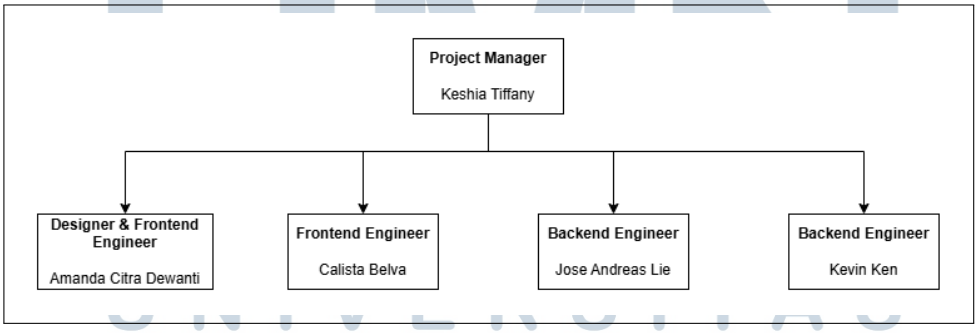
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kerja magang yang dilakukan di PT Ganda Visi Jayatama pada tanggal 13 Januari 2025 hingga 13 Juli 2025 mempunyai struktur kerja yang dapat dilihat pada Gambar 3.1. Selama proses kerja magang di PT Ganda Visi Jayatama, kedudukan yang ditempati adalah *frontend engineer intern*. Praktek kerja magang dipimpin dan dikoordinasikan oleh Keshia Tiffany selaku project manager. Project manager bertugas untuk mengawasi kegiatan, membimbing, dan mengarahkan proses kerja magang. Dalam pelaksanaan kerja magang, tugas yang diberikan yaitu mengembangkan HRIS perusahaan untuk membantu pengelolaan sumber daya manusia. Dalam pengerjaan HRIS perusahaan, tim terdiri dari satu project manager, satu orang designer, dan empat orang *software engineer* dengan dua subdivisi yang berbeda. Subdivisi tersebut terdiri dari *frontend* dan *backend engineer* dengan pembagian tugas yang berbeda. Selama proses pengerjaan, komunikasi antar tim dilakukan secara langsung. Setiap hari jumat dilakukan weekly meeting untuk membahas progres dari pengerjaan HRIS atau tugas yang akan dikerjakan selanjutnya, serta menerima feedback terkait progres yang telah dikerjakan oleh tim, sehingga dapat diimplementasikan untuk sprint berikutnya.



Gambar 3.1. Struktur Organisasi Pelaksanaan Kerja Magang Project HRIS
Sumber: [9]

Praktek kerja magang menggunakan Git sebagai sarana pengontrol versi project. Setiap anggota dalam project HRIS memiliki branch masing-masing. Selain itu, perusahaan menggunakan Jira sebagai pengelola alur kerja project. Jira menyediakan fitur active sprints yang terdiri dari to do, in progress, in review, dan

done, sehingga memudahkan pengembang dalam mengatur tugas-tugas yang sudah selesai ataupun yang belum selesai.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama kerja magang adalah mengembangkan modul *reimbursement* dan *payroll* HRIS (*Human Resource Internal System*) perusahaan berbasis web dengan bahasa pemrograman Typescript dan SCSS dan melakukan manual testing terhadap HRIS. Pengujian secara manual dilakukan dengan mengecek fungsionalitas pada halaman yang dikerjakan serta pengalaman *user* untuk memastikan bahwa aplikasi berfungsi sesuai dengan yang diharapkan. Pengujian ini dilakukan dengan menentukan berbagai kasus yang bertujuan untuk menemukan bug atau masalah dalam pengembangan.

3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan magang, terdapat tugas-tugas yang dikerjakan dan dapat dilihat pada Tabel 3.27. Pada minggu pertama kerja magang, dilakukan sesi *onboarding*. Sesi *onboarding* bertujuan untuk memperkenalkan kepada karyawan baru dalam suatu organisasi mengenai struktur organisasi perusahaan serta alur kerja dalam perusahaan. Selanjutnya, *supervisor* memberikan penugasan untuk mempelajari *boilerplate frontend* perusahan. *Boilerplate* adalah kerangka dasar yang berisi kode pemrograman yang sering digunakan berulang-ulang dalam perusahaan yang bertujuan untuk memudahkan dan mempercepat proses pengembangan. Lalu, dilakukan pembaharuan validasi pada formulir karyawan dan *project product* pada minggu kedua. Sebelumnya validasi hanya bersifat wajib diisi dan diperbaharui sesuai dengan aturan dan ketentuan yang berlaku pada masing-masing *form field*.

Selanjutnya pada minggu ketiga hingga minggu keempat, dilakukan pengembangan antarmuka pada halaman *Reimbursement Management*, *Permit*, dan *Input*. Kemudian, pada minggu kelima hingga keenam, dilaukan integrasi API untuk *pagination* pada halaman *User Management*, *Role Management*, *Project Product Management*, *Stand Up Feed*, *Activity Log*, dan *People Report*. Sebelumnya proses *pagination* dilakukan sepenuhnya di *frontend* dan telah diperbaharui dengan melakukan integrasi API. Pada minggu ketujuh, dilakukan integrasi API pada halaman *Reimbursement Management*, *Permit*, dan *Input*.

Dilanjutkan pada minggu kedelepan, dilakukan *end to end testing* atau pengujian secara lengkap dari awal hingga akhir untuk memastikan halaman *Reimbursement Management*, *Permit*, dan *Input* berjalan sesuai dengan yang diharapkan.

Pengembangan antarmuka pada halaman *Configuration Payroll List*, *Configuration Payroll Input*, *Salary Slip List*, *Salary Slip*, dan *Payroll* pada halaman *User Management* dilakukan pada minggu kesembilan dan kesepuluh. Kemudian pada minggu kesebelas hingga ketigabelas, dilakukan integrasi API pada halaman *Configuration Payroll List*, *Configuration Payroll Input*, dan *Payroll* pada halaman *User Management*. Pada minggu keempatbelas, dilakukan *end to end testing* atau pengujian secara lengkap dari awal hingga akhir untuk memastkan halaman *Configuration Payroll List*, *Configuration Payroll Input*, dan *Payroll* pada halaman *User Management* berjalan sesuai dengan yang diharapkan. Integrasi API pada halaman *Salary Slip List* dan *Salary Slip* dilakukan pada minggu kelimabelas hingga keenambelas. Pada minggu ketujuhbelas, dilakukan *end to end testing* atau pengujian secara lengkap dari awal hingga akhir untuk memastkan halaman *Salary Slip List* dan *Salary Slip* berjalan sesuai dengan yang diharapkan.

Tabel 3.1. Tugas yang dilakukan setiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Tugas yang dilakukan
1	Mengikuti *onboarding* dan mempelajari *boilerplate frontend* perusahaan
2	*Revamp employee form validation* dan *revamp project/ product form validation*
3	Mengerjakan halaman *Reimbursement Permit* dan *Reimbursement Input*
4	Mengerjakan halaman *Reimbursement Management*
5	Integrasi API untuk *pagination* pada halaman *User, Role*, dan *Project Product Management*
6	Integrasi API untuk *pagination* pada halaman *Stand Up Feed, Activity Log*, dan *People Report*
7	Integrasi API pada halaman *Reimbursement Management, Permit*, dan *Input*
8	Melakukan *end to end testing* pada halaman *Reimbursement Management, Permit*, dan *Input*
9	Mengerjakan halaman *Configuration Payroll List, Configuration Payroll Input*, dan *Salary Slip List*
10	Mengerjakan halaman *Salary Slip* dan *Payroll* pada halaman *User Management*
11	Integrasi API untuk halaman *Configuration Payroll List* dan *Input*
12	Integrasi API *Payroll* untuk halaman *User Management*
13	Integrasi API *Payroll* untuk halaman *User Management*
14	Melakukan *end to end testing* pada halaman *Configuration Payroll List, Input*, dan *User Payroll*
15	Integrasi API *Payroll* untuk halaman *Salary Slip*
16	Integrasi API *Payroll* untuk halaman *Salary Slip*
17	Melakukan *end to end testing* pada halaman *Salary Slip List* dan *Input*

3.4 Perancangan Aplikasi

Pada sub bab ini, akan dijelaskan perancangan HRIS yang dilakukan mulai dari pembuatan *User Requirement*, *Sitemap*, *Usecase Diagram*, *Flowchart*, dan *Wireframe*. Perusahaan telah menerapkan konsep *Component-Based Development*. *Component-Based Development* adalah proses perancangan aplikasi yang terbentuk dari komponen-komponen yang lebih sederhana dan dapat digunakan kembali [10]. Konsep komponen memberikan dampak yang signifikan dalam proses pengembangan terutama dalam penghematan waktu dan biaya. Dengan waktu pengembangan yang lebih singkat, jumlah jam kerja yang diperlukan juga berkurang, sehingga biaya operasional tim pengembang lebih minim.

3.4.1 User Requirement

Sebelum memulai pengembangan, *project manager* memberikan arahan terkait apa yang akan dikerjakan dalam pengembangan *Human Resource Internal System*, antara lain

1. Mengembangkan halaman *Reimbursement Management* yang terdapat daftar semua *reimbursement* yang merupakan anak-anak dari supervisor yang ada pada perusahaan. Pada halaman ini, terdapat opsi untuk melihat detail data-data reimbursement. Selain itu, didukung oleh fitur *search*, *sort*, serta filter data-data yang ada.

2. Mengembangkan halaman *Reimbursement Permit* yang hanya terdapat semua daftar *reimbursement*, *user* yang telah melakukan *login*. Pada halaman ini, terdapat opsi untuk melakukan penambahan *reimbursement* baru serta opsi untuk melihat detail. Dari halaman *Reimbursement Permit*, *user* dapat masuk ke halaman *Reimbursement Input* untuk melakukan input terhadap data *reimbursement* yang ingin dibuat atau melihat detail data yang telah dibuat. Selain itu, didukung oleh fitur *search*, *sort*, serta filter data-data yang ada.

3. Mengembangkan halaman *Reimbursement Input* yang didalamnya terdapat nama pengaju, opsi untuk memilih jenis *project* yang bersangkutan, jumlah yang diajukan, serta bukti transaksi yang harus diunggah. Pada halaman ini, digunakan untuk input serta *submit* data, melihat detail, *accept* atau *reject reimbursement*, dan *cancel request reimbursement* yang telah di *submit*.

4. Mengembangkan halaman *Configuration Payroll* yang terdapat daftar *configuration payroll*. Pada halaman ini, terdapat opsi untuk melakukan penambahan *configuration* baru serta opsi untuk mengedit dan menghapus *configuration*. Dari halaman *Configuration*, *user* dapat masuk ke halaman *Add Configuration*. Selain itu, didukung oleh fitur *search* dan *sort* data-data yang ada.

5. Mengembangkan halaman *Add Configuration* yang didalamnya terdapat opsi untuk memilih jenis *employee* status, *configuration name*, dan *allowance type*. Selain itu, terdapat *add configuration* untuk menambahkan *allowance type* jika lebih dari satu dan *icon* berbentuk tempat sampah untuk menghapus *allowance type*. Pada halaman ini, digunakan untuk input serta *submit* data dan mengedit *configuration* yang telah di *submit*.

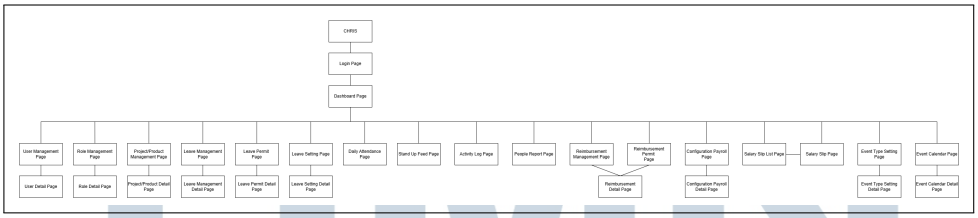
6. Mengembangkan halaman *Salary Slip List* yang terdapat daftar semua *salary slip* yang telah diatur pada halaman *User Management Input*. Pada halaman ini, terdapat opsi untuk melihat detail *salary slip* masing-masing karyawan serta dapat memilih bulan dan atau tahun untuk melihat daftar semua *salary slip* berdasarkan bulan dan atau tahun yang telah dipilih. Selain itu, didukung oleh fitur *search* dan *sort* data-data yang ada.

7. Mengembangkan halaman *Salary Slip* yang menampilkan data-data *salary slip*, antara lain nama pemilik *salary slip*, NIK, *contact*, *job title*, *employment type*, *basic salary*, *allowances*, *deductions*, total akhir gaji, nama bank serta nomor rekening. Selain itu, terdapat tombol untuk mengunduh *salary slip* khusus untuk karyawan.

8. Mengembangkan *Payroll Information* pada halaman *User Management Input* yang didalamnya terdapat *basic salary*, opsi untuk memilih jenis *configuration payroll* yang telah dibuat pada halaman *Configuration*. Selain itu, terdapat *add allowance* atau *add deduction* untuk menambahkan *allowance* atau *deduction* khusus untuk karyawan tersebut serta *icon* berbentuk tempat sampah untuk menghapus *allowance* atau *deduction* yang telah ditambahkan.

3.4.2 Sitemap

Gambar 3.2 adalah gambar *sitemap* dari *Human Resource Internal System*. Berdasarkan *sitemap* tersebut, halaman pertama yang akan terlihat oleh *user* adalah halaman *login*. Ketika *user* berhasil melakukan *login*, maka *user* akan diarahkan ke halaman *Dashboard*. Selanjutnya *user* dapat mengakses halaman-halaman berikutnya, antara lain *User Management* yang dapat diarahkan ke halaman *User Detail*, *Role Management* yang dapat diarahkan ke halaman *Role Detail*, *Project/Products* yang dapat diarahkan ke halaman *Project/Products Detail*, *Leave Management* yang dapat diarahkan ke halaman *Leave Management Detail*, *Leave Permit* yang dapat diarahkan ke halaman *Leave Permit Detail*, *Leave Setting* yang dapat diarahkan ke halaman *Leave Setting Detail*, *Daily Attendance*, *Stand Up Feed*, *Activity Log*, *People Report*, *Reimbursement Management* yang dapat diarahkan ke halaman *Reimbursement Detail*, *Reimbursement Permit* yang dapat diarahkan ke halaman *Reimbursement Detail*, *Event Calendar* yang dapat diarahkan ke halaman *Event Calendar Detail*, *Event Type Setting* yang dapat diarhkan ke halaman *Event Type Setting Detail Configuration Payroll* yang dapat diarahkan ke halaman *Configuration Payroll Detail*, dan *Salary Slip List* yang dapat diarahkan ke halaman *Salary Slip*.

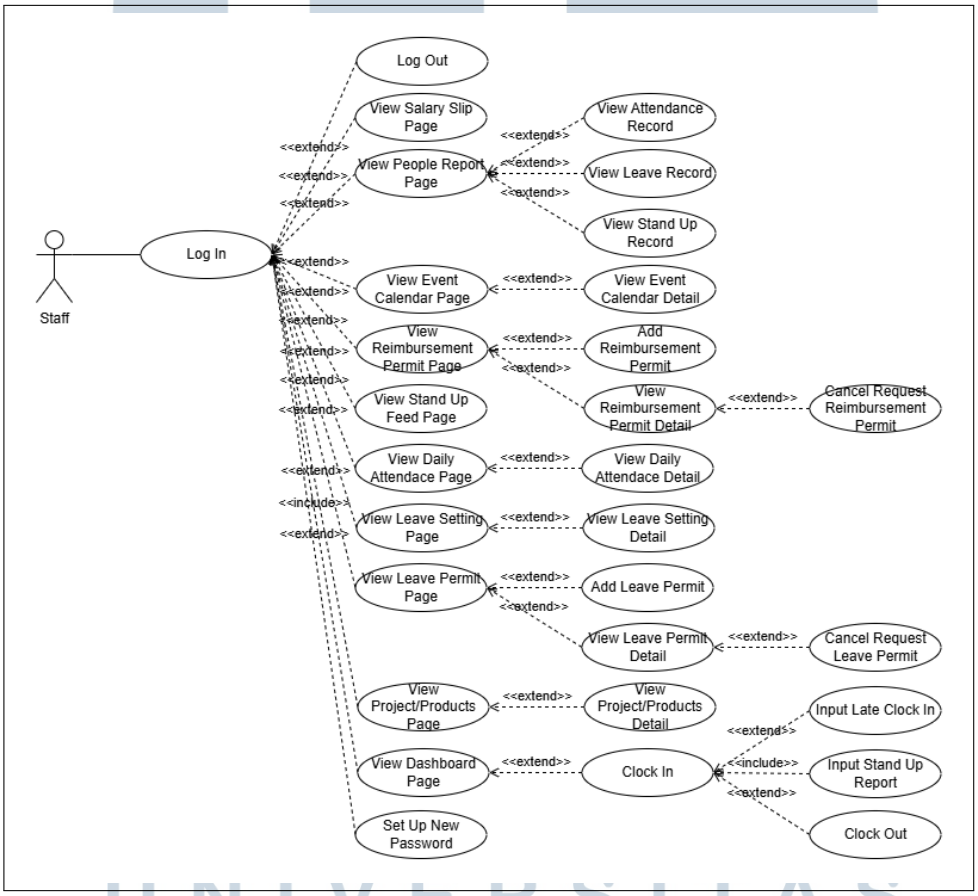


Gambar 3.2. *Sitemap Human Resource Internal System*

3.4.3 Usecase Diagram

Gambar 3.3 merupakan *Usecase* diagram untuk aktor *staff*. Langkah pertama yang dilakukan oleh *staff* yaitu melakukan *log in* terlebih dahulu. Bagi *staff* baru, maka akan diarahkan untuk melakukan *set up new password*. Setelah berhasil *log in*, *staff* secara otomatis akan diarahkan ke halaman *Dashboard*. Pada halaman *Dashboard*, *staff* dapat melakukan *clock in* serta mengisi laporan *stand up*. Jika *staff* terlambat melakukan *clock in*, maka *staff* dapat mengisi alasan keterlambatan. Kemudian, *staff* dapat mengisi laporan *stand up*. Setelah melakukan clock in dan mengisi laporan *stand up*, maka *staff* dapat melakukan *clock out*. *Staff* juga dapat

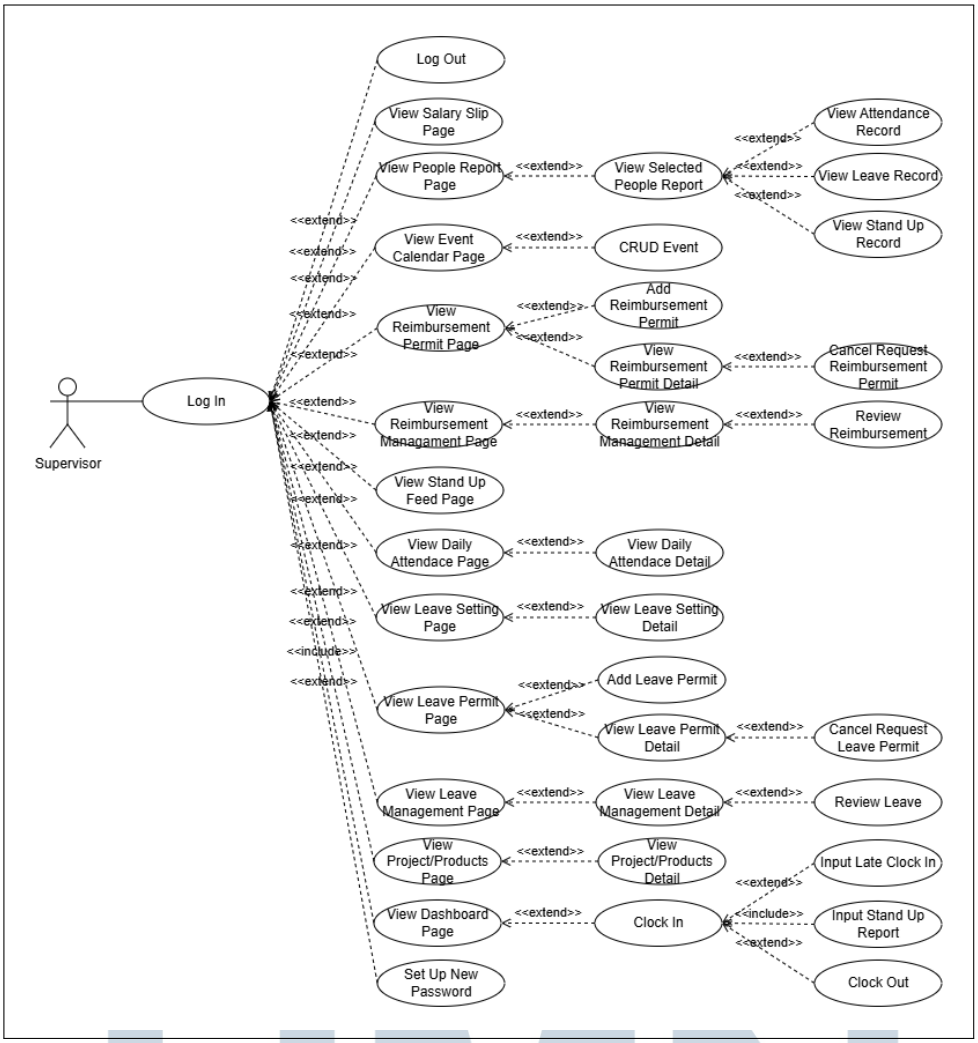
mengakses berbagai halaman seperti *Project/Products, Leave Permit, Leave Setting, Daily Attendance, Stand Up Feed, Reimbursement Permit, Event Calendar, People Report, Salary Slip* dan melakukan *log out*. Selain itu, terdapat fitur tambahan seperti *staff* dapat melihat secara detail untuk halaman *project/products, leave permit, leave setting, daily attendance, reimbursement permit, event calendar*. *Staff* juga dapat melihat record tertentu seperti *attendance record, leave record, stand up record* pada halaman *People Report*, *staff* dapat menambahkan data *leave permit* dan *reimbursement permit* serta melakukan pembatalan permintaan atau *cancel request* pada *leave permit detail* dan *reimbursement permit detail*.



Gambar 3.3. Usecase Diagram Staff

Gambar 3.4 merupakan *Usecase* diagram untuk aktor *supervisor*. Langkah pertama yang dilakukan oleh *supervisor* yaitu melakukan *log in* terlebih dahulu. Bagi *supervisor* baru, maka akan diarahkan untuk melakukan *set up new password*. Setelah berhasil *log in*, *supervisor* secara otomatis akan diarahkan ke halaman *Dashboard*. Pada halaman *Dashboard*, *supervisor* dapat melakukan *clock in* serta mengisi laporan *stand up*. Jika *supervisor* terlambat melakukan *clock in*,

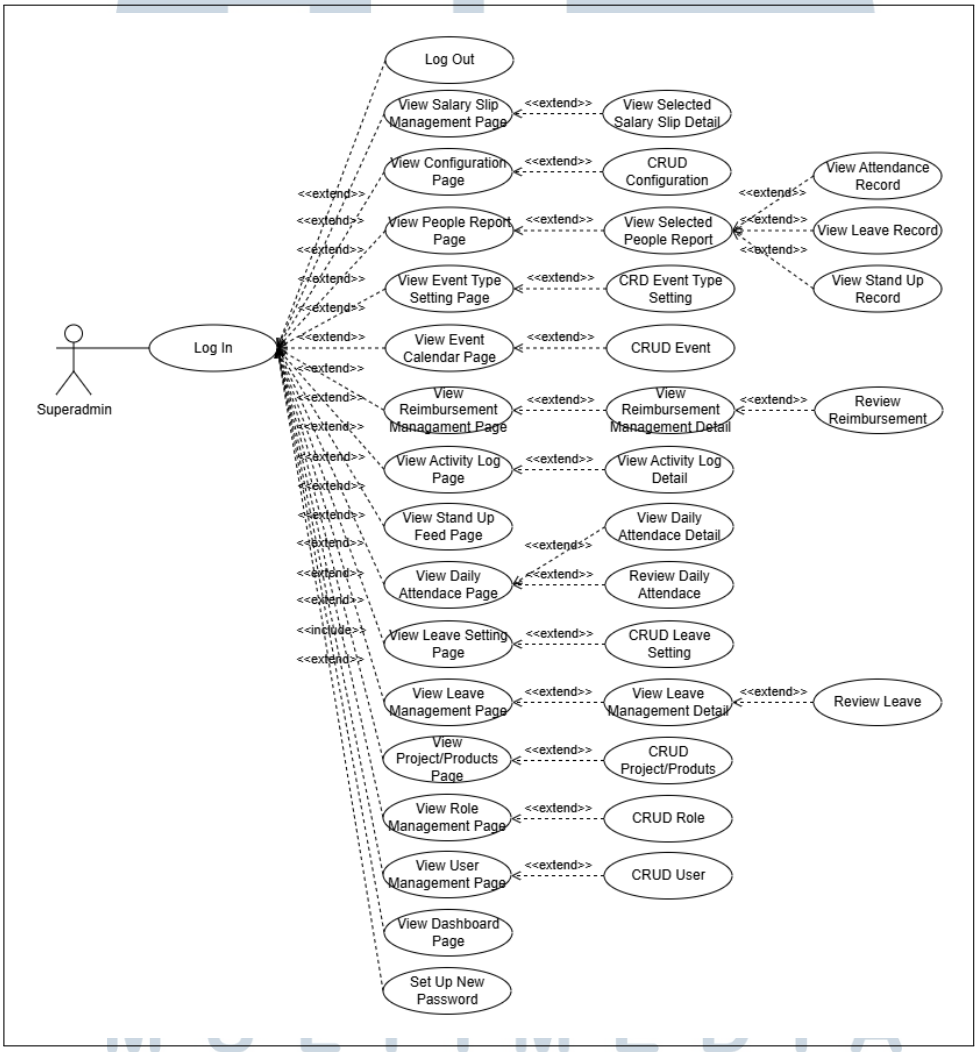
maka *supervisor* dapat mengisi alasan keterlambatan. Kemudian, *supervisor* dapat mengisi laporan *stand up*. Setelah melakukan clock in dan mengisi laporan *stand up*, maka *supervisor* dapat melakukan *clock out*. *supervisor* juga dapat mengakses berbagai halaman seperti *Project/Products, Leave Management, Leave Permit, Leave Setting, Daily Attendance, Stand Up Feed, Reimbursement Management, Reimbursement Permit, Event Calendar, People Report, Salary Slip* dan melakukan *log out*. Selain itu, terdapat fitur tambahan seperti *supervisor* dapat melihat secara detail untuk halaman *Project/Products, Leave Management, Leave Permit, Leave Setting, Daily Attendance, Reimbursement Management, Reimbursement Permit, Event Calendar*. *Supervisor* juga dapat melihat record tertentu seperti *attendance record, leave record, stand up record* milik *staff* yang berada di bawah supervisinya yang telah dipilih pada halaman *People Report*. *Supervisor* dapat menambahkan data *leave permit*, *reimbursement permit*, dan *event*. *Supervisor* juga dapat melakukan pembatalan permintaan atau *cancel request* pada *leave permit detail* dan *reimbursement permit detail*. Selain itu, *supervisor* juga dapat melakukan *review* berupa *accept* atau *reject leave dan reimbursement*. *Supervisor* juga memiliki akses untuk *update* dan *delete event* yang telah ditambahkan.



Gambar 3.4. Usecase Diagram Supervisor

Gambar 3.5 merupakan *Usecase* diagram untuk aktor *superadmin*. Langkah pertama yang dilakukan oleh *superadmin* yaitu melakukan *log in* terlebih dahulu. Bagi *superadmin* baru, maka akan diarahkan untuk melakukan *set up new password*. Setelah berhasil *log in*, *superadmin* secara otomatis akan diarahkan ke halaman *Dashboard*. *Superadmin* juga dapat mengakses berbagai halaman seperti *User Management, Role Management, Project/Products, Leave Management, Leave Setting, Daily Attendance, Stand Up Feed, Activity Log, Reimbursement Management, Event Calendar, Event Type Setting, People Report, Configuration, Salary Slip Management*, dan melakukan *log out*. Selain itu, terdapat fitur tambahan seperti *superadmin* dapat melihat secara detail untuk halaman *User Management, Role Management, Project/Products, Leave Management, Leave Setting, Daily Attendance, Activity Log, Reimbursement Management, Event, Event Type Setting,*

Configuration, Salary Slip. *Superadmin* juga dapat melihat record tertentu seperti *attendance record, leave record, stand up record* yang telah dipilih pada halaman *People Report*. *Superadmin* dapat menambahkan data *user*, *role*, *project/product*, *leave setting*, *event*, *event type*, dan *configuration*. Selain itu, *superadmin* juga dapat melakukan *review* berupa *accept* atau *reject leave, daily attendance, dan reimbursement*. *Superadmin* juga memiliki akses untuk *update* dan *delete user, role, project/product, leave setting, event, dan configuration* yang telah ditambahkan serta *delete event type setting*.



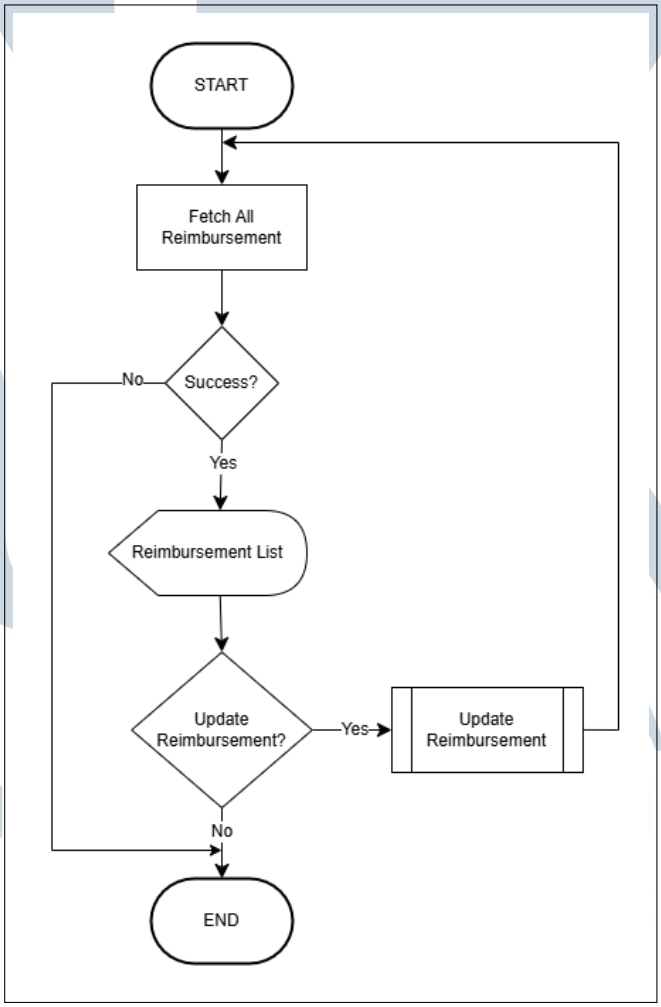
Gambar 3.5. Usecase Diagram Superadmin

3.4.4 Flowchart & Sequence Diagram

Berikut ditampilkan *flowchart* dan *sequence diagram* untuk menggambarkan alur antarmuka dari sisi user.

A Flowchart Reimbursement Management

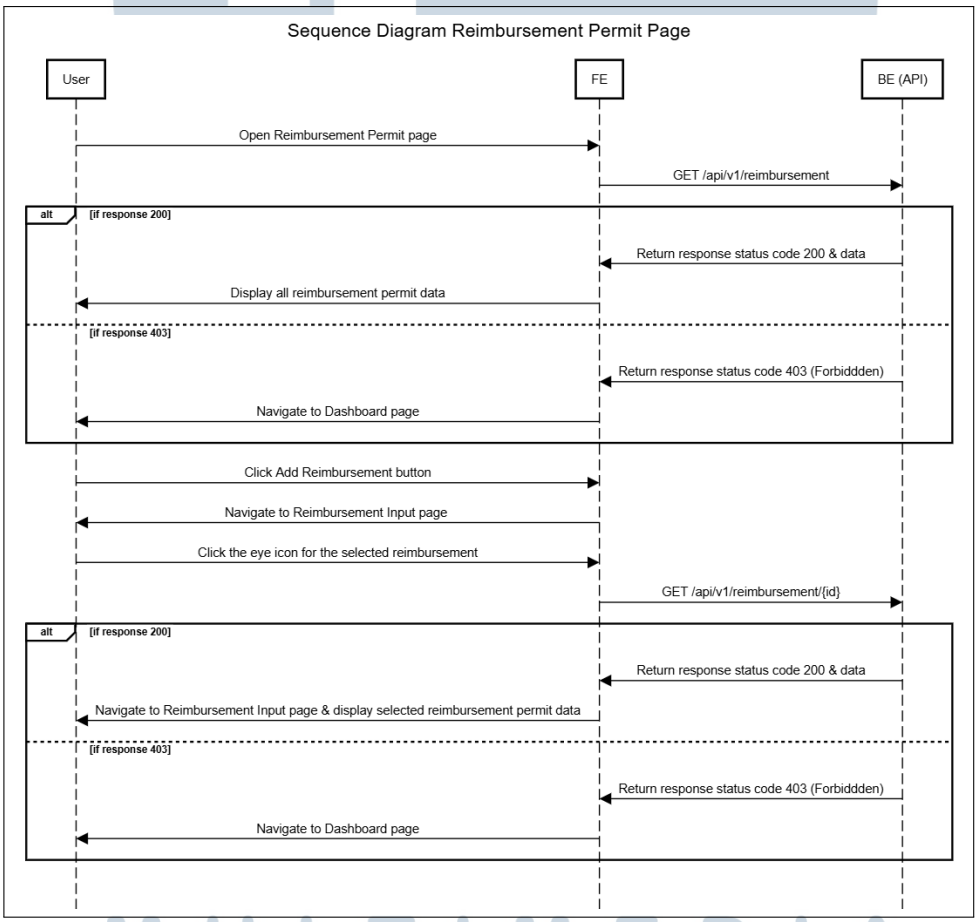
Gambar 3.6 adalah gambar *flowchart* pada halaman *Reimbursement Management*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”reimbursement/all” yang ditunjukkan pada Gambar 3.2. Jika permintaan GET API berhasil dengan status *code* 200 dan *message success*, maka semua daftar *reimbursement management* akan ditampilkan yang ditunjukkan pada Gambar 3.28. Selanjutnya, *user* dapat memilih untuk *update reimbursement*.



Gambar 3.6. Flowchart Reimbursement Management

B Sequence Diagram Reimbursement Permit

Gambar 3.7 adalah gambar *sequence* diagram pada halaman *Reimbursement Permit*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”reimbursement” yang ditunjukkan pada Gambar 3.3. Jika permintaan GET API berhasil dengan status *code* 200 dan *message success*, maka semua daftar *reimbursement permit* akan ditampilkan yang ditunjukkan pada Gambar 3.29. *User* juga dapat memilih untuk menambahkan *reimbursement* baru dengan menekan tombol *add reimbursement* dan menghapus *reimbursement* dengan menekan *icon* berbentuk mata dan akan diarahkan ke halaman *Reimbursement Input*.

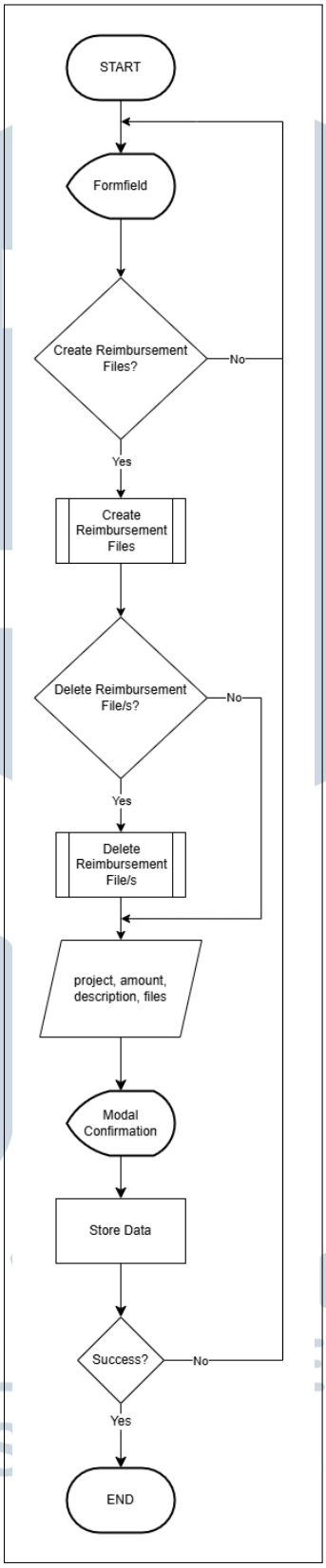


Gambar 3.7. Sequence Diagram Reimbursement Permit

C Flowchart Create Reimbursement

Gambar 3.8 adalah *flowchart* pada halaman *Reimbursement Input*. Proses akan dimulai dari ditampilkannya *form field* untuk mengisi data

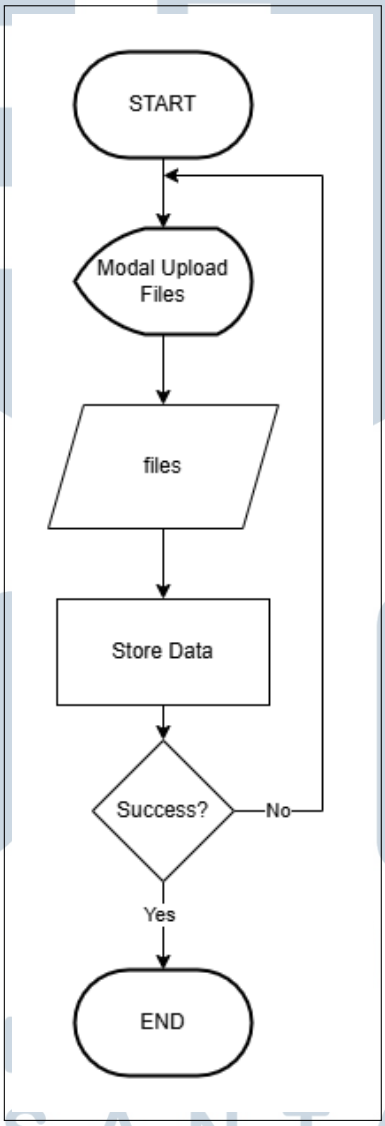
reimbursement yaitu *project, amount, dan description* dan akan dilanjutkan dengan proses pengunggahan file melalui pemanggilan API ke *endpoint* ”/api/v1/file/upload?module={module} yang ditunjukkan pada Gambar 3.8”. *User* dapat memilih untuk menghapus file yang telah diunggah dan akan dilanjutkan dengan pemanggilan API dengan metode DELETE ke *endpoint* ”file/{id} yang ditunjukkan pada Gambar 3.11”. Selanjutnya akan ditampilkan modal untuk konfirmasi pengisian. Setelah itu, data yang terdiri dari *project id*, *amount*, *description*, dan *id files* yang telah diunggah dan disimpan dalam bentuk array (kumpulan data) akan dikirimkan dengan metode POST API ke *endpoint* ”reimbursement” yang ditunjukkan pada Gambar 3.7.



Gambar 3.8. Flowchart Create Reimbursement

D Flowchart Create Reimbursement Files

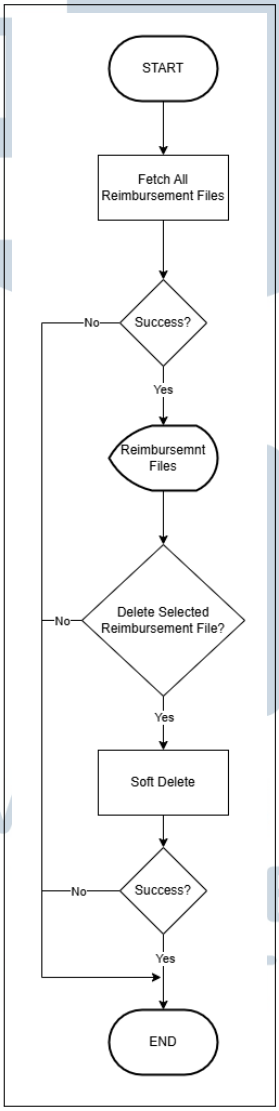
Gambar 3.9 adalah *flowchart* pada halaman *Reimbursement Input*. Proses akan dimulai dari ditampilkannya modal *upload files*. Selanjutnya, *user* memilih *files* yang akan diunggah. Setelah itu, *files* yang diunggah akan dikirimkan dalam bentuk *InputData* berupa *array* dan parameter *reimbursement* dengan metode POST API ke *endpoint* "/api/v1/file/upload?module={module} yang ditunjukkan pada Gambar 3.8".



Gambar 3.9. Flowchart Create Reimbursement Files

E Flowchart Delete Reimbursement Files

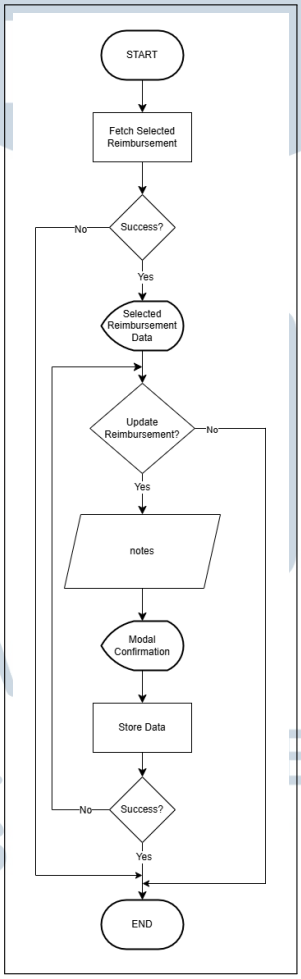
Gambar 3.10 adalah *flowchart* pada halaman *Reimbursement Input*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”file/generate-url/{id}” yang ditunjukkan pada Gambar 3.5. Jika permintaan GET API berhasil dengan status *code* 200 dan *message success*, maka semua daftar *reimbursement files* akan ditampilkan. Selanjutnya, *user* dapat memilih untuk menghapus *reimbursement file* yang dipilih yang akan dilanjutkan dengan pemanggilan API dengan metode DELETE ke *endpoint* ”file/{id} yang ditunjukkan pada Gambar 3.11”. Jika permintaan berhasil dengan status *code* 200 dan *message file deleted successfully*, maka file yang telah dipilih akan dihapus.



Gambar 3.10. Flowchart Delete Reimbursement Files

F Flowchart Update Reimbursement

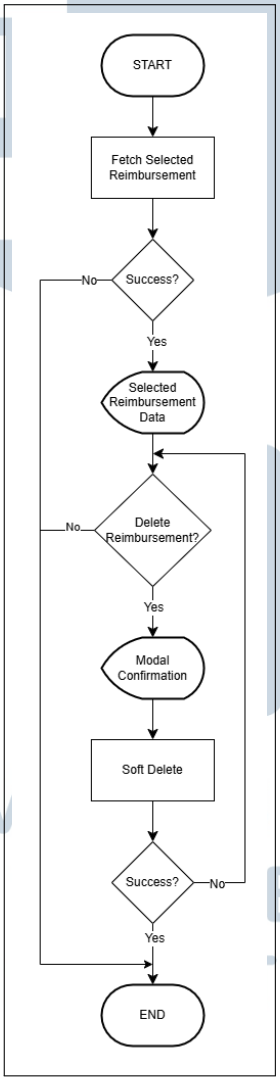
Gambar 3.11 adalah flowchart pada halaman *Reimbursement Input*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”reimbursement/{id}” yang ditunjukkan pada Gambar 3.4 yang akan menampilkan data *reimbursement* yang telah dipilih untuk di *review*. *Review* terbagi menjadi dua pilihan, yang pertama *reject* dan yang kedua yaitu *accept*. Terdapat penambahan *notes* jika diperlukan. Setelah melakukan perubahan, data *reimbursement* yang terdiri dari notes dan status akan dilanjutkan dengan pemanggilan API dengan metode PUT ke *endpoint* ”reimbursement/{id}” yang ditunjukkan pada Gambar 3.9 untuk *user* yang menekan tombol *accept*atau 3.9 untuk *user* yang menekan tombol *reject*. Jika permintaan berhasil dengan status *code* 200 dan *message reimbursement successfully accepted* atau *rejected*, maka file yang telah dipilih sudah dilakukan *review*.



Gambar 3.11. Flowchart Update Reimbursement

G Flowchart Delete Reimbursement

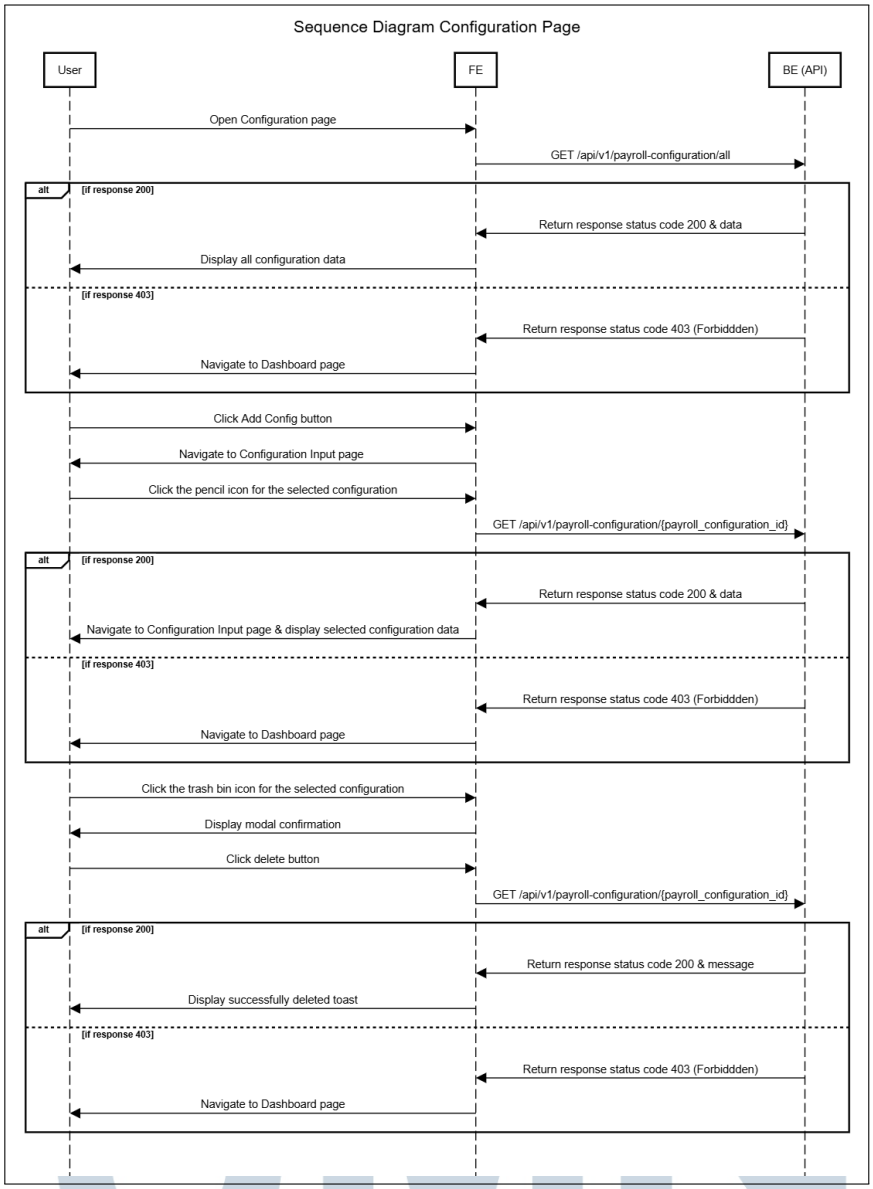
Gambar 3.12 adalah *flowchart* pada halaman *Reimbursement Input*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”reimbursement/{id}” yang ditunjukkan pada Gambar 3.4. Jika permintaan GET API berhasil dengan status *code* 200 dan *message successfully fetched a single reimbursement*, maka data *reimbursement* yang telah dipilih akan ditampilkan. *User* dapat memilih untuk menghapus *reimbursement* yang dipilih dengan menekan tombol *cancel request*, dan selanjutnya akan muncul modal konfirmasi. Setelah itu, data reimbursement akan dihapus dengan metode DELETE API ke *endpoint* ”reimbursement/{id} yang ditunjukkan pada Gambar 3.10”.



Gambar 3.12. Flowchart Delete Reimbursement

H Sequence Diagram Configuration

Gambar 3.13 adalah gambar *sequence* diagram pada halaman *Configuration*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”payroll-configuration/all yang akan ditunjukkan pada Gambar 3.12”. Jika permintaan GET API berhasil dengan status *code* 200 dan *message* , maka semua daftar *configuration* akan ditampilkan. Selanjutnya, *user* dapat memilih untuk menambahkan *configuration* baru dengan menekan tombol *Add Configuration*, mengedit dengan menekan *icon* berbentuk pencil yang akan diarahkan ke halaman *Configuration Payroll Input*, serta menghapus *configuration* dengan menekan *icon* berbentuk tempat sampah.

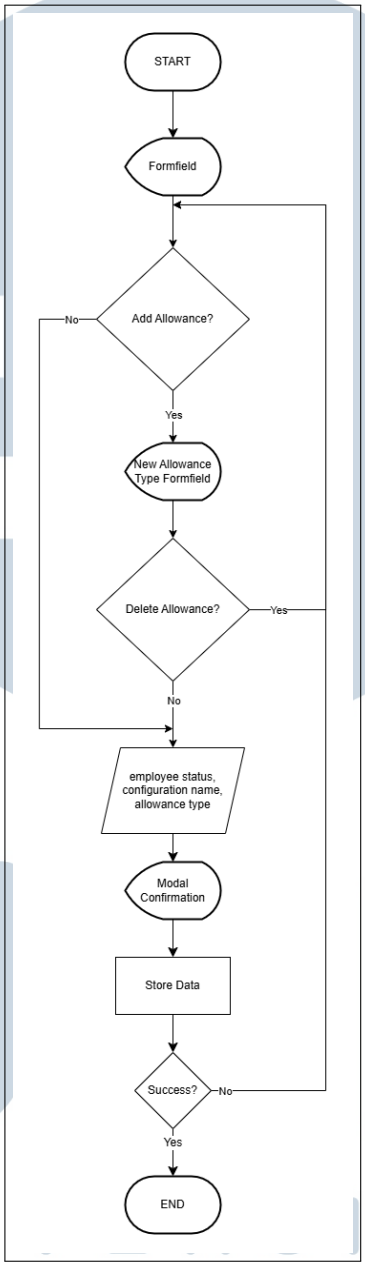


Gambar 3.13. Sequence Diagram Configuration

I Flowchart Create Configuration

Gambar 3.14 adalah *flowchart* pada halaman *Configuration Payroll Input*. Proses akan dimulai dari ditampilkannya *form field* untuk mengisi data *configuration* dan dilanjutkan untuk menambahkan *allowance*. *User* dapat memilih untuk menghapus *allowance type* yang telah ditambahkan. Jika data telah diisi dan *user* telah menekan tombol *submit*, maka selanjutnya akan ditampilkan modal untuk konfirmasi. Setelah itu, data yang terdiri dari *employee status*, *configuration name*, dan *allowance type* yang disimpan dalam bentuk array (kumpulan data) akan

dikirimkan dengan metode POST API ke *endpoint* ”payroll-configuration” yang ditunjukkan pada Gambar 3.15.

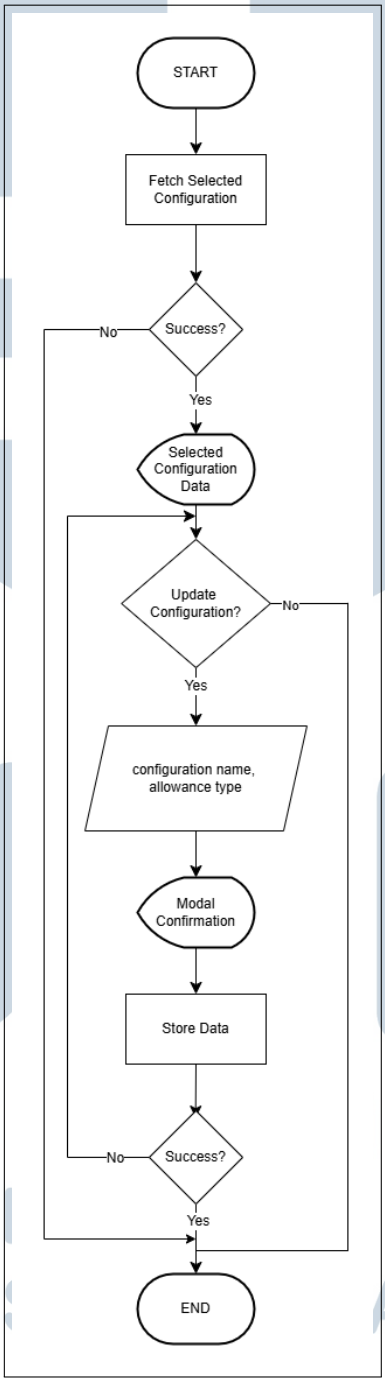


Gambar 3.14. Flowchart Create Configuration

J Flowchart Update Configuration

Gambar 3.15 adalah flowchart pada halaman *Configuration Payroll Input*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”payroll-configuration/{id}” yang ditunjukkan pada Gambar 3.13 untuk menampilkan data

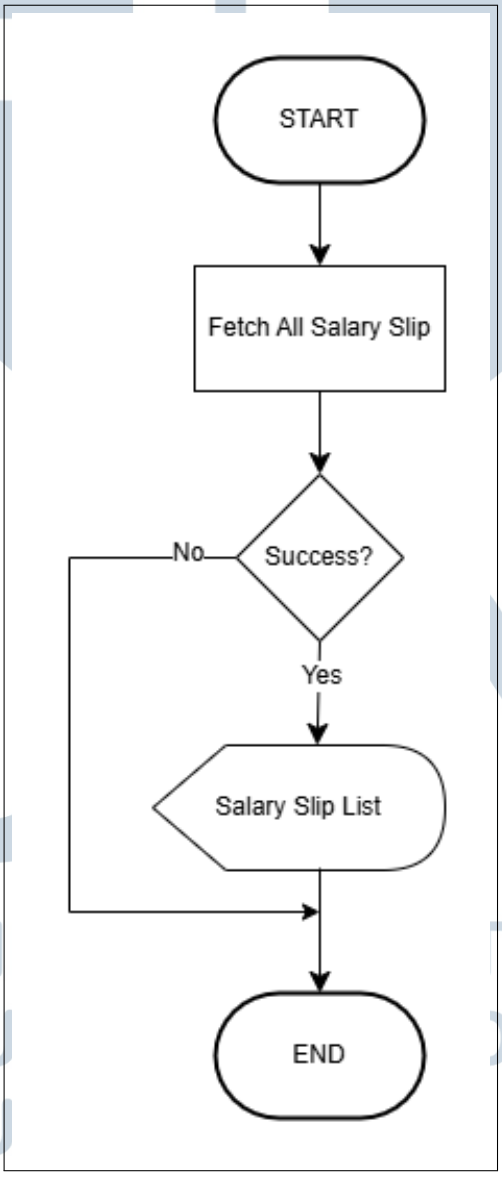
configuration yang telah dipilih untuk di *update*. Setelah melakukan perubahan, data *configuration* yang terdiri dari *configuration name* dan *allowance type* akan dikirimkan dengan metode PUT API ke *endpoint* "payroll-configuration/{id}" yang ditunjukkan pada Gambar 3.16.



Gambar 3.15. Flowchart Update Configuration

K Flowchart Salary Slip Management

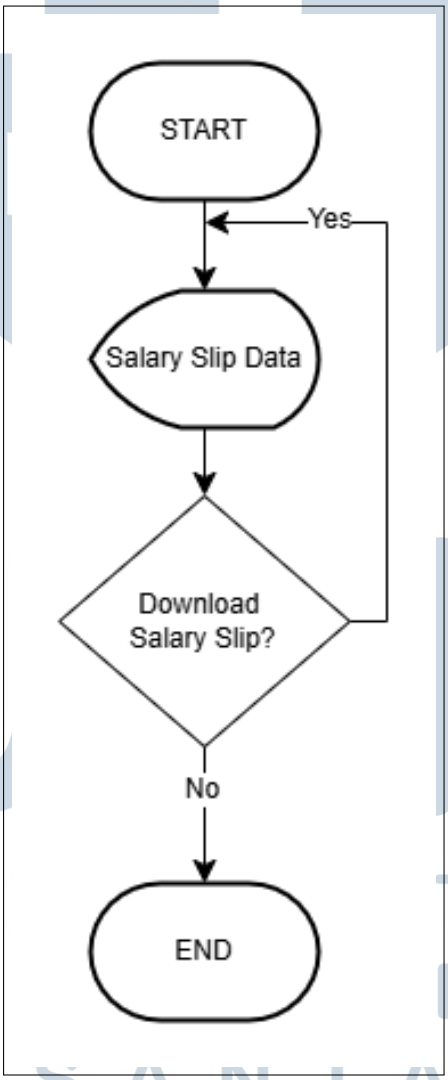
Gambar 3.16 adalah gambar *flowchart* pada halaman *Salary Slip List*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”salary-slip/all?date={date}” yang ditunjukkan pada Gambar 3.24. Jika permintaan GET API berhasil dengan status *code* 200 dan *message*, maka semua daftar *salary slip* akan ditampilkan. *User* dapat memilih bulan dan atau tahun untuk melihat *salary slip* sesuai dengan bulan yang dipilih.



Gambar 3.16. Flowchart Salary Slip List

L Flowchart Salary Slip

Gambar 3.17 adalah gambar *flowchart* pada halaman *Salary Slip*. Proses akan dimulai dari pemanggilan GET API ke *endpoint* ”salary-slip?date={date} yang ditunjukkan pada Gambar 3.26”. Jika permintaan GET API berhasil dengan status *code* 200 dan *message*, maka semua data *salary slip* yang terdiri dari informasi pemilik *salary slip*, *allowances*, *deductions*, serta total gaji yang diterima akan ditampilkan. *User* dapat memilih bulan dan atau tahun untuk melihat *salary slip* sesuai dengan bulan yang dipilih.



Gambar 3.17. Flowchart Salary Slip

3.5 API Contract

Berikut adalah *API Contract* antara *backend* dan *frontend* yang digunakan untuk berinteraksi dalam proses integrasi dan pengembangan fitur.

Tabel 3.2. Get All Reimbursement Management

Relative Path	/api/v1/reimbursement/all
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 2,
        "rows": [
            {
                "id": "9a23e007-9616-488e-bb78-d4a6bdc834b7",
                "app_id": "202506-003",
                "amount": "200000",
                "description": "Sushi",
                "status": "rejected",
                "notes": "no",
                "created_at": "2025-06-12T05:04:30.161Z",
                "updated_at": "2025-06-12T05:05:14.032Z",
                "user": {
                    "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
                    "fullname": "Superadmin",
                    "email": "concise@concise.co.id",
                    "supervisor_id": null
                },
                "reviewed_by": {
                    "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
                    "fullname": "Superadmin",
                    "email": "concise@concise.co.id"
                },
                "project_product": {
                    "id": "d567f0d0-468c-11f0-bb87-2d3fd7c426dd",
                    "name": "project 1",
                    "description": "gk mo"
                },
                "files": [
                    {
                        "id": "df7cbc76-ad80-4db3-adf4-fc23b1f39b76",
                        "name": "salmon nigiri.jpg",
                        "original_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/
                        user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=efda334c-7ce4-44b5-83af-775ecb94350e&name=salmon
                        nigiri.jpg",
                        "created_at": "2025-06-12T05:04:16.564Z"
                    }
                ]
            },
```


Tabel 3.3. Get All Reimbursement Permit

Relative Path	/api/v1/reimbursement
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 2,
        "rows": [
            {
                "id": "9a23e007-9616-488e-bb78-d4a6bdc834b7",
                "app_id": "202506-003",
                "amount": "200000",
                "description": "Sushi",
                "status": "rejected",
                "notes": "no",
                "created_at": "2025-06-12T05:04:30.161Z",
                "updated_at": "2025-06-12T05:05:14.032Z",
                "user": {
                    "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
                    "fullname": "Superadmin",
                    "email": "concise@concise.co.id"
                },
                "reviewed_by": {
                    "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
                    "fullname": "Superadmin",
                    "email": "concise@concise.co.id"
                },
                "project_product": {
                    "id": "d567f0d0-468c-11f0-bb87-2d3fd7c426dd",
                    "name": "project 1",
                    "description": "gk mo"
                },
                "files": [
                    {
                        "id": "df7cbc76-ad80-4db3-adf4-fc23b1f39b76",
                        "name": "salmon nigiri.jpg",
                        "original_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/
                        id=efda334c-7ce4-44b5-83af-775ecb94350e&name=salmon nigiri.jpg",
                        "created_at": "2025-06-12T05:04:16.564Z"
                    }
                ]
            },
```

Tabel 3.4. Get Reimbursement By Id

Relative Path	/api/v1/reimbursement/{id}
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Successfully Fetched A Single Reimbursement",
    "data": {
        "id": "9a23e007-9616-488e-bb78-d4a6bdc834b7",
        "app_id": "202506-003",
        "amount": "200000",
        "description": "Sushi",
        "status": "rejected",
        "notes": "no",
        "created_at": "2025-06-12T05:04:30.161Z",
        "updated_at": "2025-06-12T05:05:14.032Z",
        "user": {
            "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
            "fullname": "Superadmin",
            "email": "concise@concise.co.id"
        },
        "reviewed_by": {
            "id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
            "fullname": "Superadmin",
            "email": "concise@concise.co.id"
        },
        "project_product": {
            "id": "d5679b08-468c-11f0-bb87-2d3fd7c426dd",
            "name": "project 1",
            "description": "gk mo"
        },
        "files": [
            {
                "id": "df7cbc76-ad80-4db3-adf4-fc23b1f39b76",
                "name": "salmon nigiri.jpg",
                "original_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=efda334c-7ce4-44b5-83af-775ecb94350e&name=salmon nigiri.jpg",
                "created_at": "2025-06-12T05:04:16.564Z"
            }
        ]
    }
}
```


Tabel 3.5. Get Reimbursement File

Relative Path	/api/v1/file/generate-url/{id}
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 1,
        "rows": [
            {
                "id": "df7cbc76-ad80-4db3-adf4-fc23b1f39b76",
                "name": "salmon nigiri.jpg",
                "original_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/
                user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=efda334c-7ce4-44b5-83af-775ecb94350e&name=salmon nigiri.jpg",
                "resized_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/
                user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=efda334c-7ce4-44b5-83af-775ecb94350e&name=resized_salmon
                nigiri.jpg",
                "thumbnail_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/
                user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=efda334c-7ce4-44b5-83af-775ecb94350e&name=thumbnail_salmon
                nigiri.jpg",
                "mime_type": "jpeg",
                "is_uploaded": true,
                "module_name": "reimbursement",
                "created_at": "2025-06-12T05:04:16.564Z",
                "updated_at": "2025-06-12T05:04:16.564Z"
            }
        ]
    }
}
```

Tabel 3.6. Get All Project Product

Relative Path	/api/v1/project-products/all
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 1,
        "rows": [
            {
                "id": "d567f0d0-468c-11f0-bb87-2d3fd7c426dd",
                "client_name": "belpa",
                "client_phone": "8121214111",
                "client_email": "belpa@email.com",
                "name": "project 1",
                "description": "gk mo",
                "start_date": "2025-06-11T00:00:00.000Z",
                "end_date": "2025-06-19T00:00:00.000Z",
                "status": "On Hold",
                "is_project": false,
                "abbreviation": "BP",
                "created_at": "2025-06-11T06:25:13.821Z",
                "updated_at": "2025-06-11T06:25:13.821Z"
            }
        ]
    }
}
```

Tabel 3.7. Create Reimbursement

Relative Path	/api/v1/reimbursement
Method	POST

Body	<pre>{
	"project_id": "xxxxxxxx",
	"amount": xxxxxxxx,
	"description": "xxxxxxxx",
	"file_ids": ["xxxxxxxx"]
	}</pre>

Success Response	HTTP Status Code 201
	<pre>{
	"code": 201,
	"message": "Reimbursement created successfully",
	"data": {
	"id": "97e203ed-f62a-4746-ae73-2401b8fe92da",
	"user_id": "8f83cfaf-2655-4c54-88dd-38889ada3b7d",
	"project_id": "d567f0d0-468c-11f0-bb87-2d3fd7c426dd",
	"app_id": "202506-002",
	"amount": "100000",
	"description": "Sushi",
	"status": "pending",
	"notes": null,
	"reviewed_by_id": null,
	"created_at": "2025-06-12T04:56:33.018Z",
	"updated_at": "2025-06-12T04:56:33.019Z",
	"deleted_at": null,
	"files": [
	{
	"id": "2928db58-8fc9-4f19-83ed-0aeb56913f02",
	"name": "salmon nigiri.jpg",
	"original_file_path": "https://storage-lan.concise.co.id:443/hris/reimbursement/2025-06-12/
	user=8f83cfaf-2655-4c54-88dd-38889ada3b7d/id=7ba436b7-dca1-4e04-a2a4-394e6ff28b40&name=salmon nigiri.jpg"
	}
	]
	}
	}</pre>

Tabel 3.8. Create Reimbursement File

Relative Path	/api/v1/file/upload?module={module}
Method	POST
Body	{ "files": "xxxxxxxx" }
Query Params	module
Success Response	HTTP Status Code 201 

Tabel 3.9. Update Reimbursement By Id

Relative Path	/api/v1/reimbursement/{id}
Method	PUT
Body	{ "status": "xxxxxxxx", "notes": "xxxxxxxx" }
Success Response (Accepted)	HTTP Status Code 200
Success Response (Rejected)	HTTP Status Code 200

Tabel 3.10. Delete Reimbursement By Id

Relative Path	/api/v1/reimbursement/{id}
Method	DELETE

Success Response (Rejected)	HTTP Status Code 200
	{
	"code": 200,
	"message": "Reimbursement deleted successfully"
	}

Tabel 3.11. Delete Reimbursement File

Relative Path	/api/v1/file/{id}
Method	DELETE

Success Response	HTTP Status Code 200
	{
	"code": 200,
	"message": "File deleted successfully"
	}

Tabel 3.12. Get All Configuration Payroll

Relative Path	/api/v1/payroll-configuration/all
Method	GET

Success Response	HTTP Status Code 200
	{
	"code": 200,
	"message": "Success",
	"data": {
	"count": 1,
	"rows": [
	{
	"id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
	"name": "Config Payroll Intern",
	"created_at": "2025-06-12T07:44:38.829Z",
	"updated_at": "2025-06-12T07:44:38.829Z",
	"employment_status": {
	"id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
	"employment_type": "Internship"
	}
	}
	]
	}
	}

Tabel 3.13. Get Configuration Payroll By Id

Relative Path	/api/v1/payroll-configuration/{payroll_configuration_id}
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Successfully Fetched A Single Payroll",
    "data": {
        "id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
        "name": "Config Payroll Intern",
        "employment_status": {
            "id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
            "employment_type": "Internship"
        },
        "allowance_types": [
            {
                "id": "15c73d40-cea1-4ae5-9d00-97ea179bd8cd",
                "name": "THR"
            },
            {
                "id": "d52fbdff-f989-4774-91a1-ebe3f2edcf32",
                "name": "Makan"
            }
        ]
    }
}
```

Tabel 3.14. Get User Employment Status

Relative Path	/api/v1/employee-statuses
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "List Employment Status Fetched Successfully",
    "count": 4,
    "data": [
        {
            "id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
            "employment_type": "Internship",
            "created_at": "2025-06-11T06:01:24.810Z",
            "updated_at": "2025-06-11T06:01:24.810Z",
            "deleted_at": null
        },
        {
            "id": "03ec1028-9537-4de8-9b34-1cbd98c15451",
            "employment_type": "Probation",
            "created_at": "2025-06-11T06:01:24.810Z",
            "updated_at": "2025-06-11T06:01:24.810Z",
            "deleted_at": null
        },
        {
            "id": "e97cf5b1-b661-49a7-9545-938f48546ebb",
            "employment_type": "Part Time",
            "created_at": "2025-06-11T06:01:24.810Z",
            "updated_at": "2025-06-11T06:01:24.810Z",
            "deleted_at": null
        },
        {
            "id": "ad12375b-befa-4f29-8743-e1c765581cbf",
            "employment_type": "Full Time",
            "created_at": "2025-06-11T06:01:24.810Z",
            "updated_at": "2025-06-11T06:01:24.810Z",
            "deleted_at": null
        }
    ]
}
```

Tabel 3.15. Create Configuration Payroll

Relative Path	/api/v1/payroll-configuration
Method	POST

Body	<pre>{
	"employment_status_id": "xxxxxxxx",
	"name": "xxxxxxxx",
	"allowance_types": [
	"xxxxxxxx",
	"xxxxxxxx"
	]
	}</pre>

Success Response	HTTP Status Code 201
	<pre>{
	"code": 201,
	"message": "Successfully Created A Single Payroll",
	"data": {
	"id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
	"name": "Config Payroll Intern",
	"employment_status_id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
	"allowance_types": [
	{
	"id": "15c73d40-cea1-4ae5-9d00-97ea179bd8cd",
	"name": "THR"
	},
	{
	"id": "d52fbdff-f989-4774-91a1-ebe3f2edcf32",
	"name": "Makan"
	}
	]
	}
	}</pre>

Tabel 3.16. Update Configuration Payroll By Id

Relative Path	/api/v1/payroll-configuration/{id}
Method	PUT

Body	<pre>{
	  "name": "xxxxxxxx",
	  "employment_status_id": "xxxxxxxx",
	  "allowance_types": [
	  "xxxxxxxx",
	  "xxxxxxxx",
	  "xxxxxxxx"
	  ]
	}</pre>

Success Response	HTTP Status Code 200
	<pre>{
	  "code": 200,
	  "message": "Successfully Updated A Single Payroll",
	  "data": {
	  "id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
	  "name": "Config Pyaroll Intern",
	  "employment_status": {
	  "id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
	  "employment_type": "Internship"
	  },
	  "allowance_types": [
	  "THR",
	  "Makan",
	  "Tunjangan 1"
	  ]
	  }
	}</pre>

Tabel 3.17. Delete Configuration Payroll By Id

Relative Path	/api/v1/payroll-configuration/{payroll_configuration_id}
Method	DELETE

Success Response	HTTP Status Code 200
	<pre>{
	  "code": 200,
	  "message": "Successfully Deleted A Single Payroll"
	}</pre>

Tabel 3.18. Get Configuration Payroll Allowance

Relative Path	/api/v1/users/{user_id}/payroll-configuration
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Successfully Fetched All Users",
    "data": {
        "count": 1,
        "rows": [
            {
                "id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
                "name": "Config Pyaroll Intern"
            }
        ]
    }
}
```


Tabel 3.19. Get User Allowance By Id

Relative Path	/api/v1/user-allowance/{user_id}/{payroll_configuration_id}
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Successfully Fetched All User Allowance",
    "data": {
        "allowances": [
            {
                "allowance_type_id": "15c73d40-cea1-4ae5-9d00-97ea179bd8cd",
                "allowance_type_name": "THR",
                "is_custom": false,
                "is_deduction": false,
                "amount": 0
            },
            {
                "allowance_type_id": "d52fbdff-f989-4774-91a1-ebe3f2edcf32",
                "allowance_type_name": "Makan",
                "is_custom": false,
                "is_deduction": false,
                "amount": 0
            },
            {
                "allowance_type_id": "5cd18476-019b-486a-b905-ce9449627409",
                "allowance_type_name": "Tunjangan 1",
                "is_custom": false,
                "is_deduction": false,
                "amount": 0
            }
        ]
    }
}
```

Tabel 3.20. Get User Salary

Relative Path	/api/v1/users/{user_id}/salary
Method	GET

Success Response	HTTP Status Code 200
	{
	"code": 200,
	"message": "No active salary found for user",
	"amount": 0
	}

Tabel 3.21. Update User Payroll

Relative Path	/api/v1/‘users/{user_id}/with-details
Method	PATCH
Body	{ "userData": { "fullname": "xxxxxxxx", "email": "xxxxxxxx", "gender": "xxxxxxxx", "contact": "xxxxxxxx", "emergency_contact": "xxxxxxxx", "emergency_contact_holder": "xxxxxxxx", "job_title": "xxxxxxxx", "dob": "xxxxxxxx", "nik": "xxxxxxxx", "npwp": "xxxxxxxx", "employment_status_id": "xxxxxxxx", "supervisor_id": xxxxxxxx, "role_id": ["xxxxxxxx"], "bank_id": xxxxxxxx, "bank_number": "xxxxxxxx", "bank_holder": "xxxxxxxx", "bank_branch": "xxxxxxxx" },

Tabel 3.22. Update User Payroll

<table>
<tr><td></td><td>

```
"salary": "xxxxxxxx",
"allowances": {
  "payroll_configuration_id": "xxxxxxxx",
  "allowances": [
    {
      "allowance_type_id": "xxxxxxxx",
      "amount": xxxxxxxx
    },
    {
      "allowance_type_id": "xxxxxxxx",
      "amount": xxxxxxxx
    },
    {
      "allowance_type_id": "xxxxxxxx",
      "amount": xxxxxxxx
    },
    {
      "name": "xxxxxxxx",
      "amount": xxxxxxxx,
      "is_custom": xxxxxxxx,
      "is_deduction": xxxxxxxx
    }
  ]
}
}
```

</td></tr>
<tr><td>Success Response</td><td>HTTP Status Code 200

```
{
    "code": 200,
    "message": "User updated successfully with all information"
}
```

</td></tr>
</table>

Tabel 3.23. Delete User Allowance By Id

Relative Path	/api/v1/user-allowance/{allowance_type_id}
Method	DELETE

Success Response	HTTP Status Code 200
	<pre>{
	"code": 200,
	"message": "The allowance type has been deleted successfully."
	}</pre>

Tabel 3.24. Get All Salary Slip List

Relative Path	/api/v1/salary-slip/all?date={date}
Method	GET

Success Response	HTTP Status Code 200
	<pre>{
	"code": 200,
	"message": "Success",
	"data": {
	"count": 2,
	"rows": [
	{
	"user": {
	"id": "0179ff7d-c224-4bc6-ae1b-d0f55812e5d2",
	"fullname": "Muhammad Alwin Alamsyah Handoko Putra",
	"employment_status": {
	"id": "ad12375b-befa-4f29-8743-e1c765581cbf",
	"employment_type": "Full Time"
	}
	}
	},
	{
	"user": {
	"id": "0f9f0667-60d0-42a0-ba44-5849a621f254",
	"fullname": "Nicholas Christian",
	"employment_status": {
	"id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
	"employment_type": "Internship"
	}
	}
	}
	]
	}
	}</pre>

Tabel 3.25. Get Salary Slip List By Id

Relative Path	/api/v1/salary-slip/{user_id}?date={date}
Method	GET
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Salary slip generated successfully",
    "data": {
        "user_data": {
            "id": "0179ff7d-c224-4bc6-ae1b-d0f55812e5d2",
            "fullname": "Muhammad Alwin Alamsyah Handoko Putra",
            "is_active": true,
            "supervisor_id": null,
            "job_title": "Backend Engineer",
            "dob": "1995-06-10T00:00:00.000Z",
            "employment_status_id": "ad12375b-befa-4f29-8743-e1c765581cbf",
            "daily_status": null,
            "contact": "087859825943",
            "emergency_contact": "087880452809",
            "employee_number": "19",
            "email": "alwin.alamsyah@concise.co.id",
            "nik": "3275091006950010",
            "npwp": "866936909407000",
            "gender": "male",
            "bank_id": null,
            "bank_name": null,
            "bank_number": "8831524839",
            "bank_branch": null,
            "bank_holder": "Muhammad Alwin Alamsyah Handoko Putra",
            "emergency_contact_holder": "Muhammad Bintang Ramadhan Handoko Putra",
            "active_payroll_id": "a2dd9953-e3b6-447b-80b5-31481613fd3f",
            "employment_status": {
                "id": "ad12375b-befa-4f29-8743-e1c765581cbf",
                "employment_type": "Full Time"
            },
            "bank": null,
            "salary": [
                {
                    "amount": "2000000",
                    "created_at": "2025-06-16T04:18:15.846Z"
                }
            ],
            "payroll_application": [
                {
                    "id": "e16c8fce-5cd7-4850-b64b-9332a6a21e6c",
                    "payroll_configuration_id": "a2dd9953-e3b6-447b-80b5-31481613fd3f"
                }
            ]
        },
        "allowances": [
            {
                "name": "THR",
                "amount": 2000000,
                "is_deduction": false
            },
            {
                "name": "Makan",
                "amount": 500000,
                "is_deduction": false
            },
            {
                "name": "Project Bonus",
                "amount": 100000,
                "is_deduction": false
            }
        ],
        "salary_data": {
            "basic_salary": 2000000,
            "total_allowances": 4600000,
            "total_deductions": 0,
            "take_home_pay": 4600000
        },
        "period": {
            "start_date": "2025-05-25T00:00:00.000Z",
            "end_date": "2025-06-24T23:59:59.999Z"
        },
        "date": "June 2025"
    }
}
```

Tabel 3.26. Get Salary Slip Self

Relative Path	/api/v1/salary-slip?date={date}
Method	GET
Query Params	date
Success Response	HTTP Status Code 200


```
{
    "code": 200,
    "message": "Salary slip generated successfully",
    "data": {
        "user_data": {
            "id": "0f9f0667-60d0-42a0-ba44-5849a621f254",
            "fullname": "Nicholas Christian",
            "is_active": true,
            "supervisor_id": null,
            "job_title": "Backend Engineer",
            "dob": "2003-04-18T00:00:00.000Z",
            "employment_status_id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
            "daily_status": null,
            "contact": "085214910973",
            "emergency_contact": "081399971979",
            "employee_number": "6",
            "email": "nicholas.christian@concise.co.id",
            "nik": "3173011804031000",
            "npwp": "654588516034000",
            "gender": "male",
            "bank_id": null,
            "bank_name": null,
            "bank_number": "3500034954",
            "bank_branch": "KCP BCA Kosambi Baru",
            "bank_holder": "Nicholas Christian",
            "emergency_contact_holder": "Lidiana",
            "active_payroll_id": "4d94ba99-13c8-42c0-94b6-bde66cc93276",
            "employment_status": {
                "id": "8a82bcbf-18d8-41ae-94a5-b3cd6b91590b",
                "employment_type": "Internship"
            },
            "bank": null,
            "salary": [
                {
                    "amount": "2000000",
                    "created_at": "2025-06-12T08:37:38.640Z"
                }
            ],
            "payroll_application": [
                {
                    "id": "5ae86cc6-a4c2-46ad-a7f3-890f1433ec68",
                    "payroll_configuration_id": "4d94ba99-13c8-42c0-94b6-bde66cc93276"
                }
            ]
        },
        "allowances": [
            {
                "name": "THR",
                "amount": 2000000,
                "is_deduction": false
            },
            {
                "name": "Makan",
                "amount": 500000,
                "is_deduction": false
            },
            {
                "name": "Tunjangan 1",
                "amount": 100000,
                "is_deduction": false
            },
            {
                "name": "Project Bonus",
                "amount": 750000,
                "is_deduction": false
            }
        ],
        "salary_data": {
            "basic_salary": 2000000,
            "total_allowances": 5350000,
            "total_deductions": 0,
            "take_home_pay": 5350000
        },
        "period": {
            "start_date": "2025-05-25T00:00:00.000Z",
            "end_date": "2025-06-24T23:59:59.999Z"
        },
        "date": "June 2025"
    }
}
```

Tabel 3.27. Error Responses

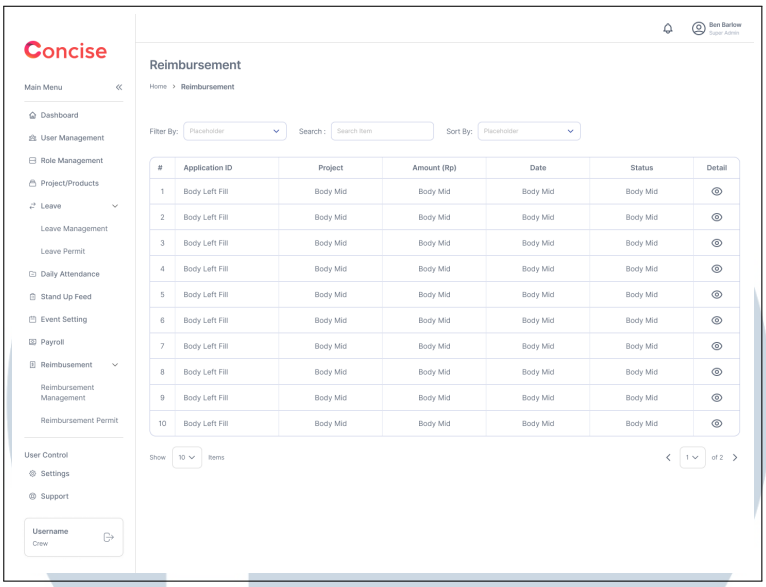
400	HTTP Status Code 400 (Bad Request) `{ "status": 400, "message": "Missing required fields: date (YYYY-MM)" }`
401	HTTP Status Code 401 (Unauthorized) `{ "code": 401, "message": "Please Authenticate" }`
403	HTTP Status Code 403 (Forbidden) `{ "code": 403, "message": "You Have No Permission" }`
404	HTTP Status Code 404 (Not Found) `{ "code": 404, "message": "Not found" }`

3.5.1 Wireframe

Wireframe adalah kerangka dasar yang merepresentasikan visual dari antarmuka aplikasi[11]. *Wireframe* terbagi menjadi 2, yaitu *wireframe low fidelity* dan *wireframe high fidelity*. *Wireframe low fidelity* adalah kerangka dasar yang dirancang untuk memberikan gambaran yang paling sederhana [12]. Sedangkan *wireframe high fidelity* adalah desain antarmuka yang lebih detail, lengkap, dan realistis [13]. Selama proses pengembangan sistem, *wireframe* dikerjakan oleh tim desain dengan menggunakan *tools* berupa Figma.

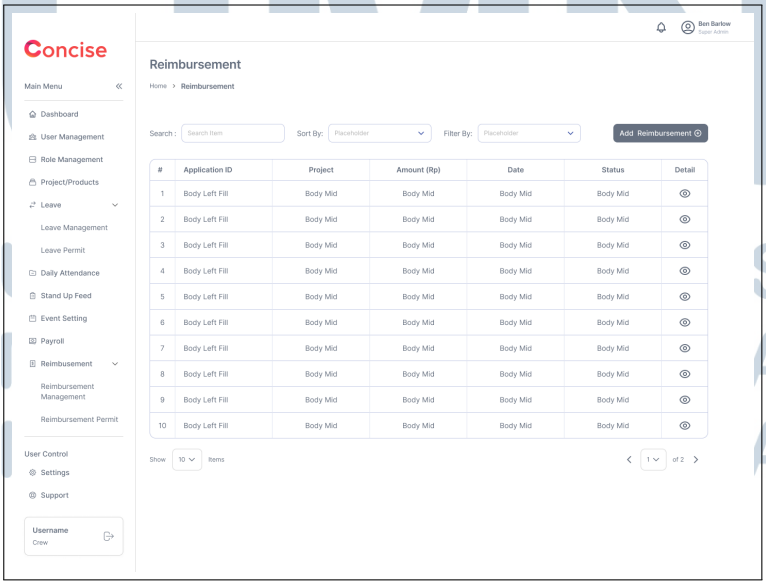
Gambar 3.18 adalah gambar *wireframe* pada halaman *Reimbursement Management*. Terdapat tabel yang terdiri dari *Application ID*, *Name*, *Project*, *Amount* (Rp), *Date*, Status, dan Detail. Terdapat fitur untuk melakukan pencarian berdasarkan *Application ID* atau *Name*, pengurutan secara *ascending* dan *descending*, dan filter berdasarkan Status. Terdapat simbol mata untuk melihat

detail dari *reimbursement* yang dipilih.



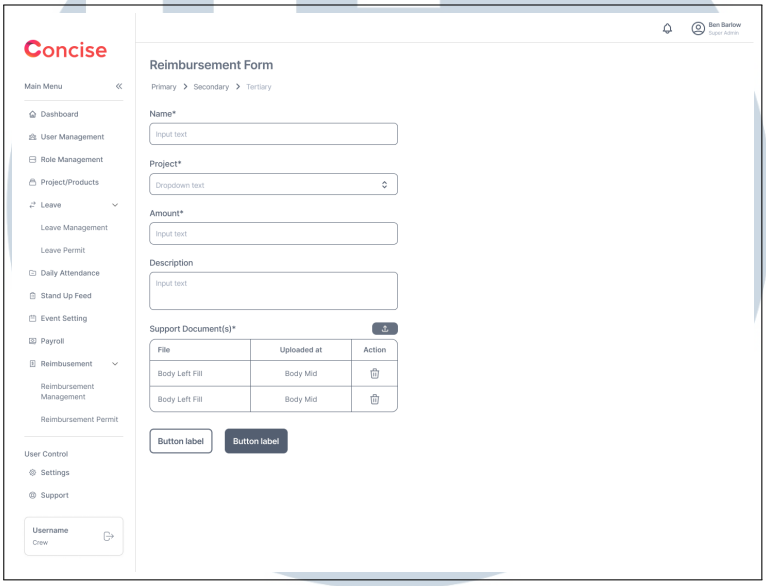
Gambar 3.18. Wireframe Reimbursement Management

Gambar 3.19 adalah gambar *wireframe* pada halaman *Reimbursement Permit*. Terdapat tabel yang terdiri dari *Application ID*, *Project*, *Amount* (Rp), *Date*, Status, dan Detail. Terdapat fitur untuk melakukan pencarian berdasarkan *Application ID*, pengurutan secara *ascending* dan *descending*, dan filter berdasarkan Status. Terdapat tombol *Add Reimbursement* untuk menambahkan *reimbursement* baru. Terdapat simbol mata untuk melihat detail dari *reimbursement* yang dipilih.



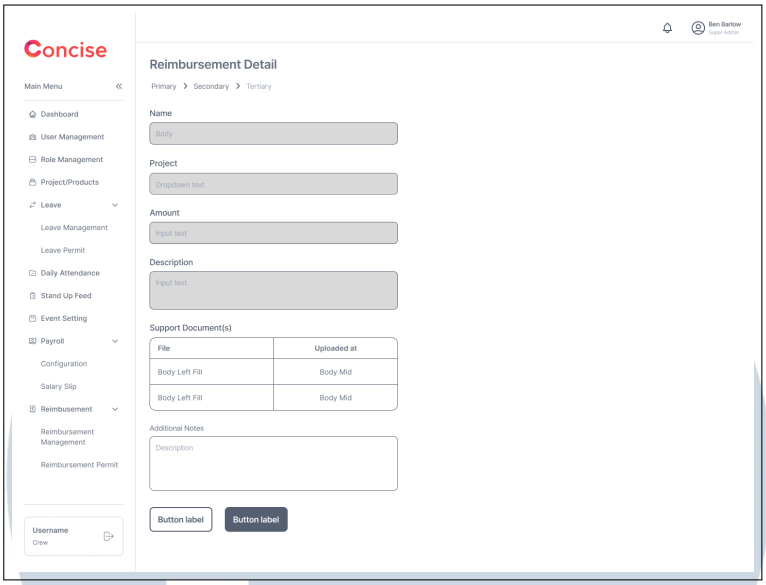
Gambar 3.19. Wireframe Reimbursement Permit

Gambar 3.20 adalah gambar *wireframe* pada halaman *Reimbursement Input*. Terdapat beberapa *form field*, yaitu *Name*, *Project*, *Amount*, dan *Description*, serta terdapat tabel *files*. *Files* dapat diunggah dengan menekan tombol dengan simbol *upload* di sebelah kanan tulisan *Support Document(s)*. Terdapat dua tombol di bawah tabel, yaitu tombol *submit* dan *cancel*.



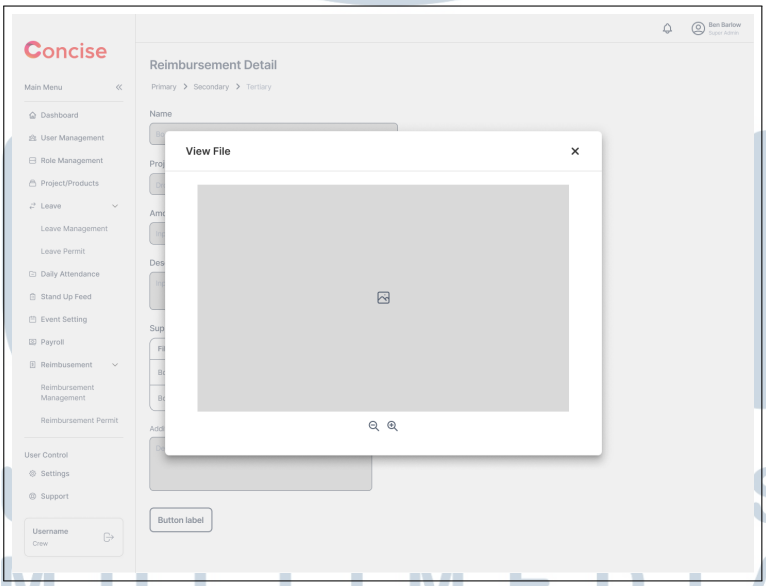
Gambar 3.20. Wireframe Reimbursement Input

Gambar 3.21 adalah gambar *wireframe review* pada halaman *Reimbursement Input*. Jika *form field* telah diisi, maka *form field* akan berubah warna menjadi warna abu-abu. Kemudian, terdapat *form field* untuk menambahkan *notes* pada saat proses *review*. Terdapat dua tombol di bawah *form field notes*, yaitu tombol *Reject* dan *Accept*.



Gambar 3.21. Wireframe Review Reimbursement Input

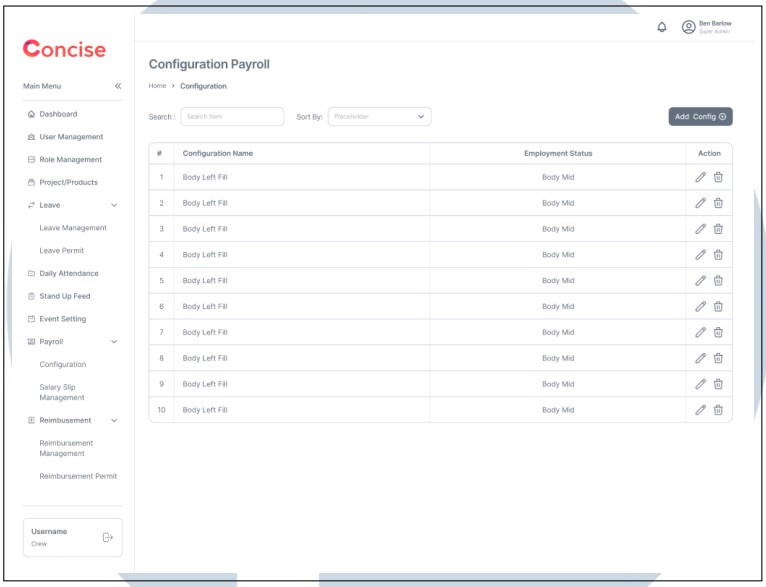
Gambar 3.22 adalah gambar *wireframe view file* pada halaman *Reimbursement Input*. Terdapat modal yang muncul ketika melihat file yang dipilih. *User* dapat memperbesar dan memperkecil file.



Gambar 3.22. Wireframe View File Reimbursement Input

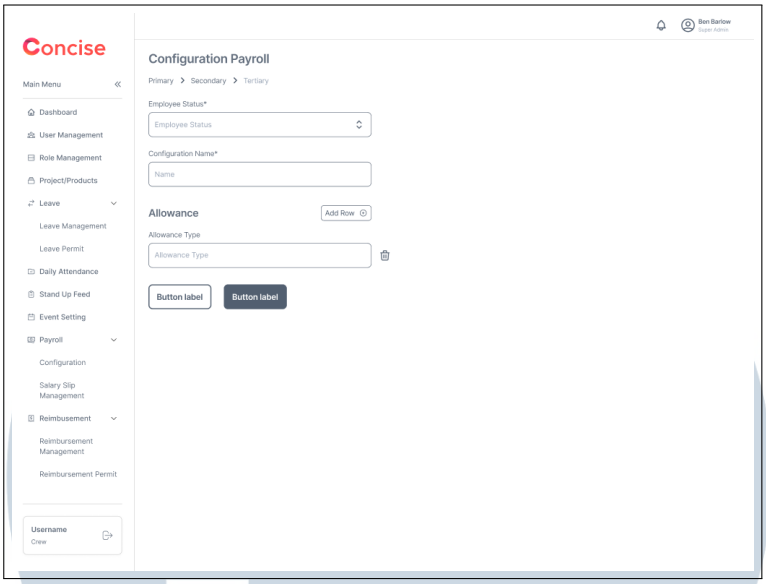
Gambar 3.23 adalah gambar *wireframe* pada halaman *Configuration Payroll*. Terdapat tabel yang terdiri dari *Configuration Name*, *Employee Status*, dan *Action*. Terdapat fitur untuk melakukan pencarian berdasarkan *Configuration Name* atau *Employee Status* dan pengurutan secara *ascending* dan *descending* untuk

Configuration Name atau *Employee Status*. Terdapat simbol pensil untuk mengedit *configuration* yang dipilih serta terdapat simbol tempat sampah untuk menghapus *configuration*.



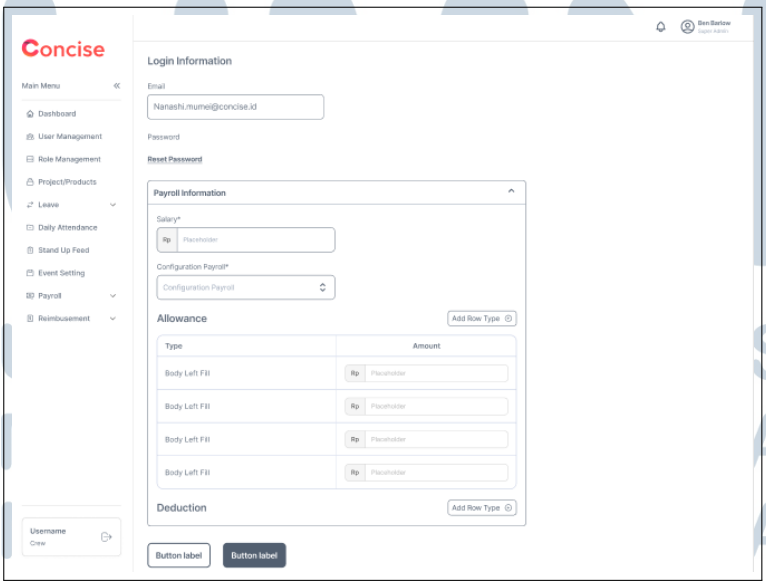
Gambar 3.23. Wireframe Configuration Payroll

Gambar 3.24 adalah gambar *wireframe* pada halaman *Configuration Payroll Input*. Terdapat beberapa *form field*, yaitu *Employee Status*, *Configuration Name*, dan *Allowance Type*. Menambahkan *Allowance Type* baru dengan menekan tombol *Add Allowance*. Terdapat dua tombol di bawah *form field Allowance Type*, yaitu tombol *Cancel* dan *Submit*.



Gambar 3.24. Wireframe Configuration Payroll Input

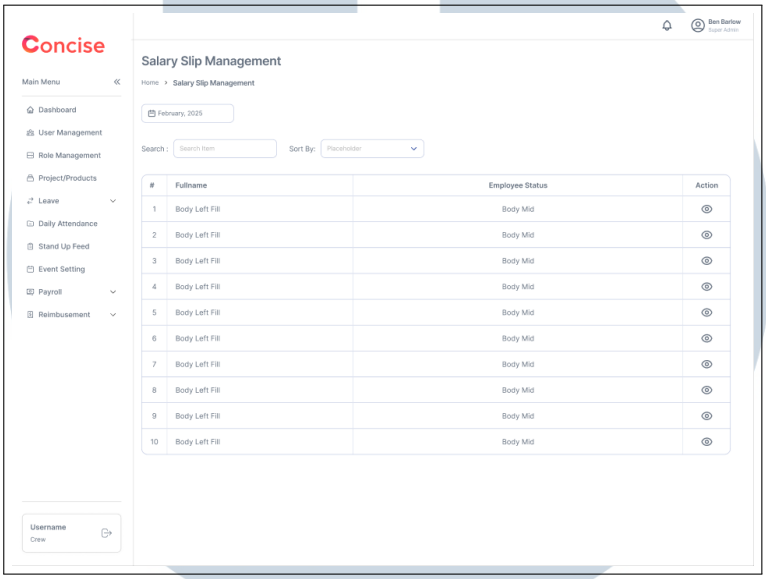
Gambar 3.25 adalah gambar *wireframe Payroll Information* pada halaman *Edit User* yang terdiri dari *Salary*, *Configuration Payroll*, *Allowances*, dan *Deductions*. Terdapat tombol *Add Allowance* untuk menambahkan *allowance* serta terdapat tombol *Add Deduction* untuk menambahkan *Deduction* baru khusus untuk *user* yang sedang diedit. *User* yang memiliki *salary slip* dapat mengunduh *salary slip* dengan menekan tombol *Download Salary Slip* yang berada di kanan atas.



Gambar 3.25. Wireframe Payroll Information

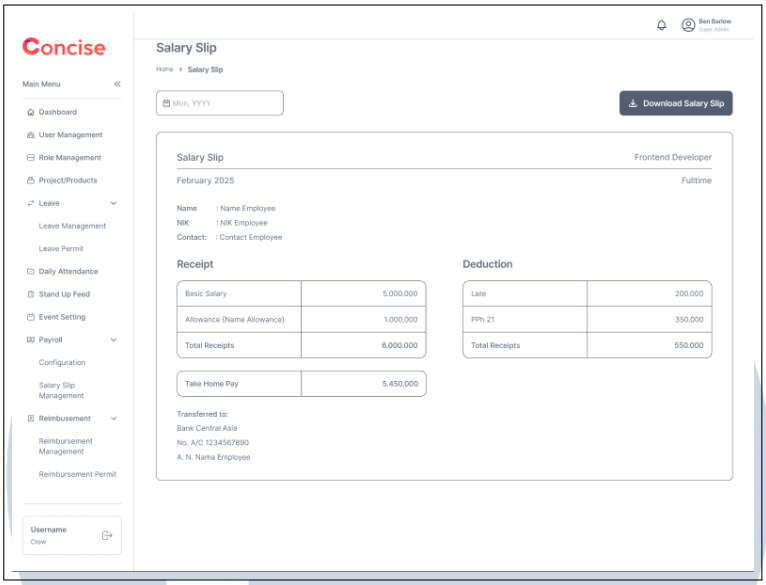
Gambar 3.26 adalah gambar *wireframe* pada halaman *Salary Slip List*.

Terdapat tabel yang terdiri dari *Fullname*, *Employee Status*, dan *Action*. Terdapat fitur untuk melakukan pencarian berdasarkan *Fullname* atau *Employee Status* dan pengurutan secara *ascending* dan *descending* untuk *Fullname* atau *Employee Status*. Terdapat simbol mata untuk melihat detail dari *salary slip* yang dipilih.



Gambar 3.26. Wireframe Salary Slip List

Gambar 3.27 adalah gambar *wireframe* pada halaman *Salary Slip* yang terdiri dari *Job Title*, *Employment Type*, *Fullname*, NIK, *Contact*, *Basic Salary*, *Allowances*, *Total Receipt*, *Deductions*, *Total Deduction*, *Take Home Pay*, *Bank Name*, dan nomor rekening pemilik *salary slip*. Terdapat fitur untuk memilih bulan dan atau tahun yang diinginkan untuk melihat *salary slip* berdasarkan bulan dan atau tahun tersebut. *User* yang memiliki *salary slip* dapat mengunduh *salary slip* dengan menekan tombol *Download Salary Slip* yang berada di kanan atas.

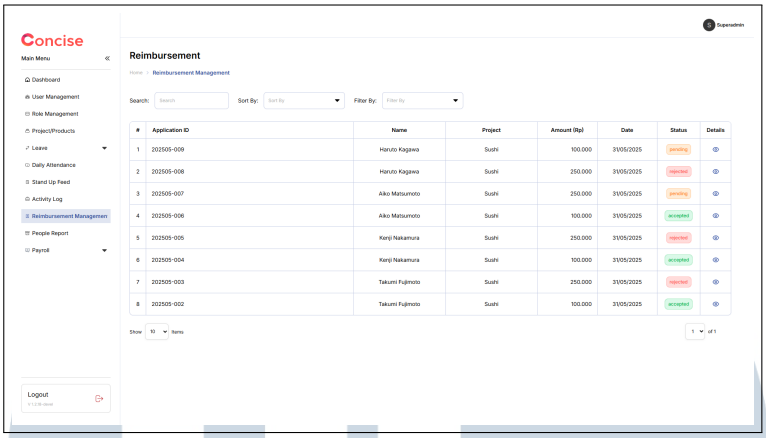


Gambar 3.27. Wireframe Salary Slip

3.6 Implementasi

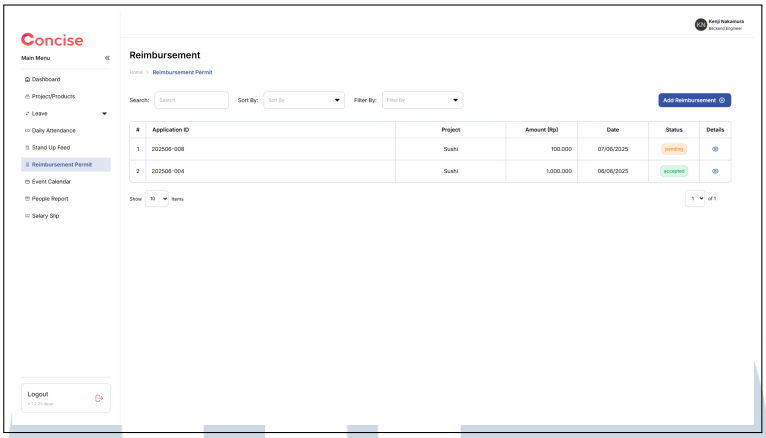
Pengembangan modul *Reimbursement Human Resource Internal System* telah diimplementasikan di *mode production* dan telah digunakan oleh semua karyawan pada PT Ganda Visi Jayatama. Berikut merupakan hasil akhir modul *Reimbursement* HRIS yang telah selesai dikembangkan.

Pada halaman *Reimbursement Management*, terdapat daftar semua *reimbursement*. Daftar *reimbursement* muncul setelah proses GET API ke *endpoint* ”reimbursement/all” dengan mengirimkan params berupa *page*, *row*, *pagination*, *search* jika ada, *sort* jika ada, dan *filter* jika ada. Setelah mendapatkan respons, data diambil dengan memasukkan *array* yang berisi daftar *reimbursement* yaitu res.data.data.rows ke dalam suatu variabel yaitu reimbursements. Data disimpan ke dalam *state* dengan setReimbursementData(reimbursements). Kemudian, terdapat simbol mata pada tabel yang berfungsi untuk melihat secara detail dan berfungsi untuk melakukan *update reimbursement* yang dipilih, jika memiliki *permission update*. *User* dapat mencari *Application ID*, *Name*, dan *Project* dengan fitur *search*. *User* juga dapat melakukan *sort* terhadap tabel *reimbursement* berdasarkan *Application ID* secara *ascending* atau mengurutkan data dari terkecil ke terbesar maupun *descending* atau mengurutkan data dari terbesar ke terkecil. Selain itu, *user* juga dapat melakukan filter berdasarkan status, antara lain *pending*, *accept*, atau *reject*.



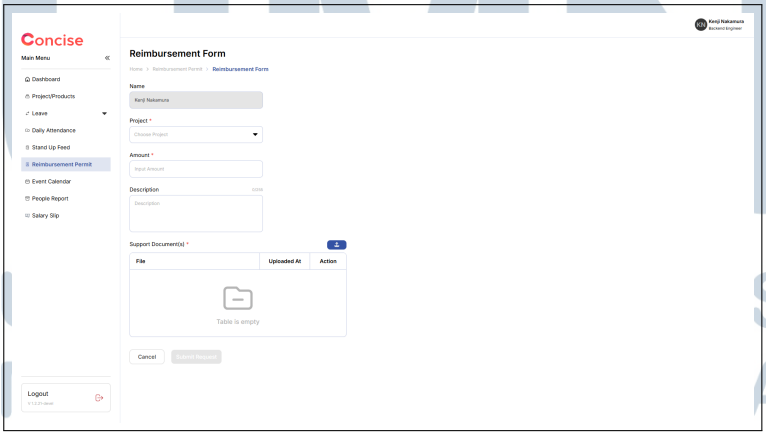
Gambar 3.28. Tampilan Reimbursement Management

Pada halaman *Reimbursement Permit* terdapat daftar semua *reimbursement*. Daftar *reimbursement* muncul setelah proses GET API ke *endpoint* ”reimbursement” dengan mengirimkan params berupa *page*, *row*, *pagination*, *search* jika ada, *sort* jika ada, dan *filter* jika ada. Setelah mendapatkan respons, data diambil dengan memasukkan *array* yang berisi daftar *reimbursement* yaitu res.data.data.rows ke dalam suatu variabel yaitu reimbursements. Data disimpan ke dalam *state* dengan setReimbursementData(reimbursements). Kemudian, *user* dapat menambahkan *reimbursement* baru dengan menekan tombol *add reimbursement* di kanan atas. Selain itu, terdapat simbol mata pada tabel yang berfungsi untuk melihat secara detail dan berfungsi untuk melakukan *delete reimbursement* yang dipilih, jika memiliki *permission delete*. *User* dapat mencari *Application ID* dan *Project* dengan fitur *search*. *User* juga dapat melakukan *sort* terhadap tabel *reimbursement* berdasarkan *Application ID* secara *ascending* maupun *descending*. Selain itu, *user* juga dapat melakukan filter berdasarkan status, antara lain *pending*, *accept*, atau *reject*.



Gambar 3.29. Tampilan Reimbursement Permit

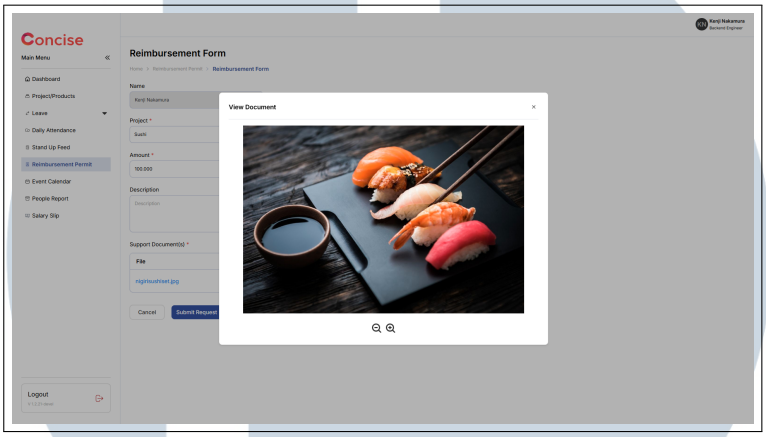
Pada halaman *Reimbursement Input*, *user* dapat mengisi data-data yang wajib diisi seperti *Fullname*, *Project*, *Amount*, dan *Support Document(s)*. *Support Document(s)* dapat diunggah dengan menekan tombol dengan simbol *upload* yang berada pada kanan tulisan *Support Document(s)*. *Files* yang diunggah akan dikirimkan dalam bentuk *InputData* berupa *array* dan parameter *reimbursement* dengan metode POST API melalui *endpoint* "file/upload". Kemudian, pada halaman *Reimbursement Input*, *user* dapat melakukan *delete reimbursement* yang telah diisi dan dipilih. Selain itu, *user* yang memiliki *permission update*, dapat melakukan *update reimbursement* dengan penambahan *notes* jika diperlukan. *Update* tersebut meliputi tombol yang dapat dipilih, yaitu *Reject* dan *Accept*.



Gambar 3.30. Tampilan Reimbursement Input

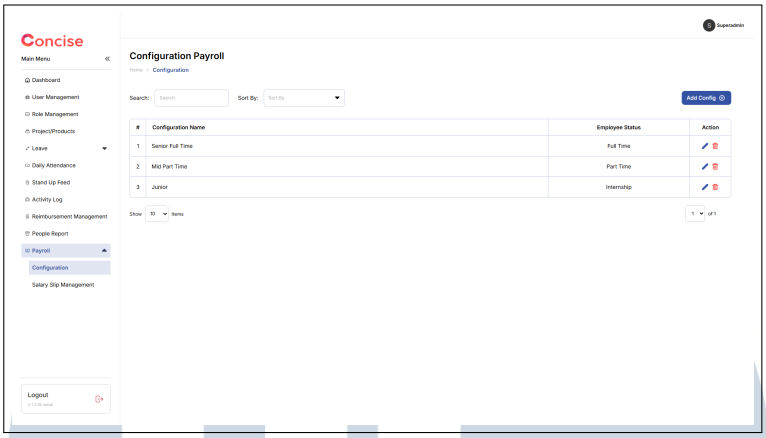
Pada halaman *Reimbursement Input*, *user* dapat melihat *file* yang telah diunggah pada tabel *Support Document(s)*. Di dalam tabel *Support Document(s)* terdapat simbol tempat sampah yang berfungsi untuk menghapus *file* pada baris

yang dipilih. Untuk melihat *file* secara detail, *user* dapat menekan nama *file* yang telah diunggah. Kemudian, akan menampilkan modal yang berisi *file* yang telah dipilih. *User* juga dapat memperbesar atau memperkecil file dengan menekan tombol dengan simbol *zoom in* atau *zoom out*.



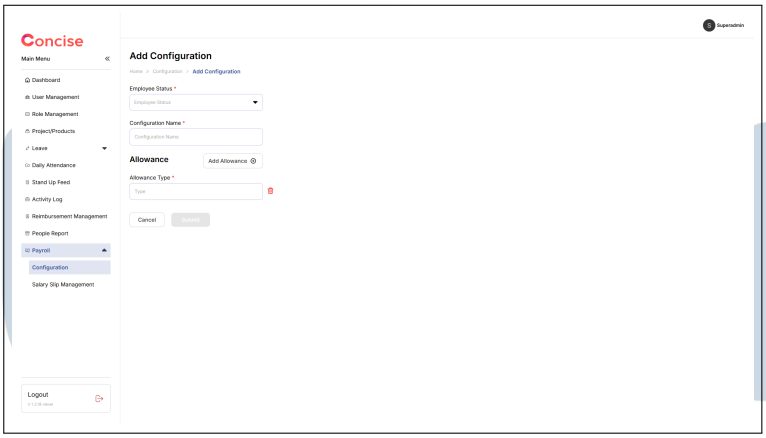
Gambar 3.31. Tampilan View File Pada Halaman Reimbursement Input

Pada halaman *Configuration Payroll*, terdapat daftar semua *configuration*. Daftar *configuration* muncul setelah proses GET API ke *endpoint* "payroll-configuration/all" dengan mengirimkan params berupa *page*, *row*, *pagination*, *search* jika ada, dan *sort* jika ada. Setelah mendapatkan respons, data diambil dengan memasukkan *array* yang berisi daftar *configuration* yaitu res.data.data.rows ke dalam suatu variabel yaitu configurations. Data disimpan ke dalam *state* dengan setPayrollConfigurationData(configurations). Kemudian, terdapat simbol pensil pada tabel yang berfungsi untuk melihat secara detail dan berfungsi untuk melakukan *update configuration* yang dipilih, jika memiliki *permission update*. Selain itu, terdapat simbol tempat sampah pada tabel yang berfungsi untuk menghapus *configuration* yang dipilih. *User* dapat mencari *Configuration Name* dan *Employee Status* dengan fitur *search*. *User* juga dapat melakukan *sort* terhadap tabel *configuration* berdasarkan *Configuration Name* atau *Employee Status* secara *ascending* maupun *descending*.



Gambar 3.32. Tampilan Configuration Payroll

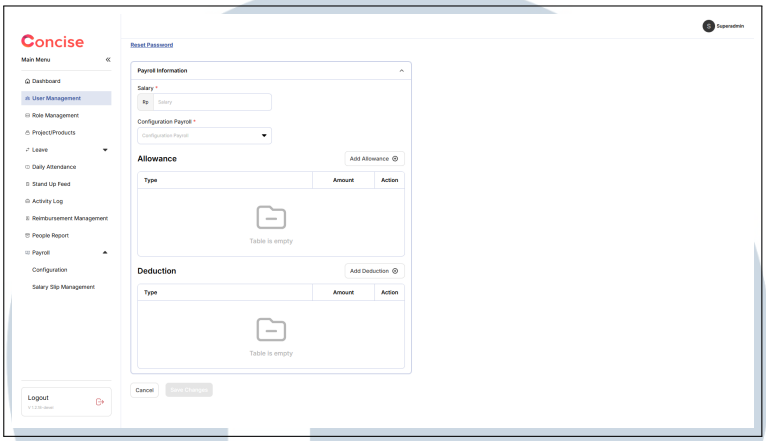
Pada halaman *Configuration Payroll Input*, *user* dapat mengisi data-data yang wajib diisi seperti *Employee Status*, *Configuration Name*, dan *Allowance Type*. *Allowance Type* dapat ditambah dengan menekan tombol *Add Allowance* yang berada pada kanan tulisan *Allowance*. Jika data sudah diisi semua, maka *user* dapat menekan tombol *Submit*. Data akan dikirim dengan metode POST API melalui *endpoint* ”payroll-configuration”.



Gambar 3.33. Tampilan Configuration Payroll Input

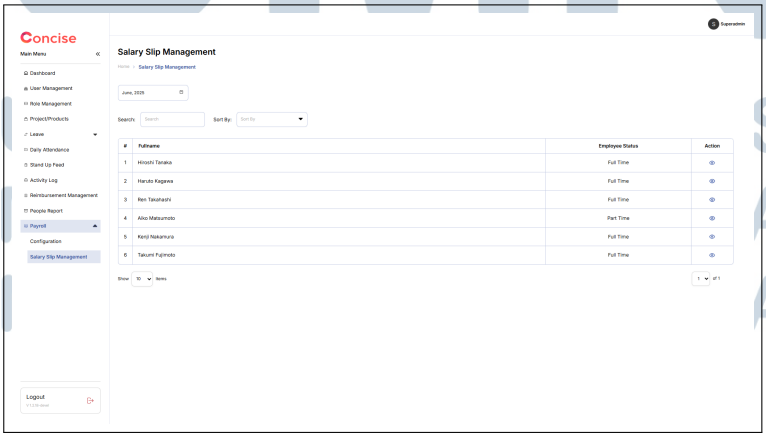
Pada halaman *User Input* terdapat penambahan *Payroll Information* untuk *update user*, *user* dapat mengisi data-data yang wajib diisi, jika *accordion* ditekan atau kondisi terbuka antara lain *Salary*, *Configuration Name*, dan *Allowance Type*. Sedangkan *Deduction* bersifat tidak wajib diisi. *Allowance Type* dapat ditambah dengan menekan tombol *Add Allowance* yang berada pada kanan tulisan *Allowance*. Jika data sudah diisi semua, maka *user* dapat menekan tombol *Save Changes*. Data

akan dikirim dengan metode PATCH API melalui *endpoint* ”users/{user_id}/with-details”.



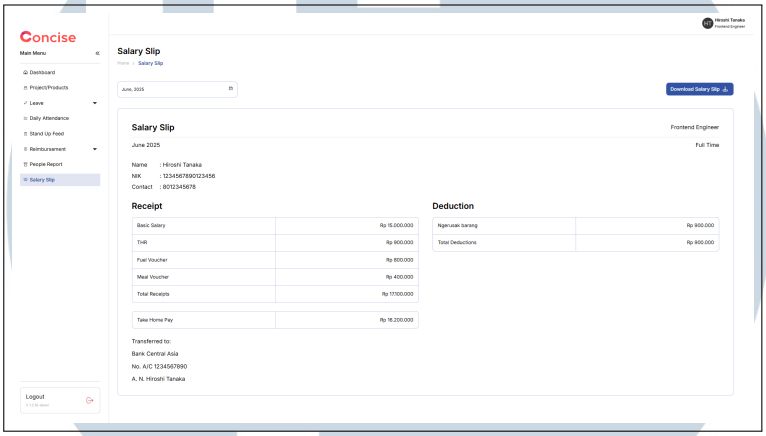
Gambar 3.34. Tampilan Payroll Information Pada Halaman User Input

Pada halaman *Salary Slip List*, terdapat daftar semua *salary slip*. Daftar *salary slip* muncul setelah proses GET API ke *endpoint* ”salary-slip/all?date={date}” dengan mengirimkan params berupa *date*, *page*, *row*, *pagination*, *search* jika ada, dan *sort* jika ada. Setelah mendapatkan respons, data diambil dengan memasukkan *array* yang berisi daftar *salary slip* yaitu res.data.data.rows ke dalam suatu variabel yaitu slip. Data disimpan ke dalam *state* dengan setSalarySlipData(slip). Kemudian, terdapat simbol mata pada tabel yang berfungsi untuk melihat secara detail. *User* dapat mencari *Fullname* dan *Employee Status* dengan fitur *search*. *User* juga dapat melakukan *sort* terhadap tabel *Salary Slip List* berdasarkan *Fullname* atau *Employee Status* secara *ascending* maupun *descending*.



Gambar 3.35. Tampilan Salary Slip List

Pada halaman *Salary Slip*, *user* dapat memilih bulan dan atau tahun untuk melihat data-data salary slip di bulan tersebut seperti *job title*, *employee status*, *name*, NIK, *Contact*, *Basic Salary*, *Allowance*, *Deduction*, *Take Home Pay*, *Bank Name*, dan *Bank Account Number*. *User* dapat mengunduh dengan menekan tombol *Download Salary Slip* yang berada pada kanan atas.



Gambar 3.36. Tampilan Halaman Salary Slip User

3.7 Kendala dan Solusi yang Ditemukan

Beberapa kendala yang dialami selama melakukan magang di PT Ganda Visi Jayatama sebagai berikut.

1. Kurangnya pengetahuan terhadap *boilerplate* perusahaan atau kode pemrograman yang dapat dipakai berulang.
2. Kurangnya komunikasi antara tim backend dan tim *frontend* pada awal pelaksanaan kerja magang, sehingga terjadi konflik selama pengerjaan yang memperlambat pengerjaan.
3. Kesulitan dalam memahami logika kode yang sudah ada pada sistem HRIS.

Solusi yang ditemukan untuk menghadapi masalah tersebut sebagai berikut.

1. Memerlukan waktu untuk mempelajari *boilerplate* perusahaan, sebelum memulai pengembangan fitur-fitur pada HRIS.
2. Meningkatkan komunikasi antara tim *backend* dan *frontend* setiap melakukan perubahan yang akan menimbulkan konflik.

3. Bertanya kepada senior yang lebih berpengalaman untuk membantu proses pengembangan.