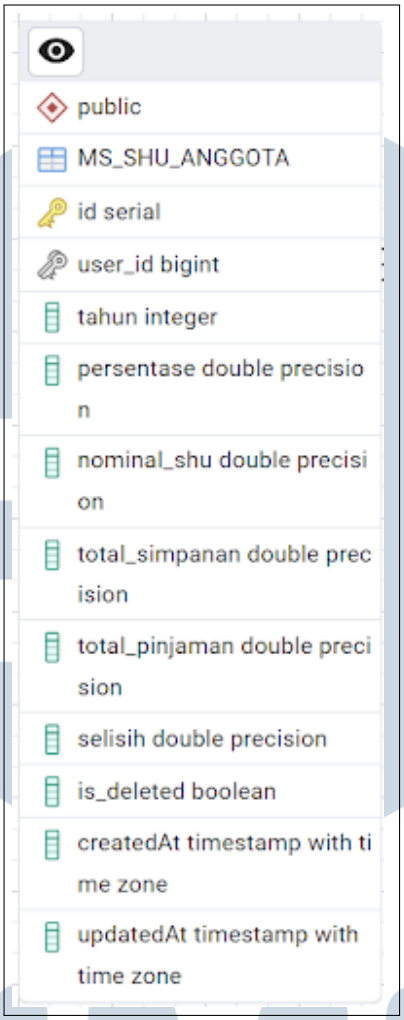
C.2 Model MS_SHU_ANGGOTA

Tabel *MS_SHU_ANGGOTA* digunakan untuk mencatat hasil perhitungan dan distribusi Sisa Hasil Usaha (SHU) bagi masing-masing anggota koperasi. Setiap

entri dalam tabel ini merepresentasikan data SHU per anggota untuk satu periode tahun tertentu. Identitas anggota ditautkan melalui *field user_id,* yang merupakan *foreign key* yang merujuk ke *primary key* dalam tabel *MS_USER.*

Data yang dicatat dalam tabel ini mencakup nilai persentase SHU, nominal SHU yang diterima, total simpanan, total pinjaman, serta selisih antara simpanan dan pinjaman. Seluruh nilai disimpan dalam format numerik presisi tinggi (*double precision*) untuk menjaga akurasi perhitungan. Kolom tahun digunakan untuk mengelompokkan data berdasarkan periode laporan tahunan.

Selain itu, tabel ini menyertakan kolom *is_deleted* sebagai indikator status penghapusan logis, serta *createdAt* dan *updatedAt* sebagai informasi waktu pembuatan dan pembaruan data. Tabel *MS_SHU_ANGGOTA* menjadi komponen penting dalam sistem, karena berfungsi sebagai tempat penyimpanan hasil distribusi SHU yang telah dihitung dan diinput secara manual oleh pengurus berdasarkan data keuangan masing-masing anggota.



Gambar 3.25. Model Tabel *MS_SHU_ANGGOTA*

D Hasil Development

D.1 Tampilan Halaman Sisa Hasil Usaha

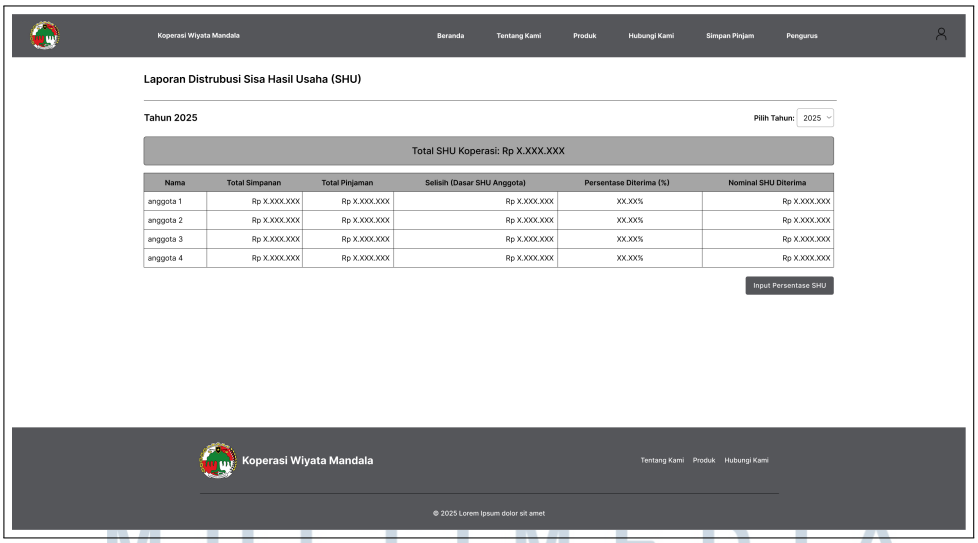
Halaman ini ditampilkan ketika pengguna dengan peran "Pengurus" memilih menu "Sisa Hasil Usaha" melalui *dropdown* pada navigasi utama sistem. Modul ini dirancang untuk menampilkan laporan distribusi Sisa Hasil Usaha (SHU) berdasarkan data total simpanan dan pinjaman setiap anggota koperasi.

Proses perancangan halaman ini diawali dengan pembuatan *wireframe low-fidelity* untuk menentukan struktur awal antarmuka pengguna, komponen inti, serta alur visualisasi data. *Wireframe* ini membantu dalam merancang pengalaman pengguna yang efisien dan terstruktur sejak tahap awal. Setelah disetujui, desain dikembangkan lebih lanjut ke bentuk *medium-fidelity* yang memperlihatkan elemen

UI secara lebih detail dan mendekati tampilan sistem akhir, seperti *layout* tabel, teks label, tombol interaktif, dan konsistensi posisi elemen.

Gambar 3.26 dan Gambar 3.27 berikut menunjukkan hasil desain wireframe *low-fidelity* dan *medium-fidelity* untuk halaman distribusi SHU.

Gambar 3.26. *Wireframe Low-fidelity* Halaman Sisa Hasil Usaha



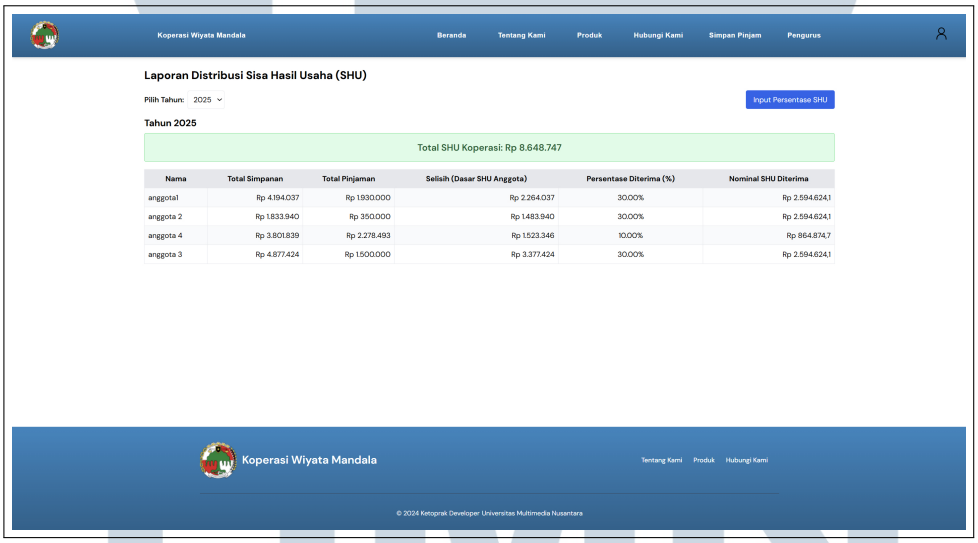
Gambar 3.27. *Wireframe Medium-fidelity* Halaman Sisa Hasil Usaha

Setelah tahap desain, pengembangan halaman dilakukan dengan struktur sebagai berikut. Pada bagian atas halaman, pengguna dapat memilih tahun laporan melalui komponen *dropdown* "Pilih Tahun", yang akan memfilter laporan berdasarkan data tahun tersebut. Tepat di bawahnya, ditampilkan nilai Total SHU

Koperasi yang dihitung secara otomatis dari total selisih antara simpanan dan pinjaman anggota koperasi. Nilai ini menjadi dasar utama dalam distribusi SHU kepada seluruh anggota.

Tabel distribusi SHU ditampilkan secara komprehensif dengan kolom: Nama, Total Simpanan, Total Pinjaman, Selisih (dasar SHU), Persentase Diterima (%), dan Nominal SHU Diterima. Untuk melakukan pengaturan nilai persentase, pengguna dapat menggunakan tombol "Input Persentase SHU" yang akan membuka halaman input manual. Semua data pada halaman ini dikelola melalui tabel *MS_SHU_ANGGOTA* dan diperbarui secara otomatis ketika data disimpan atau dimodifikasi.

Tampilan akhir halaman distribusi SHU yang telah dikembangkan secara fungsional ditunjukkan pada Gambar 3.28 berikut.



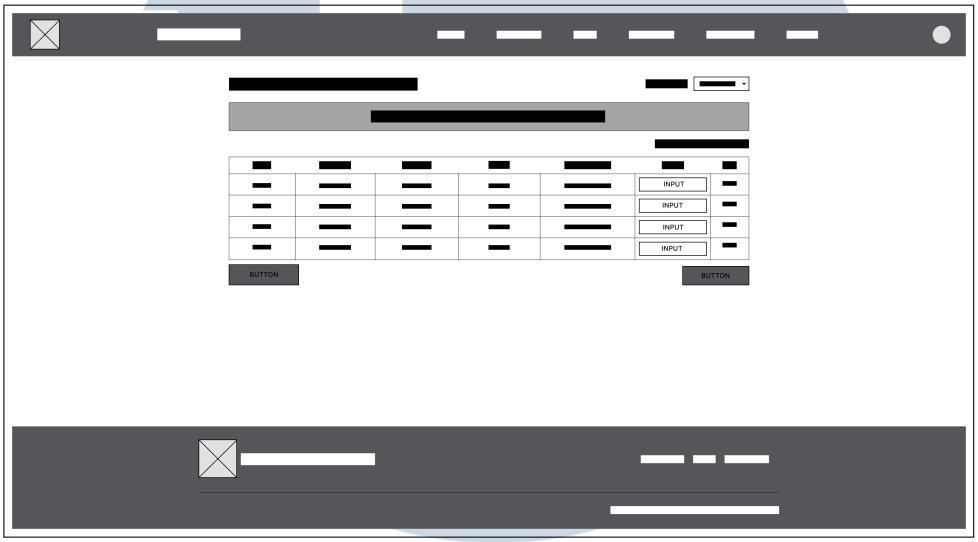
Gambar 3.28. Tampilan Halaman Sisa Hasil Usaha setelah Pengembangan

D.2 Tampilan Halaman Input SHU

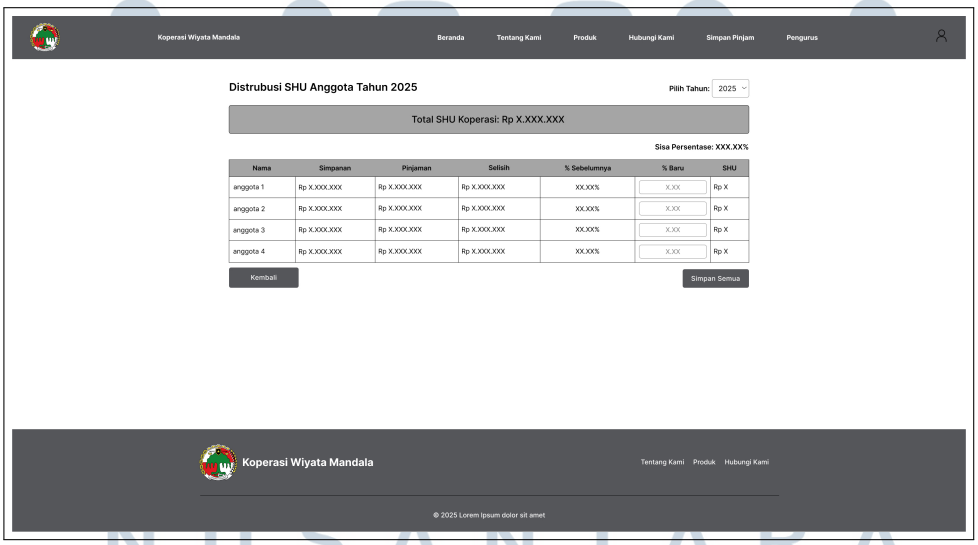
Halaman ini merupakan lanjutan dari proses pengelolaan Sisa Hasil Usaha (SHU), dan hanya dapat diakses oleh pengguna dengan peran "Pengurus" melalui tombol "Input Persentase SHU" pada halaman utama SHU. Tujuan utama dari halaman ini adalah untuk memasukkan atau memperbarui besaran persentase SHU yang diterima masing-masing anggota koperasi berdasarkan data keuangan mereka selama satu tahun.

Sebelum sampai pada tampilan akhir halaman ini, perancangan dilakukan terlebih dahulu melalui tahapan desain *wireframe*. Tahapan ini mencakup dua

versi: *low-fidelity wireframe* yang menggambarkan susunan dasar komponen dan struktur layout secara garis besar, serta *medium-fidelity wireframe* yang mulai memperlihatkan elemen interaktif, teks, dan penyusunan tabel yang lebih mendekati versi akhir. Tujuan dari tahap *wireframe* ini adalah untuk menguji dan memvalidasi tata letak serta alur fungsionalitas sebelum dilakukan pengembangan antarmuka secara menyeluruh.



Gambar 3.29. *Wireframe Low Fidelity* Halaman Input SHU



Gambar 3.30. *Wireframe Medium Fidelity* Halaman Input SHU

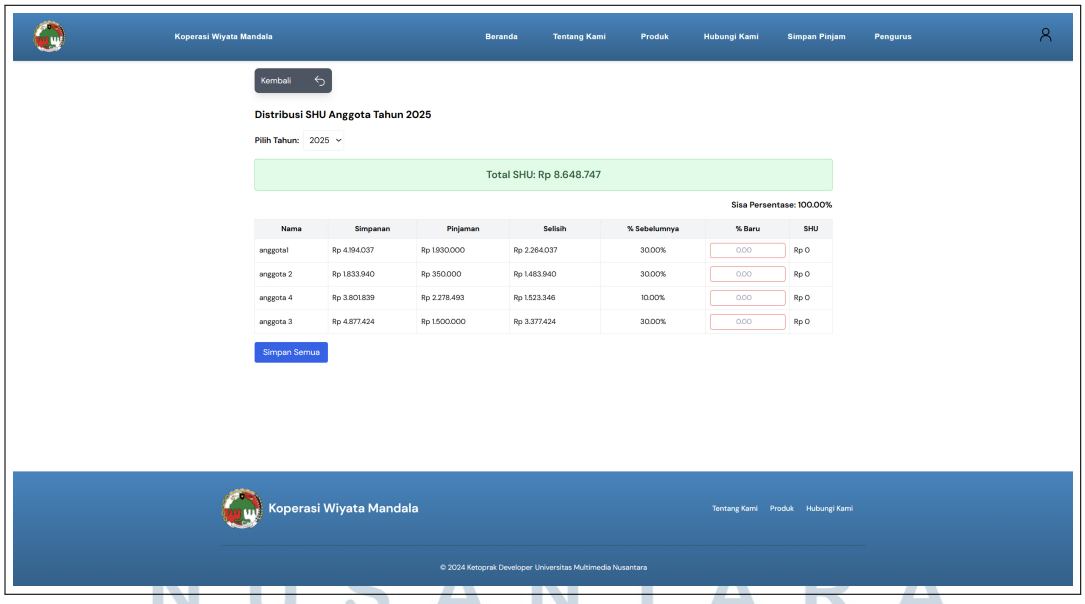
Setelah tahap desain disetujui, pengembangan UI dilakukan berdasarkan struktur *wireframe* tersebut. Di bagian atas halaman, pengguna dapat memilih

tahun distribusi SHU menggunakan *dropdown* "Pilih Tahun". Setelah tahun dipilih, sistem akan menampilkan informasi Total SHU koperasi yang menjadi dasar pembagian. Nilai ini dihitung dari total selisih antara simpanan dan pinjaman seluruh anggota.

Tabel pada halaman ini menampilkan data masing-masing anggota koperasi, termasuk kolom nama, jumlah simpanan, pinjaman, dan selisihnya. Selisih ini merupakan dasar proporsional pembagian SHU per anggota. Kolom "% Sebelumnya" menunjukkan persentase pembagian SHU yang telah disimpan sebelumnya dalam sistem (jika ada), sementara kolom "% Baru" merupakan *input field* yang dapat diisi oleh pengguna untuk menetapkan persentase terbaru secara manual.

Nilai nominal SHU yang diterima oleh anggota secara otomatis akan diperbarui berdasarkan persentase baru yang diinput dan akan ditampilkan dalam kolom "SHU". Di sisi kanan atas tabel, ditampilkan informasi "Sisa Persentase" yang berguna untuk memastikan total distribusi tidak melebihi 100%.

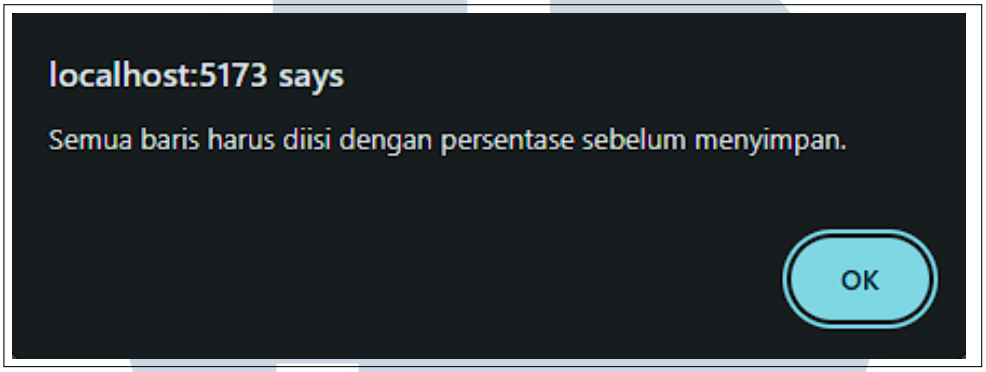
Setelah seluruh data persentase diperbarui, pengguna dapat menekan tombol "Simpan Semua" di bagian bawah halaman untuk menyimpan pembagian SHU ke dalam sistem. Data yang diinput akan direkam ke dalam tabel *MS_SHU_ANGGOTA* sebagai acuan pembagian resmi SHU pada tahun berjalan.



Gambar 3.31. Tampilan Halaman Input SHU setelah pengembangan

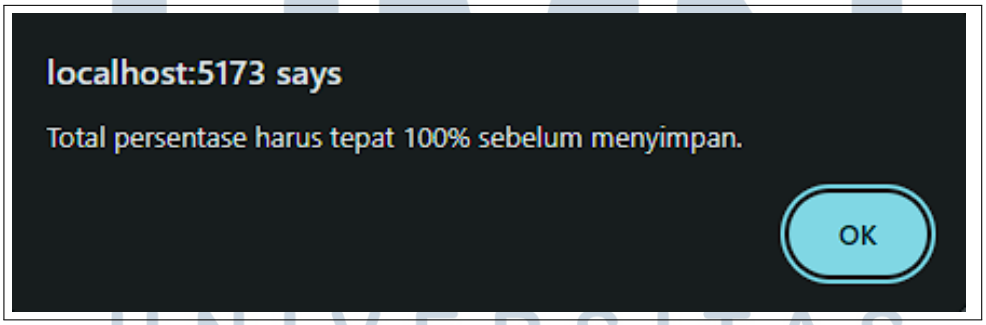
Sebagai bentuk validasi input, sistem dirancang untuk memastikan bahwa seluruh data yang dimasukkan telah lengkap dan sesuai sebelum disimpan. Salah

satu validasi yang diterapkan adalah ketika pengguna mencoba menyimpan data tanpa mengisi seluruh kolom "% Baru" pada tabel. Jika ada satu atau lebih baris yang belum diisi dengan nilai persentase, maka akan muncul peringatan seperti pada Gambar 3.32



Gambar 3.32. Alert peringatan saat kolom persentase belum diisi sepenuhnya

Selain itu, sistem juga melakukan pengecekan apakah jumlah total seluruh persentase distribusi SHU yang dimasukkan telah mencapai 100%. Jika total persentase belum tepat 100%, maka sistem akan menampilkan peringatan yang serupa, untuk mencegah ketidakseimbangan distribusi SHU koperasi. Peringatan ini bertujuan memastikan keadilan dan validitas dalam pembagian hasil usaha kepada seluruh anggota koperasi.



Gambar 3.33. Alert saat total persentase SHU belum mencapai 100%

Dengan adanya fitur validasi ini, pengurus didorong untuk melakukan input data secara teliti agar hasil pembagian SHU yang tercatat benar-benar akurat dan proporsional sesuai kontribusi masing-masing anggota.

D.3 Logika Backend Sisa Hasil Usaha

Pada proses distribusi Sisa Hasil Usaha (SHU), sistem dirancang untuk memastikan bahwa seluruh input persentase dari setiap anggota koperasi telah memenuhi syarat pembagian yang adil dan seimbang. Salah satu validasi penting yang diterapkan adalah memastikan bahwa total seluruh persentase yang dimasukkan pengguna tepat 100%. Validasi ini bersifat mutlak karena ketidakseimbangan dalam total persentase akan menyebabkan nilai pembagian SHU menjadi tidak konsisten dan dapat menimbulkan kesalahan akuntansi.

Sebelum proses penyimpanan dilakukan, sistem akan menghitung total seluruh persentase yang dimasukkan oleh pengguna melalui form input. Jika total persentase tersebut tidak mencapai 100%, maka sistem akan menolak permintaan penyimpanan dan memberikan respons berupa pesan kesalahan.

```
const totalPersentase = distribusi.reduce((sum, d) => sum +
    parseFloat(d.percentage || 0), 0);

  if (Math.abs(totalPersentase - 100) > 0.01) {
    return res.status(400).json({ message: "Total persentase harus
    100%" });
  }
```

Kode 3.11: Kode Logika Validasi Persentase SHU

Setelah validasi berhasil dilewati, proses dilanjutkan dengan penyimpanan atau pembaruan data distribusi SHU ke dalam basis data. Untuk menjaga efisiensi dan konsistensi data, digunakan pendekatan *findOrCreate*, yang memungkinkan sistem melakukan pengecekan terlebih dahulu apakah entri distribusi untuk anggota tertentu dan tahun tertentu sudah tersedia. Jika belum ada, maka sistem akan membuat entri baru; namun jika entri sudah tersedia, maka sistem akan memperbarui data yang lama. Pendekatan ini mendukung fleksibilitas pengguna dalam memperbarui distribusi secara manual tanpa menciptakan duplikasi data.

```
const [shuRecord, created] = await SHU.findOrCreate({
          where: { user_id: d.userId, tahun: year },
          defaults: {
            persentase: d.percentage,
            nominal_shu: (d.percentage / 100) * totalSHU,
            total_simpanan: d.totalSimpanan,
            total_pinjaman: d.totalPinjaman,
            selisih: d.selisih,
            is_deleted: false,
```

```
        },
      });

      if (!created) {
        await shuRecord.update({
          persentase: d.percentage,
          nominal_shu: (d.percentage / 100) * totalSHU,
          total_simpanan: d.totalSimpanan,
          total_pinjaman: d.totalPinjaman,
          selisih: d.selisih,
        });
      }
```

Kode 3.12: Kode Logika *findOrCreate* untuk data entri SHU

Dengan dua blok kode tersebut, sistem distribusi SHU menjadi lebih andal dan terhindar dari kesalahan input yang dapat mempengaruhi perhitungan laporan akhir. Selain itu, pendekatan ini juga mendukung fleksibilitas pengurus koperasi dalam melakukan revisi alokasi SHU tanpa harus menghapus data yang sudah ada secara manual.

3.3.3 Pengembangan Lainnya

A Perbaikan Sistematika Penyimpanan Gambar

A.1 Latar Belakang Permasalahan

Pada sistem awal, proses penyimpanan gambar dilakukan dengan cara menyimpan data dalam format *base64* langsung ke dalam kolom *LOB* pada tabel *TR_LOB_BERITA*. Meskipun gambar dapat tersimpan dan dikembalikan, pendekatan ini menimbulkan sejumlah kendala teknis dalam implementasi jangka panjang. Salah satu permasalahan utama muncul saat gambar yang tersimpan ingin ditampilkan kembali di antarmuka pengguna (*frontend*). Karena *data base64* memiliki ukuran string yang jauh lebih besar dari *file* aslinya, proses *decoding* dan pengiriman data menjadi sangat berat. Hal ini mengakibatkan sistem mengalami *internal server error* (500), terutama saat menangani gambar berukuran besar atau ketika terjadi pemrosesan paralel.

Selain dari sisi ukuran, format *base64* yang disimpan tanpa penanganan yang tepat juga dapat menyebabkan data tidak dikenali oleh *browser*. Hal ini terjadi ketika data *base64* tidak dilengkapi dengan *prefix MIME type* yang sesuai, atau saat

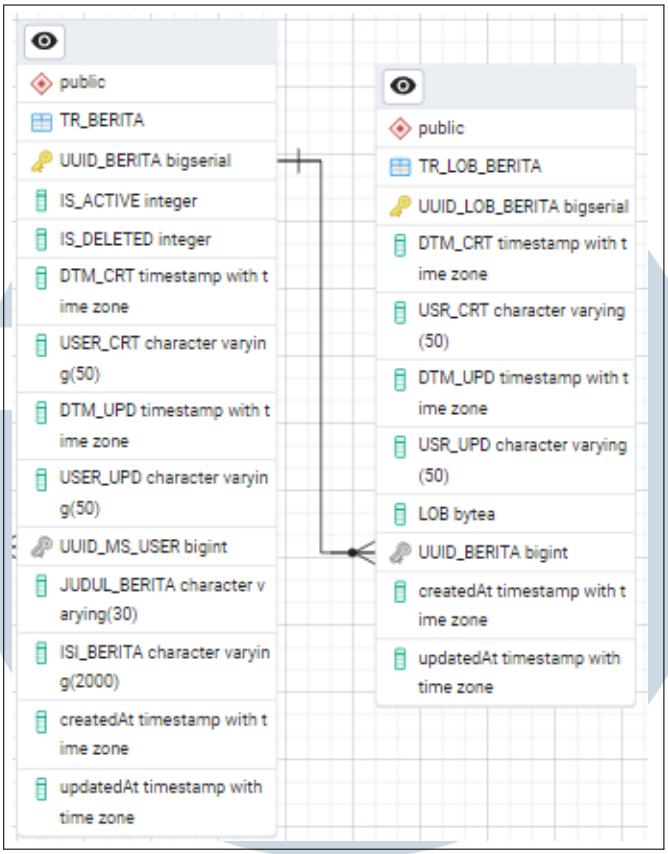
terjadi kesalahan struktur encoding, yang menyebabkan gambar gagal ditampilkan di halaman web. Masalah ini menunjukkan bahwa format penyimpanan *base64* secara langsung tidak ideal untuk sistem yang mengandalkan pengambilan dan penyajian data visual secara dinamis.

A.2 Solusi dan Perubahan Implementasi

Sebagai solusi terhadap kendala sebelumnya, sistem diperbarui dengan pendekatan baru dalam menangani data gambar. Perubahan ini dilakukan dengan cara menyimpan gambar yang sebelumnya berbentuk *base64* menjadi bentuk *buffer binary*. Pada saat pengguna mengunggah berita melalui antarmuka frontend, sistem *backend* akan melakukan proses konversi *string base64* menjadi objek *buffer* menggunakan fungsi *"Buffer.from(...)"*. Hasil buffer inilah yang kemudian disimpan ke kolom *LOB* pada tabel *TR_LOB_BERITA*. Dengan penyimpanan dalam bentuk *binary*, sistem menjadi lebih efisien dan stabil dalam menangani data gambar, serta menghindari masalah ukuran *payload* yang besar atau error saat *decode* di *frontend*.

Perubahan serupa juga diterapkan pada proses pembaruan data berita. Jika pengguna mengedit berita dan mengunggah gambar baru, sistem akan melakukan validasi dan menggantikan data buffer gambar sebelumnya di tabel *TR_LOB_BERITA*. Relasi antara data utama dan gambar dijaga dengan penggunaan *foreign* key *UUID_BERITA* yang menghubungkan tabel *TR_BERITA* dan *TR_LOB_BERITA*. Dengan mekanisme ini, struktur data tetap terorganisir dan relasi antar informasi dapat ditelusuri secara logis.

Struktur relasional antar tabel yang menunjang penyimpanan berita dan gambar secara terpisah dapat dilihat pada Gambar 3.34.



Gambar 3.34. ERD Relasi Tabel *TR_BERITA* dan *TR_LOB_BERITA*

A.3 Mekanisme Penyimpanan dan Penampilan Gambar

Setelah sistem dikembangkan untuk menyimpan gambar dalam bentuk *binary* (*buffer*), diperlukan alur khusus agar data gambar tersebut dapat digunakan kembali dan ditampilkan pada antarmuka pengguna. Ketika pengguna mengunggah gambar dari *frontend*, data gambar yang dikirim umumnya berupa format *base64*. Data ini kemudian diproses oleh backend untuk dikonversi menjadi format *buffer* menggunakan *"Buffer.from(...)"*, lalu disimpan ke dalam kolom *LOB* pada tabel *TR_LOB_BERITA*. Proses ini ditunjukkan pada potongan kode 3.13.

```
// Konversi base64 ke Buffer
        const base64Data = fotoBerita.replace(/^data:image\/[a-zA-Z]+;base64,/, '');
        const binaryData = Buffer.from(base64Data, 'base64');
```

Kode 3.13: Kode Konversi *Base64* ke *Buffer* di *Backend*

Ketika data berita ditampilkan kembali, sistem harus mengubah data gambar dari bentuk *buffer* menjadi *string base64* agar dapat dirender oleh *browser*. Di

sisi *frontend*, dilakukan proses pengecekan tipe data dan konversi *buffer* ke *string base64*, kemudian ditambahkan *prefix MIME type* (*data:image/jpeg;base64* atau *data:image/png;base64*) agar gambar dapat dikenali oleh HTML sebagai elemen ¡img¿. Ilustrasi alur kerja proses ini dapat dilihat pada Gambar 3.14.

```
// Jika LOB adalah Buffer, konversi ke base64
    let lobString;
    if (lobBerita.LOB.type === "Buffer" && Array.isArray(lobBerita
    .LOB.data)) {
        lobString = Buffer.from(lobBerita.LOB.data).toString('
    base64');
    } else if (typeof lobBerita.LOB === "string") {
        lobString = lobBerita.LOB;
    } else {
        return 'https://i.pinimg.com/736x/ea/f9/66/
    eaf966d05a3fa1ccf3669b6153b98528.jpg'; // URL gambar default
    }

    // Periksa apakah string base64 adalah gambar
    if (lobString.startsWith("/9j/")) {
        return `${lobString}`;
    } else if (lobString.startsWith("iVBORw0KGgo")) {
        return `${lobString}`;
    } else if (lobString.startsWith("data:image/")) {
        return lobString;
    }

    return 'https://i.pinimg.com/736x/ea/f9/66/
    eaf966d05a3fa1ccf3669b6153b98528.jpg'; // URL gambar default
```

Kode 3.14: Kode Konversi *Buffer* ke *Base64* di *Frontend*

Jika data gambar tidak tersedia atau tidak valid, sistem akan secara otomatis menggunakan gambar *default* sebagai *fallback* agar tampilan tidak terganggu. Dengan pendekatan ini, proses penyimpanan dan penampilan gambar menjadi lebih efisien, stabil, serta mencegah *error internal* seperti yang terjadi pada implementasi sebelumnya. Selain itu, penggunaan metode konversi ini juga memastikan bahwa semua gambar tampil secara konsisten di berbagai browser dan perangkat pengguna.

B Perbaikan Fitur Account Management

B.1 Latar Belakang Permasalahan

Pada saat proses pembaruan data akun pengguna melalui halaman *Account Management*, ditemukan kendala ketika sistem mencoba mengirimkan nilai peran (*role*) dalam bentuk *string* seperti ”PENGURUS” atau ”ADMIN”. Permasalahan muncul karena sistem *backend* secara teknis membutuhkan nilai peran dalam format numerik yang merujuk pada ID dari tabel *MS_JOB*, bukan dalam bentuk *string*. Akibat ketidaksesuaian *format* tersebut, *request* yang dikirim dari *frontend* ditolak oleh *backend* karena tidak sesuai dengan tipe data yang diharapkan. Kesalahan ini menyebabkan data akun tidak dapat diperbarui, dan sistem mengembalikan *error “Invalid role specified”*.

B.2 Solusi dan Implementasi Pemetaan Role

Untuk mengatasi permasalahan tersebut, dikembangkan beberapa fungsi pemetaan yang berfungsi untuk mengonversi nama role (dalam bentuk string) menjadi nilai ID numerik yang sesuai, serta sebaliknya. Tujuannya adalah untuk menjembatani perbedaan format data antara frontend dan backend agar proses update akun tetap dapat dilakukan dengan lancar dan akurat. Pertama, dibuat sebuah objek pemetaan bernama roleMapping yang mendefinisikan hubungan antara nama peran dan nilai ID-nya. Objek ini menjadi referensi utama saat melakukan konversi nama ke ID maupun sebaliknya.

```
// Helper function to map role IDs to names and vice versa
const roleMapping = {
  ANGGOTA: 1,
  PENGURUS: 2,
  ADMIN: 3
};
```

Kode 3.15: Potongan kode objek pemetaan antara nama peran dan ID numerik

Selanjutnya, dibuat fungsi *resolveJobValue* yang berfungsi untuk mendeteksi apakah nilai yang dikirim berupa *string* (nama *role*) atau sudah dalam bentuk numerik. Jika berupa *string*, maka fungsi ini akan secara otomatis dikonversikan menggunakan referensi dari *roleMapping*.

```
// Helper function to determine if the job is a string (role name)
    or already an ID
```

```
  const resolveJobValue = (jobValue) => {
    // If the job value is a string that matches a role name, map it to its ID
    if (typeof jobValue === 'string' && roleMapping[jobValue.toUpperCase()] !== undefined) {
      return roleMapping[jobValue.toUpperCase()];
    }
    // Otherwise, just return the value (could be numeric already)
    return jobValue;
  };
```

Kode 3.16: Potongan kode fungsi *resolveJobValue* untuk mengonversi *string role* menjadi ID

Selain itu, dibuat pula dua fungsi utilitas tambahan yaitu *mapRoleToId* dan *mapIdToRole* untuk memetakan nilai role secara dua arah. Fungsi *mapRoleToId* digunakan untuk memastikan bahwa setiap input *role* dari *frontend* bisa dikonversi ke bentuk numerik, sementara *mapIdToRole* digunakan untuk menampilkan kembali nama peran berdasarkan ID yang tersimpan di *database*.

```
// Role mapping utility function
const mapRoleToId = (jobCode) => {
    // Convert job code to uppercase to ensure case-insensitive matching
    const role = typeof jobCode === 'string' ? jobCode.toUpperCase() : jobCode;

    // Map role names to their corresponding IDs
    switch (role) {
        case 'ANGGOTA':
            return 1;
        case 'PENGURUS':
            return 2;
        case 'ADMIN':
            return 3;
        default:
            // If it's already a number, return it as is
            return isNaN(parseInt(jobCode)) ? null : parseInt(jobCode);
    }
};

// Reverse mapping function (useful for displaying role names)
const mapIdToRole = (roleId) => {
```

```
    switch (parseInt(roleId)) {
        case 1:
            return 'ANGGOTA';
        case 2:
            return 'PENGURUS';
        case 3:
            return 'ADMIN';
        default:
            return 'UNKNOWN';
    }
};
```

Kode 3.17: Fungsi konversi dua arah antara ID dan nama peran

B.3 Penanganan Update Role Pengguna

Fungsi *updateUser* pada *backend* diperbarui dengan menambahkan logika pemetaan role sebelum menyimpan data ke database. Nilai role yang dikirim dari frontend akan dicek terlebih dahulu, lalu dikonversi ke dalam bentuk ID numerik menggunakan fungsi *mapRoleToId*. Jika konversi tidak berhasil (misalnya *string* tidak sesuai dengan role yang ada), maka sistem akan menghentikan proses dan mengembalikan *error* agar tidak terjadi kesalahan penyimpanan data.

Setelah berhasil dikonversi ke ID yang sesuai, nilai *role* ini akan disimpan ke dalam kolom *UUID_MS_JOB* pada tabel *MS_USER*, memastikan bahwa peran pengguna telah tercatat dengan format yang tepat. Pendekatan ini memungkinkan *frontend* untuk tetap fleksibel dalam mengirim data peran, sembari menjaga integritas data di sisi *backend*.

```
// Map the job code to its corresponding ID if needed
        const mappedJobId = mapRoleToId(UUID_MS_JOB);

        if (mappedJobId === null) {
            return res.status(400).json({ message: "Invalid role specified" });
        }

        // Prepare the update data
        let updateData = {
            ...data,
            UUID_MS_JOB: mappedJobId
```

```
12          };
```

Kode 3.18: Potongan kode fungsi *updateUser* dengan logika pemetaan dan validasi *role*

Dengan perubahan ini, proses pembaruan akun menjadi lebih stabil dan mencegah kesalahan akibat format data yang tidak sesuai. Solusi ini juga memperkuat integrasi antar sistem, terutama dalam memastikan konsistensi antara data *frontend* dan *backend* untuk bagian hak akses pengguna.

3.4 Kendala yang Dihadapi

Selama pelaksanaan magang, ditemukan berbagai kendala teknis dan non-teknis yang cukup signifikan dalam pengembangan sistem, di antaranya:

1. **Kesalahan Format Role pada Manajemen Akun**
   Sistem *frontend* mengirimkan peran pengguna (*role*) dalam bentuk *string* (seperti ”ADMIN”, ”PENGURUS”), sementara *backend* mengharuskan format numerik (ID).

2. **Masalah Penyimpanan Gambar (Base64)**
   Gambar disimpan dalam format *base64* di database, menyebabkan ukuran data menjadi terlalu besar, memperlambat loading, dan berpotensi menyebabkan *internal server error* (500).

3. **Distribusi SHU Ganda**
   Potensi entri ganda dalam distribusi SHU ketika data baru dimasukkan tanpa pengecekan terhadap data lama.

4. **Token Autentikasi Kadaluarsa**
   Jika token akses pengguna telah kadaluarsa, sistem tidak dapat melanjutkan request, sehingga mengganggu proses pengguna.

5. **Kurangnya Validasi Awal pada Input Persentase SHU**
   Tidak ada sistem validasi untuk memastikan bahwa total persentase SHU yang dimasukkan pengguna berjumlah tepat 100%.

6. **Belum Tersedianya Riwayat Perubahan Transaksi**
   Pada awalnya, tidak ada pencatatan perubahan jika suatu transaksi manual diedit, sehingga menyulitkan pelacakan audit.

7. **Minimnya Dokumentasi Otomatis Laporan Bulanan**
   Laporan keuangan sebelumnya tidak otomatis terdokumentasi saat diakses atau diekspor, menyebabkan kehilangan data historis.

8. **Kesulitan Memahami Alur Sistem dan Struktur Data**
   Diperlukan waktu untuk memahami alur sistem, relasi data, dan cara kerja modul yang telah dibangun oleh tim sebelumnya.

9. **Tantangan dalam Penggunaan Tools Baru**
   Penggunaan tools baru seperti *Sequelize* atau *REST Client* sempat menjadi tantangan karena belum pernah digunakan sebelumnya.

10. **Pengambilan Keputusan Secara Mandiri**
    Sebagian besar pengembangan dilakukan secara individual, sehingga diperlukan kemampuan untuk mengambil keputusan teknis sendiri.

3.5 Solusi yang Diterapkan

Untuk mengatasi kendala-kendala tersebut, dilakukan beberapa langkah perbaikan dan penyesuaian, antara lain:

1. **Pemrosesan Role Otomatis**
   Dibuat fungsi pemetaan (*roleMapping*) dan logika deteksi otomatis (*resolveJobValue*) agar string role dapat dikonversi ke ID sebelum dikirim ke *backend*.

2. **Optimalisasi Penyimpanan Gambar**
   Format penyimpanan diubah menjadi *buffer binary* menggunakan *Buffer.from(...)*, sehingga lebih efisien dan stabil. Validasi dan *fallback image* juga ditambahkan untuk menghindari tampilan kosong jika terjadi kesalahan.

3. **Validasi Distribusi SHU**
   Implementasi metode *findOrCreate* dan *update* untuk memeriksa dan memperbarui entri SHU, sehingga sistem hanya menyimpan satu data per pengguna per tahun.

4. **Refresh Token Otomatis**
   Diterapkan *axiosInterceptor* yang secara otomatis mendeteksi token kadaluarsa dan melakukan refresh token tanpa perlu login ulang manual.

5. **Validasi Total Persentase SHU**
   Ditambahkan logika validasi *frontend* dan *backend* untuk menampilkan *alert warning* jika total belum 100%, mencegah proses distribusi dijalankan sebelum data valid.

6. **Fitur Riwayat Edit Transaksi**
   Dibuat tabel *TR_MANUAL_TRANSACTIONS_HISTORY* dan fitur modal "History Edit" untuk merekam dan menampilkan riwayat edit transaksi secara rinci.

7. **Pencatatan Otomatis Laporan Bulanan**
   Sistem kini menyimpan setiap hasil ekspor ke tabel *TR_MONTHLY_FINANCE_STATEMENT* sebagai arsip digital permanen.

8. **Pembelajaran Mandiri Terhadap Alur Sistem**
   Melakukan pembelajaran mandiri terhadap *source code*, diskusi dengan rekan tim, serta mengamati langsung alur data dan fitur melalui antarmuka.

9. **Eksplorasi dan Pencatatan Solusi Teknis**
   Belajar melalui dokumentasi resmi, tutorial, dan eksperimen langsung hingga terbiasa, sambil mencatat solusi dari setiap masalah teknis yang ditemui.

10. **Penguatan Inisiatif dan Evaluasi Mandiri**
    Mengasah inisiatif dan tanggung jawab dengan menetapkan prioritas pekerjaan, mengevaluasi solusi yang diambil, dan mencatat prosesnya secara terstruktur.