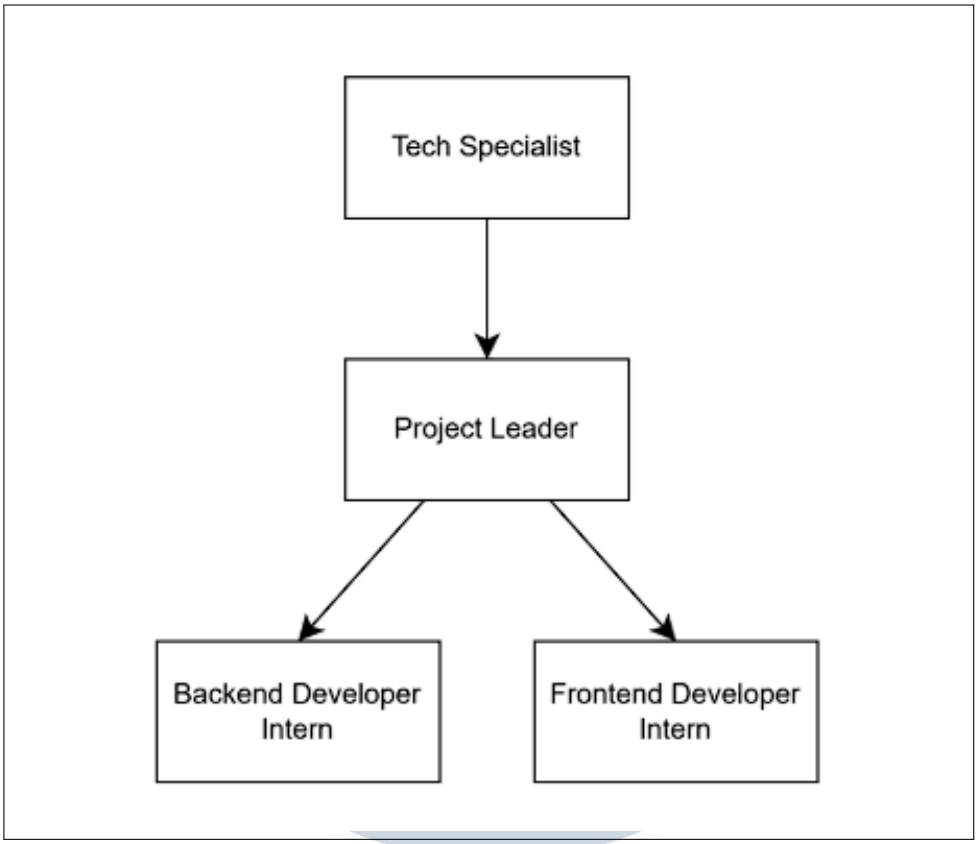
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama magang, posisi yang dipegang sebagai IT associate intern yang memiliki fokus ke backend developer dalam proyek berbasis ERP. Selama periode tersebut, bimbingan dan pengawasan dilakukan oleh Bapak Robert Theo dan dilanjut oleh Bapak Damar Sinatrio yang menjabat sebagai Project Leader . Tugas utama tim meliputi perancangan dan pembangunan website ERP-Backslash dengan fitur manajemen aset dan pengajuan reimbursement sesuai dengan kebutuhan perusahaan menggunakan PHP dengan *framework* Laravel. Koordinasi terkait progres pekerjaan dilakukan melalui pertemuan langsung di kantor atau melalui WhatsApp jika ada hal mendesak yang perlu diklarifikasi. Setiap minggu, hasil kemajuan proyek disampaikan kepada supervisor untuk mendapatkan evaluasi dan arahan lebih lanjut.

Pelaksanaan kerja magang terdapat struktur tim proyek *website* untuk memperlancar koordinasi dan komunikasi selama pelaksanaan magang dengan perancangan dan pembangunan *website* ERP-Backslash pada PT Backslash Creative Nusantara dapat dilihat pada Gambar 2.2.



Gambar 3.1. Struktur tim proyek *website*

Berdasarkan Gambar 2.2, struktur tim proyek *website* dalam pengembangan *website* ERP-Backslash di PT Backslash Creative Nusantara. Tim proyek terdiri dari Tech Specialist yang bertanggung jawab untuk mengarahkan dan mengawasi progres proyek, *Project Leader* memiliki peran untuk membantu koordinasi dan pengelolaan tugas proyek, *Frontend Developer Intern* berperan dalam implementasi tampilan *layout* dan UI, serta *Backend Developer Intern* berperan dalam *application logic*, *database*, dan API. Pembagian peran ini bertujuan untuk memperjelas tugas masing - masing, koordinasi, dan memastikan pengembangan proyek berjalan sesuai target.

3.2 Tugas yang Dilakukan

Dalam periode magang di PT Backslash Creative Nusantara, tugas utama yang diberikan termasuk perancangan dan pembangunan website ERP-Backslash pada PT Backslash Creative Nusantara. Pembuatan website ini didasarkan pada alasan bahwa pengelolaan aset dan *reimbursement* sebelumnya dilakukan secara manual dengan menggunakan dokumen fisik. Proses manual ini memiliki berbagai

kelemahan, seperti risiko kesalahan pencatatan, lambatnya proses verifikasi, dan kurangnya transparansi dalam pengelolaan data. Dalam konteks ini, diperlukan solusi berbasis teknologi yang mampu mengatasi kendala tersebut dan mendukung efisiensi operasional perusahaan. Tujuan dari pengembangan sistem ini adalah untuk meningkatkan efisiensi dalam pengelolaan aset dan *reimbursement* dengan meminimalkan intervensi manual, mengurangi risiko kesalahan pencatatan, dan memastikan keamanan data melalui pengaturan hak akses berbasis peran. Selain itu, sistem ini dirancang untuk mempermudah karyawan dalam mengajukan *reimbursement* melalui proses yang lebih transparan dan terintegrasi.

Perancangan sistem website ERP-Backslash dibagi menjadi dua bagian utama, yaitu *frontend* dan *backend*. *Frontend* berfungsi sebagai antarmuka yang terlihat dan digunakan oleh pengguna, sementara *backend* berperan dalam pengelolaan pemrosesan serta penyimpanan data di website. Pemisahan kedua bagian ini bertujuan untuk membuat pengembangan sistem lebih terorganisir, di mana *frontend* berfokus pada tampilan serta pengalaman pengguna, sedangkan *backend* lebih menitikberatkan pada aspek keamanan dan manajemen data. Dengan arsitektur yang terpisah, kedua komponen ini dapat dikembangkan secara mandiri sesuai dengan kebutuhan masing-masing.

3.3 Uraian Pelaksanaan Magang

Proses kerja mengikuti siklus pengembangan mingguan yang terstruktur berdasarkan prioritas proyek dan kebutuhan sistem. pleaksanaan kerja magang minggu pertama hingga ke enam fokus dalam mengautomasi proses ekstrasi data dari file

Pelaksanaan kerja magang diuraikan seperti pada Tabel **??**.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Orientasi dengan perusahaan, memahami WIP Report, mulai pemindahan data ke spreadsheet dari query SQL, eksplorasi otomasi dengan Python, serta konversi file .rpt ke .xml.
2	Melengkapi file WIP Report, ekstraksi data dari file .mrt dengan bantuan Stimulsoft dan ChatGPT, memindahkan 2.268 data dari aplikasi web ke spreadsheet, serta *scraping* data dari folder Scriptcase menggunakan Python dan PowerShell.
3	Penggabungan worksheet data Scriptcase dan Native, diskusi cara pengisian kolom purpose, serta analisis data aplikasi web native melalui eksplorasi file PHP dan koneksi basis data.
4	Melanjutkan dokumentasi purpose job Scriptcase, pengecekan aplikasi native yang berada di folder Scriptcase, *web scraping* untuk mengetahui tabel dan basis data dari setiap modul, serta penggabungan data WIP, Scriptcase, Stimulsoft, dan lainnya.
5	Pemindahan data dari file VisualCron, melengkapi kolom kosong di Excel, melakukan pencocokan silang serta pembersihan worksheet, dan dokumentasi purpose dari aplikasi native per modul.
6	Menambahkan dan melengkapi kolom purpose dan mark di berbagai file (Stimulsoft, WIP, Scriptcase_PCT), finalisasi file, serta pemindahan hasil akhir ke situs *Knowledge Base* PT Paul Buana Indonesia.
7	*Briefing* proyek yang akan dikerjakan. Mengerjakan fitur login, termasuk autentikasi, otorisasi, dan *testing* awal.
8	Melanjutkan fitur login dan mengimplementasikan fitur logout serta *redirect*. Melakukan *testing* fitur login dan penggantian password.
9	Mengimplementasikan fitur manajemen pengguna (*read*, *create*, *delete*, *edit*) serta melakukan *testing*.
10	Mengimplementasikan fitur pencatatan inventaris.
11	Menyempurnakan dan menguji fitur inventaris.

12	Mengerjakan fitur pengajuan reimbursement.
13	Melakukan revisi pada fitur reimbursement.
14	Menyelesaikan dan menguji fitur reimbursement. Melakukan pengujian menyeluruh untuk proses pengajuan dan persetujuan.
15	Melakukan revisi dan penyempurnaan pada fitur asset dan reimbursement.
16	Melakukan *testing* akhir pada fitur asset dan reimbursement, serta menyampaikan presentasi akhir proyek website.

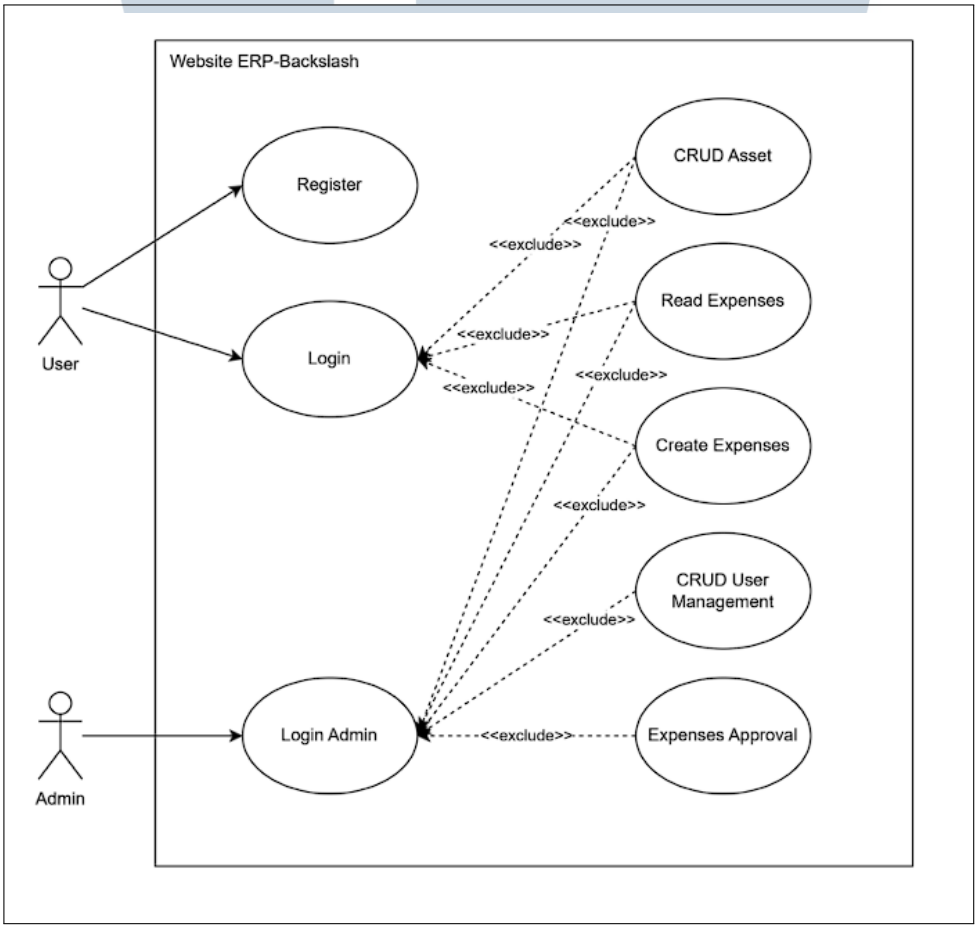
3.3.1 Komponen

Dalam sistem ERP-Backslash, setiap modul memiliki fungsi spesifik yang mendukung operasional sistem. Modul *Login* bertugas untuk mengautentikasi pengguna terdaftar dengan memverifikasi email dan *password* terhadap *database*. Jika berhasil, sistem akan menghasilkan *Session ID* dan *Token* untuk menjaga keamanan sesi dan autentikasi pengguna. Selanjutnya, modul *Register* memungkinkan pengguna baru untuk mendaftar dengan memasukkan data seperti nama, email, dan *password* yang divalidasi oleh sistem. Akun baru kemudian dibuat di *database* dengan peran *default* sebagai pengguna.

Modul *User Management*, yang hanya dapat diakses oleh *admin*, memungkinkan pengelolaan data pengguna melalui operasi *Create*, *Read*, *Update*, dan *Delete* (CRUD). Sistem memvalidasi peran *admin* sebelum memberikan akses untuk menjamin keamanan. *Admin* dapat menambah, mengedit, atau menghapus data pengguna sesuai kebutuhan. Modul lain yang mendukung manajemen adalah *Asset Management*, yang juga berfokus pada operasi CRUD, khususnya untuk mengelola data aset perusahaan. *Admin* dapat menambahkan informasi aset baru, memperbarui detail, atau menghapus aset yang tidak lagi digunakan. Sistem ini mendukung penyimpanan file lampiran terkait aset untuk memastikan kelengkapan data.

Untuk pengelolaan pengajuan *reimbursement*, modul *Expenses* dirancang agar pengguna dapat menambah, memperbarui, melacak, atau membatalkan pengajuan. Sistem mengintegrasikan data dari tabel *expenses* untuk mendukung proses ini. Proses ini diperkuat dengan modul *Expenses Approval*, yang memberikan *admin* wewenang untuk menyetujui atau menolak pengajuan *reimbursement*. *Admin* dapat melihat daftar pengajuan berstatus *Pending* dan membuat keputusan berdasarkan informasi yang tersedia.

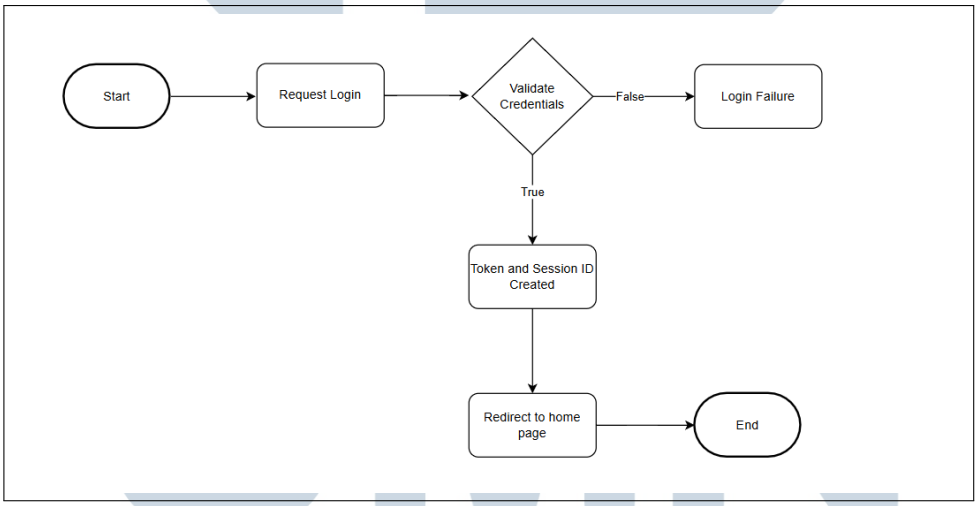
Gambar 3.2 memberikan visualisasi struktur hierarki modul-modul ini dalam website ERP-Backslash. Modul *Login* dan *Register* berada di bagian awal, berfungsi sebagai titik akses pengguna untuk masuk atau mendaftar ke sistem. Setelah itu, modul *User Management* menjadi pusat pengelolaan data pengguna, diikuti oleh *Asset Management*, yang bertugas mengelola data aset perusahaan. Modul *Expenses* mendukung pengajuan *reimbursement* oleh pengguna, sementara *Expenses Approval* memungkinkan *admin* untuk memproses persetujuan atau penolakan pengajuan tersebut. Hubungan linear antara modul-modul ini menunjukkan alur kerja sistem yang terstruktur dan terintegrasi, dengan setiap modul memainkan perannya untuk memastikan efisiensi, transparansi, dan keamanan dalam pengelolaan data.



Gambar 3.2. Use Case Diagram Website ERP-Backslash

A Modul Login

Fitur *Login* berfungsi untuk autentikasi pengguna yang sudah terdaftar. *Flowchart Login* menggambarkan proses *login* pada sebuah aplikasi. Proses dimulai dengan mengirimkan permintaan *login* dengan memasukkan kredensial pengguna (email dan *password*). Kredensial ini kemudian divalidasi oleh sistem dengan mencocokkannya ke data yang tersimpan di *database*. Jika kredensial tidak valid, proses akan berakhir dengan kegagalan *login*, dan pesan kesalahan akan ditampilkan kepada pengguna. Sebaliknya, jika kredensial valid, sistem akan membuat *Session ID* dan *Token* untuk pengguna yang berhasil terautentikasi. *Session ID* digunakan untuk menjaga sesi pengguna di dalam aplikasi, sedangkan *Token* digunakan untuk otentikasi yang aman. Terakhir, pengguna akan diarahkan ke halaman utama atau tujuan yang diinginkan di dalam aplikasi, sehingga proses *login* selesai dengan sukses. *Flowchart* alur modul *Login* dapat terlihat pada Gambar 3.3.



Gambar 3.3. Flowchart Modul Login

Endpoint utama untuk fitur ini adalah POST /login untuk proses autentikasi. Berikut merupakan detail dari API Login.

Tabel 3.2. Endpoint Login

Relative Path	`/login`
Method	`POST`

Body	`{`
	`"email": "user@example.com",`
	`"password": "password"`
	`}`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": {`
	`"token": "yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",`
	`"user_id": "12345",`
	`"redirect_url": "/dashboard"`
	`},`
	`"message": "Login successful."`
	`}`

Failed Response	`HTTP Status Code: 401`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Invalid credentials. Please check your email and password."`
	`}`
	`}`

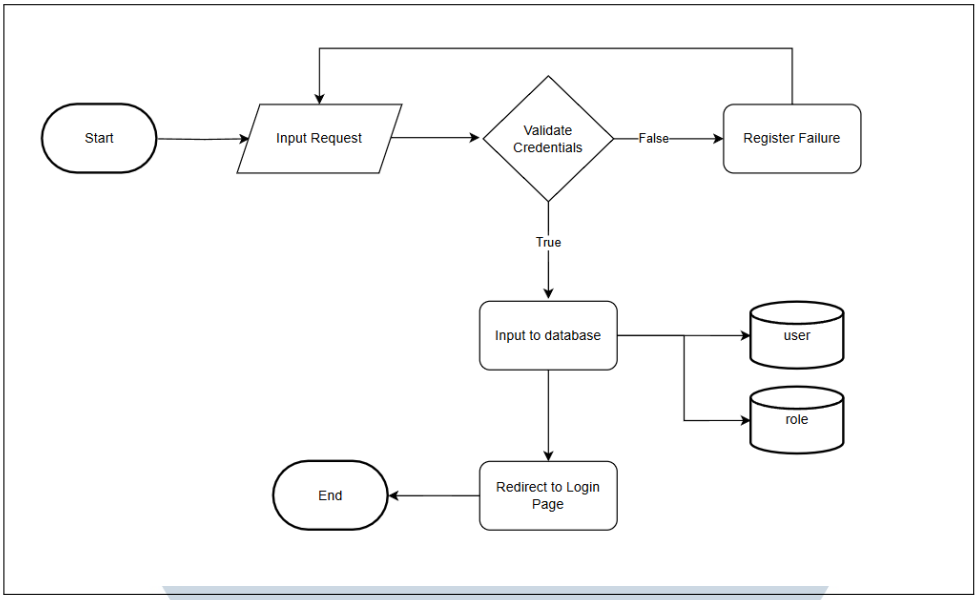
B Modul Register

Fitur *Register* berfungsi untuk pendaftaran pengguna baru ke dalam sistem. Gambar 3.4 menggambarkan alur proses yang dimulai ketika pengguna mengakses formulir pendaftaran. Pengguna kemudian mengisi formulir dengan data yang diperlukan, termasuk nama depan, nama belakang, alamat *email*, dan *password* (serta konfirmasi *password*). Setelah semua data dimasukkan, sistem akan melakukan validasi untuk memastikan bahwa semua *field* diisi dengan benar, *email* belum pernah terdaftar di *database* (*users table*), dan *password* memenuhi persyaratan panjang serta sesuai dengan konfirmasi *password*.

Jika validasi gagal, proses akan berhenti dan pengguna akan diminta untuk memperbaiki data yang salah atau tidak valid. Sebaliknya, jika validasi berhasil, sistem akan membuat akun baru di *database* dengan menyimpan informasi pengguna, seperti nama depan, nama belakang, nama lengkap, *email*, dan *password* yang telah di-*hash*. Selain itu, jika sistem menggunakan paket *roles*, pengguna akan diberikan peran *default* yaitu *user*.

Setelah akun berhasil dibuat, pengguna akan diarahkan ke halaman *login* dengan pesan sukses. Dengan demikian, proses pendaftaran selesai, dan pengguna dapat

melanjutkan ke tahap *login* setelah akun diverifikasi.



Gambar 3.4. Flowchart Modul Register

Endpoint utama untuk fitur ini sebagai berikut

Tabel 3.3. Endpoint Post Modul *Register*

Relative Path	`/register`
Method	`POST`

Body	`{`
	`"first_name": "John",`
	`"last_name": "Doe",`
	`"email": "john.doe@example.com",`
	`"password": "password",`
	`"password_confirmation": "password"`
	`}`

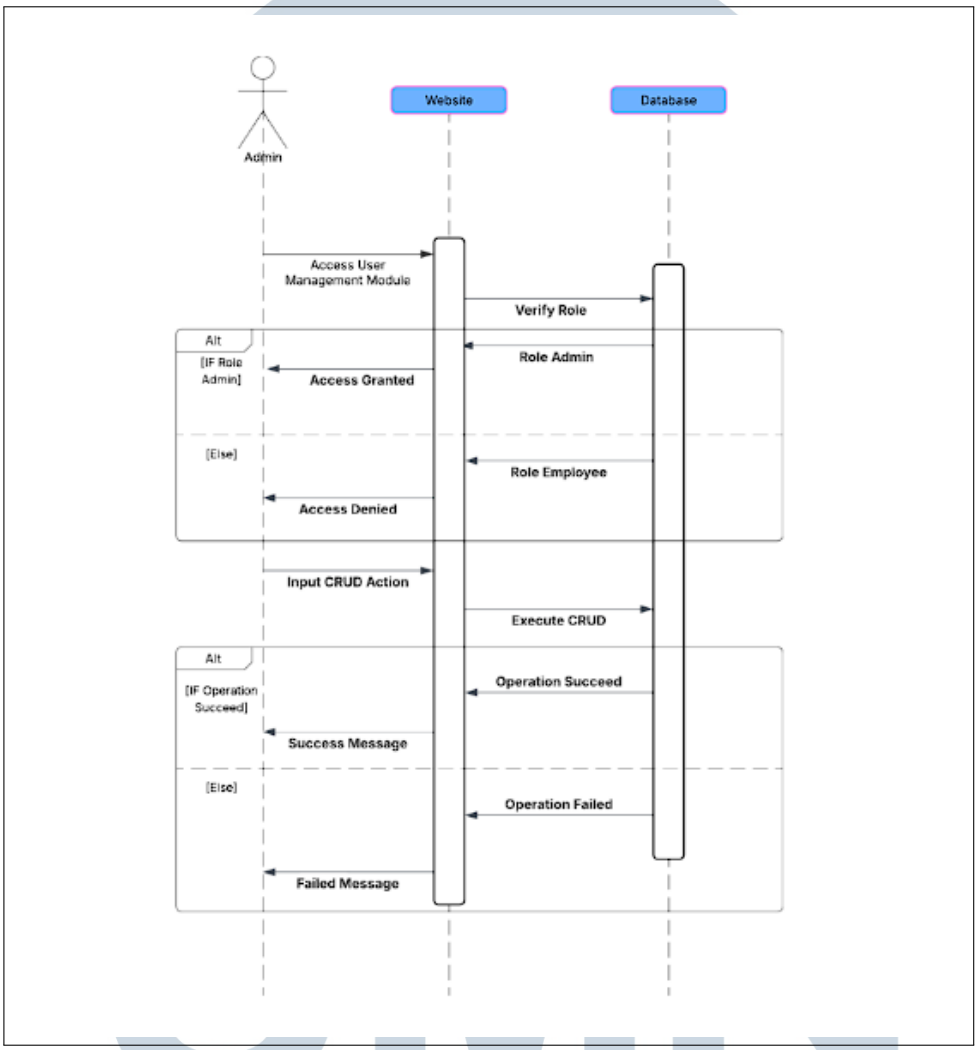
Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": {`
	`"user_id": "12345",`
	`"name": "John Doe",`
	`"email": "john.doe@example.com",`
	`"role": "User"`
	`},`
	`"message": "Registration successful! Please check your email to verify your account, then you can log in."`
	`}`

Failed Response	`HTTP Status Code: 400`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Validation failed.",`
	`"details": {`
	`"email": "The email has already been taken.",`
	`"password": "The password confirmation does not match."`
	`}`
	`}`
	`}`

C Modul User Management

Fitur *User Management* hanya dapat diakses oleh pengguna dengan *role* admin. Gambar 3.5 menggambarkan proses *user management* yang dimulai ketika admin mengakses modul manajemen pengguna. *Website* kemudian memverifikasi peran admin melalui *database*. Jika perannya adalah ”Admin”, akses diberikan; jika tidak, akses ditolak. Setelah akses diberikan, admin dapat melakukan aksi CRUD seperti menambahkan, mengedit, atau menghapus data pengguna. *Website* akan meneruskan aksi tersebut ke *database* untuk dieksekusi. Jika operasi berhasil, pesan sukses akan dikirimkan kepada admin, jika gagal, pesan kesalahan akan ditampilkan. Diagram ini menunjukkan bagaimana sistem memastikan keamanan

melalui validasi peran dan memberikan umpan balik yang jelas kepada admin untuk setiap aksi yang dilakukan. Diagram *Sequence User* Management menggambarkan urutan tindakan admin dalam fitur ini.



Gambar 3.5. Sequence Diagram Modul User Management

Endpoint untuk fitur user management sebagai berikut.

1. POST /users/create: Menambahkan data aset user baru
   Fitur ini memungkinkan penambahan pengguna baru melalui pengisian formulir. Terdapat proses validasi untuk menjamin bahwa data yang diinput memenuhi kriteria format yang telah ditentukan.

Tabel 3.4. Endpoint Post Modul *User Management*

Relative Path	`/users/create`
Method	`POST`

Body	`{`
	`"first_name": "John",`
	`"last_name": "Doe",`
	`"email": "john.doe@example.com",`
	`"password": "password",`
	`"role": "Admin"`
	`}`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": {`
	`"user_id": "12345",`
	`"message": "User created successfully."`
	`}`
	`}`

Failed Response	`HTTP Status Code: 400`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Validation failed.",`
	`"details": {`
	`"email": "The email has already been taken.",`
	`"password": "The password must be at least 8 characters."`
	`}`
	`}`
	`}`

2. GET /users: Mengambil daftar user
   Endpoint ini memungkinkan admin untuk melihat daftar *user* yang telah terdaftar.

Tabel 3.5. Endpoint Get Modul *User Management*

Relative Path	`/users`
Method	`GET`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": [`
	`{`
	`"id": 1,`
	`"name": "John Doe",`
	`"email": "john.doe@example.com",`
	`"roles": ["Admin"],`
	`"created_at": "2025-06-01T12:00:00Z"`
	`}`
	`]`
	`}`

3. PUT /users/id/edit: Memperbarui data user
   Fitur ini memungkinkan admin untuk memperbarui data *user*.

Tabel 3.6. Endpoint Put Modul *User Management*

Relative Path	`/users/{id}/edit`
Method	`PUT`

Body	`{`
	`"first_name": "John",`
	`"last_name": "Smith",`
	`"email": "john.smith@example.com",`
	`"role": "Admin"`
	`}`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": {`
	`"id": 1,`
	`"name": "John Smith",`
	`"email": "john.smith@example.com",`
	`"roles": ["Admin"]`
	`},`
	`"message": "User updated successfully."`
	`}`

Failed Response	`HTTP Status Code: 400`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Validation failed.",`
	`"details": {`
	`"email": "The email has already been taken."`
	`}`
	`}`
	`}`

4. DELETE /users/id: Menghapus user

   Fitur ini memungkinkan admin untuk menghapus *user*

Tabel 3.7. Endpoint Delete Modul *User Management*

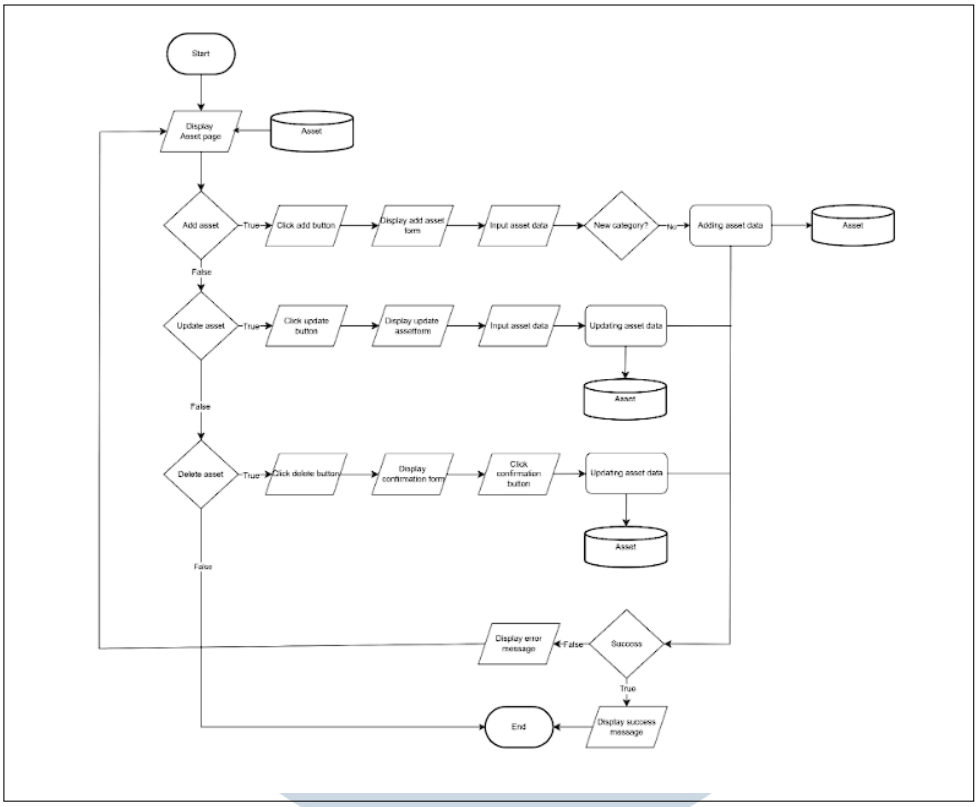
Relative Path	`/users/{id}`
Method	`DELETE`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"message": "User deleted successfully."`
	`}`

Failed Response	`HTTP Status Code: 404`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "User not found."`
	`}`
	`}`

D Modul Asset

Fitur *Asset Management* memungkinkan pengguna untuk mengelola data aset perusahaan, seperti menambah aset baru, mengubah informasi aset, dan menghapus aset yang tidak lagi digunakan. *Flowchart* 3.5 menggambarkan alur proses pengelolaan data aset dalam sistem manajemen aset yang dimulai dari inisialisasi sistem hingga pengguna mengakses halaman utama aset. Sistem terlebih dahulu memproses data dari tabel Aset yang berisi data aset, lalu menampilkan halaman aset kepada pengguna. Pada halaman ini, pengguna dapat melakukan tiga operasi utama, yaitu menambah, memperbarui, dan menghapus data aset. Saat menambah aset, pengguna menekan tombol "*Add*", mengisi formulir, lalu sistem menyimpan data ke tabel *Asset*. Untuk memperbarui data, pengguna menekan tombol "*Edit*", melengkapi formulir pembaruan, dan sistem akan memperbarui data pada tabel *Asset*. Sementara itu, untuk menghapus aset, pengguna menekan tombol "*Delete*", mengisi formulir konfirmasi, dan sistem akan menghapus data dari tabel Asset. Setelah salah satu proses tersebut selesai, sistem akan melakukan verifikasi: jika berhasil, ditampilkan pesan sukses, dan jika gagal, muncul pesan *error*.



Gambar 3.6. Flowchart Modul Asset

Endpoint utama pada modul ini sebagai berikut.

1. POST /asset/create: Menambahkan data aset baru
   Fitur ini dibuat sehingga pengguna dapat menambahkan data aset baru dengan mengisi formulir. Validasi dilakukan untuk memastikan data yang dimasukkan sesuai dengan format yang diharapkan. Jika ada lampiran (*attachment*), *file* tersebut akan disimpan dalam direktori attachments.

Tabel 3.8. Endpoint Post Modul *Asset*

Relative Path	/assets/create
Method	POST

Body	{
	"asset_id": "ASSET-003",
	"name": "Desk",
	"category": "Furniture",
	"condition": "New",
	"assigned_to": "Alice Johnson",
	"purchase_date": "2025-06-05",
	"status": "Available",
	"attachment": "<Base64EncodedFile>"
	}

Success Response	HTTP Status Code: 200
	{
	"success": true,
	"data": {
	"id": 3,
	"asset_id": "ASSET-003",
	"name": "Desk",
	"category": "Furniture",
	"condition": "New",
	"assigned_to": "Alice Johnson",
	"purchase_date": "2025-06-05",
	"status": "Available",
	"attachment": "attachments/desk.jpg"
	},
	"message": "Asset created successfully."
	}

Failure Response	HTTP Status Code: 500
	{
	"success": false,
	"error": {
	"message": "There was a problem creating the asset.
	Please try again."
	}
	}

2. GET /asset: Mengambil daftar aset

   Endpoint ini memungkinkan pengguna untuk melihat daftar aset yang telah terdaftar. Daftar ini dapat di *filter* berdasarkan kata kunci yang dicari oleh pengguna.

Tabel 3.9. Endpoint Get Modul *Asset*

Relative Path	`/assets`
Method	`GET`
Query Parameters	`Search (optional)`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": [`
	`{`
	`"id": 1,`
	`"asset_id": "ASSET-001",`
	`"name": "Laptop",`
	`"category": "Electronics",`
	`"condition": "New",`
	`"assigned_to": "John Doe",`
	`"purchase_date": "2025-06-01",`
	`"status": "Available",`
	`"attachment": "attachments/laptop.pdf"`
	`}`
	`]`
	`}`

3. PUT /asset/id/edit: Memperbarui data aset

   Fitur ini memungkinkan pengguna untuk memperbarui data aset, termasuk mengganti lampiran (*attachment*) dengan menghapus file lama jika ada.

Tabel 3.10. Endpoint Put Modul *Asset*

Relative Path	/assets/{id}/edit
Method	PUT

Body	{
	"asset_id": "ASSET-003",
	"name": "Desk",
	"category": "Furniture",
	"condition": "Good",
	"assigned_to": "Bob Anderson",
	"purchase_date": "2025-06-05",
	"status": "Assigned",
	"attachment": "<Base64EncodedFile>"
	}

Success Response	HTTP Status Code: 200
	{
	"success": true,
	"data": {
	"id": 3,
	"asset_id": "ASSET-003",
	"name": "Desk",
	"category": "Furniture",
	"condition": "Good",
	"assigned_to": "Bob Anderson",
	"purchase_date": "2025-06-05",
	"status": "Assigned",
	"attachment": "attachments/desk_updated.jpg"
	},
	"message": "Asset updated successfully."
	}

Failure Response	HTTP Status Code: 500
	{
	"success": false,
	"error": {
	"message": "There was a problem updating the asset.
	Please try again."
	}
	}

4. DELETE /asset/id: Menghapus aset

   Fitur ini memungkinkan pengguna untuk menghapus aset, termasuk *file* lampiran jika ada.

Tabel 3.11. Endpoint Delete Modul *Asset*

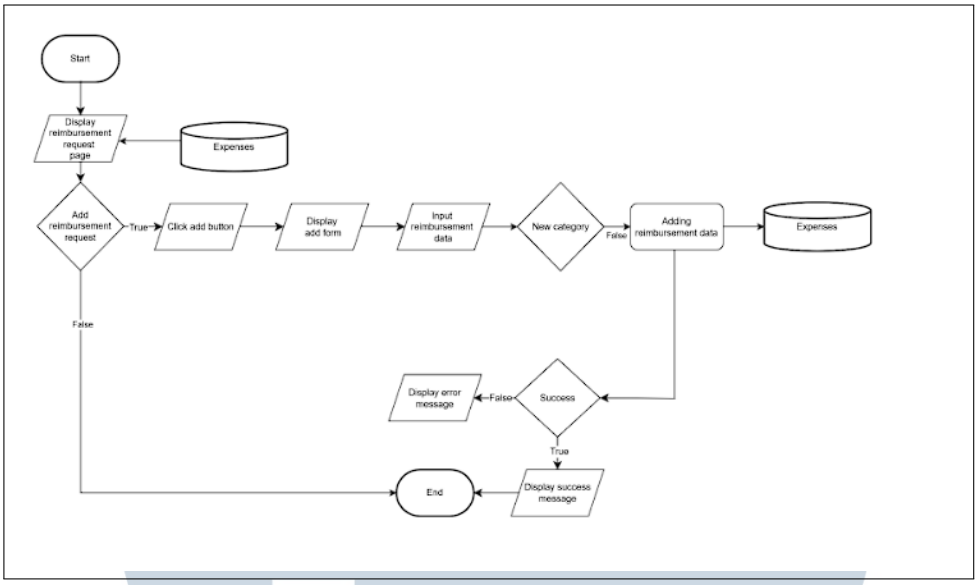
Relative Path	`/assets/{id}`
Method	`DELETE`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"message": "Asset deleted successfully."`
	`}`

Failure Response	`HTTP Status Code: 500`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "There was a problem deleting the asset. It might be in use or another issue occurred."`
	`}`
	`}`

E Modul Expenses

Fitur *Expenses* dirancang untuk mempermudah pengguna dalam mengajukan dan melacak pengajuan *reimbursement*. *Flowchart* pada gambar 3.7 menggambarkan alur proses permintaan *reimbursement* dalam sistem, dimulai dari tahap pemrosesan data halaman permintaan *reimbursement* yang mengambil data dari tabel expenses. Setelah data berhasil diproses, sistem akan menampilkan halaman permintaan *reimbursement* kepada pengguna. Di halaman ini, pengguna dapat melakukan operasi utama yaitu menambah data permintaan *reimbursement*. Untuk menambah permintaan *reimbursement*, pengguna menekan tombol ”Add”, kemudian sistem menampilkan formulir penambahan, pengguna mengisi data yang diperlukan, dan data akan disimpan ke dalam tabel *expenses*. Setelah proses penambahan selesai dilakukan, sistem akan mengecek apakah proses berhasil. Jika berhasil, maka sistem akan menampilkan pesan sukses, namun jika gagal, sistem akan menampilkan pesan *error*. Alur ini memastikan bahwa semua proses pengelolaan permintaan *reimbursement* berjalan secara sistematis dan terkontrol dalam sistem.



Gambar 3.7. Flowchart Modul Expenses

Endpoint utama pada modul terlihat dibawah ini

1. POST /expenses/create: Mengajukan data reimbursement baru
   Admin atau pengguna dapat melihat daftar *reimbursement*. Data yang ditampilkan diurutkan berdasarkan tanggal pengeluaran secara menurun. Penggunaan relasi dengan user memastikan data pengguna juga dimuat.

Tabel 3.12. Endpoint Post Modul *Expenses*

Relative Path	/expenses/create
Method	POST

Body	{
	"title": "Team Lunch",
	"category": "Food",
	"project": "Year-End Party",
	"amount": 250.00,
	"description": "Lunch with the marketing team",
	"date_of_expense": "2025-06-01",
	"receipt": "<Base64EncodedFile>"
	}

Success Response	HTTP Status Code: 200
	{
	"success": true,
	"data": {
	"id": 2,
	"title": "Team Lunch",
	"category": "Food",
	"project": "Year-End Party",
	"amount": 250.00,
	"description": "Lunch with the marketing team",
	"date_of_expense": "2025-06-01",
	"status": "Pending",
	"receipt": "receipts/team_lunch.pdf",
	"user_id": 2
	},
	"message": "Expense submitted."
	}

Failure Response	HTTP Status Code: 500
	{
	"success": false,
	"error": {
	"message": "There was a problem submitting your expense. Please try again."
	}
	}

2. GET /expenses: Mengambil daftar reimbursement pengguna dengan pagination.

   Pengguna dapat mengajukan *reimbursement* baru dengan mengisi formulir. Validasi dilakukan untuk memastikan kelengkapan data, dan file bukti dapat diunggah dalam format tertentu.

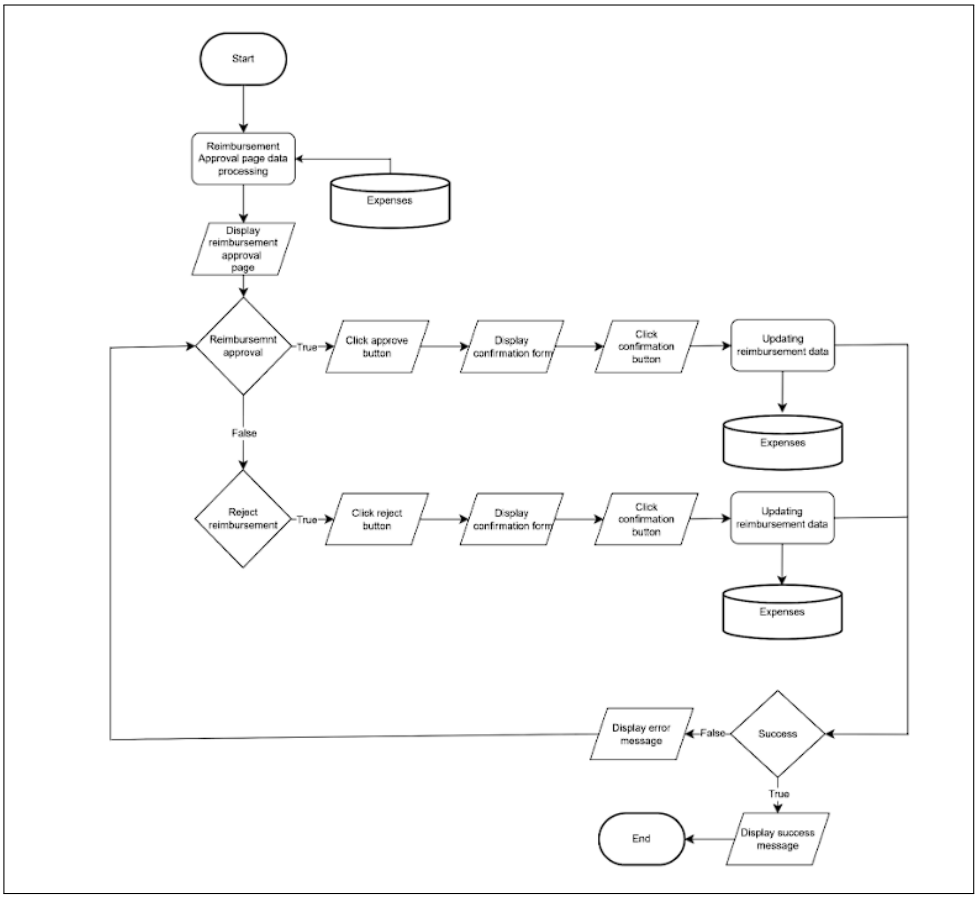
Tabel 3.13. Endpoint Get Modul *Expenses*

Relative Path	`/expenses`
Method	`GET`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": [`
	`{`
	`"id": 1,`
	`"title": "Office Supplies",`
	`"category": "Stationery",`
	`"project": "Redesign Project",`
	`"amount": 150.00,`
	`"description": "Purchased pens and notebooks",`
	`"date_of_expense": "2025-06-01",`
	`"status": "Approved",`
	`"receipt": "receipts/office_supplies.pdf",`
	`"user": {`
	`"id": 1,`
	`"name": "John Doe"`
	`}`
	`}`
	`]`
	`}`

- Modul Expenses Approval

  Fitur *Expenses* memiliki lanjutan dimana pengguna yang hanya memiliki *role* admin mendapatkan wewenang untuk mempersetujui atau menolak request yang telah dibuat oleh pengguna. *Flowchart* pada Gambar 3.8 dibawah ini menggambarkan alur proses persetujuan *reimbursement* dalam sistem. Proses dimulai dengan admin membuka halaman persetujuan *reimbursement*, di mana data *reimbursement* dengan status ”*Pending*” diambil dari *database* tabel *reimbursements* dan ditampilkan dalam antarmuka sistem. Pada tahap ini, admin dapat mengambil keputusan apakah pengajuan *reimbursement* akan disetujui atau ditolak. Jika admin memilih untuk menyetujui *reimbursement*, mereka akan menekan tombol ”*Approve*” yang akan menampilkan formulir konfirmasi. Setelah formulir tersebut diisi dan disetujui, data *reimbursement* diperbarui di *database* dengan status ”Approved”. Sebaliknya, jika admin memilih untuk menolak *reimbursement*, tombol ”*Reject*” akan ditekan dan sistem menampilkan formulir konfirmasi untuk verifikasi keputusan. Setelah formulir ini disetujui, status *reimbursement* di *database* akan diperbarui menjadi ”*Rejected*”.



Gambar 3.8. Flowchart Modul Expenses Approval

Endpoint modul ini merupakan:

1. GET /expenses/approval: Menampilkan daftar *reimbursement* yang menunggu persetujuan

   Pengguna dengan role admin dapat melihat daftar pengajuan *reimbursement* dengan status Pending. Daftar ini disediakan untuk membantu admin memproses persetujuan.

Tabel 3.14. Endpoint Get Modul *Expenses Approval*

Relative Path	`/expenses/approval`
Method	`GET`
Description	Fetches all expenses with the status Pending.

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"data": [`
	`{`
	`"id": 3,`
	`"title": "Travel Reimbursement",`
	`"category": "Transport",`
	`"project": "Client Meeting",`
	`"amount": 500.00,`
	`"description": "Taxi fare to client meeting",`
	`"date_of_expense": "2025-06-02",`
	`"status": "Pending",`
	`"receipt": "receipts/travel_reimbursement.pdf",`
	`"user": {`
	`"id": 3,`
	`"name": "Jane Smith"`
	`}`
	`}`
	`]`
	`}`

2. PUT /expenses/id/approve: Menyetujui reimbursement

Tabel 3.15. Endpoint Put Modul *Expenses Approved*

Relative Path	`/expenses/{id}/approve`
Method	`PUT`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"message": "Expense approved."`
	`}`

Failure Response	`HTTP Status Code: 500`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Could not approve the expense. Please try again."`
	`}`
	`}`

3. PUT /expenses/id/reject: Menolak reimbursement.

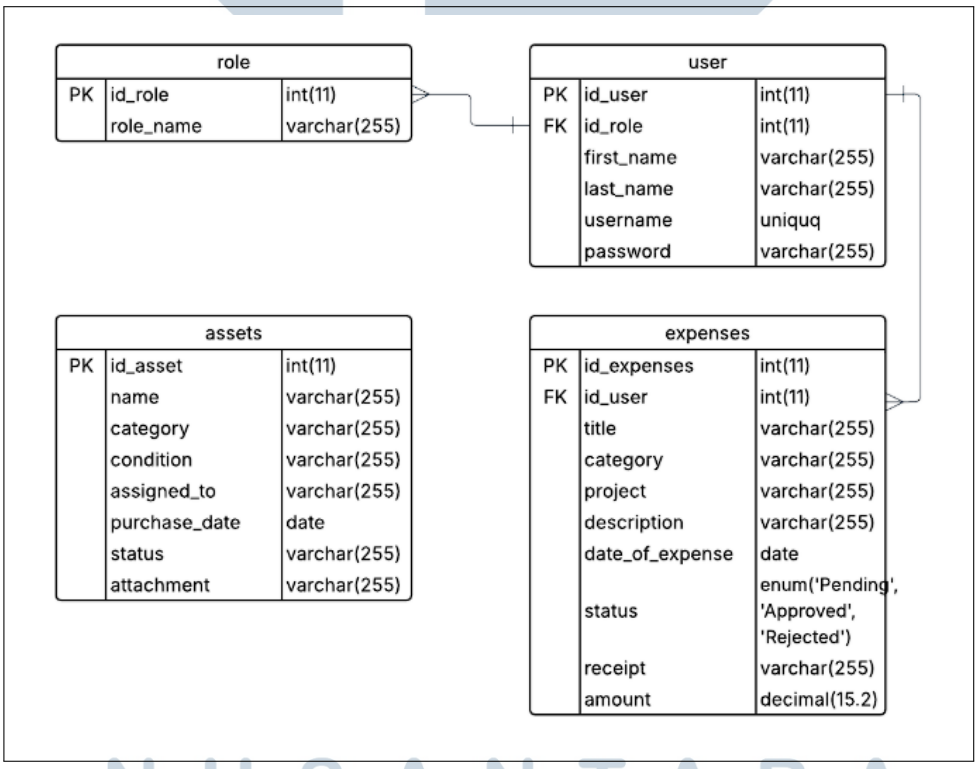
Tabel 3.16. Endpoint Put Modul *Expenses Rejected*

Relative Path	`/expenses/{id}/reject`
Method	`POST`

Success Response	`HTTP Status Code: 200`
	`{`
	`"success": true,`
	`"message": "Expense rejected."`
	`}`

Failure Response	`HTTP Status Code: 500`
	`{`
	`"success": false,`
	`"error": {`
	`"message": "Could not reject the expense. Please try again."`
	`}`
	`}`

F Database



Gambar 3.9. ERD Database

Gambar 3.9 menggambarkan diagram yang menunjukkan hubungan antar entitas dalam sistem. Tabel trole memiliki relasi *one-to-many* dengan tabel `user`, yang berarti setiap peran dalam tabel `role` dapat dimiliki oleh banyak pengguna di tabel `user`, namun setiap pengguna hanya dapat memiliki satu peran. Hubungan ini

dihubungkan melalui atribut id_role, di mana id_role dalam tabel user bertindak sebagai *Foreign Key* yang merujuk ke atribut id_role sebagai *Primary Key* di tabel role.

Selanjutnya, tabel user memiliki relasi *one-to-many* dengan tabel expenses. Hal ini memungkinkan satu pengguna untuk membuat banyak pengajuan *reimbursement*, tetapi setiap pengajuan hanya terkait dengan satu pengguna. Relasi ini terhubung melalui atribut id_user, di mana id_user dalam tabel expenses bertindak sebagai *Foreign Key* yang merujuk ke atribut id_user sebagai *Primary Key* dalam tabel user.

Sementara itu, tabel assets tidak memiliki relasi langsung dengan tabel lainnya dalam sistem. Hal ini menunjukkan bahwa data aset dikelola secara independen tanpa adanya keterkaitan dengan data pengguna, peran, atau *reimbursement*. Relasi yang dirancang dalam diagram ini memastikan integritas data dalam sistem dan memungkinkan pengelolaan informasi yang terstruktur dan efisien.

Berikut merupakan struktur tabel database yang digunakan dalam website ERP-Backslash:

1. Tabel role
   Tabel role menyimpan daftar peran yang dapat dimiliki oleh pengguna dalam sistem. Setiap entri dalam tabel memiliki identifikasi unik dalam kolom id_role, yang berfungsi sebagai *Primary Key*. Kolom role_name digunakan untuk menyimpan nama peran dan bersifat wajib diisi. Informasi dari tabel ini digunakan untuk mengelompokkan pengguna berdasarkan peran yang dimiliki. Struktur tabel role dapat dilihat pada Tabel 3.17.

Tabel 3.17. Struktur Tabel *role*

Nama kolom	Tipe data	Nullable
id_role	int(11)	Primary Key
role_name	varchar(255)	Not Null

2. Tabel user
   Tabel user berfungsi untuk menyimpan data karyawan yang terdaftar dalam sistem. Setiap karyawan memiliki identifikasi unik pada kolom id_user sebagai *Primary Key*. Kolom id_role digunakan sebagai *Foreign Key* yang merujuk ke tabel role, untuk menentukan peran karyawan tersebut. Nama karyawan disimpan dalam kolom employee_name dan wajib diisi. Selain itu, informasi login seperti username yang bersifat unik dan password yang wajib diisi juga disimpan dalam tabel ini. Struktur tabel user dapat dilihat pada Tabel 3.18.

3. Tabel assets
   Tabel assets digunakan untuk menyimpan data aset yang dimiliki oleh organisasi. Setiap aset memiliki id_asset sebagai *Primary Key*. Nama aset

Tabel 3.18. Struktur Tabel *user*

Nama kolom	Tipe data	Nullable
id_user	int(11)	Primary Key
id_role	int(11)	Foreign Key
first_name	varchar(255)	Not Null
last_name	varchar(255)	Not Null
username	varchar(255)	Unique
password	varchar(255)	Not Null

disimpan dalam kolom name dan wajib diisi. Kolom state menunjukkan kondisi aset (baru, bekas, atau rusak), sedangkan kolom assigned_to berisi informasi kepada siapa atau tempat aset tersebut ditugaskan. Tanggal pembelian disimpan dalam purchase_date, dan status menunjukkan status aset apakah aktif, diarsipkan, atau dalam perawatan. Struktur tabel assets dapat dilihat pada Tabel 3.19.

Tabel 3.19. Struktur Tabel *assets*

Nama kolom	Tipe data	Nullable
id_asset	int(11)	Primary Key
name	varchar(255)	Not Null
category	varchar(255)	Not Null
condition	varchar(255)	Not Null
assigned_to	varchar(255)	Default Null
purchase_date	date	Not Null
status	varchar(255)	Not Null
attachment	varchar(255)	Default Null

4. Tabel expenses
   Tabel expenses menyimpan data pengajuan *reimbursement* oleh karyawan. Setiap pengajuan memiliki id_expenses sebagai *Primary Key*. Kolom id_user menunjukkan pengguna yang mengajukan *reimbursement*, yang merupakan *Foreign Key* ke tabel user. Kolom approval berisi status persetujuan (Pending, Approved, Rejected) dengan nilai default Pending. amount menyimpan jumlah nominal *reimbursement* dan wajib diisi. Bukti pengeluaran disimpan pada kolom receipt, dan description berisi penjelasan tambahan. Tanggal pengeluaran disimpan dalam date_of_expense, dan category digunakan untuk mengelompokkan jenis *reimbursement* sesuai kategori. Struktur tabel expenses dapat dilihat pada Tabel 3.20.

Tabel 3.20. Struktur Tabel *expenses*

Nama kolom	Tipe data	Nullable
id_expenses	int(11)	Primary Key
id_user	int(11)	Foreign Key
title	varchar(255)	Not Null
category	varchar(255)	Not Null
project	varchar(255)	Not Null
status	enum('Pending', 'Approved', 'Rejected')	Default 'Pending'
amount	decimal(15,2)	Not Null
receipt	varchar(255)	Not Null
description	varchar(255)	Default Null
date_of_expense	date	Default Null

G Ad Hoc

Selama masa magang, intern melakukan tugas tambahan seperti pembuatan skrip program untuk mempercepat proses ekstraksi data dari file dan basis data. Dalam tahap ini, data dikumpulkan dari file eksternal (rpt). Pengambilan data dari file dilakukan dengan membaca konten dan memprosesnya sesuai kebutuhan.

Berdasarkan Gambar 3.9 *Flowchart Script Python*, program ini dirancang menggunakan bahasa Python untuk mengekstrak informasi dari file XML yang sebelumnya rpt disimpan dalam folder tertentu dan mengelolanya dalam format terstruktur untuk mendukung kebutuhan pelaporan.

Proses dimulai dengan memindai semua file XML di folder *input*, kemudian mengekstrak informasi seperti judul laporan, penulis, nama database, tabel data, kolom yang digunakan, rumus, parameter, grup, grafik, dan filter laporan. Jika elemen tertentu tidak ditemukan, nilai default seperti "None" akan digunakan. Data yang berhasil diekstraksi disimpan dalam sebuah file Excel (output.xlsx) untuk memudahkan analisis, sementara file yang gagal diproses dicatat dalam file teks terpisah (failed_list.txt) beserta penyebab kegagalannya. Program juga mencetak ringkasan jumlah file yang berhasil dan gagal diproses, termasuk lokasi file *output*.

Program ini dibuat untuk mengatasi tantangan dalam proses ekstraksi data yang sebelumnya dilakukan secara manual. Metode manual cenderung memakan waktu yang lama dan rentan terhadap kesalahan, terutama dengan jumlah data yang tinggi untuk diproses. Selain itu, proses manual tidak menyediakan mekanisme untuk menangani file yang gagal diproses atau memberikan laporan ringkasan yang memadai, sehingga menyulitkan analisis data lebih lanjut.

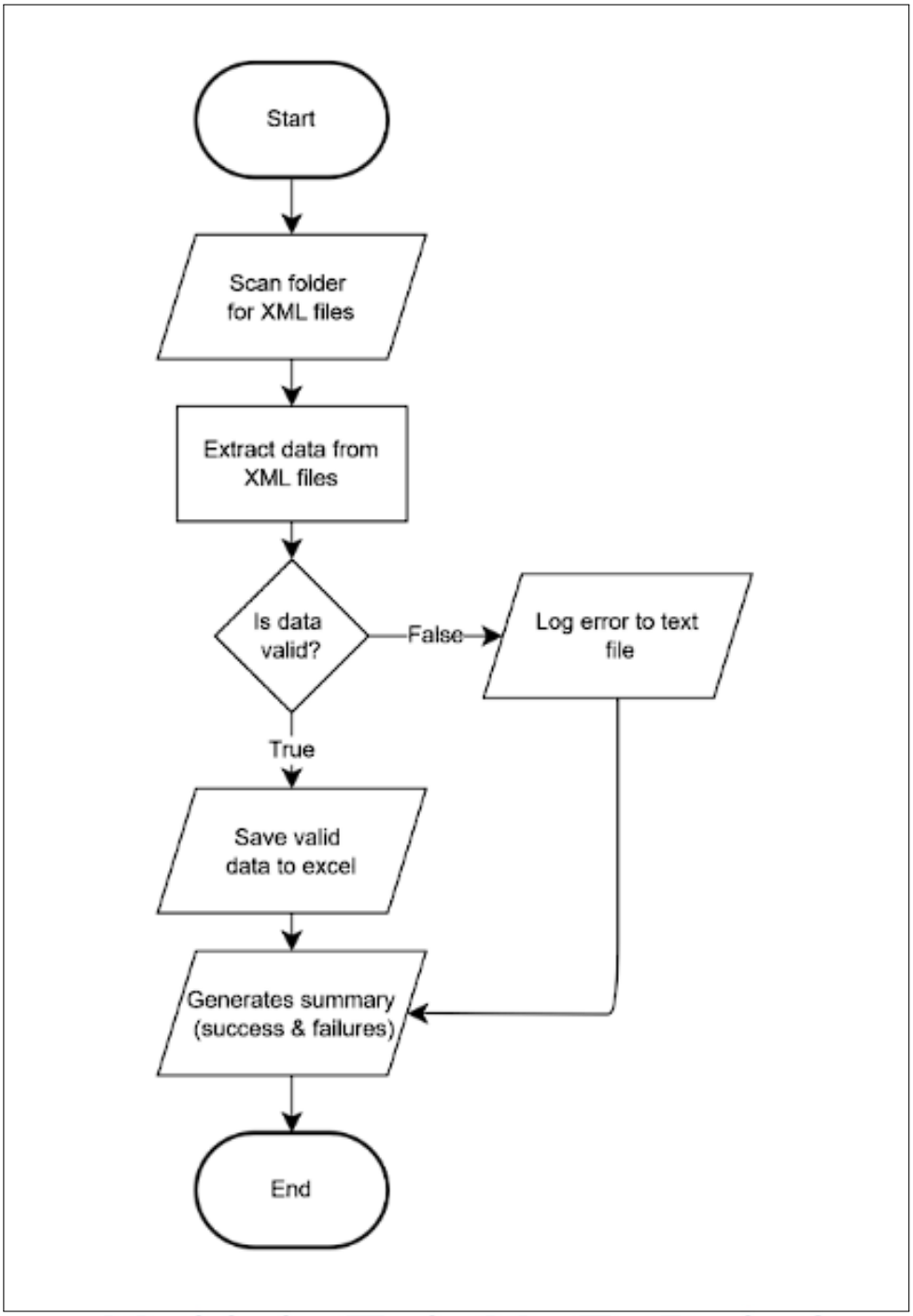
Dengan scripting ini, tujuan utamanya adalah:

- Mempercepat proses ekstraksi data,
- Mengurangi risiko kesalahan,

- Menghasilkan data dalam format terstruktur yang siap digunakan untuk analisis dan pelaporan,
- Meningkatkan efisiensi kerja dengan menyediakan file *output* terorganisir dan dokumentasi kegagalan yang jelas.

Hasil yang dicapai menunjukkan efisiensi, di mana proses manual membutuhkan waktu sekitar 30 menit untuk memproses 6 file. Dengan menggunakan program scripting, seluruh 2.014 file dapat diproses hanya dalam waktu 2 menit.

File Excel (`output.xlsx`) yang dihasilkan memberikan kemudahan dalam analisis data karena telah terstruktur dengan baik. Dokumentasi kegagalan dalam file teks (`failed_list.txt`) juga membantu pengguna memahami penyebab kesalahan dan memperbaiki file yang bermasalah. Dengan adanya sistem ini, pengambilan keputusan berbasis data menjadi lebih cepat, akurat, dan dapat diandalkan. Selain itu, efisiensi tim meningkat secara signifikan karena waktu yang sebelumnya digunakan untuk tugas manual dapat dialokasikan untuk pekerjaan lain yang lebih strategis.



Gambar 3.10. Flowchart Script Python

3.4 Kendala yang ditemukan

Bagian ini berisi kendala yang ditemukan selama proses kerja magang:

- Kebutuhan pengguna yang sering berubah menyebabkan terjadinya miskomunikasi.

- Kesalahan dalam menginterpretasikan kebutuhan fitur sistem inventaris dan reimbursement telah menghambat proses pengembangan website ERP-Backslash. Masalah ini bersumber dari pemahaman yang tidak memadai mengenai Standar Operasional Prosedur (SOP) perusahaan yang relevan.

3.5 Solusi yang ditemukan

Bagian ini berisi solusi atas kendala yang ditemukan selama proses kerja magang:

- Untuk memastikan akurasi dalam proses pengembangan *website ERP-Backslash*, setiap diskusi dengan pengguna direkam. Rekaman ini berfungsi sebagai dokumentasi valid dan referensi utama bagi tim pengembang. Adanya bukti autentik ini sangat penting untuk memverifikasi setiap detail saat pengguna mengajukan perubahan atau permintaan baru, sehingga potensi miskomunikasi di masa depan dapat diminimalkan.
- Untuk mencapai keselarasan pemahaman mengenai kebutuhan fitur, dilakukan analisis mendalam terhadap Standar Operasional Prosedur (SOP) perusahaan. Hal ini didukung dengan sesi diskusi dan peninjauan (review) yang diadakan secara berkala bersama pihak perusahaan.