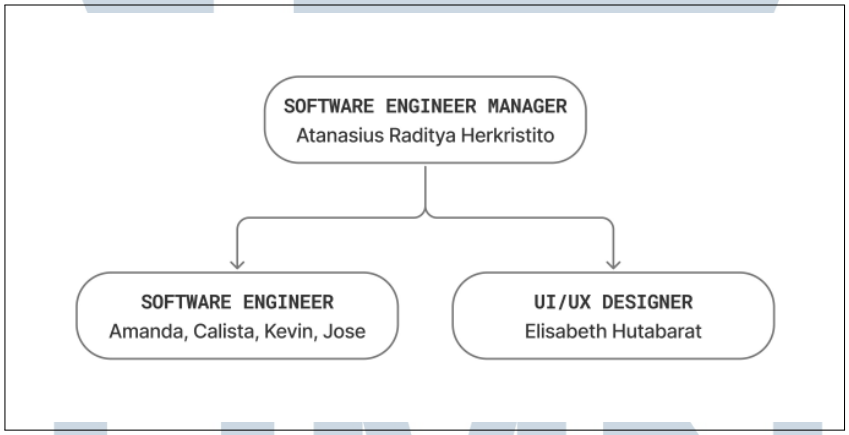
# BAB 3
# PELAKSANAAN KERJA MAGANG

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

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

| **Minggu Ke -** | **Pekerjaan yang dilakukan** |
|---|---|
| 4 | Revamp pada modul *User Management*, *Leave Management*, dan *Permit Detail*, 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

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. |

## 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.

## 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.

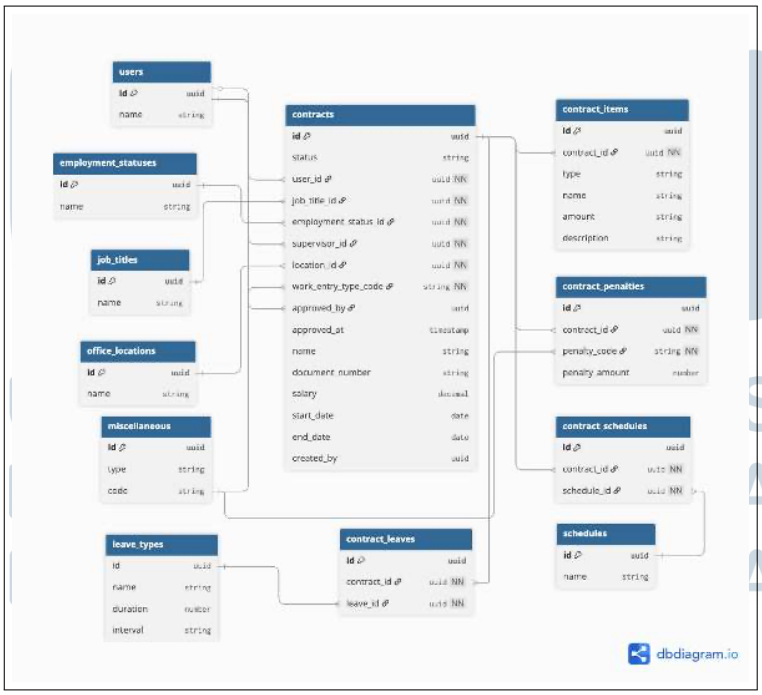
#### A Diagram ERD Modul *Contract*

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

Pada 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.
- *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.




Gambar 3.2. Diagram ERD untuk modul *Contract*

## B *Contract* API *Endpoints*

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

Tabel 3.2. Daftar *endpoints* API 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 *user* berdasarkan ID |
| Approve Contract by ID | PATCH | /contract/:id/approve | Menyetujui kontrak *user* berdasarkan ID |
| Terminate Contract by ID | DELETE | /contract/:id | Mengakhiri kontrak *user* berdasarkan ID |
| Delete Contract by ID | DELETE | /contract/:id/delete | Menghapus kontrak *user* berdasarkan ID |

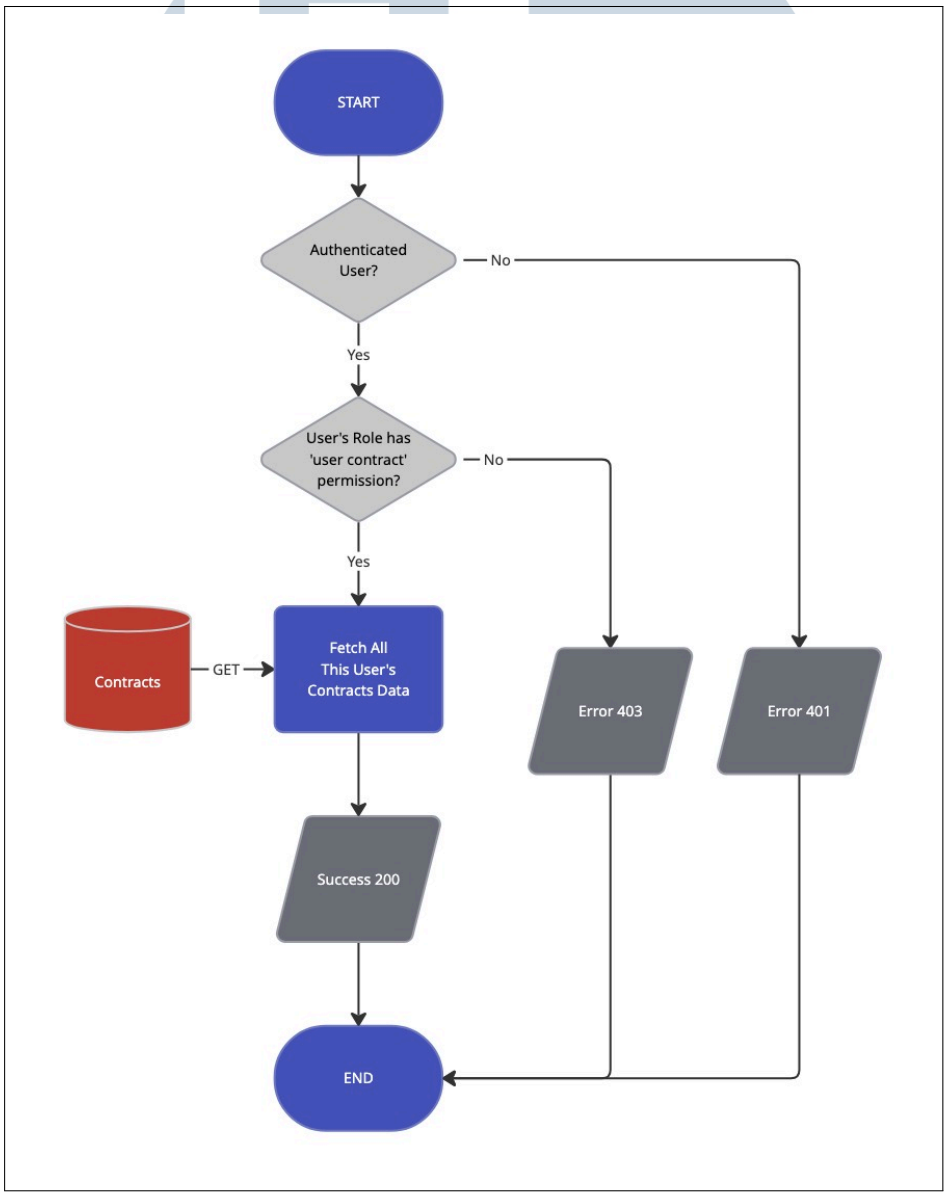
Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Contract* seperti pada Tabel 3.2. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola kontrak kerja pegawai, mulai dari pengambilan data kontrak, persetujuan, hingga penghapusan kontrak. Selanjutnya, akan dijelaskan alur sistem untuk *endpoints* dalam pada modul ini.

### *Get All User's Contracts*

*Get All User's Contracts* adalah *endpoint* API yang berfungsi untuk mengambil semua kontrak yang dimiliki oleh pengguna tertentu. API ini digunakan di sisi *frontend* untuk menampilkan daftar kontrak yang terkait dengan pengguna yang sedang masuk (*logged in user*) dalam page *Contract*. Gambar 3.4 dan 3.5 menunjukkan contoh *request* dan *response* untuk API *Get All User's Contracts*.

*Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan

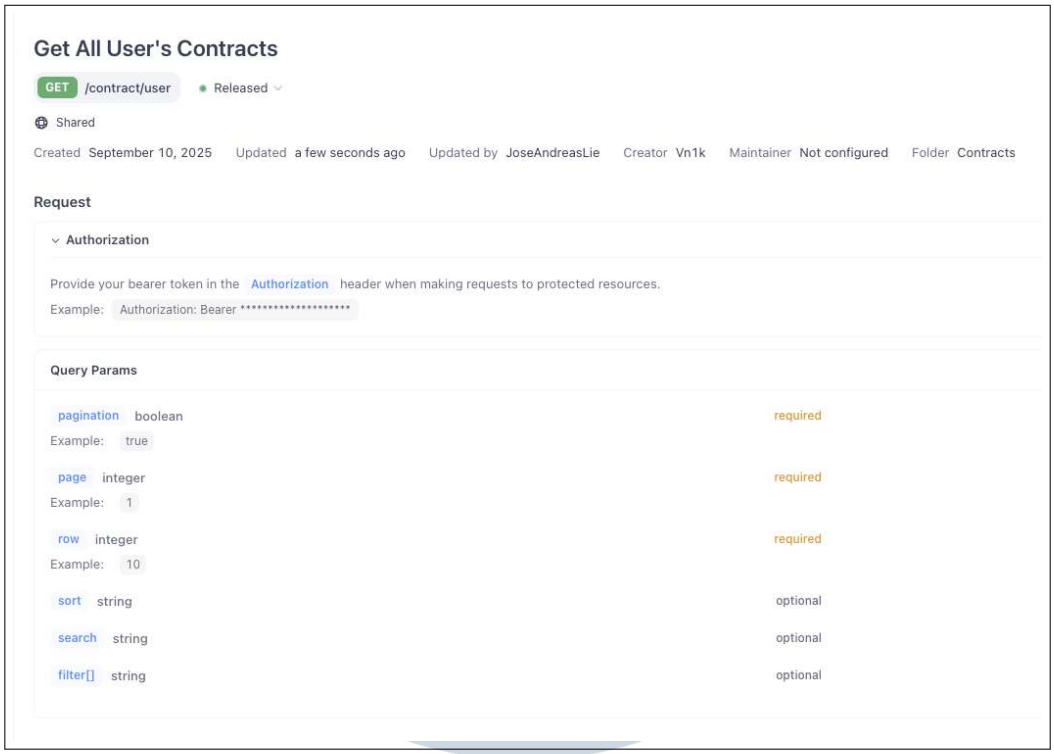
memeriksa *permission user contract* untuk mengakses data kontrak. Jika pengguna tidak ada *permission user contract*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data kontrak dari basis data berdasarkan ID pengguna yang terautentikasi. Data kontrak yang diambil kemudian dikemas dalam format JSON dan dikirimkan kembali sebagai respons kepada pengguna.




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*. Pada gambar 3.4 menunjukkan contoh *request* yang dikirimkan ke *endpoint* `/contract/user` menggunakan metode HTTP *GET* dengan menyertakan

*query parameters* untuk *pagination*.




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

Pada gambar 3.5 menunjukkan contoh *response* yang diterima dari *server*, berisi daftar kontrak yang dimiliki oleh pengguna beserta informasi terkait seperti *contract ID*, nama kontrak, status kontrak, *start date*, *end date*, nama karyawan, pekerjaan karyawan, *employment status* karyawan, dan karyawan yang membuat kontrak tersebut.

UNIVERSITAS MULTIMEDIA NUSANTARA

```
{
    "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-d9657cfa9f1d",
                    "name": "Full Time"
                },
                "created": {
                    "id": "b3f5713e-2fca-4e63-8f3c-69dab2341633",
                    "fullname": "Supervisor 1"
                }
            }
        ]
    }
}
```

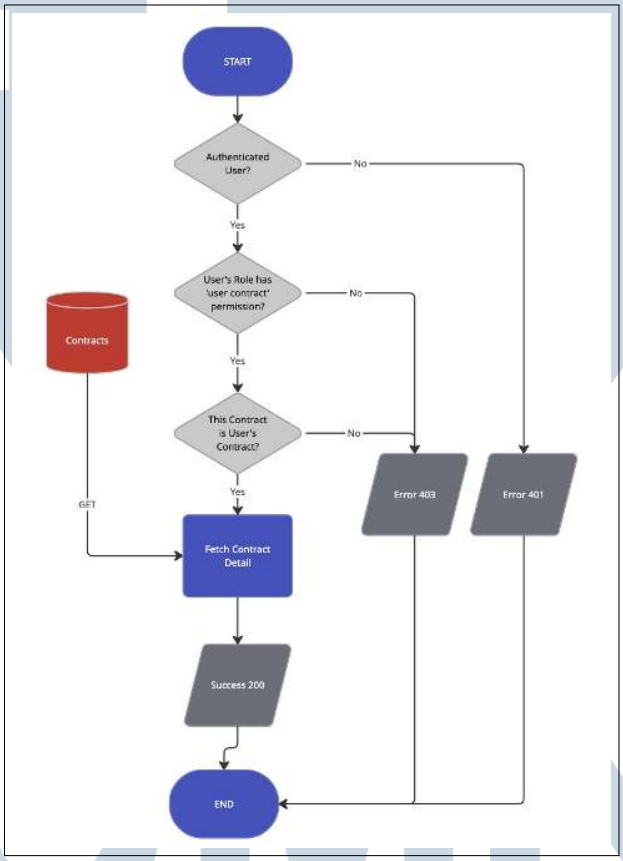

Gambar 3.5. Contoh *response* API *Get All User's Contracts*

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

*Get User's Contract Detail by ID* adalah *endpoint* API yang berfungsi untuk mengambil detail kontrak milik pengguna berdasarkan *ID* kontrak yang diberikan. API ini digunakan di sisi *frontend* untuk menampilkan informasi rinci mengenai kontrak tertentu dalam page *Contract Detail*.

Pada gambar 3.6 merupakan *flowchart* API *Get User's Contract Detail by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission user contract* untuk mengakses data kontrak. Jika pengguna tidak ada *permission user contract*, sistem akan mengembalikan *error* 403. Setelah

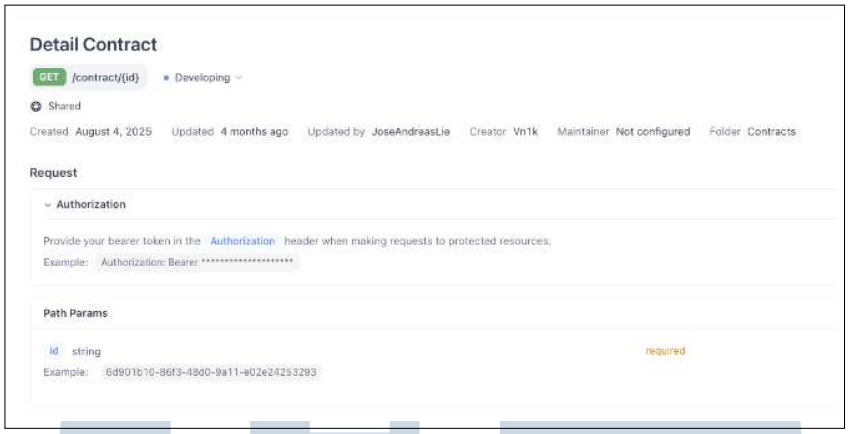
itu, sistem akan memeriksa apabila pemilik kontrak yang diminta tidak sama dengan pengguna yang melakukan permintaan. Jika tidak, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data detail kontrak dari basis data berdasarkan *ID* kontrak yang diberikan dari *path parameters*, `contract_id`. Data detail kontrak yang diambil kemudian dikemas dalam format JSON dan dikirimkan kembali sebagai respons kepada pengguna.




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

Pada gambar 3.7 menunjukkan contoh *request* yang dikirimkan ke *endpoint* `/contract/user/:id` menggunakan metode HTTP *GET* dengan menyertakan *path parameters*, `contract_id`, untuk menentukan kontrak yang ingin diambil detailnya.

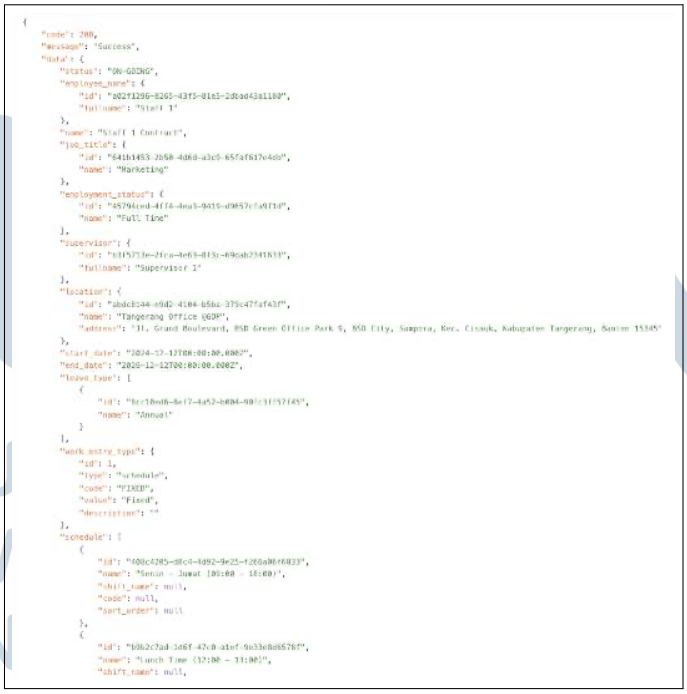
UNIVERSITAS MULTIMEDIA NUSANTARA




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

Pada gambar 3.8 menunjukkan contoh *response* yang diterima dari *server*, berisi informasi rinci mengenai kontrak tertentu, termasuk data kontrak, data pegawai terkait, jadwal kerja, jenis cuti, rincian komponen gaji kontrak (*contract items*), serta aturan penalti yang berlaku dalam kontrak tersebut.

Berikut merupakan contoh *response* untuk API *Get User's Contract Detail by ID*.




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

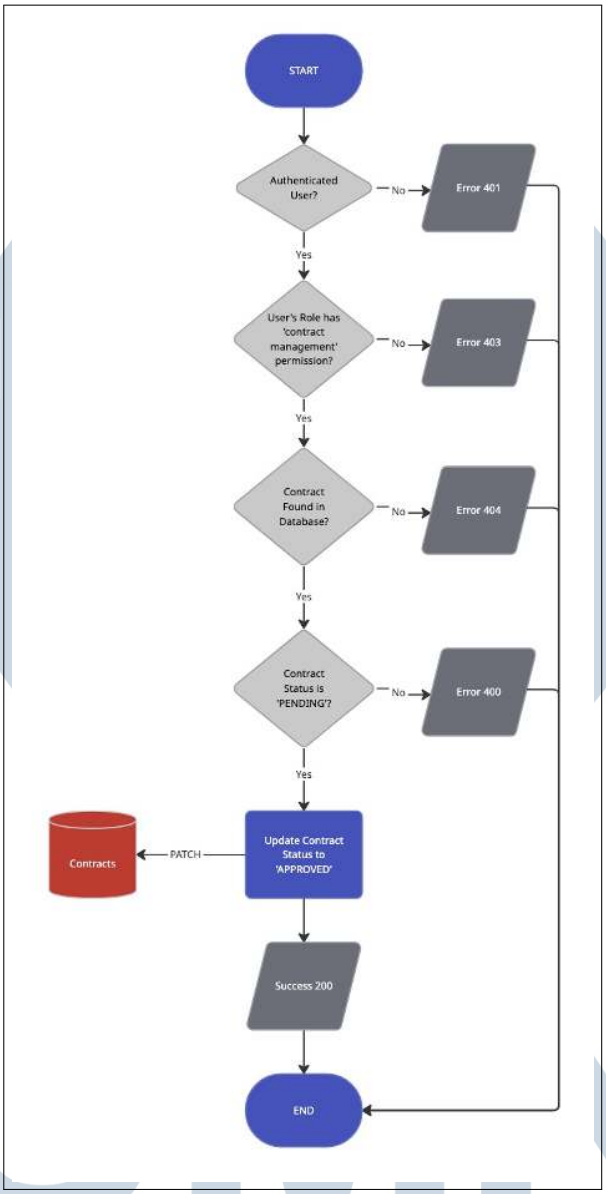
.

***Approve Contract by ID***

*Approve Contract by ID* adalah *endpoint* API yang berfungsi untuk menyetujui kontrak milik pengguna berdasarkan *ID* kontrak yang diberikan. API ini digunakan di sisi *frontend* untuk mengubah status kontrak menjadi disetujui (*Approved*) dalam page *Contract Detail*.

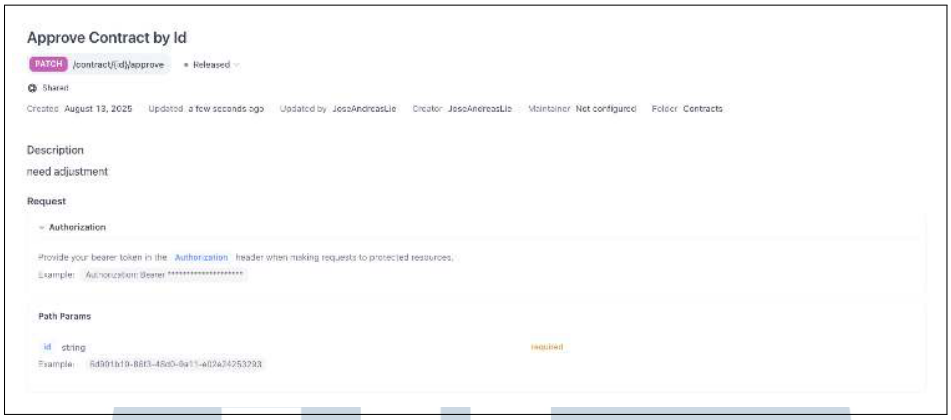
Pada gambar 3.9 merupakan *flowchart* API *Approve Contract by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission contract management* untuk menyetujui kontrak. Jika pengguna tidak ada *permission contract management*, sistem akan mengembalikan *error* 403. Setelah itu, sistem akan mencari kontrak berdasarkan *ID* yang diberikan pada *request path parameters*. Jika kontrak tidak ditemukan, sistem akan mengembalikan *error* 404. Lalu, kontrak yang ditemukan akan diperiksa status kontrak. Apabila status kontrak bukan *Pending*, sistem akan mengembalikan *error* 400. Setelah sistem memperbarui status kontrak menjadi disetujui (*Approved*) berdasarkan *ID* yang diberikan dari *path parameters*, `contract_id`. Setelah status kontrak berhasil diperbarui, sistem mengembalikan *success* 200 dan menyatakan bahwa kontrak berhasil di.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA




Gambar 3.9. *Flowchart* alur sistem API *Approve Contract by ID*

Pada gambar 3.10 merupakan contoh *request* untuk API *Approve Contract by ID*. *Request* dikirimkan ke *endpoint* `/contract/id/approve` dengan metode HTTP *PATCH* dengan menyertakan *path parameters*, `contract_id` untuk menentukan kontrak yang ingin dimodifikasi.




Gambar 3.10. Contoh *request* API *Approve Contract by ID*

Pada gambar 3.11 merupakan contoh *response* untuk API *Approve Contract by ID*. *Response* yang diberikan hanya berupa *status code* 200 dengan pesan sukses.

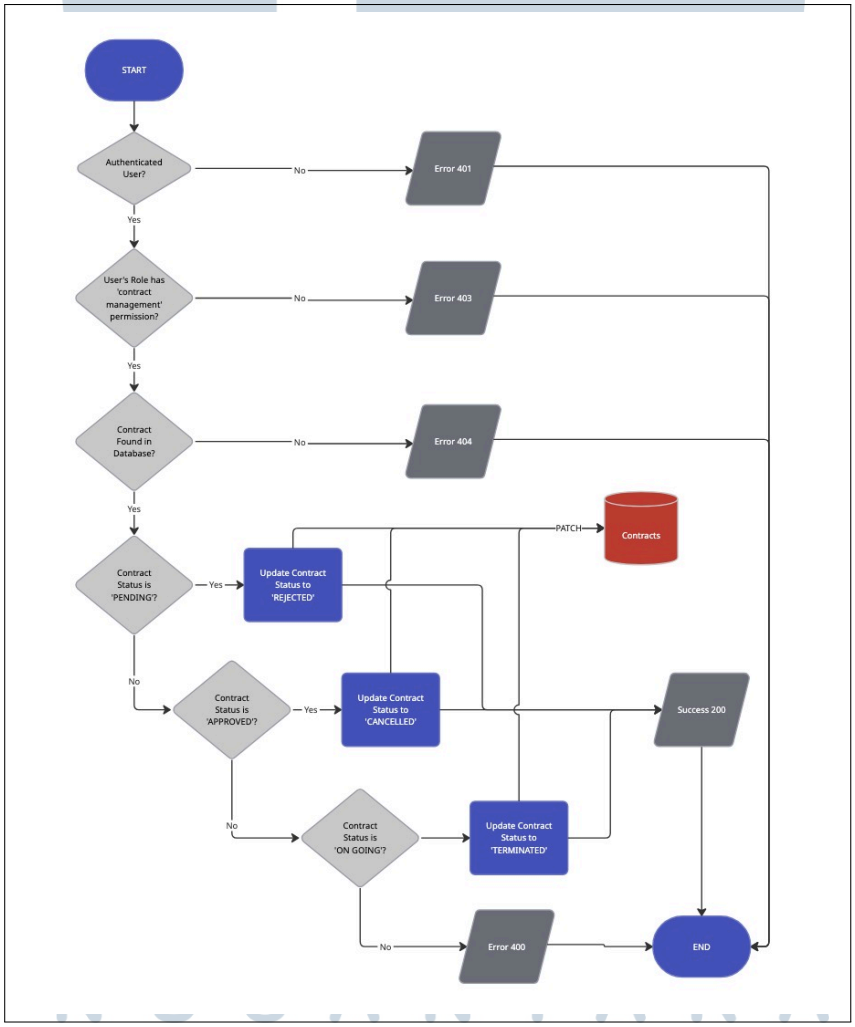

Gambar 3.11. Contoh *response* API *Approve Contract by ID*

***Terminate Contract by ID***

*Terminate Contract by ID* adalah *endpoint* API yang berfungsi untuk mengakhiri kontrak milik pengguna berdasarkan *ID* kontrak yang diberikan. API ini digunakan di sisi *frontend* untuk mengubah status kontrak menjadi dihentikan (*Terminated*) dalam page *Contract Detail*.

Pada gambar 3.12 merupakan *flowchart* API *Terminate Contract by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission contract management* untuk mengakhiri kontrak. Jika pengguna tidak ada *permission contract management*, sistem akan mengembalikan *error* 403. Setelah itu, sistem akan mencari kontrak berdasarkan *ID* yang diberikan pada *request path parameters*. Jika kontrak tidak ditemukan, sistem akan mengembalikan *error* 404. Lalu, kontrak yang ditemukan akan diperiksa status kontrak. Apabila status kontrak tidak sesuai dengan ketentuan, sistem akan mengembalikan *error* 400. Setelah itu, sistem akan memeriksa apa status

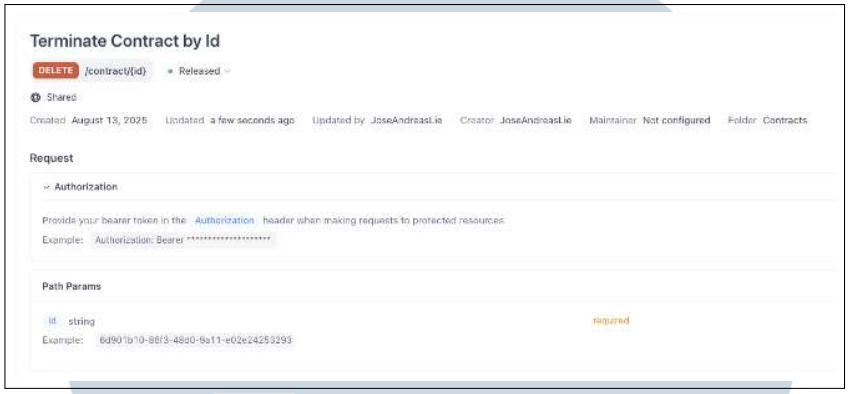
kontrak sekarang. Kalau status kontak sekarang adalah *Pending*, maka status kontrak akan diperbarui menjadi *Rejected* mengindikasikan bahwa kontrak ditolak. Jika status kontrak sekarang adalah *Approved*, status kontrak diperbarui menjadi *Cancelled* mengindikasikan bahwa kontrak sudah diakhiri sebelum dimulai. Jika status kontrak sekarang adalah *On Going*, status kontrak akan diperbarui menjadi *Terminated* mengindikasikan bahwa kontrak sempat aktif lalu diakhiri. Jika status kontrak tidak ada yang sesuai dengan yang diatas, sistem akan mengembalikan *error* 400. Setelah status kontrak berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi kontrak yang telah dihentikan.




Gambar 3.12. *Flowchart* alur sistem API *Terminate Contract by ID*

Pada gambar 3.13 dan 3.14 menunjukkan contoh *request* dan *response*

untuk API *Terminate Contract by ID*. *Request* dikirimkan ke *endpoint* `/contract/id` dengan metode HTTP *DELETE* dengan menyertakan *path parameters*, `contract_id` untuk menentukan kontrak yang ingin diakhiri.




Gambar 3.13. Contoh *request* API *Terminate Contract by ID*

Pada gambar 3.14 merupakan contoh *response* untuk API *Terminate Contract by ID*. *Response* yang diberikan berisi *status code* 200 dengan pesan sukses mengidikasikan bahwa kontrak berhasil dihentikan.

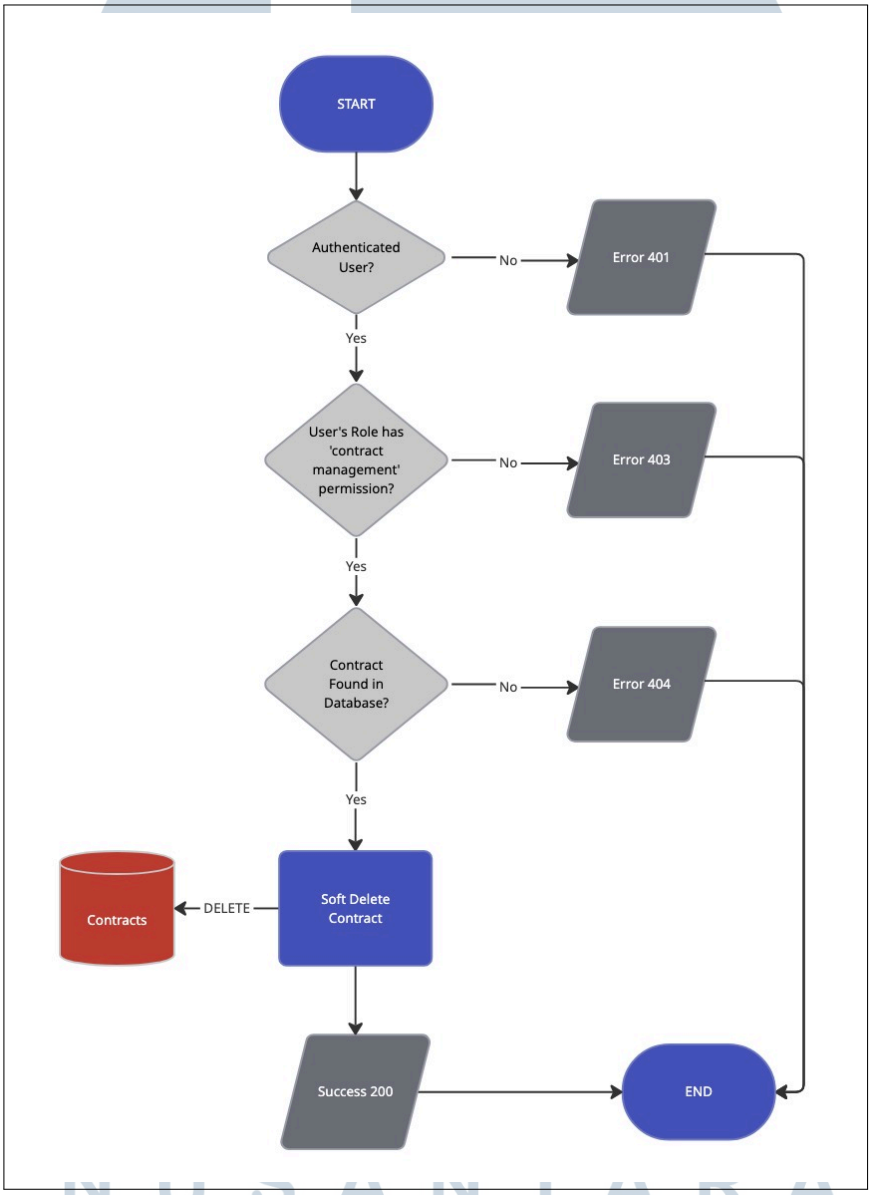

Gambar 3.14. Contoh *response* API *Terminate Contract by ID*

***Delete Contract by ID***

*Delete Contract by ID* adalah *endpoint* API yang berfungsi untuk menghapus kontrak dari sistem berdasarkan *ID* kontrak yang diberikan. API ini digunakan di sisi *frontend* untuk menghapus data kontrak dalam page *Contract Detail*.

Pada gambar 3.15 merupakan *flowchart* API *Delete Contract by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission contract management* untuk menghapus kontrak. Jika pengguna tidak ada *permission contract management*, sistem akan mengembalikan

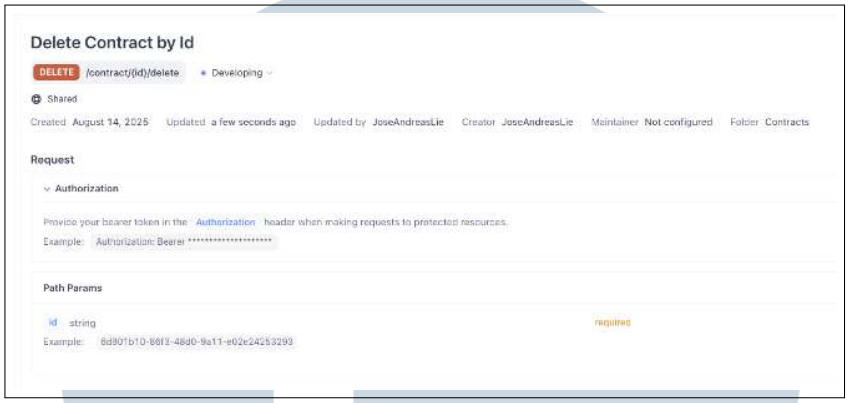
*error* 403. Setelah itu, sistem akan mencari kontrak berdasarkan *ID* yang diberikan pada *request path parameters*. Jika kontrak tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah itu, sistem menghapus data kontrak berdasarkan *ID* yang diberikan dari *path parameters*, `contract_id`. Setelah data kontrak berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi keberhasilan penghapusan kontrak.




Gambar 3.15. *Flowchart* alur sistem API *Delete Contract by ID*

Pada gambar 3.16 menunjukkan contoh *request* untuk API *Delete Contract by ID* yang dikirimkan ke *endpoint* `/contract/id/delete` menggunakan metode

HTTP *DELETE* dengan menyertakan *path parameters*, `contract_id`, untuk menentukan kontrak yang ingin dihapus.




Gambar 3.16. Contoh *request* API *Delete Contract by ID*

Pada gambar 3.17 merupakan contoh *response* untuk API *Delete Contract by ID*. *Response* yang diberikan berisi *status code* 200 dengan pesan sukses mengidikasikan bahwa kontrak berhasil dihapus.

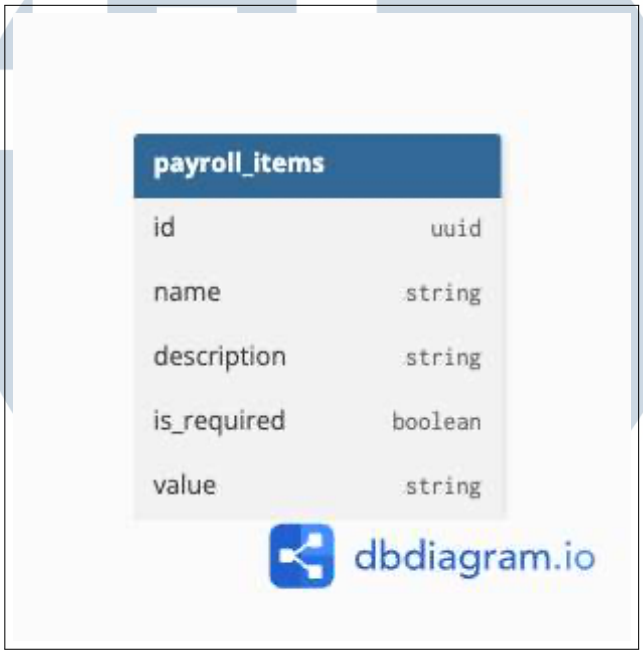

Gambar 3.17. Contoh *response* API *Delete Contract by ID*

### 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 3.18 menunjukkan struktur basis data untuk modul *Payroll Item*.




Gambar 3.18. Diagram ERD untuk modul *Payroll Item*

## B *Payroll Item* API *Endpoints*

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

UNIVERSITAS MULTIMEDIA NUSANTARA

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

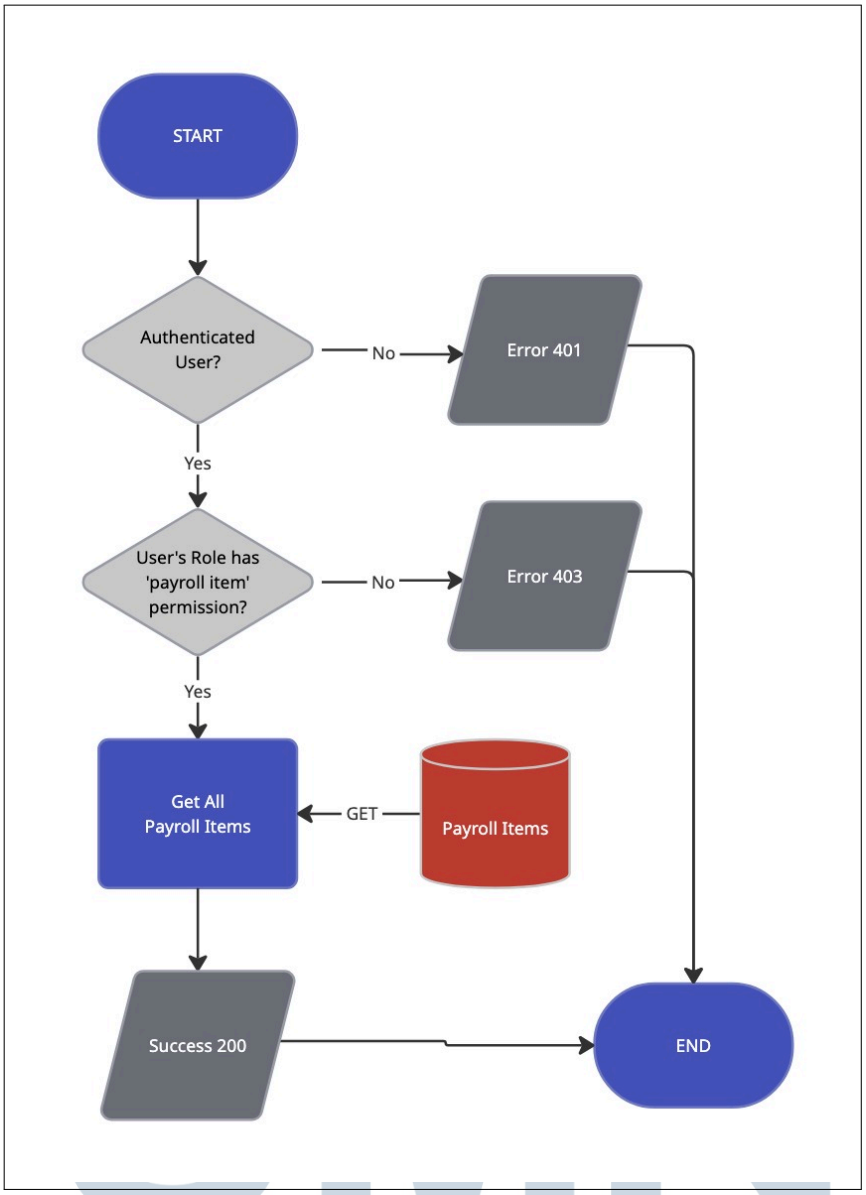
| **Nama API** | **Metode** | **Endpoint** | **Deskripsi** |
|---|---|---|---|
| Get All Payroll Item | GET | /payroll-items | Mengambil semua komponen gaji (*Payroll Item*) |
| Get Payroll Item by ID | GET | /payroll-items/:id | Mengambil detail komponen gaji berdasarkan ID |
| Create Payroll Item | 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 |

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***

*Get All Payroll Item* adalah *endpoint* API yang berfungsi untuk mengambil semua komponen gaji (*Payroll Item*) yang tersedia dalam sistem. API ini digunakan di sisi *frontend* untuk menampilkan daftar komponen gaji dalam page *Payroll Items*.

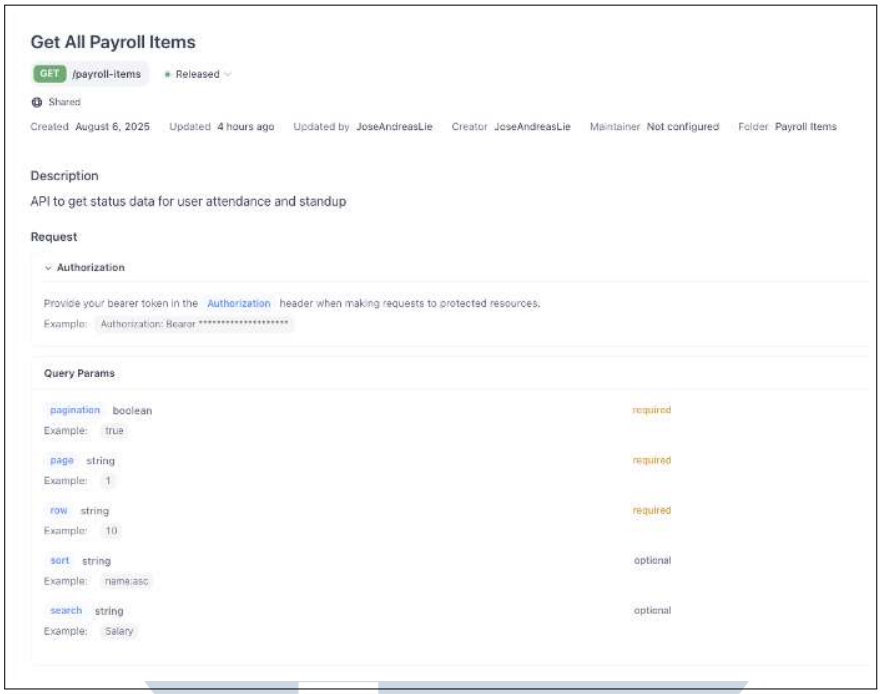
Berikut merupakan *flowchart* API *Get All Payroll Item*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payroll item* untuk mengakses data komponen gaji. Jika pengguna tidak ada *permission payroll item*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil semua data komponen gaji dari basis data. Data komponen gaji yang diambil kemudian dikemas dalam format JSON dan dikirimkan kembali sebagai respons kepada pengguna.




Gambar 3.19. *Flowchart* alur sistem API *Get All Payroll Item*

Pada gambar 3.20 menunjukkan contoh *request* yang dikirimkan ke *endpoint* `/payroll-items` menggunakan metode HTTP *GET* dengan menyertakan *query parameters* untuk *pagination* seperti *page*, *row*, *sort*, dan *search*.

UNIVERSITAS MULTIMEDIA NUSANTARA




Gambar 3.20. Contoh *request* API *Get All Payroll Item*

Pada gambar 3.21 menunjukkan contoh *response* untuk API *Get All Payroll Item* yang diterima dari *server*, berisi daftar komponen gaji beserta informasi terkait seperti *payroll item ID*, nama komponen gaji, jenis komponen gaji, besaran komponen gaji, serta status komponen gaji (aktif atau tidak aktif).

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

```
{
    "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"
            }
        ]
    }
}
```

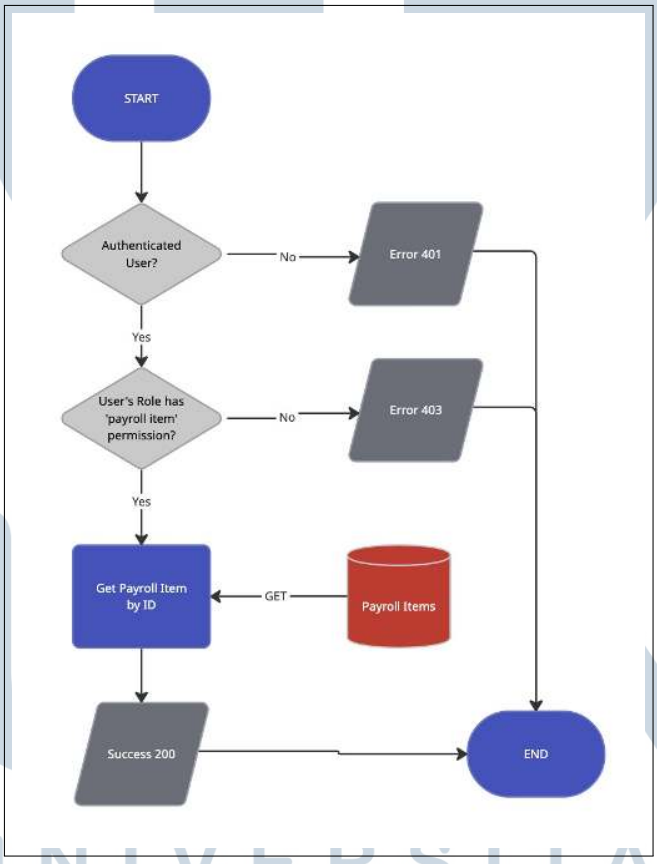

Gambar 3.21. Contoh *response* API *Get All Payroll Item*

UNIVERSITAS MULTIMEDIA NUSANTARA

***Get Payroll Item by ID***

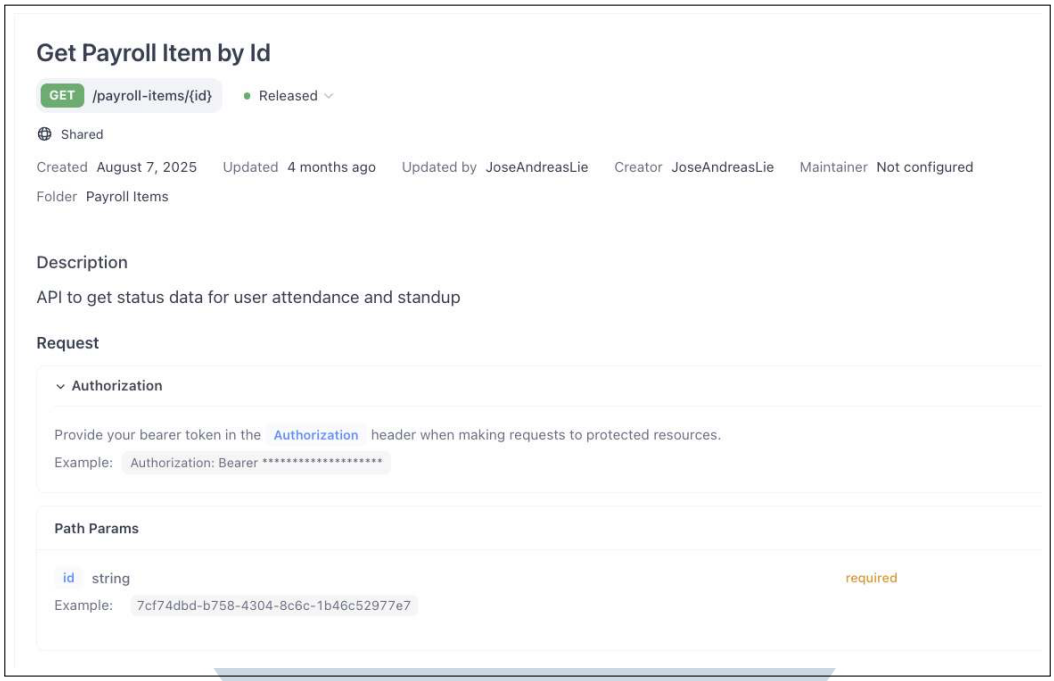
*Get Payroll Item by ID* adalah *endpoint* API yang berfungsi untuk mengambil detail komponen gaji (*Payroll Item*) berdasarkan *ID* komponen gaji yang diberikan. API ini digunakan di sisi *frontend* untuk menampilkan informasi rinci mengenai komponen gaji tertentu dalam page *Payroll Item Detail*.

Gambar 3.22 merupakan *flowchart* API *Get Payroll Item by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payroll item* untuk mengakses data komponen gaji. Jika pengguna tidak ada *permission payroll item*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data komponen gaji berdasarkan *ID* yang diberikan dari *path parameters*, `payroll_item_id`. Data komponen gaji yang diambil kemudian dikemas dalam format JSON dan dikirimkan kembali sebagai respons kepada pengguna.




Gambar 3.22. *Flowchart* alur sistem API *Get Payroll Item by ID*

Pada gambar 3.23 menunjukkan contoh *request* yang dikirimkan ke *endpoint* `/payroll-items/:id` menggunakan metode HTTP *GET* dengan menyertakan *path parameters*, `payroll_item_id`, untuk menentukan komponen gaji yang ingin diambil detailnya.




Gambar 3.23. Contoh *request* API *Get Payroll Item by ID*

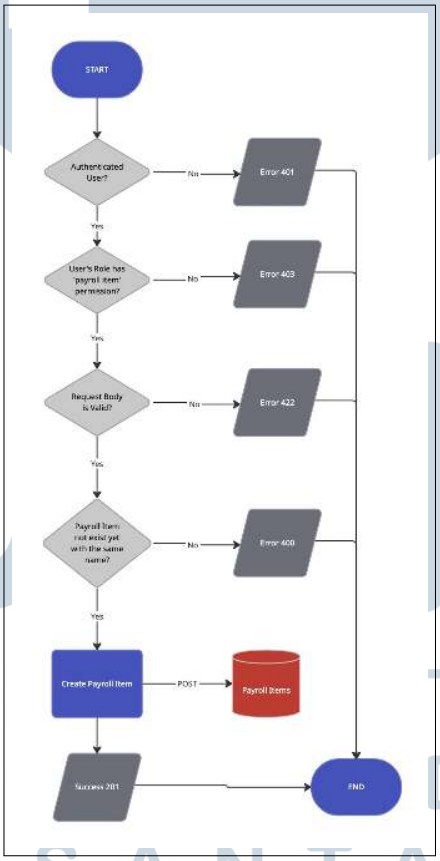
Pada gambar 3.24 menunjukkan contoh *response* yang diterima dari *server*, berisi informasi rinci mengenai komponen gaji tertentu, termasuk nama komponen gaji, jenis komponen gaji, besaran komponen gaji, serta status komponen gaji (aktif atau tidak aktif).


```
{
    "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 3.24. Contoh *response* API *Get Payroll Item by ID*

***Create Payroll Item***

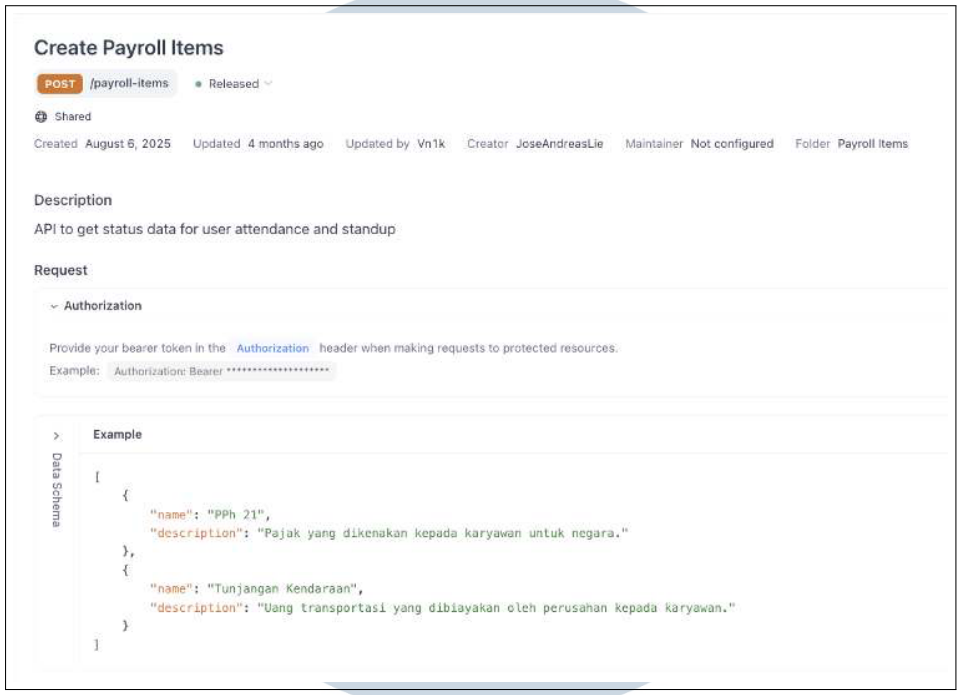
Gambar 3.25 merupakan *flowchart* API *Create Payroll Item*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payroll item* untuk membuat komponen gaji baru. Jika pengguna tidak ada *permission payroll item*, sistem akan mengembalikan *error* 403. Setelah itu, sistem memeriksa apakah *request body* valid sesuai dengan ketentuan. Jika *request body* tidak valid, sistem akan mengembalikan *error* 400. Jika valid, sistem 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.




Gambar 3.25. *Flowchart* alur sistem API *Create Payroll Item*

Pada gambar 3.26 menunjukkan contoh *request* untuk API *Create Payroll Item* yang dikirimkan ke *endpoint* `/payroll-items` menggunakan metode HTTP

*POST* dengan menyertakan *request body* yang berisi data komponen gaji baru yang akan dibuat.




Gambar 3.26. Contoh *request* API *Create Payroll Item*

Pada gambar 3.27 menunjukkan contoh *response* untuk API *Create Payroll Item* yang diterima dari *server*, berisi informasi mengenai komponen gaji yang telah dibuat, termasuk nama komponen gaji, jenis komponen gaji, besaran komponen gaji, serta status komponen gaji (aktif atau tidak aktif).

UNIVERSITAS MULTIMEDIA NUSANTARA

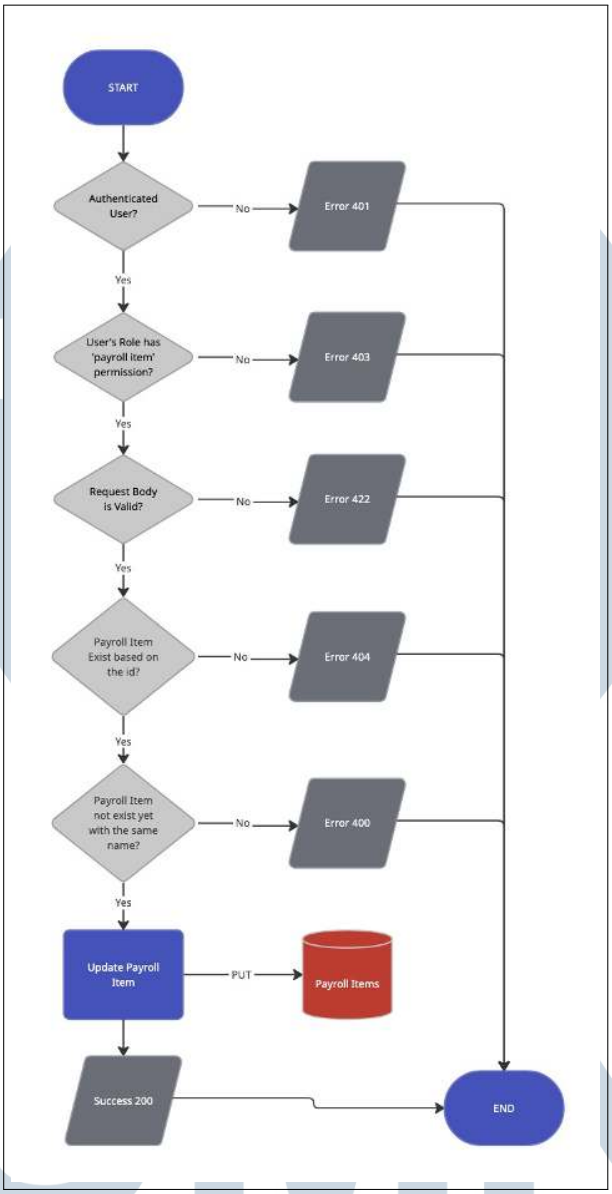
```
{
    "code": 201,
    "message": "Success",
    "data": [
        {
            "id": "5a347071-1985-47a2-bce8-adf509b732d6",
            "is_required": false,
            "name": "PPh 21",
            "description": "Pajak yang dikenakan kepada karyawan untuk negara.",
            "value": "pph 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 perusahan 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 3.27. Contoh *response* API *Create Payroll Item*

### *Update Payroll Item*

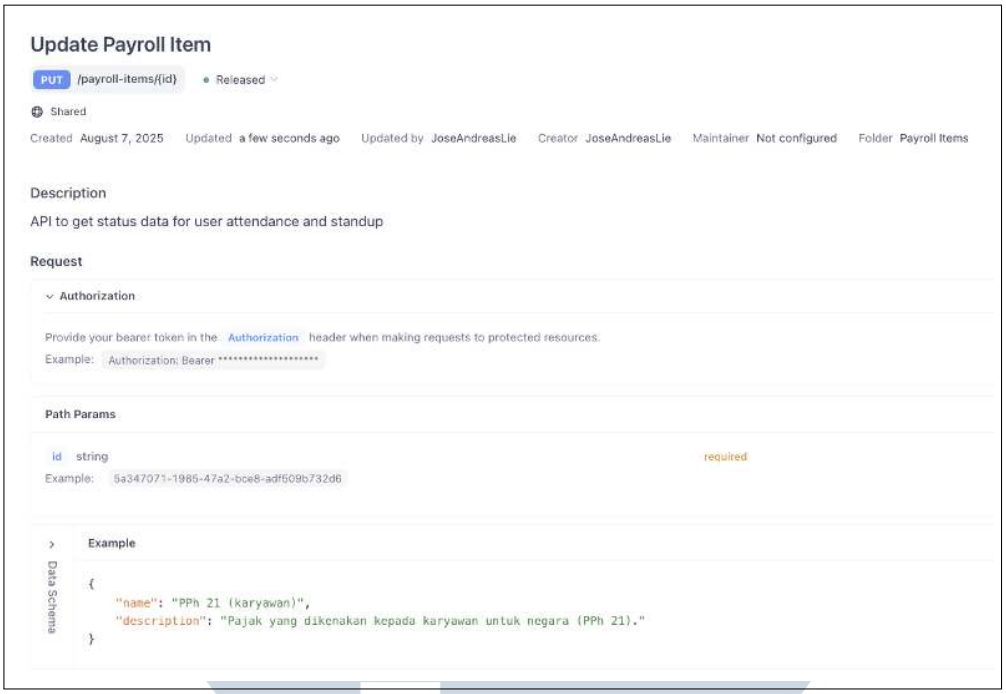
*Update Payroll Item* adalah *endpoint* API yang berfungsi untuk memperbarui komponen gaji (*Payroll Item*) berdasarkan *ID* komponen gaji yang diberikan. API ini digunakan di sisi *frontend* untuk mengubah informasi komponen gaji dalam page *Payroll Item Detail*.

Gambar 3.28 merupakan *flowchart* API *Update Payroll Item*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payroll item* untuk memperbarui komponen gaji. Jika pengguna tidak ada *permission payroll item*, sistem akan mengembalikan *error* 403. Setelah itu, sistem memeriksa apakah *request body* valid sesuai dengan ketentuan. Jika *request body* tidak valid, sistem akan mengembalikan *error* 400. Jika valid, sistem mencari data komponen gaji 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.




Gambar 3.28. *Flowchart* alur sistem API *Update Payroll Item*

Pada gambar 3.29 menunjukkan contoh *request* untuk API *Update Payroll Item* yang dikirimkan ke *endpoint* `/payroll-items/:id` menggunakan metode HTTP *PUT* dengan menyertakan *path parameters*, `payroll_item_id`, untuk menentukan komponen gaji yang ingin diperbarui serta *request body* yang berisi data komponen gaji yang akan diperbarui.




Gambar 3.29. Contoh *request* API *Update Payroll Item*

Pada gambar 3.30 menunjukkan contoh *response* untuk API *Update Payroll Item* yang diterima dari *server*, berisi informasi mengenai komponen gaji yang telah diperbarui, termasuk nama komponen gaji, jenis komponen gaji, besaran komponen gaji, serta status komponen gaji (aktif atau tidak aktif).

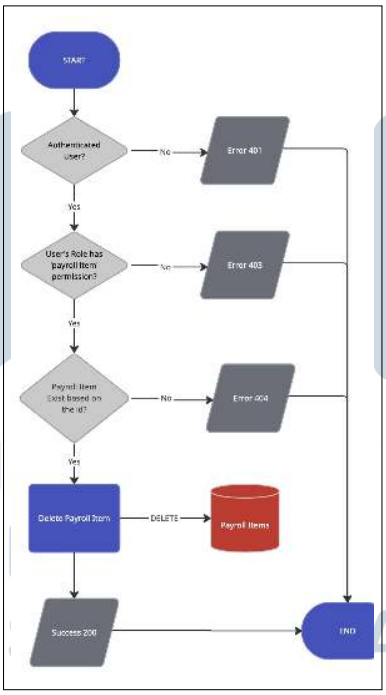

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "id": "5a347071-1985-47a2-bce8-adf509b732d6",
        "name": "PPh 21 (karyawan)",
        "is_required": false,
        "value": "pph 21 (karyawan)",
        "description": "Pajak yang dikenakan kepada karyawan untuk negara (PPh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


Gambar 3.30. Contoh *response* API *Update Payroll Item*

***Delete Payroll Item***

*Delete Payroll Item* adalah *endpoint* API yang berfungsi untuk menghapus komponen gaji (*Payroll Item*) berdasarkan *ID* komponen gaji yang diberikan. API ini digunakan di sisi *frontend* untuk menghapus data komponen gaji dalam page *Payroll Item Detail*.

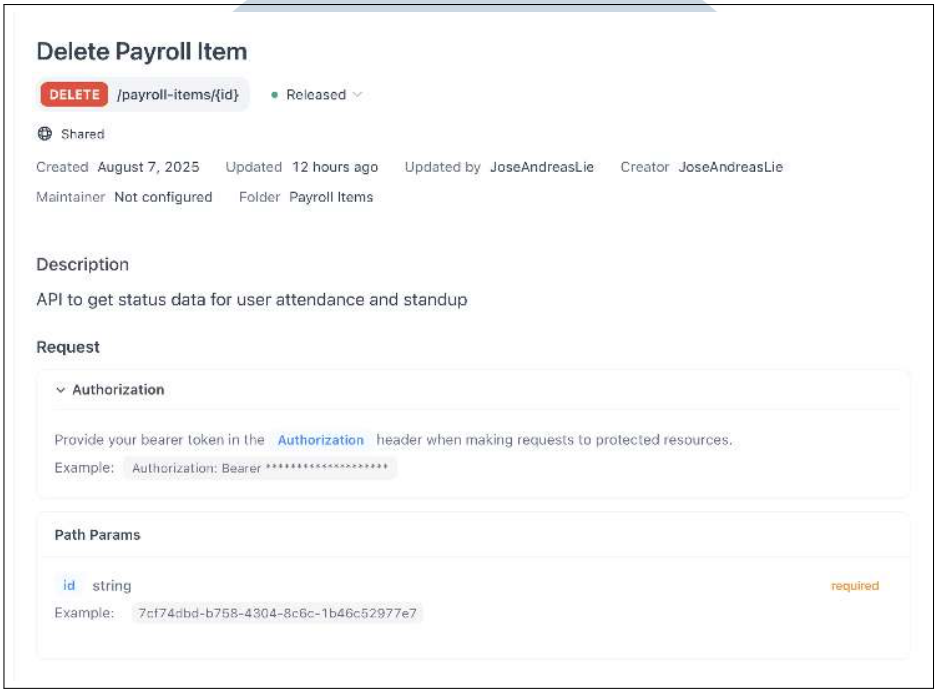
Gambar 3.31 merupakan *flowchart* API *Delete Payroll Item*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payroll item* untuk menghapus komponen gaji. Jika pengguna tidak ada *permission payroll item*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mencari data komponen gaji berdasarkan *ID* yang diberikan dari *path parameters*, `payroll_item_id`. Jika komponen gaji tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah itu, sistem menghapus data komponen gaji berdasarkan *ID* yang diberikan dari *path parameters*, `payroll_item_id`. Setelah data komponen gaji berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai keberhasilan penghapusan komponen gaji.




Gambar 3.31. *Flowchart* alur sistem API *Delete Payroll Item*

Pada gambar 3.32 menunjukkan contoh *request* untuk API *Delete Payroll*

*Item* yang dikirimkan ke *endpoint* `/payroll-items/:id` menggunakan metode HTTP *DELETE* dengan menyertakan *path parameters*, `payroll_item_id`, untuk menentukan komponen gaji yang ingin dihapus.




Gambar 3.32. Contoh *request* API *Delete Payroll Item*

Pada gambar 3.33 menunjukkan contoh *response* untuk API *Delete Payroll Item* yang diterima dari *server*, berisi *status code* 200 dengan pesan sukses untuk mengindikasikan bahwa penghapusan komponen gaji berhasil dilakukan.

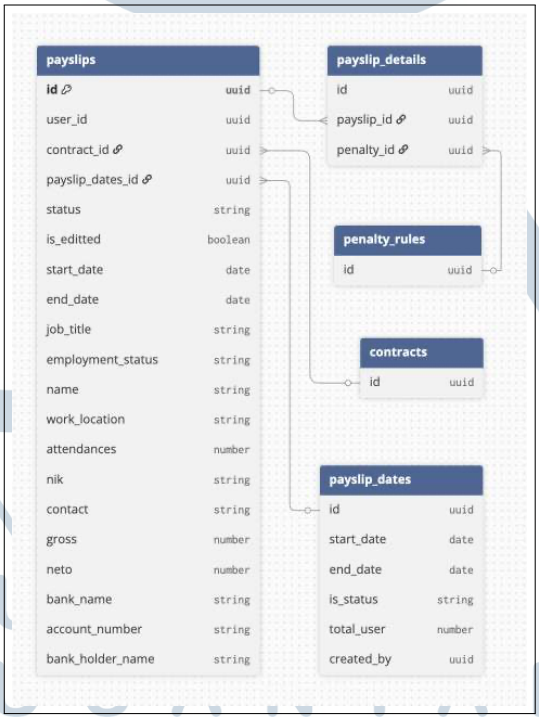

Gambar 3.33. Contoh *response* API *Delete Payroll Item*

### 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 3.34 menunjukkan struktur basis data untuk modul *Payslip*.




Gambar 3.34. Diagram ERD untuk modul *Payslip*

## B *Payslip* API *Endpoints*

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

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

| **Nama API** | **Metode** | **Endpoint** | **Deskripsi** |
|---|---|---|---|
| Get Payslip Detail | GET | /payslip/date/payslip_date_id /contract/contract_id | Mengambil detail slip gaji *user* |
| Save Payslip Detail | PATCH | /payslip/date/payslip_date_id /contract/contract_id | Mengambil detail slip gaji *user* 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 *user* berdasarkan ID |

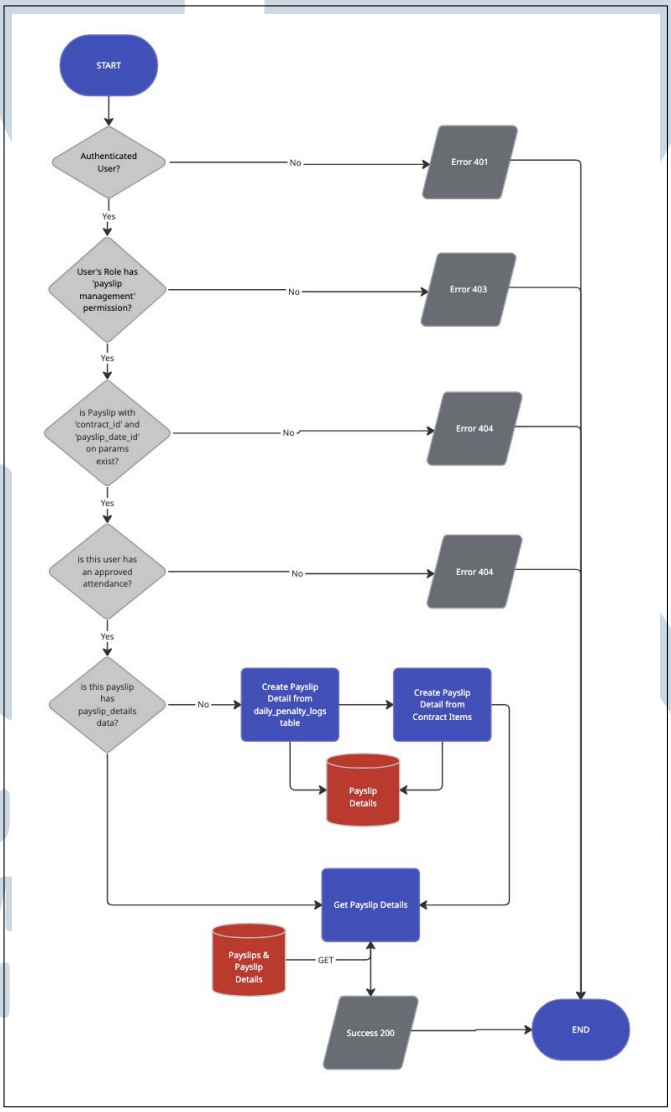
Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Payslip* seperti pada Tabel 3.4. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola slip gaji pegawai, mulai dari pengambilan data slip gaji, hingga penyimpanan detail slip gaji. Selanjutnya, akan dijelaskan alur sistem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

### *Get Payslip Detail*

*Get Payslip Detail* adalah *endpoint* API yang berfungsi untuk mengambil detail slip gaji (*Payslip Detail*) berdasarkan *payslip_date_id* dan *contract_id* yang diberikan. API ini digunakan di sisi *frontend* untuk menampilkan informasi rinci mengenai slip gaji tertentu dalam page *Payslip Detail*.

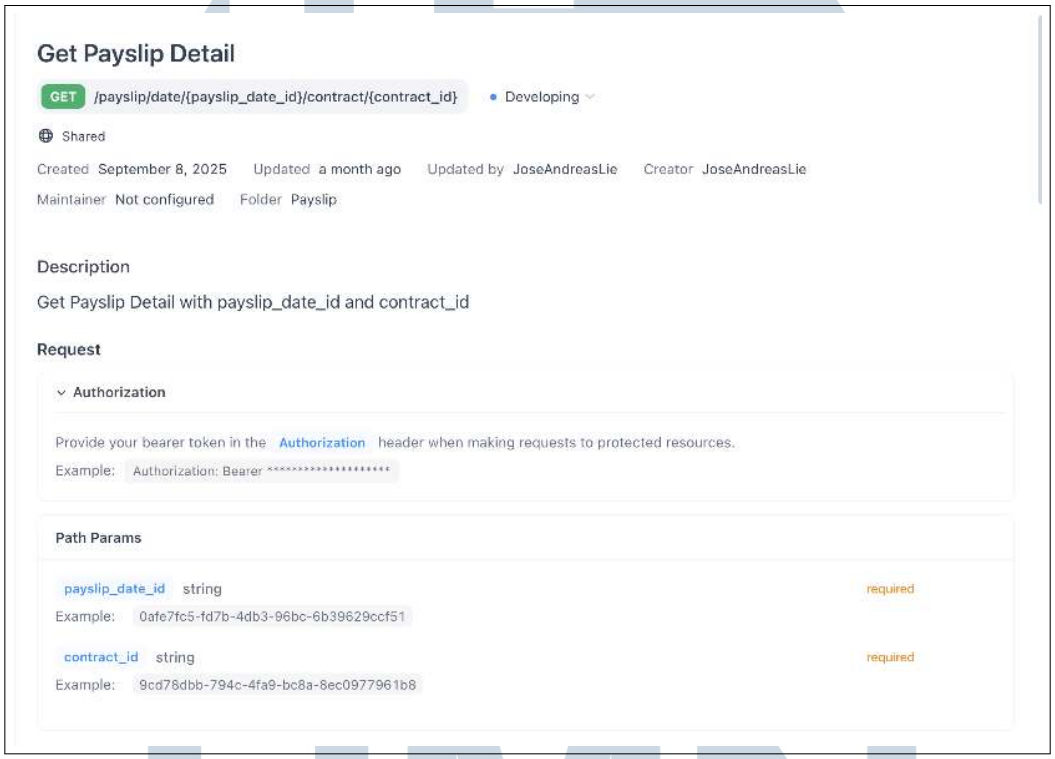
Gambar 3.35 merupakan *flowchart* API *Get Payslip Detail*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payslip management* untuk mengakses data slip gaji. Jika pengguna tidak ada *permission payslip management*, sistem akan mengembalikan *error*

403. Setelah itu, sistem mengambil data slip gaji berdasarkan *payslip_date_id* dan *contract_id* yang diberikan dari *path parameters*. Jika data slip gaji tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah itu, sistem 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.




Gambar 3.35. *Flowchart* alur sistem API *Get Payslip Detail*

Pada gambar 3.36 menunjukkan contoh *request* untuk API *Get Payslip Detail* yang dikirimkan ke *endpoint* `/payslip/date/payslip_date_id/contract/contract_id` menggunakan metode HTTP *GET* dengan menyertakan *path parameters*, `payslip_date_id` dan `contract_id`, untuk menentukan slip gaji yang ingin diambil detailnya.




Gambar 3.36. Contoh *request* API *Get Payslip Detail*

Pada gambar 3.37 menunjukkan contoh *response* untuk API *Get Payslip Detail* yang diterima dari *server*, berisi informasi rinci mengenai slip gaji tertentu, termasuk rincian komponen gaji, potongan, tunjangan, serta total gaji yang diterima.

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": "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-c6a31ddea119",
                "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-c6a31ddea119",
                "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"
        }
    }
}
```

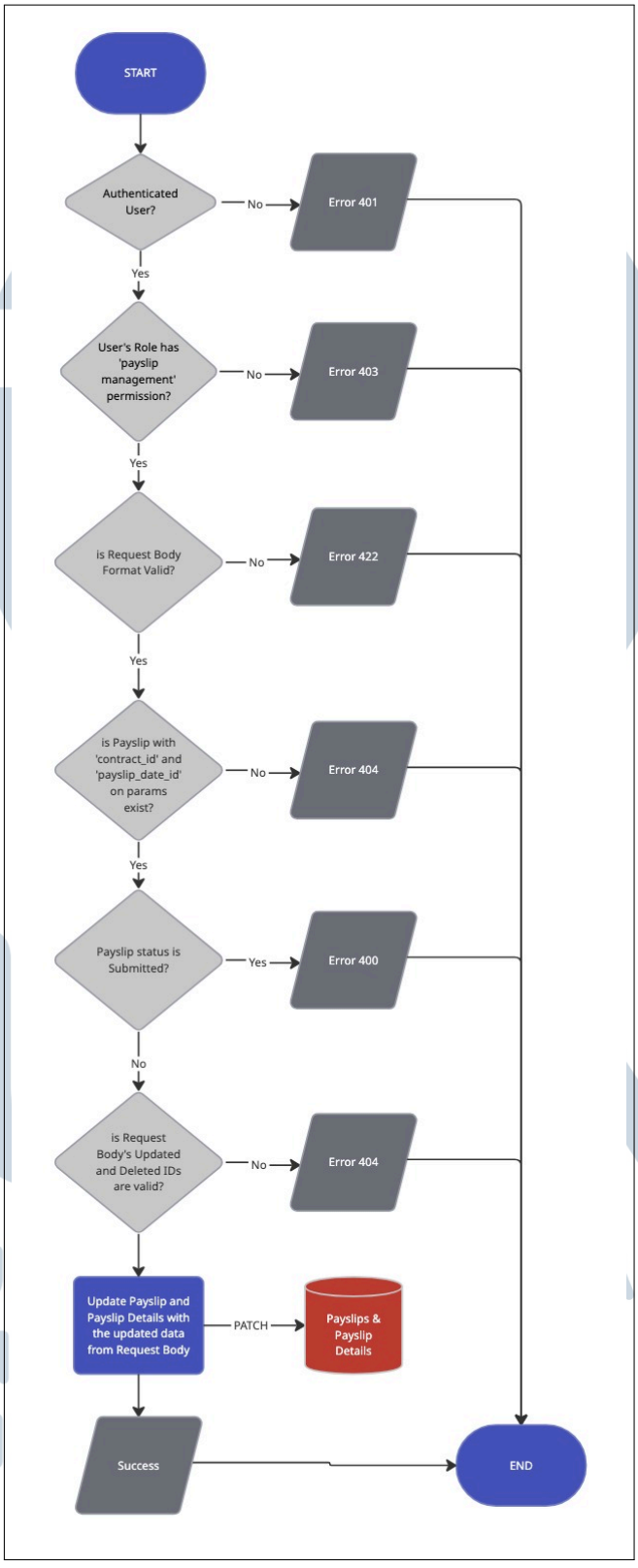

Gambar 3.37. Contoh *response* API *Get Payslip Detail*

***Save Payslip Detail***

*Save Payslip Detail* adalah *endpoint* API yang berfungsi untuk menyimpan atau memperbarui detail slip gaji (*Payslip Detail*) berdasarkan *payslip_date_id* dan *contract_id* yang diberikan. API ini digunakan di sisi *frontend* untuk menyimpan perubahan atau pembaruan pada detail slip gaji dalam page *Payslip Detail*.

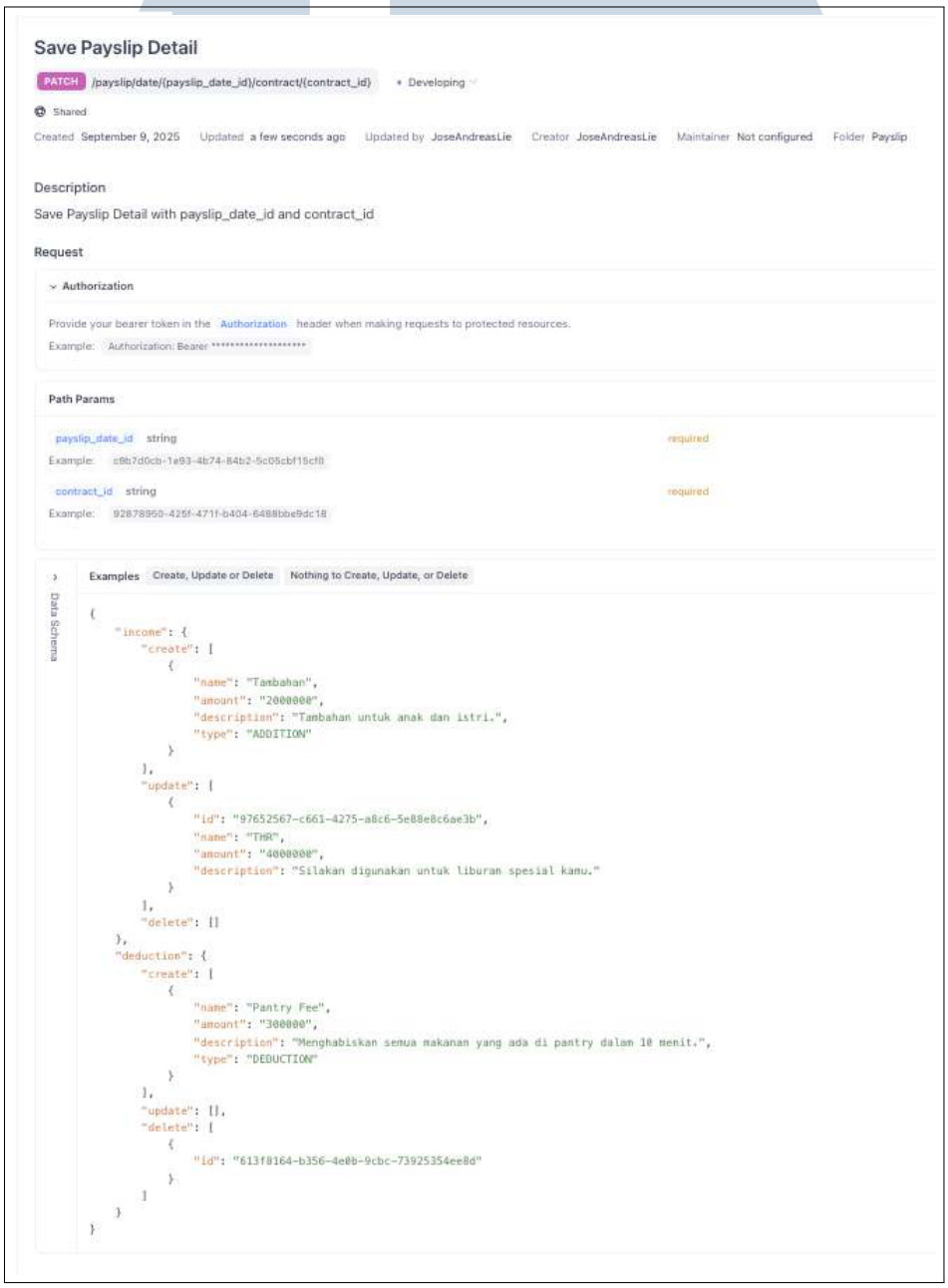
Gambar 3.38 merupakan *flowchart* API *Save Payslip Detail*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payslip* untuk memperbarui detail slip gaji. Jika pengguna tidak ada *permission payslip*, sistem akan mengembalikan *error* 403. Setelah itu, sistem memeriksa apakah *request body* valid sesuai dengan ketentuan. Jika *request body* tidak valid, sistem akan mengembalikan *error* 422. Jika valid, sistem akan memeriksa apakah *payslip* yang dicari itu ada pada sistem. Jika tidak ada, sistem akan mengembalikan *error* 404. Setelah itu, sistem akan memeriksa kembali status dari *payslip* yang dicari, apakah status *payslip* tersebut merupakan *Submitted*. Bila iya, sistem akan mengembalikan error 400. Bila tidak, sistem memperbarui data detail slip gaji berdasarkan *payslip_date_id* dan *contract_id* yang diberikan dari *path parameters* dengan data yang diberikan dalam *request body*. Setelah data detail slip gaji berhasil diperbarui, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai detail slip gaji yang telah diperbarui.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA




Gambar 3.38. *Flowchart* alur sistem API *Save Payslip Detail*

Pada gambar 3.39 menunjukkan contoh *request* untuk API *Save Payslip Detail* yang dikirimkan ke *endpoint* `/payslip/date/payslip_date_id/contract/contract_id` menggunakan metode HTTP *PATCH* dengan menyertakan *path parameters*, `payslip_date_id` dan `contract_id`, untuk menentukan slip gaji yang ingin diperbarui serta *request body* yang berisi data detail slip gaji yang akan diperbarui.




Gambar 3.39. Contoh *request* API *Save Payslip Detail*

Pada gambar 3.40 menunjukkan contoh *response* untuk API *Save Payslip Detail* yang diterima dari *server*, berisi informasi mengenai detail slip gaji yang telah diperbarui, termasuk rincian komponen gaji, potongan, tunjangan, serta total gaji yang diterima.

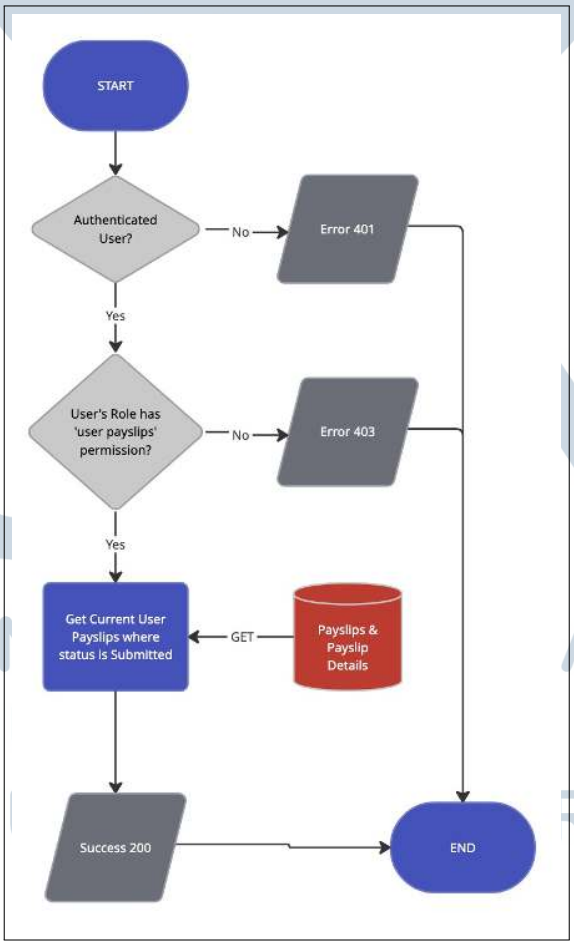
```
{
    "code": 200,
    "message": "Success",
    "data": {
        "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
        "income": [
            {
                "id": "218b5cfc-7ad4-43e9-885e-f9cd82f3de35",
                "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
                "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-4706-8928-c6a31ddea119",
                "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-896184447e2f",
                "payslip_id": "312e9e36-9140-4254-8c2b-72965b4e0586",
                "penalty_id": null,
                "type": "ADDITION",
                "name": "Tunjangan Kehormatan",
                "description": "Tunjangan yang diberikan untuk jabatan karyawan.",
                "amount": "5000000",
                "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": "3a3b02a0-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 3.40. Contoh *response* API *Save Payslip Detail*

### *Get User Payslips List*

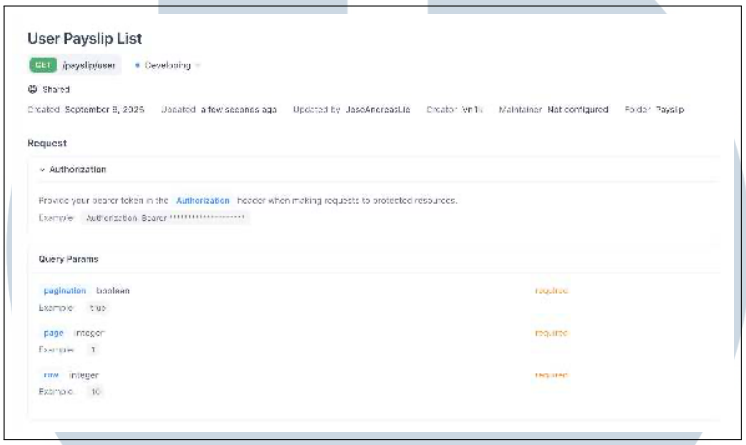
*Get User Payslips List* adalah *endpoint* API yang berfungsi untuk mengambil semua slip gaji (*Payslips*) milik pengguna yang sedang melakukan autentikasi. API ini digunakan di sisi *frontend* untuk menampilkan daftar slip gaji milik pengguna dalam page *Payslips*.

Gambar 3.41 merupakan *flowchart* API *Get User Payslips List*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payslip* untuk mengakses data slip gaji. Jika pengguna tidak ada *permission payslip*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data semua slip gaji milik pengguna yang sedang melakukan autentikasi. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar slip gaji milik pengguna.




Gambar 3.41. *Flowchart* alur sistem API *Get User Payslips List*

Pada gambar 3.42 menunjukkan contoh *request* untuk API *Get User Payslips List* yang dikirimkan ke *endpoint* `/payslips/user` menggunakan metode HTTP *GET* dengan menyertakan beberapa *query parameters* untuk kebutuhan *pagination*, seperti `page` dan `row`.




Gambar 3.42. Contoh *request* API *Get User Payslips List*

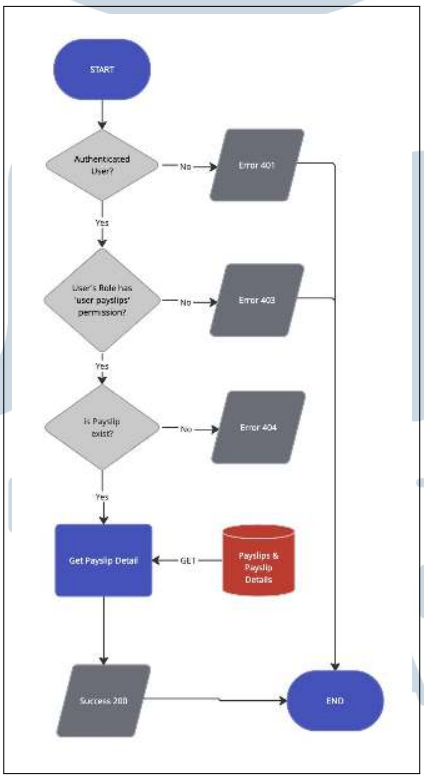
Pada gambar 3.43 menunjukkan contoh *response* untuk API *Get User Payslips List* yang diterima dari *server*, berisi daftar slip gaji milik pengguna yang sedang melakukan autentikasi, termasuk informasi seperti *payslip ID*, tanggal slip gaji, *job title*, dan *employment status*.


Gambar 3.43. Contoh *response* API *Get User Payslips List*

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

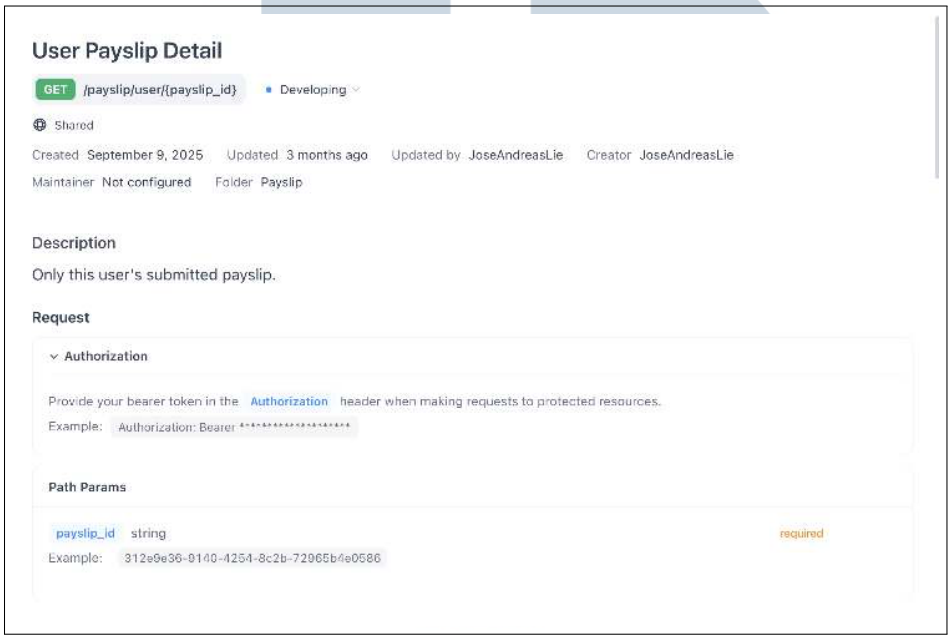
*Get User Payslip Detail by ID* adalah *endpoint* API yang berfungsi untuk mengambil detail slip gaji (*Payslip Detail*) milik pengguna berdasarkan *ID* slip gaji yang diberikan. API ini digunakan di sisi *frontend* untuk menampilkan informasi rinci mengenai slip gaji tertentu milik pengguna dalam page *Payslip Detail*.

Gambar 3.44 merupakan *flowchart* API *Get User Payslip Detail by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission payslip* untuk mengakses data slip gaji. Jika pengguna tidak ada *permission payslip*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data slip gaji milik pengguna berdasarkan *ID* slip gaji yang diberikan dari *path parameters*. Jika data slip gaji tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah itu, sistem mengambil data *payslip_details* yang berelasi dengan *payslip* tersebut. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail slip gaji yang diminta.




Gambar 3.44. *Flowchart* alur sistem API *Get User Payslip Detail by ID*

Pada gambar 3.45 menunjukkan contoh *request* untuk API *Get User Payslip Detail by ID* yang dikirimkan ke *endpoint* `/payslip/user/:id` menggunakan metode HTTP *GET* dengan menyertakan *path parameters*, `id`, untuk menentukan slip gaji yang ingin diambil detailnya.




Gambar 3.45. Contoh *request* API *Get User Payslip Detail by ID*

Pada gambar 3.46 menunjukkan contoh *response* untuk API *Get User Payslip Detail by ID* yang diterima dari *server*, berisi informasi rinci mengenai slip gaji tertentu milik pengguna, termasuk rincian komponen gaji, potongan, tunjangan, serta total gaji yang diterima.

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": "1122334455667788 99",
            "contact": "08123456789",
            "job_title": "Director",
            "employment_status": "Full Time",
            "attendance": 2
        },
        "income": [
            {
                "id": "218b5cfc-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": "5000000",
                "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": "Back Central Asia",
            "bank_account": "11223344556677",
            "bank_account_name": "HoHR 1"
        }
    }
}
```


Gambar 3.46. Contoh *response* API *Get User Payslip Detail by ID*

### 3.5.4 Modul On Boarding

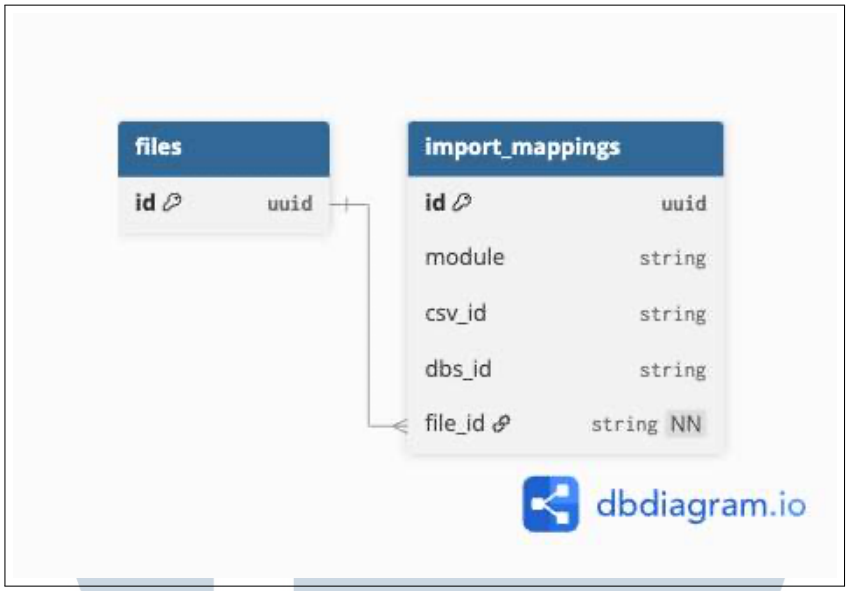
Modul On Boarding dikembangkan untuk mempermudah proses penambahan data klien baru ke dalam sistem, dikembangkan modul On Boarding yang memungkinkan pengguna mengimpor data melalui *file* CSV dengan *template* yang sudah disiapkan. Modul ini mencakup fitur pemetaan kolom data dari berkas CSV ke struktur basis data yang sesuai, serta validasi data untuk memastikan integritas dan konsistensi informasi yang diimpor. 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 menggunggah 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.

UNIVERSITAS MULTIMEDIA NUSANTARA




Gambar 3.47. Diagram ERD untuk modul *On Boarding*

Gambar 3.47 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*:

UNIVERSITAS MULTIMEDIA NUSANTARA

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

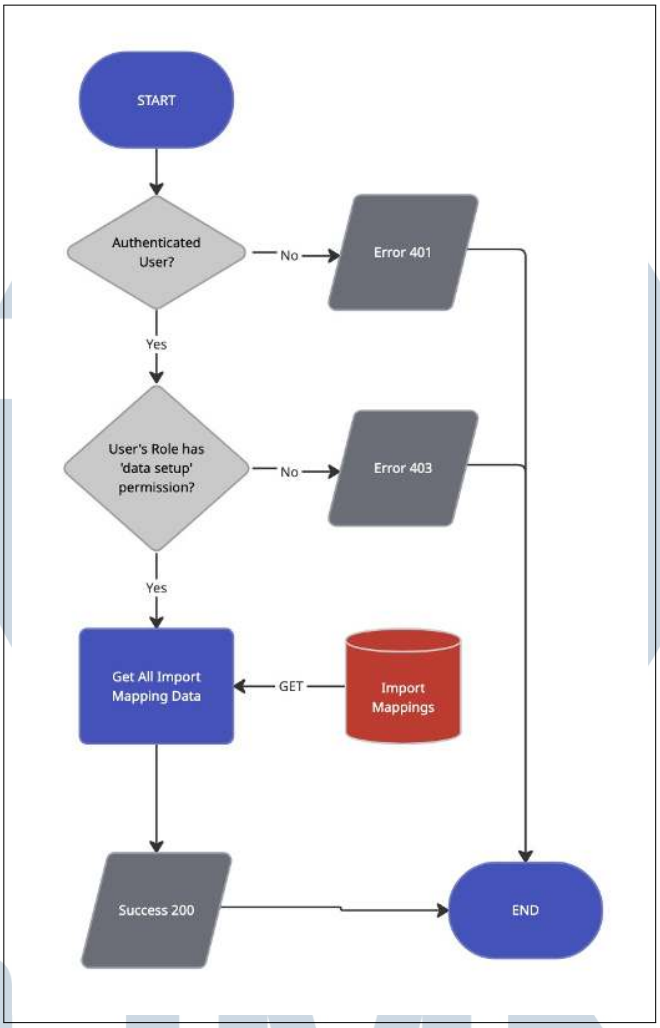
| **Nama API** | **Metode** | **Endpoint** | **Deskripsi** |
|---|---|---|---|
| Get On Boarding | GET | /on-boarding | Mengambil semua data impor |
| On Boarding Import | POST | /on-boarding/import-csv | Mengimpor data klien baru dari berkas CSV |
| Download Second Batch Template | GET | /on-boarding/download-template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

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***

Gambar 3.48 merupakan *flowchart* API *Get On Boarding*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission data setup* untuk mengakses data impor. Jika pengguna tidak ada *permission on boarding*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data semua riwayat impor data klien baru. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar riwayat impor data.

UNIVERSITAS MULTIMEDIA NUSANTARA




Gambar 3.48. *Flowchart* alur sistem API *Get On Boarding*

Pada gambar 3.49 merupakan contoh *request* untuk API *Get On Boarding* yang dikirimkan ke *endpoint* `/on-boarding` menggunakan metode HTTP *GET*.


Gambar 3.49. Contoh *request* API *Get On Boarding*

Pada gambar 3.50 merupakan contoh *response* untuk API *Get On Boarding*

yang diterima dari *server*, berisi daftar riwayat impor data klien baru, termasuk informasi seperti *ID*, nama berkas (*file name*), dan modul yang diimpor.

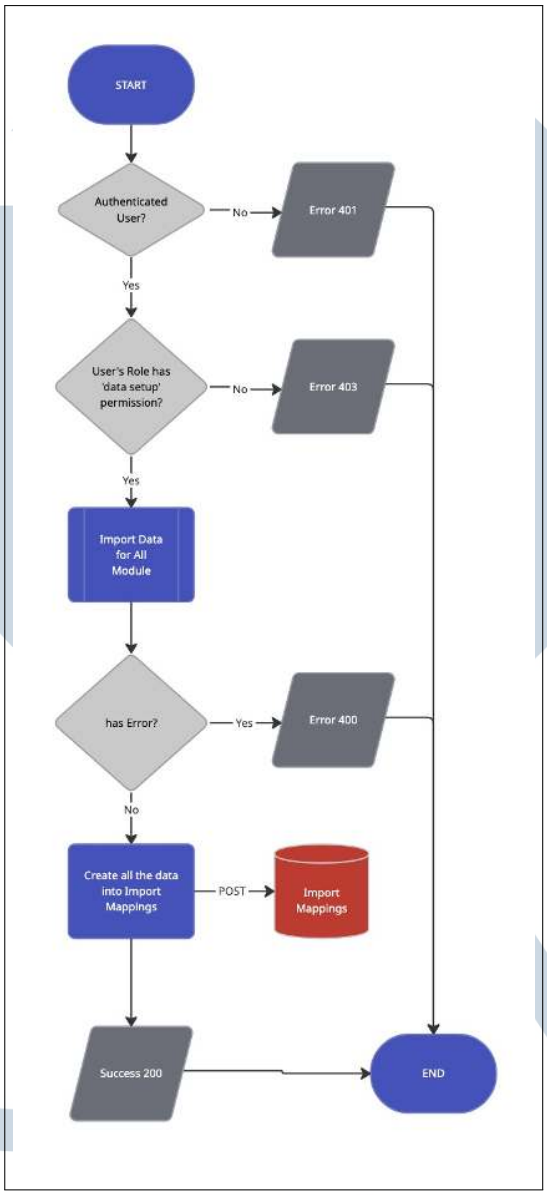
```
{
    "code": 200,
    "message": "On Boarding fetched successfully",
    "data": [
        {
            "id": "2d7287cc-599b-4e11-9867-46f7803ff699",
            "module": "leave_type",
            "file_id": "06cd0a4c-074f-40d5-9cab-b630a1486c8c",
            "name": "levtp.csv",
            "created_at": "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 3.50. Contoh *response* API *Get On Boarding*

***On Boarding Import***

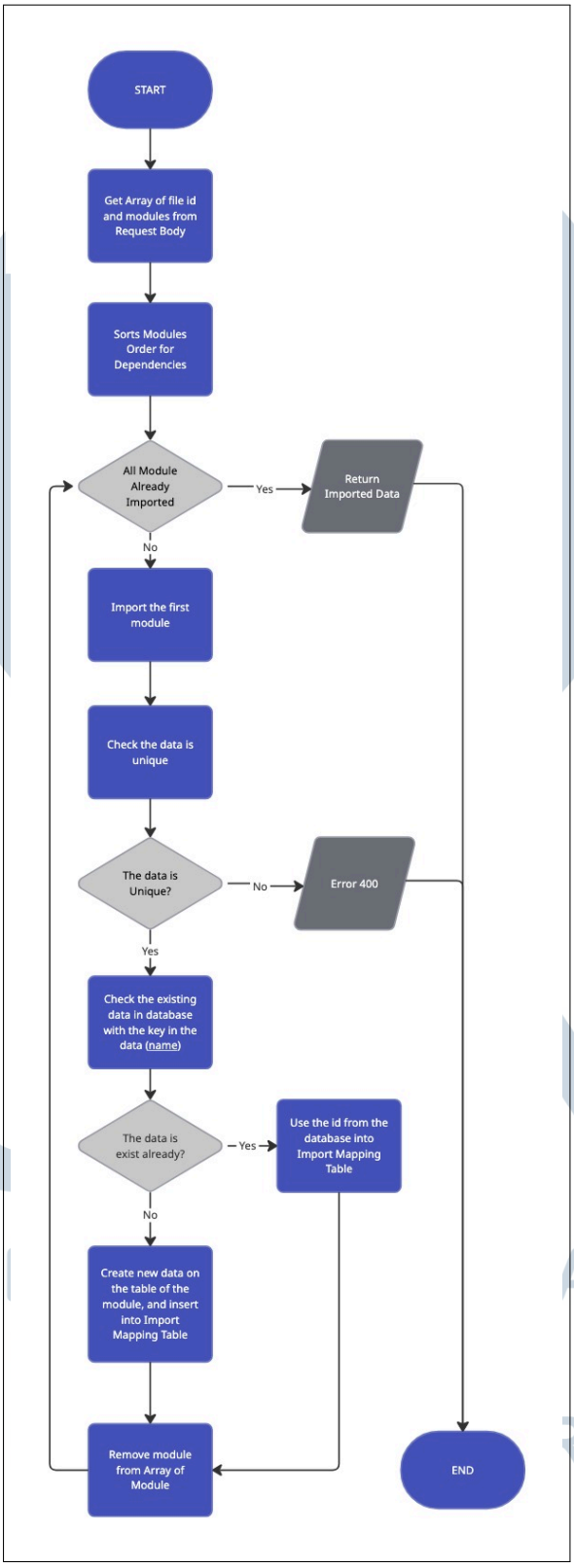
*On Boarding Import* adalah *endpoint* API yang berfungsi untuk mengimpor data klien baru dari berkas CSV yang diunggah oleh pengguna. API ini digunakan di sisi *frontend* untuk memproses impor data klien baru ke dalam sistem berdasarkan data yang terdapat pada berkas CSV. Gambar 3.51 dan Gambar 3.52 merupakan *flowchart* API *On Boarding Import*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission data setup* untuk mengimpor data klien baru. Jika pengguna tidak ada *permission on boarding*, sistem akan mengembalikan *error* 403. Setelah itu, sistem 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.



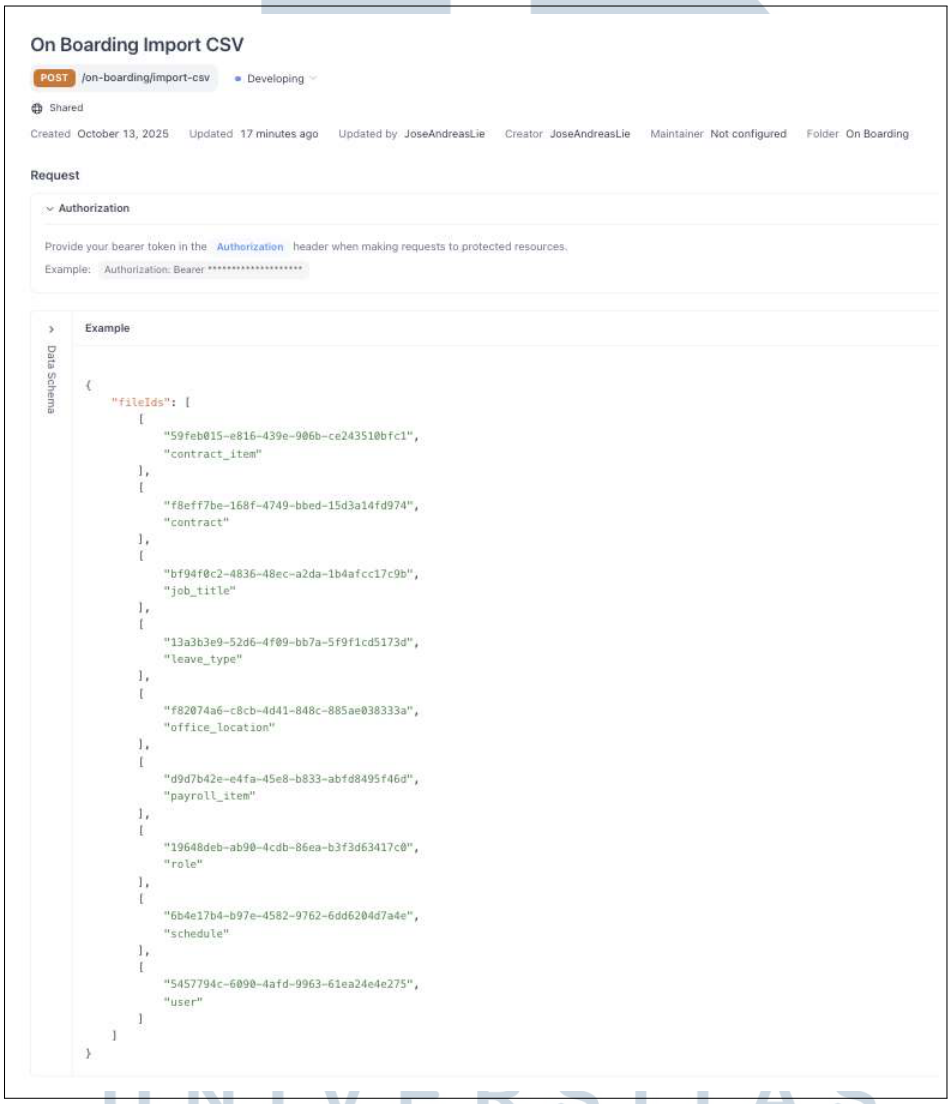

Gambar 3.51. *Flowchart* alur sistem API *On Boarding Import*

Pada gambar 3.52 merupakan subproses dari *flowchart* API *On Boarding Import* yang bernama *Import Data for All Module* yang menjelaskan proses untuk memastikan tidak ada data yang duplikat dan memproses memasukkan data dari masing-masing modul.




Gambar 3.52. *Flowchart* alur sistem API *On Boarding Import* (Lanjutan)

Pada gambar 3.53 menunjukkan contoh *request* untuk API *On Boarding Import* yang dikirimkan ke *endpoint* `/on-boarding/import-csv` menggunakan metode HTTP *POST* dengan menyertakan *request body* yang berisi data dari berkas CSV yang diunggah oleh pengguna.




Gambar 3.53. Contoh *request* API *On Boarding Import*

Pada gambar 3.54 menunjukkan contoh *response* untuk API *On Boarding Import* yang diterima dari *server*, berisi informasi mengenai hasil impor data klien baru, termasuk jumlah data yang berhasil diimpor dan jumlah data yang gagal diimpor beserta alasan kegagalannya.


Gambar 3.54. Contoh *response* API *On Boarding Import*

### 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.


Gambar 3.55. Diagram ERD untuk modul Overtime

Gambar 3.55 menunjukkan struktur basis data untuk modul Overtime.

## B *Overtime* API *Endpoints*

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

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

| **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 |

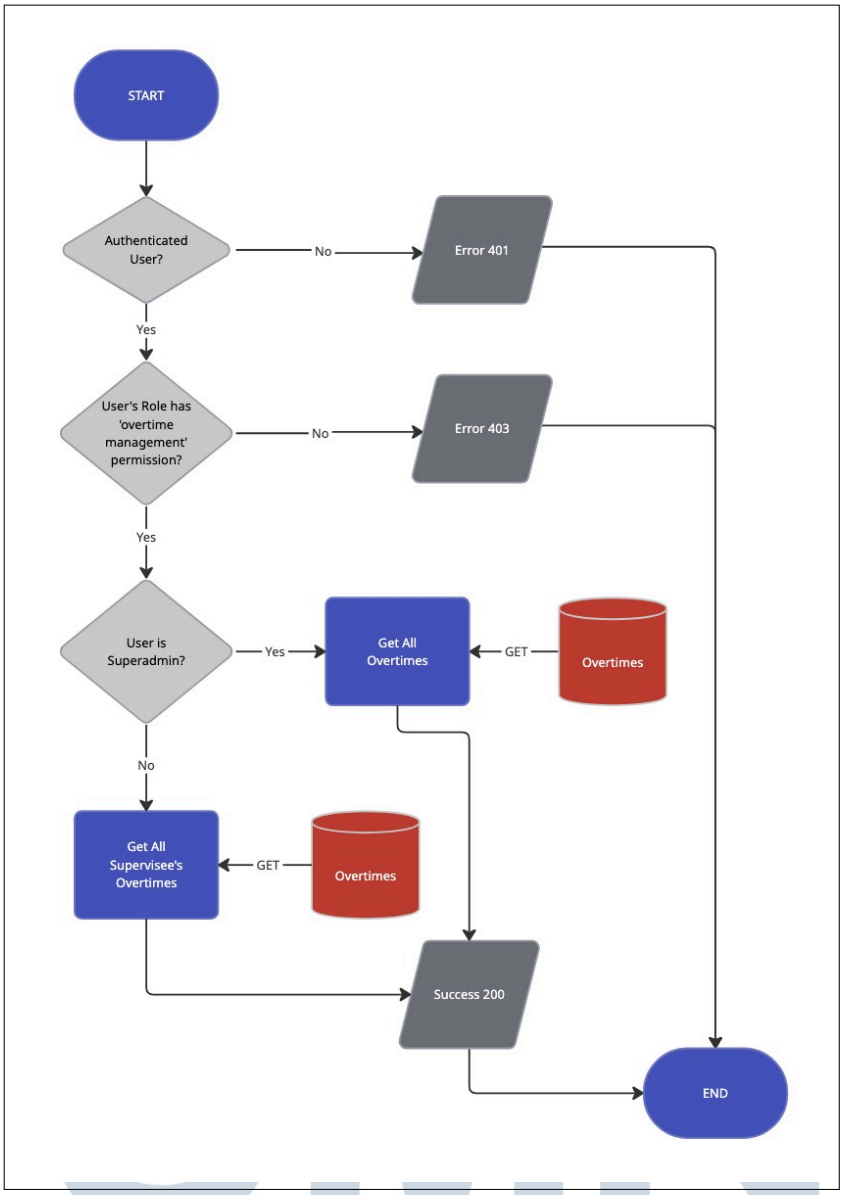
Berikut merupakan daftar *endpoints* API yang telah dikembangkan untuk modul *Overtime* seperti pada Tabel 3.6. Setiap *endpoint* memiliki fungsi spesifik dalam mengelola permintaan lembur pegawai, mulai dari pengambilan data lembur, hingga penghapusan permintaan lembur. Selanjutnya, akan dijelaskan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

***Get Overtime List***

*Get Overtime List* adalah *endpoint* API yang berfungsi untuk mengambil semua permintaan lembur (*Overtime*) yang ada di dalam sistem. API ini digunakan

di sisi *frontend* untuk menampilkan daftar permintaan lembur dalam page *Overtime*. Gambar 3.56 merupakan *flowchart* API *Get Overtime List*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission overtime* untuk mengakses data permintaan lembur. Jika pengguna tidak ada *permission overtime*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data semua permintaan lembur. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar permintaan lembur.

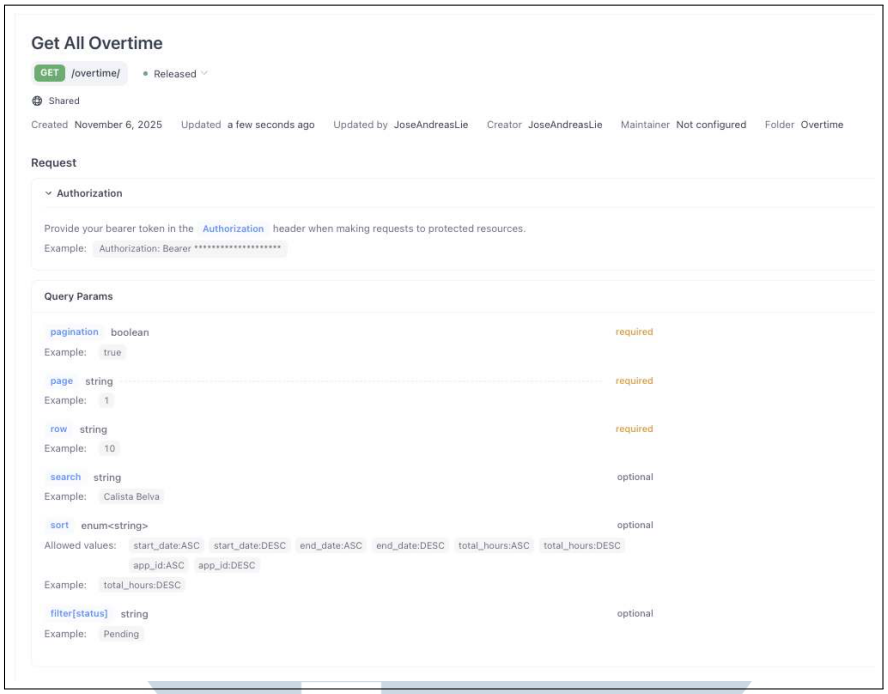
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA




Gambar 3.56. *Flowchart* alur sistem API *Get Overtime List*

Pada gambar 3.57 menunjukkan contoh *request* untuk API *Get Overtime List* yang dikirimkan ke *endpoint* `/overtime` menggunakan metode HTTP *GET* dengan menyertakan beberapa *query parameters* untuk kebutuhan *pagination*, seperti `page` dan `row`.

UNIVERSITAS MULTIMEDIA NUSANTARA




Gambar 3.57. Contoh *request* API *Get Overtime List*

Pada gambar 3.58 menunjukkan contoh *response* untuk API *Get Overtime List* yang diterima dari *server*, berisi daftar permintaan lembur, termasuk informasi seperti *overtime ID*, nama pengguna, tanggal lembur, durasi, dan status persetujuan.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

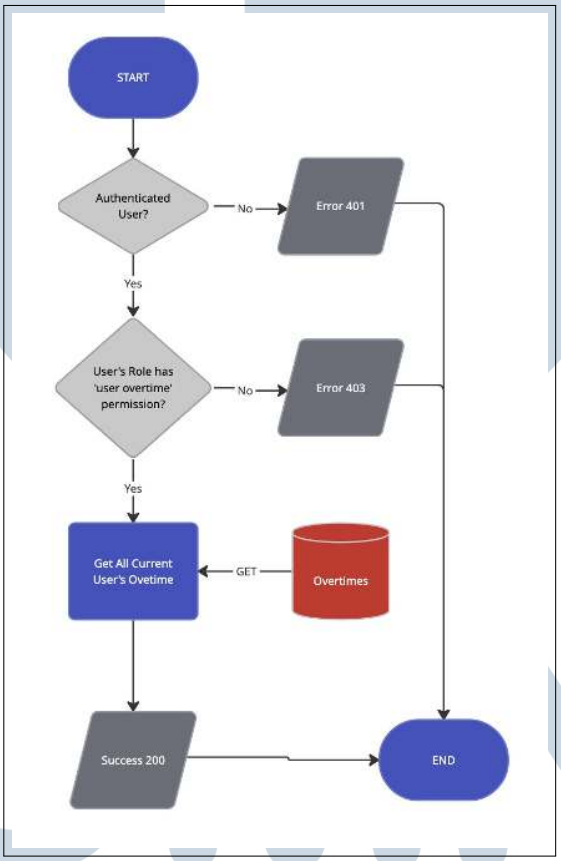
```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 2,
        "rows": [
            {
                "id": "399b26ea-b659-4261-a660-385a9f2f719b",
                "app_id": "OT-202511-010",
                "name": "Calista Belva",
                "start_date": "2025-11-06T17:00:00.000Z",
                "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": "OT-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 3.58. Contoh *response* API *Get Overtime List*

***Get User Overtime List***

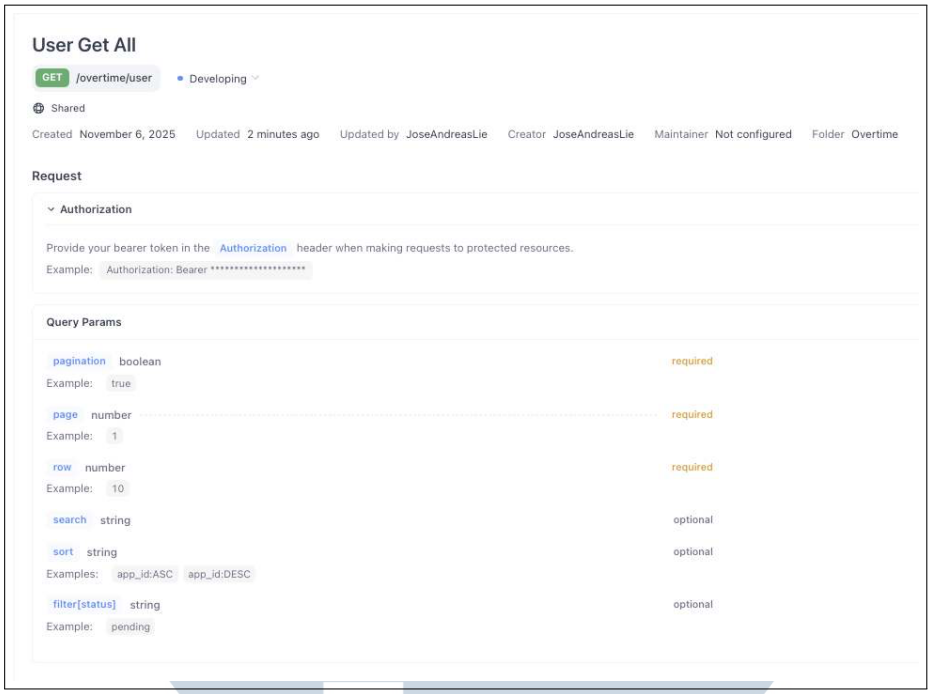
*Get User Overtime List* adalah *endpoint* API yang berfungsi untuk mengambil semua permintaan lembur (*Overtime*) milik pengguna yang sedang melakukan autentikasi. API ini digunakan di sisi *frontend* untuk menampilkan daftar permintaan lembur milik pengguna dalam page *My Overtime*. Gambar 3.59 merupakan *flowchart* API *Get User Overtime List*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan

mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission overtime* untuk mengakses data permintaan lembur. Jika pengguna tidak ada *permission overtime*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data semua permintaan lembur milik pengguna yang sedang melakukan autentikasi. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi daftar permintaan lembur milik pengguna.




Gambar 3.59. *Flowchart* alur sistem API *Get User Overtime List*

Pada gambar 3.60 menunjukkan contoh *request* untuk API *Get User Overtime List* yang dikirimkan ke *endpoint* `/overtime/user` menggunakan metode HTTP *GET* dengan menyertakan beberapa *query parameters* untuk kebutuhan *pagination*, seperti `page` dan `row`.




Gambar 3.60. Contoh *request* API *Get User Overtime List*

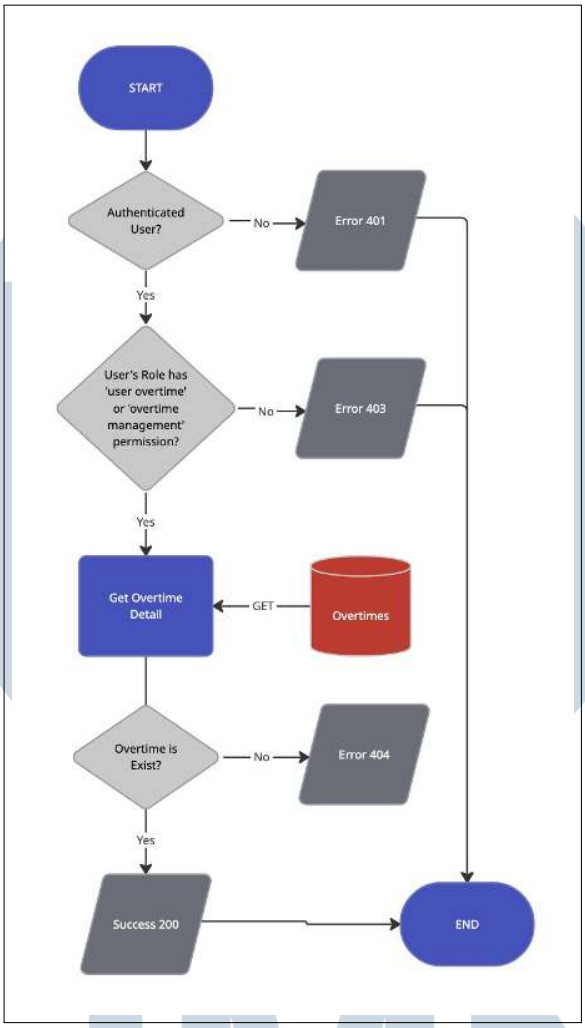
Pada gambar 3.61 menunjukkan contoh *response* untuk API *Get User Overtime List* yang diterima dari *server*, berisi daftar permintaan lembur milik pengguna yang sedang melakukan autentikasi, termasuk informasi seperti *overtime ID*, tanggal lembur, durasi, dan status persetujuan.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA


Gambar 3.61. Contoh *response* API *Get User Overtime List*

***Get Overtime Detail by ID***

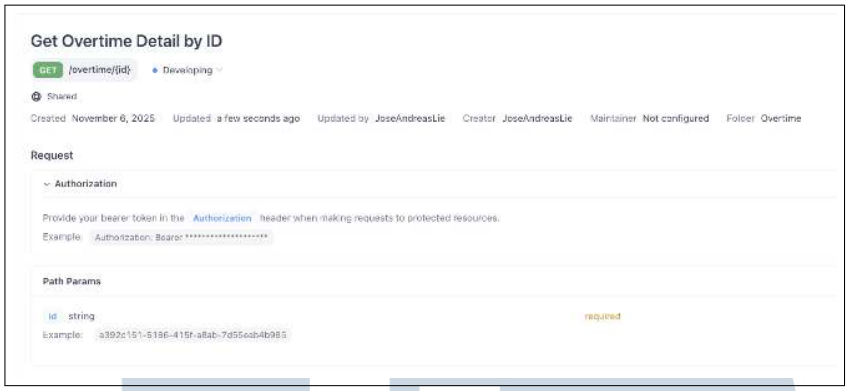
*Get Overtime Detail by ID* adalah *endpoint* API yang berfungsi untuk mengambil detail permintaan lembur (*Overtime*) berdasarkan *ID* lembur yang diberikan. API ini digunakan di sisi *frontend* untuk menampilkan informasi rinci mengenai permintaan lembur tertentu dalam page *Overtime Detail*. Gambar 3.62 merupakan *flowchart* API *Get Overtime Detail by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission user overtime* atau *permission overtime management* untuk mengakses data permintaan lembur. Jika pengguna tidak ada *permission user overtime* atau *permission overtime management*, sistem akan mengembalikan *error* 403. Setelah itu, sistem mengambil data detail permintaan lembur berdasarkan *ID* lembur yang diberikan dari *path parameters*. Jika data permintaan lembur tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah data berhasil diambil, sistem mengembalikan respons (*response*) kepada pengguna yang berisi detail permintaan lembur yang diminta.




Gambar 3.62. *Flowchart* alur sistem API *Get Overtime Detail by ID*

Pada gambar 3.63 menunjukkan contoh *request* untuk API *Get Overtime Detail by ID* yang dikirimkan ke *endpoint* `/overtime/:id` menggunakan metode HTTP *GET* dengan menyertakan *path parameters*, `id`, untuk menentukan permintaan lembur yang ingin diambil detailnya.

UNIVERSITAS MULTIMEDIA NUSANTARA




Gambar 3.63. Contoh *request* API *Get Overtime Detail by ID*

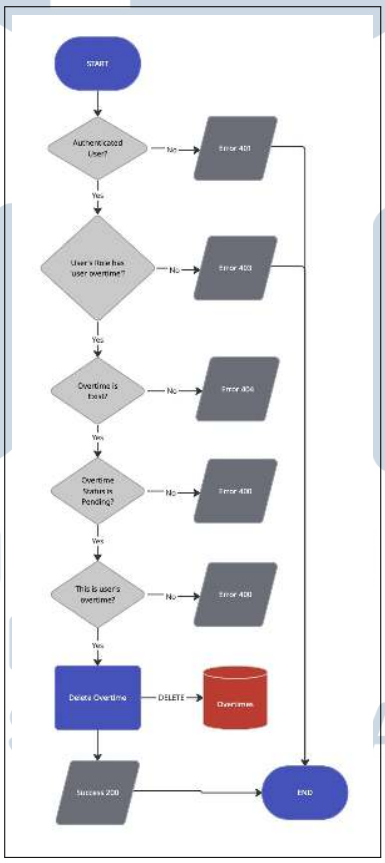
Pada gambar 3.64 menunjukkan contoh *response* untuk API *Get Overtime Detail by ID* yang diterima dari *server*, berisi informasi rinci mengenai permintaan lembur tertentu, termasuk rincian seperti nama pengguna, tanggal lembur, jam lembur, durasi, alasan lembur, status persetujuan, deskripsi lembur, dan lain-lain.


```
{
    "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 3.64. Contoh *response* API *Get Overtime Detail by ID*

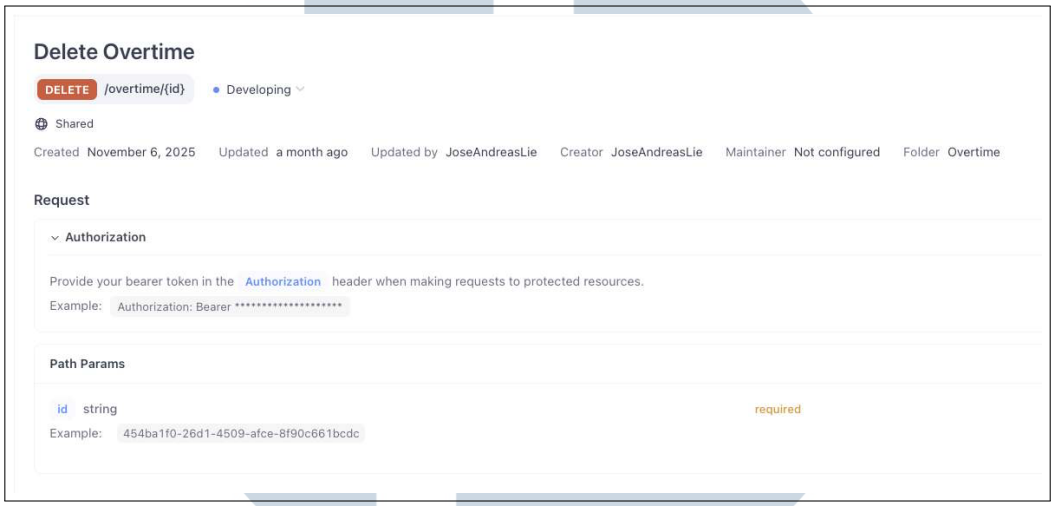
***Delete Overtime by ID***

*Delete Overtime by ID* adalah *endpoint* API yang berfungsi untuk menghapus permintaan lembur (*Overtime*) berdasarkan *ID* lembur yang diberikan. API ini digunakan di sisi *frontend* untuk menghapus permintaan lembur tertentu dalam page *Overtime Detail*. Gambar 3.65 merupakan *flowchart* API *Delete Overtime by ID*. *Flowchart* dimulai dari pengecekan autentikasi pengguna. Jika autentikasi gagal, sistem akan mengembalikan *error* 401. Jika autentikasi berhasil, sistem akan memeriksa *permission overtime management* untuk menghapus data permintaan lembur. Jika pengguna tidak ada *permission overtime management*, sistem akan mengembalikan *error* 403. Setelah itu, sistem menghapus data permintaan lembur berdasarkan *ID* lembur yang diberikan dari *path parameters*. Jika data permintaan lembur tidak ditemukan, sistem akan mengembalikan *error* 404. Setelah data permintaan lembur berhasil dihapus, sistem mengembalikan respons (*response*) kepada pengguna yang berisi informasi mengenai keberhasilan penghapusan permintaan lembur.




Gambar 3.65. *Flowchart* alur sistem API *Delete Overtime by ID*

Pada gambar 3.66 menunjukkan contoh *request* untuk API *Delete Overtime by ID* yang dikirimkan ke *endpoint* `/overtime/:id` menggunakan metode HTTP *DELETE* dengan menyertakan *path parameters*, `id`, untuk menentukan permintaan lembur yang ingin dihapus.




Gambar 3.66. Contoh *request* API *Delete Overtime by ID*

Pada gambar 3.67 menunjukkan contoh *response* untuk API *Delete Overtime by ID* yang diterima dari *server*, berisi *status code* 200 dengan pesan sukses mengindikasikan bahwa permintaan lembur telah berhasil dihapus.


Gambar 3.67. Contoh *response* API *Delete Overtime by ID*

UNIVERSITAS MULTIMEDIA NUSANTARA

## 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, salah satu contohnya adalah pada *Payslip Detail*, *penalty* yang dikenakan itu tidak sesuai dengan jangka waktu pada *Payslip* tersebut. 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