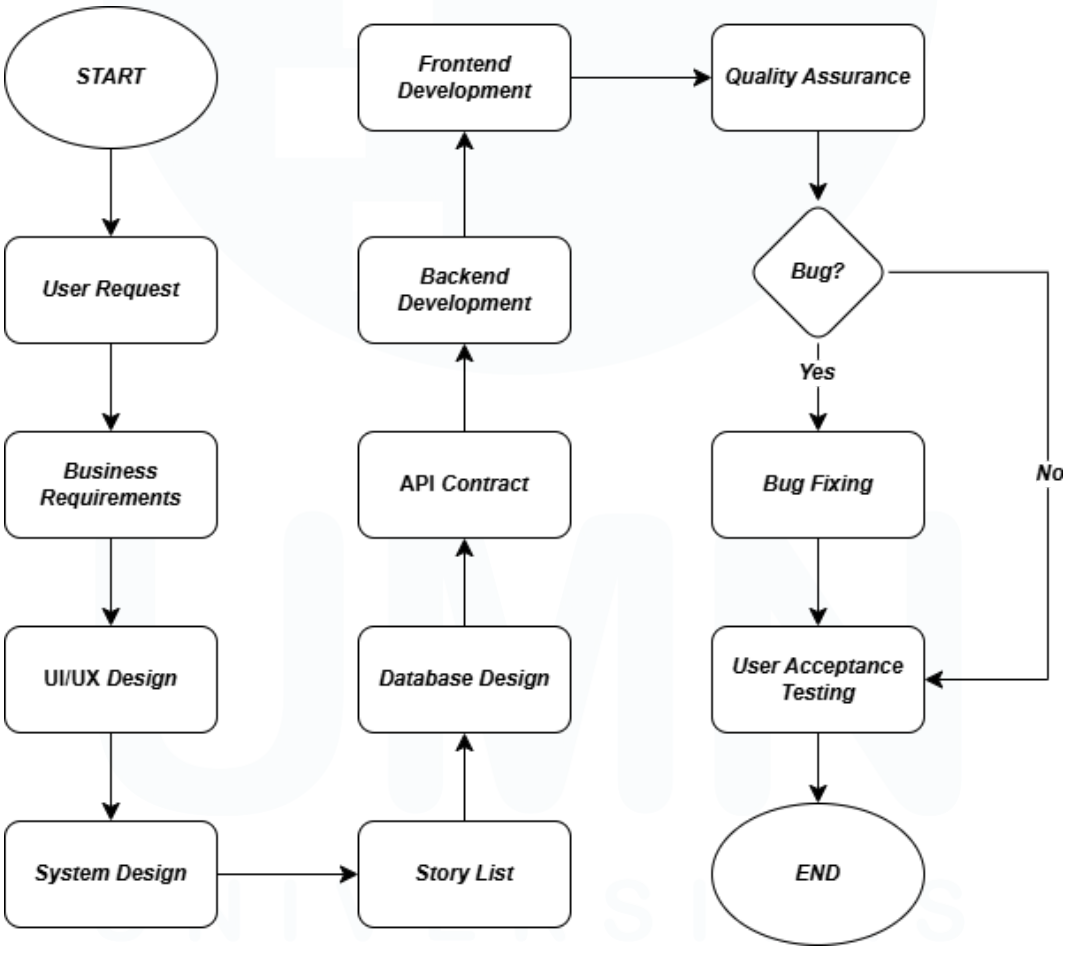
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Sebagai *Intern* IT *Corporate Backend Engineer* di PT. Global Loyalty Indonesia (GLI), kedudukan penulis berada di bawah koordinasi *Manager* IT *Development* dan bekerja secara kolaboratif dengan tim *Project Manager, Frontend Developer*, *Quality Assurance* (QA), dan UI/UX *Designer*. Alur kerja dan koordinasi dijelaskan sebagai berikut:



Gambar 3.1 Alur Kerja & Koordinasi.

Gambar 3.1 menunjukkan bahwa proyek dimulai ketika *user* menghubungi *Project Manager* (PM) IT *Corporate* dengan permintaan tertentu, seperti

pengembangan fitur baru, perbaikan, atau pembuatan aplikasi. PM akan mendokumentasikan kebutuhan user dan memastikan bahwa semua persyaratan bisnis (*Business Requirements*) telah dipahami dengan jelas. Setelah itu, PM berkoordinasi dengan UI/UX *Designer* untuk merancang desain antarmuka pengguna yang sesuai dengan kebutuhan bisnis. Tahap ini meliputi pembuatan *wireframe*, *mockup*, dan *prototype* yang menggambarkan alur pengguna (*user flow*) dan tampilan aplikasi. Desain ini kemudian dipresentasikan kepada user untuk mendapatkan persetujuan sebelum melanjutkan ke tahap berikutnya.

Setelah desain UI/UX disetujui, langkah berikutnya adalah membuat *System Design*. *System Design* mencakup perencanaan arsitektur sistem, *scope* proyek, *business rules*, dan *activity diagram*. Dokumen ini menjadi panduan bagi tim *developer* untuk memahami bagaimana sistem akan bekerja secara keseluruhan. Selain itu, PM juga menyediakan *Story List* sebagai panduan bagi tim *developer* untuk merancang aplikasinya. Lalu, tim *Backend Developer* akan membuat rancangan *Database Design* dalam bentuk *Entity Relationship Diagram* (ERD). ERD ini menggambarkan struktur *database*, termasuk tabel, kolom, tipe data, dan hubungan antar tabel, yang penting untuk memastikan bahwa data dapat disimpan dan diakses dengan efisien.

Selanjutnya, *Backend Developer* menyusun API *Contract*, yang berisi spesifikasi API yang akan digunakan oleh *Frontend Developer* untuk berkomunikasi dengan *backend*. API *Contract* mencakup *endpoint*, metode HTTP, parameter, dan respons yang diharapkan. Dengan kontrak API yang telah disepakati, tim *Frontend* dan *Backend* dapat memulai pengembangan secara paralel. *Frontend Developer* mengembangkan antarmuka pengguna berdasarkan desain UI/UX, sementara *Backend Developer* mengimplementasikan logika bisnis dan integrasi *database*. Proses *development* ini juga diakhiri dengan integrasi yang dilakukan oleh baik *Frontend* dan *Backend* Developer.

Setelah pengembangan selesai, aplikasi akan melalui tahap *Quality Assurance* (QA). Tim QA melakukan berbagai jenis *testing*, seperti *functional testing*, *performance testing*, dan lainnya untuk mengidentifikasi dan melaporkan

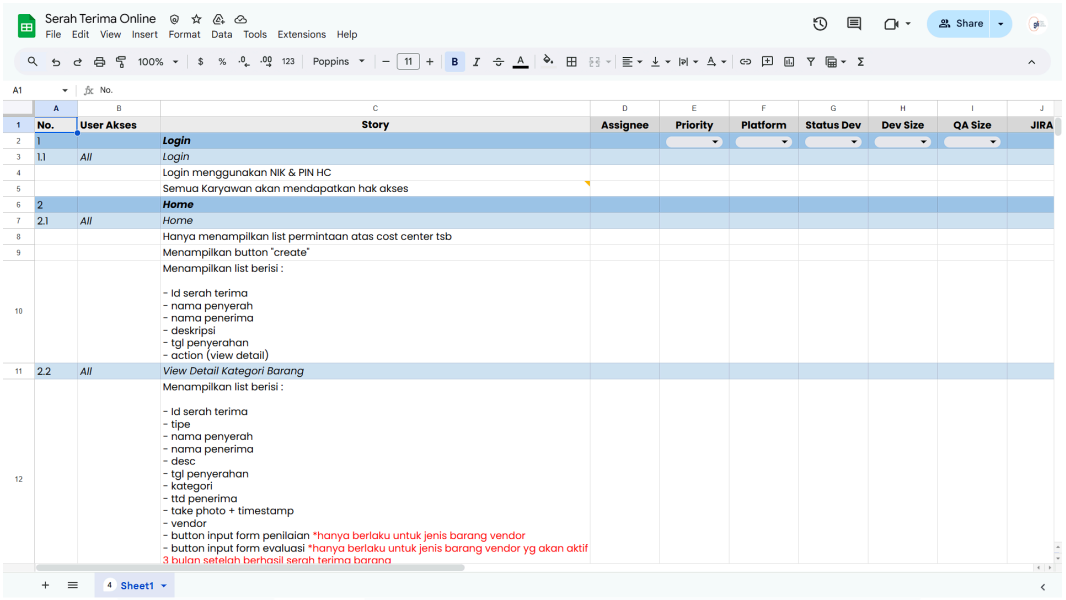
bug. Jika ditemukan *bug*, tim *developer* akan melakukan *Bug Fixing* hingga aplikasi bebas dari *error*. Jika tidak ada *bug* yang ditemukan, aplikasi akan menuju tahap *User Acceptance Testing* (UAT), di mana *user* menguji aplikasi untuk memastikan bahwa semua fitur berfungsi sesuai dengan kebutuhan awal. Jika *user* menyetujui, aplikasi siap diluncurkan (*production*).

Setelah UAT berhasil, aplikasi siap digunakan oleh *user* sesuai dengan permintaan awal. Tim IT memastikan bahwa aplikasi berjalan dengan baik di lingkungan *production* dan memberikan dukungan *post-production* jika diperlukan. Dengan alur kerja yang terstruktur ini, setiap tahap proyek dapat berjalan dengan efisien, memastikan bahwa aplikasi yang dihasilkan memenuhi kebutuhan *user* dan standar kualitas yang ditetapkan.

No	Business Requirements - In Scope
1	Enhance Aplikasi Dashboard Promo Alfagift
2	Penambahan fitur upload bulky program Promo Alfagift pada halaman home
3	Pada halaman home tambahkan status voucher berhasil generate atau tidak
4	Pada halaman home tambahkan status point / voucher sudah atau belum terinput
5	Pada halaman home tambahkan status item whitelist / blacklist sudah atau belum terinput
6	Pada halaman create jika tipe reward "Voucher" maka wajib input promoId dan nama promo akan auto generated dalam bentuk greyout sesuai dengan promoId yg di input
7	Pada halaman create jika tipe reward "Voucher" maka wajib input juklakId vlp atau id SO voucher dan kuota voucher akan auto generated dalam bentuk greyout sesuai dengan juklakId vlp arau id SO voucher yg di input
8	Pada halaman create jika tipe reward "Voucher" maka wajib upload file kode voucher dan kuota voucher akan auto generated dalam bentuk greyout sesuai dengan file voucher yg di upload
9	Penambahan menu report dengan range periode tgl startEndDate dalam bentuk excel

Gambar 3.2 Contoh *Business Requirements*.

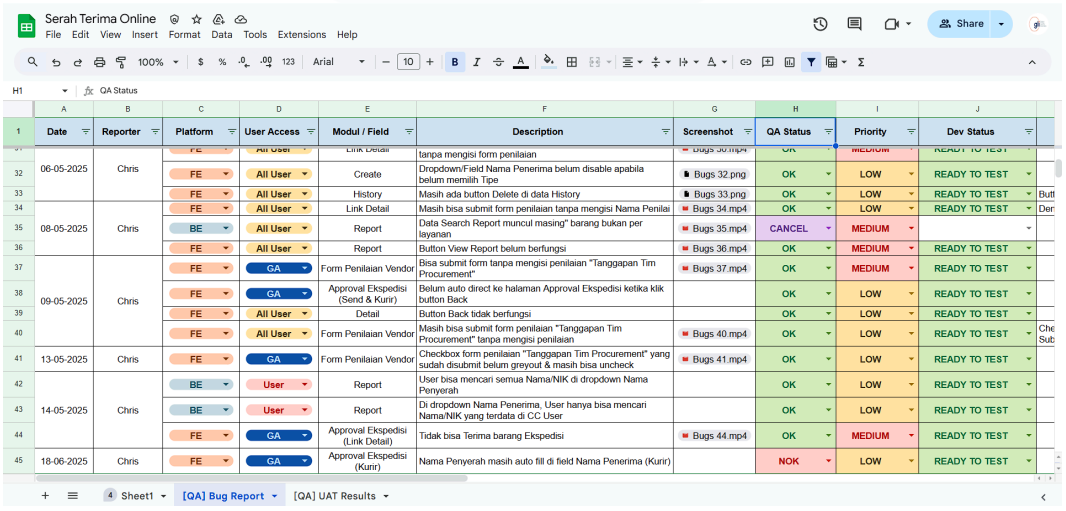
Gambar 3.2 menunjukkan detail kebutuhan bisnis dari sebuah aplikasi yang ingin dikembangkan.



No.	User Akses	Story	Assignee	Priority	Platform	Status Dev	Dev Size	QA Size	JIRA
1		***Login***							
1.1	*All*	*Login*							
		Login menggunakan NIK & PIN HC							
		Semua Karyawan akan mendapatkan hak akses							
2		***Home***							
2.1	*All*	*Home*							
		Hanya menampilkan list permintaan atas cost center tsb							
		Menampilkan button "create"							
		Menampilkan list berisi : - Id serah terima - nama penyerah - nama penerima - deskripsi - tgl penyerahan - action (view detail)							
2.2	*All*	*View Detail Kategori Barang*							
		Menampilkan list berisi : - Id serah terima - tipe - nama penyerah - nama penerima - desc - tgl penyerahan - kategori - ttd penerima - take photo + timestamp - vendor - button input form penilaian *hanya berlaku untuk jenis barang vendor - button input form evaluasi *hanya berlaku untuk jenis barang vendor yg akan aktif 3 bulan setelah berhasil serah terima barang							

Gambar 3.3 Contoh *Story List*.

Gambar 3.3 menunjukkan daftar keperluan yang perlu dibuat oleh para *developer*, menjadikan panduan dalam mengembangkan aplikasi.



Date	Reporter	Platform	User Access	Modul / Field	Description	Screenshot	QA Status	Priority	Dev Status
					tanpa mengisi form penilaian				
06-05-2025	Chris	FE	All User	Create	Dropdown/Field Nama Penerima belum disable apabila belum memilih Tipe	Bugs 32.png	OK	LOW	READY TO TEST
		FE	All User	History	Masih ada button Delete di data History	Bugs 33.png	OK	LOW	READY TO TEST
		FE	All User	Link Detail	Masih bisa submit form penilaian tanpa mengisi Nama Penilai	Bugs 34.mp4	OK	LOW	READY TO TEST
08-05-2025	Chris	BE	All User	Report	Data Search Report muncul masing" barang bukan per layanan	Bugs 35.mp4	CANCEL	MEDIUM	
		FE	All User	Report	Button View Report belum berfungsi	Bugs 36.mp4	OK	MEDIUM	READY TO TEST
		FE	GA	Form Penilaian Vendor	Bisa submit form tanpa mengisi penilaian "Tanggapan Tim Procurement"	Bugs 37.mp4	OK	MEDIUM	READY TO TEST
09-05-2025	Chris	FE	GA	Approval Ekspedisi (Send & Kurir)	Belum auto direct ke halaman Approval Ekspedisi ketika klik button Back		OK	LOW	READY TO TEST
		FE	All User	Detail	Button Back tidak berfungsi		OK	LOW	READY TO TEST
		FE	All User	Form Penilaian Vendor	Masih bisa submit form penilaian "Tanggapan Tim Procurement" tanpa mengisi penilaian	Bugs 40.mp4	OK	LOW	READY TO TEST
13-05-2025	Chris	FE	GA	Form Penilaian Vendor	Checkbox form penilaian "Tanggapan Tim Procurement" yang sudah disubmit belum greyout & masih bisa uncheck	Bugs 41.mp4	OK	LOW	READY TO TEST
		BE	User	Report	User bisa mencari semua Nama/NIK di dropdown Nama Penyerah		OK	LOW	READY TO TEST
14-05-2025	Chris	BE	User	Report	Di dropdown Nama Penerima, User hanya bisa mencari Nama/NIK yang terdata di CC User		OK	LOW	READY TO TEST
		FE	GA	Approval Ekspedisi (Link Detail)	Tidak bisa Terima barang Ekspedisi	Bugs 44.mp4	OK	MEDIUM	READY TO TEST
18-06-2025	Chris	FE	GA	Approval Ekspedisi (Kurir)	Nama Penyerah masih auto fill di field Nama Penerima (Kurir)		NOK	LOW	READY TO TEST

Gambar 3.4 Contoh *List Bug Quality Assurance*.

Gambar 3.4 menunjukkan daftar *bug* atau isu yang perlu diperbaiki oleh para *developer* sesudah selesai melewati tahap QA.

3.2 Tugas dan Uraian Kerja Magang

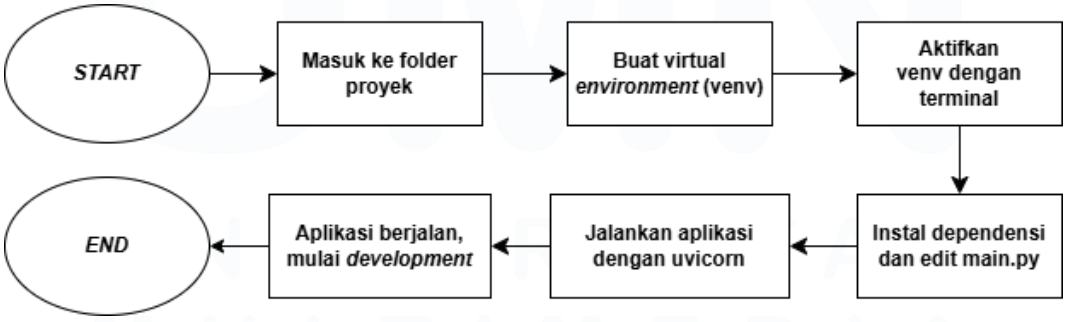
Tabel 3.1 Waktu Pelaksanaan Magang Perusahaan

Minggu ke-	Pekerjaan yang dilakukan
1-2	Mempelajari *framework* Python FastAPI untuk *service backend* dan Swagger untuk dokumentasi API.
3	Mempelajari *template* perusahaan dalam membuat *service backend* menggunakan FastAPI.
4	Memulai pengerjaan proyek Serah Terima *Online* dengan mendiskusikan ERD dan kontrak API dengan tim.
5	Membuat *database* PostgreSQL secara lokal yang dijalankan melalui Docker dan *service backend* menggunakan *template* perusahaan.
6	Mengerjakan *endpoint* GET, PUT, POST, dan DELETE *requests* untuk layanan dan item serah terima.
7	Mengerjakan *endpoint* GET dan POST *requests* untuk penilaian dan evaluasi layanan.
8	Mengerjakan *endpoint* GET *requests* untuk *reporting* dan rekap evaluasi dalam bentuk *JavaScript Object Notation* (JSON) dan Excel.
9	Mengerjakan fitur *share link* untuk halaman beserta fiturnya yang bisa diakses tanpa harus melalui perlindungan sistem *Single Sign-On* (SSO) v3 milik perusahaan.
10	Mengerjakan integrasi dengan aplikasi Ekspedisi.
11	*Deployment* ke *staging* dan membantu dalam proses integrasi API dengan tim *Frontend Developer*.
12	Mendiskusikan hasil kerja dengan mentor dan PM untuk diteruskan ke QA.
13	Melakukan perbaikan fitur yang dilaporkan oleh QA.
14	Melakukan pertemuan dengan *user* (UAT) serta melakukan perbaikan fitur yang direvisi oleh *user* saat UAT.

- Mempelajari *framework* Python FastAPI untuk *service backend* dan Swagger untuk dokumentasi API.

Pada minggu pertama dan kedua, penulis mempelajari *framework* FastAPI, yang merupakan *framework* web modern berbasis Python yang digunakan untuk membangun layanan *backend* dengan dukungan *asynchronous programming* serta integrasi dokumentasi API otomatis menggunakan Swagger OpenAPI 3.0. Dalam mempelajari FastAPI, *library* Pydantic sebagai model validasi data FastAPI juga digunakan, SQLAlchemy sebagai *Object Relational Mapping* (ORM) ke *database*, serta JWT *authentication* dengan *library* python-jose[cryptography] untuk memastikan keamanan akses API. Dokumentasi API dibuat menggunakan Swagger, namun sebagai alternatif, penulis juga menggunakan Postman untuk melakukan pengujian terhadap *request* API, terlebih terhadap *endpoint* yang tidak dibuat menggunakan *framework* FastAPI, yaitu seperti Flask.

Untuk memastikan isolasi lingkungan pengembangan, penulis mengaktifkan *Python Virtual Environment* (venv) sejak awal proyek. *Virtual environment* (venv) adalah fitur bawaan Python yang memungkinkan pengembang membuat lingkungan Python yang terisolasi, sehingga dependensi yang diinstal tidak mempengaruhi sistem global maupun proyek lain. Penulis menggunakan Windows sebagai sistem operasi utama, dan dalam proses pembuatan serta aktivasi venv, langkah-langkah berikut dilakukan seperti pada Gambar 3.5:



Gambar 3.5 Alur Pembuatan Proyek Python dengan *Virtual Environment*.

Gambar 3.5 menggambarkan proses standar dalam mempersiapkan dan menjalankan aplikasi Python menggunakan virtual *environment* serta server

Asynchronous Server Gateway Interface (ASGI) uvicorn. Proses diawali dengan masuk ke folder proyek, diikuti oleh pembuatan virtual *environment* menggunakan perintah python -m venv venv. Langkah selanjutnya adalah mengaktifkan lingkungan virtual tersebut melalui PowerShell dengan perintah .\venv\Scripts\Activate.ps1. Setelah aktivasi, seluruh instalasi dependensi seperti FastAPI atau uvicorn akan terisolasi dalam lingkungan ini, memastikan tidak terjadi konflik dengan proyek lain.

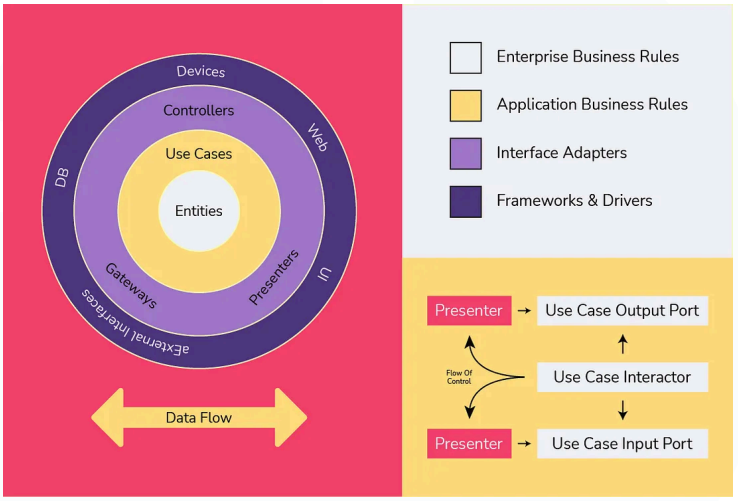
Sebelum menjalankan aplikasi, kita juga perlu membuat atau mengedit file main.py. File ini harus memuat definisi objek FastAPI yang akan dijalankan oleh uvicorn. Sebagai contoh, kita bisa mendefinisikan objek app = FastAPI() dan menetapkan minimal satu endpoint, misalnya @app.get("/") untuk mengembalikan respons sederhana. Ini penting karena perintah uvicorn main:app akan mencari file main.py dan variabel app di dalamnya.

Setelah main.py siap dan semua dependensi terpasang, aplikasi dapat dijalankan dengan perintah uvicorn main:app --host 0.0.0.0 --port 8080 --env-file .env.dev, yang akan memulai server pada port 8080 dan memuat konfigurasi dari file .env.dev. Proses ini memastikan bahwa aplikasi berjalan dalam konteks yang bersih, konsisten, dan sesuai dengan konfigurasi proyek.

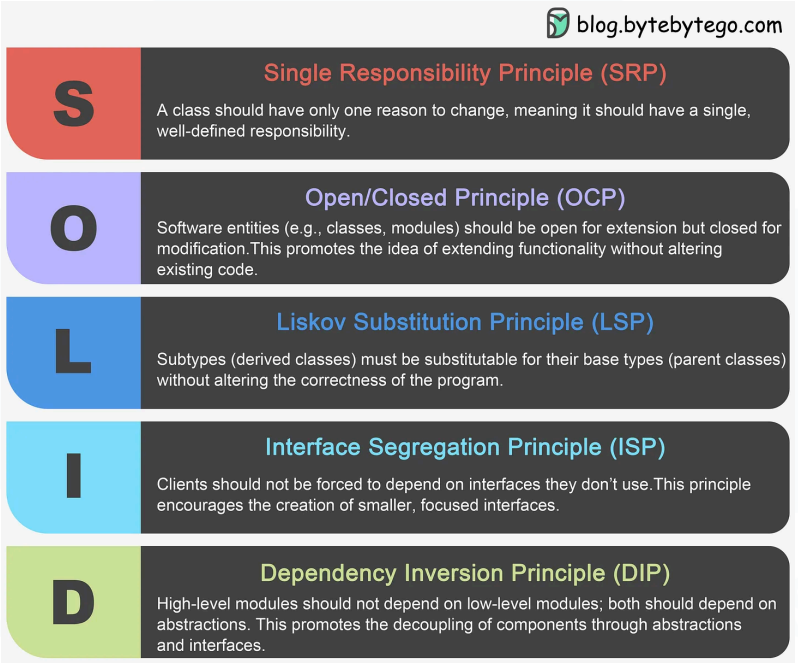
Dalam melakukan pengembangan *backend* aplikasi, penulis menggunakan Visual Studio Code (VSCode) sebagai *editor* utama. VSCode dipilih karena memiliki dukungan ekstensif untuk Python melalui ekstensi *Python by Microsoft*, yang memungkinkan *debugging*, *linting*, serta format kode secara otomatis. Selain itu, VSCode memiliki integrasi terminal yang memudahkan pengelolaan *virtual environment* dan eksekusi perintah langsung dari dalam *editor*.

Dalam dua minggu pertama ini, penulis juga mulai mempelajari prinsip *Clean Code Architecture* dan *SOLID Principles* yang diperkenalkan oleh Robert C. Martin (Uncle Bob) untuk memastikan kode yang dibuat mudah dipelihara dan dapat dikembangkan lebih lanjut tanpa banyak perubahan struktural. Kedua prinsip ini juga merupakan anjuran dari perusahaan secara langsung karena juga

merupakan fundamental yang digunakan dalam pengembangan aplikasi di perusahaan [5][6]. Gambar 3.6 dan 3.7 memberikan gambaran terhadap kedua prinsip tersebut.



Gambar 3.6 *Flow Clean Code Architecture* [5].



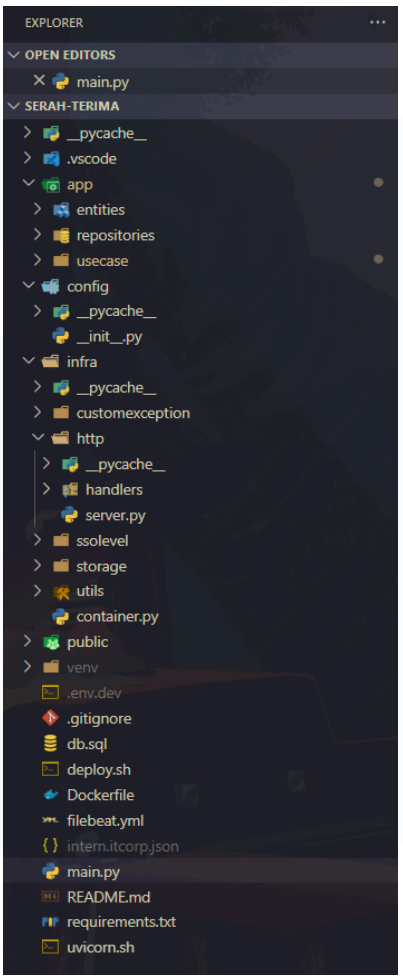
Gambar 3.7 Penjelasan SOLID *Principles* [7].

- Mempelajari *template* perusahaan dalam membuat *service backend* menggunakan FastAPI.

Pada minggu ketiga, penulis mempelajari proyek-proyek *backend* sebelumnya yang telah dikembangkan di perusahaan untuk memahami pola dan *best practice* yang diterapkan. Selain itu, penulis mengikuti pertemuan bersama tim backend developer untuk membahas *template backend service* yang baru dan telah diintegrasikan dengan *toolkit* internal kantor yang di instal di virtual *environment template*. *Toolkit* ini mencakup beberapa fitur penting seperti:

- Single Sign-On (SSO) v3, yang menangani autentikasi dan menyertakan informasi karyawan seperti NIK, *cost center*, dan *access level* dalam *payload* JWT.
- Integrasi *database* dengan *library* psycopg2, yang telah diintegrasikan dalam *template* kantor untuk menyederhanakan deklarasi dan penggunaan *query* dari berbagai jenis Sistem Manajemen Basis Data (DBMS) seperti MySQL, PostgreSQL, MongoDB, dan lain-lain.
- Standarisasi struktur proyek, yang mencakup pemisahan antara *handler* (*routes*), *use case* (*business logics*), *repository* (*database interactions*), dan *dependency injection* (*container*) agar lebih modular dan mudah diuji.

Gambar 3.8 menunjukkan struktur folder secara umum yang terdiri dari beberapa *layer*, *layer* aplikasi yang terdiri dari definisi *entities*, daftar *query* di dalam *repositories,* penerapan *business logics* dalam *usecase*, definisi *routes* dalam *handler*, dan *tools* lainnya untuk menunjang pengembangan *backend*.



Gambar 3.8 Contoh struktur folder *template* FastAPI perusahaan.

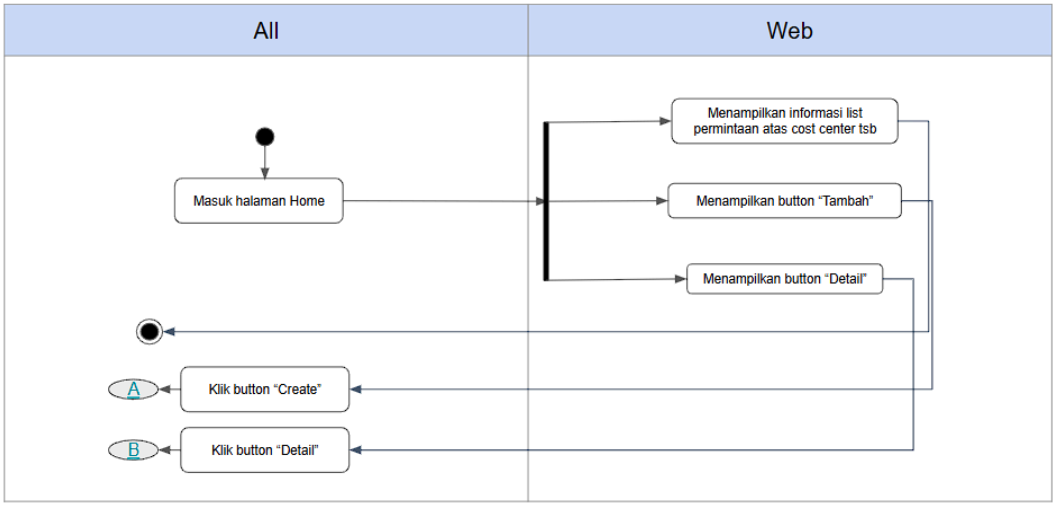
- Memulai pengerjaan proyek Serah Terima *Online* dengan mendiskusikan ERD dan kontrak API dengan tim.

Pada minggu keempat, penulis mendapatkan *briefing* dari PM terkait *Business Requirements* dan mulai mengembangkan proyek Serah Terima *Online*. Proses pertama dalam pengembangan ini adalah perancangan *database*, yang dilakukan menggunakan draw.io untuk membuat *Entity Relationship Diagram* (ERD). Setelah rancangan ERD awal dibuat, penulis melakukan sesi *review* bersama mentor untuk memastikan bahwa struktur tabel *master* dan relasinya telah sesuai dengan kebutuhan bisnis. Beberapa revisi dilakukan untuk mengakomodasi skenario tertentu, seperti penilaian dan evaluasi layanan, serta

integrasi antara karyawan, vendor, dan kategori barang. Selengkapnya dapat dilihat di Gambar 3.15.

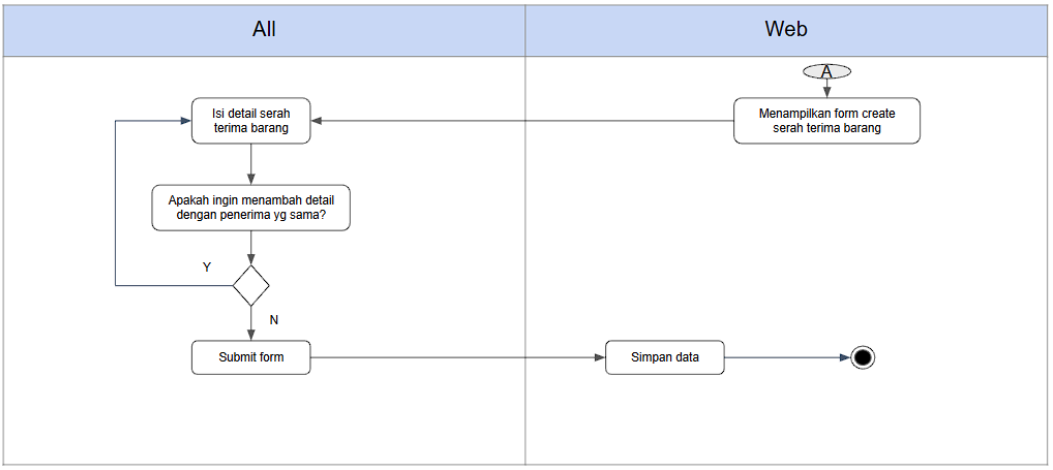
Setelah desain *database* disetujui, penulis menyusun kontrak API dalam format YAML menggunakan Swagger OpenAPI 3.0. Kontrak API ini memungkinkan tim *frontend* untuk memahami *endpoint* yang akan dikembangkan tanpa perlu menunggu *backend* selesai dibuat. Setelah kontrak API selesai disusun, penulis melakukan *review* bersama mentor kembali sebelum melanjutkan ke tahap implementasi. Selengkapnya pada Gambar 3.14.

Rangkuman *flow* bisnis dapat dilihat pada Gambar 3.9 hingga Gambar 3.13.



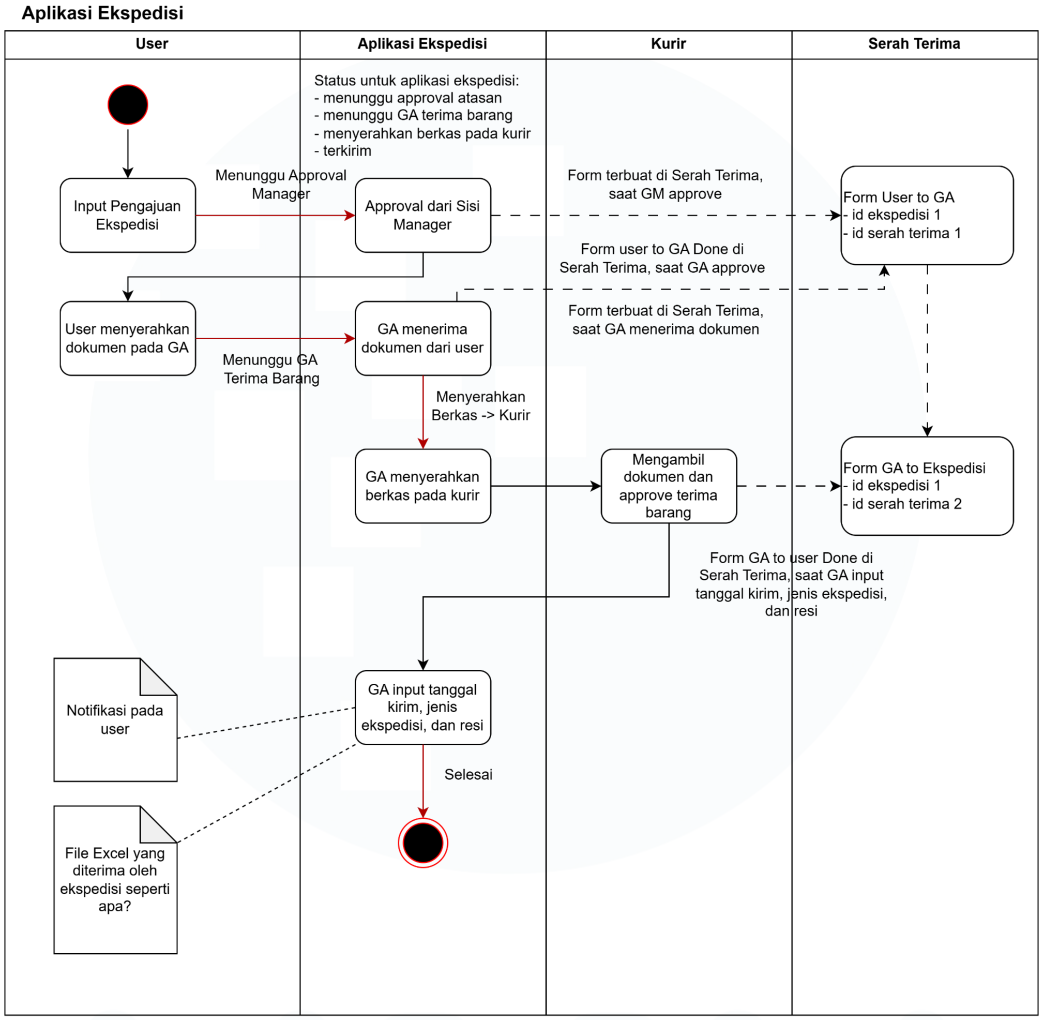
Gambar 3.9 Contoh *Use Case Home* Serah Terima *Online*.

Gambar 3.9 menjelaskan bahwa *users* setelah login dapat melihat daftar layanan serah terima yang dikirimkan atau diterima oleh *cost center user* tersebut. *Users* juga dapat membuat atau melengkapi layanan serah terima miliknya sendiri atau mewakili rekan *cost center* dengan klik *button* yang sesuai untuk berpindah halaman.



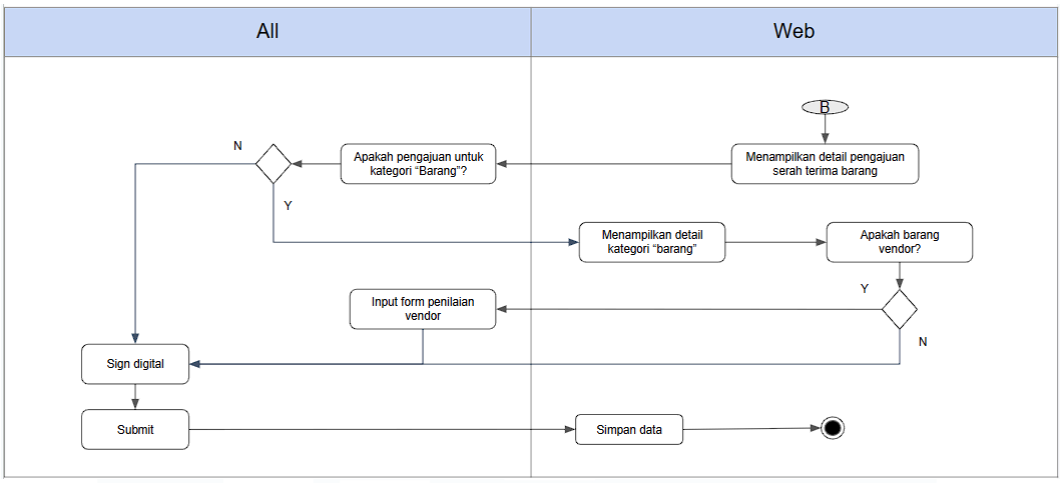
Gambar 3.10 Contoh *Use Case Create* Serah Terima *Online*.

Gambar 3.10 menunjukkan bahwa *users* yang ingin membuat layanan serah terima harus mengisi beberapa detail, termasuk detail *item* yang ingin dikirim. Total *item* yang dapat diakomodasi dalam satu layanan serah terima adalah maksimal 5 *items* dengan 3 jenis *item* berbeda, yaitu *item* (*vendor*/non-*vendor*), uang tunai, dan dokumen. Setiap *item* dapat diberikan foto barang terlebih dahulu yang sifatnya opsional, sehingga orang yang menerima layanan serah terima nantinya bisa mengantisipasi *item* yang sama saat menerima barangnya.



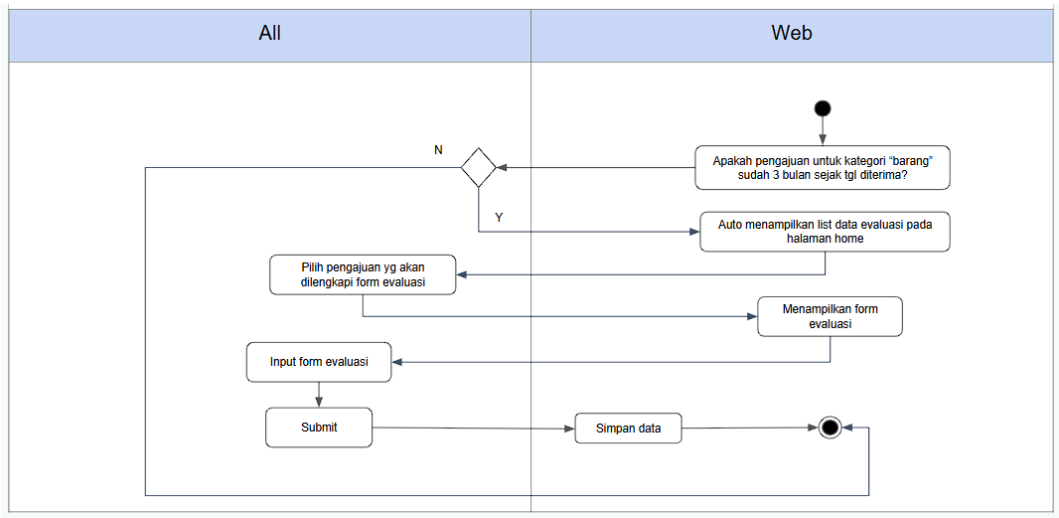
Gambar 3.11 Contoh *Use Case Auto Create Sync* dari *Dashboard* Ekspedisi.

Gambar 3.11 secara khusus menjelaskan *flow* integrasi barang ekspedisi yang dilakukan di situs web ekspedisi, yaitu situs web berbeda dengan Serah Terima *Online* yang lebih berfokus pada *flow* ekspedisi barang. *Users* yang sudah membuat layanan ekspedisi sebelumnya, akan tampil daftar layanannya di halaman Serah Terima khusus GA untuk ditindak lanjuti. Barang dari ekspedisi ini nantinya harus diserahkan ke GA terlebih dahulu, lalu jika sudah selesai melengkapi detailnya, akan terbuat layanan serah terima baru secara otomatis untuk GA dapat menyerahkan barang ekspedisi tersebut kepada kurir.



Gambar 3.12 Contoh *Use Case* Detail Layanan Serah Terima *Online*.

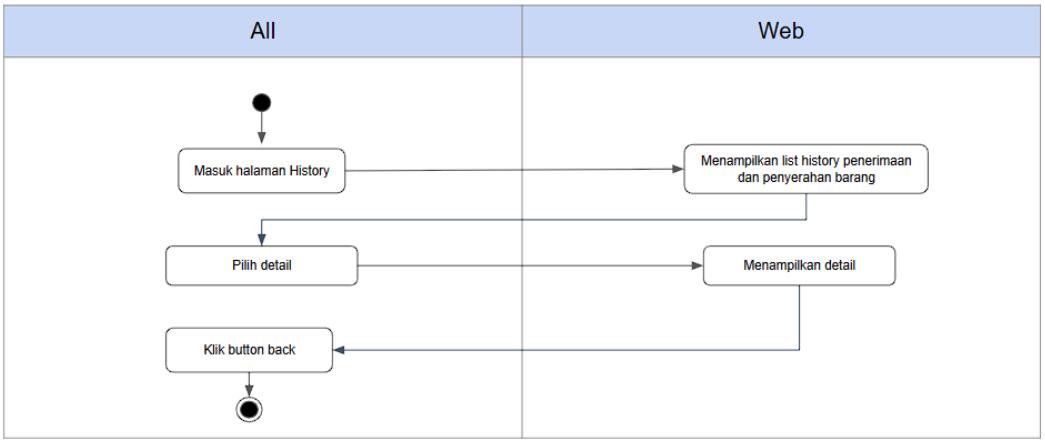
Gambar 3.12 menjelaskan jika halaman detail untuk layanan tertentu dikunjungi. Halaman akan menampilkan detail tersebut dan dapat dilengkapi detailnya. Detail tersebut meliputi *item* yang berkategori barang dengan kondisi barang tersebut apakah barang *vendor* atau bukan. Jika barang tersebut merupakan barang *vendor*, maka penilaian terhadap barang tersebut harus dilakukan terlebih dahulu, lalu baru bisa melakukan finalisasi layanan setelah menyertakan bukti foto penyerahan serta tanda tangan digital.



Gambar 3.13 Contoh *Use Case Form* Evaluasi Serah Terima *Online*.

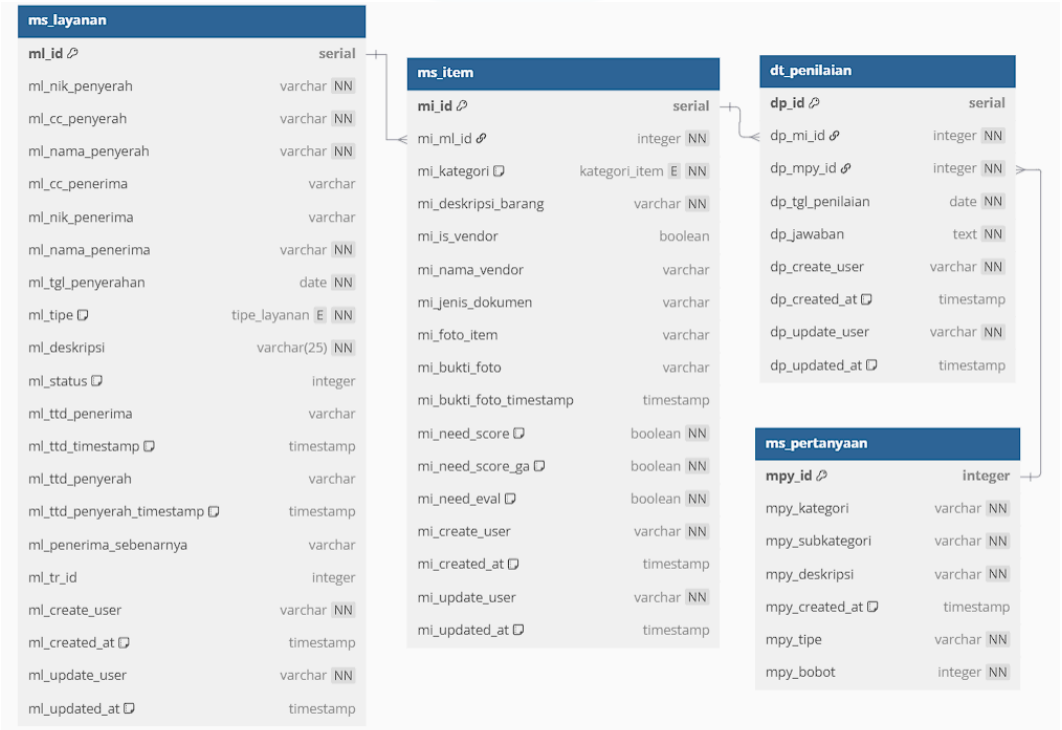
Gambar 3.13 menjelaskan bahwa sesudah 3 bulan sejak sebuah layanan dengan *item* barang *vendor* selesai difinalisasi, maka pada halaman *Home* GA

akan secara otomatis menampilkan *form* evaluasi terhadap barang tersebut. *Form* tersebut meliputi 2 pertanyaan *True or False* dan Kritik dan Saran.



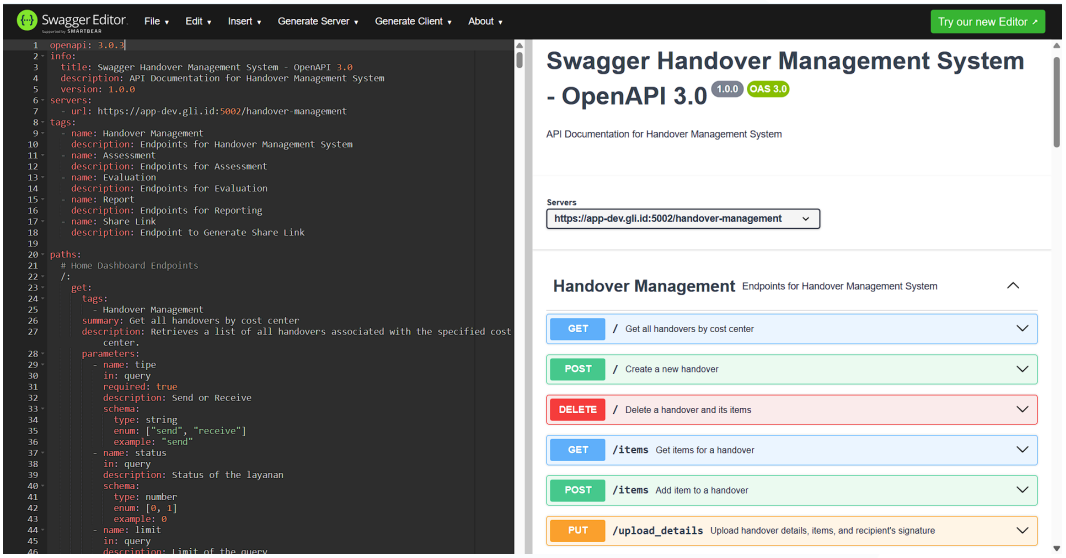
Gambar 3.14 Contoh *Use Case Report* dan *History* Serah Terima *Online*.

Gambar 3.14 menjelaskan bahwa terdapat halaman *History* untuk melihat semua layanan yang sudah diselesaikan/difinalisasi. Layanan tersebut juga masih dapat dilihat detailnya.



Gambar 3.15 Contoh ERD Serah Terima *Online*.

Gambar 3.15 merupakan ERD final yang saat ini digunakan dalam aplikasi Serah Terima *Online*.



Gambar 3.16 Contoh Kontrak API Serah Terima *Online*.

Gambar 3.16 merupakan gambaran kontrak API yang dikerjakan penulis untuk dijadikan acuan dalam membuat *backend services* Serah Terima *Online*.

- Membuat database PostgreSQL secara lokal yang dijalankan melalui Docker dan *service backend* menggunakan template perusahaan.

Pada minggu kelima, setelah desain database dalam ERD disetujui, penulis mulai merealisasikannya dengan membuat *database* secara lokal menggunakan DBeaver sebagai *database client*. Penulis mengatur *database* menggunakan Docker, mencoba dua metode, yaitu docker run dan docker compose, untuk memahami perbedaannya. Akhirnya, dipilih metode docker run sebab kontainer yang ingin dijalankan hanya 1 saja, docker run dijalankan untuk membuat *image* Docker berisikan PostgreSQL versi 12 dan port 5432. Identitas database dibuat langsung melalui *Command Line Interface* (CLI).

Untuk implementasi aksi *Create, Read, Update, and Delete* (CRUD), penulis menggunakan psycopg2, yang telah di-*wrap* dalam *toolkit* internal kantor. Implementasi baik untuk CRUD dan database yang dilakukan secara lokal ini agar

tidak terlalu banyak revisi yang dilakukan jika sudah masuk fase *staging*, terlebih akses database akan terbatas untuk *user* magang. Contoh *query* yang dilakukan menggunakan *toolkit* ada pada Gambar 3.17.

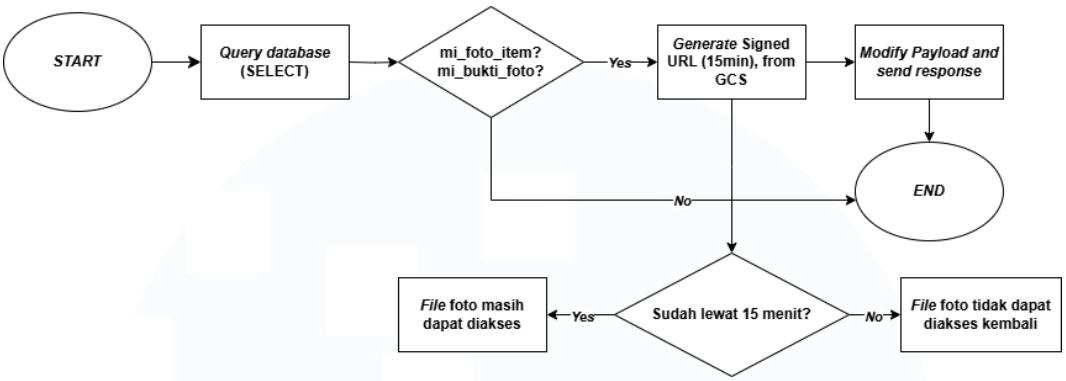
```
def select_by_id(self, ml_id: int) -> entities.Layanan:
    try:
        sql = """
        SELECT ml_id, ml_nama_penyerah, ml_nama_penerima, ml_tipe, ml_deskripsi, ml_tgl_penyerahan, ml_status
        FROM serah_terima.ms_layanan
        WHERE ml_id = %s
        """

        layanan = self.__db.getConnection().selectWithBindOutput(
            sql,
            entities.Layanan,
            ml_id,
        )
```

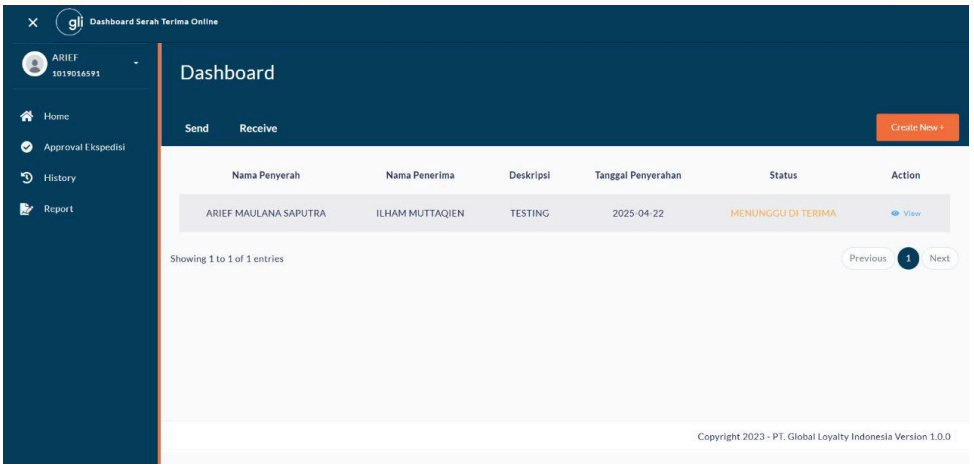
Gambar 3.17 Contoh *Query Database* dengan *Toolkit*.

- Mengerjakan *endpoint* GET, PUT, POST, dan DELETE *requests* untuk layanan dan item serah terima.

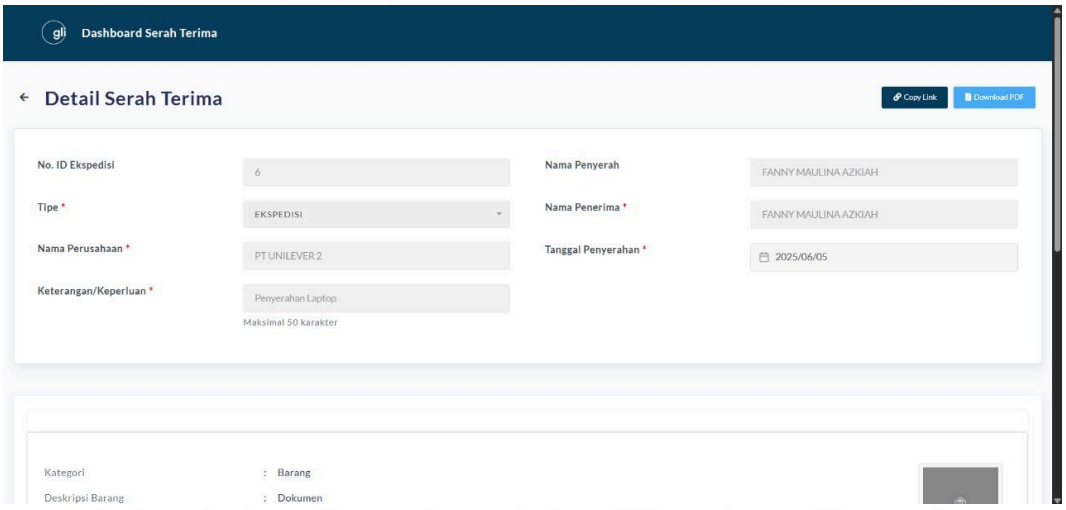
Pada minggu keenam, pengembangan *endpoint* utama untuk fitur Serah Terima, yaitu GET, POST, PUT, dan DELETE mulai dikerjakan. Untuk *endpoint* GET, dua skenario disiapkan: pertama, untuk mengambil seluruh layanan serah terima berdasarkan *cost center*; dan kedua, untuk mengambil detail dari layanan tertentu berdasarkan ID layanan. Pada endpoint ini, penulis juga menerapkan penggunaan Signed URL dari Google Cloud Storage (GCS), yang memungkinkan file gambar dikembalikan melalui URL yang aman dan hanya dapat diakses dalam jangka waktu tertentu. Alur kerja GET *request* untuk layanan dan detailnya yang melibatkan GCS untuk membuat Signed URL dapat dilihat pada Gambar 3.18.



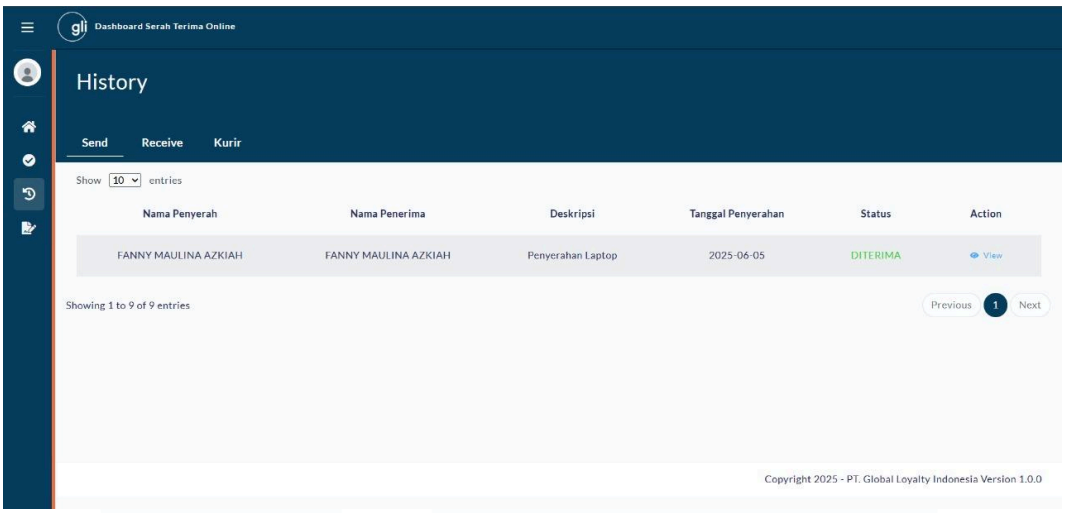
Gambar 3.18 *Flowchart* GET *Request* Layanan Serah Terima *Online*.



Gambar 3.19 Halaman *Dashboard* Serah Terima *Online*.



Gambar 3.20 Halaman Detail Layanan Serah Terima *Online*.

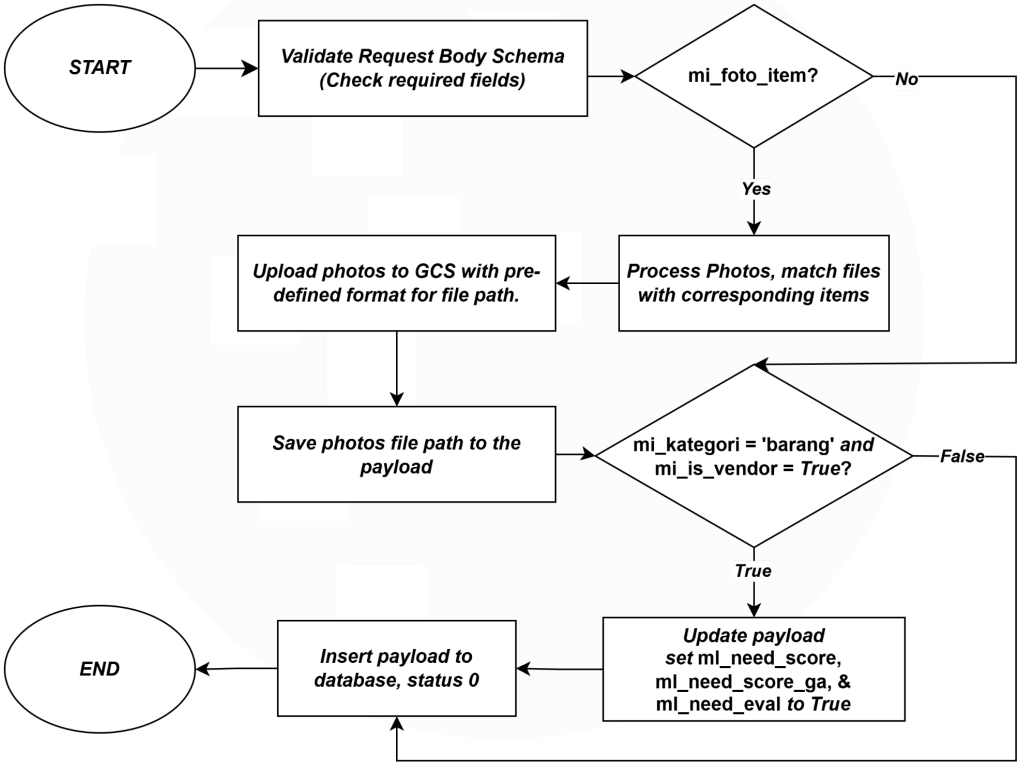


Gambar 3.21 Halaman *History* Layanan Serah Terima *Online*.

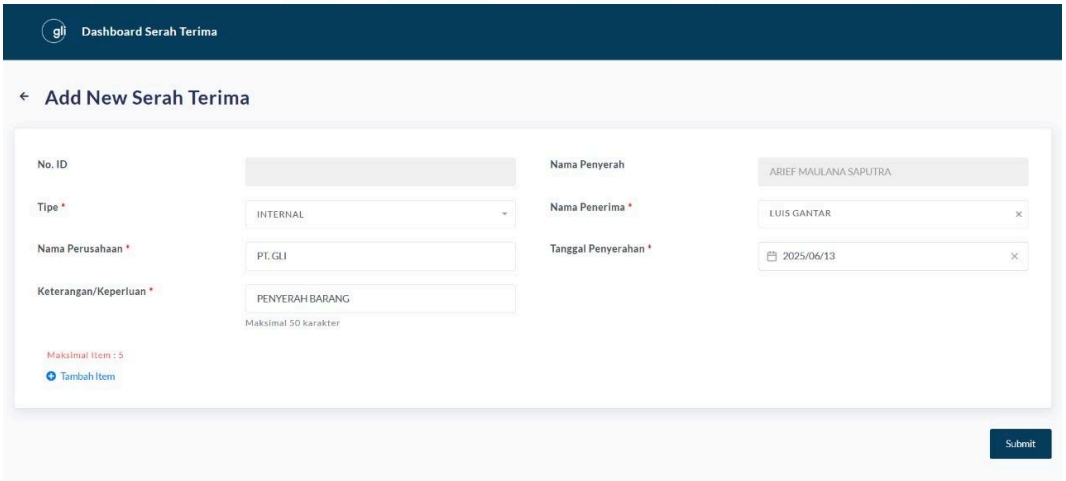
Gambar 3.19, Gambar 3.20, dan Gambar 3.21 merupakan tampilan halaman dari *frontend* berupa halaman *Dashboard,* detail, dan *History* layanan yang menggunakan data yang didapat dari *endpoint* GET *request* dengan mengatur parameter yang sesuai agar mendapat data yang sesuai pula.

Untuk *endpoint* POST, logika pembuatan layanan baru beserta daftar *item* serah terima yang terkait diimplementasikan. Pada proses ini, *body* data yang dikirim, dikodekan menggunakan multipart/form-data mencakup informasi layanan dan *items* terkait. *Items* tersebut dapat menyertakan foto *item* (opsional) yang akan diunggah ke GCS, serta metadata terkait lainnya. Validasi gambar dilakukan menggunakan *library* Pillow, dengan ketentuan maksimum 5MB per file dan format JPG, PNG, serta JPEG, maksimal hingga 5 foto per layanan. Setelah data berhasil tervalidasi, foto akan diproses dengan mencocokkan foto-foto yang diunggah terhadap *items* pada layanan tersebut. Pencocokkan dilakukan agar *file path* yang sesuai dapat disimpan ke *database* sesuai dengan ID *items* layanan terkait. Foto-foto kemudian akan diunggah lebih lanjut ke GCS dengan format layananid_UUID(16).format agar menghindarkan dari potensi kesamaan nama file di kemudian hari. Namun, jika tidak ada foto yang diunggah, maka langsung proses berakhir tanpa operasi file foto. Layanan yang berhasil

dibuat otomatis berstatus ‘0’ alias ‘Menunggu diterima’. Detail langkah *create* layanan dapat dilihat pada Gambar 3.22.

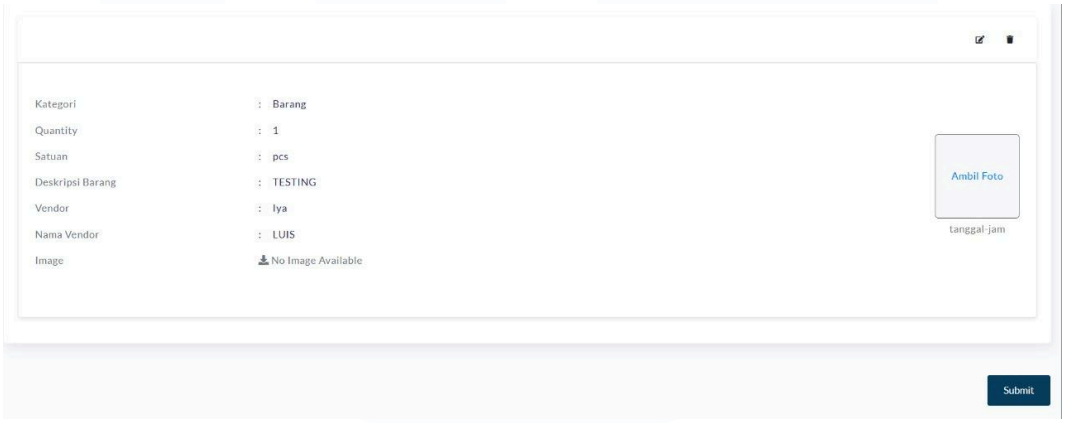


Gambar 3.22 *Flowchart* POST *Request* Layanan Serah Terima *Online*.



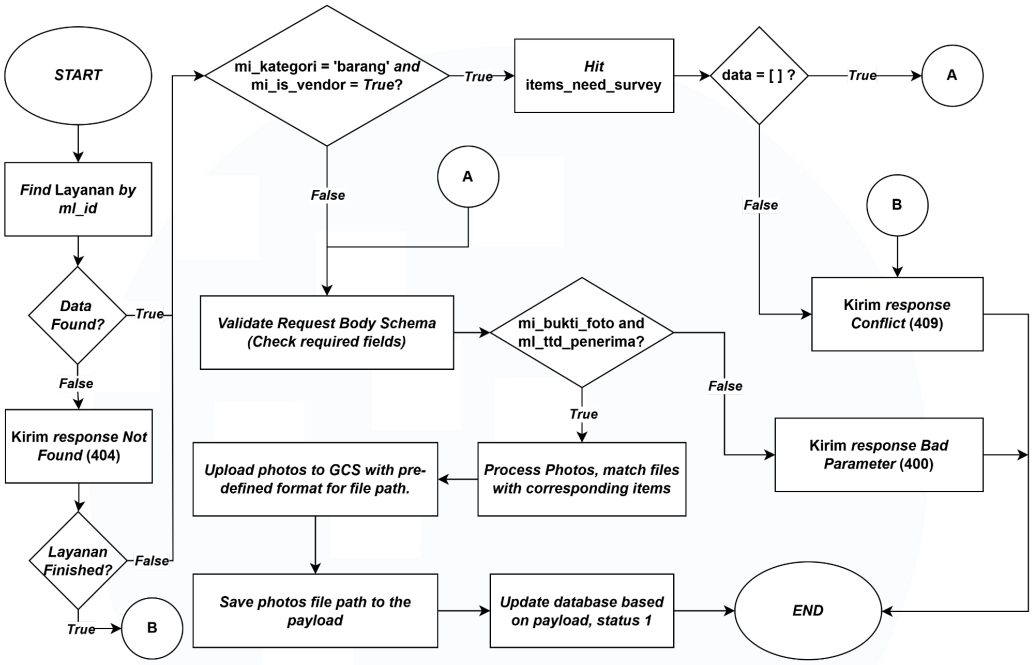
Gambar 3.23 Halaman *Create* Layanan Serah Terima *Online*.

Gambar 3.24 Menu Tambah Barang/*Item* Layanan Serah Terima *Online*.



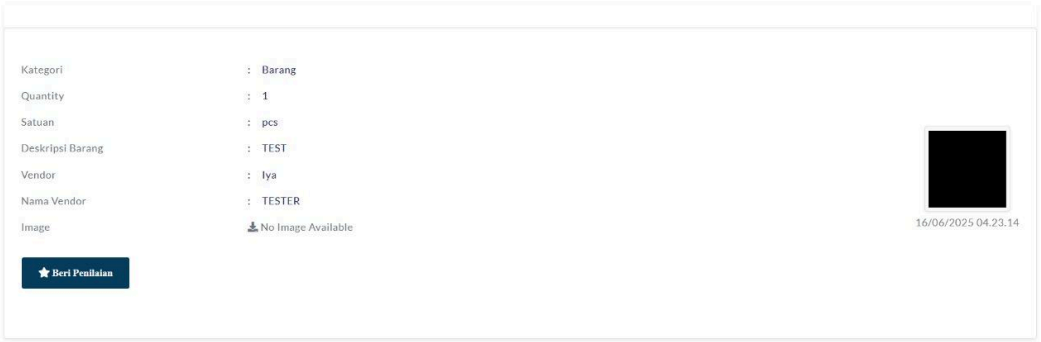
Gambar 3.25 Tampilan Menu Setelah Tambah Barang/*Item*.

Gambar 3.23 hingga Gambar 3.25 memberikan gambaran ketika *users* akan mengisi beberapa data dalam *form* yang akan dikirim *payload*-nya ke API POST. Gambar 3.24 secara spesifik memberikan gambaran tambah *item* pada layanan terkait. Pertama, *users* memilih kategori yang ada (barang, dokumen, dan uang tunai), kemudian mengisi *form* pada menu *pop-up* yang ada, lalu simpan. Jika berhasil ditambahkan, maka akan muncul tampilan seperti Gambar 3.25 pada halaman detail layanan.

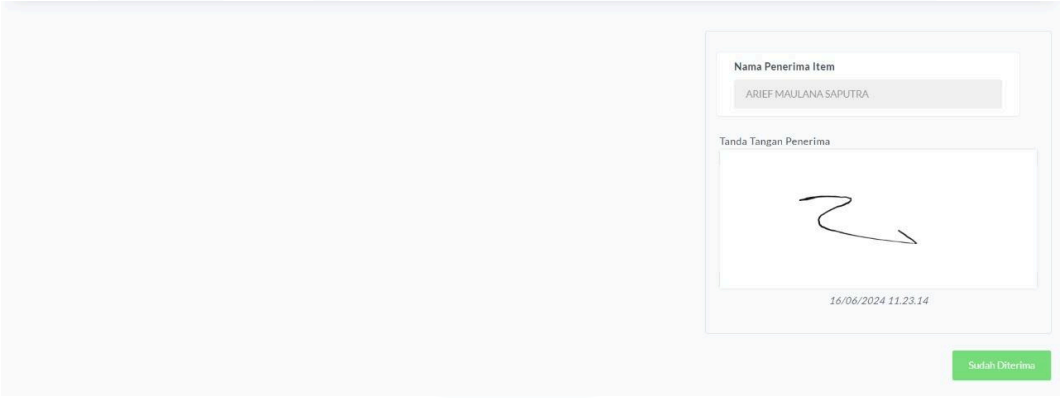


Gambar 3.26 *Flowchart* PUT *Request* Layanan Serah Terima *Online*.

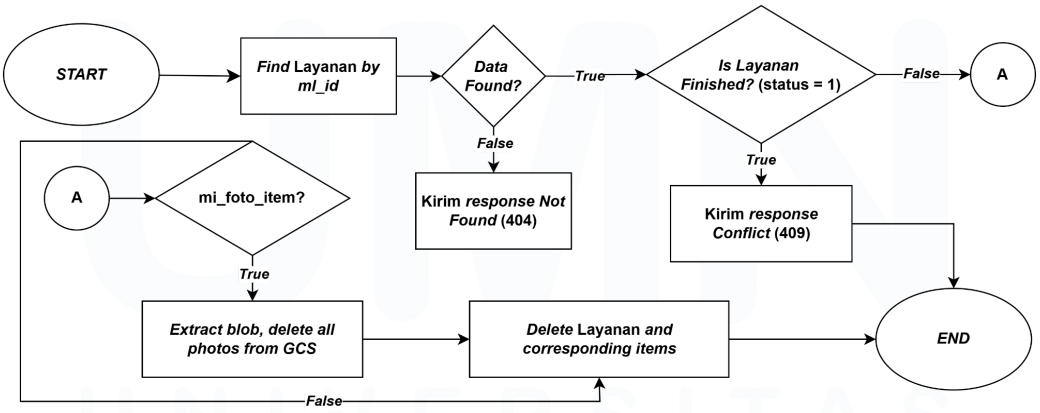
Pada *endpoint* PUT, fitur pembaruan data layanan diimplementasikan mirip dengan *endpoint* POST, hanya beberapa data yang tidak bisa diperbarui/diubah. *Endpoint* ini menerima parameter yang berisi pembaruan terhadap detail layanan dan *item*, termasuk diantaranya *update*/melengkapi foto-foto bukti serah terima serta tanda tangan penerima yang digambar langsung melalui *frontend* dan dikirim sebagai *file* gambar. Setiap perubahan akan memicu proses validasi ulang dan re-*upload* file baru ke GCS dengan *file path* yang baru ditambahkan. PUT *request* menganut *flow* sistem yang digambarkan pada Gambar 3.26. Untuk gambaran proses pengisian detail layanan untuk keperluan *update* layanan dapat dilihat pada Gambar 3.26 dan Gambar 3.27. Catatan, terkhusus pada tipe *item* barang *vendor*, maka sebelum dapat menyelesaikan layanan, harus mengisi penilaian terlebih dahulu. Akhir dari *flow update* adalah dengan mengubah status layanan dari belum selesai (diwakili di *database* dengan angka 0) ke selesai (diwakili di *database* dengan angka 1).



Gambar 3.26 Menu *Update* Layanan Serah Terima *Online*.



Gambar 3.27 Menu Tanda Tangan Layanan Serah Terima *Online*.

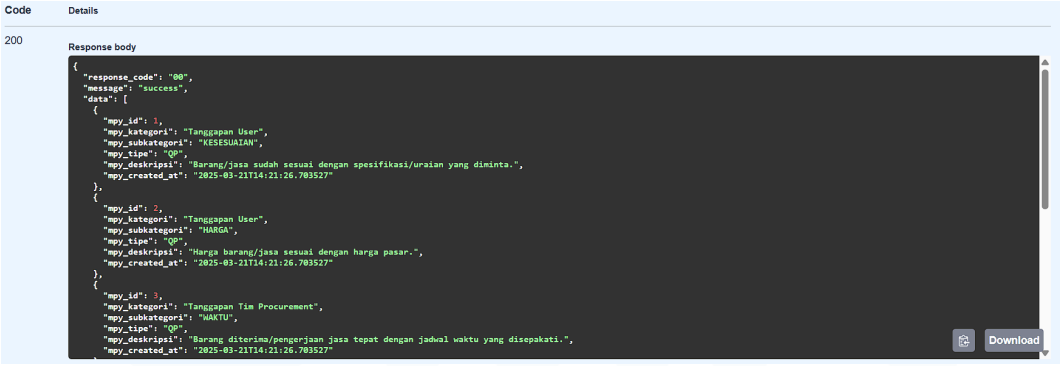


Gambar 3.28 *Flowchart* DELETE *Request* Layanan Serah Terima *Online*.

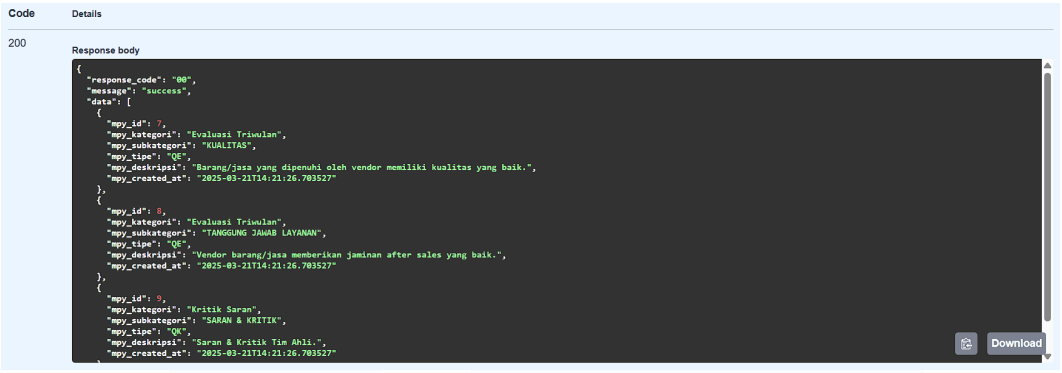
Terakhir, Gambar 3.28 menjelaskan *endpoint* DELETE, fitur untuk menghapus layanan beserta *items* terkait secara permanen. Tak hanya itu, jika layanan tersebut mengandung foto yang diunggah ke GCS, akan dihapus juga.

- Mengerjakan *endpoint* GET *requests* untuk *reporting* dan rekap evaluasi dalam bentuk *JavaScript Object Notation* (JSON) dan Excel.

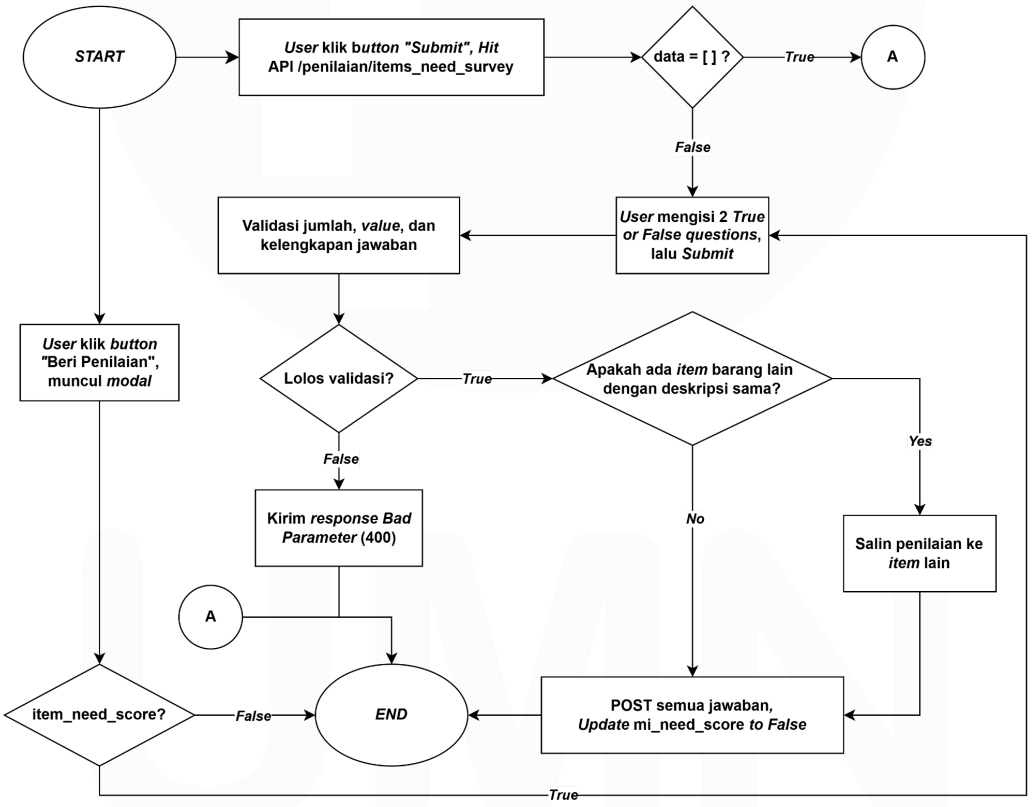
Pada minggu ketujuh, pengembangan *endpoint* untuk menyimpan dan menampilkan hasil penilaian *user* terhadap layanan yang berkategori barang dan *vendor* dikerjakan. Penilaian berisi 2 pertanyaan untuk *user* dan 4 pertanyaan untuk tim *Procurement* yang wajib diisi. Semua pertanyaan berupa pertanyaan *True or False* dan terkhusus untuk 4 pertanyaan tim *Procurement*, akan muncul di halaman *Home* tim *Procurement* setelah layanan diselesaikan oleh *user.* Pertanyaan-pertanyaan yang muncul pada setiap *form* penilaian atau evaluasi seperti pada Gambar 3.32 dan Gambar 3.35 diambil dari *database* via *endpoint* GET /penilaian atau /evaluasi untuk menjaga kesatuan data dan memudahkan pemeliharaan data jika nantinya ada perubahan. Sampel *response* dari kedua *endpoint* tersebut dapat dilihat pada Gambar 3.29 dan 3.30. Termasuk ketika *user* ingin melihat riwayat penilaian untuk barang-barang *vendor* di kemudian hari, maka data tersebut tetap tersimpan di *database* dan dapat dilihat jawabannya dengan tepat dan sesuai saat mengisi penilaian sebelumnya.



Gambar 3.29 Contoh Pertanyaan Penilaian.



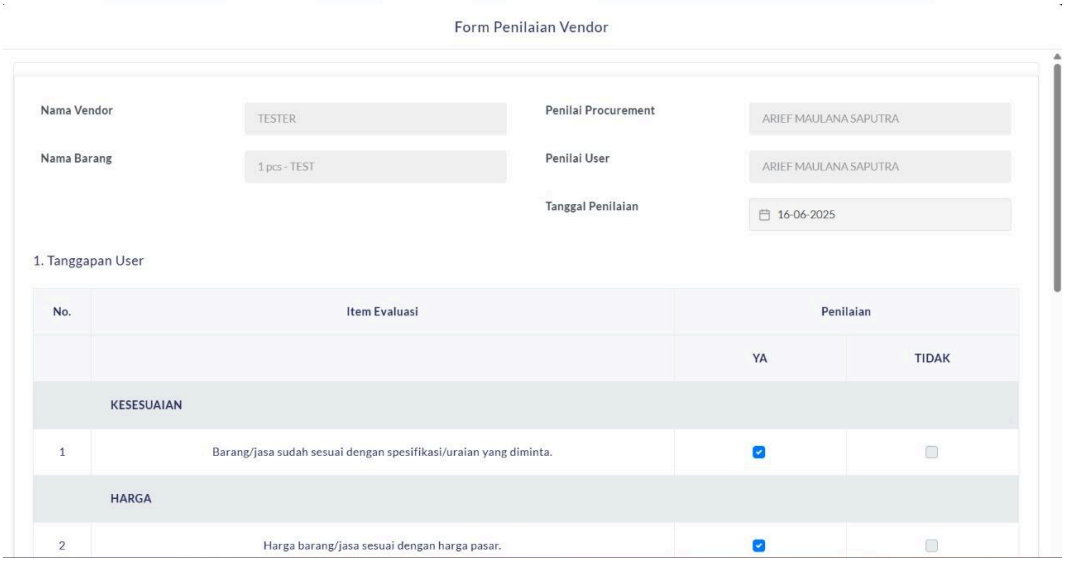
Gambar 3.30 Contoh Pertanyaan Evaluasi.



Gambar 3.31 Alur Penilaian *User*.

Gambar 3.31 memberikan gambaran alur *user* dalam memberikan penilaian terhadap *item* berkategori barang *vendor*. Validasi dilakukan untuk jawaban agar data yang akan dimasukkan ke *database* valid. Sebagai tambahan, setiap ada *item* barang yang memiliki deskripsi sama, maka satu penilaian mewakili barang lainnya. *Flag* status *item* mi_need_score untuk barang-barang

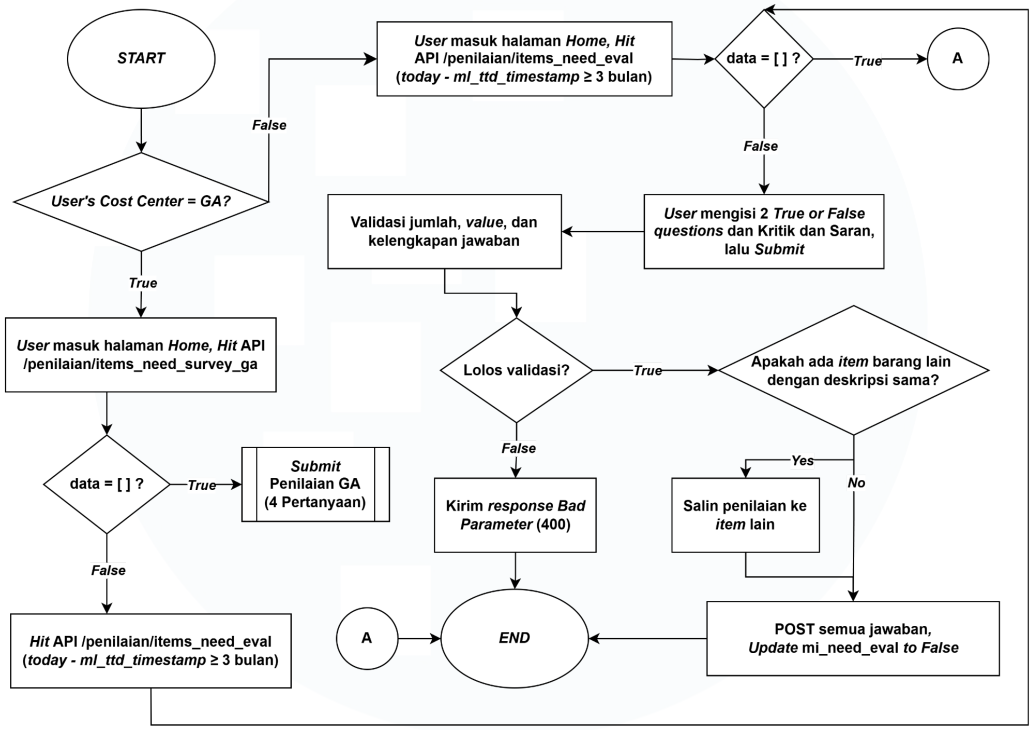
terkait akan diubah ke *False*, menandakan barang tersebut sudah selesai dinilai dan menunggu penilaian dari tim *Procurement.* Tim *Procurement* dapat kemudian memberikan penilaian untuk 4 pertanyaan sisanya di Halaman *Home*. Di halaman tersebut akan muncul *modal form* sama seperti Gambar 3.32 secara otomatis apabila *pooling* terhadap *endpoint* /penilaian/items_need_survey_ga mengembalikan respon data yang perlu dinilai. *Flow* terkait mekanisme *pooling* penilaian ini pada halaman GA dapat digambarkan melalui Gambar 3.33.



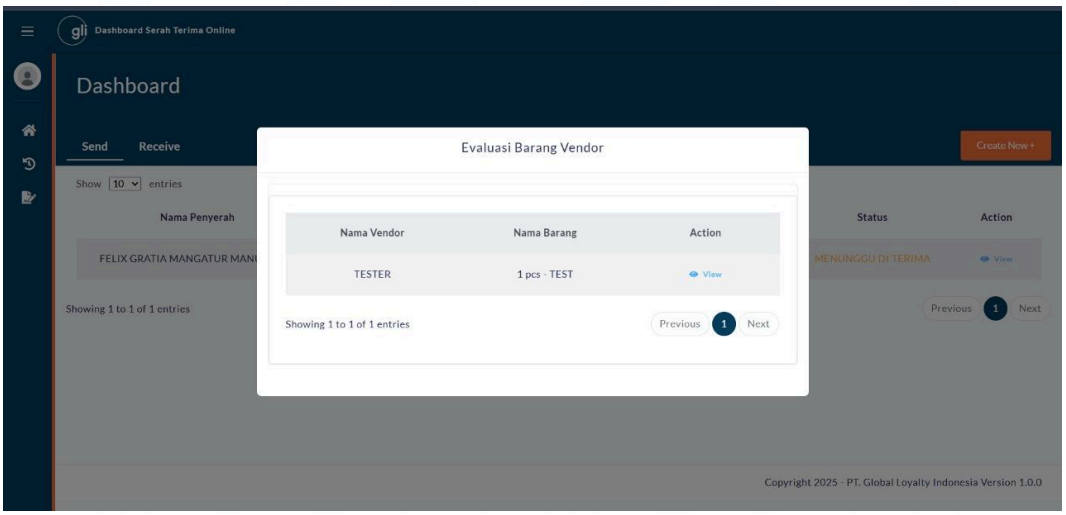
Gambar 3.32 Form Penilaian Barang Vendor.

Selanjutnya ada pengembangan *endpoint* evaluasi layanan, dimana evaluasi adalah hal yang wajib dilakukan oleh *users* tiga bulan setelah submit tanda tangan dan bukti foto layanan. Pertanyaan yang terdapat pada *form* evaluasi berbeda dari *form* penilaian dan terdapat tempat isian untuk Kritik dan Saran. Evaluasi akan muncul di halaman masing-masing *user* yang pernah membuat layanan berisi barang *vendor* dengan cara *frontend* melakukan *pooling* terhadap *endpoint* GET items_need_eval seperti pada. *User* tidak dapat lanjut ke halaman utama jika belum mengisi kewajiban evaluasi yang ada. *Flow* terkait mekanisme *pooling*, sekaligus pengisian evaluasi pada halaman setiap *user* dapat digambarkan melalui Gambar 3.33. Sedangkan, tampilan yang muncul di *frontend*

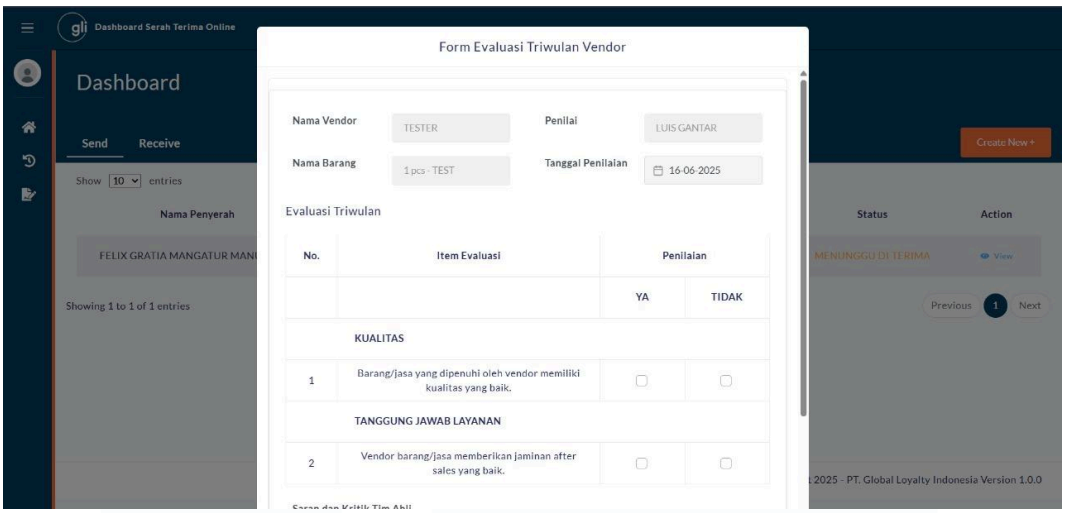
adalah seperti Gambar 3.34 dan apabila *action* untuk layanan tersebut dipilih, maka akan muncul *form* seperti Gambar 3.35.



Gambar 3.33 *Flow Pooling Items* Butuh Penilaian dan Evaluasi.



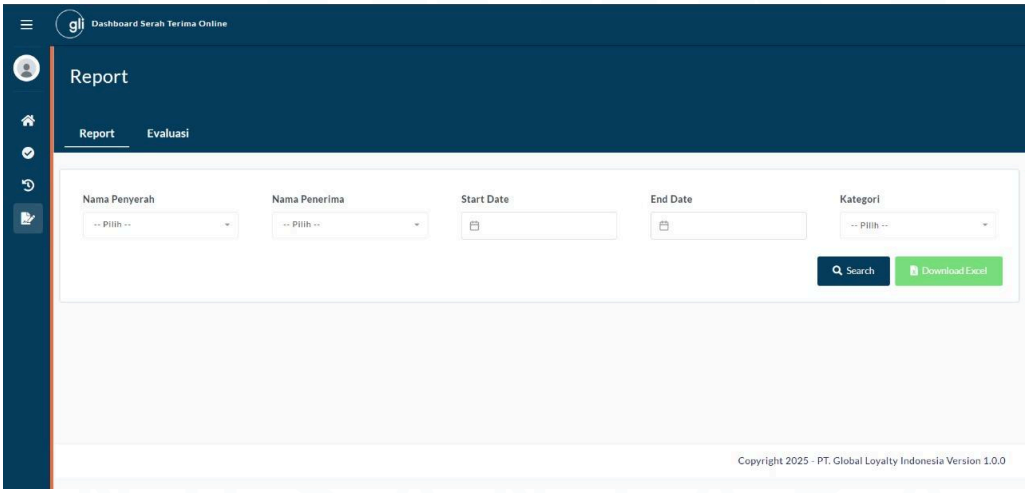
Gambar 3.34 Form Wajib Isi Evaluasi Pasca 3 Bulan Barang Vendor.



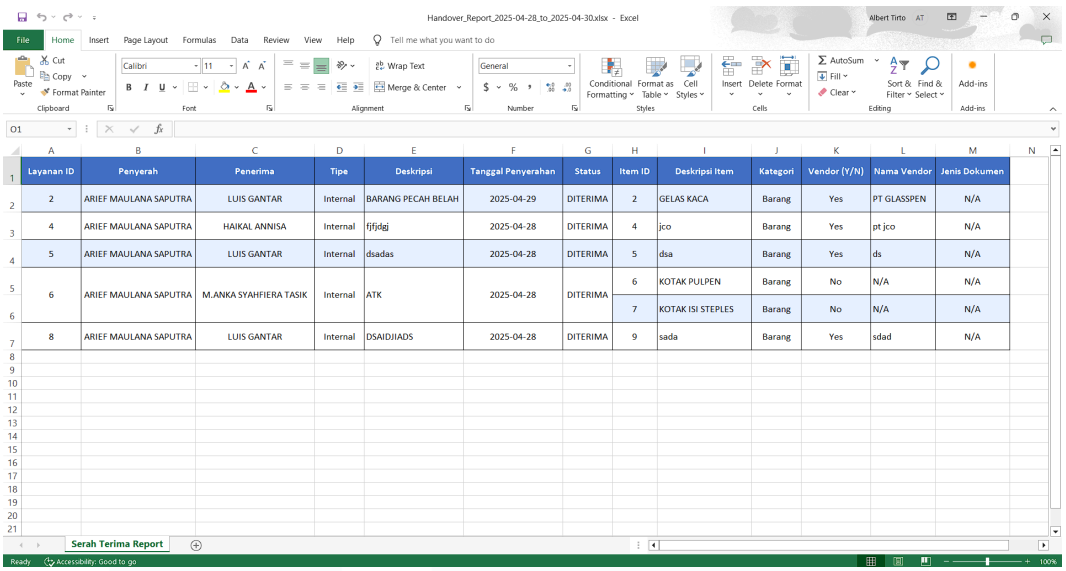
Gambar 3.35 Form Evaluasi Barang *Vendor* Pasca 3 Bulan.

- Mengerjakan *endpoint* GET *requests* untuk *reporting* dan rekap evaluasi dalam bentuk JSON dan Excel.

Fitur *endpoint* GET *request* untuk *reporting* dan rekap evaluasi dalam bentuk JSON dan penambahan fitur *export* data ke format Excel menggunakan *library* openpyxl dikerjakan pada minggu ini. *Endpoint* GET *request* ini memiliki parameter-parameter tertentu sebagai filter seperti rentang tanggal penyerahan, kategori, nama penyerah, dan nama penerima. Halaman *reporting* dan hasil generasi *file* Excel dapat dilihat pada Gambar 3.36 dan Gambar 3.37.



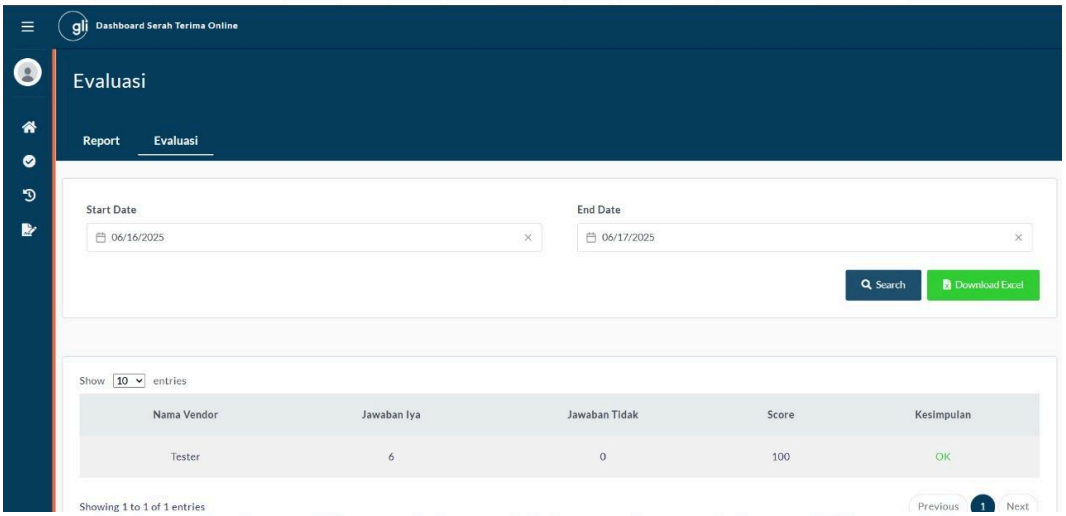
Gambar 3.36 Halaman *Report*.



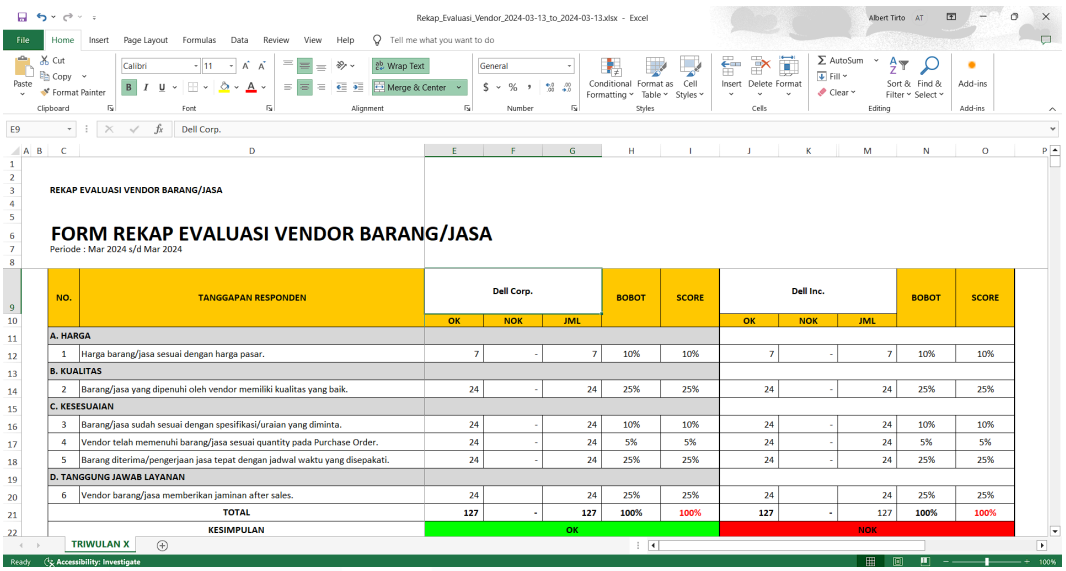
Layanan ID	Penyerah	Penerima	Tipe	Deskripsi	Tanggal Penyerahan	Status	Item ID	Deskripsi Item	Kategori	Vendor (Y/N)	Nama Vendor	Jenis Dokumen
2	ARIEF MAULANA SAPUTRA	LUIS GANTAR	Internal	BARANG PECAH BELAH	2025-04-29	DITERIMA	2	GELAS KACA	Barang	Yes	PT GLASSPEN	N/A
4	ARIEF MAULANA SAPUTRA	HAIKAL ANNISA	Internal	fjfjdgj	2025-04-28	DITERIMA	4	jco	Barang	Yes	pt jco	N/A
5	ARIEF MAULANA SAPUTRA	LUIS GANTAR	Internal	dsadas	2025-04-28	DITERIMA	5	dsa	Barang	Yes	ds	N/A
6	ARIEF MAULANA SAPUTRA	M.ANKA SYAHFIERA TASIK	Internal	ATK	2025-04-28	DITERIMA	6	KOTAK PULPEN	Barang	No	N/A	N/A
							7	KOTAK ISI STEPLES	Barang	No	N/A	N/A
8	ARIEF MAULANA SAPUTRA	LUIS GANTAR	Internal	DSAIDJIADS	2025-04-28	DITERIMA	9	sada	Barang	Yes	sdad	N/A

Gambar 3.37 Contoh File Excel *Reporting*.

Fitur *report* seperti di atas dapat dilakukan oleh semua *users*. Namun, untuk fitur rekap evaluasi hanya dapat dilihat atau dilakukan oleh *user* GA saja. Fitur rekap evaluasi ditujukan untuk menampilkan hasil evaluasi yang sudah pernah dilakukan terhadap barang-barang *vendor* yang sudah selesai serah terima. Detail yang muncul meliputi bobot per pertanyaan, total nilai, kesimpulan skor, nama *vendor*, dll. Detail tersebut dapat dilihat pada Gambar 3.39 yang dapat digenerasi melalui halaman *report* evaluasi seperti pada Gambar 3.38.



Gambar 3.38 Halaman Rekap Evaluasi Khusus GA dan *Procurement*.



REKAP EVALUASI VENDOR BARANG/JASA

FORM REKAP EVALUASI VENDOR BARANG/JASA

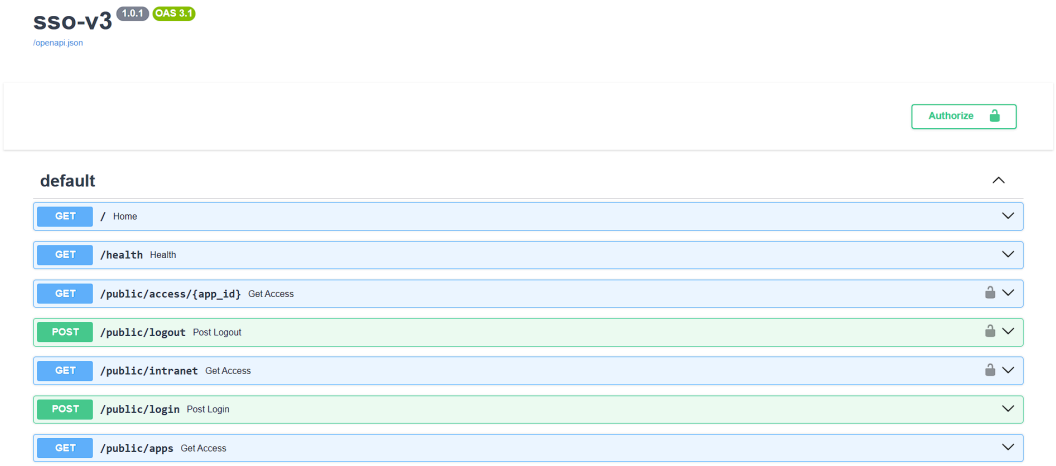
Periode : Mar 2024 s/d Mar 2024

NO.	TANGGAPAN RESPONDEN	Dell Corp.			BOBOT	SCORE	Dell Inc.			BOBOT	SCORE
		OK	NOK	JML			OK	NOK	JML		
A. HARGA											
1	Harga barang/jasa sesuai dengan harga pasar.	7	-	7	10%	10%	7	-	7	10%	10%
B. KUALITAS											
2	Barang/jasa yang dipenuhi oleh vendor memiliki kualitas yang baik.	24	-	24	25%	25%	24	-	24	25%	25%
C. KESESUAIAN											
3	Barang/jasa sudah sesuai dengan spesifikasi/uraian yang diminta.	24	-	24	10%	10%	24	-	24	10%	10%
4	Vendor telah memenuhi barang/jasa sesuai quantity pada Purchase Order.	24	-	24	5%	5%	24	-	24	5%	5%
5	Barang diterima/pengerjaan jasa tepat dengan jadwal waktu yang disepakati.	24	-	24	25%	25%	24	-	24	25%	25%
D. TANGGUNG JAWAB LAYANAN											
6	Vendor barang/jasa memberikan jaminan after sales.	24		24	25%	25%	24		24	25%	25%
TOTAL		127	-	127	100%	100%	127	-	127	100%	100%
KESIMPULAN		OK					NOK				

Gambar 3.39 Contoh File Excel Rekap Evaluasi GA dan *Procurement*.

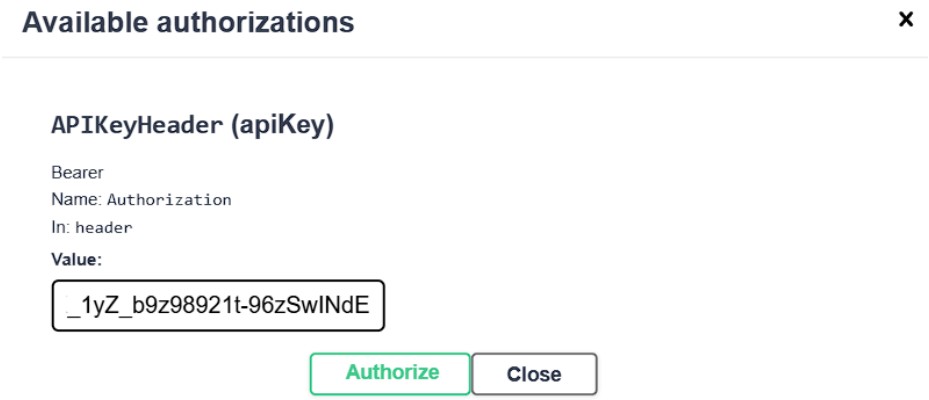
- Mengerjakan fitur *share link* untuk halaman beserta fiturnya yang bisa diakses tanpa harus melalui perlindungan sistem *Single Sign-On* (SSO) v3 milik perusahaan.

Pada dasarnya, semua fitur *endpoint* yang sudah dibuat sebelumnya dilindungi oleh perlindungan sistem SSO v3 perusahaan, sehingga memerlukan token untuk mengaksesnya seperti pada Gambar 3.41. Fitur *Share Link* Publik dibuat agar memungkinkan pengguna eksternal mengakses halaman layanan serah terima tanpa harus melalui autentikasi SSO v3 perusahaan. Fitur ini sangat penting dalam mendukung kolaborasi eksternal, terutama dengan pihak *vendor* atau mitra yang tidak memiliki akses ke sistem internal perusahaan. Selain itu, *Office Boy* (OB) yang tidak terdaftar di SSO v3, juga dapat membantu proses serah terima barang.



Gambar 3.40 *Endpoint* SSO v3 Perusahaan.

Gambar 3.40 di atas memberikan gambaran terkait *endpoint* yang digunakan dalam sistem SSO v3 perusahaan. Token didapat melalui *endpoint* /public/login berupa token JWT. Token tersebut akan diinput ke Cookies *browser user* dan valid selama beberapa waktu. Token juga akan hilang atau tidak valid lagi jika *user* melakukan *logout* via *endpoint* /public/logout.

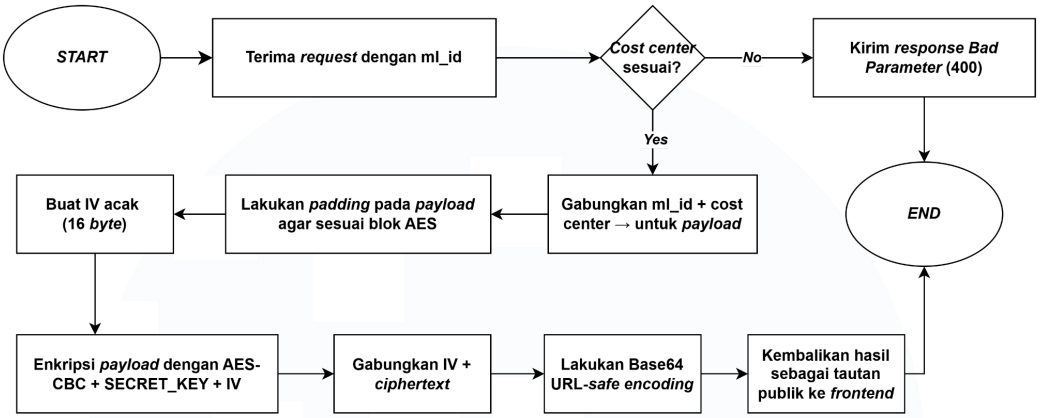


Gambar 3.41 *Authorize User* Menggunakan Token SSO v3 Perusahaan.

Fitur diawali dengan pembuatan *endpoint* /public/generate yang digunakan untuk menghasilkan tautan terenkripsi berdasarkan ID layanan (ml_id) dan *cost center* pengguna. Proses enkripsi dilakukan menggunakan algoritma AES-CBC

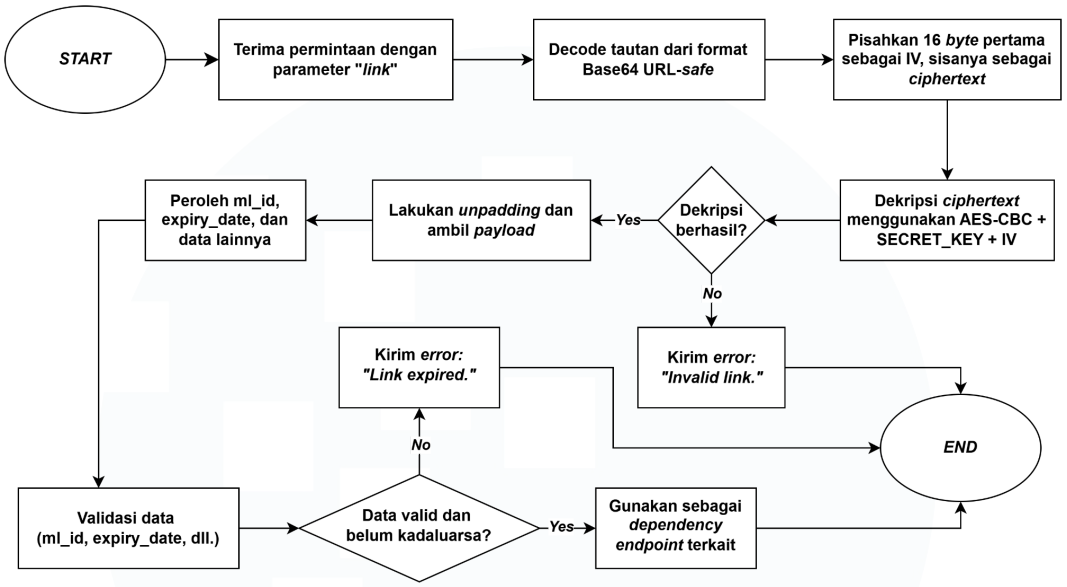
(*Advanced Encryption Standard - Cipher Block Chaining*) melalui modul CryptoService, di mana data terenkripsi akan dikodekan dengan Base64 URL-safe agar dapat digunakan secara aman dalam URL. *Initialization Vector* (IV) sepanjang 16 *byte* dihasilkan secara acak untuk setiap permintaan, memastikan setiap tautan bersifat unik meskipun isi data sama. Selanjutnya, input teks yang akan dienkripsi diproses melalui padding terlebih dahulu agar sesuai dengan panjang blok AES menggunakan metode pad() dengan format UTF-8 dari *library* Crypto.Util.Padding.

Proses enkripsi dilakukan dengan algoritma AES dalam mode CBC, menggunakan kunci rahasia (*SECRET_KEY*) dari konfigurasi *environment* aplikasi. Dalam mode CBC, setiap blok data *plaintext* terlebih dahulu dioperasikan dengan teknik XOR terhadap blok *ciphertext* sebelumnya sebelum melalui proses enkripsi. Untuk blok pertama, karena belum ada *ciphertext* sebelumnya, digunakan IV sebagai gantinya. Mekanisme ini memastikan bahwa meskipun dua blok *plaintext* memiliki isi yang sama, hasil enkripsinya akan berbeda karena bergantung pada *output* blok sebelumnya atau IV. Dengan kata lain, keamanan mode CBC diperoleh dari rantai ketergantungan antar blok, yang membuat pola data tidak mudah ditebak dan memperkuat perlindungan terhadap serangan kriptografi. Setelah dienkripsi, hasilnya digabungkan bersama IV, lalu dikodekan dengan format Base64 URL-*safe* agar tautan dapat digunakan secara aman dalam URL tanpa menghasilkan karakter yang tidak valid. Hasil final kemudian dikembalikan ke *frontend* dalam bentuk URL publik. Alur generasi *link* ini dapat dilihat pada Gambar 3.42 berikut:



Gambar 3.42 Alur Pembuatan *Share Link* Terenkripsi.

Selanjutnya, *endpoint* /public/validate memungkinkan sistem memvalidasi isi dari tautan tersebut untuk memastikan data yang diakses sesuai dengan yang diotorisasi. Langkah dekripsi dimulai dengan melakukan verifikasi terhadap tautan terenkripsi dengan mendekode data dari format Base64, memisahkan *Initialization Vector* (IV), dan mendekripsi ciphertext menggunakan algoritma AES-CBC dengan SECRET_KEY. Jika proses dekripsi gagal — misalnya karena tautan rusak, tidak sah, atau telah dimodifikasi — maka sistem akan langsung menolak permintaan dan memberikan respons *error*. Jika dekripsi berhasil, data kemudian dihilangkan *padding*-nya (*unpad*) dan diuraikan (*parsing*) untuk mendapatkan *payload* asli berisi ml_id, expiry_date, dan data lainnya. Validasi dilanjutkan untuk memastikan data masih berlaku dan otorisasi sesuai. Hanya jika seluruh proses validasi berhasil, *payload* dapat digunakan sebagai *dependency* pada *endpoint* publik seperti /upload_details, /penilaian, dan /penilaian/items_need_survey. Proses validasi ini melibatkan dekripsi data dengan tahap sebagai berikut:



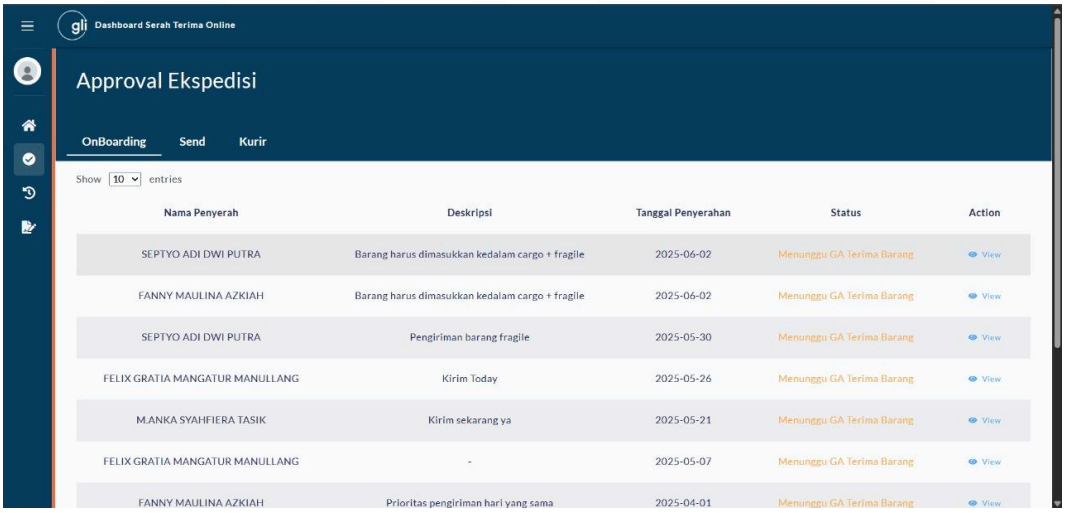
Gambar 3.43 Alur Dekripsi dan Validasi *Share Link*.

Setelah link berhasil divalidasi melalui *endpoint* /public/validate, hasil *payload* digunakan sebagai *dependency* pada *endpoint* publik seperti /upload_details dan /public/penilaian. Pada *endpoint* /upload_details, pengguna eksternal dapat mengunggah foto bukti serah terima dan tanda tangan penerima dalam format digital, yang akan disimpan ke GCS menggunakan mekanisme Signed URL seperti pada proses internal. Sementara itu, melalui /public/penilaian, pengguna eksternal juga dapat mengisi penilaian terhadap layanan. Kedua *endpoint* bergantung pada hasil dekripsi dan otorisasi yang sudah dipastikan sebelumnya melalui *middleware/dependency* validasi. Dengan fitur ini, proses bisnis yang sebelumnya hanya bisa dilakukan oleh karyawan internal kini dapat dilakukan juga oleh pihak eksternal dengan kontrol yang aman dan *user-friendly*. Ini menjadi salah satu langkah penting dalam mendorong digitalisasi proses serah terima secara menyeluruh dan kolaboratif.

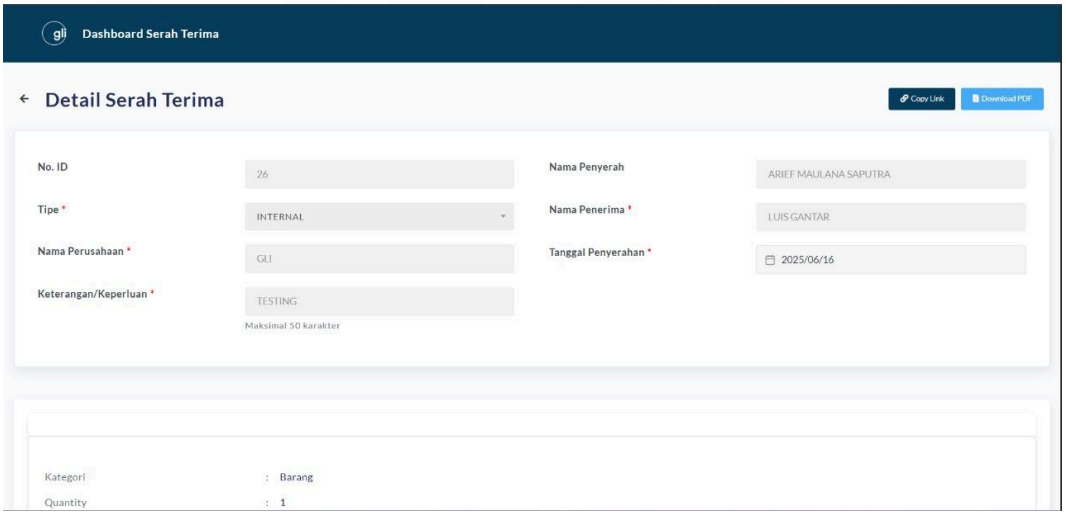
- Mengerjakan integrasi dengan aplikasi Ekspedisi.

Selain membangun fitur serah terima internal, integrasi dengan aplikasi Ekspedisi juga telah dikerjakan. Aplikasi Ekspedisi sendiri merupakan sistem

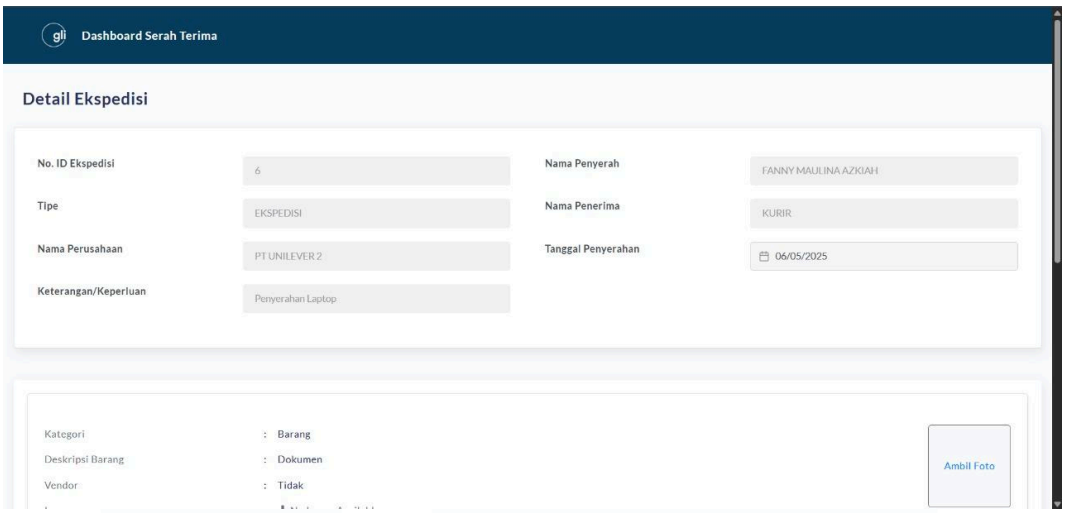
terpisah yang dikembangkan oleh *intern* lain dan berfungsi sebagai sistem pengelolaan pengiriman barang yang masih berkaitan erat dengan proses serah terima. Dalam integrasi ini, data barang yang dikirim melalui ekspedisi akan terlebih dahulu masuk ke aplikasi ekspedisi dan menunggu proses *approval* dari pihak manajer. Setelah disetujui, barang akan otomatis muncul di *Dashboard Approval* Ekspedisi aplikasi ekspedisi khusus tim GA seperti pada Gambar 3.44, sebagai rekapan bahwa barang siap diserahkan secara fisik.



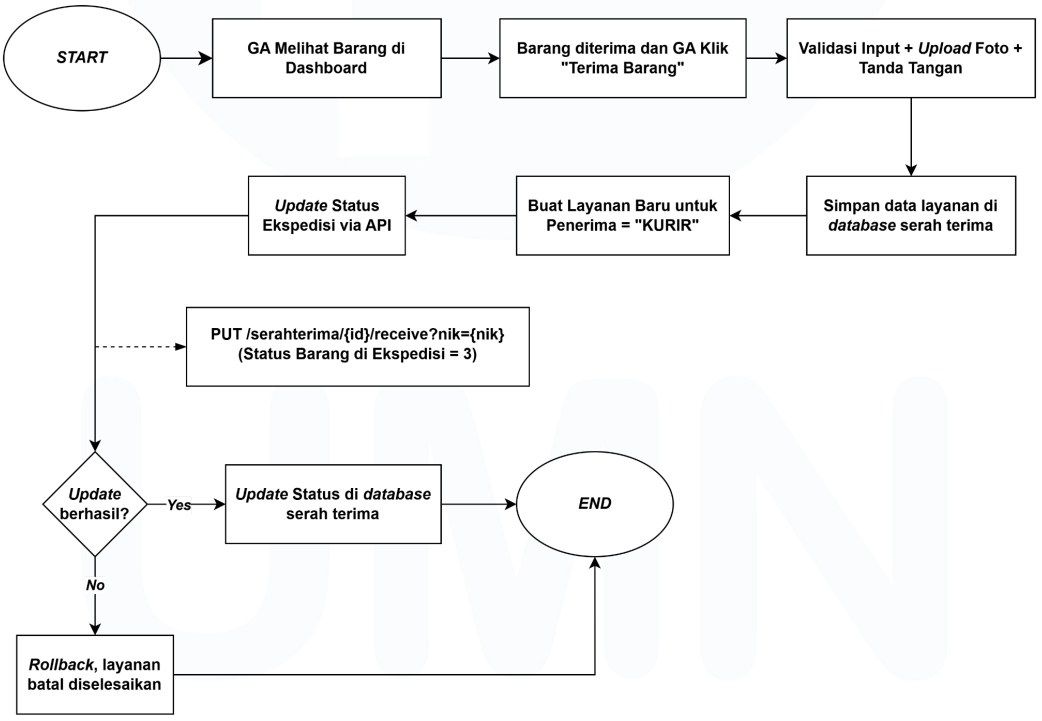
Gambar 3.44 Halaman *Dashboard Approval* Ekspedisi.



Gambar 3.45 Halaman Detail Layanan Serah Terima Ekspedisi kepada GA.



Gambar 3.46 Halaman Detail Layanan Serah Terima GA kepada Kurir.



Gambar 3.47 Alur Integrasi Aplikasi Serah Terima dan Ekspedisi.

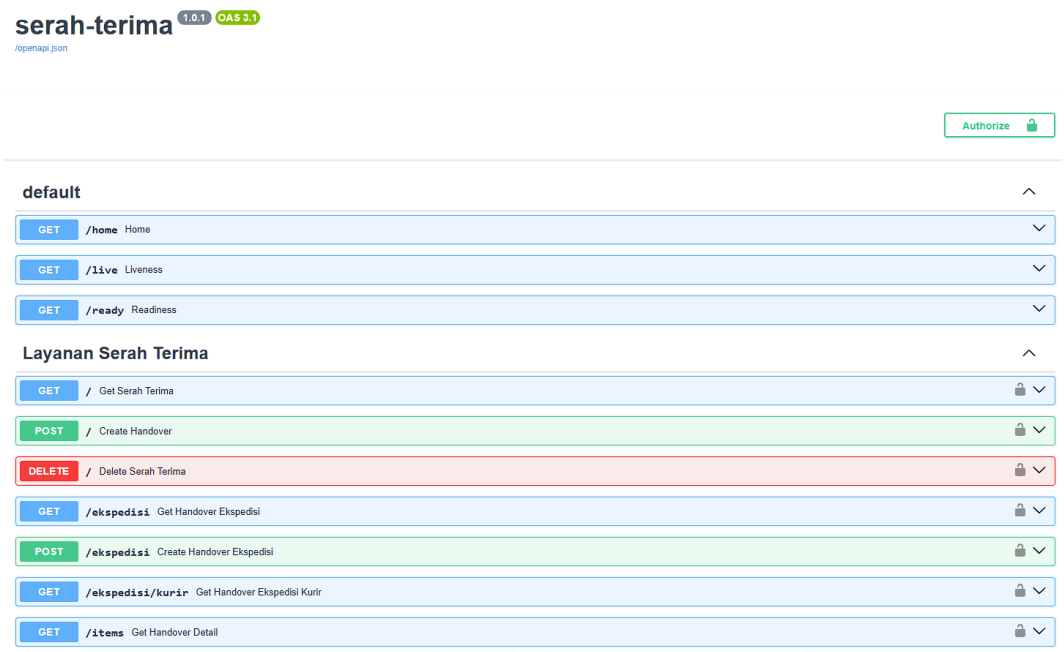
Gambar 3.45 menjelaskan bahwa setelah barang muncul di dashboard GA dari sistem ekspedisi, tim GA akan menerimanya melalui aplikasi serah terima seperti Gambar 3.46 dengan memanfaatkan fitur update_handover. Proses ini akan

secara otomatis membentuk layanan baru dengan penerima "KURIR" sebagai pihak berikutnya dalam alur distribusi seperti pada Gambar 3.47. Setelah serah terima ke kurir selesai, *backend* aplikasi serah terima akan memanggil API milik aplikasi ekspedisi melalui fungsi update_ekspedisi, yang mengirimkan *request* ke *endpoint* HTTP PUT /serahterima/{ml_tr_id}/receive?nik={nik}. Pemanggilan ini dilakukan secara *real-time* untuk menyinkronkan status pengiriman antar sistem, dengan status diperbarui menjadi "3" yang menandakan barang telah diberikan ke kurir. Untuk menghindari inkonsistensi data, struktur URL, parameter, dan respons API telah disepakati sejak awal, dan setiap kegagalan dalam pemanggilan API akan memicu mekanisme pengamanan atau pencegahan (*fail-safe*) berupa *exception FailedExternalHit* yang membatalkan *update* lokal (*rollback*) agar status tidak berubah tanpa kepastian bahwa sistem ekspedisi telah mencatatnya.

- *Deployment* ke *staging* dan membantu dalam proses integrasi API dengan tim *Frontend Developer*.

Sebelum proses integrasi dimulai, *deployment* ke lingkungan *staging* dilakukan sebagai bagian dari alur pengembangan dan pengujian internal. Proses ini dibantu langsung oleh mentor (Mas Resa) dengan pembuatan repositori proyek di Bitbucket sebagai *remote repository* utama. Setelah repositori dibuat, langkah awal dilakukan dengan *clone* repositori kosong tersebut ke lokal, kemudian seluruh hasil pengembangan *backend* dimasukkan dan dicatat sebagai perubahan (*commit)* sebagai *initial commit*. Seluruh *commit* selanjutnya mengikuti standar *semantic commit message*, seperti *feat*:, *fix*:, atau *refactor*:, yang berfungsi untuk menandai jenis perubahan secara eksplisit. Pendekatan ini tidak hanya membantu dalam membaca histori perubahan dengan lebih mudah, tetapi juga menjadi dasar otomasi dalam integrasi berkelanjutan maupun *release note* di masa mendatang. Proyek juga disusun menggunakan struktur modular dan *environment* yang terpisah (*development*, *staging*, *production*), sehingga memudahkan proses *deployment* terkontrol dan integrasi API bersama tim *frontend* berjalan secara mulus.

Dalam proses integrasi API, kolaborasi aktif dilakukan bersama tim *Frontend Developer*, khususnya dengan Luis sebagai rekan utama. Seluruh API yang dikembangkan telah terdokumentasi secara otomatis melalui Swagger (OpenAPI 3.0), sebuah fitur bawaan dari *framework* FastAPI. Dokumentasi ini sangat membantu proses integrasi karena menyajikan deskripsi *endpoint*, parameter, tipe data, hingga contoh respons dalam format interaktif yang mudah diakses oleh tim *frontend*. Untuk mendukung keterbacaan, *endpoint* juga telah dikelompokkan menggunakan *tag* pada Swagger agar lebih terorganisir sesuai dengan fitur masing-masing, lebih jelasnya dapat dilihat pada Gambar 3.48.



Gambar 3.48 Contoh Dokumentasi API Menggunakan Swagger OpenAPI 3.0.

Selain dokumentasi otomatis, catatan tambahan juga disiapkan melalui platform Notion yang memuat contoh permintaan (*request*) dan respons sukses maupun gagal (*error response*), sehingga dapat digunakan sebagai referensi praktis saat pengembangan antarmuka. Untuk mempercepat pengujian, *mock data* sesuai kontrak API sering kali diberikan ketika *backend* belum sepenuhnya stabil atau masih dalam tahap pengembangan. Validasi input dan output pada setiap *endpoint* telah disesuaikan dengan skema yang didefinisikan dalam model

Pydantic, memastikan konsistensi data dan meminimalkan potensi kesalahan saat integrasi. Komunikasi antara tim dilakukan secara langsung maupun secara asinkron melalui grup diskusi internal, dan alur kolaborasi dijaga agar tetap efisien dan responsif terhadap kebutuhan tim antarmuka.

- Mendiskusikan hasil kerja dengan mentor dan PM untuk diteruskan ke QA.

Dalam proses *review*, mentor memberikan sejumlah masukan penting terkait praktik pengembangan agar sistem lebih mudah dirawat dan diperluas ke depannya. Beberapa arahan utama mencakup penerapan prinsip *atomic function* dalam penulisan kode, sehingga setiap fungsi atau modul memiliki satu tanggung jawab yang jelas untuk meningkatkan *maintainability*. Selain itu, penggunaan *container* sebagai pusat deklarasi *repository* dan *use case* ditekankan untuk menyederhanakan proses pengujian dan memperkuat struktur arsitektur aplikasi. Validasi *request body* dan struktur *response* juga disarankan untuk dimatangkan menggunakan Pydantic schema agar komunikasi antara *backend* dan *frontend* tetap konsisten dan terstandarisasi. Di sisi pengelolaan proyek, alur kerja Git yang lebih sistematis diterapkan, dengan pembagian branch seperti *feat*/*, *staging*, dan *main*, di mana semua perubahan perlu melalui proses *staging* sebelum digabungkan ke branch utama.

Sementara itu, masukan dari PM lebih menekankan pada kesesuaian implementasi terhadap alur bisnis yang telah ditentukan. Hal ini mencakup validasi bahwa akses layanan dibatasi hanya untuk pengguna dengan *cost center* yang sesuai, serta memastikan bahwa laporan yang dihasilkan mencerminkan kondisi sebenarnya di lapangan. PM juga turut meninjau ketepatan data yang dikirimkan antar sistem, baik dari sisi isi maupun format, agar dapat langsung dimanfaatkan oleh sistem pelaporan dan audit internal.

- Melakukan perbaikan fitur yang dilaporkan oleh QA.

Dalam proses QA, penulis turut terlibat dalam *bug fixing* dan *refinement* terhadap fitur yang telah diuji. QA melakukan pengujian menyeluruh mencakup

validasi input, pengujian respon API, serta integrasi dengan *frontend*. Temuan bug didokumentasikan dalam bentuk *bug list*, dan penulis bertanggung jawab menyelesaikan *issue* tersebut dengan memperbaiki logika kode, memperketat validasi, atau menyesuaikan struktur response API jika diperlukan. Semua perbaikan diuji ulang oleh QA sebelum dinyatakan lulus tahap UAT.Dalam proses Quality Assurance (QA), keterlibatan dilakukan dalam kegiatan *bug fixing* dan *refinement* terhadap fitur yang telah diuji. Pengujian dilakukan secara menyeluruh oleh tim QA, meliputi validasi input, pengujian response API, serta integrasi dengan sisi frontend. Temuan dari proses pengujian tersebut didokumentasikan dalam bentuk *bug list* dan diserahkan untuk diperbaiki sesuai standar fungsionalitas sistem.

Perbaikan dilakukan dengan menyesuaikan logika kode, memperkuat validasi data, maupun memperbaiki struktur response API apabila diperlukan. Seluruh pembaruan diverifikasi ulang oleh tim QA hingga dinyatakan lulus tahap UAT. Koordinasi lintas tim juga dilakukan untuk memastikan perbaikan telah sesuai dengan alur bisnis yang diharapkan. Beberapa *bug* yang ditemukan saat aplikasi diterapkan ke environment *staging*, dapat dilihat pada Gambar 3.49.

C	D	E	F	G	H	I	J
Platform	User Access	Modul / Field	Description	Screenshot	QA Status	Priority	Dev Status
BE	User	Report	User bisa mencari semua Nama/NIK di dropdown Nama Penyerah		OK	LOW	READY TO TEST
BE	User	Report	Di dropdown Nama Penerima, User hanya bisa mencari Nama/NIK yang terdata di CC User		OK	LOW	READY TO TEST

Gambar 3.49 *List Bug* QA.

Masalah tersebut ditelusuri ke bagian *query* yang belum menyaring data berdasarkan peran atau hak akses pengguna. Setelah perbaikan diterapkan, pencarian telah dibatasi sesuai dengan ketentuan yang berlaku.

Secara keseluruhan, jumlah bug yang muncul setelah deployment tergolong minim. Hal ini dikarenakan pengembangan pada lingkungan lokal telah dilakukan secara intensif bersamaan, dengan uji logika bisnis dan validasi yang

diterapkan sejak awal. Dengan pendekatan tersebut, potensi kesalahan berhasil ditekan sebelum pengujian QA dimulai.

- Melakukan pertemuan dengan *user* (UAT) serta melakukan perbaikan fitur yang direvisi oleh *user* saat UAT.

Sesi UAT turut diikuti bersama *user* dari divisi terkait, seperti GA, *Procurement, Finance*, *Merchandise*, dan lain-lain. dalam proyek Serah Terima. Dalam sesi ini, sistem diuji langsung oleh *user* melalui simulasi penggunaan secara *end-to-end* untuk memastikan bahwa fitur yang dikembangkan telah sesuai dengan kebutuhan operasional serta ekspektasi lapangan.

Masukan yang diberikan selama sesi UAT bersifat praktis dan langsung menyasar aspek penggunaan harian, seperti perubahan label, perbaikan alur UI/UX, hingga penyesuaian pada logika bisnis di sisi *backend*. Masukan tersebut kemudian didokumentasikan dan ditindaklanjuti melalui serangkaian revisi teknis lintas *platform* (*frontend* dan *backend*). Setiap perubahan dikonfirmasi kembali kepada tim QA dan user untuk menjamin kesesuaian hasil akhir dengan permintaan. *List* masukan tersebut yang berkaitan dengan *backend* ada pada Gambar 3.50.

Feedback	Platform	User Acess	Modul/Field	Status QA	FE Developer
Penambahan field input Nama Perusahaan saat Create data Serah Terima	FE + BE	All User	Create	OK	OK
Penambahan jumlah karakter "Keterangan/Keperluan" menjadi 50 karakter	FE + BE	All User	Create	OK	OK
Penambahan kolom nama (auto generate by nik) & TTD di bagian Create awal saat membuat pengajuan tipe external	FE + BE	All User	Create (External)	OK	OK
Akun tim Procurement di lock untuk : Mba Hena - & Mba Lena -	BE	GA (Procurement)	User Access	OK	OK

Gambar 3.50 *List Feedback* Pasca UAT.

Semua perubahan tersebut telah diperbaiki, diuji ulang, dan dikonfirmasi oleh QA sebelum sistem dinyatakan siap untuk produksi (tahap *production*). Pada tahap produksi ini, aplikasi dapat digunakan oleh semua *user* baik internal maupun eksternal yang memerlukan penggunaan situs web Serah Terima *Online* ini Proses UAT dan perbaikannya menjadi tahapan penting untuk memastikan bahwa implementasi fitur benar-benar memenuhi kebutuhan operasional di lapangan.

3.3 Limitasi

Pengembangan sistem *backend website* Serah Terima *Online* di PT. Global Loyalty Indonesia menemui beberapa limitasi dan keputusan desain yang perlu diperjelas, terutama terkait struktur basis data dan alur integrasi antar aplikasi.

- Redundansi Data pada Basis Data

  Dalam perancangan basis data Serah Terima *Online*, terdapat implementasi redundansi data yang disengaja. Contohnya, informasi pengguna seperti Nomor Induk Karyawan (NIK), *Cost Center* (CC), dan nama diduplikasi dari tabel *master user* dan disimpan juga pada tabel *master* layanan serah terima. Keputusan ini diambil dengan mempertimbangkan beberapa aspek performa dan validasi. Tujuan utama dari redundansi ini adalah untuk mengurangi beban *query* ke tabel *master user*. Dengan menduplikasi data krusial langsung ke tabel layanan, proses *fetching* data dapat dilakukan dengan lebih cepat karena tidak memerlukan operasi *join* yang kompleks antar tabel saat mengambil detail layanan. Perlu dicatat bahwa praktik duplikasi data master pada tabel-tabel transaksional juga umum diterapkan dalam arsitektur basis data di aplikasi lain, termasuk di dalam lingkungan perusahaan, untuk tujuan optimasi performa dan kemudahan akses data. Terlepas dari keputusan menggunakannya, terdapat kelemahan utamanya yaitu potensi inkonsistensi data. Jika data asli di tabel *master user* mengalami perubahan, data duplikat di tabel layanan harus diperbarui secara manual atau melalui mekanisme sinkronisasi, dan jika ini gagal, akan terjadi ketidakcocokan data.

- Optimalisasi Integrasi Aplikasi Ekspedisi dan Serah Terima

  Integrasi antara aplikasi Ekspedisi dan aplikasi Serah Terima telah dibangun sesuai *request*, namun terdapat ruang untuk optimalisasi pengalaman pengguna (UX), khususnya bagi tim GA. Saat ini, setelah barang ekspedisi disetujui oleh atasan di aplikasi Ekspedisi, tim GA masih

perlu membuat layanan baru secara manual di aplikasi Serah Terima, meskipun data barang sudah tersedia dari aplikasi Ekspedisi. Proses manual ini menyebabkan duplikasi pekerjaan bagi tim GA. Untuk memaksimalkan efisiensi dan meningkatkan pengalaman pengguna, alur ini dapat diotomatisasi. Solusinya adalah dengan menambahkan *endpoint* POST pada *backend* aplikasi Serah Terima yang dapat dipanggil oleh *backend* aplikasi Ekspedisi. Dengan demikian, pada saat barang ekspedisi disetujui atasannya, layanan serah terima akan secara otomatis terbuat di aplikasi Serah Terima. Hal ini akan mengurangi beban kerja tim GA, di mana mereka hanya perlu melanjutkan proses penerimaan barang dan penyerahan kepada kurir dari layanan yang sudah otomatis terbentuk.

Berikut adalah perbandingan kondisi *existing* dan *expected* terkait integrasi ini:

Tabel 3.2 Perbandingan Kondisi Integrasi Aplikasi Ekspedisi dan Serah Terima

Aspek	Kondisi Sekarang	Kondisi yang Diharapkan
Pembuatan Layanan Serah Terima Barang Ekspedisi	Manual oleh tim GA di aplikasi Serah Terima setelah persetujuan di aplikasi Ekspedisi.	Otomatis terbuat di aplikasi Serah Terima melalui panggilan API dari backend aplikasi Ekspedisi.
Tingkat Efisiensi GA	Kurang efisien, membutuhkan kerja ganda (membuat layanan dan memproses serah terima).	Lebih efisien, tim GA fokus pada proses fisik penerimaan dan penyerahan barang.
Keterlibatan Tim GA	Memulai proses layanan dari awal di aplikasi Serah Terima.	Melanjutkan proses layanan yang sudah ada secara otomatis.

Aspek	Kondisi Sekarang	Kondisi yang Diharapkan
Interaksi antar Aplikasi	Data dari Ekspedisi diperoleh, tetapi pembuatan layanan di Serah Terima masih terpisah.	Integrasi penuh: aplikasi Ekspedisi dapat memicu pembuatan layanan di aplikasi Serah Terima.

Alasan mengapa optimalisasi ini belum diimplementasikan sejak awal adalah karena koordinasi awal antara pengembang dan PM belum mempertimbangkan skenario otomatisasi penuh ini sebagai prioritas utama. Fokus awal lebih pada komunikasi sistem aplikasi Ekspedisi dan Serah Terima *Online*. Mengingat aplikasi sudah berada di tahap produksi (*production*), implementasi perubahan signifikan seperti ini menjadi lebih kompleks dan berisiko jika dilakukan secara mendadak. Oleh karena itu, penambahan fitur otomatisasi ini menunggu *request* versi 2.0 dari *user* GA, yang akan dijadwalkan dalam siklus pengembangan selanjutnya.

3.4 Kendala yang Ditemukan

- Logika bisnis yang diterapkan dalam sistem serah terima cukup kompleks, terutama karena adanya berbagai kondisi khusus seperti integrasi dengan aplikasi ekspedisi, mekanisme *share link*, tata cara validasi penilaian, evaluasi, dan lain sebagainya.
- Penanganan kasus baru selama proses pengembangan dihadapi, seperti beberapa fitur tambahan di luar standar CRUD muncul, di antaranya:
    - Pengiriman email notifikasi otomatis.
    - Upload dan validasi file gambar (foto barang/TTD) ke GCS.
    - Pembuatan dan *export* file Excel sebagai *reporting*.

3.5 Solusi atas Kendala yang Ditemukan

- Klarifikasi dan diskusi langsung dengan PM untuk memahami logika dan alur bisnis. Hal ini membantu memastikan fitur yang dibuat sesuai kebutuhan pengguna.

- Diskusi teknis secara aktif dilakukan bersama tim *frontend* untuk menyamakan pemahaman terhadap alur data, termasuk validasi, struktur permintaan/respons API, dan penyesuaian UI sesuai kebutuhan *backend*.
- Inisiatif pembelajaran mandiri untuk kasus baru untuk fitur non-standar seperti integrasi ke GCS dan generasi *file* Excel:
    - Eksplorasi melalui dokumentasi resmi (misalnya FastAPI, openpyxl, Google Cloud SDK).
    - Penggunaan AI tools sebagai asisten teknis untuk mencari pendekatan umum dan best practices.
    - Studi kasus melalui forum teknis seperti Stack Overflow dan blog pengembang lain sebagai referensi implementasi.
- Penerapan prinsip *Trial and Error* yang Terstruktur dalam menghadapi fitur baru, pendekatan dilakukan secara iteratif—mulai dari implementasi dasar, kemudian dilakukan pengujian secara bertahap untuk memastikan bahwa logika bisnis dan integrasi berjalan dengan benar.