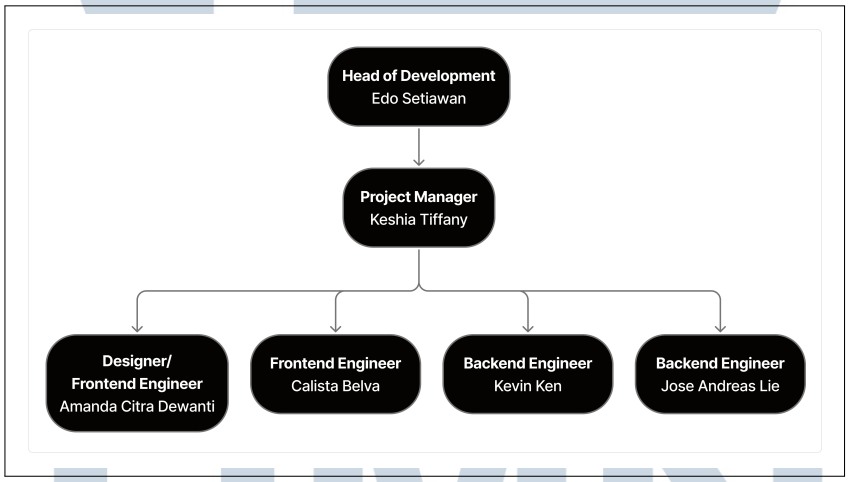
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang di PT Ganda Visi Jayatama, peran yang dijalankan berada dalam tim pengembangan sistem CHRIS (Concise Human Resources Information System) sebagai *Backend Engineer Intern*. Tanggung jawab utamanya mencakup pengembangan dan pengujian *Application Programming Interface* (API) yang digunakan oleh sistem, serta kolaborasi dengan anggota tim lainnya dalam penyusunan dan penyempurnaan fitur-fitur sistem kepegawaian berbasis *web*.



Gambar 3.1. Struktur Tim CHRIS

Gambar 3.1 merupakan struktur dari tim CHRIS yang terdiri dari sejumlah anggota dengan peran yang saling terintegrasi. Bimbingan diberikan oleh Bapak Edo Setiawan selaku *Supervisor* sekaligus Head of Development, yang secara rutin melaksanakan evaluasi mingguan terhadap progres dan melakukan *code review* atas hasil pengembangan *backend*. Koordinasi teknis lebih lanjut dilaksanakan bersama Bapak Muhammad Alwin Alamsyah Handoko Putra selaku *Backend Lead*, yang memimpin diskusi internal tim *backend* setiap hari Jumat melalui *Backend Internal Meeting*. Perencanaan serta distribusi tugas dikoordinasikan oleh *Project Manager*, Ibu Keshia Tiffany, yang bertanggung jawab dalam pembagian *backlog* kepada anggota tim, serta mengadakan sesi evaluasi pribadi (*one-on-one*) dengan masing-masing anggota tim magang.

Dalam pengembangan tampilan antarmuka sistem, kolaborasi dilakukan bersama *Designer* Amanda Citra Dewanti yang merancang desain akhir dari *web*, serta dua *Frontend Engineer Intern*, Amanda Citra Dewanti dan Calista Belva, yang membangun antarmuka *web* menggunakan React. Sementara itu, pengembangan API menggunakan Express.js dan Node.js, serta pengelolaan basis data dengan PostgreSQL, dijalankan oleh dua Backend Engineer, yaitu Kevin Ken dan satu rekan lainnya dalam tim.

Seluruh kegiatan kerja magang dilakukan secara langsung di kantor (*Work From Office*). Koordinasi dilakukan melalui *daily standup* setiap pagi untuk melaporkan progres harian, menyampaikan rencana kerja, serta mendiskusikan kendala yang dihadapi. Setiap dua minggu sekali, tim juga melaksanakan *sprint retrospective* untuk mengevaluasi hasil kerja dalam satu *sprint* dan menentukan perbaikan serta target *sprint* berikutnya. Penugasan proyek dikelola menggunakan *platform* Jira (Atlassian) dalam bentuk *backlog sprint* yang dibagikan kepada setiap anggota tim secara terstruktur dan terukur.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di PT Ganda Visi Jayatama, terdapat tanggung jawab utama dalam satu proyek utama, yaitu pengembangan aplikasi *Internal System*. Tugas-tugas yang dijalankan selama magang terbagi ke dalam beberapa aktivitas utama sebagai berikut:

1. Mengembangkan API untuk kebutuhan aplikasi *Internal System*, yang mencakup pembuatan fitur-fitur backend sesuai dengan spesifikasi fungsional.
2. Melakukan dokumentasi terhadap API yang telah dikembangkan menggunakan *platform* dokumentasi API, Apidog.
3. Melakukan pengujian secara mandiri terhadap API yang dibuat untuk memastikan bahwa seluruh *endpoint* berjalan sesuai dengan fungsinya, serta menangani *error handling* dan validasi data.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mempelajari boilerplate backend dan mulai mengembangkan API untuk Employee Status pada sistem CHRIS.
2	Melanjutkan pengembangan dan penyempurnaan API Employee Status serta melakukan validasi ulang pada form Employee di sistem CHRIS.
3	Melakukan revisi minor pada API employee form, menyelesaikan tabel User dan Employee Status, serta berpartisipasi dalam Sprint Retro.
4	Mengembangkan fitur pagination untuk berbagai modul (User, Leave, Attendance), membuat API form pengajuan cuti, serta melakukan code review dan diskusi dalam monthly meeting.
5	Fokus pada revisi dan pengembangan API perizinan cuti, integrasi dengan frontend, serta showcase sistem CHRIS dan implementasi pagination untuk Leave Types.
6	Melakukan revisi dan filtering pada Leave Permit Dashboard, menambahkan fitur cancel, serta aktif dalam code review dan weekly meeting tim backend.
7	Melakukan berbagai pengujian dan UAT untuk Leave Management, membangun sistem tree berbasis jabatan untuk izin, serta menangani revisi migrasi dan API CHRISM (CHRIS Mobile).
8	Mengembangkan sistem hierarki supervisi berbasis tree, menerapkan biometrik pada login API, dan mulai membangun user report summary API serta mempersiapkan People Report.
9	Fokus pada penyempurnaan fitur User Report, termasuk penambahan filter tanggal dan perbaikan minor, serta melakukan hashing biometrik dan refactor pada dashboard Leave Permit.
10	Memulai riset intensif terkait sistem Payroll dan skema tabelnya, membuat dokumentasi di Apidog.

Lanjut pada halaman berikutnya

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

Minggu Ke -	Pekerjaan yang dilakukan
11	Melanjutkan pengembangan API Payroll berdasarkan hasil riset skema tabel, memperbarui dokumentasi di Apidog, serta mengikuti kegiatan Backlog Planning dan Sprint Closing.
12	Fokus pada pengembangan lanjutan API Payroll termasuk fitur Create, Get, Update, dan Delete, serta mulai menangani logika data untuk User Allowances.
13	Melanjutkan secara intensif pengembangan API Payroll khusus untuk pengelolaan dan perhitungan Each User Allowances secara berkelanjutan sepanjang minggu.
14	Mulai mengembangkan dan menyempurnakan Salary Slip APIs serta melakukan bugfix dan refactor pada User Allowance dan konfigurasi Payroll untuk integrasi dengan frontend.
15	Menambahkan fitur penghapusan User Allowance, memperbaiki konfigurasi endpoint Payroll, dan membuat API gabungan untuk manajemen detail user, payroll, serta tunjangan.
16	Melanjutkan integrasi Salary Slip dengan frontend serta melakukan pengujian menyeluruh terhadap modul Payroll, Allowance, dan Salary Slip.
17	Melakukan perbaikan pada logika dan pagination Salary Slip serta Payroll Config, merevisi sistem, dan menyiapkan internal report serta showcase Payroll.

3.4 Pengumpulan dan Analisis Kebutuhan

Kebutuhan sistem dalam proyek ini diperoleh melalui koordinasi langsung dengan *supervisor* dan tim *backend internal*. Sebagian besar *requirement* ditentukan secara iteratif berdasarkan kebutuhan bisnis dan sprint mingguan yang telah direncanakan oleh tim. Proses pengumpulan *requirement* dilakukan melalui diskusi teknis, *retrospective meeting*, dan *task assignment* harian.

Berikut ini adalah uraian *requirement* utama yang berhasil diidentifikasi dan diimplementasikan dalam proyek selama masa kerja praktik.

A Refaktor User Management dan Validasi Data

Pengembangan dimulai dengan perbaikan sistem **User Management**, termasuk validasi *form input* dan *refactor* struktur tabel seperti *user* dan *employment status*. Hal ini bertujuan untuk memastikan integritas data pengguna dan kemudahan pengelolaan melalui *backend* maupun *frontend*.

B Optimalisasi Leave Permit

Modul **Leave Permit** dikembangkan agar lebih efisien dan intuitif. Perubahan meliputi *refactor* pada proses *form submission*, penambahan tombol pembatalan (*cancel*) pengajuan cuti, serta tampilan daftar cuti untuk atasan. Fitur-fitur ini dirancang agar mencerminkan alur persetujuan yang realistis dan terstruktur.

C Implementasi Pagination

Untuk mendukung jumlah data yang besar, sistem pagination ditambahkan pada beberapa modul utama seperti **User**, **Leave**, dan **Attendance**. Hal ini dilakukan guna menjaga performa dan kenyamanan pengguna.

D Penambahan Fitur Biometrik untuk CHRIS Mobile

Fitur biometrik ditambahkan untuk mendukung proses autentikasi pada sistem CHRIS Mobile (CHRISM). Pengguna dapat melakukan login menggunakan data biometrik seperti sidik jari yang telah di-hash dan disimpan dalam kolom khusus pada tabel *users*. Fitur ini ditujukan untuk meningkatkan keamanan serta kenyamanan akses pengguna terhadap sistem.

E Pengembangan Sistem Hierarki

Dibuat fungsi *tree hierarchy* berdasarkan struktur jabatan untuk mendukung fitur-fitur seperti izin cuti (*accept*/*reject*) dan tampilan dashboard atasan. Fungsi ini menjadi dasar logika akses dan pengelolaan hubungan antar pegawai.

F Modul Payroll

Modul **Payroll** dikembangkan untuk menghasilkan slip gaji setiap bulannya yang dihitung berdasarkan tunjangan. Termasuk di dalamnya pengembangan API untuk CRUD *data payroll*, penyusunan *salary slip*, dan integrasi dengan *frontend*.

3.5 Perancangan dan Pengembangan Sistem

Bagian ini menjelaskan perancangan dan pengembangan sistem yang mencakup struktur basis data dan alur sistem untuk fitur-fitur yang dikembangkan selama masa kerja praktik.

3.5.1 *User Management* dan Validasi Data

Sistem CHRIS telah dilengkapi dengan modul *User Management* yang berfungsi untuk mengelola data pegawai secara efisien. Modul ini mencakup fitur untuk menambahkan, memperbarui, dan menghapus data pegawai, serta melakukan validasi terhadap input yang dimasukkan melalui formulir. Validasi ini bertujuan untuk memastikan bahwa data yang diterima sesuai dengan format dan ketentuan yang berlaku. Meskipun demikian, sejumlah aspek dari modul ini memerlukan penyempurnaan guna meningkatkan integritas data dan mempermudah pengelolaan sistem.

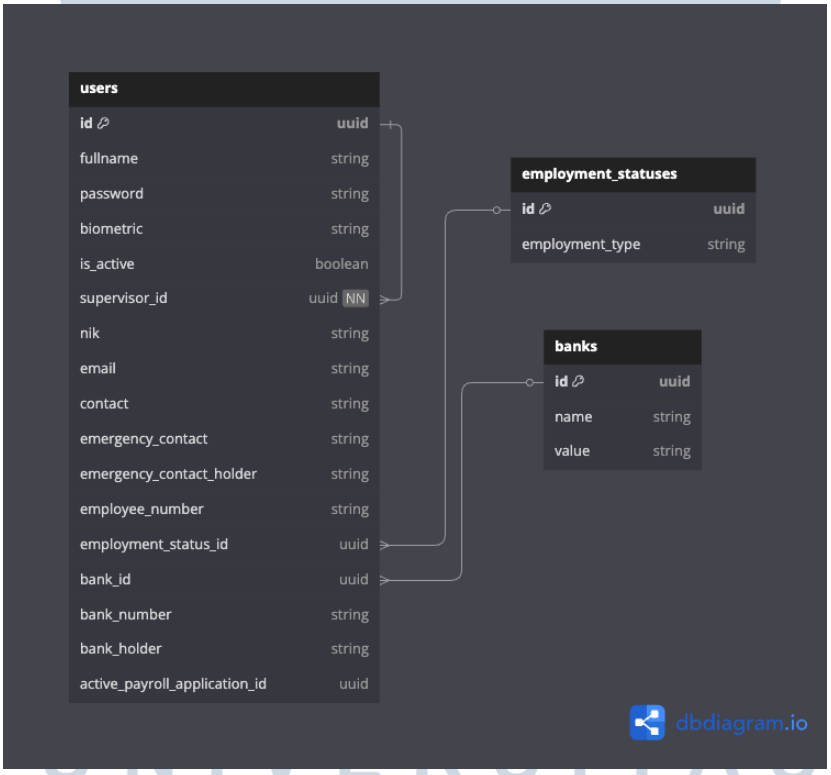
Adapun perbaikan dan pengembangan yang telah dilakukan antara lain:

- ***Refactor User Management***: Alur pengelolaan data pegawai diperbarui agar setiap entri memiliki status kepegawaian yang terdefinisi dengan baik, sehingga struktur data menjadi lebih sistematis dan mudah diakses.
- **Validasi Form Input**: Validasi terhadap form input ditingkatkan, mencakup pengecekan format email, nomor telepon, serta memastikan bidang yang wajib diisi tidak terlewat, guna mencegah terjadinya inkonsistensi data.
- **Penyempurnaan Struktur Tabel**: Struktur tabel *users* diperbarui dengan menambahkan beberapa referensi eksternal untuk meningkatkan normalisasi data, antara lain:
  - Penambahan *employment_status_id* yang mereferensikan tabel *employment_statuses*, menggantikan pendekatan enumerasi yang sebelumnya digunakan secara *hardcoded*.

– Penambahan *bank_id* yang mereferensikan tabel *banks*, menggantikan kolom nama bank dalam bentuk *string* pada tabel *users* untuk menjamin konsistensi data dan memudahkan pengelompokan informasi perbankan.

Perubahan ini dilakukan sebagai bagian dari penerapan praktik terbaik dalam pengembangan sistem backend berbasis relasional. Selain itu, modifikasi ini juga memberikan fleksibilitas lebih tinggi dalam pengelolaan data serta meningkatkan skalabilitas modul User Management dalam jangka panjang.

A Diagram ERD User Management



Gambar 3.2. Diagram ERD untuk modul User Management

Gambar 3.2 menunjukkan struktur basis data untuk modul User Management yang telah dimodifikasi. Terdapat beberapa tabel utama yang saling berhubungan, yaitu:

- **Users**: Tabel ini menyimpan data pegawai, termasuk informasi pribadi, status kepegawaian, dan referensi bank.

- **Employment Statuses**: Tabel ini menyimpan berbagai status kepegawaian yang dapat dimiliki oleh pegawai, seperti aktif, cuti, atau tidak aktif.
- **Banks**: Tabel ini menyimpan informasi mengenai bank yang digunakan oleh pegawai untuk penggajian.

Sebelumnya modul User Management menggunakan pendekatan *hardcoded* untuk status kepegawaian dan bank, namun kini telah diubah menjadi referensi tabel yang lebih fleksibel. Hal ini memungkinkan penambahan atau perubahan status kepegawaian dan bank tanpa perlu mengubah kode sumber, sehingga meningkatkan efisiensi pengelolaan data.

B Validasi Data pada Formulir User Management

Validasi data pada formulir *User Management* dilakukan untuk menjamin integritas, konsistensi, dan keamanan data yang masuk ke dalam sistem. Validasi dilakukan baik di sisi *frontend* maupun di sisi *backend*, dengan ketentuan sebagai berikut:

- ***Fullname***: Nama lengkap pegawai harus diisi dengan format yang benar, yaitu merupakan huruf *alphanumeric*. Validasi ini bertujuan untuk memastikan bahwa nama pegawai dapat dikenali dan diidentifikasi dengan jelas dalam sistem.
- ***Email***: Hanya alamat email dengan domain @*concise.co.id* yang diperbolehkan. Validasi ini diterapkan untuk memastikan bahwa hanya pegawai internal yang terdaftar di sistem. Format email juga diverifikasi menggunakan ekspresi reguler untuk menghindari entri tidak valid.
- **NIK**: Nomor Induk Kependudukan (NIK) harus diisi dengan format yang benar, yaitu terdiri dari 16 digit angka. Validasi ini penting untuk memastikan bahwa NIK yang dimasukkan sesuai dengan standar yang berlaku di Indonesia.
- ***Contact***: Hanya nomor telepon yang dimulai dengan *prefix +62* atau *0* yang diterima. Validasi ini bertujuan untuk memastikan bahwa nomor telepon yang dimasukkan sesuai dengan format nomor telepon pada umumnya.

- ***Emergency Contact***: Sama seperti nomor telepon, hanya nomor yang dimulai dengan *prefix +62* atau *0* yang diterima. Hal ini untuk memastikan bahwa kontak darurat yang dimasukkan dapat dihubungi dengan mudah.
- ***Bank Holder***: Nama pemegang rekening bank harus diisi dengan format yang benar, yaitu merupakan huruf *alphanumeric*. Validasi ini bertujuan untuk memastikan bahwa nama pemegang rekening sesuai dengan nama pegawai yang terdaftar dalam sistem.
- ***Bank Account Number***: Validasi dilakukan untuk memastikan bahwa nomor rekening bank yang dimasukkan hanya terdiri dari angka. Hal ini penting untuk menghindari kesalahan dalam proses penggajian.

Validasi ini tidak hanya berfungsi untuk memperbaiki pengalaman pengguna, tetapi juga mencegah terjadinya kesalahan logika dan duplikasi data di tingkat basis data. Seluruh ketentuan ini dirancang berdasarkan standar praktik terbaik dalam pengelolaan data karyawan di lingkungan perusahaan.

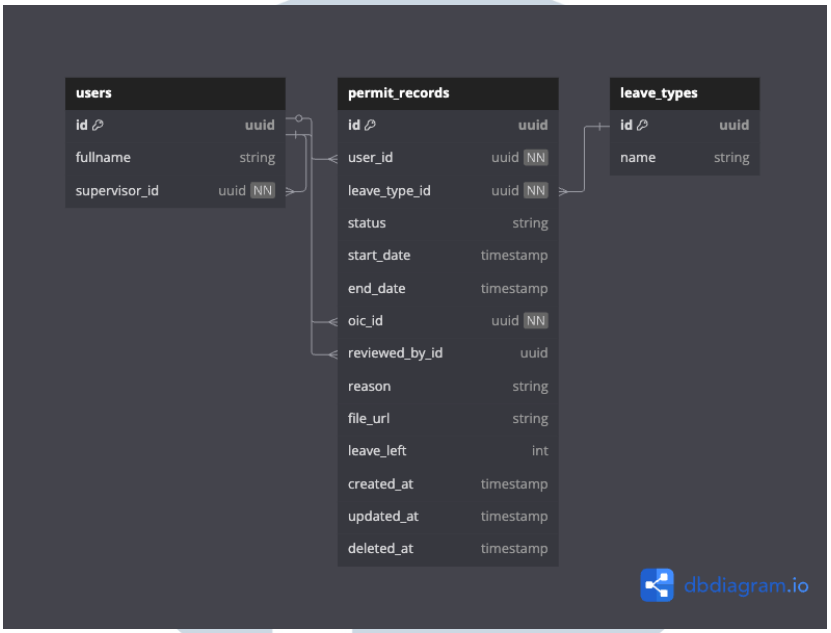
3.5.2 Leave Permit

Modul *Leave Permit* merupakan fitur yang memungkinkan pegawai mengajukan permohonan cuti, serta memberikan wewenang kepada atasan untuk menyetujui atau menolak permohonan tersebut. Setiap jenis cuti memiliki jatah tersendiri, dan modul ini juga berfungsi untuk menghitung secara otomatis sisa cuti yang dimiliki oleh masing-masing pegawai. Sistem ini dirancang untuk mencerminkan alur persetujuan yang terstruktur dan realistis, dengan memperhatikan hierarki jabatan di dalam perusahaan.

Sistem ini sudah pernah digunakan sebelumnya, namun mengalami beberapa kendala yang perlu diperbaiki. Beberapa perbaikan yang dilakukan antara lain adalah:

- **Refactor Form Submission**: Proses pengajuan cuti ditambahkan *officer in charge* (*oic*) dengan tujuan sebagai pengganti pegawai saat ia cuti.
- **Cancel Button**: Ditambahkan fitur pembatalan (*cancel*) pengajuan cuti, sehingga pegawai dapat membatalkan permohonan yang belum disetujui.
- **Leave Permit Dashboard**: Tampilan daftar cuti ditampilkan di *home page* supaya semua pegawai dapat melihat siapa saja yang mengajukan cuti di minggu itu.

A Diagram ERD Leave Permit

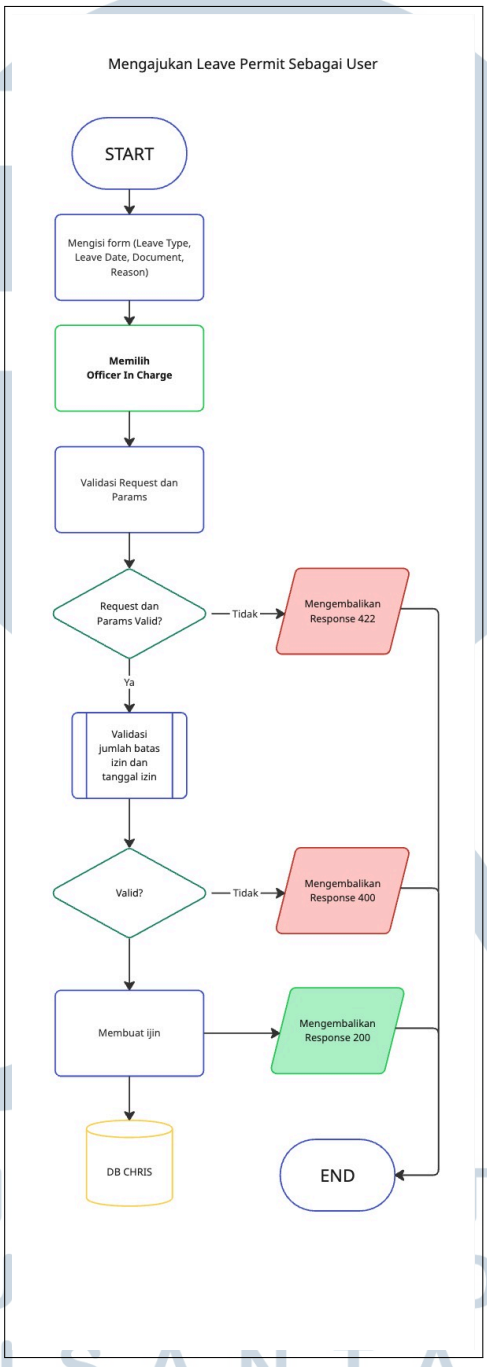


Gambar 3.3. Diagram ERD untuk modul Leave Permit

Gambar 3.3 menunjukkan struktur basis data untuk modul Leave Permit yang dimodifikasi. Terdapat beberapa tabel utama yang saling berhubungan, yaitu:

- **User**: Tabel ini menyimpan data pegawai yang mengajukan cuti, termasuk informasi pribadi dan status kepegawaian.
- **Leave Types**: Tabel ini menyimpan jenis-jenis cuti yang tersedia, termasuk nama, deskripsi, dan jatah cuti yang diberikan kepada pegawai.
- **Permit Records**: Tabel ini menyimpan data permohonan cuti yang diajukan oleh pegawai, termasuk tanggal pengajuan, tanggal mulai dan selesai cuti, status persetujuan, dan pegawai lain yang menggantikan dia sebagai *office in charge*.

B Alur Sistem Leave Permit



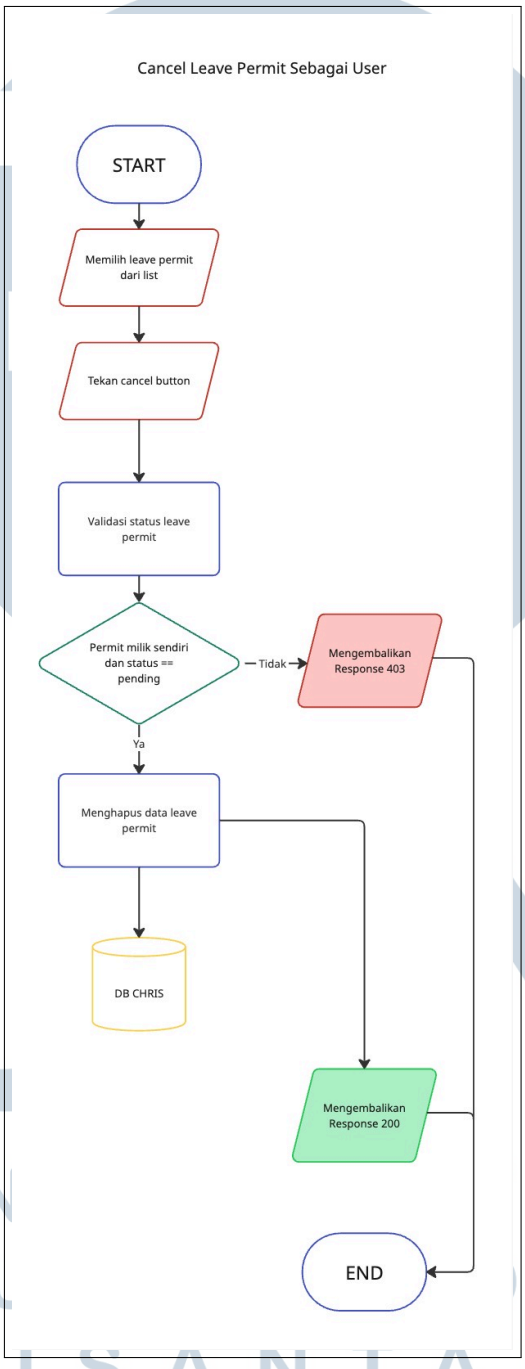
Gambar 3.4. *Flowchart* alur sistem Leave Permit

Gambar 3.4 menunjukkan alur sistem *Leave Permit* yang diawali oleh pegawai yang mengajukan cuti melalui formulir yang tersedia. Informasi yang

diisi mencakup jenis cuti, rentang tanggal, alasan pengajuan, serta penunjukan *officer in charge* sebagai pengganti selama periode cuti. Setelah permohonan dikirimkan, sistem akan menyimpan data ke dalam tabel *Permit Records*, dan atasan dapat memberikan persetujuan atau penolakan. Selama permohonan belum diproses, pegawai memiliki opsi untuk membatalkannya. Jika disetujui, sistem secara otomatis akan memperbarui sisa jatah cuti sesuai jenis cuti yang diajukan. Status pengajuan dapat dipantau melalui dashboard yang menampilkan daftar cuti yang aktif dalam minggu berjalan.

Sebagai bagian dari pengembangan lanjutan modul ini, dilakukan penyesuaian struktur data dengan menambahkan kolom *oic_id* pada tabel *Permit Records*. Penambahan ini ditujukan untuk memenuhi kebutuhan bisnis dalam menjamin kesinambungan operasional saat pegawai cuti, dengan menunjuk rekan kerja yang bertanggung jawab selama periode tersebut. Perubahan ini turut memperkuat logika bisnis sistem dan memastikan distribusi tugas tetap berjalan secara efisien.

C Alur Sistem Cancel Leave Permit



Gambar 3.5. *Flowchart* alur sistem Cancel Leave Permit

Gambar 3.5 menggambarkan alur proses pembatalan permohonan cuti oleh pegawai. Setelah pengajuan dilakukan, pegawai dapat membatalkan permohonan

selama statusnya belum disetujui oleh atasan. Permintaan pembatalan dikirim melalui formulir yang tersedia, kemudian sistem akan memverifikasi status permohonan. Jika permohonan belum disetujui, sistem akan melakukan *soft delete* pada data di tabel *Permit Records*. Namun, apabila permohonan telah disetujui, sistem akan menolak proses pembatalan dan menampilkan notifikasi kesalahan bahwa pengajuan tidak dapat dibatalkan.

D Alur Sistem Leave Permit Dashboard

Alur pengambilan dan penampilan data cuti pada halaman *dashboard* digunakan untuk menampilkan pegawai-pegawai yang sedang cuti di minggu berjalan. Proses dimulai saat pegawai mengakses dashboard dan sistem kemudian melakukan query terhadap data *permit records* yang memiliki rentang tanggal cuti berada dalam minggu berjalan dan status pengajuan sudah aktif. Data yang diambil mencakup nama pegawai, jenis cuti, tanggal mulai dan selesai, serta *officer in charge* yang ditunjuk.

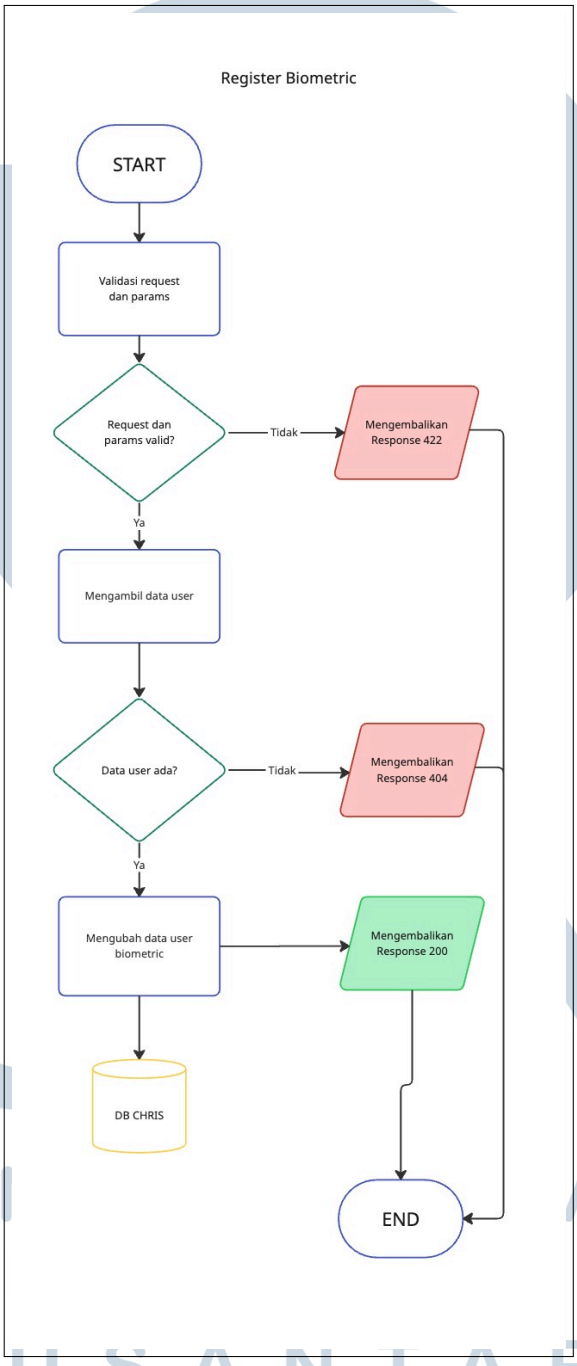
Informasi tersebut disajikan dalam bentuk tabel agar mudah dipahami dan dapat digunakan oleh pegawai untuk mengetahui siapa saja yang sedang atau akan cuti, serta mengetahui siapa rekan pengganti yang dapat dihubungi untuk keperluan operasional. Fitur ini ditujukan untuk meningkatkan transparansi dan mendukung koordinasi lintas tim selama periode cuti berlangsung.

3.5.3 Fitur Autentikasi Biometrik pada CHRIS Mobile

Sebagai bagian dari pengembangan sistem CHRIS Mobile (CHRISM), ditambahkan fitur autentikasi berbasis biometrik untuk meningkatkan kenyamanan, kecepatan dan keamanan akses pengguna. Implementasi ini dilakukan dengan menambahkan kolom baru biometric pada tabel *users*. Kolom ini menyimpan hasil *hash* sepanjang maksimal 255 karakter dari data biometrik pengguna seperti sidik jari.

Fitur ini memberikan alternatif *login* selain kata sandi serta mendukung praktik keamanan modern, termasuk *multi-factor authentication*. Karena data biometrik telah melalui proses hashing, informasi yang disimpan tetap aman dan tidak dapat digunakan kembali secara langsung. Fitur ini hanya tersedia pada aplikasi CHRIS Mobile dan tidak memengaruhi sistem versi web atau *desktop*.

A Alur Register Biometrik



Gambar 3.6. *Flowchart* alur sistem registrasi biometrik pada CHRIS Mobile

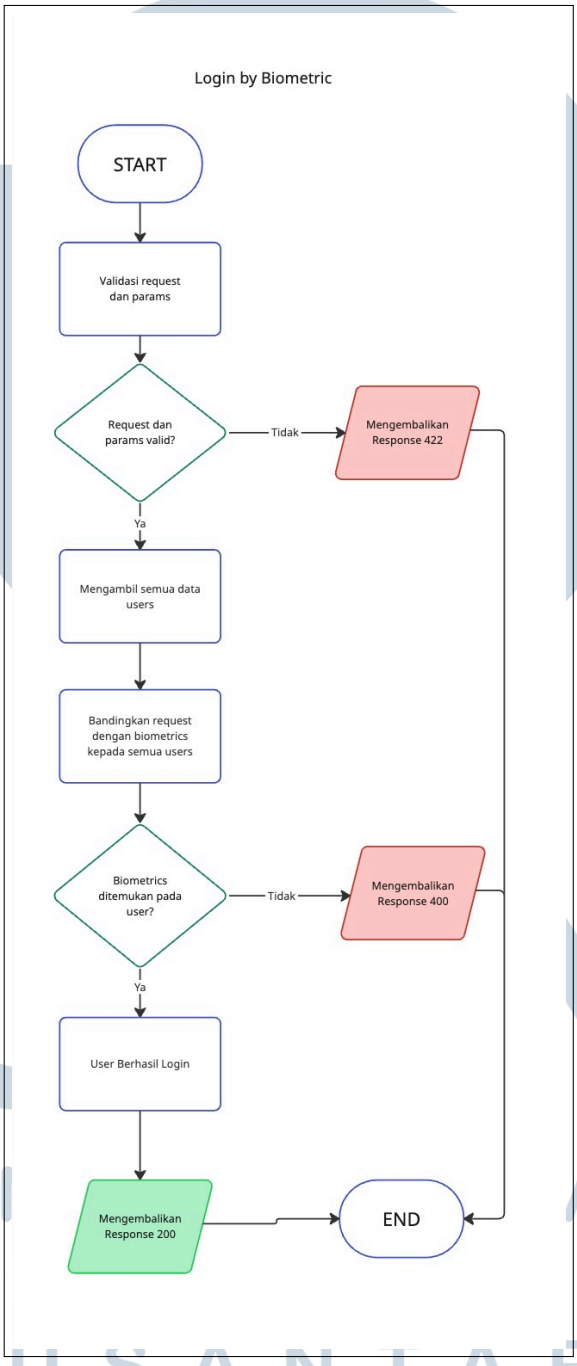
Gambar 3.6 menunjukkan alur sistem registrasi dan autentikasi biometrik pada aplikasi CHRIS Mobile (CHRISM). Proses dimulai ketika pengguna

mengakses halaman profil dan memilih opsi "*Activate Biometric*". Setelah itu, aplikasi akan memicu pemindaian biometrik menggunakan sensor sidik jari pada perangkat. Jika proses pemindaian berhasil, sistem akan secara otomatis menghasilkan *random* string sepanjang 255 karakter yang mewakili identitas biometrik pengguna. Nilai ini kemudian dikirimkan ke *backend* melalui API khusus untuk proses pendaftaran biometrik.

Di sisi *backend*, data tersebut akan di-hash dan disimpan pada kolom *biometric* di tabel *users* untuk keperluan autentikasi selanjutnya.

Dalam implementasi autentikasi, saat aplikasi dibuka, CHRISM akan kembali meminta verifikasi biometrik dari perangkat. Jika sidik jari cocok, sistem akan menembakkan *payload* berupa *random string* 255 karakter yang identik dengan yang telah didaftarkan sebelumnya, lalu mengirimkannya ke *endpoint login biometrik*. *Backend* akan mencocokkan hasil *hash* dari *string* tersebut dengan data yang tersimpan di basis data. Jika sesuai, maka autentikasi dinyatakan berhasil dan pengguna dapat langsung masuk ke sistem tanpa perlu menggunakan kata sandi. Proses ini dirancang untuk meningkatkan keamanan serta memberikan pengalaman masuk aplikasi yang lebih praktis dan efisien bagi pengguna CHRISM.

B Alur Autentikasi Login Menggunakan Biometrik



Gambar 3.7. *Flowchart* alur sistem autentikasi biometrik pada CHRIS Mobile

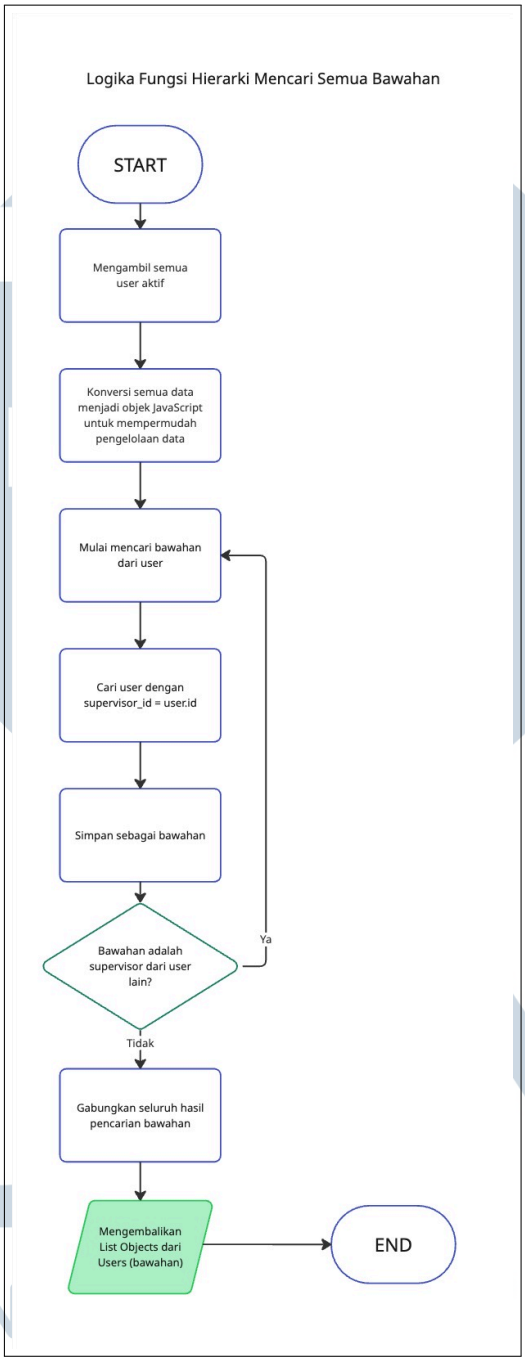
Gambar 3.7 menunjukkan alur sistem autentikasi biometrik pada CHRIS Mobile. Proses dimulai ketika pengguna memilih opsi *login* menggunakan

biometrik. Sistem kemudian akan meminta data biometrik dari perangkat, yang selanjutnya di-hash dan dibandingkan dengan data yang tersimpan di basis data. Jika cocok, pengguna akan berhasil masuk ke dalam aplikasi. Jika tidak, sistem akan menampilkan pesan kesalahan dan meminta pengguna untuk mencoba kembali. Fitur ini dirancang untuk memberikan pengalaman pengguna yang lebih cepat dan aman, serta mengurangi ketergantungan pada kata sandi yang dapat dilupakan atau dicuri.

3.5.4 Sistem Hierarki Supervisi

Sistem CHRIS menerapkan struktur hierarki berbasis pohon (*tree hierarchy*) untuk mengelola hubungan antara pegawai dan atasan. Modul-modul dalam sistem ini, seperti pengajuan cuti, bergantung pada struktur tersebut, di mana permohonan cuti hanya dapat disetujui oleh atasan langsung dari pegawai yang bersangkutan.

Diagram pada Gambar 3.8 menggambarkan alur logika sistem dalam mencari seluruh bawahan dari seorang pegawai. Fungsi ini dimulai dengan mengambil seluruh data pengguna dari basis data, kemudian melakukan pencarian rekursif terhadap pegawai yang memiliki *supervisor_id* yang sesuai dengan *id* pegawai tersebut. Pencarian dilakukan secara berlapis hingga seluruh struktur bawahan ditemukan.



Gambar 3.8. Logika Fungsi Hierarki untuk Mencari Semua Bawahan

Dengan pendekatan ini, sistem mampu menentukan siapa saja yang berada dalam rantai struktur supervisi, baik secara langsung maupun tidak langsung. Hal ini memungkinkan sistem untuk secara efisien menetapkan pihak yang berwenang dalam proses seperti persetujuan cuti, pelacakan struktur organisasi, maupun

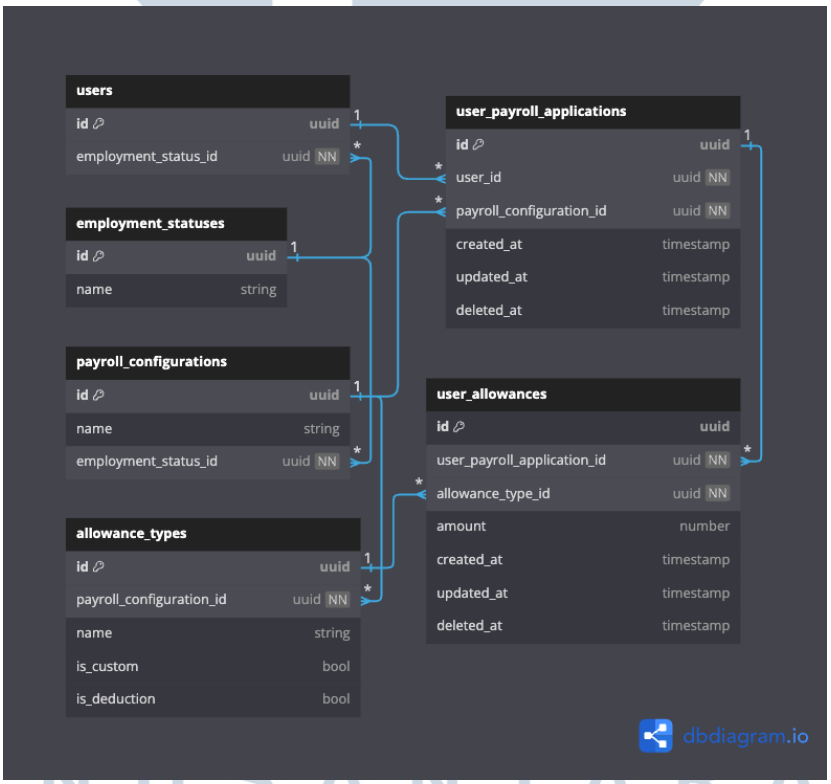
pengelolaan akses modul internal.

Fungsi ini telah digunakan secara langsung dalam modul *Leave Permit* untuk memastikan bahwa pengajuan cuti hanya dapat ditinjau dan disetujui oleh atasan yang sesuai. Selain itu, logika ini juga dapat diterapkan pada modul lain yang membutuhkan pemetaan hubungan antarpegawai secara hierarkis, seperti monitoring kinerja, delegasi tugas, atau manajemen tim lintas divisi.

3.5.5 Payroll

Modul *Payroll* merupakan salah satu fitur utama dalam sistem CHRIS. Modul ini bertujuan untuk mengelola data penggajian pegawai, termasuk perhitungan gaji berdasarkan tunjangan yang telah ditentukan.

A Diagram ERD Payroll



Gambar 3.9. Diagram ERD untuk modul Payroll

Struktur basis data untuk modul Payroll terdiri dari beberapa tabel utama yang saling berhubungan. Berikut adalah penjelasan singkat mengenai tabel-tabel

tersebut:

- **User**: Tabel ini menyimpan data pegawai yang mencakup informasi pribadi, status kepegawaian, dan referensi ke konfigurasi penggajian yang digunakan.
- **Employment Status**: Tabel ini menyimpan data status kepegawaian yang digunakan sebagai referensi pada berbagai modul dalam sistem, salah satunya adalah modul *Payroll*.
- **Payroll Configuration**: Tabel ini menyimpan konfigurasi penggajian yang mencakup nama, status kepegawaian, dan tunjangan yang berlaku. Setiap konfigurasi dapat memiliki beberapa tunjangan yang terkait.
- **Allowances Types**: Tabel ini menyimpan jenis-jenis tunjangan yang tersedia pada payroll configuraion yang telah dibuat, dan juga tunjangan tambahan untuk pegawai tertentu. Setiap jenis tunjangan memiliki nama, dan tipe (tunjangan atau potongan).
- **User Payroll Application**: Tabel ini menyimpan data Payroll yang telah diisi oleh *Superadmin* untuk setiap pegawai.
- **User Allowances**: Tabel ini menyimpan data tunjangan spesifik untuk setiap pegawai. Tabel ini berisi informasi mengenai jenis tunjangan, jumlah, dan referensi ke pegawai yang bersangkutan.

B Alur Sistem Payroll

Alur sistem *Payroll* diawali dengan pembuatan data *Payroll Configuration*, yang mencakup nama konfigurasi, status kepegawaian (*Employment Status*), serta daftar tunjangan (*Allowances*) yang berlaku. Setelah konfigurasi dibuat, *Superadmin* melanjutkan ke modul *User Management* untuk mengatur data gaji setiap pegawai secara individual.

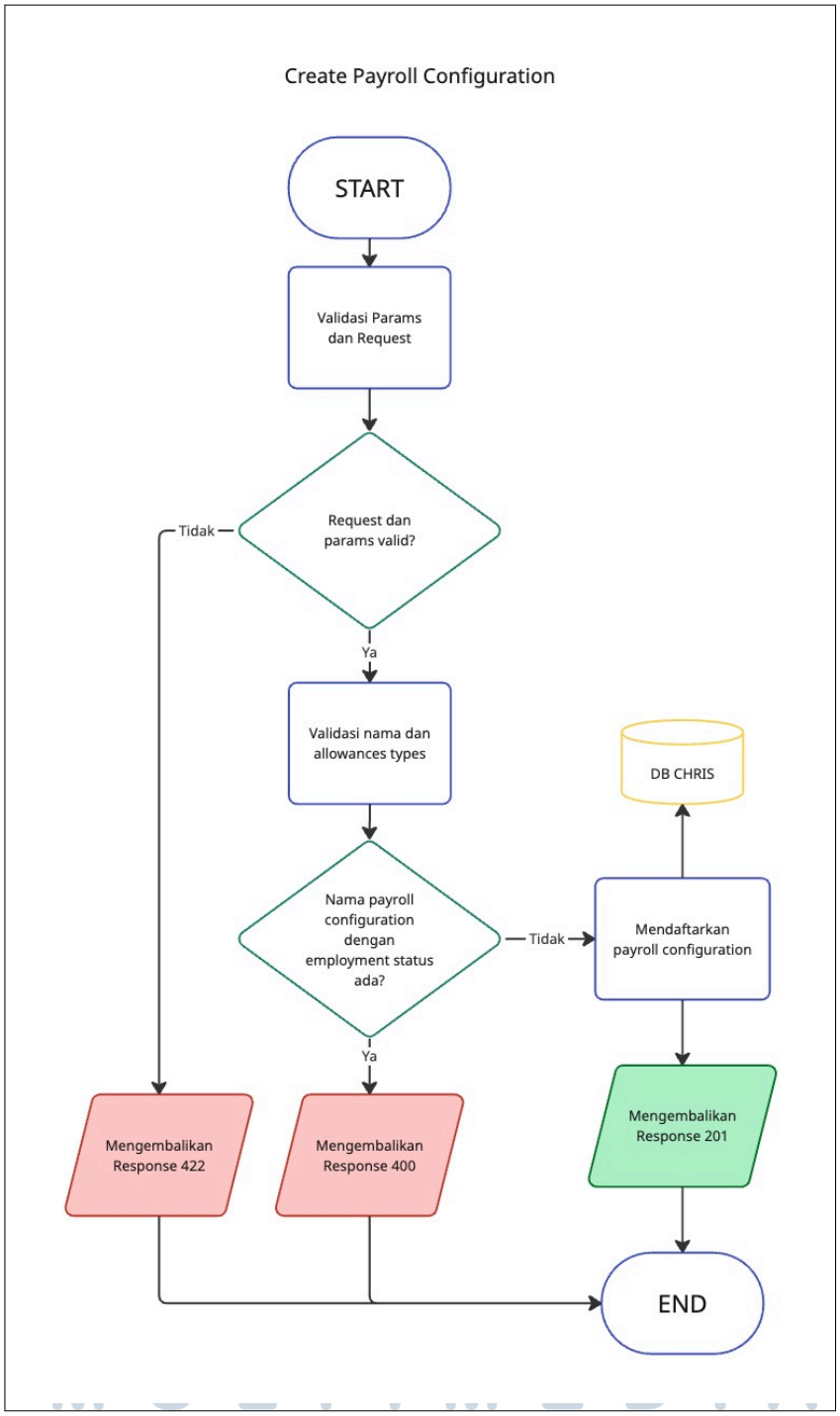
Dalam modul *User Management*, *Superadmin* memilih *Payroll Configuration* berdasarkan status kepegawaian pengguna, mengisi besaran gaji pokok, serta melengkapi jumlah masing-masing tunjangan yang ditetapkan. Selain itu, *Superadmin* juga dapat menambahkan tunjangan (*allowance*) atau potongan (*deduction*) khusus yang hanya berlaku bagi pengguna tersebut, guna menyesuaikan skema gaji secara fleksibel.

Setelah data selesai disimpan, *Superadmin* dapat mengakses modul *Salary Slip* untuk melakukan finalisasi gaji. Finalisasi ini memungkinkan pengecekan akhir terhadap rincian gaji sebelum tanggal gajian. Di PT Ganda Visi Jayatama, proses penggajian dilakukan setiap tanggal 25, sehingga proses finalisasi disarankan dilakukan pada tanggal 24 setiap bulannya. Setelah tanggal 25, data tidak dapat lagi diubah.

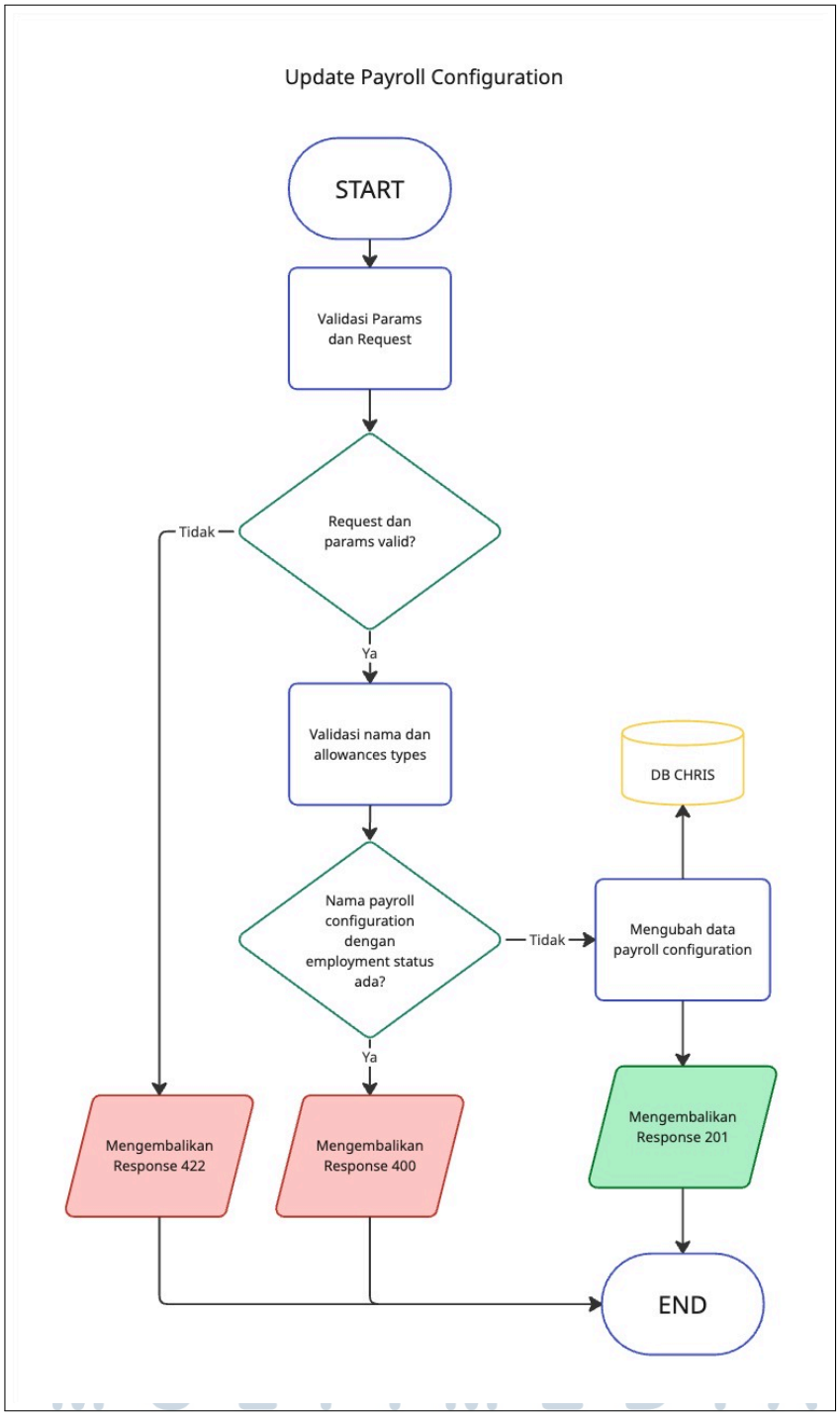
Pegawai yang telah memiliki data gaji terverifikasi dapat melihat slip gaji mereka masing-masing pada halaman *Salary Slip* dan mengunduhnya dalam format PDF.

B.1 Payroll Configuration

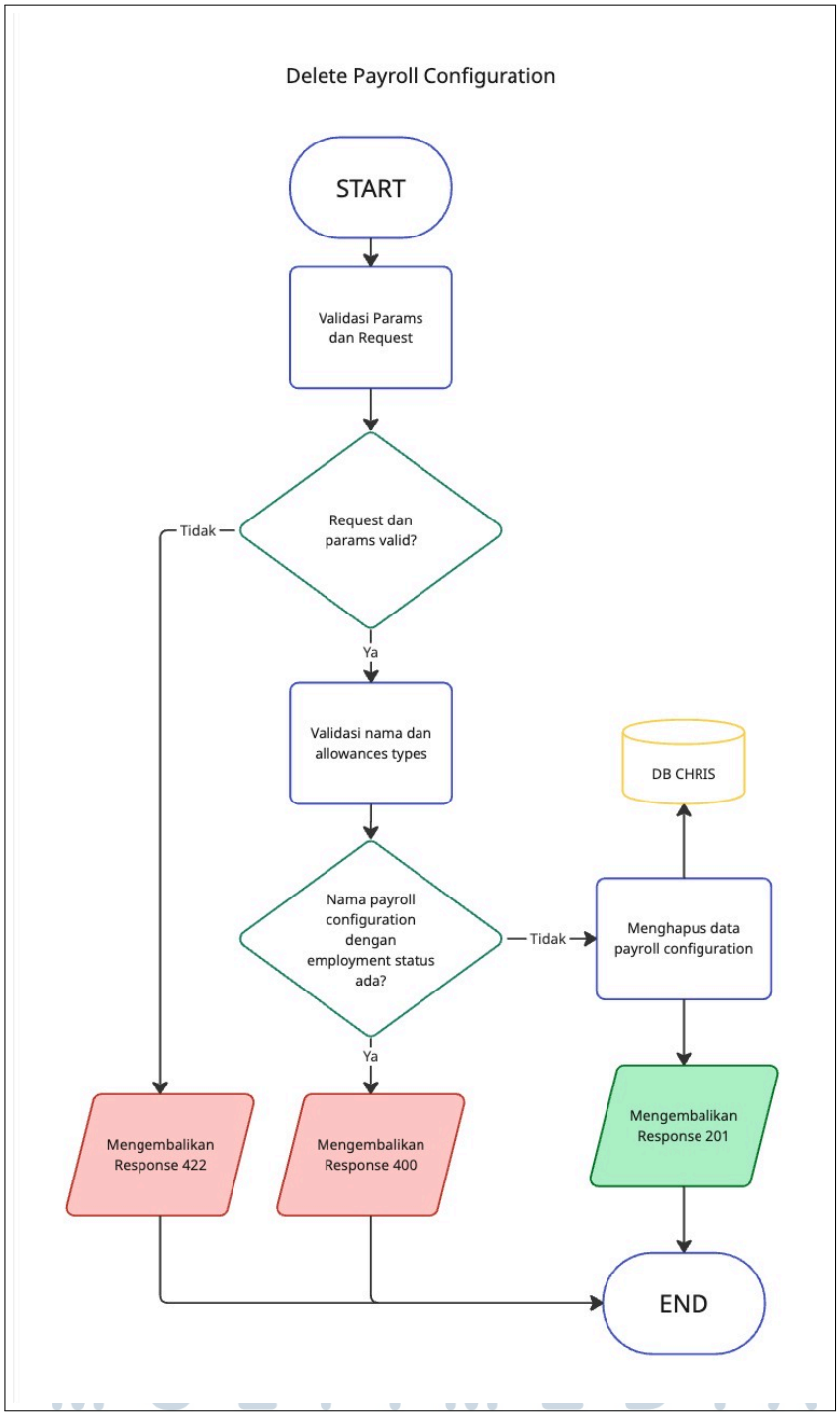
Pada Gambar **??**, **??**, dan **??** merupakan alur sistem untuk Payroll Configuration yang dimulai dari pembuatan konfigurasi penggajian, di mana *Superadmin* membuat konfigurasi baru dengan mengisi nama konfigurasi, status kepegawaian, dan daftar tunjangan (*Allowances*) yang berlaku. Setelah itu, *Superadmin* dapat mengakses modul *User Management* untuk mengatur data gaji setiap pegawai.



Gambar 3.10. *Flowchart create payroll configuration*



Gambar 3.11. *Flowchart update payroll configuration*



Gambar 3.12. *Flowchart delete payroll configuration*

Gambar 3.10, 3.11, dan 3.12 menggambarkan alur proses pembuatan, pengubahan, dan penghapusan data *Payroll Configuration*. Ketiga proses tersebut menerapkan validasi yang sama, yaitu pengecekan *request* dan *params*, dan

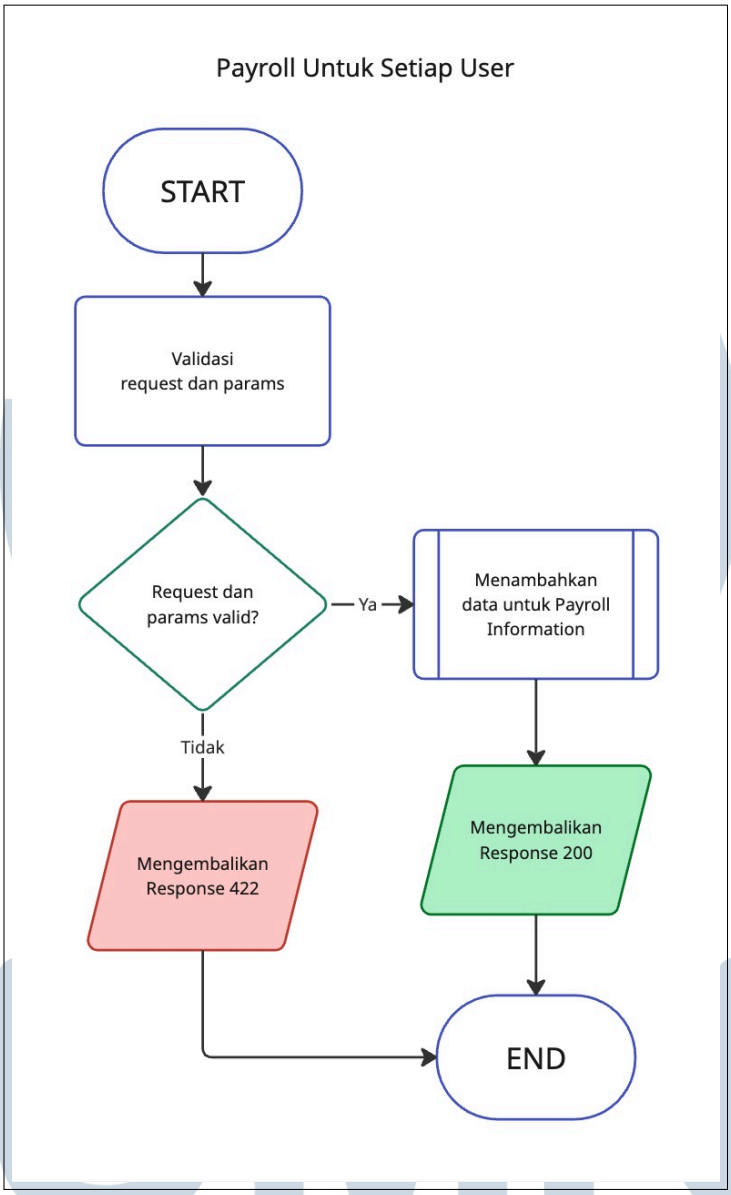
pengecekan terhadap kombinasi nama Payroll Configuration dan Employment *Status* yang sudah terdaftar sebelumnya.

Apabila *request* atau *parameter* yang dikirimkan tidak sesuai dengan format atau aturan yang telah ditentukan, sistem akan merespons dengan kode 422 (*Invalid Format*) sebagai penolakan terhadap permintaan yang tidak valid. Selain itu, jika kombinasi nama *Payroll Configuration* dan *Employment Status* telah terdaftar sebelumnya, sistem akan mengembalikan respons kode 400 (*Bad Request*) untuk mencegah terjadinya duplikasi data.

Validasi ini bertujuan untuk menjaga konsistensi dan integritas data dalam sistem. Jika seluruh validasi berhasil dilewati, maka proses pembuatan, pengubahan, atau penghapusan akan dilanjutkan, dengan sistem memberikan respons berupa kode 201 (Created) untuk pembuatan, serta kode 200 (OK) untuk pengubahan dan penghapusan data.

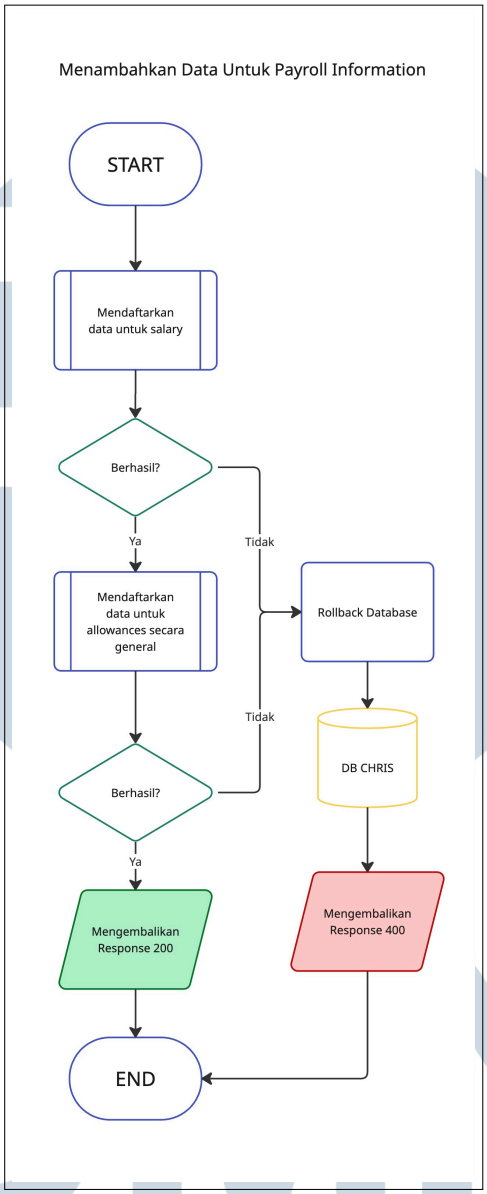
B.2 User Allowances

Setelah *Superadmin* membuat *payroll configuration*, langkah selanjutnya adalah mengatur data gaji setiap pegawai. Proses ini dilakukan melalui modul *User Management*, di mana *Superadmin* memilih *Payroll Configuration* berdasarkan status kepegawaian pengguna, mengisi besaran gaji pokok, serta melengkapi jumlah masing-masing tunjangan yang ditetapkan.



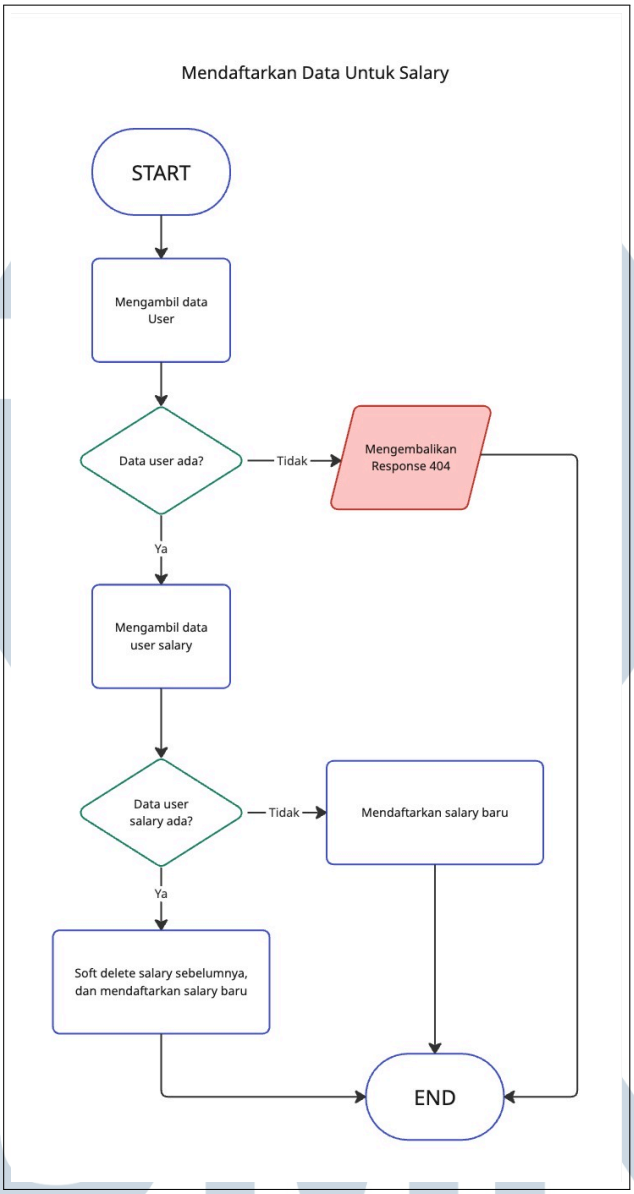
Gambar 3.13. *Flowchart* implementasi payroll untuk setiap user

Gambar 3.13 menunjukkan proses untuk validasi *request* dan *parameter* akan dilanjuti dengan proses menambahkan data untuk *payroll information* secara *general*.



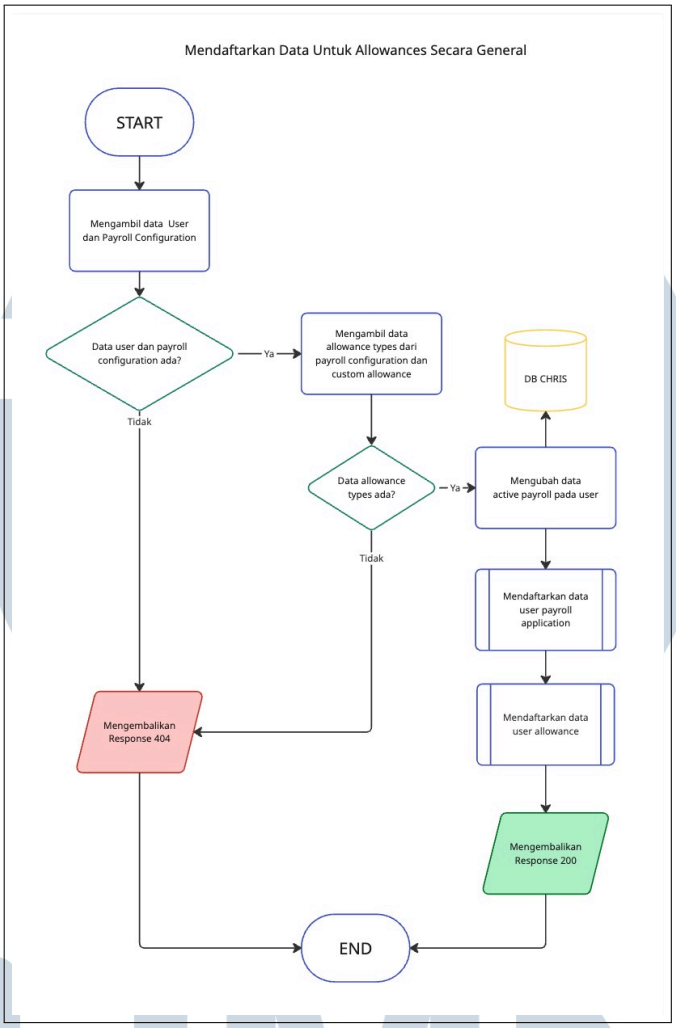
Gambar 3.14. *Flowchart* menambahkan data untuk payroll information

Pada *user management*, terdapat suatu field bernama *Payroll Information* yang berisi data gaji pegawai, dan field untuk memilih *Payroll Configuration* yang telah dibuat sebelumnya. Setelah memilih *Payroll Configuration*, *Superadmin* dapat mengisi data gaji pokok, tunjangan, dan potongan yang berlaku untuk pegawai tersebut. Gambar 3.13 menunjukkan alur sistem untuk pembuatan data gaji pegawai.



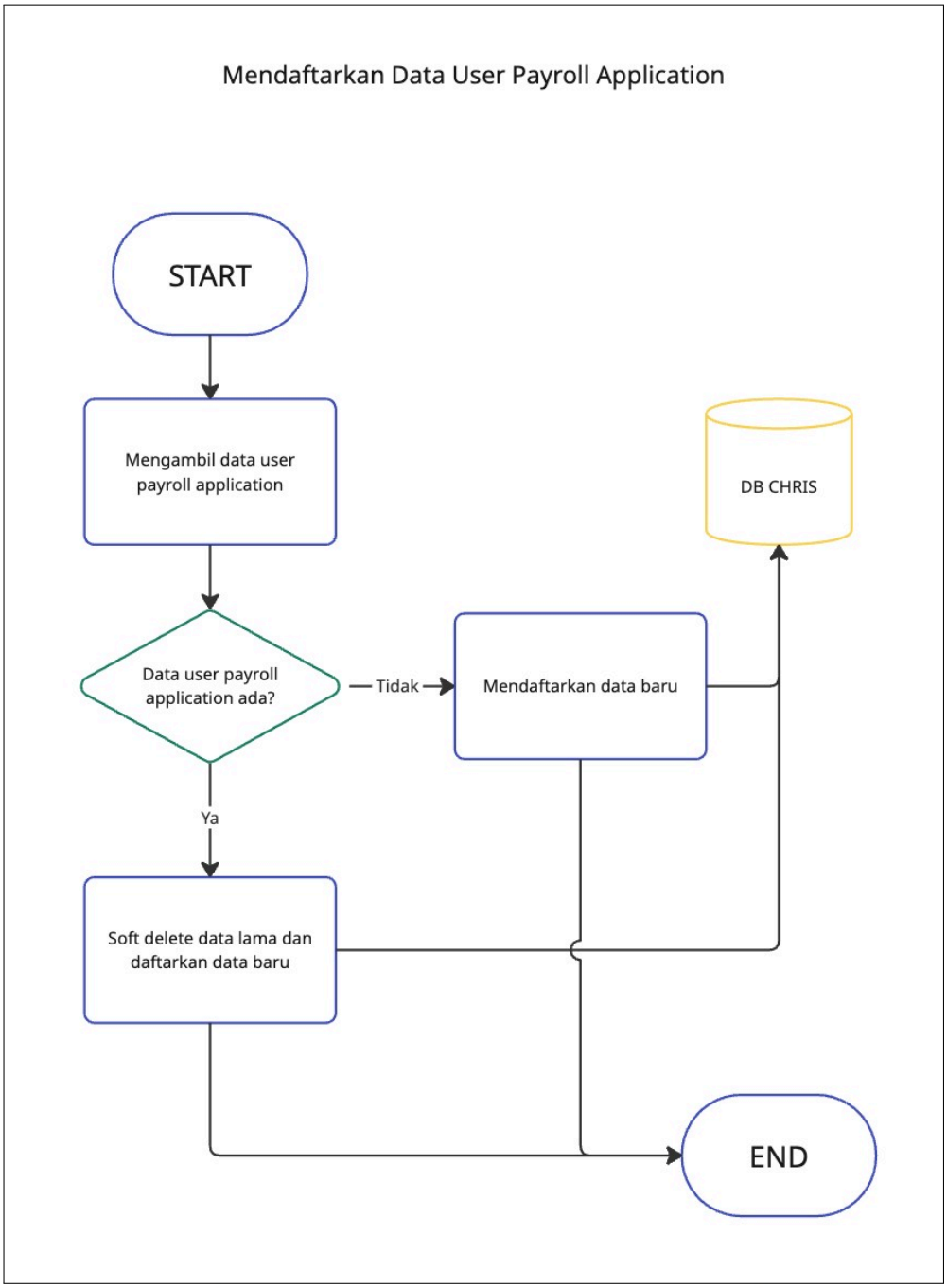
Gambar 3.15. *Flowchart* mendaftarkan data untuk salary

Gambar 3.15 menunjukkan proses pembuatan data gaji pegawai yang dimulai dengan mencari data pegawai dan mengambil data gaji pegawai tersebut. Setelah itu, jika data tersebut ditemukan, sistem akan menghapus data gaji pegawai yang lama secara *soft delete* dan membuat data gaji pegawai yang baru.



Gambar 3.16. *Flowchart* mendaftarkan data user allowances secara *general*

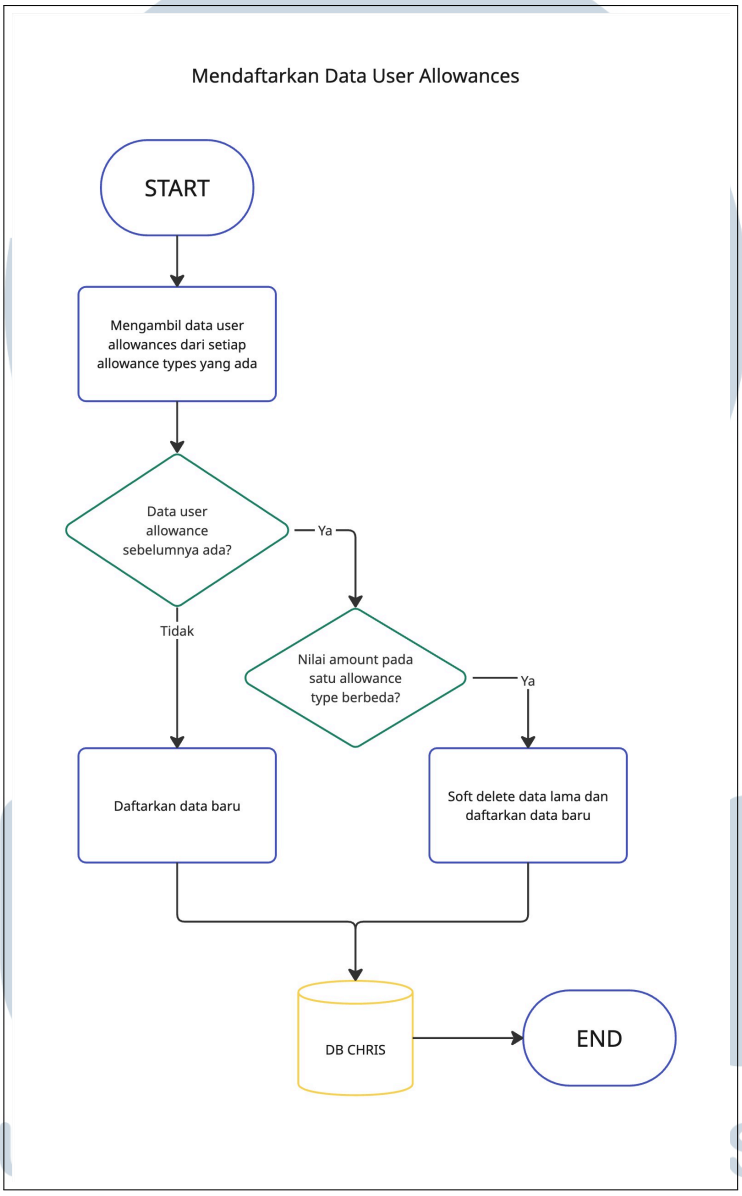
Gambar 3.16 menggambarkan alur proses pendaftaran data tunjangan pegawai secara umum. Proses ini diawali dengan pengambilan data pegawai dan *Payroll Configuration* yang telah tersedia. Selanjutnya, sistem mengekstraksi daftar tunjangan yang tercantum dalam *Payroll Configuration* tersebut. Setelah itu, sistem akan memperbarui informasi *Active Payroll* pada data pegawai, dan mendaftarkan entri baru pada *User Payroll Application* guna menghubungkan pegawai dengan konfigurasi payroll yang aktif. Terakhir, sistem akan mendaftarkan data *User Allowances* beserta nominal masing-masing tunjangan dan potongan yang telah ditentukan sebelumnya.



Gambar 3.17. *Flowchart* mendaftarkan data *user payroll application*

Gambar 3.17 menggambarkan proses pendaftaran data *User Payroll Application* yang diawali dengan pencarian entri sebelumnya pada pegawai terkait. Jika tidak ditemukan, sistem akan langsung membuat entri baru pada tabel *User Payroll Application*. Namun, apabila entri sudah ada, sistem akan terlebih dahulu

melakukan *soft delete* terhadap data tersebut, kemudian membuat entri baru guna memastikan hanya satu konfigurasi aktif yang tercatat untuk setiap pegawai. Pendekatan ini menjaga integritas historis tanpa menghapus data secara permanen.



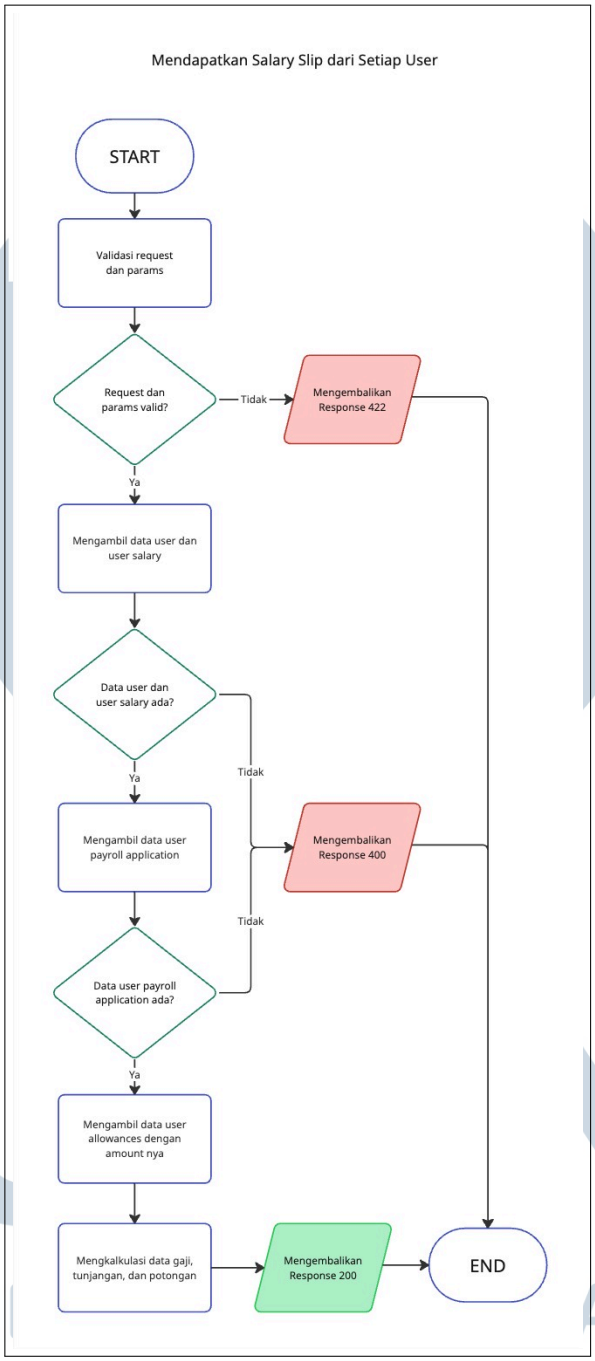
Gambar 3.18. *Flowchart* mendaftarkan data *user allowances*

Gambar 3.18 menggambarkan proses pendaftaran data tunjangan pegawai yang diawali dengan pencarian data *User Allowances* berdasarkan setiap *Allowance Type* yang dimiliki pegawai. Apabila data tidak ditemukan, sistem akan membuat entri baru dengan mengisikan *Allowance Type* beserta nominal tunjangan yang telah ditentukan. Namun, jika data ditemukan, sistem akan melakukan pengecekan

terhadap kesesuaian nominal tunjangan. Jika nominal yang ditemukan sama, maka tidak ada perubahan yang dilakukan. Sebaliknya, jika terdapat perbedaan, sistem akan memperbarui nilai nominal dengan yang baru, serta melakukan *soft delete* pada data lama. Pendekatan ini diterapkan untuk menjaga riwayat data dan memungkinkan pelacakan perubahan secara historis.

B.3 Salary Slip

Setelah data gaji pegawai selesai dibuat, *Superadmin* dapat mengakses modul *Salary Slip* untuk melakukan finalisasi gaji. Proses finalisasi ini memungkinkan pengecekan akhir terhadap rincian gaji sebelum tanggal gajian. Di PT Ganda Visi Jayatama, proses penggajian dilakukan setiap tanggal 25, sehingga proses finalisasi disarankan dilakukan pada tanggal 24 setiap bulannya. Setelah tanggal 25, data tidak dapat lagi diubah.



Gambar 3.19. *Flowchart* mendapatkan salary slip dari setiap user

Gambar 3.19 menunjukkan proses mendapatkan slip gaji pegawai yang dimulai dengan mencari data pegawai berdasarkan *Payroll Configuration* yang telah dibuat sebelumnya. Setelah itu, sistem akan mengambil data gaji pegawai, tunjangan-tunjangan yang telah dibuatkan sebelumnya, dan membuat slip gaji

berdasarkan data tersebut.

Untuk pegawai yang telah memiliki data gaji terverifikasi, mereka dapat melihat slip gaji mereka masing-masing pada halaman *Salary Slip* dan mengunduhnya dalam format PDF. Hal ini memungkinkan pegawai untuk mengakses informasi gaji mereka secara transparan dan mudah.

Untuk mendapatkan slip gaji historis, sistem akan terlebih dahulu menentukan periode waktu berdasarkan tanggal yang diminta, yaitu awal hingga akhir bulan tersebut. Setelah itu, sistem akan mencari seluruh entri data gaji (*User Salary*) yang memiliki tanggal pembuatan (*created_at*) sebelum atau sama dengan akhir periode, dan belum dihapus atau dihapus setelah periode tersebut berakhir. Seluruh data yang ditemukan kemudian diurutkan berdasarkan tanggal pembuatan dari yang terbaru ke yang terlama. Dari hasil pengurutan tersebut, sistem akan memilih satu entri data gaji paling terbaru yang masih berlaku pada periode tersebut. Entri tersebut kemudian digunakan untuk mengambil informasi tunjangan (*User Allowances*) dan pemotongan berdasarkan konfigurasi yang berlaku, lalu disusun menjadi slip gaji pegawai.

3.6 Implementasi Sistem

Implementasi sistem menghasilkan total 24 *endpoint API* yang mendukung fungsionalitas berbagai modul yang dikembangkan selama proyek. Setiap *endpoint* menggunakan salah satu dari lima metode HTTP standar, yaitu *GET*, *POST*, *PUT*, *DELETE*, dan *PATCH*. Metode *GET* digunakan untuk mengambil data dari server, *POST* untuk menambahkan data baru, *PUT* untuk mengganti keseluruhan data pada entitas tertentu, sedangkan *PATCH* memungkinkan pembaruan sebagian data tanpa harus mengirimkan seluruh payload. Sementara itu, metode *DELETE* digunakan untuk menghapus data secara permanen atau melakukan *soft delete* tergantung pada kebijakan implementasi sistem. Pentingnya penggunaan metode-metode ini dalam RESTful API telah dibahas secara mendalam dalam studi mengenai metodologi pengujian REST API [10], serta dokumentasi formal penambahan metode *PATCH* dalam protokol HTTP [11].

Setiap API memiliki standar struktur *request* dan *response* yang wajib diterapkan. Pada *request*, data identifikasi dikirim melalui parameter kueri (*query params*). Untuk *response*, format yang digunakan harus mencakup tiga elemen utama, yaitu *code* (kode status), *message* (pesan status), dan *data* (isi data). Contoh implementasi struktur *request* dan *response* dapat dilihat pada gambar yang

disediakan.

3.6.1 Implementasi *User Management* dan Validasi Data

A API *Endpoints*

Berikut adalah daftar *endpoint* yang telah dimodifikasi dan ditambahkan untuk modul *User Management*:

Tabel 3.2. *User Management* API *Endpoints*

No.	Endpoint	Method	Deskripsi Singkat
1	`/auth/register`	POST	Membuat entri data pegawai baru untuk menggunakan sistem CHRIS.
2	`/users/update`	PATCH	Mengubah data pegawai yang sudah ada sebelumnya.
3	`/employee-statuses`	GET	Mengambil daftar semua jenis kepegawaian (*employment status*).
4	`/banks`	GET	Mengambil daftar semua jenis bank .

Pada Tabel 3.2, ditampilkan sejumlah endpoint yang telah ditambahkan dan dimodifikasi untuk modul *User Management*. Setiap *endpoint* dilengkapi dengan metode HTTP yang sesuai dengan tujuannya: *POST* digunakan untuk pembuatan data baru, *PATCH* untuk pembaruan data, dan *GET* untuk pengambilan data dari basis data.

Secara khusus, *endpoint* seperti `/users/register` dan `/users/update` memiliki proses validasi input untuk memastikan data yang masuk telah memenuhi format dan ketentuan yang berlaku. *Field* yang divalidasi ulang meliputi:

- *fullname*: Nama lengkap pegawai, wajib diisi dan harus berupa karakter alfanumerik yang valid.
- *email*: Alamat *email* pegawai, Hanya menerima domain @concise.co.id dengan format yang sesuai menggunakan ekspresi reguler dan wajib diisi.

- *nik*: Nomor Induk Kependudukan pegawai, Wajib 16 digit angka.
- *contact* dan *emergency_contact*: Nomor telepon dan nomor telepon darurat, wajib diisi dan hanya menerima nomor dengan awalan +62 atau 0.
- *bank_holder*: Nama terdaftar pada bank, wajib diisi dan harus alfanumerik.
- *bank_number*: Nomor rekening bank pegawai, wajib diisi dan hanya angka.

Validasi dilakukan untuk memastikan integritas data dan menghindari kesalahan yang dapat terjadi dalam pengolahan data pegawai, termasuk proses penggajian dan pelacakan informasi.

Sementara itu, beberapa *endpoint* seperti `/employee-statuses` dan `/banks` hanya berfungsi untuk mengambil data dari basis data dan tidak melalui proses validasi karena semata-mata digunakan untuk mengisi data pendukung pada *dropdown* dalam formulir pendaftaran atau pembaruan data pegawai.

B *Request* dan *Response*

Berikut adalah struktur *request* dan *response* untuk *endpoint* `/auth/register` pada modul *User Management*:

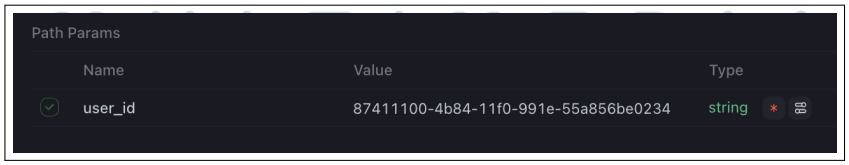
```json
{
    "fullname": "Jose Andreas Lie",
    "emergency_contact_holder": "Vannes Vincent Lie",
    "bank_holder": "Jose Andreas Lie",
    "employee_number": "18",
    "email": "jose.andreas@concise.co.id",
    "nik": "3659876543210987",
    "contact": "85123456789",
    "emergency_contact": "851234567898",
    "password": "Passw0rd",
    "confirmPassword": "Passw0rd",
    "npwp": "-",
    "bank_number": "6040928624",
    "bank_branch": "Jakarta",
    "gender": "male",
    "job_title": "Backend Engineer",
    "supervisor_id": "71f01488-5f38-4c1f-8d33-be176a03d8a1",
    "employment_status_id": "a10cb53d-1dc9-4d9a-9d99-4665a2d1b7c2",
    "bank_id": "4bf72b76-9635-4e52-a0f0-163d146fe47b",
    "role_id": [
        "5caa0a2c-a845-4049-96be-d3ce5014e57e"
    ],
    "roles": [
        "Staff"
    ],
    "dob": "2004-08-22T00:00:00.000Z"
}
```


Gambar 3.20. Contoh struktur *request* untuk `/auth/register`

```
{
    "code": 201,
    "message": "The user has been successfully added.",
    "data": {
        "id": "87411100-4b84-11f0-991e-55a856be0234",
        "is_pwd_resetted": false,
        "fullname": "Jose Andreas Lie",
        "emergency_contact_holder": "Vannes Vincent Lie",
        "bank_holder": "Jose Andreas Lie",
        "employee_number": "18",
        "email": "jose.andreas@concise.co.id",
        "nik": "3659876543210987",
        "contact": "85123456789",
        "emergency_contact": "851234567898",
        "npwp": "-",
        "bank_number": "6040928624",
        "bank_branch": "Jakarta",
        "gender": "male",
        "job_title": "Backend Engineer",
        "supervisor_id": "71f01488-5f38-4c1f-8d33-be176a03d8a1",
        "employment_status_id": "a10cb53d-1dc9-4d9a-9d99-4665a2d1b7c2",
        "bank_id": "4bf72b76-9635-4e52-a0f0-163d146fe47b",
        "dob": "2004-08-22T00:00:00.000Z",
        "is_active": true,
        "updated_at": "2025-06-17T14:08:22.544Z",
        "created_at": "2025-06-17T14:08:22.544Z",
        "daily_status": null,
        "bank_name": null,
        "deleted_at": null,
        "active_payroll_id": null
    },
    "tokens": {
        "access": {
            "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI4NzQxMTEwMC00Yjg0LTExZjAtOTkxZS01NWE
            "expires": "Wed Jun 18 2025 05:28:22 GMT+0700 (Western Indonesia Time)"
        },
        "refresh": {
            "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI4NzQxMTEwMC00Yjg0LTExZjAtOTkxZS01NWE
            "expires": "Thu Jul 17 2025 21:08:22 GMT+0700 (Western Indonesia Time)"
        }
    }
}
```

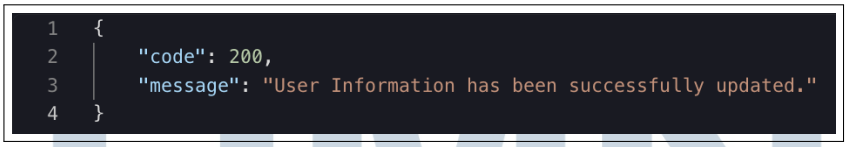

Gambar 3.21. Contoh struktur *response* untuk `/auth/register`

Berikut adalah struktur *request* dan *response* untuk *endpoint* `/users/update` pada modul *User Management*:



Gambar 3.22. Contoh struktur *parameters* untuk `/users/update`

Gambar 3.23. Contoh struktur *request* untuk `/users/update`



Gambar 3.24. Contoh struktur *response* untuk `/users/update`

Pada gambar 3.20, terlihat struktur *request* yang dikirimkan ke *endpoint* `/auth/register`. Data yang dikirim mencakup informasi pegawai seperti nama lengkap, alamat email, nomor induk kependudukan (NIK), nomor telepon, kontak darurat, nama pemegang rekening bank, nomor rekening bank, dan masih banyak lagi. Setelah request berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.21, yang menunjukkan bahwa pendaftaran pegawai baru berhasil dengan kode status 201 (Created) dan pesan yang sesuai.

Pada gambar 3.22 dan 3.23, terlihat struktur *request* yang dikirimkan ke *endpoint* `/users/update`. Data yang dikirim sama seperti yang dikirim pada *request* pendaftaran, namun dengan tambahan *id* pegawai yang akan diperbarui pada bagian parameter. Setelah *request* berhasil diproses, sistem akan

mengembalikan respons seperti pada gambar 3.24, yang menunjukkan bahwa pembaruan data pegawai berhasil dengan kode status 200 (OK) dan pesan yang sesuai.

Berikut adalah struktur *response* untuk *endpoint* `/employee-statuses` pada modul *User Management*:

Gambar 3.25. Contoh struktur *request* untuk `/employee-statuses`

Berikut adalah struktur *response* untuk *endpoint* `/banks` pada modul *User Management*:

```
{
    "code": 200,
    "message": "bank fetched successfully",
    "data": [
        {
            "id": "4bf72b76-9635-4e52-a0f0-163d146fe47b",
            "name": "Bank Central Asia",
            "value": "bca",
            "created_at": "2025-06-13T08:19:32.014Z",
            "updated_at": "2025-06-13T08:19:32.014Z",
            "deleted_at": null
        },
        {
            "id": "99f5361a-d9e7-49c0-832e-e23c7485026d",
            "name": "Bank Mandiri",
            "value": "mandiri",
            "created_at": "2025-06-13T08:19:32.014Z",
            "updated_at": "2025-06-13T08:19:32.014Z",
            "deleted_at": null
        },
        {
            "id": "393b816e-91d0-478a-b436-1748db12a7d9",
            "name": "BNI (Bank Negara Indonesia)",
            "value": "bni",
            "created_at": "2025-06-13T08:19:32.014Z",
            "updated_at": "2025-06-13T08:19:32.014Z",
            "deleted_at": null
        },
        {
            "id": "05972fb1-16a9-4940-ac83-1a600c86baea",
            "name": "Bank Rakyat Indonesia",
            "value": "bri",
            "created_at": "2025-06-13T08:19:32.014Z",
            "updated_at": "2025-06-13T08:19:32.014Z",
            "deleted_at": null
        },
        {
            "id": "6dd9c6bd-10db-4730-89ea-55ee8aaa9415",
            "name": "CIMB Niaga & CIMB Niaga Syariah",
            "value": "cimb",
            "created_at": "2025-06-13T08:19:32.014Z",
            "updated_at": "2025-06-13T08:19:32.014Z",
            "deleted_at": null
```


Gambar 3.26. Contoh struktur *request* untuk `/banks`

3.6.2 Implementasi Fitur Autentikasi Biometrik pada CHRIS *Mobile*

A API *Endpoints*

Berikut adalah daftar *endpoint* yang telah ditambahkan untuk fitur autentikasi biometrik pada CHRIS *Mobile*:

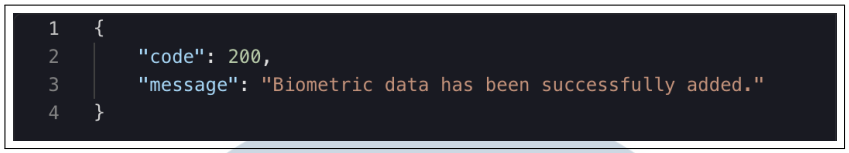
Tabel 3.3. *Biometric Authentication* API *Endpoints*

No.	Endpoint	Method	Deskripsi Singkat
1	`/auth/biometric/register`	POST	Mendaftarkan data biometrik pegawai baru untuk autentikasi.
2	`/auth/biometric/login`	POST	Melakukan autentikasi pegawai menggunakan data biometrik yang telah didaftarkan.

Pada Tabel 3.3, ditampilkan daftar *endpoint* yang telah ditambahkan untuk fitur autentikasi biometrik pada CHRIS *Mobile*. Setiap *endpoint* dilengkapi dengan metode HTTP yang sesuai dengan tujuannya: *POST* digunakan untuk pendaftaran data biometrik pegawai baru dan autentikasi pegawai menggunakan data biometrik yang telah didaftarkan.

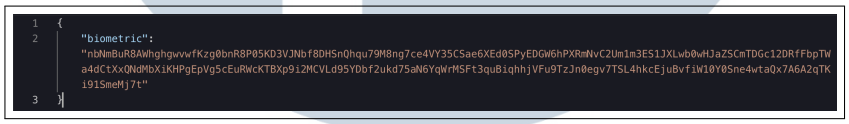
B Contoh *Request* dan *Response*

Gambar 3.27. Contoh struktur *request* untuk `/auth/biometric/register`



Gambar 3.28. Contoh struktur *response* untuk `/auth/biometric/register`

Pada gambar 3.27, terlihat struktur *request* yang dikirimkan ke *endpoint* `/auth/biometric/register`. Data yang dikirim berupa nilai *biometric* yang berisi string acak sepanjang 255 karakter yang dihasilkan oleh aplikasi CHRISM dan akan disimpan pada penyimpanan lokal perangkat. String ini berfungsi sebagai representasi dari data biometrik pengguna, bukan data biometrik aktual yang dikirim ke server. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.28, yang menunjukkan bahwa pendaftaran data biometrik pegawai berhasil dengan kode status 201 (*Created*) dan pesan yang sesuai.



Gambar 3.29. Contoh struktur *request* untuk `/auth/biometric/login`

Gambar 3.30. Contoh struktur *response* untuk `/auth/biometric/login`

Pada gambar 3.29, terlihat struktur *request* yang dikirimkan ke *endpoint* `/auth/biometric/login`. Data yang dikirim hanya berupa nilai *biometric* yang berisi string acak sepanjang 255 karakter yang telah dibuat dan disimpan di storage lokal CHRISM saat proses registrasi biometrik sebelumnya. Nilai ini berfungsi sebagai representasi dari data biometrik pengguna. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.30, yang menunjukkan bahwa autentikasi pegawai menggunakan data biometrik berhasil dengan kode status 200 (*OK*), pesan yang sesuai dan data lainnya.

3.6.3 Implementasi *Leave Permit*

A API *Endpoints*

Berikut adalah daftar *endpoint* yang telah ditambahkan untuk modul *Leave Permit*:

Tabel 3.4. *Leave Permit* API *Endpoints*

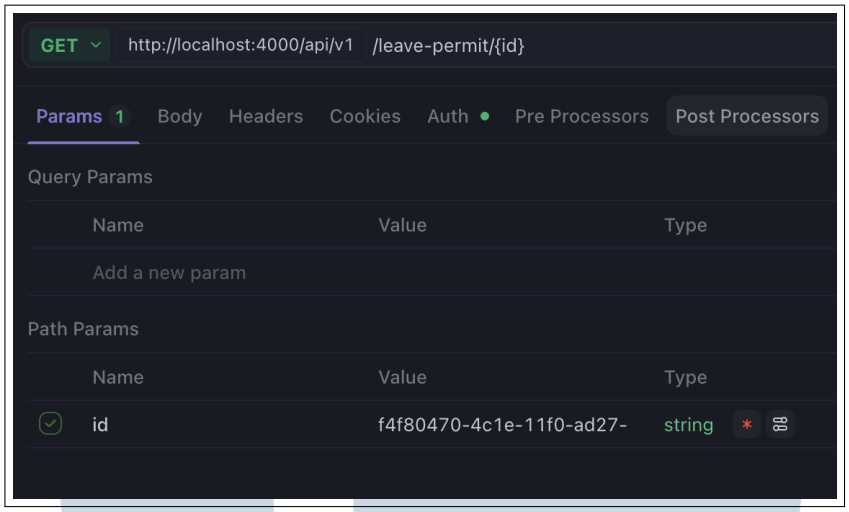
No.	Endpoint	Method	Deskripsi Singkat
1	`/leave-permit`	GET	Menampilkan data leave permit pada user yang login.
2	`/leave-permit/:id`	GET	Mengambil data leave permit berdasarkan id.
3	`/leave-permit`	POST	Membuat data leave permit baru.
4	`/leave-permit/:id`	DELETE	Menghapus data leave permit berdasarkan id.
5	`/leave-permit/dashboard`	GET	Mengambil data leave permit yang telah diterima oleh atasan dan yang berjalan di minggu ini untuk dashboard page.
6	`/leave-permit/all`	GET	Mengambil data leave permit berdasarkan hirarki atasan.

Pada Tabel 3.4, ditampilkan sebagian *endpoint* yang telah dimodifikasi dan ditambahkan untuk modul *Leave Permit*. Tabel ini tidak mencakup seluruh *endpoint* yang ada, melainkan hanya yang relevan dengan pengembangan selama masa kerja praktik. Setiap *endpoint* dilengkapi dengan metode HTTP yang sesuai dengan tujuannya: *GET* digunakan untuk mengambil data dari basis data, *POST* digunakan untuk pembuatan data baru, dan *DELETE* digunakan untuk menghapus data.

B Contoh *Request* dan *Response*

Gambar 3.31. Contoh struktur *response* untuk GET `/leave-permit`

Pada gambar 3.35, terlihat struktur *response* yang dikembalikan oleh sistem ketika melakukan permintaan GET ke *endpoint* `/leave-permit`. Respons ini berisi daftar permohonan cuti yang telah dibuat oleh pegawai yang sedang login, dengan informasi seperti id, jenis cuti, tanggal mulai dan berakhir, status permohonan dan lain-lain.

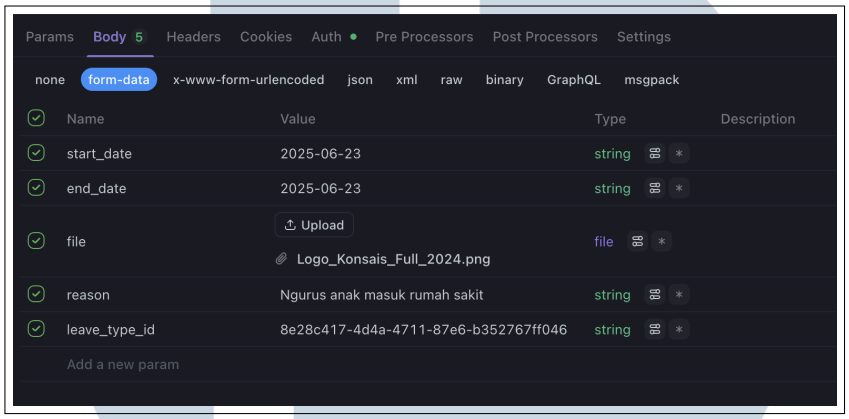


Gambar 3.32. Contoh struktur *request* untuk GET `/leave-permit/id`

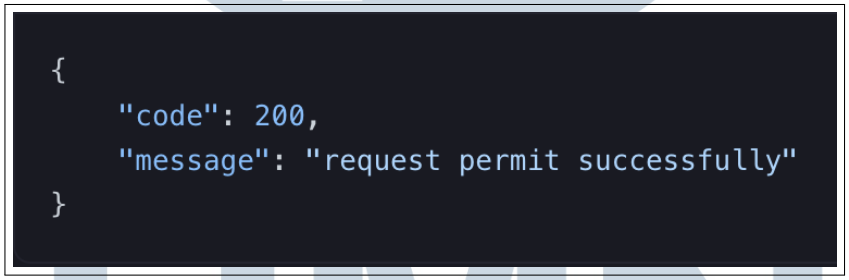
Gambar 3.33. Contoh struktur *response* untuk GET `/leave-permit/:id`

Pada gambar 3.32, terlihat struktur *request* yang dikirimkan ke *endpoint* `/leave-permit/id`. Data yang dikirim hanya berupa *id* permohonan cuti

yang ingin diambil datanya. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.37, yang menunjukkan bahwa pengambilan data leave permit berdasarkan id berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.

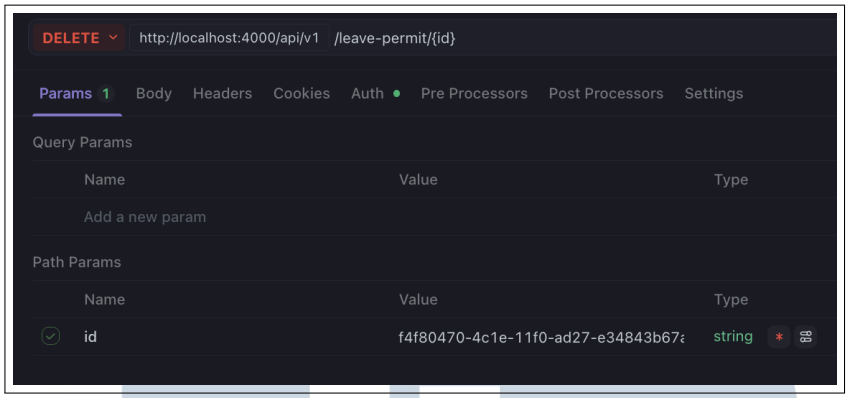


Gambar 3.34. Contoh struktur *request* untuk POST `/leave-permit`



Gambar 3.35. Contoh struktur *response* untuk POST `/leave-permit`

Pada gambar 3.36, terlihat struktur *request* yang dikirimkan ke *endpoint* `/leave-permit`. Data yang dikirim berupa informasi terkait permohonan cuti, seperti jenis cuti, tanggal mulai dan berakhir, serta alasan permohonan. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.35, yang menunjukkan bahwa pembuatan data leave permit berhasil dengan kode status 201 (*Created*) dan pesan yang sesuai.



Gambar 3.36. Contoh struktur *request* untuk DELETE `/leave-permit`

Gambar 3.37. Contoh struktur *response* untuk DELETE `/leave-permit/:id`

Pada gambar **??**, terlihat struktur *request* yang dikirimkan ke *endpoint* `/leave-permit/id`. Data yang dikirim hanya berupa *id* permohonan cuti yang ingin dihapus. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.37, yang menunjukkan bahwa penghapusan data leave permit berdasarkan id berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 2,
        "rows": [
            {
                "id": "daf25b50-4aa9-11f0-ba5a-ddf1d091e5e5",
                "status": "accepted",
                "start_date": "2025-06-18T00:00:00.000Z",
                "end_date": "2025-06-18T00:00:00.000Z",
                "oic": {
                    "id": "ad3de583-95de-46b0-9605-98ad0501be1f",
                    "fullname": "Sakura Yamamoto"
                },
                "user": {
                    "id": "71f01488-5f38-4c1f-8d33-be176a03d8a1",
                    "fullname": "Hiroshi Tanaka"
                },
                "leave_type": {
                    "id": "9f60890b-e0f5-4da9-9b2d-c84664c9adc3",
                    "name": "Sick"
                },
                "reviewed_by": {
                    "id": "6cf3b9ac-27ce-4514-b88e-bd4bd5b00769",
                    "fullname": "Superadmin"
                }
            },
            {
                "id": "c10482b0-4c23-11f0-8661-07da1cc1c872",
                "status": "accepted",
                "start_date": "2025-06-20T00:00:00.000Z",
```


Gambar 3.38. Contoh struktur *response* untuk GET `/leave-permit/dashboard`

Pada gambar 3.38, terlihat struktur *response* yang dikembalikan oleh sistem ketika melakukan permintaan GET ke *endpoint* `/leave-permit/dashboard`. Respons ini berisi data leave permit yang telah diterima oleh atasan dan yang sedang berjalan di minggu ini, dengan informasi seperti id, jenis cuti, tanggal mulai dan berakhir, status permohonan, pegawai yang mengajukan cuti, pegawai yang menerima cuti, dan *officer in charge* yang dipilih. Data ini digunakan untuk menampilkan informasi terkait permohonan cuti pada halaman dashboard.

Gambar 3.39. Contoh struktur *response* untuk GET `/leave-permit/all`

Pada gambar 3.39, terlihat struktur *response* yang dikembalikan oleh sistem ketika melakukan permintaan GET ke *endpoint* `/leave-permit/all`. Respons ini berisi data leave permit berdasarkan hirarki atasan, dengan informasi seperti id, jenis cuti, tanggal mulai dan berakhir, status permohonan, pegawai yang mengajukan cuti, pegawai yang menerima cuti, dan *officer in charge* yang dipilih. Data ini digunakan untuk menampilkan informasi terkait permohonan cuti pada halaman yang dapat diakses oleh atasan.

3.6.4 Implementasi *Payroll*

A API *Endpoints*

Berikut adalah daftar *endpoint* yang telah ditambahkan untuk modul *Payroll*:

Tabel 3.5. *Payroll* API *Endpoints*

No.	Endpoint	Method	Deskripsi Singkat
1	`/payroll-configuration/all`	GET	Mengambil daftar data konfigurasi penggajian yang telah dibuat.
2	`/payroll-configuration/:id`	GET	Mengambil data konfigurasi penggajian berdasarkan id.
3	`/payroll-configuration`	POST	Membuat data konfigurasi penggajian baru.
4	`/payroll-configuration/:id`	DELETE	Menghapus data konfigurasi penggajian berdasarkan id.
5	`/payroll-configuration/:id`	PATCH	Mengubah data konfigurasi penggajian berdasarkan id.
6	`/users/:user_id/ payroll-configuration`	GET	Mengambil daftar data konfigurasi penggajian berdasarkan id pegawai.
7	`/users/:user_id/salary`	GET	Mengambil data gaji pegawai berdasarkan id pegawai.
8	`/user-allowance/ :user_id/ :payroll_configuration_id`	GET	Mengambil data tunjangan berdasarkan id pegawai dan id konfigurasi penggajian.
9	`/users/:user_id/ with-details`	PATCH	Menyimpan data pegawai, gaji, dan semua tunjangan yang sudah dikirimkan.
10	`/salary-slip/all?date=YYYY-MM`	GET	Mengambil daftar slip gaji pegawai yang telah dibuat.

Lanjut di halaman berikutnya

Tabel 3.5 *Payroll* API *Endpoints* (lanjutan)

No.	Endpoint	Method	Deskripsi Singkat
11	`/salary-slip/:id`	GET	Mengambil data slip gaji pegawai berdasarkan id.
12	`/salary-slip?date=YYYY-MM`	GET	Mengambil data slip gaji pengguna yang sedang aktif pada bulan tertentu.

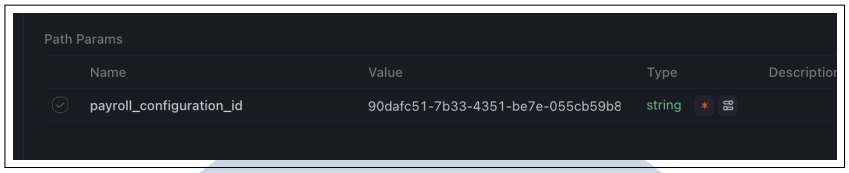
Pada Tabel 3.5, ditampilkan daftar *endpoint* yang telah ditambahkan untuk modul *Payroll*. Setiap *endpoint* dilengkapi dengan metode HTTP yang sesuai dengan tujuannya: *GET* digunakan untuk mengambil data dari basis data, *POST* digunakan untuk pembuatan data baru, *DELETE* digunakan untuk menghapus data, dan *PATCH* digunakan untuk memperbarui data yang ada.

B Contoh *Request* dan *Response*


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 5,
        "rows": [
            {
                "id": "90dafc51-7b33-4351-be7e-055cb59b88ed",
                "name": "Senior Part Time",
                "created_at": "2025-06-13T08:21:46.620Z",
                "updated_at": "2025-06-13T08:21:46.620Z",
                "employment_status": {
                    "id": "1b5254d2-10fa-4fcd-bd07-edf174098ed5",
                    "employment_type": "Part Time"
                }
            },
            {
                "id": "6027a18c-fd82-4b6a-9ecc-6a337f92e21a",
                "name": "Newbie",
                "created_at": "2025-06-13T08:21:58.631Z",
                "updated_at": "2025-06-13T08:21:58.631Z",
                "employment_status": {
                    "id": "a10cb53d-1dc9-4d9a-9d99-4665a2d1b7c2",
                    "employment_type": "Internship"
                }
            },
            {
                "id": "588eb642-4d3b-4040-8a27-fc37b39c16da",
                "name": "Fulltime Senior",
                "created_at": "2025-06-13T08:21:27.633Z",
                "updated_at": "2025-06-13T14:37:14.675Z",
                "employment_status": {
                    "id": "bf6be756-3be8-41e0-9dae-fff2824e340f",
                    "employment_type": "Full Time"
                }
            },
            {
                "id": "1f063e94-c12d-4d7a-9b58-b9bb465f5e0a",
                "name": "Newbie",
```


Gambar 3.40. Contoh struktur *request* untuk GET `/payroll-configuration/all`

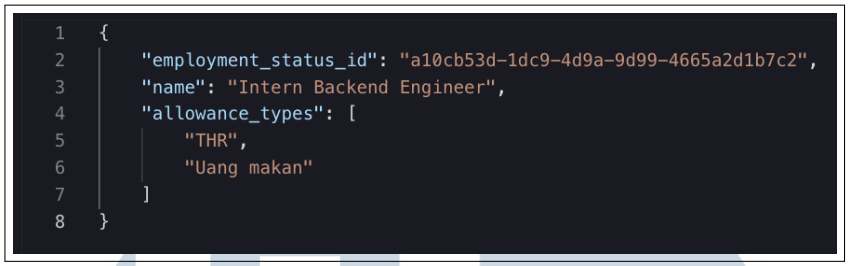
Pada gambar 3.40, terlihat struktur *response* yang dikembalikan oleh sistem ketika melakukan permintaan GET ke *endpoint* `/payroll-configuration/all`. Respons ini berisi daftar konfigurasi penggajian yang telah dibuat, dengan informasi seperti id, nama konfigurasi, dan status kepegawaian.



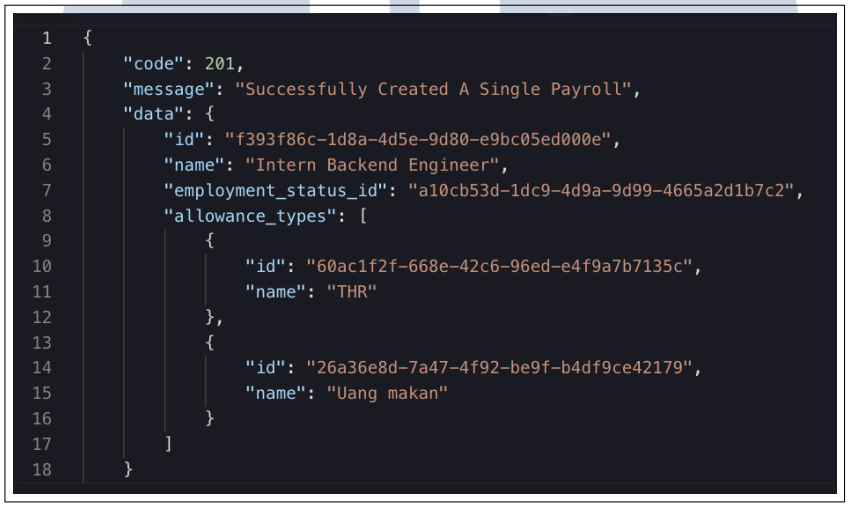
Gambar 3.41. Contoh struktur *request* untuk GET `/payroll-configuration/:id`

Gambar 3.42. Contoh struktur *response* untuk GET `/payroll-configuration/:id`

Pada gambar 3.41, terlihat struktur *request* yang dikirimkan ke *endpoint* `/payroll-configuration/id`. Data yang dikirim hanya berupa *id* konfigurasi penggajian yang ingin diambil datanya. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.42, yang menunjukkan bahwa pengambilan data konfigurasi penggajian berdasarkan id berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.

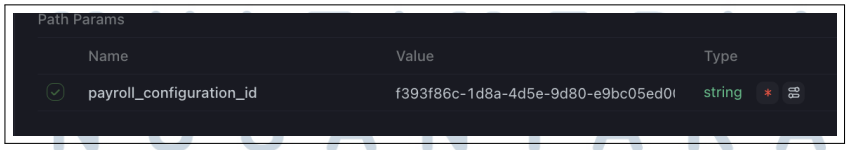


Gambar 3.43. Contoh struktur *request* untuk POST `/payroll-configuration`

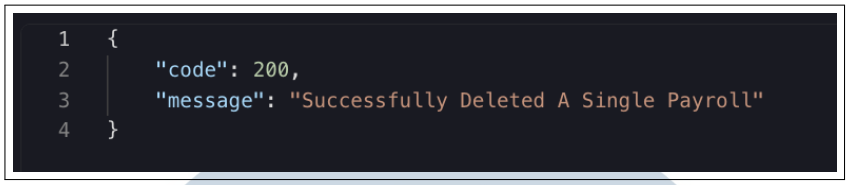


Gambar 3.44. Contoh struktur *response* untuk POST `/payroll-configuration`

Pada gambar 3.43, terlihat struktur *request* yang dikirimkan ke *endpoint* `/payroll-configuration`. Data yang dikirim berupa informasi terkait konfigurasi penggajian, seperti nama konfigurasi, status kepegawaian, dan tunjangan-tunjangan. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.44, yang menunjukkan bahwa pembuatan data konfigurasi penggajian berhasil dengan kode status 201 (*Created*) dan pesan yang sesuai.

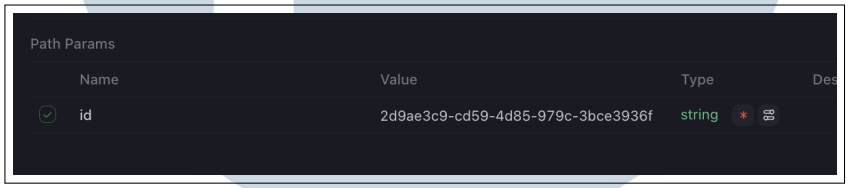


Gambar 3.45. Contoh struktur *request* untuk DELETE `/payroll-configuration/:id`



Gambar 3.46. Contoh struktur *response* untuk DELETE `/payroll-configuration/:id`

Pada gambar 3.45, terlihat struktur *request* yang dikirimkan ke *endpoint* `/payroll-configuration/id`. Data yang dikirim hanya berupa *id* konfigurasi penggajian yang ingin dihapus. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.46, yang menunjukkan bahwa penghapusan data konfigurasi penggajian berdasarkan id berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.

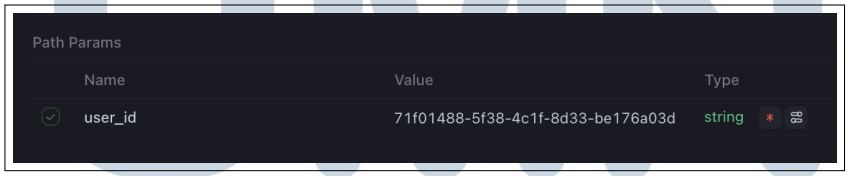


Gambar 3.47. Contoh struktur *params* untuk PATCH `/payroll-configuration/:id`

Gambar 3.48. Contoh struktur *request* untuk PATCH `/payroll-configuration/:id`

Gambar 3.49. Contoh struktur *response* untuk PATCH `/payroll-configuration/:id`

Pada gambar **??**, terlihat parameter yang dikirimkan ke path *endpoint* `/payroll-configuration/:id` yang berisi ID konfigurasi penggajian yang ingin dimodifikasi. Sementara pada gambar 3.48, terlihat struktur *request body* yang berisi field-field yang ingin direvisi, seperti nama konfigurasi, status kepegawaian, dan tunjangan-tunjangan. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.49, yang menunjukkan bahwa pembaruan data konfigurasi penggajian berdasarkan id berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.

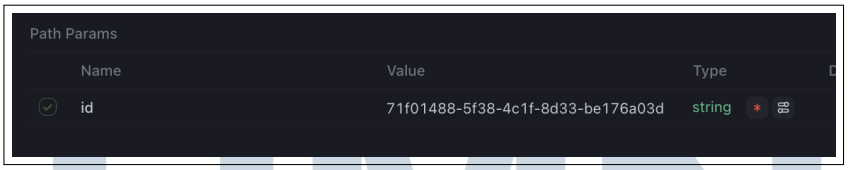


Gambar 3.50. Contoh struktur *request* untuk GET `/users/:user_id/payroll-configuration`

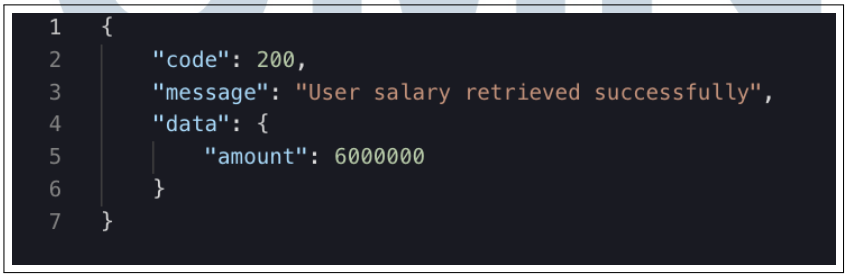
p

Gambar 3.51. Contoh struktur *response* untuk GET `/users/:user_id/payroll-configuration`

Pada gambar 3.51, terlihat struktur *request* yang dikirimkan ke *endpoint* `/users/:user_id/payroll-configuration`, yang berisi parameter id pegawai yang ingin diambil data konfigurasi penggajiannya. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.50, yang menunjukkan daftar konfigurasi penggajian yang tersedia berdasarkan status kepegawaian pegawai tersebut, dengan kode status 200 (*OK*) dan pesan yang sesuai.



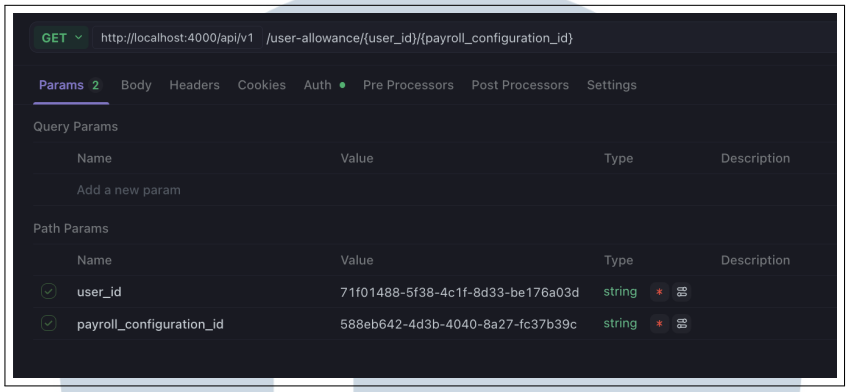
Gambar 3.52. Contoh struktur *request* untuk GET `/users/:user_id/salary`



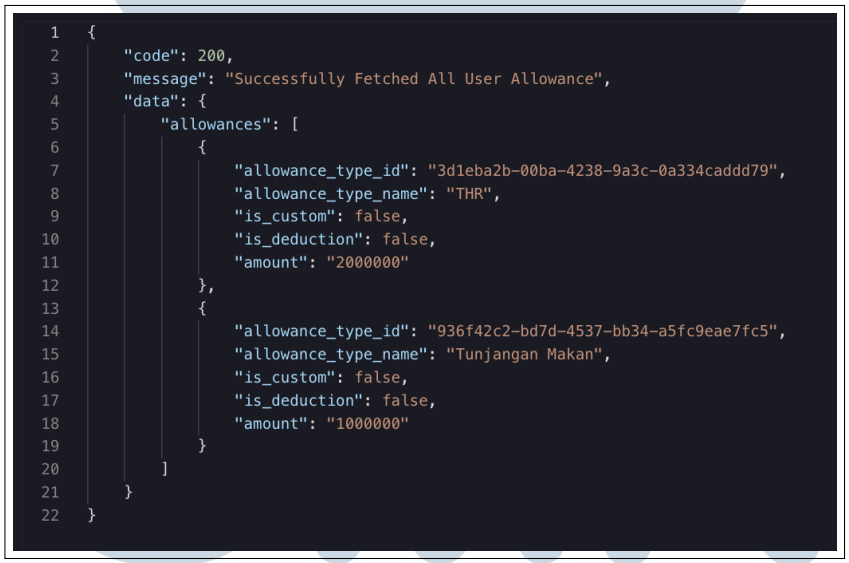
Gambar 3.53. Contoh struktur *response* untuk GET `/users/:user_id/salary`

Pada gambar 3.52, terlihat struktur *request* yang dikirimkan ke *endpoint* `/users/:user_id/salary`, yang berisi parameter id pegawai yang ingin diambil data gajinya. Setelah *request* berhasil diproses, sistem akan mengembalikan respons

seperti pada gambar 3.53, yang menunjukkan data gaji pegawai berdasarkan id pegawai tersebut, dengan kode status 200 (*OK*) dan pesan yang sesuai.

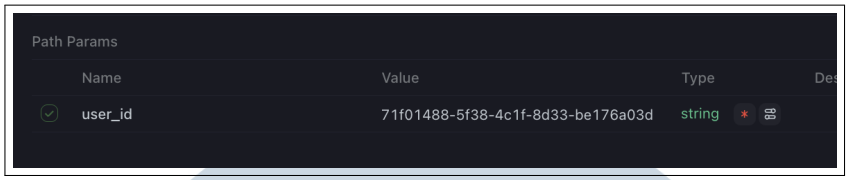


Gambar 3.54. Contoh struktur *request* untuk GET `/user-allowance/:user_id/:payroll_configuration_id`



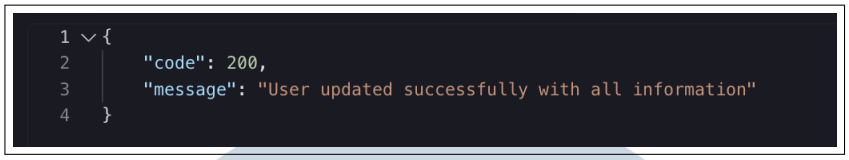
Gambar 3.55. Contoh struktur *response* untuk GET `/user-allowance/:user_id/:payroll_configuration_id`

Pada gambar 3.54, terlihat struktur *request* yang dikirimkan ke *endpoint* `/user-allowance/:user_id/ :payroll_configuration_id`, yang berisi parameter id pegawai dan id konfigurasi penggajian yang ingin diambil data tunjangannya. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.55, yang menunjukkan data tunjangan pegawai berdasarkan id pegawai dan id konfigurasi penggajian tersebut, dengan kode status 200 (*OK*) dan pesan yang sesuai.



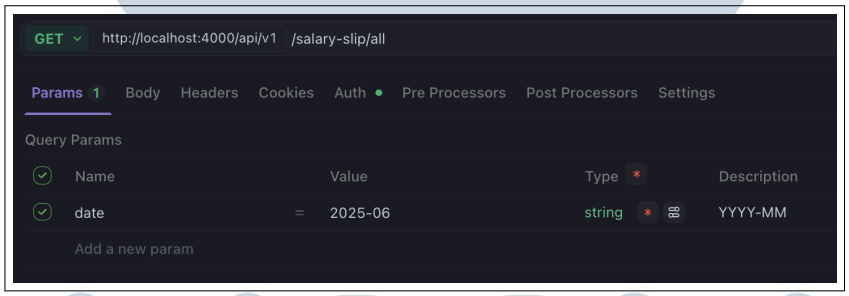
Gambar 3.56. Contoh struktur *params* untuk PATCH `/users/:user_id/with-details`

Gambar 3.57. Contoh struktur *request* untuk PATCH `/users/:user_id/with-details`



Gambar 3.58. Contoh struktur *response* untuk PATCH `/users/:user_id/with-details`

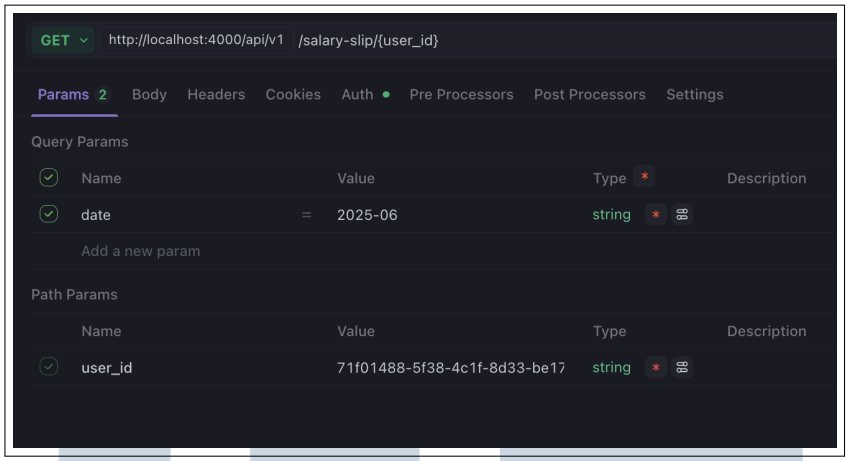
Pada gambar 3.56, terlihat parameter yang dikirimkan ke path *endpoint* `/users/:user_id/with-details` yang berisi ID pegawai yang ingin dimodifikasi. Sementara pada gambar 3.57, terlihat struktur *request body* yang berisi field-field yang ingin direvisi, seperti data pegawai, gaji, dan tunjangan-tunjangan. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.58, yang menunjukkan bahwa pembaruan data pegawai, gaji, dan tunjangan berdasarkan id pegawai berhasil dengan kode status 200 (*OK*) dan pesan yang sesuai.



Gambar 3.59. Contoh struktur *request* untuk GET `/salary-slip/all`

Gambar 3.60. Contoh struktur *response* untuk GET `/salary-slip/all`

Pada gambar 3.59, terlihat struktur *request* yang dikirimkan ke *endpoint* `/salary-slip/all`, yang berisi query parameter `date=YYYY-MM` untuk menyaring slip gaji pada bulan dan tahun tertentu. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.60, yang menunjukkan daftar slip gaji pegawai yang telah dibuat untuk periode yang diminta, dengan kode status 200 (*OK*) dan pesan yang sesuai.

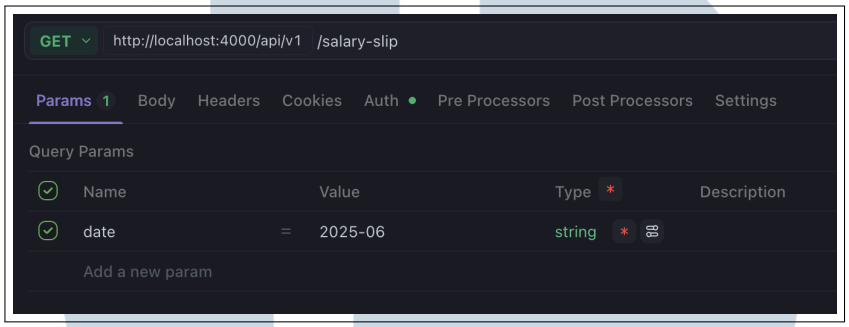


Gambar 3.61. Contoh struktur *request* untuk GET `/salary-slip/:user_id`

Gambar 3.62. Contoh struktur *response* untuk GET `/salary-slip/:user_id`

Pada gambar 3.61, terlihat struktur *request* yang dikirimkan ke *endpoint* `/salary-slip/:id`, yang berisi parameter id pegawai yang ingin diambil datanya. Selain itu, terdapat query parameter `date=YYYY-MM` yang digunakan untuk

menyaring slip gaji pada bulan dan tahun tertentu. Setelah *request* berhasil diproses, sistem akan mengembalikan respons seperti pada gambar 3.62, yang menunjukkan data slip gaji pegawai berdasarkan id pegawai tersebut pada periode waktu yang diminta, dengan kode status 200 (*OK*) dan pesan yang sesuai.



Gambar 3.63. Contoh struktur *request* untuk GET `/salary-slip`

```
{
    "code": 200,
    "message": "Salary slip generated successfully",
    "data": {
        "user_data": {
            "id": "71f01488-5f38-4c1f-8d33-be176a03d8a1",
            "fullname": "Hiroshi Tanaka",
            "is_active": true,
            "supervisor_id": null,
            "job_title": "Project Manager",
            "dob": "1985-11-11T00:00:00.000Z",
            "employment_status_id": "bf6be756-3be8-41e0-9dae-fff2824e340f",
            "daily_status": null,
            "contact": "8012345678",
            "emergency_contact": "8087654321",
            "employee_number": "1",
            "email": "hiroshi.tanaka@concise.co.id",
            "nik": "1234567890123456",
            "npwp": "123456789012345",
            "gender": "male",
            "bank_id": "4bf72b76-9635-4e52-a0f0-163d146fe47b",
            "bank_name": null,
            "bank_number": "1234567890",
            "bank_branch": "Tokyo Central Branch",
            "bank_holder": "Hiroshi Tanaka",
            "emergency_contact_holder": "Taro Yamada",
            "active_payroll_id": "588eb642-4d3b-4040-8a27-fc37b39c16da",
            "employment_status": {
                "id": "bf6be756-3be8-41e0-9dae-fff2824e340f",
                "employment_type": "Full Time"
            },
            "bank": {
                "id": "4bf72b76-9635-4e52-a0f0-163d146fe47b",
                "name": "Bank Central Asia"
            },
            "salary": [
                {
                    "amount": "6000000",
                    "created_at": "2025-06-18T10:18:14.616Z"
                }
            ],
            "payroll_application": [
                {
                    "id": "3b8a8cb7-fdcf-461a-b420-5936b7aee096",
                    "payroll_configuration_id": "588eb642-4d3b-4040-8a27-fc37b39c16da"
                }
            ]
        },
        "allowances": [
            {
                "name": "THR",
                "amount": 2000000,
                "is_deduction": false
            },
            {
                "name": "Tunjangan Makan",
                "amount": 1000000,
                "is_deduction": false
            },
            {
                "name": "Tambahan untuk Hiroshi",
                "amount": 10000000,
                "is_deduction": false
            },
            {
                "name": "Merusak kursi kantor",
                "amount": 700000,
                "is_deduction": true
            }
        ],
        "salary_data": {
            "basic_salary": 6000000,
            "total_allowances": 19000000,
            "total_deductions": 700000,
            "take_home_pay": 18300000
        },
        "period": {
            "start_date": "2025-05-25T00:00:00.000Z",
            "end_date": "2025-06-24T23:59:59.999Z"
        },
        "date": "June 2025"
    }
}
```


Gambar 3.64. Contoh struktur *response* untuk GET `/salary-slip?date=YYYY-MM`

Pada gambar 3.63, terlihat struktur *request* yang dikirimkan ke *endpoint* `/salary-slip`, yang berisi query parameter `date=YYYY-MM` untuk menyaring slip gaji pada bulan dan tahun tertentu. Setelah *request* berhasil diproses, sistem akan

mengembalikan respons seperti pada gambar 3.64, yang menunjukkan data slip gaji pengguna yang sedang aktif pada periode waktu yang diminta, dengan kode status 200 (*OK*) dan pesan yang sesuai.

3.7 Hasil Implementasi

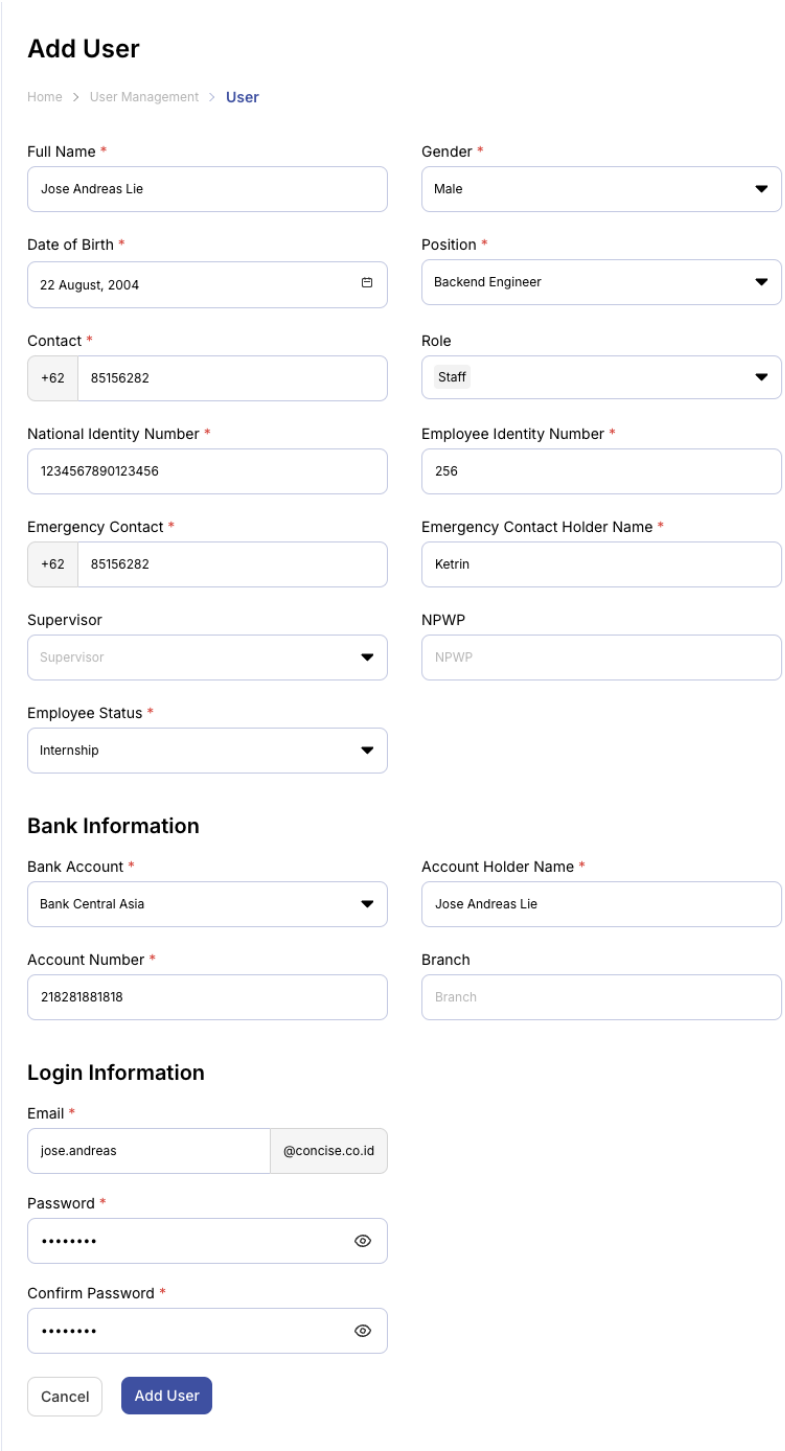
Bagian ini menjelaskan hasil implementasi dari pengembangan *backend* yang telah diintegrasikan dengan antarmuka pengguna (*frontend*). Implementasi meliputi fitur-fitur baru dan perbaikan pada sistem yang sebelumnya telah ada, termasuk modul *User Management*, *Leave Permit*, dan *Payroll*. Seluruh integrasi diuji secara manual, dan tangkapan layar (*screenshot*) berikut ini menggambarkan respons visual dari sistem setelah proses integrasi dilakukan.

3.7.1 User Management

Modul *User Management* telah diperbarui dengan validasi input form yang lebih ketat untuk memastikan konsistensi dan keakuratan data pegawai. Validasi mencakup beberapa aturan berikut:

- Email wajib menggunakan domain `@concise.co.id`
- Nomor kontak atau kontak darurat hanya diperbolehkan angka dengan awalan `+62` atau `0`
- Nama lengkap dan nama pemegang rekening bank harus berupa karakter alfabet atau alfanumerik
- NIK terdiri dari 16 digit numerik
- Field wajib lainnya tidak boleh kosong

Validasi juga diterapkan pada sisi *frontend* untuk memberikan umpan balik secara langsung kepada pengguna saat mengisi formulir. Apabila terdapat isian yang tidak valid, sistem akan menolak penyimpanan data dan menampilkan pesan kesalahan yang sesuai. Pendekatan ini bertujuan untuk meminimalkan kesalahan input yang berpotensi mengganggu proses bisnis. Namun, untuk skenario tertentu yang tidak dapat ditangani oleh validasi *frontend*, proses validasi tetap dilanjutkan pada sisi *backend* guna menjamin integritas dan konsistensi data secara menyeluruh.



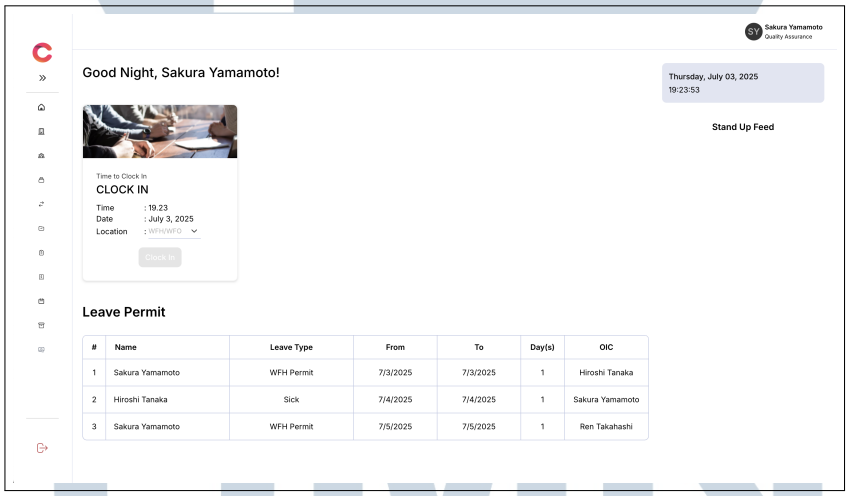
Gambar 3.65. Validasi Form Input pada Modul User Management

3.7.2 Leave Permit

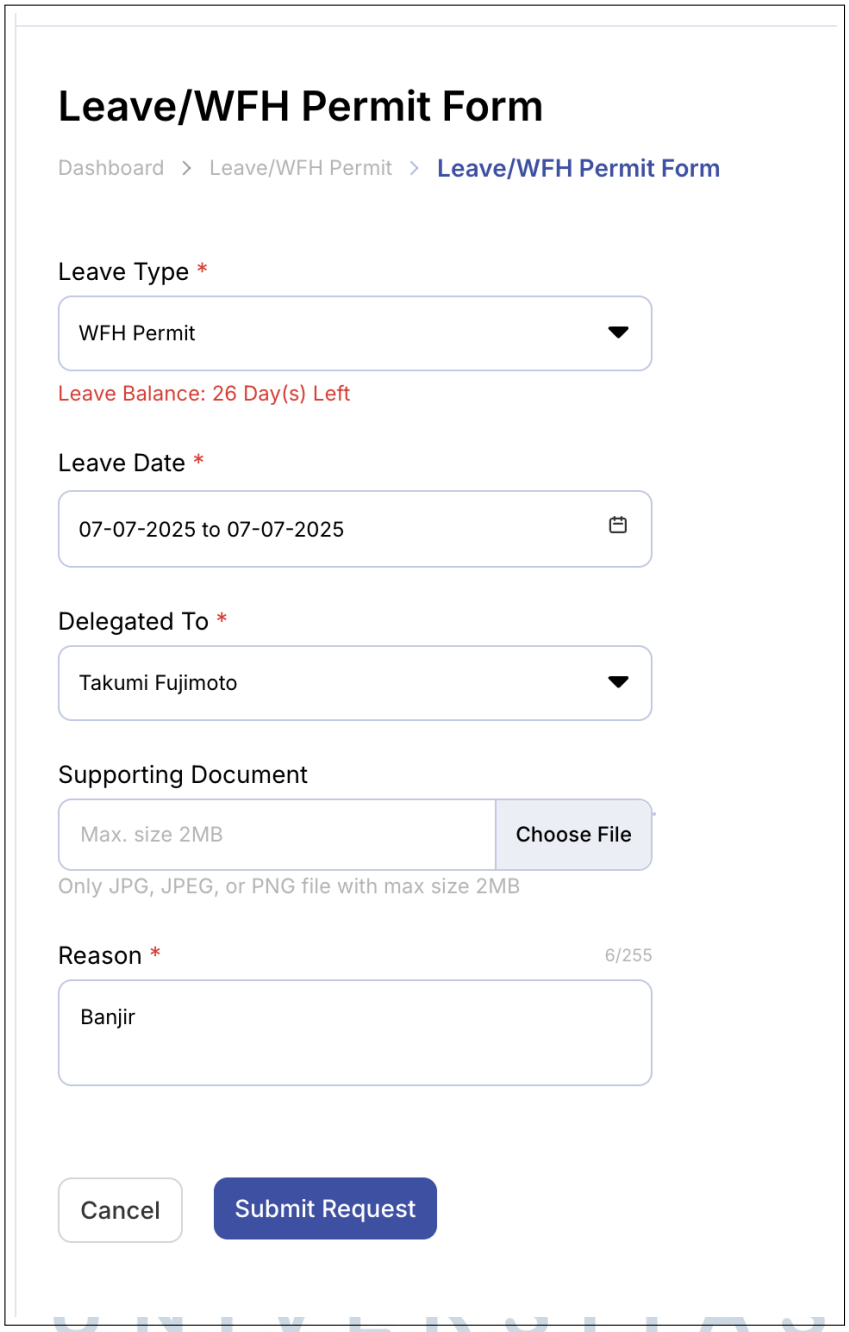
Modul *Leave Permit* telah diperbarui untuk meningkatkan pengalaman pengguna dalam mengajukan permohonan cuti. Perubahan utama meliputi:

- Penambahan fitur untuk menampilkan daftar pegawai yang melakukan cuti per minggu dari sekarang.
- Penambahan field untuk memilih officer in charge yang berlaku sebagai pengganti dari pegawai yang sedang cuti untuk kebutuhan bisnis.
- Penambahan fitur untuk membatalkan permohonan yang belum disetujui.

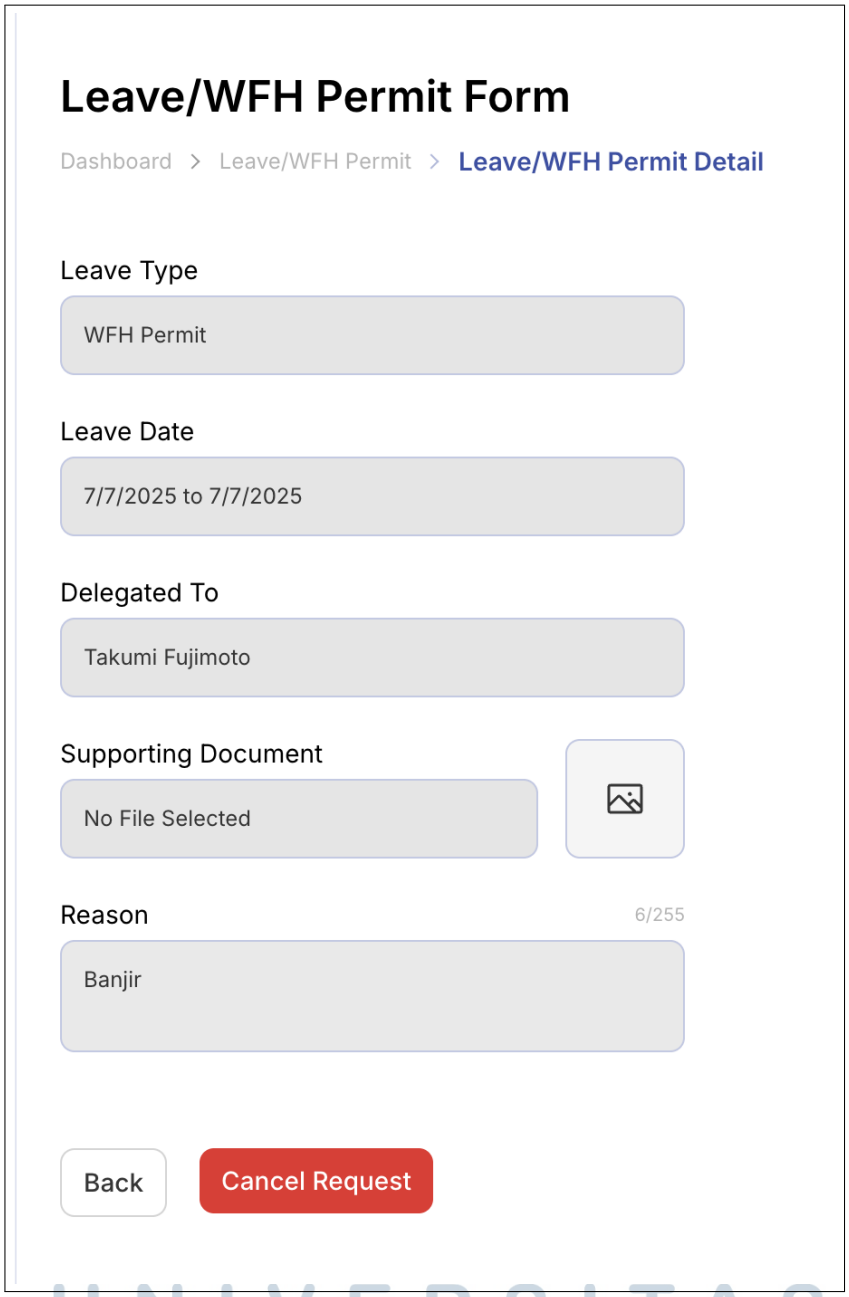
Contoh tampilan antarmuka pengguna untuk modul *Leave Permit* dapat dilihat pada gambar 3.66, gambar 3.68, dan gambar 3.67.



Gambar 3.66. Tampilan Antarmuka Pengguna untuk Modul Leave Permit pada dashboard



Gambar 3.67. Tampilan Antarmuka Pengguna untuk Modul Leave Permit pada form pengajuan cuti



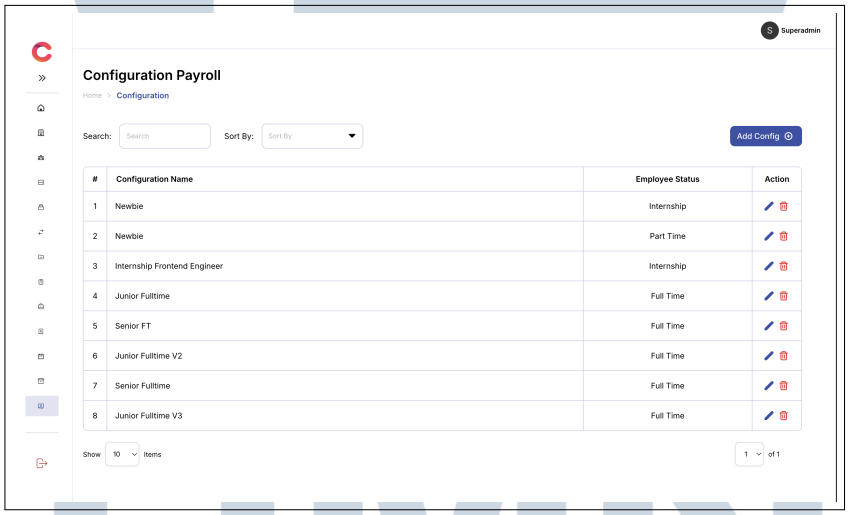
Gambar 3.68. Tampilan Antarmuka Pengguna untuk Modul Leave Permit pada form pembatalan permohonan cuti

3.7.3 Payroll

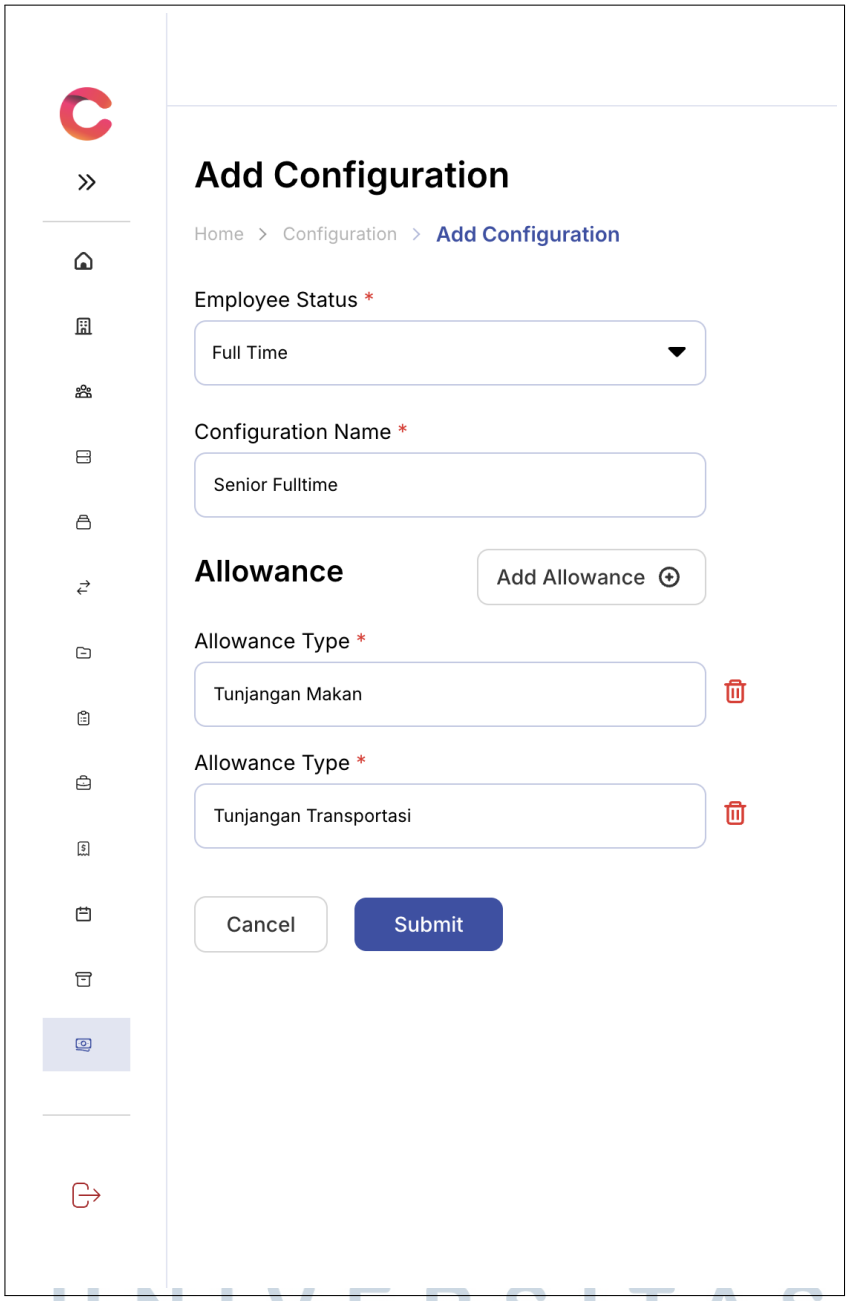
Modul *Payroll* telah diperbarui dengan penambahan fitur-fitur baru yang mendukung pengelolaan gaji pegawai secara lebih efisien. Fitur-fitur baru yang ditambahkan meliputi:

- Penambahan konfigurasi penggajian yang dapat disesuaikan dengan kebutuhan perusahaan, termasuk tunjangan-tunjangan yang dapat ditambahkan atau dihapus sesuai dengan kebijakan perusahaan.
- Penambahan fitur untuk menyimpan data pegawai, gaji, dan tunjangan-tunjangan dalam satu proses, sehingga memudahkan pengelolaan data pegawai.
- Penambahan fitur untuk menghasilkan slip gaji pegawai secara otomatis berdasarkan konfigurasi penggajian yang telah ditentukan.

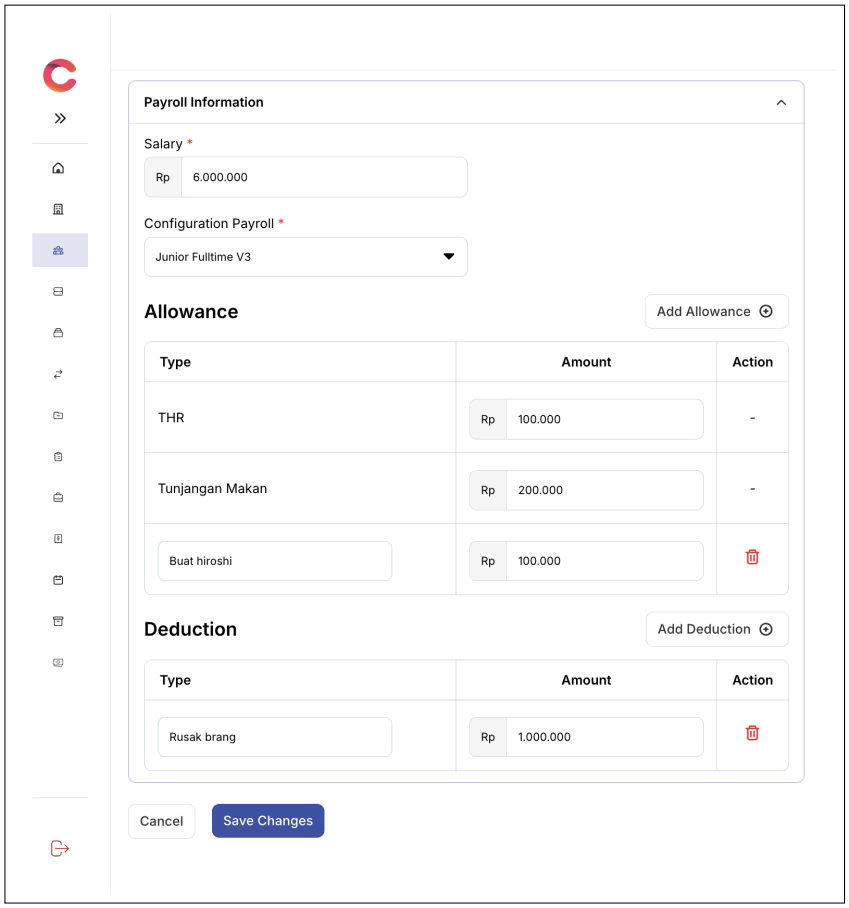
Contoh tampilan antarmuka pengguna untuk modul *Payroll* dapat dilihat pada gambar 3.69, gambar 3.70, gambar 3.71, gambar 3.72, dan gambar 3.73.



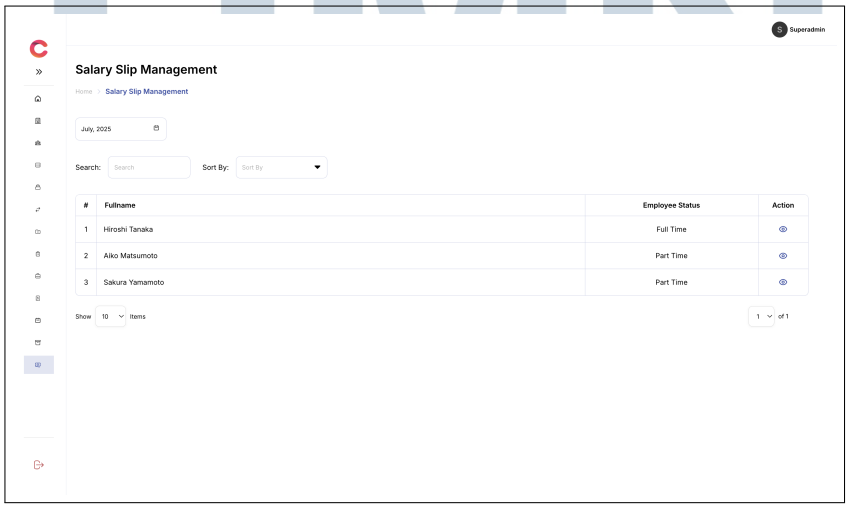
Gambar 3.69. Tampilan Antarmuka Pengguna untuk Modul Payroll pada daftar konfigurasi penggajian



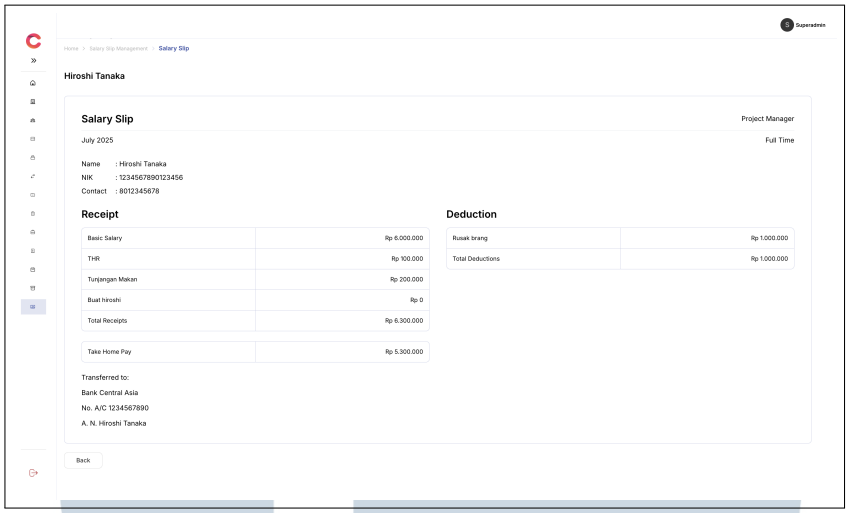
Gambar 3.70. Tampilan Antarmuka Pengguna untuk Modul Payroll pada form konfigurasi penggajian



Gambar 3.71. Tampilan Antarmuka Pengguna untuk Modul Payroll pada informasi gaji pegawai



Gambar 3.72. Tampilan Antarmuka Pengguna untuk Modul Payroll pada daftar slip gaji pegawai



Gambar 3.73. Tampilan Antarmuka Pengguna untuk Modul Payroll pada slip gaji pegawai

3.8 Kendala dan Solusi yang Ditemukan

3.8.1 Kendala

Selama pelaksanaan kerja praktik, ditemukan beberapa kendala teknis dan struktural dalam proses pengembangan sistem, yaitu sebagai berikut:

1. Beberapa bagian kode tidak mengikuti prinsip *best practice* dalam pengembangan backend, sehingga menyulitkan proses pengembangan lanjutan dan pemeliharaan sistem.
2. Terdapat penggunaan nilai yang masih ditulis secara *hardcoded*, seperti *job title*, yang seharusnya dikelola melalui relasi ke dalam basis data agar lebih fleksibel dan dapat disesuaikan.
3. Modul Payroll memiliki kompleksitas tinggi, khususnya dalam pengambilan data historis pengguna, yang menjadi tantangan dalam perhitungan gaji berdasarkan riwayat data.
4. Proses *migration* dan *seeding* data belum optimal dan membutuhkan waktu lama karena struktur data yang belum efisien.

3.8.2 Solusi

Berikut adalah langkah-langkah solusi yang dilakukan untuk mengatasi kendala-kendala tersebut:

1. Melakukan proses *code review* secara berkala bersama tim *backend* untuk memastikan bahwa setiap kode yang ditulis mengikuti standar *best practice* dan dapat dikembangkan dengan mudah di masa depan.
2. Melakukan *refactoring* pada bagian kode yang masih menggunakan pendekatan *hardcoded*, seperti pengelolaan *job title*, dengan menggantinya menjadi referensi ke tabel *job_titles*.
3. Menambahkan tabel transaksi (*transaction table*) pada modul Payroll untuk menyimpan data historis penggajian secara terpisah, sehingga proses perhitungan gaji dapat dilakukan secara lebih akurat dan efisien.
4. Mengoptimalkan proses *migration* dan *seeding* dengan menyusun ulang struktur data serta membuat script yang lebih modular untuk mempercepat inisialisasi sistem.