

# Umn Libtii TI 3

## MBKM\_JOSE\_ANDREAS\_LIE

-  JOSE ANDREAS LIE
-  2025 GANJIL - MAGANG (REGULER & MBKM) TEKNIK INFORMATIKA
-  Universitas Multimedia Nusantara

### Document Details

**Submission ID**

trn:oid:::1:3451387202

78 Pages

**Submission Date**

Dec 29, 2025, 6:19 PM GMT+7

9,651 Words

**Download Date**

Dec 29, 2025, 6:24 PM GMT+7

62,290 Characters

**File Name**

MBKM\_JOSE\_ANDREAS\_LIE.pdf

**File Size**

5.5 MB

# 17% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Filtered from the Report

- ▶ Bibliography
  - ▶ Quoted Text
- 

## Top Sources

17%	 Internet sources
0%	 Publications
0%	 Submitted works (Student Papers)

---

## Top Sources

- 17% Internet sources  
0% Publications  
0% Submitted works (Student Papers)
- 

## Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

Rank	Type	Source	Percentage
1	Internet	kc.umn.ac.id	16%
2	Internet	123dok.com	<1%
3	Internet	docplayer.info	<1%
4	Internet	repositori.kemdikbud.go.id	<1%
5	Internet	e-procurement.telin.co.id	<1%
6	Internet	pustikom.uinssc.ac.id	<1%
7	Internet	sis.binus.ac.id	<1%
8	Internet	blog.w3loker.com	<1%
9	Internet	falseidlepunk.com	<1%
10	Internet	fmipa-unpak.academia.edu	<1%
11	Internet	islamicmarkets.com	<1%

12 Internet

lokeroilgas.com <1%

13 Internet

qdoc.tips <1%

14 Internet

www.scribd.com <1%

15 Internet

www.theseus.fi <1%

1

# PENGEMBANGAN BACKEND SISTEM INFORMASI KEPEGAWAIAN NINE TO SIX PADA PT VISI KARYA NUSANTARA

Jose Andreas Lie

## ABSTRAK

Laporan ini menjelaskan pengembangan sistem informasi kepegawaian Nine to Six (NTS) di PT Visi Karya Nusantara, sebuah perusahaan konsultan teknologi informasi yang berfokus pada solusi digital inovatif. Digitalisasi pengelolaan sumber daya manusia (SDM) telah menjadi kebutuhan kritis untuk meningkatkan efisiensi operasional, akurasi penggajian, dan kepatuhan regulasi. Penelitian menunjukkan bahwa integrasi modul-modul seperti *Contract Management*, *Overtime Management*, dan *Payslip* secara signifikan mengurangi kesalahan manual dan meningkatkan transparansi data. Proyek ini dirancang untuk mengembangkan sistem terpadu yang menghubungkan data statis (kontrak karyawan), data transaksional (kehadiran dan lembur), dan proses finalisasi kompensasi (*payslip*) dalam satu platform berbasis *cloud*. Menggunakan PERN Stack (PostgreSQL, Express.js, React.js, Node.js), sistem dikembangkan dengan metodologi *Agile* selama periode magang dari 4 Agustus 2025 hingga 4 Januari 2026. Hasil pengembangan mencakup implementasi modul *Contract Management*, *Payroll* dengan komponen otomatis, *Overtime Management*, *On Boarding* berbasis CSV, serta *Payslip* terintegrasi dengan sistem *Daily Attendance*. Sistem ini diharapkan meningkatkan efisiensi HR, menjamin akurasi perhitungan kompensasi, dan memberikan transparansi penuh kepada karyawan melalui akses *self-service*.

8

1

1

1

**Kata Kunci:** *Backend development, Express, Human resource information system, On boarding, Payroll.*

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

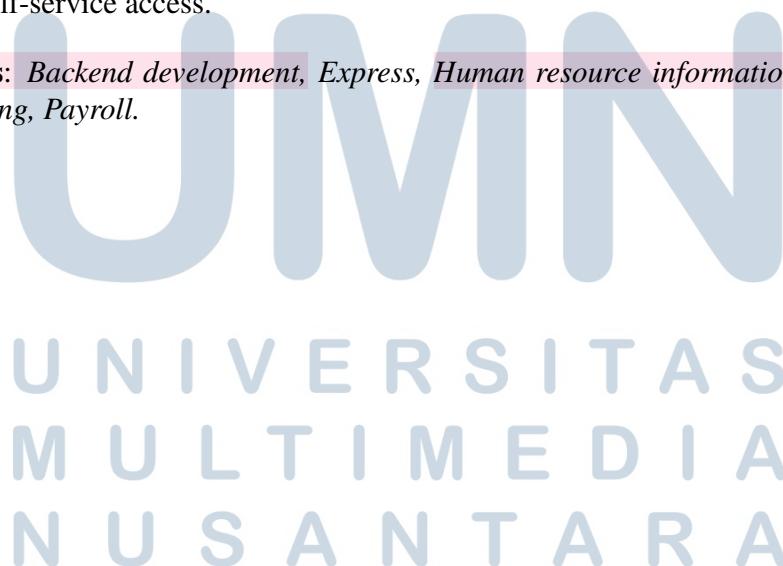
**BACKEND ENHANCEMENT OF THE HR INFORMATION SYSTEM AT PT  
VISI KARYA NUSANTARA**

Jose Andreas Lie

**ABSTRACT**

This report describes the development of the Nine to Six (NTS) human resource information system at PT Visi Karya Nusantara, a technology information consulting company focused on innovative digital solutions. The digitalization of human resource management has become a critical necessity to improve operational efficiency, payroll accuracy, and regulatory compliance. Research demonstrates that the integration of modules such as Contract Management, Overtime Management, and Payslip significantly reduces manual errors and enhances data transparency. This project was designed to develop an integrated system that connects static data (employee contracts), transactional data (attendance and overtime), and compensation finalization processes (payslip) within a single cloud-based platform. Using the PERN Stack (PostgreSQL, Express.js, React.js, Node.js), the system was developed with Agile methodology during the internship period from August 4, 2025 to January 4, 2026. The development outcomes include the implementation of Contract Management module, Payroll with automated components, Overtime Management, CSV-based On Boarding, and Payslip integrated with the Daily Attendance system. This system is expected to enhance HR efficiency, ensure accurate compensation calculations, and provide full transparency to employees through self-service access.

**Keywords:** *Backend development, Express, Human resource information system, On boarding, Payroll.*



## BAB 1

### PENDAHULUAN

#### 1.1 Latar Belakang Masalah

Transformasi digital dalam pengelolaan sumber daya manusia (SDM) telah menjadi salah satu faktor utama dalam peningkatan efisiensi operasional perusahaan di era industri 4.0. Penerapan sistem informasi kepegawaian atau *Human Resource Information System* (HRIS) memungkinkan proses administrasi dan pengambilan keputusan berbasis data dilakukan secara lebih cepat dan akurat [1]. Digitalisasi fungsi SDM tidak hanya berdampak pada efisiensi kerja, tetapi juga membuka potensi untuk peningkatan transparansi dan konsistensi data pegawai [2].

Salah satu aspek penting dari digitalisasi SDM adalah otomatisasi proses penggajian atau *payroll automation*. Menurut penelitian oleh Onuotu dan Amaewhule (2024), penerapan sistem akuntansi terotomatisasi pada konteks usaha kecil dan menengah secara signifikan dapat meningkatkan efisiensi dan akurasi dalam persiapan penggajian, dengan mencatatkan pengurangan kesalahan perhitungan dan penurunan beban kerja manual [3]. Hal ini mengindikasikan bahwa, terutama dalam skenario transisi dari proses manual, integrasi sistem penggajian digital sangat efektif dalam mempercepat proses administrasi. Meskipun demikian, perlu dicatat bahwa otomatisasi *payroll* berfokus pada akurasi perhitungan dan tidak serta merta menjamin validitas data masukan, menyoroti pentingnya keakuratan data sumber.

Lebih lanjut, menurut Meenugu (2025), evolusi sistem penggajian dari proses manual menuju otomatisasi cerdas tidak hanya meningkatkan efisiensi dan akurasi, tetapi juga memperkuat kepatuhan terhadap regulasi [4]. Meskipun demikian, masih banyak sistem penggajian yang berjalan secara terpisah dari modul HRIS utama, sehingga menimbulkan inefisiensi akibat kebutuhan untuk memasukkan data secara berulang dari sistem lain seperti absensi dan manajemen cuti. Kondisi ini kembali membuka peluang terjadinya kesalahan manusia dan menghambat efektivitas kerja. Oleh karena itu, diperlukan integrasi yang mulus antara pengelolaan data pegawai dan proses penggajian dalam satu platform terpadu.

Perhitungan variabel lembur dan absensi menjadi sumber kerumitan dan potensi sengketa. Tanpa sistem yang terintegrasi, penetapan jam lembur yang akurat

memerlukan waktu dan rentan terhadap kesalahan manual atau manipulasi data (*buddy punching*). Modul Lembur dan Absensi (*Overtime and Absence Module*) dalam HRIS modern berfungsi sebagai alat untuk memastikan pencatatan waktu yang tepat dan otomatis. Penggunaan sistem pencatatan waktu kerja elektronik kini dianggap sebagai komponen penting untuk memastikan kepatuhan terhadap regulasi jam kerja (*working time regulations*), karena memungkinkan pemantauan jam kerja yang akurat dan efisien [5]. Data dari modul ini sangat vital karena merupakan input transaksional yang mempengaruhi Modul Payslip.

Integrasi dan akurasi data yang diupayakan oleh modul variabel seperti lembur dan penggajian bergantung secara fundamental pada ketersediaan data master pegawai yang konsisten dan terstandardisasi [2]. Oleh karena itu, Modul Kontrak Karyawan (*Employee Contract Module*) harus berfungsi sebagai basis data primer di dalam HRIS, yang menyimpan informasi krusial seperti struktur gaji dasar, tunjangan tetap, periode kontrak, dan ketentuan jadwal kerja. Konsep ini sesuai dengan evolusi HRIS modern berbasis komputasi awan (*cloud computing*) yang dirancang untuk menjadi sumber tunggal kebenaran (*single source of truth*) bagi seluruh data kepegawaian organisasi [6]. Kegagalan dalam sentralisasi ini, seperti yang terjadi pada sistem tradisional, menyebabkan data kepegawaian tersebar dalam dokumen fisik atau file terisolasi, yang menyulitkan pembaruan serentak dan validasi silang data.

Digitalisasi Modul Kontrak memiliki peran ganda sebagai fondasi data dan sebagai penjaga kepatuhan. Secara inheren, modul ini berfungsi untuk mengeliminasi risiko ketidakpatuhan hukum (*legal compliance risk*) yang timbul dari *template* kontrak yang tidak seragam atau klausul yang sudah usang. Dengan mengimplementasikan sistem yang mengacu pada kebijakan perusahaan dan regulasi ketenagakerjaan yang berlaku, HRIS dapat memastikan standardisasi kontrak bagi seluruh karyawan. Selain itu, sentralisasi data kontrak ini memungkinkan integrasi yang mulus antara data personalia dan database penggajian, sebuah faktor kritis yang telah terbukti meningkatkan transparansi dan akuntabilitas dalam pengelolaan kompensasi [7]. Hal ini menyoroti bahwa kualitas informasi SDM sangat bergantung pada seberapa baik data kontrak dikelola sejak awal.

Setelah data dasar (Data Statis), seperti yang terdapat di Modul Kontrak, dan data variabel (Data Transaksional), seperti yang diperoleh dari Modul Lembur, Absensi, dan Reimbursement berhasil dikumpulkan dan divalidasi, tantangan selanjutnya adalah pada proses finalisasi kompensasi, yang diwujudkan dalam

Modul Payslip Karyawan (*Payslip Module*). Proses pembuatan payslip secara manual adalah tugas yang sangat kompleks dan rentan kesalahan, karena harus menggabungkan perhitungan gaji pokok, tunjangan, potongan pajak, iuran wajib (statutory deduction), serta komponen variabel seperti lembur dan keterlambatan. Studi kasus menunjukkan bahwa sistem perhitungan gaji manual menghadapi inefisiensi yang signifikan [8]. Oleh karena itu, digitalisasi dan otomatisasi modul ini sangat penting untuk memastikan akurasi perhitungan dan efektivitas [9].

Peran Modul Payslip digital tidak hanya terbatas pada pencetakan dokumen, tetapi sebagai titik akhir validasi data. Dengan mengadopsi sistem berbasis web terintegrasi, Modul Payslip menyediakan audit trail yang jelas, memungkinkan HR untuk meninjau secara rinci sumber data dari setiap komponen gaji [10]. Bagi karyawan, modul ini meningkatkan transparansi karena mereka dapat mengakses slip gaji mereka secara mandiri (self-service) dan melihat perincian potongan atau penambahan secara real-time. Adopsi sistem HRIS seperti ini merupakan revolusi yang diadopsi korporasi untuk meningkatkan operasi harianya [9], yang pada akhirnya meningkatkan kepercayaan dan pengalaman pegawai (employee experience) secara keseluruhan [11].

Meskipun berbagai literatur dan implementasi HRIS telah menunjukkan manfaat signifikan dari digitalisasi fungsi SDM secara terpisah—baik itu efisiensi penggajian [3] maupun kepatuhan manajemen waktu [5] masih terdapat kesenjangan operasional dalam realisasi sistem yang benar-benar terintegrasi dan berpusat pada data master. Banyak organisasi masih menghadapi kesulitan dalam menyinkronkan data statis dari Modul Kontrak dengan data transaksional dari Modul Lembur dan Absensi untuk diolah di Modul Payslip, menunjukkan adanya *silo* data yang menghambat otomatisasi menyeluruh [9]. Kegagalan dalam mengintegrasikan data personalia dan *database* penggajian, misalnya, merupakan tantangan utama yang harus diatasi untuk menjamin transparansi dan akuntabilitas sistem kompensasi [7].

Tantangan integrasi ini menuntut adanya perancangan HRIS yang tidak hanya menyediakan fungsi-fungsi tersebut, tetapi juga berfokus pada arsitektur data tunggal yang mulus (*single source of truth*). Konsep arsitektur ini, yang didukung oleh sistem berbasis *cloud* modern, menjadi kunci untuk mendapatkan informasi SDM yang konsisten dan akurat [6]. Proyek perancangan ini hadir untuk menjembatani kesenjangan operasional tersebut. Tujuan dari perancangan sistem ini adalah untuk mengembangkan dan mengimplementasikan modul HRIS yang mengintegrasikan Modul Kontrak, Modul Lembur, dan Modul Payslip dalam satu

platform terpadu. Integrasi ini diharapkan dapat meningkatkan efisiensi operasional HR secara signifikan, memastikan keakuratan data master, menjamin kepatuhan perhitungan kompensasi, dan menyediakan transparansi penuh kepada karyawan, sehingga mendukung pengambilan keputusan strategis yang lebih baik.

## 1 1.2 Maksud dan Tujuan Kerja Magang

Pelaksanaan kerja magang di PT Visi Karya Nusantara bertujuan untuk memberikan pengalaman langsung kepada mahasiswa dalam menghadapi tantangan nyata di industri pengembangan perangkat lunak, serta berkontribusi terhadap transformasi digital di bidang pengelolaan sumber daya manusia. Melalui kegiatan magang ini, peserta memperoleh kesempatan untuk menerapkan pengetahuan akademik ke dalam praktik profesional, khususnya dalam pengembangan sistem informasi kepegawaian berbasis web.

Secara khusus, maksud dari pelaksanaan kerja magang ini adalah:

- Mengimplementasikan pengetahuan yang telah diperoleh selama perkuliahan ke dalam lingkungan kerja profesional yang menerapkan standar industri.
- Mengasah keterampilan teknis dalam pengembangan sistem berbasis *PERN stack* (PostgreSQL, Express.js, React, Node.js) serta kemampuan kolaborasi dalam tim lintas divisi.
- Memahami proses kerja profesional dalam pengembangan dan pemeliharaan sistem *backend* yang terstruktur, terdokumentasi, dan terintegrasi dengan sistem lainnya.

Adapun tujuan utama dari pelaksanaan kerja magang ini, yang difokuskan pada pengembangan sistem Nine to Six (NTS), meliputi:

1. Merancang dan mengimplementasikan modul Contract Management untuk mendukung pembuatan serta penugasan kontrak kerja kepada karyawan secara efisien.
2. Merancang dan mengimplementasikan modul Payslip Management yang terintegrasi dengan modul Daily Attendance, sehingga proses perhitungan gaji dapat dilakukan secara otomatis berdasarkan data kehadiran.

3. Menyediakan fitur pengurangan otomatis terhadap ketidakhadiran tanpa keterangan (*Missing in Action (MIA)*) agar perhitungan penggajian menjadi lebih akurat dan transparan.
4. Meningkatkan integrasi sistem antara sisi *backend* dan *frontend* untuk memastikan alur data yang konsisten dan *real-time*.
5. Mengembangkan *Application Programming Interface (API)* terstandarisasi yang dapat digunakan oleh HR maupun karyawan untuk mengakses data penggajian secara aman dan terukur.

Dengan pencapaian tujuan-tujuan tersebut, diharapkan sistem Nine to Six (NTS) dapat menjadi fondasi penting dalam mendukung efisiensi operasional perusahaan serta meningkatkan transparansi dan akurasi dalam pengelolaan penggajian di PT Visi Karya Nusantara.

### 1.3 Waktu dan Prosedur Pelaksanaan Kerja Magang

Pelaksanaan kerja magang di PT Visi Karya Nusantara (NTS) berlangsung dari tanggal 4 Agustus 2025 sampai dengan 4 Januari 2026 berdasarkan kontrak kerja yang telah disepakati dengan perusahaan. Selama periode magang ini, penulis dibimbing oleh Bapak Raditya selaku *Supervisor* dalam tim *Software Development*. Program ini dirancang untuk memberikan pengalaman langsung dalam proses pengembangan perangkat lunak profesional, termasuk pemahaman terhadap alur kerja kolaboratif dan siklus pengembangan produk di lingkungan industri.

Jadwal dan ketentuan kerja magang di PT Visi Karya Nusantara diatur sebagai berikut:

1. Aktivitas kerja magang dilaksanakan secara *remote*, dengan total waktu kerja sebesar 35 jam per minggu.
2. Waktu kerja bersifat fleksibel namun tetap mengikuti tanggung jawab proyek dan jadwal yang telah disepakati bersama *supervisor*.
3. Seluruh aktivitas kerja dilakukan secara daring dengan koordinasi melalui *platform* komunikasi Discord.

Selama menjalani program kerja magang, peserta wajib mengikuti sejumlah prosedur yang telah ditetapkan oleh perusahaan, antara lain:

1. Mengikuti sesi *onboarding* dan *orientation meeting* pada awal masa magang untuk memahami sistem kerja dan proyek yang akan dikerjakan.
2. Melakukan pelaporan harian melalui *daily updates* yang mencakup tugas yang telah diselesaikan (*yesterday tasks*), rencana pekerjaan (*today tasks*), serta hambatan yang dihadapi (*blocking issues*).
3. Berpartisipasi dalam *sprint meeting* mingguan untuk melaporkan progres, melakukan evaluasi hasil kerja, serta membahas rencana pengembangan modul selanjutnya.
4. Melakukan komunikasi aktif dengan tim pengembang lain, baik pada sisi *backend*, *frontend* maupun *design*, guna memastikan integrasi sistem berjalan dengan optimal.
5. Menyerahkan laporan akhir dan hasil proyek kepada *supervisor* pada akhir periode magang sebagai bentuk evaluasi kinerja dan kontribusi selama program berlangsung.



1

## BAB 2

### GAMBARAN UMUM PERUSAHAAN

#### 2.1 Sejarah Singkat Perusahaan

PT Visi Karya Nusantara didirikan pada tahun 2025, berdasarkan informasi yang diperoleh melalui wawancara langsung dengan Bapak Raditya selaku *Software Engineer Manager*. Sebagai perusahaan konsultan teknologi informasi, PT Visi Karya Nusantara berfokus pada penyediaan solusi IT yang inovatif dan efektif bagi berbagai klien di Indonesia. Dalam kurun waktu yang singkat, perusahaan telah berhasil membangun reputasi yang solid di industri ini dengan menyelesaikan berbagai proyek menantang dan memberikan nilai tambah yang signifikan. Berlokasi di Start Space Coworking Space Gading Serpong, Tangerang, PT Visi Karya Nusantara saat ini aktif mengerjakan beberapa proyek utama, di antaranya adalah Nine to Six (NTS), AKS, dan KAIYA [12]. Logo resmi perusahaan dapat dilihat pada Gambar 2.1.



Gambar 2.1. Logo Perusahaan PT Visi Karya Nusantara

Gambar 2.1 merupakan logo dari perusahaan PT Visi Karya Nusantara. Berdasarkan wawancara langsung dengan Bapak Raditya selaku *Software Engineer Manager*,

#### 2.2 Visi dan Misi Perusahaan

Visi dan misi perusahaan PT Visi Karya Nusantara diperoleh melalui wawancara langsung dengan Bapak Raditya selaku *Software Engineer Manager*.

Visi dari PT Visi Karya Nusantara adalah:

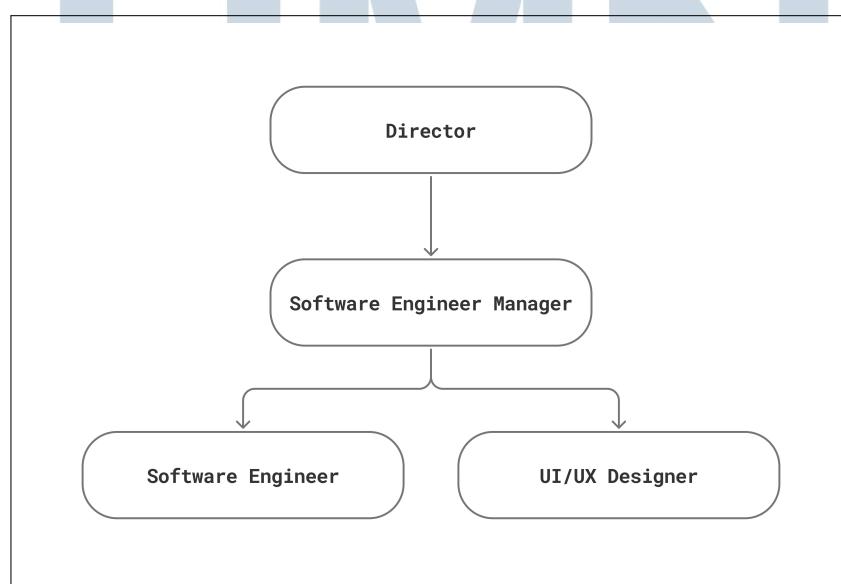
*"Menjadi perusahaan konsultan teknologi informasi yang memberikan solusi yang optimal, efektif, dan efisien, serta membangun budaya perusahaan yang berintegritas, profesional, dan berkelanjutan."*

Misi dari PT Visi Karya Nusantara adalah:

1. Menyediakan layanan konsultasi teknologi informasi yang berorientasi pada efektivitas dan efisiensi, guna memastikan setiap solusi yang diberikan selaras dengan kebutuhan dan tujuan klien.
2. Membangun organisasi yang profesional dan berintegritas melalui pengembangan sumber daya manusia yang adaptif, kompeten, dan berdaya saing, serta menumbuhkan budaya kerja yang kolaboratif, inovatif, dan berkelanjutan.
3. Berperan aktif dalam mendukung transformasi digital di Indonesia dengan mengedepankan praktik konsultasi yang beretika, berkualitas, dan berorientasi pada hasil jangka panjang.

### 2.3 Struktur Organisasi Perusahaan

Struktur organisasi perusahaan PT Visi Karya Nusantara dapat dilihat pada Gambar 2.2.



Gambar 2.2. Struktur organisasi perusahaan PT Visi Karya Nusantara

Struktur organisasi PT Visi Karya Nusantara terdiri atas tiga tingkatan utama yang membentuk alur koordinasi kerja secara efektif dan terarah. Pada tingkat tertinggi, terdapat Director, yang memegang tanggung jawab utama dalam menentukan visi, arah strategis, serta pengambilan keputusan penting perusahaan.

Di bawahnya terdapat Software Engineer Manager, yang berperan dalam mengatur dan mengawasi seluruh kegiatan teknis yang berkaitan dengan pengembangan perangkat lunak. Posisi ini juga bertugas untuk memberikan arahan teknis, melakukan evaluasi hasil kerja tim, serta memastikan proyek pengembangan berjalan sesuai standar dan tengat waktu yang telah ditetapkan.

Selanjutnya, di bawah koordinasi *Software Engineer Manager*, terdapat dua peran penting yaitu Software Engineer dan UI/UX Designer. *Software Engineer* bertanggung jawab atas proses pengembangan aplikasi, mulai dari implementasi fitur hingga integrasi sistem. Sementara itu, *UI/UX Designer* berfokus pada perancangan antarmuka pengguna serta pengalaman pengguna agar produk yang dikembangkan tidak hanya berfungsi dengan baik, tetapi juga memiliki tampilan yang menarik dan mudah digunakan.

Struktur organisasi ini menggambarkan alur kerja yang ringkas namun efektif, dengan fokus pada kolaborasi lintas peran untuk menghasilkan produk digital yang berkualitas tinggi dan sesuai dengan kebutuhan pengguna.

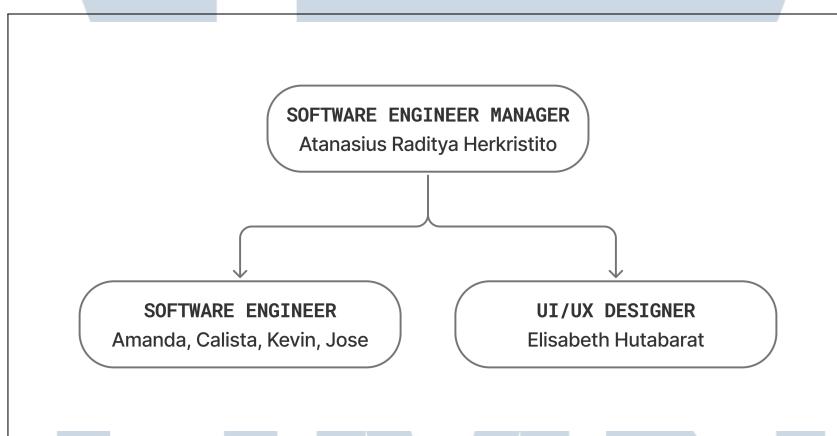


## BAB 3

### PELAKSANAAN KERJA MAGANG

#### 3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang di PT Visi Karya Nusantara (NTS), penulis menempati posisi sebagai *Software Engineer Intern* dalam *Software Development Team*. Peran utama yang dijalankan berfokus pada pengembangan dan pengujian *Application Programming Interface (API)* menggunakan *Node.js* dan *Express.js*, serta integrasinya dengan basis data *PostgreSQL*. Selain itu, penulis juga berkolaborasi dengan tim *frontend* dalam membangun dan mengimplementasi modul *Payroll* yang terhubung dengan sistem *Daily Attendance*.



Gambar 3.1. Struktur Tim Pengembang PT Visi Karya Nusantara (NTS)

Struktur tim pengembang dapat dilihat pada Gambar 3.1, yang terdiri dari beberapa anggota dengan peran yang saling terintegrasi. Tim ini berada di bawah koordinasi Bapak Atanasius Raditya Herkristito selaku *Software Engineer Manager*, yang berperan sebagai pembimbing sekaligus pengarah teknis selama masa magang. Beliau bertanggung jawab dalam memberikan umpan balik, melakukan evaluasi mingguan terhadap progres proyek, serta mengadakan sesi *code review* untuk memastikan kualitas kode sesuai standar perusahaan.

Dalam pelaksanaan proyek, kolaborasi dilakukan bersama anggota tim lainnya yang terdiri dari:

- Amanda — *Software Engineer Intern*
- Calista — *Software Engineer Intern*

- Kevin — *Software Engineer Intern*
- Elisabeth Hutabarat — *UI/UX Designer*

Seluruh kegiatan magang dilaksanakan secara *remote*, sehingga koordinasi tim dilakukan secara daring melalui *platform Discord* dan *repository management system GitHub*. Discord digunakan sebagai sarana komunikasi utama untuk berdiskusi, menyampaikan pembaruan progres, serta melakukan rapat mingguan. Sementara itu, GitHub digunakan untuk mengelola kode sumber, melakukan *pull request*, serta memfasilitasi proses *code review* dan kolaborasi antar anggota tim.

Setiap minggu diadakan *sprint meeting* yang dipimpin oleh Raditya untuk membahas perkembangan proyek, hambatan teknis, serta rencana pengembangan fitur berikutnya. Selain itu, setiap anggota tim juga melakukan *asynchronous daily updates* melalui Discord yang berisi laporan progres harian dan kendala yang dihadapi. Evaluasi hasil kerja dilakukan secara berkala melalui sesi *code review* dan *sprint retrospective* untuk memastikan kualitas pengembangan tetap terjaga serta selaras dengan standar teknis perusahaan.

### 3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di PT Visi Karya Nusantara, penulis terlibat dalam proyek pengembangan aplikasi *Internal System*, yaitu sebuah sistem kepegawaian berbasis web yang dikembangkan tidak hanya untuk kebutuhan internal perusahaan, tetapi juga ditujukan sebagai produk yang dapat digunakan oleh perusahaan lain. Fokus utama pekerjaan mencakup pengembangan sisi sistem dan integrasi antara backend dan frontend untuk mendukung fungsionalitas dan skalabilitas produk.

Tugas-tugas yang dilakukan meliputi:

1. Mengembangkan dan memelihara API untuk aplikasi *Internal System*, termasuk pembuatan fitur baru, serta memastikan kompatibilitas dengan sistem lain yang akan menggunakan produk ini.
2. Melakukan dokumentasi API menggunakan *platform Apidog* untuk mempermudah kolaborasi antar tim dan memfasilitasi integrasi dengan pihak eksternal.

3. Melakukan pengujian terhadap API dan fitur yang dikembangkan, mencakup validasi data, penanganan kesalahan, serta pengujian fungsionalitas untuk memastikan sistem berjalan stabil.
4. Melakukan koordinasi dan kolaborasi dengan tim pengembang melalui Discord dan GitHub terkait pembagian tugas, pembahasan revisi, serta proses integrasi antar modul.

Seluruh aktivitas tersebut berperan penting dalam mendukung terciptanya sistem yang andal, terukur, dan siap diimplementasikan secara luas. Dengan adanya dokumentasi dan proses pengembangan yang terstruktur, sistem ini tidak hanya memenuhi kebutuhan operasional internal, tetapi juga mampu beradaptasi dengan kebutuhan perusahaan lain yang menjadi calon pengguna produk. Hal ini menjadi bagian dari upaya strategis perusahaan dalam membangun solusi digital yang efisien dan berkelanjutan.

### 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	Melakukan perancangan awal modul <i>Payroll</i> , mencakup pembuatan skema basis data, model, serta pengembangan awal API untuk <i>Contract</i> , <i>Payslip</i> , dan <i>Payroll Setting</i> .
2	Melanjutkan pengembangan <i>Contract API</i> dengan penambahan fitur pembatalan, persetujuan, dan penghapusan kontrak, melakukan refaktor pada <i>controller</i> , serta menambahkan seeding data untuk tabel pendukung seperti <i>misc</i> , <i>penalty rules</i> , dan <i>schedules</i> .
3	Melakukan perbaikan pada <i>Contract API</i> , menambahkan fitur baru untuk <i>Job Title</i> termasuk perbaikan routing dan pembuatan API baru, serta melakukan revamp pada modul <i>Leave Permit</i> dan penyusunan ulang logika penalti pada kontrak.

Lanjut pada halaman berikutnya

1

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

Minggu Ke -	Pekerjaan yang dilakukan
4	Revamp pada modul <i>User Management</i> , <i>Leave Management</i> , dan <i>Permit Detail</i> , termasuk memperbaiki validator autentikasi serta fungsi perhitungan hari untuk meningkatkan konsistensi sistem.
5	Menambah API baru untuk persetujuan cuti, memperbaiki fungsi perhitungan hari, melakukan refaktor pada fungsi pengguna dan paginasi, serta memperbaiki data terkait proyek, produk, dan jabatan.
6	Melakukan pembaruan besar pada laporan pengguna, kontrak, dan API Stand Up, menambahkan detail pada kontrak serta item payroll, memperbaiki relasi model dan API payslip, serta meningkatkan sistem paginasi dan manajemen jenis cuti.
7	Melakukan finalisasi pada API Payslip, memperbaiki bug pada People Report, menambahkan fitur paginasi dan penyortiran, melakukan migrasi baru untuk Payslip Details, serta memperbaiki validator sistem.
8	Melakukan pengembangan fitur ekspor data ke format XLSX, perbaikan API pembuatan pengguna, perbaikan berkas migrasi dan seeding, serta penambahan sistem perizinan baru pada setiap modul.
9	Melakukan perbaikan pada berbagai modul seperti kontrak, akun, dan pengguna, menambahkan fungsi baru untuk pengelolaan kontrak aktif, serta merencanakan fitur input data klien melalui pembacaan berkas CSV untuk proses onboarding.
10	Melanjutkan pengembangan fitur onboarding klien dengan melanjutkan pengembangan sistem pembacaan berkas CSV, pembuatan tabel serta fungsi Import Mapping, dan perbaikan FileController untuk mendukung proses impor data secara efisien.
Lanjut pada halaman berikutnya	

1

1

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

Minggu Ke -	Pekerjaan yang dilakukan
11	Menyelesaikan pengembangan layanan Onboarding dengan menerapkan proses parsing data pada setiap modul, sehingga sistem impor data klien dapat berjalan secara otomatis dan terintegrasi dengan baik.
12	Menambahkan fitur penanganan kesalahan pada fungsi Onboarding untuk proses unggah dan impor berkas CSV, serta memungkinkan pengguna untuk mengimpor bagian modul secara bertahap guna meningkatkan fleksibilitas dan keandalan sistem.

1

### 3.4 Pengumpulan dan Analisis Kebutuhan

Kebutuhan sistem dalam proyek ini diperoleh melalui koordinasi langsung dengan *Software Engineer Manager*. Sebagian besar *requirement* ditentukan secara iteratif berdasarkan kebutuhan bisnis dan *sprint* mingguan yang telah direncanakan oleh tim. Proses pengumpulan *requirement* dilakukan melalui diskusi teknis mingguan di *retrospective meeting*. Setiap *requirement* yang diidentifikasi kemudian dianalisis untuk menentukan cakupan fungsionalitas, prioritas implementasi, serta dampaknya terhadap sistem yang ada. Hasil analisis ini menjadi dasar dalam perancangan dan pengembangan fitur-fitur baru yang sesuai dengan kebutuhan pengguna dan tujuan bisnis perusahaan.

Berikut ini adalah uraian *requirement* utama yang berhasil diidentifikasi selama masa kerja praktik:

#### A Modul Contract

Sebagai landasan utama bagi sistem penggajian (*Payroll*), perusahaan membutuhkan mekanisme pengelolaan data kontrak yang terpusat. Sistem harus mampu menangani seluruh siklus hidup kontrak kerja pegawai, mulai dari pembuatan draf hingga pemantauan masa berlaku. Selain itu, modul ini harus memfasilitas validasi status kontrak melalui fitur persetujuan (*approval*),

pengakhiran (*terminate*), serta penghapusan data untuk menjamin fleksibilitas dalam pengelolaan hubungan kerja.

### B Modul *Payroll Item*

Manajemen kompensasi yang dinamis membutuhkan pengelolaan komponen gaji yang terstruktur. Sistem harus mampu mendefinisikan berbagai kategori penghasilan seperti gaji pokok, tunjangan, hingga potongan secara spesifik. Modul ini dirancang untuk memfasilitasi penyesuaian komponen gaji berdasarkan kebijakan perusahaan yang berlaku, sehingga setiap perubahan parameter gaji dapat terpetakan dengan akurat ke dalam sistem.

### C Modul *Payslip*

Proses finalisasi penggajian membutuhkan akurasi tinggi dalam kalkulasi pembayaran. Sistem harus mampu melakukan kalkulasi otomatis berdasarkan integrasi data kontrak dan variabel lainnya untuk menghasilkan slip gaji yang akurat. Modul ini memfasilitasi penyediaan informasi rincian gaji, periode pembayaran, dan catatan historis secara transparan, yang dapat diakses secara mandiri oleh pegawai.

### D Modul *On Boarding*

Untuk menangani ekspansi data klien, operasional perusahaan membutuhkan metode input data yang cepat dan masal. Sistem harus mampu memproses impor data melalui berkas *Comma Separated Values* (CSV) sesuai dengan *template* yang telah ditentukan. Fitur ini memfasilitasi validasi integritas dan konsistensi data secara otomatis, sehingga mempercepat waktu implementasi bagi pengguna baru.

### E Modul *Overtime*

Pencatatan jam kerja tambahan membutuhkan sistem pemantauan yang transparan untuk menghindari kesalahan perhitungan manual. Sistem harus mampu mencatat durasi lembur secara tepat dan mengelola alur persetujuan oleh atasan. Modul ini memfasilitasi pengintegrasian kompensasi lembur ke dalam sistem

penggajian, memastikan setiap jam kerja tambahan pegawai tercatat dan dibayarkan sesuai regulasi perusahaan.

### 1 3.5 Perancangan dan Pengembangan Sistem

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

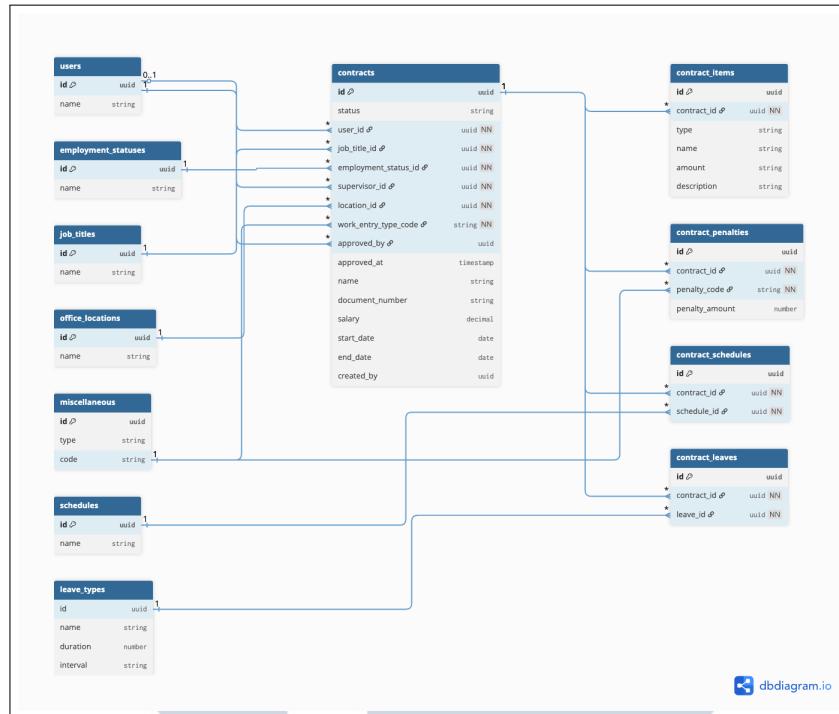
#### 3.5.1 Modul Contract

Modul *Contract* berfungsi sebagai fondasi utama dalam sistem penggajian (*Payroll*) yang dikembangkan. Modul ini mencakup pembuatan, pengelolaan, dan pemantauan kontrak kerja pegawai, termasuk fitur-fitur seperti persetujuan kontrak, pembatalan, serta penghapusan kontrak. Struktur basis data dirancang untuk mendukung berbagai jenis kontrak dan aturan terkait, sehingga memungkinkan fleksibilitas dalam pengelolaan hubungan kerja.

#### A Diagram ERD Modul Contract

Berikut merupakan *database diagram* untuk modul *Contract* yang telah dibuat selama pelaksanaan kerja praktik.



Gambar 3.2. Diagram ERD untuk modul *Contract*

Gambar 3.2 menunjukkan struktur basis data untuk modul *Contract*.

Terdapat beberapa tabel utama yang saling berhubungan, yaitu:

- **Contracts**: Tabel ini menyimpan data kontrak kerja pegawai, termasuk informasi seperti tanggal mulai dan berakhir kontrak, status persetujuan, serta referensi ke pegawai yang bersangkutan.
- **Employment Statuses**: Tabel ini menyimpan berbagai status kepegawaian yang dapat dimiliki oleh pegawai, seperti aktif, cuti, atau tidak aktif.
- **Job Titles**: Tabel ini menyimpan informasi mengenai jabatan atau posisi yang dimiliki oleh pegawai dalam perusahaan.
- **Office Location**: Tabel ini menyimpan data lokasi kantor tempat pegawai bekerja, yang dapat digunakan untuk keperluan absensi.
- **Miscellaneous**: Tabel ini menyimpan data tambahan terkait kontrak, seperti jenis kontrak, durasi kerja, dan informasi lainnya yang mendukung pengelolaan kontrak.
- **Schedules**: Tabel ini menyimpan jadwal kerja yang terkait dengan kontrak, termasuk jam kerja dan hari kerja yang ditetapkan.

1

- *Leave Types*: Tabel ini menyimpan jenis-jenis cuti yang tersedia bagi pegawai, yang dapat dikaitkan dengan kontrak kerja.
- *Contract Items*: Tabel ini menyimpan rincian *item-item* yang termasuk dalam kontrak, seperti gaji pokok, tunjangan, dan potongan.
- *Contract Penalties*: Tabel ini menyimpan data penalti yang dapat dikenakan dalam kontrak kerja, termasuk jenis pelanggaran dan besaran penalti.
- *Contract Schedules*: Tabel ini menyimpan hubungan antara kontrak dan jadwal kerja yang ditetapkan.
- *Contract Leaves*: Tabel ini menyimpan hubungan antara kontrak dan jenis cuti yang tersedia bagi pegawai.

1

## B *Contract API Endpoints*

Berikut adalah daftar *endpoints* API yang telah dikembangkan untuk modul *Contract*:

Nama API	Metode	Endpoint	Deskripsi
Get All User's Contracts	GET	/contract/user	Mengambil semua kontrak milik pengguna
Get User's Contract Detail by ID	GET	/contract/user/:id	Mengambil detail kontrak <i>user</i> berdasarkan ID
Approve Contract by ID	PATCH	/contract/:id/approve	Menyetujui kontrak <i>user</i> berdasarkan ID
Terminate Contract by ID	DELETE	/contract/:id	Mengakhiri kontrak <i>user</i> berdasarkan ID
Delete Contract by ID	DELETE	/contract/:id/delete	Menghapus kontrak <i>user</i> berdasarkan ID

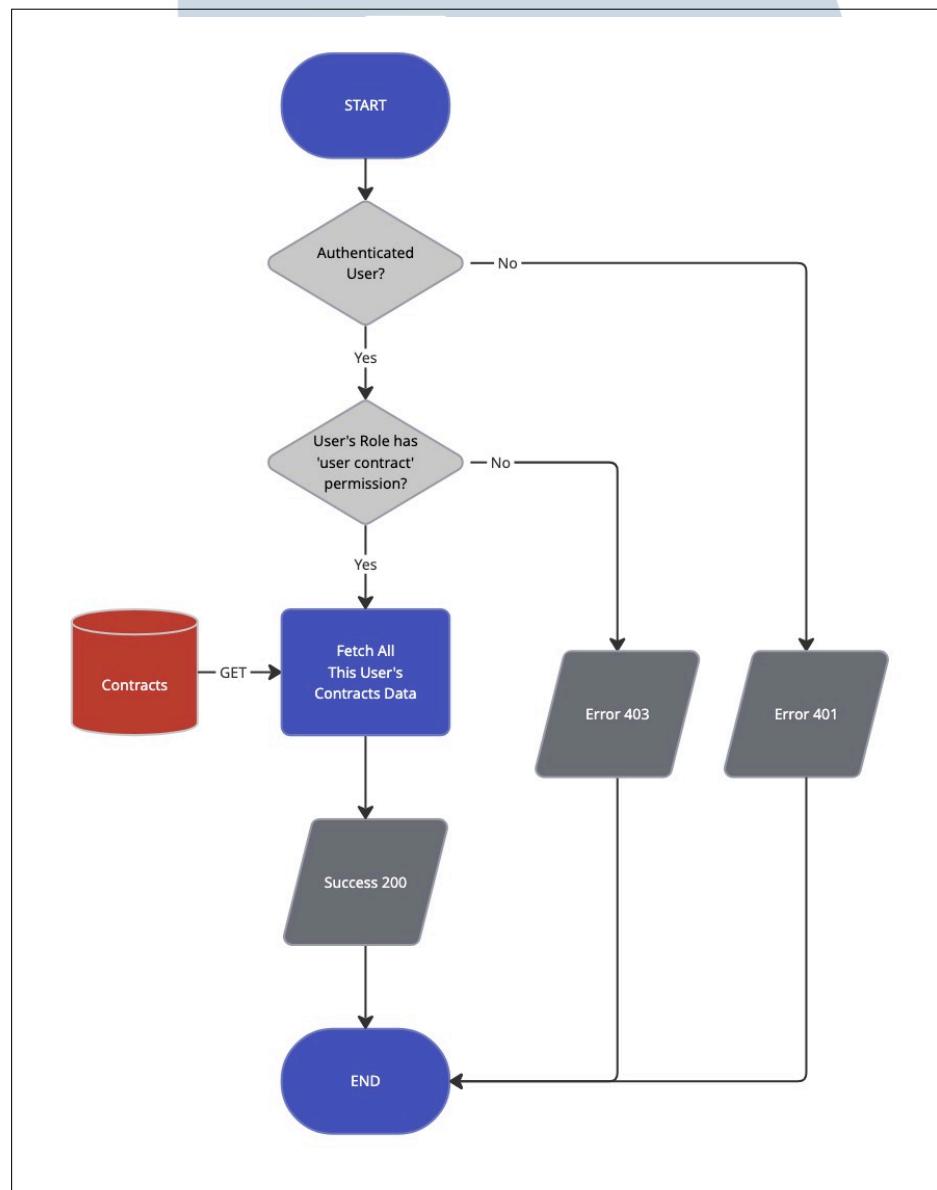
Tabel 3.2. Daftar *endpoints* API untuk modul *Contract*

Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Contract* seperti pada Tabel 3.2. Setiap *endpoint* memiliki fungsi spesifik

dalam mengelola kontrak kerja pegawai, mulai dari pengambilan data kontrak, persetujuan, hingga penghapusan kontrak. Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

### ***Get All User's Contracts***

Berikut merupakan *flowchart* API *Get All User's Contracts*.



Gambar 3.3. *Flowchart* alur sistem API *Get All User's Contracts*

Berikut merupakan contoh *request* dan *response* untuk API *Get All User's*

## Contracts.

The screenshot shows the 'Get All User's Contracts' API endpoint. It includes a 'Request' section with fields for Authorization, Query Params (pagination, page, row, sort, search, filter[]), and a note about providing a bearer token in the Authorization header. The 'Query Params' section lists required and optional parameters: pagination (required), page (required), row (required), sort (optional), search (optional), and filter[] (optional).

Gambar 3.4. Contoh request API Get All User's Contracts

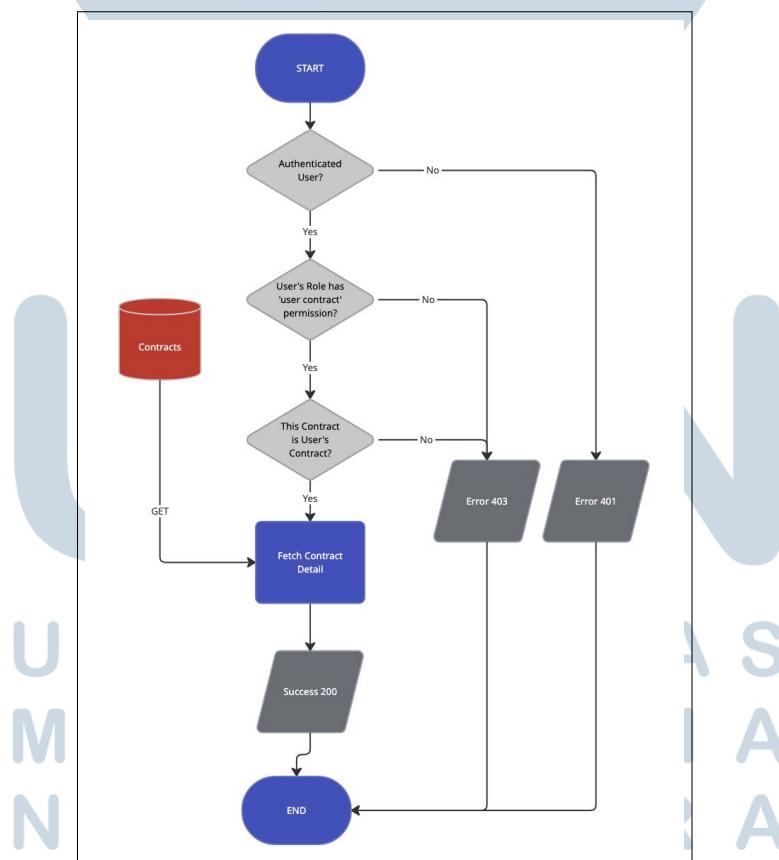
```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "count": 1,  
        "rows": [  
            {  
                "id": "45f7d2c6-4fe1-46b4-935b-71eb0d779bcf",  
                "status": "ON-GOING",  
                "name": "Staff 1 Contract",  
                "start_date": "2024-12-12T00:00:00.000Z",  
                "end_date": "2026-12-12T00:00:00.000Z",  
                "created_at": "2025-12-17T10:27:43.221Z",  
                "updated_at": "2025-12-17T10:27:43.221Z",  
                "is_lunch": true,  
                "employee_name": {  
                    "id": "a02f1296-8265-43f5-81e5-2dbad43a1180",  
                    "fullname": "Staff 1"  
                },  
                "job_title": {  
                    "id": "641b1453-2b50-4d6d-a3c9-65faf617e4db",  
                    "name": "Marketing"  
                },  
                "employment_status": {  
                    "id": "45794ced-4ff4-4ea3-9419-d9657cf9f1d",  
                    "name": "Full Time"  
                },  
                "created": {  
                    "id": "b3f5713e-2fca-4e63-8f3c-69dab2341633",  
                    "fullname": "Supervisor 1"  
                }  
            }  
        ]  
    }  
}
```

Gambar 3.5. Contoh response API Get All User's Contracts

Gambar 3.3 menunjukkan *flowchart* alur sistem API *Get All User's Contracts* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke endpoint `/contract/user` menggunakan metode HTTP *GET* seperti pada Gambar 3.4. Dalam *query parameters* terdapat beberapa variabel yang harus diisi dan dapat diisi untuk memenuhi kebutuhan *pagination* supaya data yang diterima dapat dibatasi. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil data kontrak dari pengguna tersebut. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar kontrak yang dimiliki oleh pengguna tersebut seperti pada Gambar 3.5.

### ***Get User's Contract Detail by ID***

Berikut merupakan *flowchart* API *Get User's Contract Detail by ID*.



Gambar 3.6. *Flowchart* alur sistem API *Get User's Contract Detail by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Get User's*

### Contract Detail by ID.

**Detail Contract**

**GET** /contract/{id} • Developing

Shared  
Created August 4, 2025 Updated 4 months ago Updated by JoseAndreasLie Creator Vn1k Maintainer Not configured Folder Contracts

**Request**

Authorization  
Provide your bearer token in the `Authorization` header when making requests to protected resources.  
Example: Authorization: Bearer \*\*\*\*\*

**Path Params**

`id` string required  
Example: 6d901b10-86f3-48d0-9a11-e02e24253293

Gambar 3.7. Contoh *request* API Get User's Contract Detail by ID

```
{
  "code": 200,
  "message": "Success",
  "data": {
    "status": "ON GOING",
    "employee_name": {
      "id": "a02f12196-8265-43f5-81e5-2dbad43a1100",
      "fullname": "Staff 1"
    },
    "name": "Staff 1 Contract",
    "job_title": {
      "id": "64131a53-2658-4d6d-a3c9-65faf17e4db",
      "name": "Marketing"
    },
    "employment_status": {
      "id": "45794ced-4ff4-4ea3-9419-d9657cfa9f1d",
      "name": "Full Time"
    },
    "supervisor": {
      "id": "b3f5713e-2fc4-4e63-8f3c-69dab2341633",
      "fullname": "Supervisor 1"
    },
    "location": {
      "id": "abdc8144-e9d2-4104-b5ba-379c47fa43f",
      "name": "Tangerang Office (GDP",
      "address": "Jl. Grand Boulevard, BSD Green Office Park 9, BSD City, Sampora, Kec. Cisauk, Kabupaten Tangerang, Banten 15345"
    },
    "start_date": "2024-12-12T00:00:00.000Z",
    "end_date": "2026-12-12T00:00:00.000Z",
    "type": {
      "id": "8cc1bed6-8ef7-4a52-b004-90fc3fff57f45",
      "name": "Annual"
    },
    "work_entry_type": {
      "id": "1",
      "type": "schedule",
      "code": "EXED",
      "value": "Fixed",
      "description": ""
    },
    "schedule": [
      {
        "id": "408c4205-d8c4-4d92-9e25-f256a80f6033",
        "name": "Senin - Jumat (09:00 - 18:00)",
        "shift_name": null,
        "code": null,
        "sort_order": null
      },
      {
        "id": "b9b2c7ad-1d6f-47c0-a1ef-9e33e8d578f",
        "name": "Lunch Time (12:00 - 13:00)",
        "shift_name": null
      }
    ]
  }
}
```

Gambar 3.8. Contoh *response* API Get User's Contract Detail by ID

```
{
  "id": "b9b2c7ad-1d6f-47c0-aef-9e33e8d8578",
  "name": "Lunch Time (12:00 - 13:00)",
  "shift_name": null,
  "code": null,
  "sort_order": null
},
{
  "penalty_rule": [
    {
      "type": "penalty",
      "value": "Missing In Action",
      "description": "Absent without notice or approval",
      "amount": 137000
    },
    {
      "type": "penalty",
      "value": "Pulang lebih awal -2 Jam Pulang",
      "description": "Karyawan yang pulang lebih awal dari 2 jam dari jam pulang",
      "amount": 27000
    },
    {
      "type": "penalty",
      "value": "Terlambat +2 Jam Masuk",
      "description": "Karyawan yang datang terlambat kurang dari 2 jam dari jam masuk",
      "amount": 27000
    },
    {
      "type": "penalty",
      "value": "Terlambat lebih dari +2 Jam Masuk",
      "description": "Karyawan yang datang terlambat lebih dari 2 jam dari jam masuk",
      "amount": 54000
    },
    {
      "type": "penalty",
      "value": "Late Arrival (After Lunch Break)",
      "description": "Penalty for arriving after the lunch period.",
      "amount": 110000
    },
    {
      "type": "penalty",
      "value": "Kombinasi Telat +2 Jam dan Pulang Lebih Awal",
      "description": "Karyawan yang pulang awal dan telat",
      "amount": 110000
    }
  ],
  "contract_item": [
    {
      "id": "0757ea6f-eb74-4a35-8bf5-7dbd7050f06b",
      "type": "ADDITION",
      "name": "Main Salary",
      "amount": "6400000"
    }
  ]
}
```

Gambar 3.9. Contoh response API Get User's Contract Detail by ID (lanjutan)

```
[
  "contract_item": [
    {
      "id": "0757ea6f-eb74-4a35-8bf5-7dbd7050f06b",
      "type": "ADDITION",
      "name": "Main Salary",
      "amount": "6400000"
    },
    {
      "id": "970185e2-895c-45e7-9341-bcdcb27ad7fb",
      "type": "ADDITION",
      "name": "Tunjangan Makan",
      "amount": "600000"
    },
    {
      "id": "e49fd80c-4fe2-4650-82fe-f5a2d923c62c",
      "type": "BENEFIT",
      "name": "Tunjangan BPJS",
      "amount": "35000"
    }
  ],
  "approved_by": [
    {
      "id": "2e2a4c32-49d1-4b86-afc2-dd4ab8241aa8",
      "fullname": "Superadmin"
    }
  ],
  "approved_at": "2025-12-17T10:27:43.221Z"
}
```

Gambar 3.10. Contoh response API Get User's Contract Detail by ID (lanjutan)

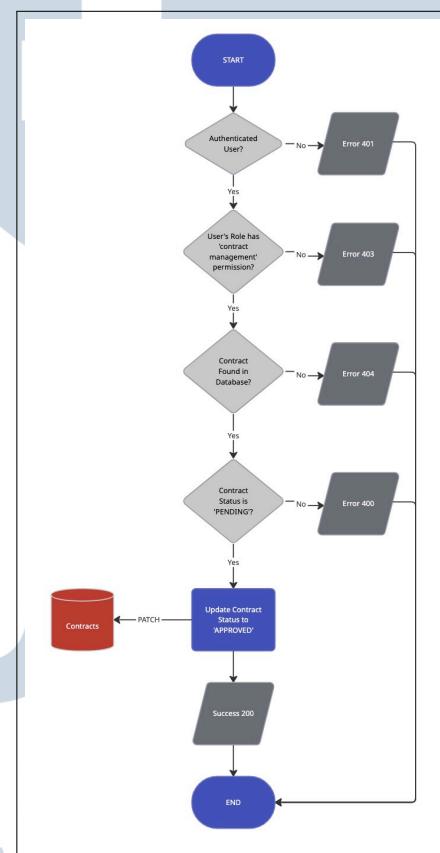
Gambar 3.6 menunjukkan flowchart alur sistem API Get User's Contract Detail by ID yang dimulai dari pengguna mengirimkan permintaan (*request*) ke

4  
1

*endpoint /contract/user/:id* menggunakan metode HTTP *GET* seperti pada Gambar 3.7. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta mengambil data detail kontrak berdasarkan *ID* yang diberikan dari *path parameters*, *contract\_id*. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail kontrak yang diminta seperti pada Gambar 3.8, 3.9, dan 3.10.

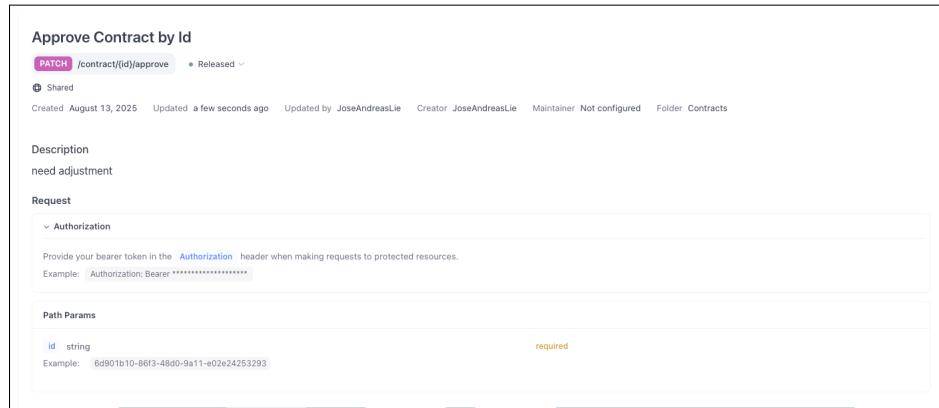
### **Approve Contract by ID**

Berikut merupakan *flowchart* API *Approve Contract by ID*.



Gambar 3.11. Contoh *response* API *Approve Contract by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Approve Contract by ID*.



Gambar 3.12. Contoh *request* API *Approve Contract by ID*

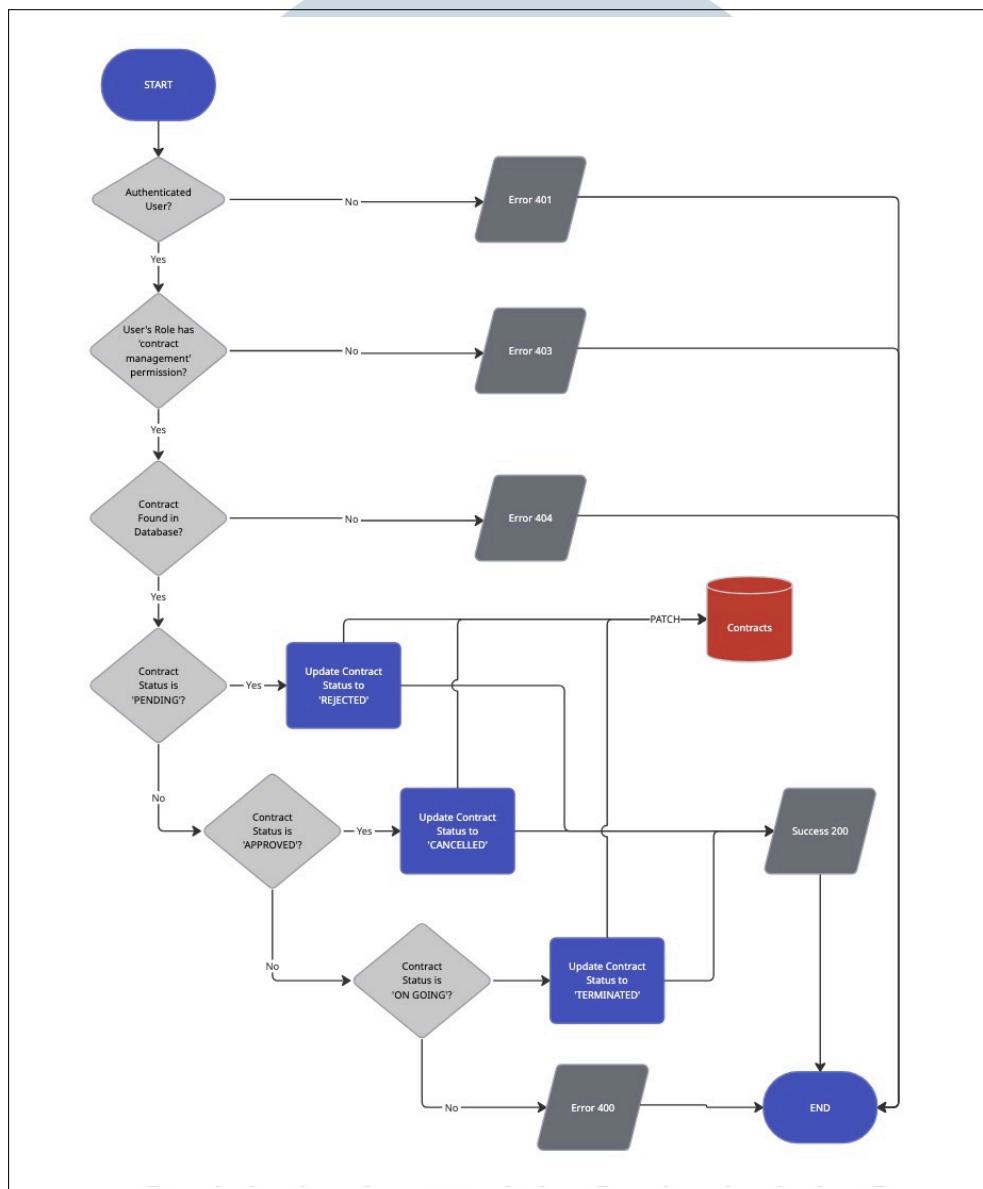


Gambar 3.13. Contoh *response* API *Approve Contract by ID*

Gambar ?? menunjukkan *flowchart* alur sistem API *Approve Contract by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* `/contract/:id/approve` menggunakan metode HTTP `PATCH` seperti pada Gambar 3.12. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta memperbarui status kontrak menjadi disetujui (*Approved*) berdasarkan *ID* yang diberikan dari *path parameters*, `contract_id`. Setelah status kontrak berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi kontrak yang telah disetujui seperti pada Gambar 3.13.

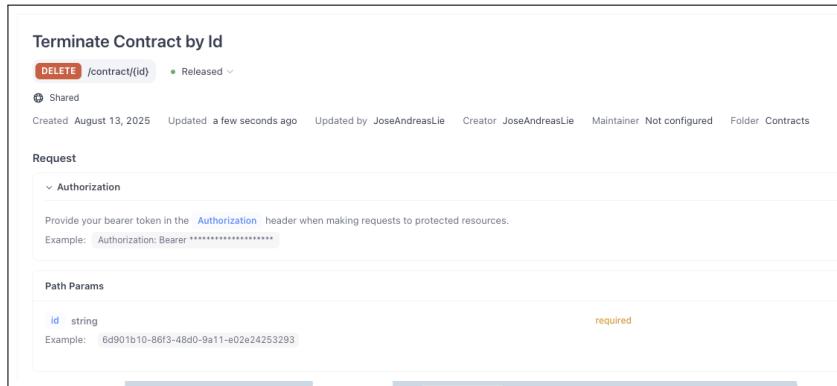
### Terminate Contract by ID

Berikut merupakan *flowchart* API *Terminate Contract by ID*.



Gambar 3.14. Flowchart alur sistem API *Terminate Contract by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Terminate Contract by ID*.



Gambar 3.15. Contoh *request* API *Terminate Contract by ID*



Gambar 3.16. Contoh *response* API *Terminate Contract by ID*

Gambar 3.14 menunjukkan *flowchart* alur sistem API *Terminate Contract by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /contract/:id menggunakan metode HTTP *DELETE* seperti pada Gambar 3.15. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, lalu mencari data kontrak berdasarkan *ID* yang diberikan dari *path parameters*, contract\_id. Setelah ditemukan, status akan berubah sesuai dengan status kontrak sekarang, dengan ketentuan sebagai berikut:

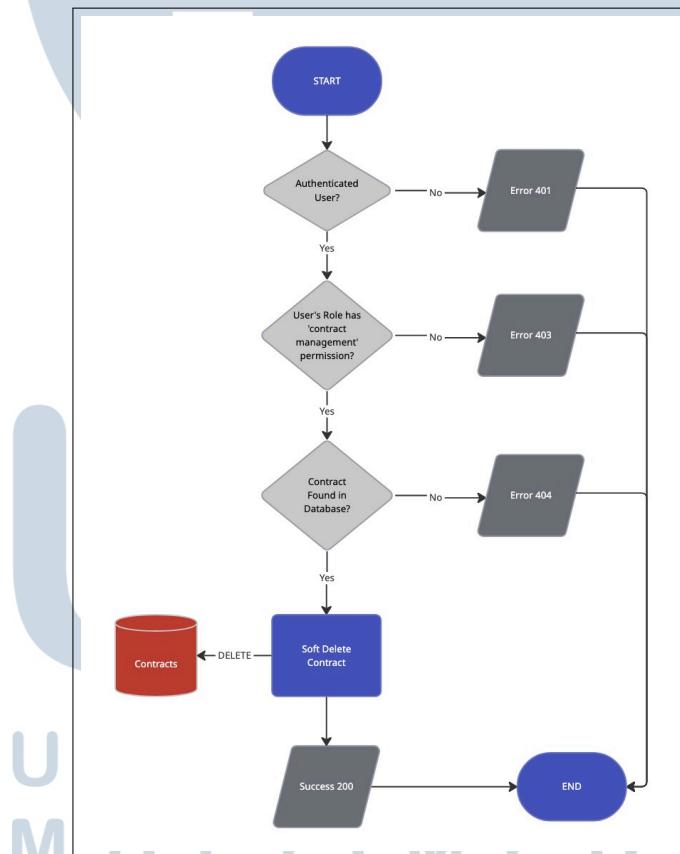
- Jika status kontrak adalah *Pending*, maka status akan diubah menjadi *Rejected*.
- Jika status kontrak adalah *Approved*, maka status akan diubah menjadi *Cancelled*.

- Jika status kontrak adalah *On Going*, maka status akan diubah menjadi *Terminated*.

5 Jika status kontrak tidak sesuai dengan ketentuan di atas, maka sistem akan mengembalikan respons kesalahan (*error response*) kepada pengguna. Setelah status kontrak berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi kontrak yang telah dihentikan seperti pada Gambar 3.16.

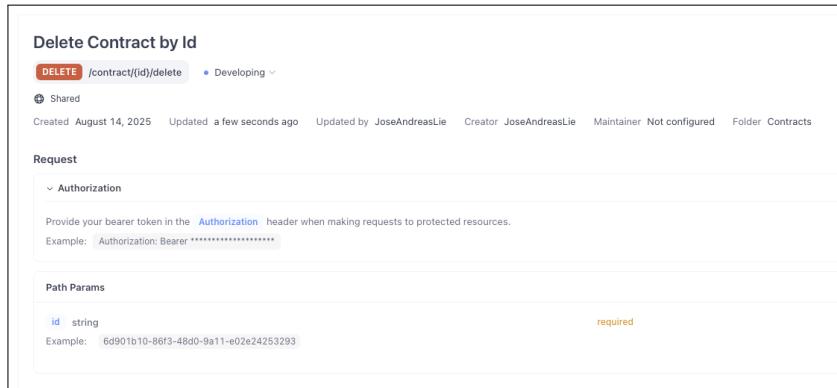
### **Delete Contract by ID**

Berikut merupakan *flowchart* API *Delete Contract by ID*.



Gambar 3.17. *Flowchart* alur sistem API *Delete Contract by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Delete Contract by ID*.



Gambar 3.18. Contoh *request* API *Delete Contract by ID*



Gambar 3.19. Contoh *response* API *Delete Contract by ID*

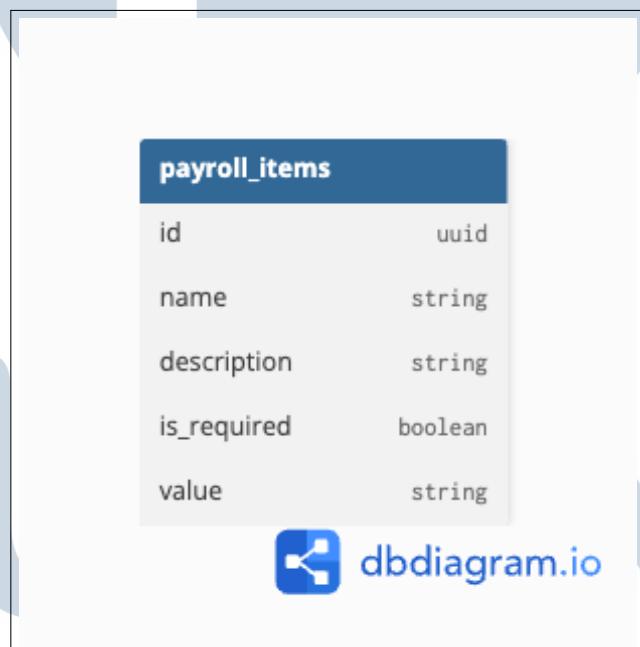
Gambar 3.17 menunjukkan *flowchart* alur sistem API *Delete Contract by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /contract/:id/delete menggunakan metode HTTP *DELETE* seperti pada Gambar 3.18. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta menghapus data kontrak berdasarkan *ID* yang diberikan dari *path parameters*, contract\_id. Setelah data kontrak berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi keberhasilan penghapusan kontrak seperti pada Gambar 3.19.

### 3.5.2 Modul Payroll Item

Modul Payroll Item dibuat untuk mengelola berbagai komponen gaji yang akan digunakan dalam perhitungan gaji pegawai. Modul ini mencakup fitur pembuatan, pengelolaan, dan penghapusan item gaji seperti gaji pokok, tunjangan, dan potongan. Struktur basis data dirancang untuk menyimpan informasi terkait item gaji, termasuk jenis item, besaran, serta aturan penerapannya. Dengan adanya modul ini, perusahaan

#### A Diagram ERD Modul *Payroll Item*

Berikut merupakan *database diagram* untuk modul *Payroll Item* yang telah dibuat selama pelaksanaan kerja praktik.



Gambar 3.20. Diagram ERD untuk modul *Payroll Item*

Gambar 3.20 menunjukkan struktur basis data untuk modul *Payroll Item*.

#### B Payroll Item API Endpoints

Berikut adalah daftar *endpoints* API yang telah dikembangkan untuk modul *Payroll Item*:

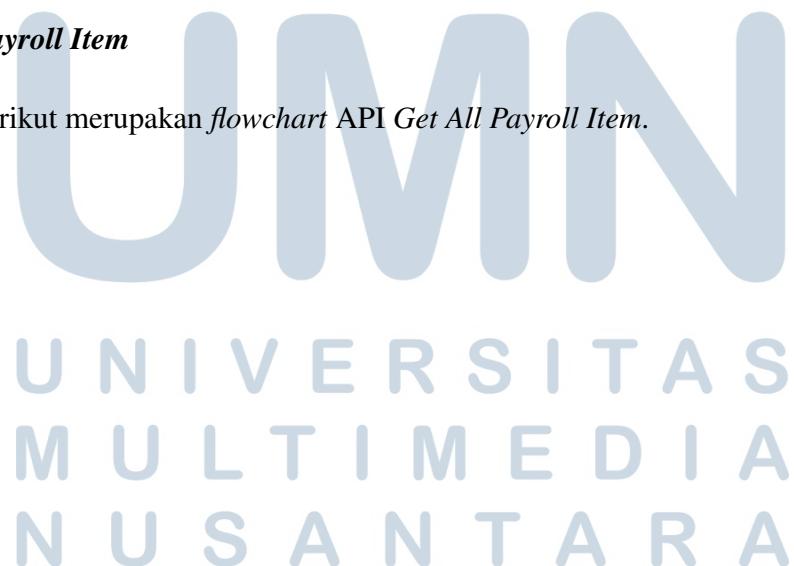
Nama API	Metode	Endpoint	Deskripsi
Get All Payroll Item	GET	/payroll-items	Mengambil semua komponen gaji ( <i>Payroll Item</i> )
Get Payroll Item by ID	GET	/payroll-items/:id	Mengambil detail komponen gaji berdasarkan ID
Create Payroll Item	POST	/payroll-items	Membuat komponen gaji baru
Update Payroll Item	PUT	/payroll-items/:id	Memperbarui komponen gaji berdasarkan ID
Delete Payroll Item	DELETE	/payroll-items/:id	Menghapus komponen gaji berdasarkan ID

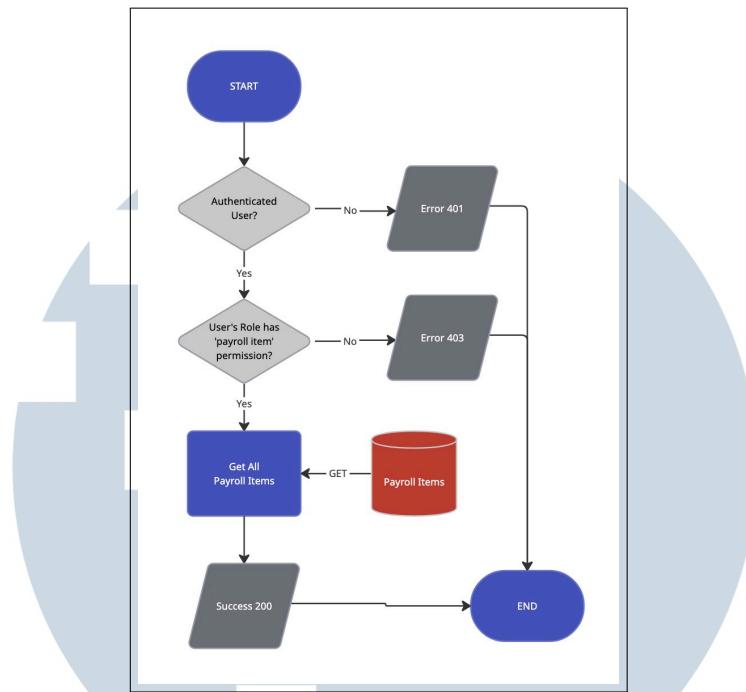
Tabel 3.3. Daftar *endpoints* API untuk modul *Payroll Item*

Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Payroll Item* seperti pada Tabel 3.3. Setiap *endpoint* memiliki fungsi spesifik untuk mengelola komponen gaji (*Payroll Item*). Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

### ***Get All Payroll Item***

Berikut merupakan *flowchart* API *Get All Payroll Item*.





Gambar 3.21. Flowchart alur sistem API Get All Payroll Item

Berikut merupakan contoh *request* dan *response* untuk API Get All Payroll Item.

The screenshot shows the 'Get All Payroll Items' API endpoint. It includes the method (GET), URL (/payroll-items), and status (Released). It also shows the creation date (August 6, 2025) and last update (4 hours ago). The 'Request' section details the required Authorization header (with an example: Authorization: Bearer \*\*\*\*\*) and the Query Params:

Query Params	Description	Required
pagination	boolean	required
page	string	required
row	string	required
sort	string	optional
search	string	optional

Gambar 3.22. Contoh *request* API Get All Payroll Item

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "count": 4,  
        "rows": [  
            {  
                "id": "64ba20fd-596a-40f4-82cc-01afab381b71",  
                "name": "Tunjangan Makan",  
                "is_required": false,  
                "value": null,  
                "description": "Tunjangan yang diberikan untuk makan karyawan.",  
                "created_at": "2025-12-17T04:26:52.630Z",  
                "updated_at": "2025-12-17T04:26:52.630Z"  
            },  
            {  
                "id": "aa5b9675-b608-487a-851c-4e9b55b170f7",  
                "name": "Tunjangan BPJS",  
                "is_required": false,  
                "value": null,  
                "description": "Tunjangan yang diberikan untuk BPJS karyawan.",  
                "created_at": "2025-12-17T04:26:52.630Z",  
                "updated_at": "2025-12-17T04:26:52.630Z"  
            },  
            {  
                "id": "7cf74dbd-b758-4304-8c6c-1b46c52977e7",  
                "name": "Tunjangan Kehormatan",  
                "is_required": false,  
                "value": null,  
                "description": "Tunjangan yang diberikan untuk jabatan karyawan.",  
                "created_at": "2025-12-17T04:26:52.630Z",  
                "updated_at": "2025-12-17T04:26:52.630Z"  
            },  
            {  
                "id": "6821d814-508c-436f-a232-423bae9997ed",  
                "name": "Main Salary",  
                "is_required": true,  
                "value": "main salary",  
                "description": "Gaji Pokok",  
                "created_at": "2025-12-17T04:07:19.539Z",  
                "updated_at": "2025-12-17T04:07:19.539Z"  
            }  
        ]  
    }  
}
```

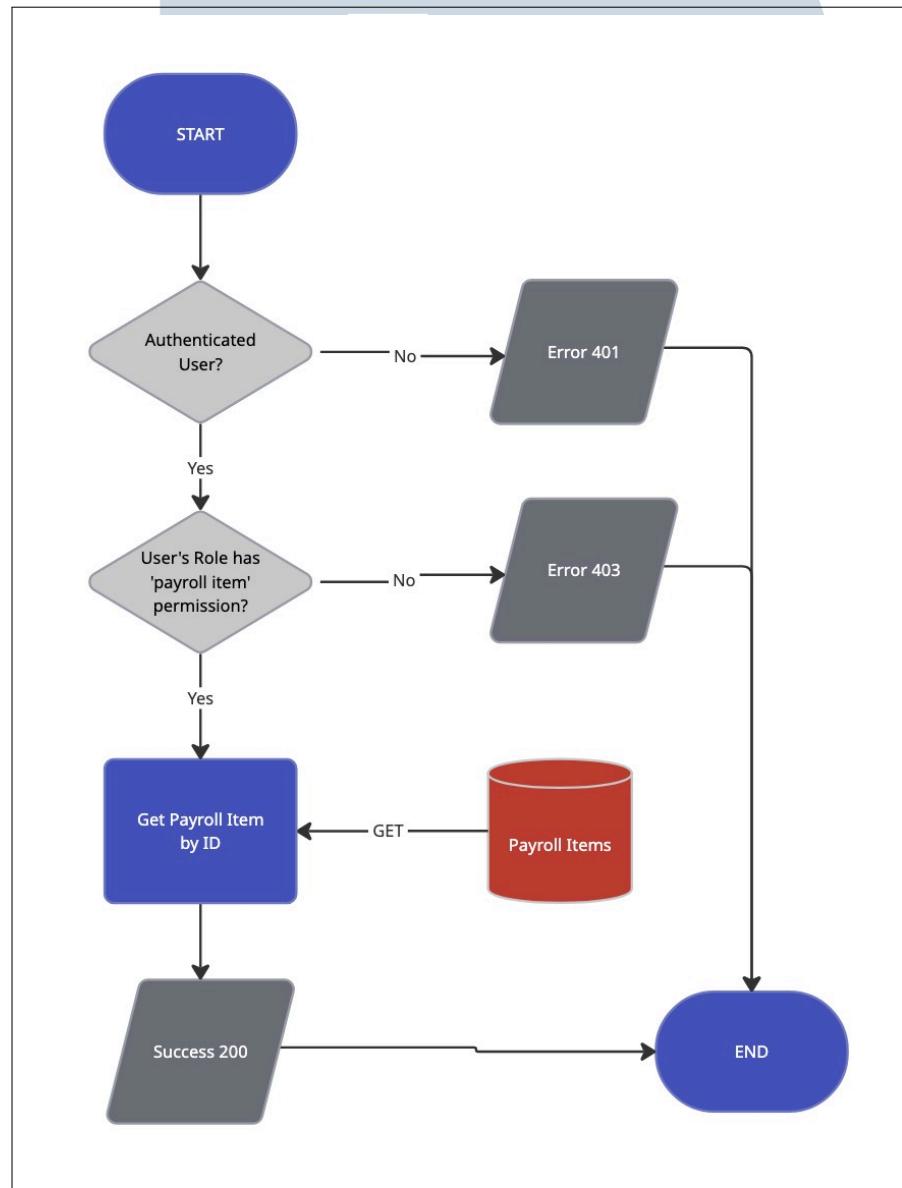
Gambar 3.23. Contoh response API Get All Payroll Item

Gambar 3.21 menunjukkan *flowchart* alur sistem API *Get All Payroll Item* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payroll-items menggunakan metode HTTP *GET* seperti pada Gambar 3.22. Dalam *query parameters* terdapat beberapa variabel yang harus diisi dan dapat diisi untuk memenuhi kebutuhan *pagination* supaya data yang diterima dapat dibatasi. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil semua data komponen

gaji (*Payroll Item*). Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar komponen gaji seperti pada Gambar 3.23.

### **Get Payroll Item by ID**

Berikut merupakan *flowchart* API *Get Payroll Item by ID*.



Gambar 3.24. *Flowchart* alur sistem API *Get Payroll Item by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Get Payroll Item*

by ID.

The screenshot shows the API endpoint `GET /payroll-items/{id}`. It includes details like 'Released' status, 'Shared' access, and creation date August 7, 2025. The 'Description' section states it's an API to get status data for user attendance and standup. The 'Request' section covers 'Authorization' (bearer token), 'Path Params' (id, required), and examples. The 'Response' section is not visible in the screenshot.

Gambar 3.25. Contoh *request* API Get Payroll Item by ID

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "id": "7cf74dbd-b758-4304-8c6c-1b46c52977e7",  
        "name": "Tunjangan Kehormatan",  
        "is_required": false,  
        "value": null,  
        "description": "Tunjangan yang diberikan untuk jabatan karyawan.",  
        "created_at": "2025-12-17T04:26:52.630Z",  
        "updated_at": "2025-12-17T04:26:52.630Z",  
        "deleted_at": null  
    }  
}
```

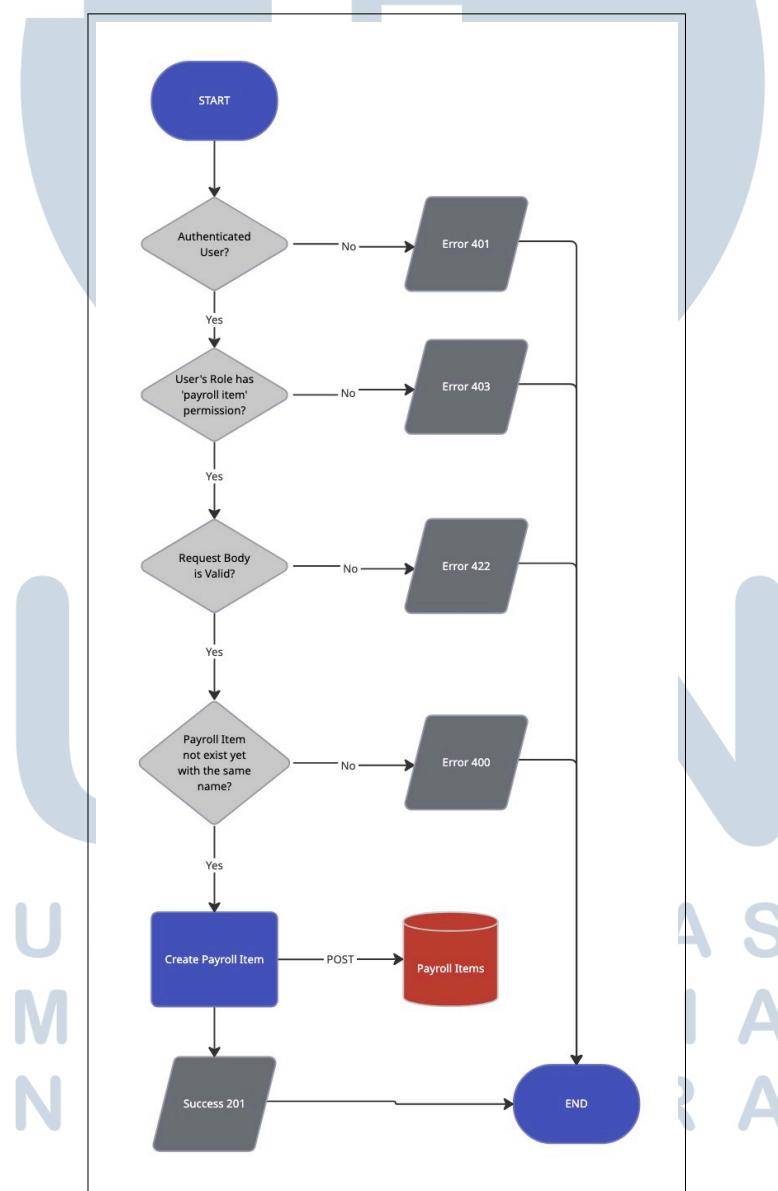
Gambar 3.26. Contoh *response* API Get Payroll Item by ID

Gambar 3.24 menunjukkan *flowchart* alur sistem API Get Payroll Item by ID yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* `/payroll-items/:id` menggunakan metode HTTP *GET* seperti pada Gambar 3.25. Sistem kemudian memproses permintaan tersebut dengan memeriksa

autentikasi pengguna, memeriksa *permission* pengguna, serta mengambil data komponen gaji berdasarkan *ID* yang diberikan dari *path parameters*. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail komponen gaji yang diminta seperti pada Gambar 3.26.

### Create Payroll Item

Berikut merupakan *flowchart* API *Create Payroll Item*.



Gambar 3.27. Flowchart alur sistem API *Create Payroll Item*

Berikut merupakan contoh *request* dan *response* untuk API *Create Payroll Item*.

The screenshot shows the "Create Payroll Items" API endpoint documentation. It includes a "Request" section with an "Example" schema:

```
[  
  {  
    "name": "PPh 21",  
    "description": "Pajak yang dikenakan kepada karyawan untuk negara."  
  },  
  {  
    "name": "Tunjangan Kendaraan",  
    "description": "Uang transportasi yang dibiayakan oleh perusahaan kepada karyawan."  
  }]
```

Gambar 3.28. Contoh *request* API *Create Payroll Item*

The screenshot shows the "Response" section of the API documentation, displaying the JSON structure of the response:

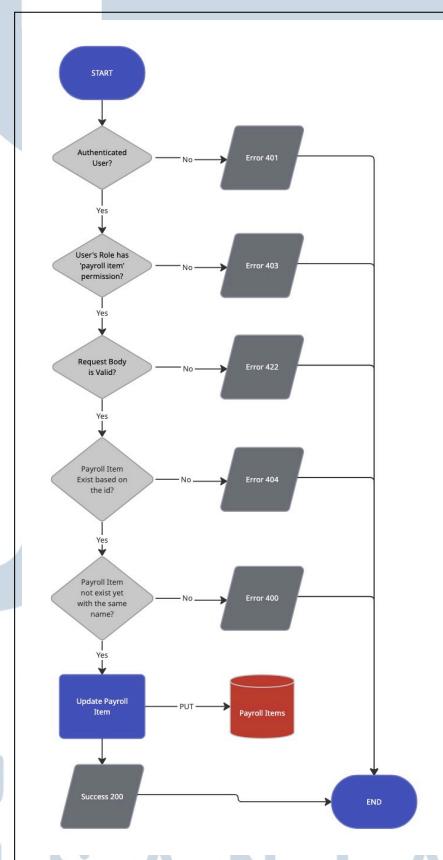
```
{  
  "code": 201,  
  "message": "Success",  
  "data": [  
    {  
      "id": "5a347071-1985-47a2-bce8-adf509b732d6",  
      "is_required": false,  
      "name": "PPh 21",  
      "description": "Pajak yang dikenakan kepada karyawan untuk negara.",  
      "value": "pph 21",  
      "created_at": "2025-12-18T08:07:11.178Z",  
      "updated_at": "2025-12-18T08:07:11.178Z",  
      "deleted_at": null  
    },  
    {  
      "id": "58af5134-a28f-440e-acc2-275cc8cdbeec",  
      "is_required": false,  
      "name": "Tunjangan Kendaraan",  
      "description": "Uang transportasi yang dibiayakan oleh perusahaan kepada karyawan.",  
      "value": "tunjangan kendaraan",  
      "created_at": "2025-12-18T08:07:11.178Z",  
      "updated_at": "2025-12-18T08:07:11.178Z",  
      "deleted_at": null  
    }  
  ]  
}
```

Gambar 3.29. Contoh *response* API *Create Payroll Item*

Gambar 3.27 menunjukkan *flowchart* alur sistem API *Create Payroll Item* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payroll-items menggunakan metode HTTP *POST* seperti pada Gambar 3.28. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, memeriksa *request body* tersebut valid sesuai dengan ketentuan, serta membuat data komponen gaji baru berdasarkan data yang diberikan dalam *request body*. Setelah data komponen gaji berhasil dibuat, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai komponen gaji yang telah dibuat seperti pada Gambar 3.29.

### ***Update Payroll Item***

Berikut merupakan *flowchart* API *Update Payroll Item*.



Gambar 3.30. *Flowchart* alur sistem API *Update Payroll Item*

Berikut merupakan contoh *request* dan *response* untuk API *Update Payroll Item*.

Update Payroll Item

PUT /payroll-items/{id} • Released

Shared  
Created August 7, 2025 Updated a few seconds ago Updated by JoseAndreasLie Creator JoseAndreasLie Maintainer Not configured Folder Payroll Items

Description  
API to get status data for user attendance and standup

Request

Authorization  
Provide your bearer token in the `Authorization` header when making requests to protected resources.  
Example: Authorization: Bearer \*\*\*\*

Path Params

`id` string required  
Example: 5a347071-1985-47a2-bce8-adf509b732d6

Example  
Datatype  
{  
    "name": "PPh 21 (karyawan)",  
    "description": "Pajak yang dikenakan kepada karyawan untuk negara (PPh 21)."  
}

Gambar 3.31. Contoh *request* API *Update Payroll Item*

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "id": "5a347071-1985-47a2-bce8-adf509b732d6",  
        "name": "PPh 21 (karyawan)",  
        "is_required": false,  
        "value": "pph 21 (karyawan)",  
        "description": "Pajak yang dikenakan kepada karyawan untuk negara (PPh 21).",  
        "created_at": "2025-12-18T08:07:11.178Z",  
        "updated_at": "2025-12-18T08:14:46.050Z",  
        "deleted_at": null  
    }  
}
```

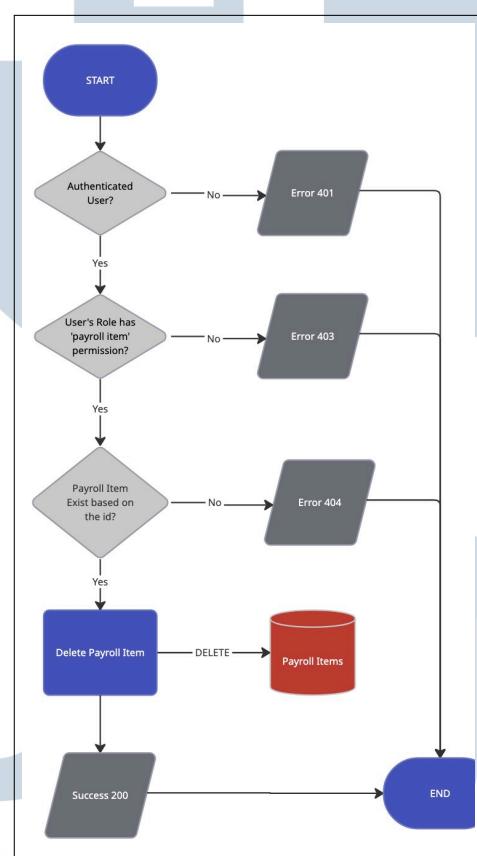
Gambar 3.32. Contoh *response* API *Update Payroll Item*

Gambar 3.30 menunjukkan *flowchart* alur sistem API *Update Payroll Item* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payroll-items/:id menggunakan metode HTTP *PUT* seperti pada Gambar 3.31. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, memeriksa *request body* tersebut valid sesuai dengan ketentuan, mencari data *Payroll Item* dengan

menggunakan *ID* yang diberikan dari *Path Parameter* serta memperbarui data komponen gaji data tersebut dengan data yang diberikan dalam *request body*. Setelah data komponen gaji berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai komponen gaji yang telah diperbarui seperti pada Gambar 3.32.

### Delete Payroll Item

Berikut merupakan *flowchart* API *Delete Payroll Item*.



Gambar 3.33. Flowchart alur sistem API Delete Payroll Item

Berikut merupakan contoh *request* dan *response* untuk API *Delete Payroll Item*.

The screenshot shows the Turnitin API documentation for the `Delete Payroll Item` endpoint. At the top, there is a red `DELETE` button followed by the URL `/payroll-items/{id}`. To the right of the URL, there is a status indicator showing a green dot and the word `Released`. Below the URL, it says `Shared`. Under the URL, there are fields for `Created`, `Updated`, `Updated by`, and `Creator`, all set to `August 7, 2025`, `12 hours ago`, `JoseAndreasLie`, and `JoseAndreasLie` respectively. There are also fields for `Maintainer` (set to `Not configured`) and `Folder` (`Payroll Items`). The main content area is divided into sections: `Description` (API to get status data for user attendance and standup), `Request`, and `Path Params`. The `Request` section contains a `Authorization` field with a note to provide a bearer token in the `Authorization` header. An example value is shown as `Authorization: Bearer *****`. The `Path Params` section shows a `id` parameter as a required string type, with an example value of `7cf74dbd-b758-4304-8c6c-1b46c52977e7`.

Gambar 3.34. Contoh request API Delete Payroll Item

The screenshot shows the Turnitin API documentation for the `Delete Payroll Item` endpoint, specifically the response section. It displays a JSON object with the following structure:

```
{  
  "code": 200,  
  "message": "Success"  
}
```

Gambar 3.35. Contoh response API Delete Payroll Item

Gambar 3.33 menunjukkan *flowchart* alur sistem API `Delete Payroll Item` yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* `/payroll-items/:id` menggunakan metode HTTP `DELETE` seperti pada Gambar 3.34. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta menghapus data komponen gaji berdasarkan *ID* yang diberikan dari *path parameters*. Setelah data

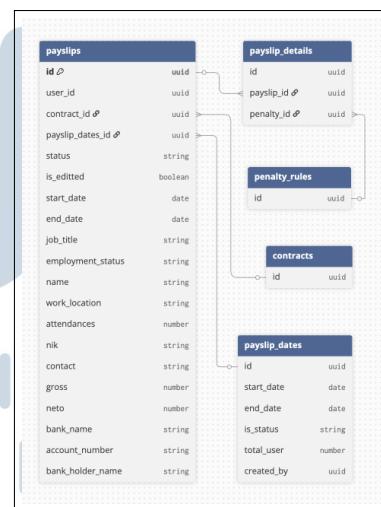
komponen gaji berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai keberhasilan penghapusan komponen gaji seperti pada Gambar 3.35.

### 3.5.3 Modul Payslip

Modul Payslip dibuat untuk mengelola dan menghasilkan slip gaji bagi pegawai berdasarkan kontrak yang telah disepakati, dan beberapa. Modul ini mencakup fitur perhitungan gaji, potongan, dan tunjangan sesuai dengan ketentuan yang berlaku dalam kontrak kerja. Struktur basis data dirancang untuk menyimpan informasi terkait gaji, termasuk rincian pembayaran, periode gaji, serta catatan historis. Modul ini juga menyediakan akses untuk menampilkan slip gaji kepada pegawai secara transparan. Modul Payslip juga merupakan langkah terakhir dari sistem penggajian (*Payroll*) yang terintegrasi dengan modul Contract dan modul lainnya.

#### A Diagram ERD Modul Payslip

Berikut merupakan *database diagram* untuk modul *Payslip* yang telah dibuat selama pelaksanaan kerja praktik.



Gambar 3.36. Diagram ERD untuk modul *Payslip*.

Gambar 3.36 menunjukkan struktur basis data untuk modul *Payslip*.

## B Payslip API Endpoints

Berikut adalah daftar *endpoints* API yang telah dikembangkan untuk modul *Payslip*:

Nama API	Metode	Endpoint	Deskripsi
Get Payslip Detail	GET	/payslip/date/payslip_date_id /contract/contract_id	Mengambil detail slip gaji <i>user</i>
Save Payslip Detail	PATCH	/payslip/date/payslip_date_id /contract/contract_id	Mengambil detail slip gaji <i>user</i> berdasarkan ID
Get User Payslips List	GET	/payslip/user	Mengambil semua slip gaji milik pengguna
Get User Payslip Detail by ID	GET	/payslip/user/:id	Mengambil detail slip gaji <i>user</i> berdasarkan ID

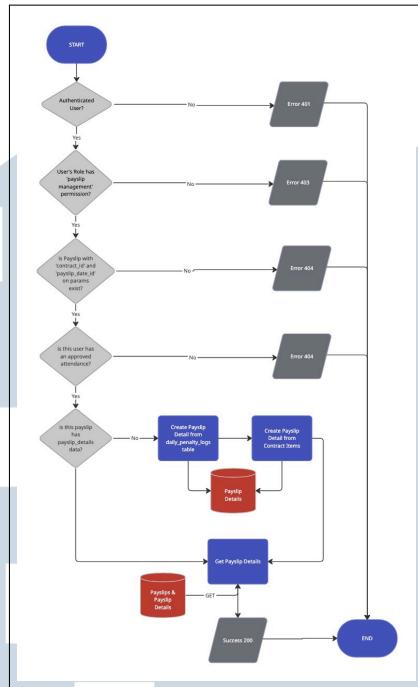
Tabel 3.4. Daftar *endpoints* API untuk modul *Payslip*

Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Payslip* seperti pada Tabel 3.4. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola slip gaji pegawai, mulai dari pengambilan data slip gaji, hingga penyimpanan detail slip gaji. Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

### Get Payslip Detail

Berikut merupakan *flowchart* API Get Payslip Detail.





Gambar 3.37. Flowchart alur sistem API Get Payslip Detail

Berikut merupakan contoh *request* dan *response* untuk API *Get Payslip Detail*.

**Get Payslip Detail**

**Request**

**Path Params**

**Response**

```

{
  "id": "5d3f3a2e-1a2c-4a2a-8a2a-1a2c4a2a4a2a",
  "date": "2020-09-08T10:00:00Z",
  "contract_id": "5d3f3a2e-1a2c-4a2a-8a2a-1a2c4a2a4a2a",
  "payslip_date_id": "5d3f3a2e-1a2c-4a2a-8a2a-1a2c4a2a4a2a"
}
  
```

Gambar 3.38. Contoh *request* API Get Payslip Detail

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "start_date": "2025-11-30T17:00:00.000Z",  
        "end_date": "2025-12-02T16:59:59.999Z",  
        "status": "DRAFTED",  
        "user_data": {  
            "name": "HoHR 1",  
            "nik": "112233445566778899",  
            "contact": "08123456789",  
            "job_title": "Director",  
            "employment_status": "Full Time",  
            "attendance": 2,  
            "workdays": 2  
        },  
        "income": [  
            {  
                "id": "643e35cb-2dbc-4706-8928-c6a31dde119",  
                "name": "Tunjangan BPJS",  
                "amount": "35000",  
                "description": "Tunjangan yang diberikan untuk BPJS karyawan.",  
                "source": "SYSTEM"  
            },  
            {  
                "id": "71ced7fd-e977-4651-aedd-096184447e2f",  
                "name": "Tunjangan Kehormatan",  
                "amount": "5000000",  
                "description": "Tunjangan yang diberikan untuk jabatan karyawan.",  
                "source": "SYSTEM"  
            },  
            {  
                "id": "3a3b02a0-cb41-4884-812f-4f24df6e3129",  
                "name": "Main Salary",  
                "amount": "10000000",  
                "description": "Gaji Pokok",  
                "source": "SYSTEM"  
            }  
        ],  
        "deduction": [  
            {  
                "id": "643e35cb-2dbc-4706-8928-c6a31dde119",  
                "name": "Tunjangan BPJS",  
                "amount": "35000",  
                "description": "Tunjangan yang diberikan untuk BPJS karyawan.",  
                "source": "SYSTEM"  
            }  
        ],  
        "take_home_pay": 15000000,  
        "bank_info": {  
            "bank_name": "Bank Central Asia",  
            "bank_account": "11223344556677",  
            "bank_account_name": "HoHR 1"  
        }  
    }  
}
```

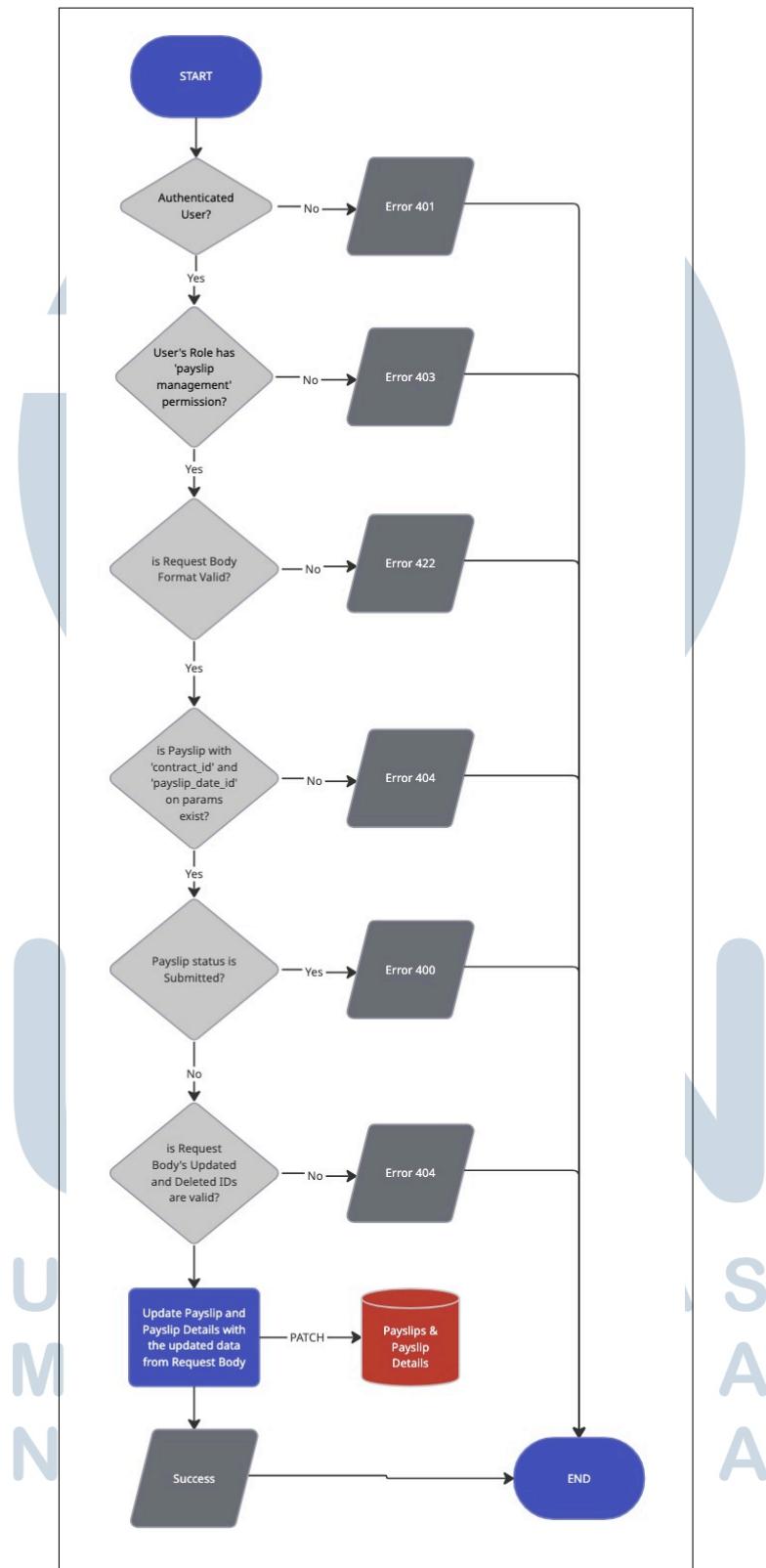
Gambar 3.39. Contoh *response API Get Payslip Detail*

Gambar 3.37 menunjukkan *flowchart* alur sistem API *Get Payslip Detail* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payslip/date/payslip\_date\_id/contract/contract\_id menggunakan metode HTTP *GET* seperti pada Gambar 3.38. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta mengambil data detail slip gaji (*payslip\_details*) berdasarkan *payslip\_date\_id* dan *contract.id* yang diberikan dari *path parameters*. Setelah didapatkan data *payslip*, sistem juga akan mengambil data *payslip\_details* yang berelasi dengan *payslip* tersebut. Jika data *payslip\_details* tidak ditemukan, maka sistem akan membuat data *payslip\_details* baru berdasarkan *contract\_id* dan *payslip\_date.id* yang diberikan. Data akan diambil dari *contract items* untuk komponen gaji yang ada pada kontrak tersebut, dan juga dari *daily\_penalty\_logs* untuk potongan denda harian yang terjadi pada periode slip gaji tersebut. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail slip gaji yang diminta seperti pada Gambar 3.39.

### ***Save Payslip Detail***

Berikut merupakan *flowchart* API *Save Payslip Detail*.





Gambar 3.40. Flowchart alur sistem API Save Payslip Detail

Berikut merupakan contoh *request* dan *response* untuk API *Save Payslip Detail*.

The screenshot shows the 'Save Payslip Detail' API endpoint documentation. It includes:

- Method:** PATCH /payslip/date/{payslip\_date\_id}/contract/{contract\_id}
- Status:** Developing
- Shared:** Shared
- Created:** September 9, 2025 | **Updated:** a few seconds ago | **Updated by:** JoseAndreasLie | **Creator:** JoseAndreasLie | **Maintainer:** Not configured | **Folder:** Payslip
- Description:** Save Payslip Detail with payslip\_date\_id and contract\_id
- Request:**
  - Authorization:** Provide your bearer token in the Authorization header when making requests to protected resources. Example: Authorization: Bearer \*\*\*\*\*
  - Path Params:**
    - payslip\_date\_id:** string (required) Example: c9b7d0cb-1e93-4b74-84b2-5c06cbf15cf0
    - contract\_id:** string (required) Example: 92878950-425f-471f-b404-6488bbe9dc18
  - Data Schema:**

```
{
  "income": {
    "create": [
      {
        "name": "Tambah",
        "amount": "200000",
        "description": "Tambah untuk anak dan istri.",
        "type": "ADDITION"
      }
    ],
    "update": [
      {
        "id": "97652567-c661-4275-a8c6-5e88e8c6ae3b",
        "name": "THR",
        "amount": "400000",
        "description": "Silakan digunakan untuk liburan spesial kamu."
      }
    ],
    "delete": []
  },
  "deduction": {
    "create": [
      {
        "name": "Pantry Fee",
        "amount": "30000",
        "description": "Menghabiskan semua makanan yang ada di pantry dalam 10 menit.",
        "type": "DEDUCTION"
      }
    ],
    "update": [],
    "delete": [
      {
        "id": "613f8164-b356-4e8b-9cbc-73925354ee8d"
      }
    ]
  }
}
```

Gambar 3.41. Contoh *request* API *Save Payslip Detail*

```
{
  "code": 200,
  "message": "Success",
  "data": [
    {
      "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
      "income": [
        {
          "id": "218b5cfc-7ad4-43e9-885e-f9cd82f3de35",
          "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
          "penalty_id": null,
          "type": "ADDITION",
          "name": "THR",
          "description": "Silakan digunakan untuk merayakan hari yang spesial!",
          "amount": "4000000",
          "source": "MANUAL",
          "source_id": null,
          "source_model": null,
          "created_at": "2025-12-20T06:54:56.457Z",
          "updated_at": "2025-12-20T06:56:20.034Z",
          "deleted_at": null
        },
        {
          "id": "643e35cb-2dbc-4786-8928-c6a31dde119",
          "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
          "penalty_id": null,
          "type": "BENEFIT",
          "name": "Tunjangan BPJS",
          "description": "Tunjangan yang diberikan untuk BPJS karyawan.",
          "amount": "35000",
          "source": "SYSTEM",
          "source_id": null,
          "source_model": null,
          "created_at": "2025-12-20T06:49:02.754Z",
          "updated_at": "2025-12-20T06:49:02.754Z",
          "deleted_at": null
        },
        {
          "id": "71ced7fd-e977-4651-aedd-096184447e2f",
          "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
          "penalty_id": null,
          "type": "ADDITION",
          "name": "Tunjangan Kehormatan",
          "description": "Tunjangan yang diberikan untuk jabatan karyawan.",
          "amount": "500000",
          "source": "SYSTEM",
          "source_id": null,
          "source_model": null,
          "created_at": "2025-12-20T06:49:02.754Z",
          "updated_at": "2025-12-20T06:49:02.754Z",
          "deleted_at": null
        },
        {
          "id": "3a3b02a8-cb41-4884-812f-4f24df6e3129",
          "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
          "penalty_id": null,
          "type": "ADDITION",
          "name": "Main Salary",
          "description": "Gaji Pokok",
          "amount": "10000000",
          "source": "SYSTEM",
          "source_id": null,
          "source_model": null,
          "created_at": "2025-12-20T06:49:02.754Z",
          "updated_at": "2025-12-20T06:49:02.754Z",
          "deleted_at": null
        }
      ],
      "deduction": [
        {
          ...
        }
      ]
    }
  ]
}
```

Gambar 3.42. Contoh response API Save Payslip Detail

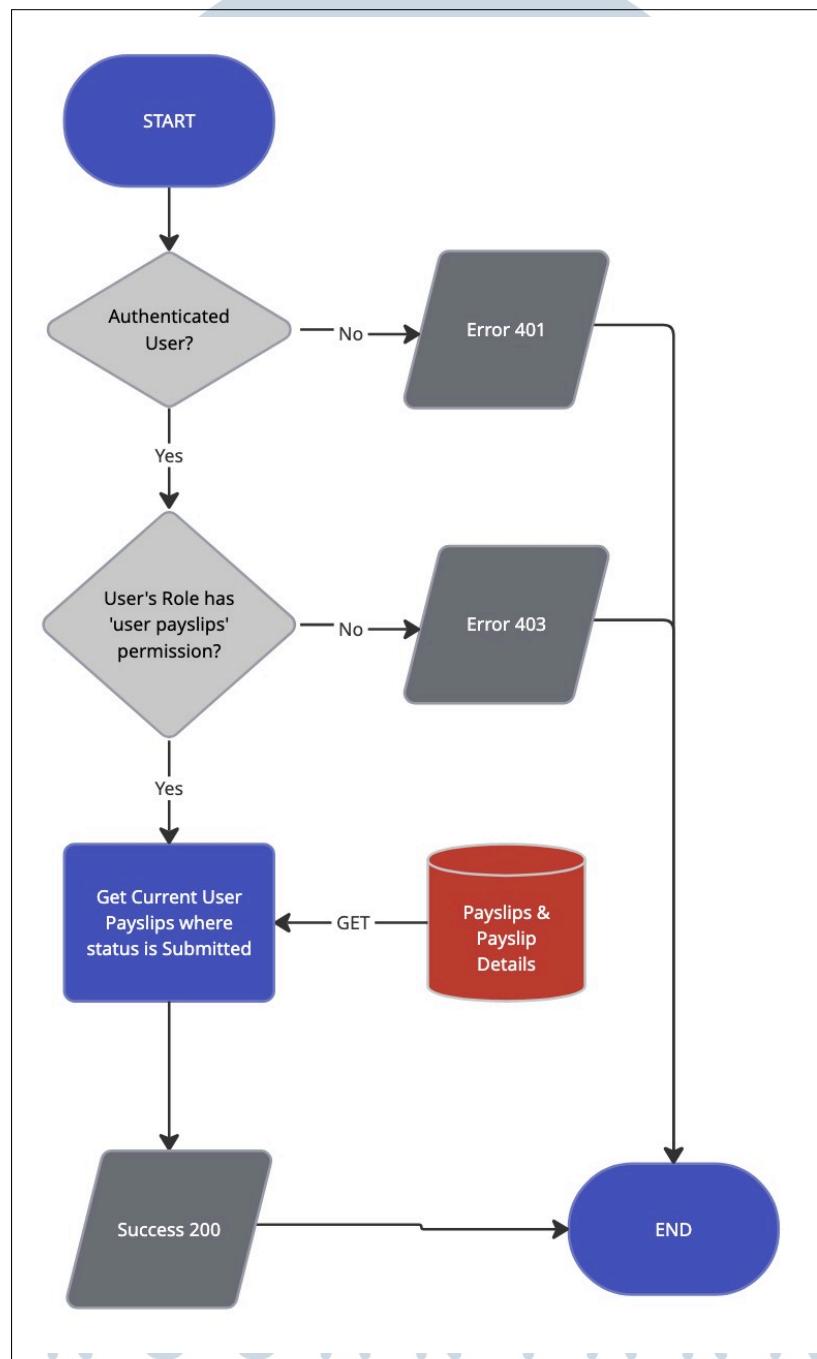
```
        "created_at": "2025-12-20T06:49:02.754Z",
        "updated_at": "2025-12-20T06:49:02.754Z",
        "deleted_at": null
    },
],
"deduction": [
{
    "id": "7ac84829-0956-462b-9eff-e13051c98b20",
    "payslip_id": "312e9e36-9148-4254-8c2b-72965b4e0586",
    "penalty_id": null,
    "type": "DEDUCTION",
    "name": "Kursi Gaming Kantor",
    "description": "Merusak Properti Kantor",
    "amount": "650000",
    "source": "MANUAL",
    "source_id": null,
    "source_model": null,
    "created_at": "2025-12-20T06:56:20.031Z",
    "updated_at": "2025-12-20T06:56:20.031Z",
    "deleted_at": null
},
],
"take_home_pay": 18385000
}
}
```

Gambar 3.43. Contoh *response* API Save Payslip Detail  
(Lanjutan)

Gambar 3.40 menunjukkan *flowchart* alur sistem API *Save Payslip Detail* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payslip/date/payslip\_date\_id/contract/contract\_id menggunakan metode HTTP *PATCH* seperti pada Gambar 3.41. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, memeriksa *request body* tersebut valid sesuai dengan ketentuan, serta memperbarui data detail slip gaji berdasarkan *payslip\_date\_id* dan *contract\_id* yang diberikan dari *path parameters* dengan data yang diberikan dalam *request body*. Di dalam *request body*, pengguna dapat mengirimkan beberapa data detail slip gaji yang ingin diperbarui, seperti tunjangan, potongan, dan lain-lain. Pengguna juga dapat menambahkan beberapa data detail slip gaji baru yang belum ada pada slip gaji tersebut dengan deskripsi dan nominal yang diinginkan. Setelah data detail slip gaji berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai detail slip gaji yang telah diperbarui seperti pada Gambar ??.

### Get User Payslips List

Berikut merupakan *flowchart* API Get User Payslips List.



Gambar 3.44. *Flowchart* alur sistem API Get User Payslips List

Berikut merupakan contoh *request* dan *response* untuk API Get User

### Payslips List.

User Payslip List

GET /payslip/user • Developing ▾

Shared

Created September 9, 2025 Updated a few seconds ago Updated by JoseAndreasLie Creator Vn1k Maintainer Not configured Folder Payslip

**Request**

Authorization

Provide your bearer token in the `Authorization` header when making requests to protected resources.  
Example: `Authorization: Bearer *****`

Query Params

`pagination` boolean required  
Example: `true`

`page` integer required  
Example: `1`

`row` integer required  
Example: `10`

Gambar 3.45. Contoh *request* API Get User Payslips List

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "count": 1,  
        "rows": [  
            {  
                "id": "312e9e36-9140-4254-8c2b-72965b4e0586",  
                "start_date": "2025-11-30T17:00:00.000Z",  
                "end_date": "2025-12-02T16:59:59.999Z",  
                "name": "HoHR 1",  
                "job_title": "Director",  
                "employment_status": "Full Time",  
                "created_at": "2025-12-02T16:59:59.999Z",  
                "updated_at": "2025-12-20T07:03:56.312Z"  
            }  
        ]  
    }  
}
```

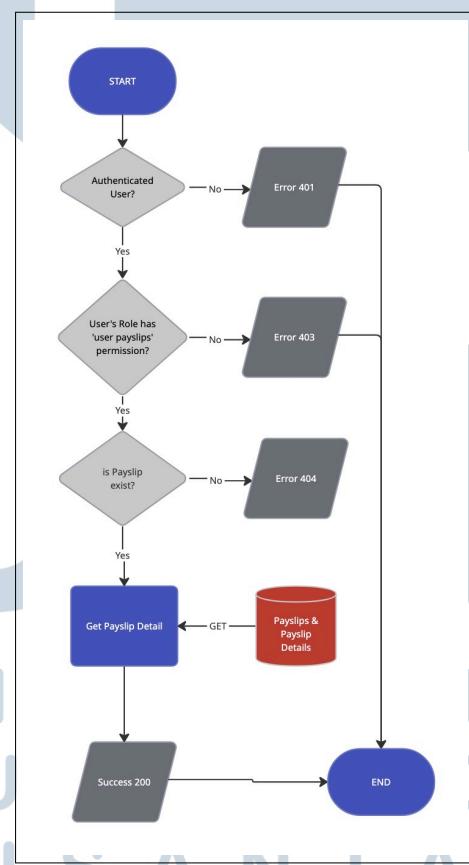
Gambar 3.46. Contoh *response* API Get User Payslips List

Gambar 3.44 menunjukkan *flowchart* alur sistem API Get User Payslips

*List* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /payslips/user menggunakan metode HTTP *GET* seperti pada Gambar 3.45. Dalam *query parameters* terdapat beberapa variabel yang harus diisi dan dapat diisi untuk memenuhi kebutuhan *pagination* supaya data yang diterima dapat dibatasi. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil data semua slip gaji milik pengguna. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar slip gaji milik pengguna seperti pada Gambar 3.46.

### **Get User Payslip Detail by ID**

Berikut merupakan *flowchart* API *Get User Payslip Detail by ID*.



Gambar 3.47. Flowchart alur sistem API *Get User Payslip Detail by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Get User Payslip Detail by ID*.

**User Payslip Detail**

**GET** /payslip/user/{payslip\_id} • Developing ✓

Shared  
Created September 9, 2025 Updated 3 months ago Updated by JoseAndreasLie Creator JoseAndreasLie  
Maintainer Not configured Folder Payslip

**Description**  
Only this user's submitted payslip.

**Request**

Authorization  
Provide your bearer token in the **Authorization** header when making requests to protected resources.  
Example: Authorization: Bearer \*\*\*\*\*

Path Params  
**payslip\_id** string required  
Example: 312e9e36-9140-4254-8c2b-72965b4e0586

Gambar 3.48. Contoh *request API Get User Payslip Detail by ID*

UMN  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "start_date": "2025-11-30T17:00:00.000Z",  
        "end_date": "2025-12-02T16:59:59.999Z",  
        "status": "SUBMITTED",  
        "user_data": {  
            "name": "HoHR 1",  
            "nik": "112233445566778899",  
            "contact": "08123456789",  
            "job_title": "Director",  
            "employment_status": "Full Time",  
            "attendance": 2  
        },  
        "income": [  
            {  
                "id": "218b5fcf-7ad4-43e9-885e-f9cd82f3de35",  
                "name": "THR",  
                "amount": "4000000",  
                "description": "Silakan digunakan untuk merayakan hari yang spesial!"  
            },  
            {  
                "id": "643e35cb-2dbc-4706-8928-c6a31ddea119",  
                "name": "Tunjangan BPJS",  
                "amount": "35000",  
                "description": "Tunjangan yang diberikan untuk BPJS karyawan."  
            },  
            {  
                "id": "71ced7fd-e977-4651-aedd-096184447e2f",  
                "name": "Tunjangan Kehormatan",  
                "amount": "500000",  
                "description": "Tunjangan yang diberikan untuk jabatan karyawan."  
            },  
            {  
                "id": "3a3b02a0-cb41-4884-812f-4f24df6e3129",  
                "name": "Main Salary",  
                "amount": "10000000",  
                "description": "Gaji Pokok"  
            }  
        ],  
        "deduction": [  
            {  
                "id": "7ac04829-0956-462b-9eff-e13051c98b20",  
                "name": "Kursi Gaming Kantor",  
                "amount": "650000",  
                "description": "Merusak Properti Kantor"  
            },  
            {  
                "id": "643e35cb-2dbc-4706-8928-c6a31ddea119",  
                "name": "Tunjangan BPJS",  
                "amount": "35000",  
                "description": "Tunjangan yang diberikan untuk BPJS karyawan."  
            }  
        ],  
        "take_home_pay": 18350000,  
        "bank_info": {  
            "bank_name": "Bank Central Asia",  
            "bank_account": "11223344556677",  
            "bank_account_name": "HoHR 1"  
        }  
    }  
}
```

Gambar 3.49. Contoh response API Get User Payslip Detail by ID

Gambar 3.47 menunjukkan *flowchart* alur sistem API *Get User Payslip Detail by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /*payslip/user/*:*id* menggunakan metode HTTP *GET* seperti pada Gambar 3.48. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta mengambil data detail slip gaji berdasarkan *ID* yang diberikan dari *path parameters*. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail slip gaji yang diminta seperti pada Gambar 3.49.

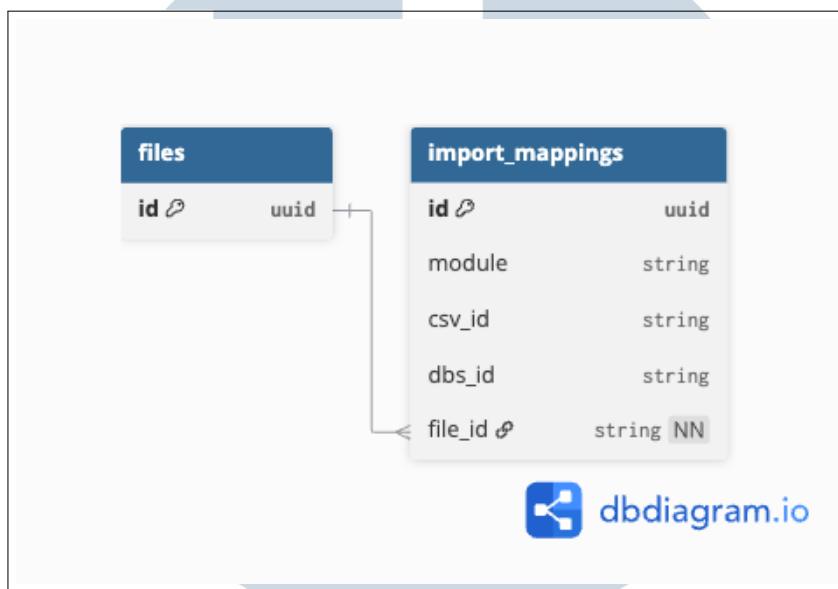
### 3.5.4 Modul On Boarding

Modul On Boarding dikembangkan untuk mempermudah proses penambahan data klien baru ke dalam sistem, dikembangkan modul On Boarding yang memungkinkan pengguna mengimpor data melalui *file CSV* dengan *template* yang sudah disiapkan. Modul ini mencakup fitur pemetaan kolom data dari berkas CSV ke struktur basis data yang sesuai, serta validasi data untuk memastikan integritas dan konsistensi informasi yang diimporkan. Dengan adanya modul ini, proses penambahan klien baru menjadi lebih efisien sehingga mempercepat waktu implementasi sistem bagi pengguna baru.

Alur utama dari modul *On Boarding* dimulai dari klien mengisi data utama seperti lokasi perusahaan (*office location*), jam kerja (*schedule*), peranan (*role*), pengguna (*user*), jabatan (*job title*), komponen gaji (*payroll item*), jenis cuti (*leave type*), dan aturan denda (*penalty rule*) di *template sheets* yang sudah disediakan. Setelah klien sudah mengisi data di *template*, klien tinggal mengunggah dari setiap sheetsnya sebagai berkas CSV. Setelah data utama diunggah, klien dapat mengunduh *template* baru, kontrak (*contract*) dan komponen gaji lainnya (*contract item*) untuk diisi dengan data yang sebelumnya sudah di siapkan. Setelah mengisi data pada *template* yang baru, klien dapat mengunduh berkas menjadi format CSV tersebut lalu mengunggahnya ke dalam sistem. Sistem kemudian memproses berkas yang diunggah dengan melakukan validasi data dan menyimpan data ke dalam basis data (tabel *Import Mappings* dan tabel modul masing-masing). Jika terdapat kesalahan pada data yang diunggah, sistem akan memberikan notifikasi kepada klien untuk memperbaiki data tersebut. Setelah data utama dan data kedua berhasil, sesi *On Boarding* sudah selesai dan sistem siap mulai digunakan.

## A Diagram ERD Modul *On Boarding*

Berikut merupakan *database diagram* untuk modul *On Boarding* yang telah dibuat selama pelaksanaan kerja praktik.



Gambar 3.50. Diagram ERD untuk modul *On Boarding*

Gambar 3.50 menunjukkan struktur basis data untuk modul *On Boarding*. Disini menggunakan table `import_mappings` untuk mencatat riwayat impor data yang dilakukan oleh pengguna. Setiap entri dalam tabel ini menyimpan informasi tentang pengguna yang melakukan impor, nama berkas (*file*) yang diunggah, serta pemetaan kolom yang digunakan selama proses impor. Hal ini memungkinkan pelacakan dan audit terhadap aktivitas impor data, serta memudahkan pengguna untuk mengelola dan meninjau pemetaan kolom yang telah mereka buat sebelumnya.

## B *On Boarding API Endpoints*

Berikut adalah daftar *endpoints* API yang telah dikembangkan untuk modul *On Boarding*:

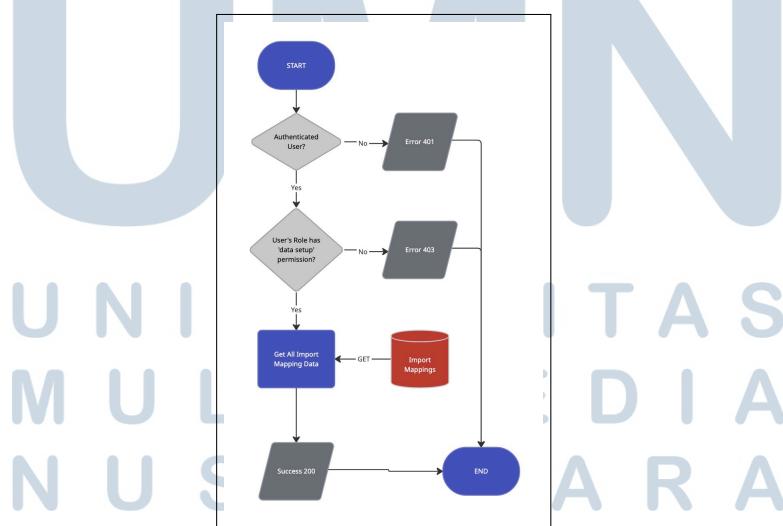
Nama API	Metode	Endpoint	Deskripsi
Get On Boarding	GET	/on-boarding	Mengambil semua data impor
On Boarding Import	POST	/on-boarding/import-csv	Mengimpor data klien baru dari berkas CSV
Download Second Batch Template	GET	/on-boarding/download-template	Mengunduh <i>template</i> Excel untuk impor batch kedua

Tabel 3.5. Daftar *endpoints* API untuk modul *On Boarding*

Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *On Boarding* seperti pada Tabel 3.5. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola proses impor data klien baru, mulai dari pengambilan data impor, hingga proses impor data dari berkas CSV. Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

### ***Get On Boarding***

Berikut merupakan *flowchart* API *Get On Boarding*.

Gambar 3.51. *Flowchart* alur sistem API *Get On Boarding*

Berikut merupakan contoh *request* dan *response* untuk API *Get On*

*Boarding.*



Gambar 3.52. Contoh *request* API *Get On Boarding*

```
{
  "code": 200,
  "message": "On Boarding fetched successfully",
  "data": [
    {
      "id": "2d7287cc-599b-4e11-9867-46f7803ff699",
      "module": "leave_type",
      "file_id": "06cd0a4c-074f-40d5-9cab-b630a1486c8c",
      "name": "levtp.csv",
      "created_at": "2025-11-27T12:18:35.450Z",
      "updated_at": "2025-11-27T12:18:35.450Z"
    },
    {
      "id": "6b2623e6-49f2-4385-b126-f07706360e5c",
      "module": "office_location",
      "file_id": "1ddcf5d9-1800-4812-a219-82c724161ebe",
      "name": "ofloc.csv",
      "created_at": "2025-11-27T12:18:35.397Z",
      "updated_at": "2025-11-27T12:18:35.397Z"
    },
    {
      "id": "25145f0f-be91-4674-8870-a378d74e6d04",
      "module": "role",
      "file_id": "20547a38-f510-490d-9e09-e6211e1b7bc1",
      "name": "role.csv",
      "created_at": "2025-11-27T12:18:35.228Z",
      "updated_at": "2025-11-27T12:18:35.228Z"
    }
  ]
}
```

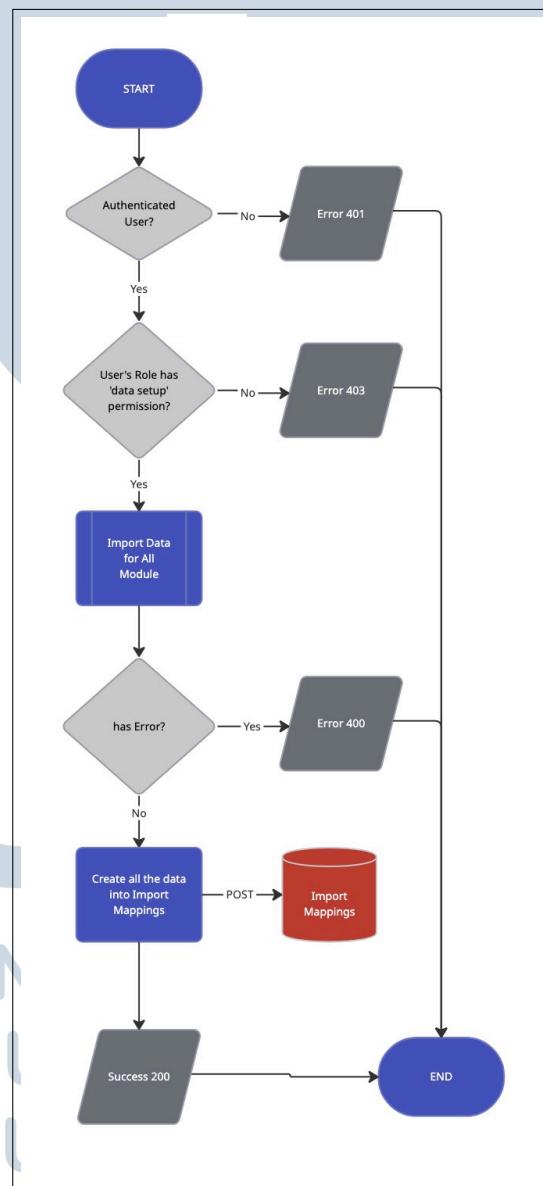
Gambar 3.53. Contoh *response* API *Get On Boarding*

Gambar 3.51 menunjukkan *flowchart* alur sistem API *Get On Boarding* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /on-boarding menggunakan metode HTTP *GET* seperti pada Gambar 3.52. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil data semua riwayat

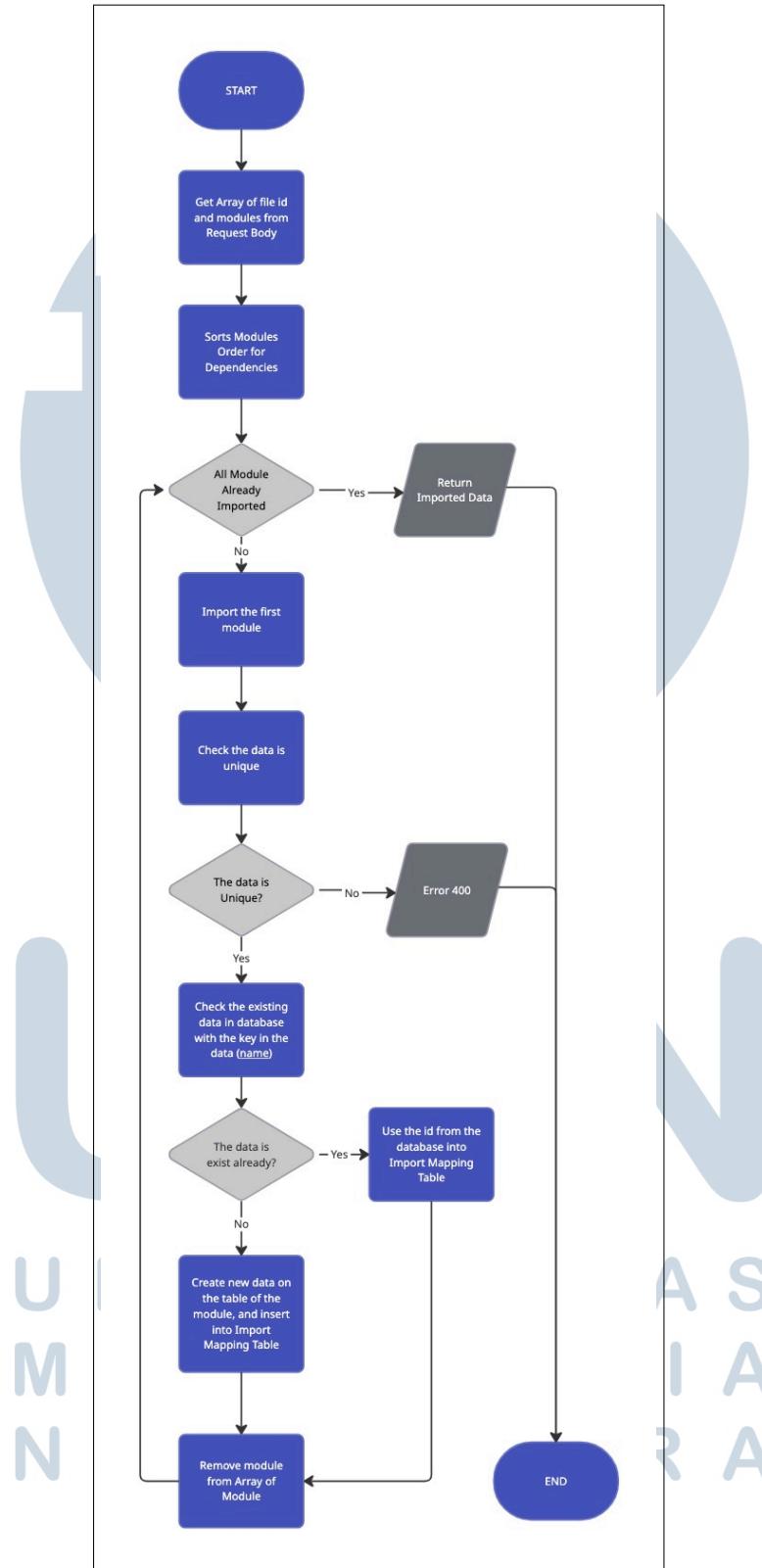
impor data klien baru. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar riwayat impor data seperti pada Gambar 3.53.

### On Boarding Import

Berikut merupakan *flowchart* API *On Boarding Import*.



Gambar 3.54. *Flowchart* alur sistem API *On Boarding Import*



Gambar 3.55. Flowchart alur sistem API On Boarding Import (Lanjutan)

Berikut merupakan contoh *request* dan *response* untuk API *On Boarding Import*.

The screenshot shows the "On Boarding Import CSV" API documentation. It includes a "Request" section with an "Example" tab containing a JSON object:

```
{
  "fileIds": [
    [
      "59feb015-e816-439e-906b-ce243510bfc1",
      "contract_item"
    ],
    [
      "f8efff7be-168f-4749-bbed-15d3a14fd974",
      "contract"
    ],
    [
      "bf94f0c2-4836-48ec-a2da-1b4afcc17c9b",
      "job_title"
    ],
    [
      "13a3b3e9-52d6-4f09-bb7a-5f9f1cd5173d",
      "leave_type"
    ],
    [
      "f82074a6-c8cb-4d41-848c-885ae038333a",
      "office_location"
    ],
    [
      "d9d7b42e-e4fa-45e8-b833-abfd8495f46d",
      "payroll_item"
    ],
    [
      "19648deb-ab90-4cdb-86ea-b3f3d63417c0",
      "role"
    ],
    [
      "6b4e17b4-b97e-4582-9762-6dd6204d7a4e",
      "schedule"
    ],
    [
      "5457794c-6090-4afd-9963-61ea24e4e275",
      "user"
    ]
  ]
}
```

Gambar 3.56. Contoh *request* API *On Boarding Import*

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "success": [  
            {  
                "module": "role",  
                "count": 4  
            },  
            {  
                "module": "user",  
                "count": 7  
            },  
            {  
                "module": "job_title",  
                "count": 4  
            },  
            {  
                "module": "office_location",  
                "count": 1  
            },  
            {  
                "module": "schedule",  
                "count": 61  
            },  
            {  
                "module": "payroll_item",  
                "count": 3  
            },  
            {  
                "module": "penalty_rule",  
                "count": 4  
            }  
        ],  
        "failed": [],  
        "mappings": {  
            "leave_type": {  
                "Annual": "deec7b77-ef46-4fc9-a4f1-6a5a42e21028",  
                "Force Majure": "df95f8ae-8403-43fa-ab69-58126a4a55f4",  
                "MFM": "cf3368cc-7d08-438a-8300-8a40e6f03b0d",  
                "MFM Director": "263c91a1-ed77-469c-9287-cebb5197e4b8"  
            },  
            "role": {  
                "Supervisor": "b6278a11-3666-4cc6-9c4c-f816742e79d0",  
                "Staff": "5f096215-7103-4f22-92bb-eb00de0c3d07",  
                "Human Resources": "bb447b00-4187-4ff4-b89c-83c0a89aba8b",  
                "Head of Human Resources": "f66ad300-42d4-4bf2-b74d-8df88e1e2250"  
            },  
            "user": {  
                "Staff 1": "720d357a-306f-48f5-937c-9b9d8d15cf39",  
                "Supervisor 1": "0721dd2a-c7b0-4da6-9f11-25891d1c756f",  
                "HR 1": "abd3bd7f-54f0-4579-ac48-533e64c0d1ec",  
                "HR 2": "fe73fe3-4501-4a66-bd73-731b10d48e41",  
                "HoMR 1": "8163393b-10bb-4af1-9ce1-73fd6748d816",  
                "Supervisor 2": "0a24dd480-e4d4-4c91-a15a-642d583f81ab",  
                "Staff 2": "3df396f2-6ce3-49ed-8d51-48d11a798197"  
            },  
            "job_title": {  
                "Human Resource": "8cafee98-22de-4d68-bb8b-5fb03d45373b",  
                "Director": "199c12ae-8622-45fd-a57d-08496fe8c226",  
                "Security": "32938e45-2a6b-4687-8fc4-af00d712cc2b",  
                "Marketing": "53f41a6a-0c8a-4e02-bade-7379d95526b7"  
            },  
            "office_location": {  
                "Tangerang Office @GOP": "26cd0609-8cb9-4856-a964-a6ee7920078e"  
            },  
            "schedule": {  
                "Senin - Jumat (09:00 - 18:00)": "f8a23a2c-cb1b-4fb4-9435-22450159d26c",  
                "Sabtu (09:00 - 16:00)": "5fd60eee6-5214-498e-b80f-d3ff724fe9e5",  
                "Lunch Time (12:00 - 13:00)": "2232f42f-19ea-43e0-85db-6783411f243a",  
                "3 Shift": "5e6b4039-2afe-4b12-941f-2b9a4cd8cd7c",  
                "2 Shift": "cf690dc8-3c6e-4966-bbe6-f345eb71ef6d",  
                "Senin - Rabu (08:00 - 17:00)": "157abccb-17e3-4eb8-a63a-381bad3693f8",  
                "Senin - Minggu (08:00 - 16:00)": "029d36f0-a003-d479-9f1c-7eb06161a6461"  
            }  
        }  
    }  
}
```

Gambar 3.57. Contoh response API On Boarding Import

Gambar 3.54 menunjukkan *flowchart* alur sistem API *On Boarding Import* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /on-boarding/import-csv menggunakan metode HTTP *POST* seperti pada Gambar 3.56. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, lalu memproses data dari *request body* untuk mengimpor data yang sudah diunggah ke dalam table modul masing-masing. Setelah data berhasil diimpor, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai hasil impor data seperti pada Gambar 3.57. Untuk proses pada gambar 3.55 merupakan proses untuk memastikan tidak ada data yang duplikat dan memproses memasukkan data dari masing-masing modul.

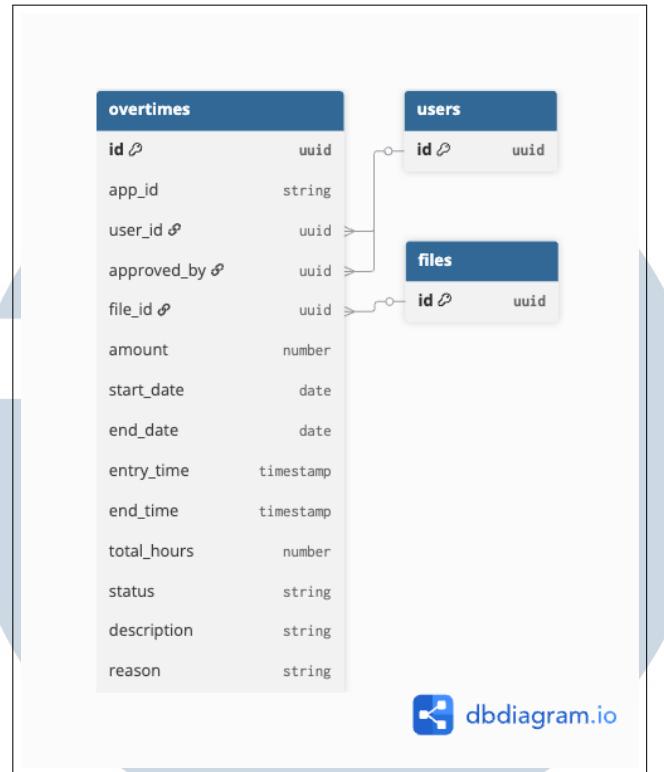
### 3.5.5 Modul Overtime

Modul Overtime dikembangkan untuk mengelola dan mencatat jam kerja lembur pegawai secara efisien. Modul ini mencakup fitur pengajuan lembur, persetujuan oleh atasan, serta memasukkan kompensasi lembur sesuai dengan kebijakan perusahaan. Struktur basis data dirancang untuk menyimpan informasi terkait jam lembur, termasuk tanggal, durasi, dan status persetujuan. Dengan adanya modul ini, perusahaan dapat memantau dan mengelola jam kerja lembur secara transparan dan akurat.

#### A Diagram ERD Modul Overtime

Berikut merupakan *database diagram* untuk modul *Overtime* yang telah dibuat selama pelaksanaan kerja praktik.

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA



1  
Gambar 3.58. Diagram ERD untuk modul Overtime

Gambar 3.58 menunjukkan struktur basis data untuk modul Overtime.

## B *Overtime API Endpoints*

Berikut adalah daftar *endpoints* API yang telah dikembangkan untuk modul *Overtime*:

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

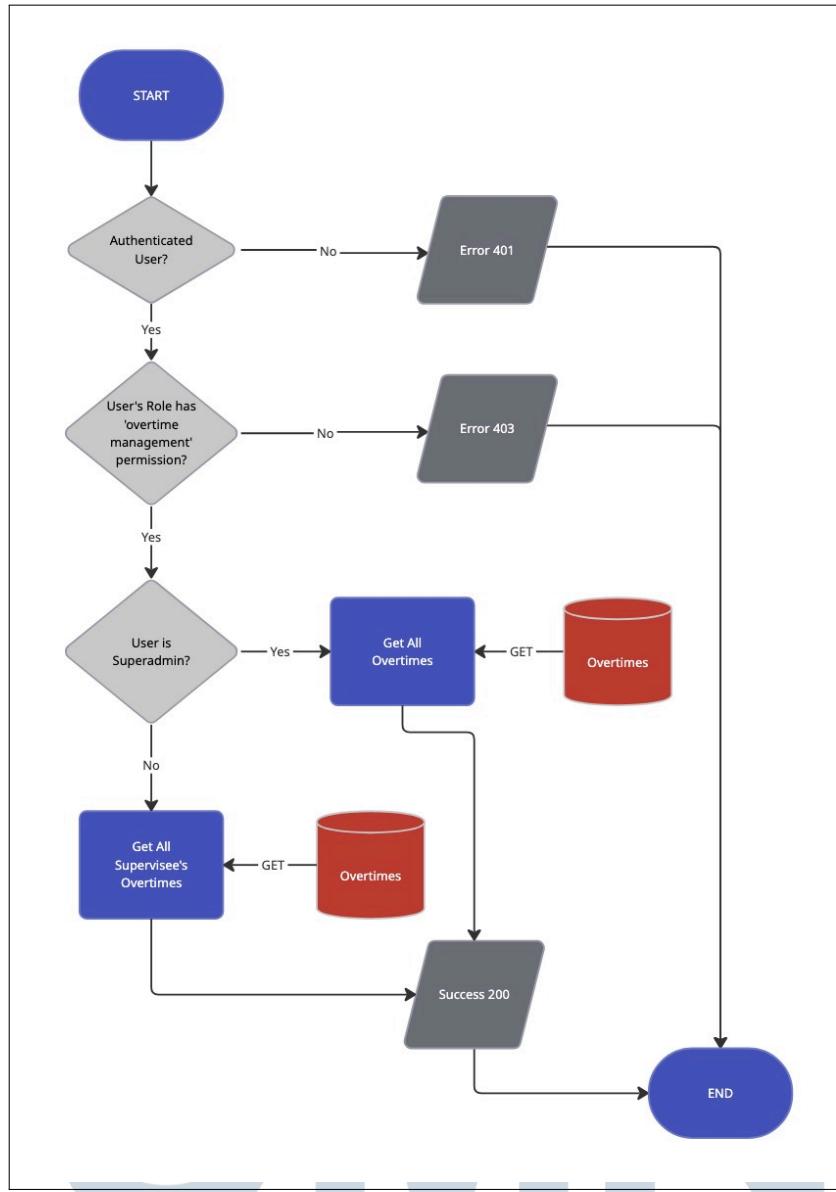
Nama API	Metode	Endpoint	Deskripsi
Get Overtime List	GET	/overtime	Mengambil semua permintaan lembur
Get User Overtime List	GET	/overtime/user	Mengambil permintaan lembur berdasarkan pengguna
Get Overtime Detail by ID	GET	/overtime/:id	Mengambil detail permintaan lembur berdasarkan ID
Delete Overtime by ID	DELETE	/overtime/:id	Menghapus permintaan lembur berdasarkan ID

Tabel 3.6. Daftar *endpoints* API untuk modul *Overtime*

Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Overtime* seperti pada Tabel 3.6. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola permintaan lembur pegawai, mulai dari pengambilan data lembur, hingga penghapusan permintaan lembur. Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

#### *Get Overtime List*

Berikut merupakan *flowchart* API *Get Overtime List*.



Gambar 3.59. Flowchart alur sistem API Get Overtime List

Berikut merupakan contoh *request* dan *response* untuk API *Get Overtime List*.

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

Get All Overtime

GET /overtime/ • Released

Shared

Created November 6, 2025 Updated a few seconds ago Updated by JoseAndreasLie Creator JoseAndreasLie Maintainer Not configured Folder Overtime

Request

Authorization

Provide your bearer token in the `Authorization` header when making requests to protected resources.

Example: Authorization: Bearer \*\*\*\*

Query Params

`pagination` boolean required

Example: true

`page` string required

Example: 1

`row` string required

Example: 10

`search` string optional

Example: Calista Belva

`sort` enum<string> optional

Allowed values: start\_date:ASC start\_date:DESC end\_date:ASC end\_date:DESC total\_hours:ASC total\_hours:DESC

app\_id:ASC app\_id:DESC

Example: total\_hours:DESC

`filter[status]` string optional

Example: Pending

Gambar 3.60. Contoh request API Get Overtime List

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "count": 2,  
        "rows": [  
            {  
                "id": "399b26ea-b659-4261-a660-385a9f2f719b",  
                "app_id": "0T-202511-010",  
                "name": "Calista Belva",  
                "start_date": "2025-11-06T17:00:00.000Z",  
                "end_date": "2025-11-06T17:00:00.000Z",  
                "entry_time": "09:00:00",  
                "end_time": "17:00:00",  
                "total_hours": "8",  
                "status": "Pending"  
            },  
            {  
                "id": "069cb6c5-32fe-4cda-b1d4-c5deb6cc5258",  
                "app_id": "0T-202511-009",  
                "name": "Calista Belva",  
                "start_date": "2025-11-05T17:00:00.000Z",  
                "end_date": "2025-11-05T17:00:00.000Z",  
                "entry_time": "09:00:00",  
                "end_time": "17:00:00",  
                "total_hours": "8",  
                "status": "Pending"  
            }  
        ]  
    }  
}
```

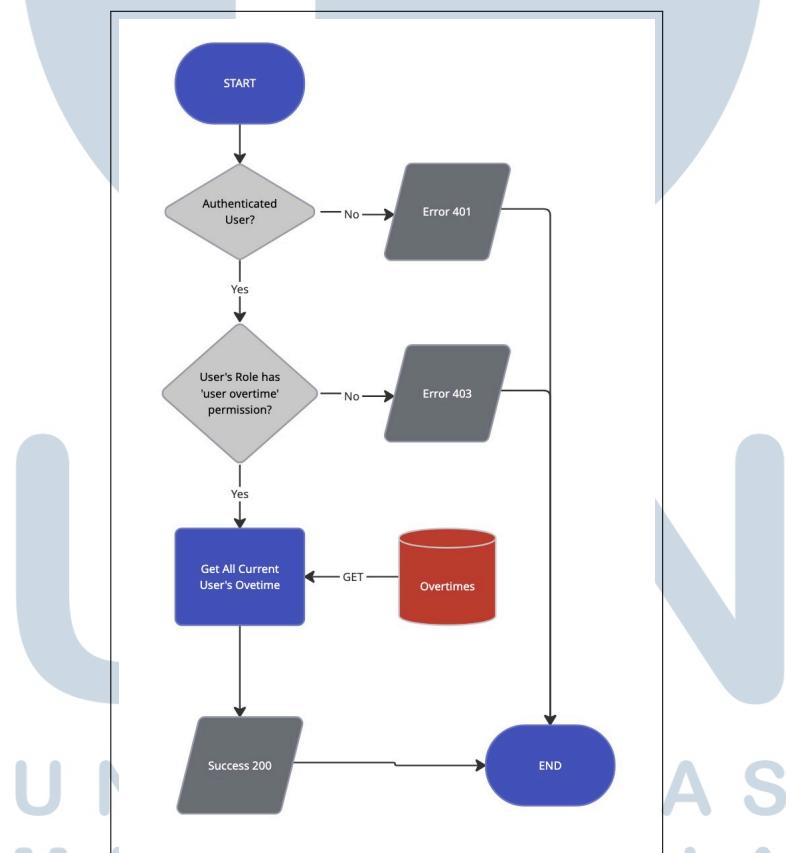
Gambar 3.61. Contoh response API Get Overtime List

Gambar 3.59 menunjukkan *flowchart* alur sistem API *Get Overtime List* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint*

/overtime menggunakan metode HTTP *GET* seperti pada Gambar 3.60. Dalam *query parameters* terdapat beberapa variabel yang harus diisi dan dapat diisi untuk memenuhi kebutuhan *pagination* supaya data yang diterima dapat dibatasi. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil data semua permintaan lembur. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar permintaan lembur seperti pada Gambar 3.61.

### ***Get User Overtime List***

Berikut merupakan *flowchart* API *Get User Overtime List*.



Gambar 3.62. Flowchart alur sistem API *Get User Overtime List*

Berikut merupakan contoh *request* dan *response* untuk API *Get User Overtime List*.

The screenshot shows the Turnitin API documentation for the 'User Get All' endpoint. It includes a 'Request' section with fields like 'Authorization' (example: Authorization: Bearer \*\*\*\*\*) and 'Query Params' such as 'pagination' (true), 'page' (1), 'row' (10), 'search' (string), 'sort' (string), and 'filter[status]' (pending). The 'Query Params' section also includes examples for 'app\_id:ASC' and 'app\_id:DESC'.

Gambar 3.63. Contoh *request* API Get User Overtime List

The screenshot shows a sample JSON response for the 'User Get All' endpoint. The response structure is as follows:

```
{  
    "code": 200,  
    "message": "Success",  
    "data": {  
        "count": 2,  
        "rows": [  
            {  
                "id": "399b26ea-b659-4261-a660-385a9f2f719b",  
                "app_id": "0T-202511-010",  
                "name": "Calista Belva",  
                "start_date": "2025-11-06T17:00:00.000Z",  
                "end_date": "2025-11-06T17:00:00.000Z",  
                "entry_time": "09:00:00",  
                "end_time": "17:00:00",  
                "total_hours": "8",  
                "status": "Pending"  
            },  
            {  
                "id": "069cb6c5-32fe-4cda-b1d4-c5deb6cc5258",  
                "app_id": "0T-202511-009",  
                "name": "Calista Belva",  
                "start_date": "2025-11-05T17:00:00.000Z",  
                "end_date": "2025-11-05T17:00:00.000Z",  
                "entry_time": "09:00:00",  
                "end_time": "17:00:00",  
                "total_hours": "8",  
                "status": "Pending"  
            }  
        ]  
    }  
}
```

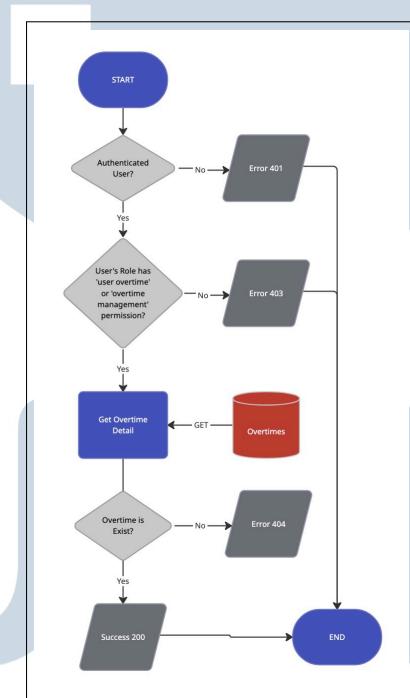
Gambar 3.64. Contoh *response* API Get User Overtime List

Gambar 3.62 menunjukkan *flowchart* alur sistem API Get User Overtime List yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint*

/overtime/user menggunakan metode HTTP *GET* seperti pada Gambar 3.63. Dalam *query parameters* terdapat beberapa variabel yang harus diisi dan dapat diisi untuk memenuhi kebutuhan *pagination* supaya data yang diterima dapat dibatasi. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna dan mengambil data semua permintaan lembur milik pengguna. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar permintaan lembur milik pengguna seperti pada Gambar 3.64.

### **Get Overtime Detail by ID**

Berikut merupakan *flowchart* API *Get Overtime Detail by ID*.



Gambar 3.65. *Flowchart* alur sistem API *Get Overtime Detail by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Get Overtime Detail by ID*.

The screenshot shows the API endpoint for retrieving overtime details by ID. It includes the method (GET), URL (/overtime/{id}), and status (Developing). It also shows the last updated date (November 6, 2025) and the creator (JoseAndreasLie). The interface includes sections for Request (Authorization header), Path Params (id), and Example responses.

1

Gambar 3.66. Contoh request API Get Overtime Detail by ID

```
{
  "code": 200,
  "message": "Success",
  "data": {
    "id": "703de794-0a2b-431f-ba83-4566168fd1ac",
    "status": "approved",
    "name": "HoHR 1",
    "start_date": "2025-12-17",
    "end_date": "2025-12-17",
    "entry_time": "18:00",
    "end_time": "22:00",
    "total_hours": "3",
    "amount": "200000",
    "description": "Lembur karena disuruh Owner.",
    "reason": null,
    "approved_by": "Superadmin",
    "created_at": "2025-12-21T12:12:42.705Z",
    "updated_at": "2025-12-21T12:15:43.349Z",
    "user_id": "5a5ecd8d-e754-4e5c-8a6b-0e086a46dd6b",
    "file": {
      "id": "20cf480a-eb2f-4945-9fc6-7c0b7b0b0b8e",
      "name": "BuktiLembur001.jpg"
    }
  }
}
```

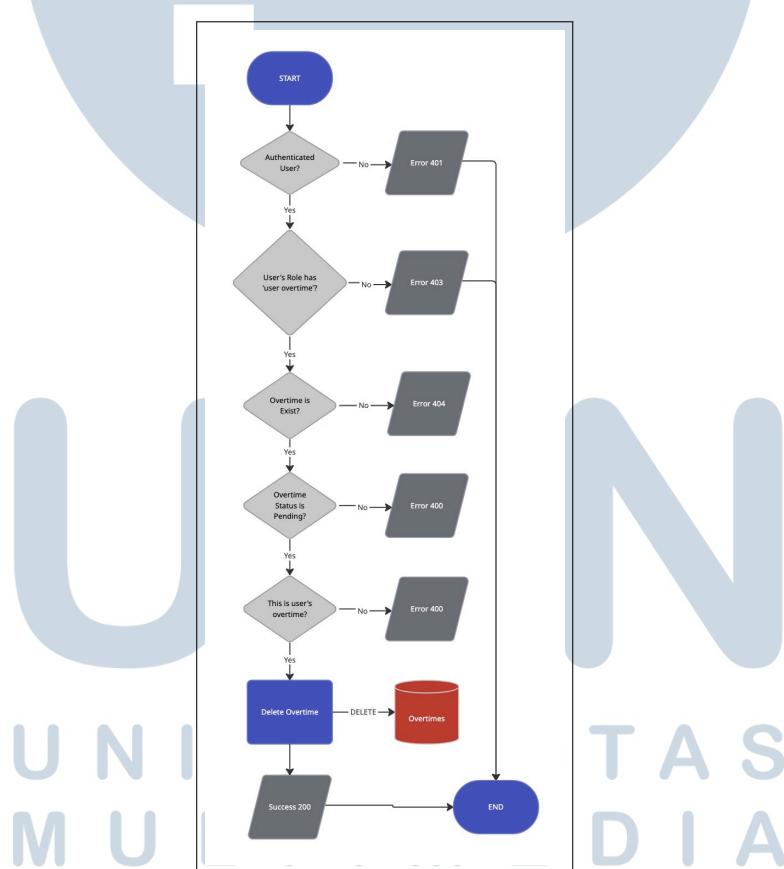
Gambar 3.67. Contoh response API Get Overtime Detail by ID

Gambar 3.65 menunjukkan flowchart alur sistem API Get Overtime

*Detail by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /overtime/:id menggunakan metode HTTP *GET* seperti pada Gambar 3.66. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta mengambil data detail permintaan lembur berdasarkan *ID* yang diberikan dari *path parameters*. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail permintaan lembur yang diminta seperti pada Gambar 3.67.

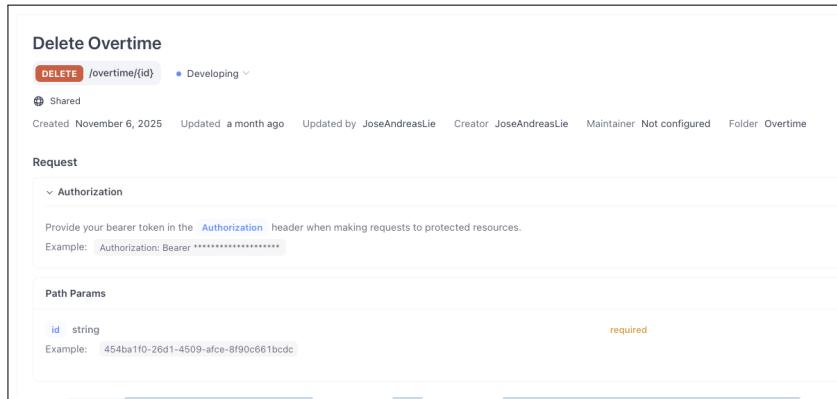
### **Delete Overtime by ID**

Berikut merupakan *flowchart* API *Delete Overtime by ID*.



Gambar 3.68. *Flowchart* alur sistem API *Delete Overtime by ID*

Berikut merupakan contoh *request* dan *response* untuk API *Delete Overtime by ID*.



Gambar 3.69. Contoh *request* API *Delete Overtime* by *ID*



Gambar 3.70. Contoh *response* API *Delete Overtime* by *ID*

Gambar 3.68 menunjukkan *flowchart* alur sistem API *Delete Overtime by ID* yang dimulai dari pengguna mengirimkan permintaan (*request*) ke *endpoint* /overtime/:id menggunakan metode HTTP *DELETE* seperti pada Gambar 3.69. Sistem kemudian memproses permintaan tersebut dengan memeriksa autentikasi pengguna, memeriksa *permission* pengguna, serta menghapus data permintaan lembur berdasarkan *ID* yang diberikan dari *path parameters*. Setelah data permintaan lembur berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai keberhasilan penghapusan permintaan lembur seperti pada Gambar 3.70.

1

### 3.6 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kerja praktik, terdapat beberapa kendala yang dihadapi dalam pengembangan sistem, antara lain:

- Beberapa *edge case* tidak teridentifikasi pada tahap pengembangan dan pengujian awal. Akibatnya, ketika sistem telah memasuki tahap produksi, masih ditemukan *bug* yang sebelumnya tidak terdeteksi. Hal ini berpotensi mengganggu stabilitas sistem dan pengalaman pengguna.
- Fokus pengembangan diarahkan pada pencapaian target agar klien dapat segera menggunakan modul utama, khususnya modul *Contract* dan *Payslip*. Kondisi ini menyebabkan keterbatasan waktu untuk melakukan *refactoring* kode lama serta perbaikan *bug* pada modul-modul yang telah dikembangkan sebelumnya.

Untuk mengatasi kendala-kendala tersebut, beberapa solusi yang diterapkan dan direkomendasikan adalah sebagai berikut:

- Pengembangan sistem dilengkapi dengan pengujian otomatis, seperti *unit testing*, untuk membantu mendeteksi *edge case* sejak tahap awal pengembangan. Dengan adanya pengujian otomatis, potensi *bug* dapat diidentifikasi lebih cepat sebelum sistem diterapkan ke lingkungan produksi.
- Untuk mengatasi keterbatasan waktu akibat prioritas pengembangan fitur utama, dilakukan perencanaan *refactoring* dan perbaikan *bug* secara bertahap. Pendekatan ini memungkinkan tim untuk tetap memenuhi target implementasi klien tanpa mengabaikan kualitas kode dan stabilitas modul yang telah ada.

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

## BAB 4

### SIMPULAN DAN SARAN

#### 4.1 Simpulan

Pengembangan dari sisi *backend* pada sistem informasi kepegawaian Nine to Six di PT Visi Karya Nusantara telah berhasil dilakukan dan diimplementasikan sesuai dengan kebutuhan yang telah ditetapkan. Selama pengembangan, terdapat *framework* dan teknologi yang digunakan seperti *ExpressJs* untuk membangun *API*, *PostgreSQL* sebagai basis data, serta *Sequelize* sebagai *ORM* untuk memudahkan interaksi dengan basis data. Beberapa modul utama yang telah dikembangkan meliputi modul *Contract* untuk manajemen kontrak karyawan, modul *Payslip* untuk pengelolaan slip gaji, serta modul *Attendance* untuk pencatatan kehadiran karyawan. Pengembangan ini juga melibatkan integrasi dengan sistem yang sudah ada, serta penerapan praktik terbaik dalam pengembangan perangkat lunak seperti penggunaan *version control* dengan Git dan penerapan *code review* untuk memastikan kualitas kode yang dihasilkan.

#### 4.2 Saran

Ada beberapa saran yang dapat dipertimbangkan untuk pengembangan sistem informasi kepegawaian Nine to Six ke depannya:

1. Disarankan untuk menambahkan *unit testing* pada setiap fungsi atau modul yang dikembangkan. Hal ini akan memastikan bahwa setiap perubahan kode dapat langsung tervalidasi, sehingga meminimalkan risiko *bug* yang tidak terdeteksi.
2. Saling integrasi untuk modul-modul yang sudah ada seperti *Leave Permit*, *Overtime*, dan *Payslip*. Karena modul *Payslip* bisa mengambil data dari modul-modul tersebut seperti data lembur dari modul *Overtime* dan data cuti dari modul *Leave Permit*.