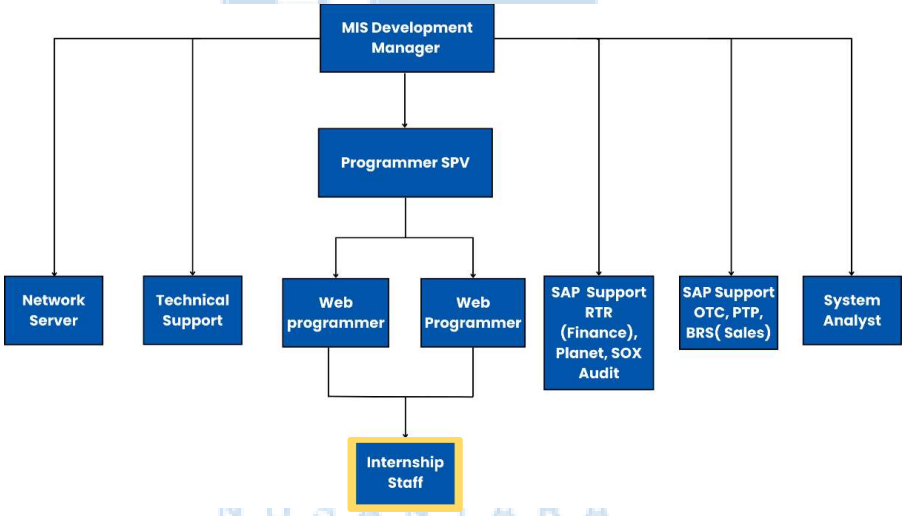
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

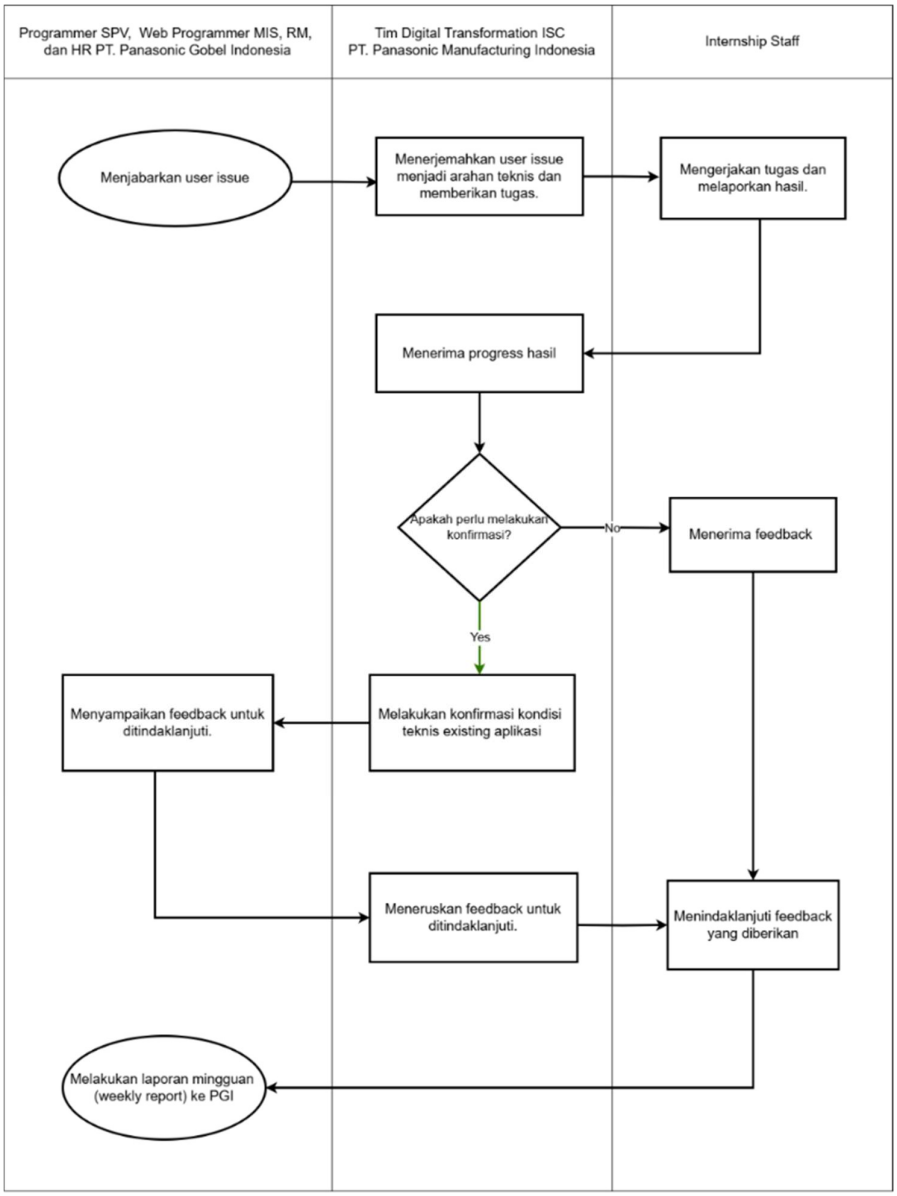
Pada pelaksanaan program magang ini, penempatan dilakukan di bawah Departemen *Management Information System* (MIS) PT. Panasonic Gobel Indonesia. Selama menjalankan kegiatan, berada dalam bimbingan *Web Programmer*, Ibu Tya Herlina, serta dalam pengawasan *Programmer Supervisor*, Bapak Lukman Hakim. Kegiatan yang dijalankan mencakup tugas-tugas yang berkaitan dengan pengelolaan sistem informasi dan pengembangan aplikasi berbasis web sesuai kebutuhan perusahaan. Posisi dan kedudukan dalam struktur organisasi ditunjukkan pada Gambar 3.1.



Gambar 3.1 Kedudukan Posisi Magang

Namun, karena pelaksanaan kegiatan magang secara langsung berlokasi di PT. Panasonic Manufacturing Indonesia, maka koordinasi tidak dilakukan secara langsung dengan pembimbing dari PT. Panasonic Gobel Indonesia. Sebagai gantinya, koordinasi dilakukan melalui Tim *Digital Transformation* yang tidak hanya berperan sebagai divisi penghubung, tetapi juga turut terlibat aktif dalam memberikan arahan, evaluasi, serta pengambilan keputusan teknis yang berkaitan dengan pekerjaan yang dilaksanakan di lokasi. Tim ini secara

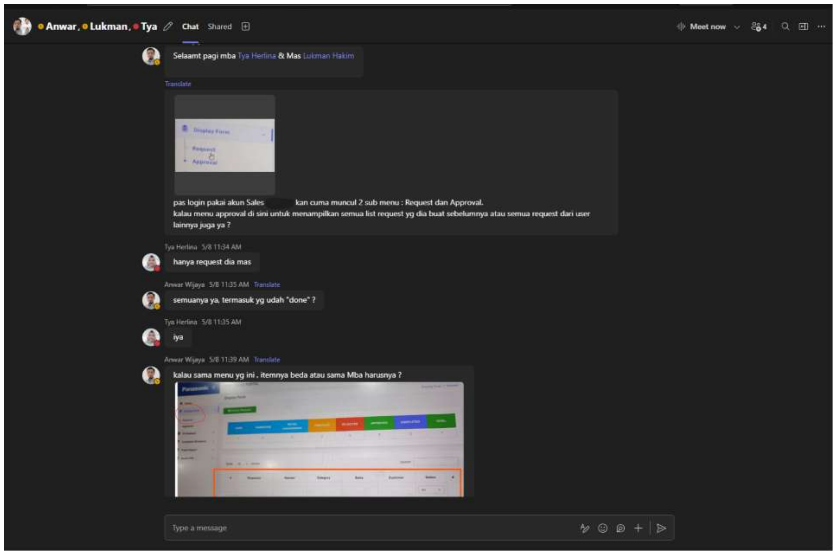
berkala memantau progres pekerjaan dan memastikan agar seluruh aktivitas magang selaras dengan kebutuhan dan kebijakan yang berlaku di lingkungan PT. Panasonic Manufacturing Indonesia serta PT. Panasonic Gobel Indonesia. Berikut merupakan alur koordinasi yang diterapkan yang dapat dilihat pada Gambar 3.2



Gambar 3.2 Alur Koordinasi

Alur koordinasi pelaksanaan tugas magang diawali dengan penjabaran *user requirement* dari *user* PT. Panasonic Gobel Indonesia, yaitu Programmer Supervisor, Bapak Lukman Hakim, dan Web Programmer, Ibu Tya Herlina. Instruksi tersebut kemudian diteruskan ke Tim Digital Transformation di PT. Panasonic Manufacturing Indonesia yang diwakili oleh Bapak Anwar Wijaya dan Bapak Novi Indra F. Tim ini berperan menerjemahkan kebutuhan tersebut menjadi arahan teknis yang dapat dilaksanakan di lapangan. Selanjutnya, tugas dikerjakan sesuai dengan arahan yang diberikan. Hasil pekerjaan kemudian didiskusikan kembali dengan Bapak Anwar dan Bapak Indra untuk mendapatkan evaluasi dan masukan.

Dalam proses koordinasi tersebut, apabila diperlukan klarifikasi lebih lanjut, Tim Digital Transformation melakukan konfirmasi kepada pihak PT. Panasonic Gobel Indonesia. Setelah memperoleh jawaban atau solusi dari PT. Panasonic Gobel Indonesia, informasi tersebut disampaikan kembali agar dapat segera ditindaklanjuti. Koordinasi ini juga umumnya dilakukan pada grup Microsoft Teams ataupun *personal chat* seperti pada Gambar 3.3 berikut ini.



Gambar 3.3 Koordinasi melalui Microsoft Teams

Dengan demikian, proses komunikasi berjalan secara dua arah dan berkesinambungan. Seluruh kegiatan magang juga dilaporkan secara berkala

melalui laporan mingguan (*weekly report*). Laporan ini disampaikan kepada PT. Panasonic Gobel Indonesia sebagai bentuk pertanggungjawaban dan pemantauan progres magang.

3.2 Tugas dan Uraian Kerja Magang

Sebagai bagian dari pelaksanaan program magang di PT Panasonic Manufacturing Indonesia, karyawan magang dilibatkan secara langsung dalam proses pengembangan sistem internal perusahaan, khususnya pada *project* yang berjalan dalam departemen terkait. Berikut Tabel 3.1 merupakan rangkaian kegiatan dan uraian tugas yang dijalankan selama masa magang.

Tabel 3.1 Timeline Kerja Magang

No		Deskripsi Pekerjaan	Start	End
Orientasi dan Pemahaman Project				
1.	1a.	Orientasi perusahaan dan pengenalan lingkungan perusahan	17 Maret 2025	18 Maret 2025
	1b.	Pengenalan alur kerja *internal* dan ruang lingkup project	19 Maret 2025	21 Maret 2025
	1c.	Penyusunan general flow (*As-Is dan To-Be*), serta detail proses sistem	22 Maret 2025	24 Maret 2025
Perancangan Desain Antarmuka (GUI)				
2.	2a.	Desain layout halaman *Request* dan *Approval* berdasarkan *flow*	24 Maret 2025	09 April 2025
	2b.	Revisi GUI berdasarkan feedback dari atasan	09 April 2025	16 April 2025
Setup Development Environment dan Eksplorasi Kode				
3.	3a.	Instalasi *tools dev* dan *setup* laptop kerja	17 April 2025	17 April 2025
	3b.	Tes koneksi server lokal, GitLab, dan eksplorasi struktur project Laravel	21 April 2025	23 April 2025
Pengembangan Fitur Dealer Current Sell In				
4.	4a.	Pembuatan tabel data *dummy* pada database *existing* dan *backend* awal untuk fitur Request Form pada column Dealer Current Sell In	24 April 2025	24 April 2025
	4b.	Implementasi serta *logic controller autofill* berdasarkan dari data *dummy* yang dipanggil dari database	25 April 2025	28 April 2025
	4c.	Melakukan *mapping* data dari SAP, serta mengimplementasikan dan mengatur *logic* controller kembali pada Dealer Current Sell	29 April 2025	30 April 2025

		In berdasarkan column invoice_date, gross_amount, dan parent_customer_number di database.		
	4d.	Finalisasi *logic autofill* dan membuat fitur Priority di form approval	02 Mei 2025	02 Mei 2025
Troubleshooting dan Pengembangan Fitur Priority				
5.	5a.	Melakukan pengembangan fitur *Priority* pada form *Request*, termasuk memeriksa alur variabel, menyelesaikan masalah nilai *null* pada logika controller saat submit, melakukan sinkronisasi pembaruan dari cabang Git, dan menyesuaikan kode sesuai perubahan yang ada.	05 Mei 2025	07 Mei 2024
	5b.	Menyelesaikan pengembangan dan perbaikan fitur Priority pada form Approval, termasuk submit form, perbaikan field yang disable, penyesuaian tampilan, serta penelusuran dan perbaikan alur penyimpanan data Dealer Current Sell In ke database.	08 Mei 2025	09 Mei 2025
Pengembangan Menu Dashboard Project Overview				
6.	6a.	Mulai pengembangan menu dashboard, menyusun struktur awal, implementasi awal serta logic controller untuk hitung project per kategori, dan rancangan chart status hasil project.	13 Mei 2025	14 Mei 2025
	6b.	Menyelesaikan Dashboard dengan chart interaktif dan real-time, serta perbaikan *logic* javascript untuk penyimpanan data Dealer Current Sell In.	15 Mei 2025	16 Mei 2025
Pembuatan Laporan Magang				
7.	7a.	Mengurus administrasi merdeka serta pengerjaan laporan	19 Mei 2025	10 Juni 2025
Pembuatan Flow System Digital Stock Card serta Konfigurasi Awal				
8.	8a.	Membuat detail flow system untuk pengimplementasian Digital Stock Card	10 Juni 2025	13 Juni 2025
	8b.	Menginstall tools yang diperlukan untuk project Digital Stock Card	16 Juni 2025	17 Juni 2025
	8c.	Menambahkan fitur filter untuk *page* Dashboard pada project PGI Sales Portal	18 Juni 2025	20 Juni 2025

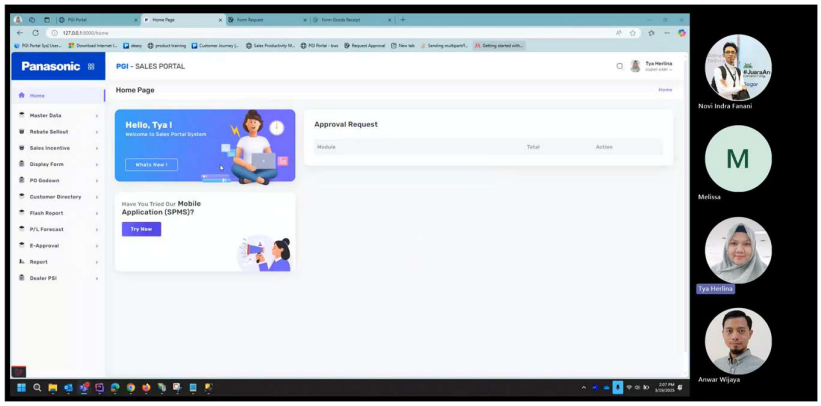
	8d.	Melakukan pemberesan error error yang terjadi serta debugging kembali	20 Juni 2025	1 Juli 2025
Pengembangan Modul Mobile Digital Stock Card				
9	9a.	Menyelaraskan tampilan UI pada aplikasi mobile, termasuk perbaikan ikon yang tidak muncul serta melakukan percobaan penambahan elemen pada salah satu fitur.	2 Juli 2025	7 Juli 2025
	9b.	Melakukan penggantian API dari WMS ke API baru untuk mendukung sistem Digital Stock Card pada aplikasi mobile	7 Juli 2025	8 Juli 2025
	9c.	Menambahkan fitur *"Sent Notification Email"* pada tampilan aplikasi sebagai bagian dari pengembangan fungsionalitas	9 Juli 2025	9 Juli 2025

3.2.1 Minggu 1-2 (17 Maret – 24 Maret 2025): Orientasi dan Pemahaman Project

Pada awal pelaksanaan kegiatan magang di PT. Panasonic Manufacturing Indonesia, karyawan magang mengikuti program orientasi yang difasilitasi oleh tim *Human Capital*. Kegiatan ini mencakup penjelasan mengenai profil perusahaan, sejarah berdirinya, serta budaya kerja yang dijalankan. Selain itu, karyawan magang juga diberikan pemahaman terkait peraturan *internal* perusahaan, tata tertib, dan etika kerja yang wajib dipatuhi. Orientasi ini bertujuan untuk membentuk sikap profesional sejak awal, agar seluruh karyawan magang dapat menjalankan peran dan tanggung jawabnya secara selaras dengan nilai-nilai perusahaan. Sesi orientasi umum kemudian diikuti dengan pengenalan terhadap alur kerja di divisi ISC (*Information System Center*), yaitu divisi tempat penempatan karyawan magang yang berfokus pada pengembangan sistem digital serta pengelolaan infrastruktur teknologi informasi di lingkungan perusahaan.

Setelah orientasi dasar dan pemahaman terhadap struktur divisi, kegiatan dilanjutkan dengan pengenalan *project* utama yang akan menjadi fokus selama masa magang, yaitu PGI Sales Portal. *Project* ini

diperkenalkan melalui sesi *briefing* yang membahas *user issue* serta kebutuhan sistem secara komprehensif. Pemaparan disampaikan oleh Ibu Tya selaku *Web Programmer* MIS di PT. Panasonic Gobel Indonesia, yang berperan sebagai pembimbing teknis utama. Turut hadir juga dalam sesi tersebut Pak Anwar, selaku Group Chief DX di PT. Panasonic Manufacturing Indonesia, serta Pak Indra selaku *System Development* di PT. Panasonic Manufacturing Indonesia yang memberikan masukan dari sisi sistem internal dan integrasi. seperti pada Gambar 3.4.



Gambar 3.4 Meeting Brief

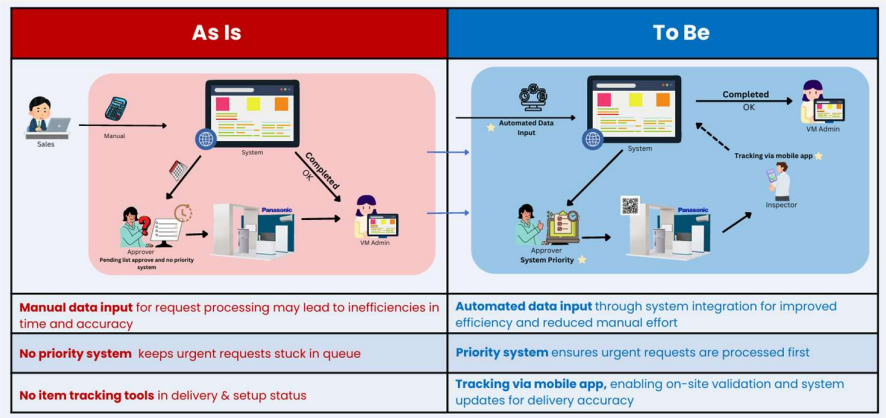
Dalam sesi tersebut, dijelaskan latar belakang kebutuhan sistem, permasalahan yang sedang dihadapi oleh *user*, serta ruang lingkup pengembangan yang diharapkan. Penjelasan ini memberikan pemahaman menyeluruh mengenai isu yang ada dan menjadi dasar bagi proses analisis sistem berikutnya. *Project* ini dirancang sebagai respon terhadap kondisi sistem *existing* yang dinilai masih belum optimal, khususnya karena masih mengandalkan input manual, tidak adanya sistem prioritas dari tiap *request*, serta minimnya transparansi dalam pelacakan progres pekerjaan. Secara keseluruhan, *project* ini merupakan bagian dari inisiatif strategis perusahaan dalam meningkatkan efisiensi operasional dan mendukung transformasi digital yang berkelanjutan.

Dalam rangka mendukung pelaksanaan *project*, instalasi *software* pendukung dilakukan terlebih dahulu pada laptop pribadi untuk

memastikan lingkungan kerja siap digunakan. Beberapa tools utama yang digunakan antara lain Visual Studio Code sebagai *integrated development environment* (IDE) utama dalam proses pengembangan aplikasi, Composer dan PHP yang menjadi pondasi *backend* sistem, Node.js sebagai *runtime* platform yang mendukung ekosistem JavaScript, serta GitLab sebagai sistem kontrol versi sekaligus media kolaborasi antar anggota tim pengembang. Setelah instalasi *software* selesai, pengujian server secara lokal dilaksanakan guna memastikan kestabilan dan kesesuaian lingkungan kerja sebelum masuk ke tahap pengembangan lebih lanjut.

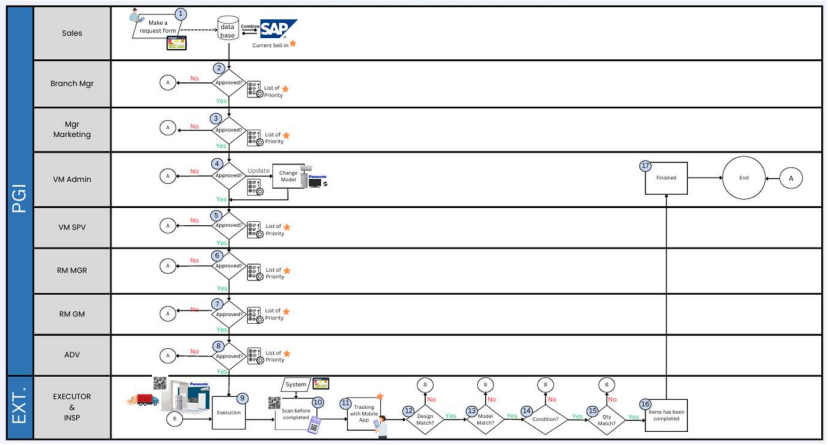
Selanjutnya, dilakukan pemetaan kondisi sistem saat ini (*As-Is*) yang menunjukkan sejumlah kendala dalam efektivitas proses pengelolaan permintaan. Proses input data yang masih manual berpotensi menimbulkan kesalahan dan memperlambat waktu pengolahan. Selain itu, belum adanya mekanisme prioritas menyebabkan permintaan mendesak tertahan dalam antrean tanpa penanganan khusus. Proses persetujuan juga masih berjalan secara berurutan tanpa mempertimbangkan tingkat urgensi, sementara fitur pelacakan status pengiriman dan pemasangan barang belum tersedia, sehingga transparansi dan koordinasi lapangan kurang optimal.

Sebagai solusi, dirancang sistem baru (*To-Be*) yang mengintegrasikan fitur otomatisasi dan digitalisasi. Proses input data akan dilakukan secara otomatis melalui integrasi antar sistem untuk meningkatkan akurasi dan mengurangi beban manual. Sistem akan dilengkapi fitur prioritas berdasarkan urgensi permintaan agar pengelolaan lebih efisien dan terstruktur. Pengembangan fitur pelacakan berbasis aplikasi mobile juga memungkinkan monitoring *real-time* dan validasi langsung di lapangan. Berikut rancangan *system flow* yang dapat dilihat pada Gambar 3.5.



Gambar 3.5 Flow As-Is dan To-Be

Sebagai bagian dari upaya untuk memperjelas alur kerja yang akan diimplementasikan dalam sistem baru, disusun sebuah diagram detail *flow system* yang menggambarkan keseluruhan tahapan pengelolaan permintaan. Diagram ini memetakan jalannya proses mulai dari *request* oleh tim terkait, dilanjutkan dengan tahapan *approval* yang melibatkan beberapa pihak internal, hingga proses eksekusi di lapangan dan pelaporan hasil akhir. Penyusunan diagram ini tidak hanya berfungsi sebagai alat bantu visualisasi, tetapi juga sebagai dasar untuk mengidentifikasi tanggung jawab setiap aktor yang terlibat serta titik-titik krusial dalam proses pengambilan keputusan. Diagram tersebut dapat dilihat pada Gambar 3.6.

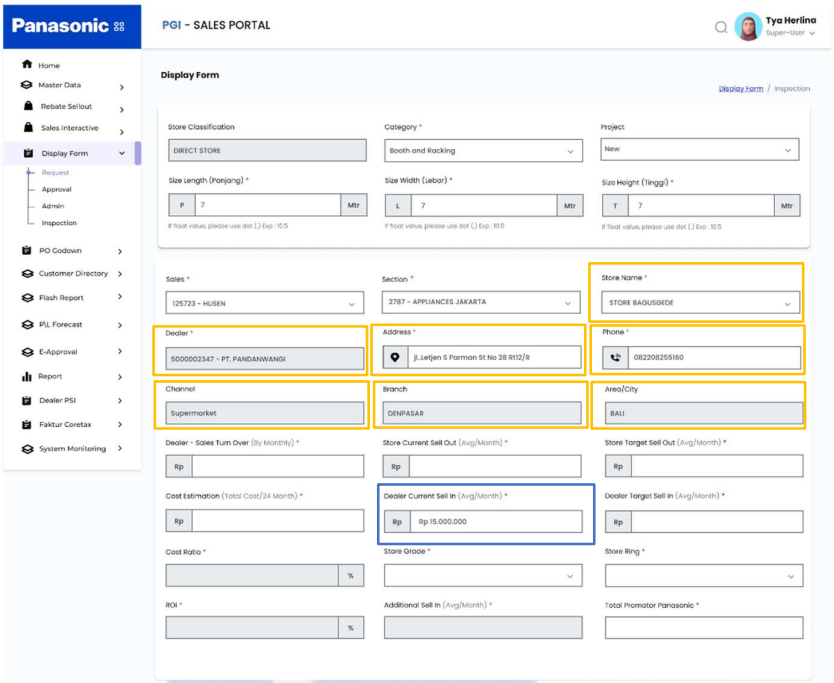


Gambar 3.6 Detail Flow System

3.2.2 Minggu 2-4 (24 Maret – 16 April 2025): Perancangan Desain Antarmuka (GUI)

Setelah dilakukan analisis terhadap kondisi sistem berjalan (As-Is) dan sistem yang dirancang (To-Be), termasuk pemetaan alur persetujuan, tahap selanjutnya difokuskan pada perancangan antarmuka pengguna atau *Graphical User Interface* (*GUI*).

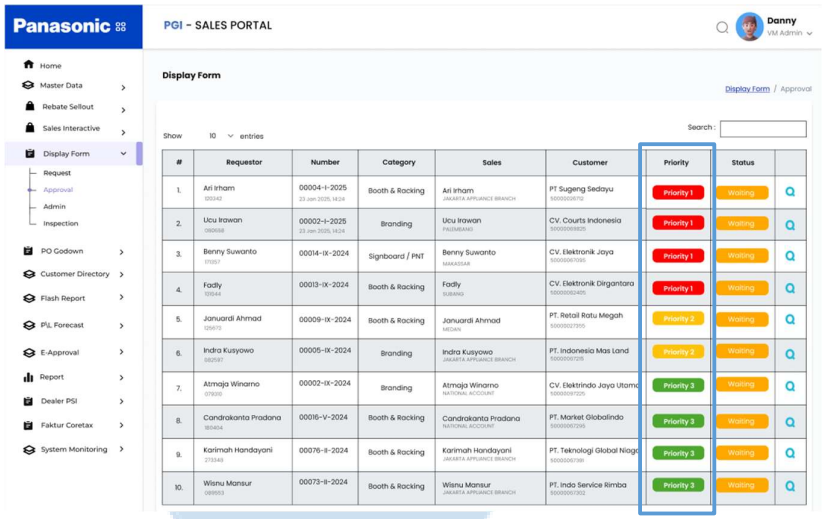
Tujuan utama dari tahap ini adalah menghasilkan tampilan antarmuka yang intuitif, efisien, serta mendukung keseluruhan alur proses mulai dari pengajuan permintaan, mekanisme persetujuan dengan prioritas, hingga *monitoring* melalui dashboard. Dalam proses perancangan, digunakan *tools* figma untuk memberikan kemudahan dalam pembuatan desain interaktif serta kolaborasi antar tim pengembang. Desain antarmuka yang dihasilkan menjadi acuan utama dalam pengembangan fitur-fitur, seperti *autofill* data *dealer*, penerapan logika *priority* dalam *approval*, serta visualisasi status *project* secara *real-time* dengan indikasi warna yang memudahkan pemahaman pengguna. Beberapa contoh hasil desain antarmuka dapat dilihat pada gambar di bawah ini.



Gambar 3.7 Tampilan GUI Request Form

Pada Gambar 3.7 ditampilkan *Graphical User Interface* (GUI) dari halaman Request Form. Tampilan ini disimulasikan menggunakan data *dummy* lalu secara keseluruhan mempertahankan struktur dan komposisi yang serupa dengan antarmuka asli yang telah diimplementasikan dalam sistem. Perbedaan utama terletak pada aspek interaksi yang disempurnakan untuk memberikan gambaran alur input data yang lebih interaktif. Dalam alur yang dirancang, setelah *requestor* mengisi informasi pada bagian *Category*, *Project*, *Size Length*, *Size Width*, dan *Size Height*, sistem akan secara otomatis menampilkan nilai pada kolom *Section* berdasarkan pilihan yang diinput pada bagian *Sales*. Selain itu, jika pada sistem sebelumnya proses pemilihan *Store Name* hanya memicu pengisian otomatis pada kolom *Dealer*, *Address*, *Phone*, *Channel*, *Branch*, dan *Area/City*, maka pada rancangan ini terdapat perluasan interaksi berdasarkan kebutuhan *user*. Ketika *requestor* memilih *Store Name*, sistem dirancang untuk turut menampilkan informasi *Dealer Current Sell In* yang diperoleh secara otomatis

berdasarkan *Dealer Code* yang muncul pada kolom *Dealer*. Rancangan alur interaktif ini bertujuan untuk meningkatkan efisiensi input data sekaligus meminimalkan potensi kesalahan dalam pengisian formulir permintaan.



Gambar 3.8 Tampilan GUI Fitur Priority pada Approval Page

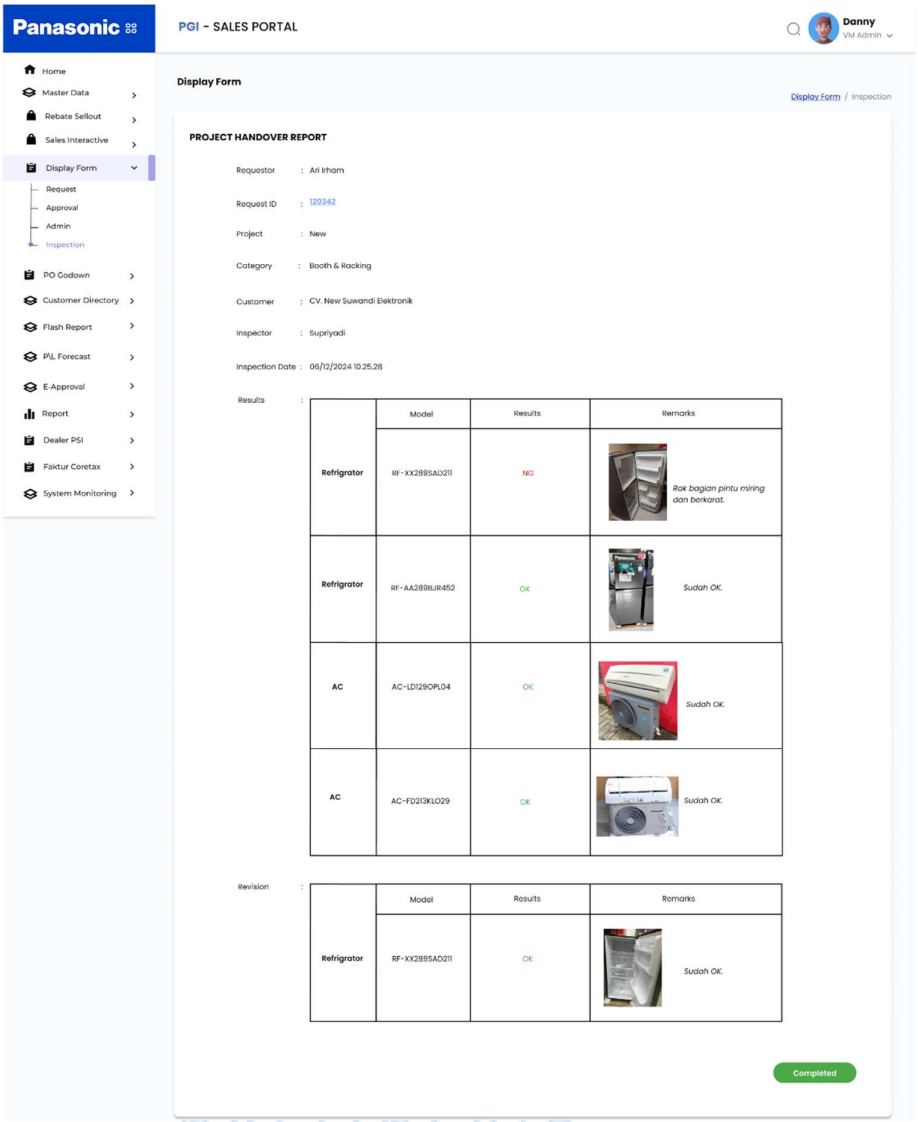
Pada Gambar 3.8 ditampilkan *Graphical User Interface* (GUI) dari halaman Approval. Tampilan ini disimulasikan menggunakan data *dummy* dan memperlihatkan adanya penambahan elemen informasi berupa fitur *Priority* pada tabel utama. Fitur ini dikembangkan dengan mengacu pada parameter-parameter tertentu yang telah ditetapkan sebelumnya, sehingga setiap *request* dapat diklasifikasikan sesuai dengan kebutuhan penanganan yang lebih cepat atau mendesak. Kehadiran fitur ini memungkinkan *approver* untuk memproses permintaan secara lebih efisien berdasarkan urutan prioritas yang telah ditentukan oleh sistem.

Gambar 3.9 Tampilan GUI Inspection

Pada Gambar 3.9 ditampilkan *Graphical User Interface* (GUI) dari halaman Inspection. Tampilan ini disimulasikan menggunakan data *dummy*, serta menampilkan penambahan submenu baru di bawah menu Display Form, yaitu submenu Inspection. Pada halaman ini, informasi divisualisasikan melalui sebuah *dashboard* yang memuat tiga komponen utama, yaitu *pie chart* untuk distribusi kategori *project*, *bar chart* untuk status hasil *project*, serta tabel Request List yang menyajikan data individual setiap request *project*.

Pie chart menampilkan distribusi jumlah *project* berdasarkan tiga kategori besar, yaitu *Booth & Racking*, *Branding*, serta *Signboard/PNT*, yang masing-masing dibedakan berdasarkan jenis *project* yang masuk. Visualisasi ini berguna untuk memberikan gambaran mengenai proporsi jumlah *project* yang diterima berdasarkan klasifikasinya. Sementara itu, *bar chart* menampilkan status dari masing-masing *project* dalam lima jenis *project*, yaitu *New*, *Renovation*, *Quick Fix*, *Dismantle*, dan *Exhibition*. Status *project* dikelompokkan menjadi dua, yaitu OK (berwarna hijau) dan NG (berwarna merah), yang menunjukkan hasil dari proses verifikasi atau pengecekan terhadap setiap *project* yang diajukan.

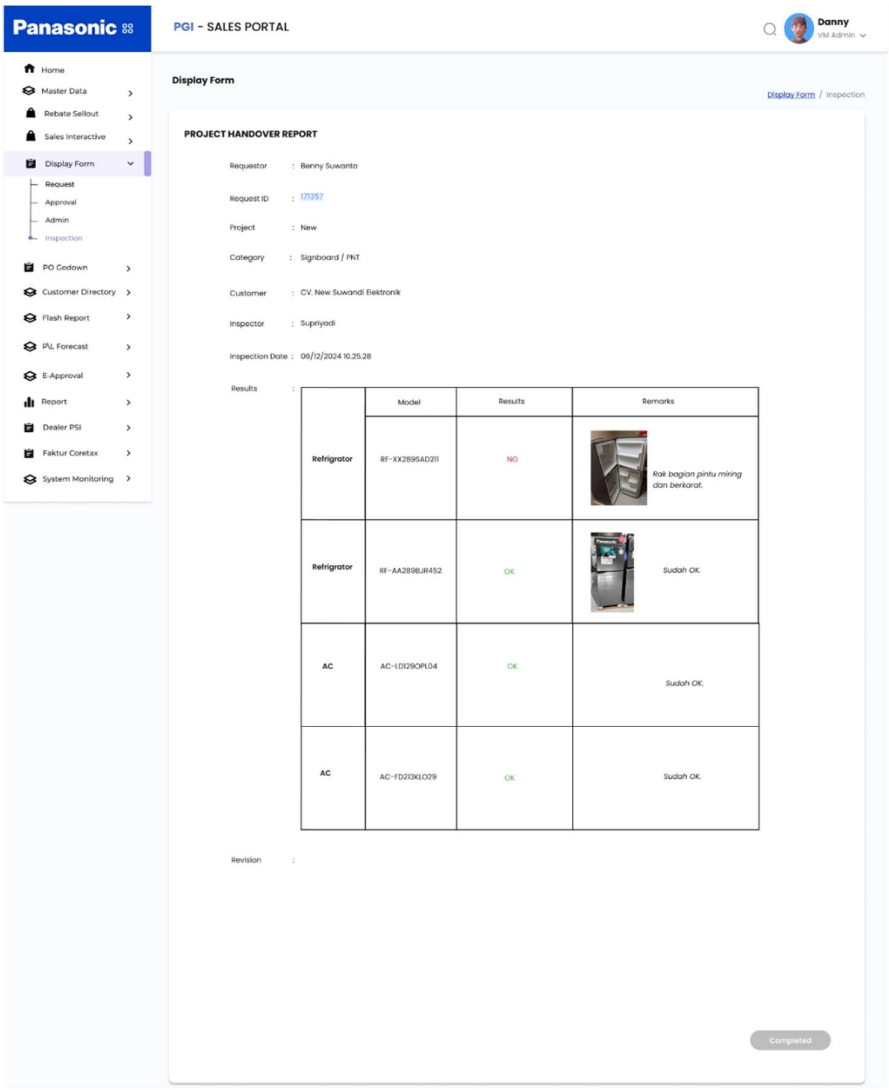
Pada bagian bawah halaman, terdapat Request List berupa tabel yang memuat informasi detail terkait project yang telah masuk dan akan diproses lebih lanjut oleh pihak terkait. Tabel ini mencakup kolom seperti *Inspection Date*, *Request ID*, *Requestor*, *Project*, *Category*, *Customer,* serta *Status*. Status yang ditampilkan dapat berupa *Revised* maupun *Pending Revision*, khususnya jika ditemukan kondisi NG pada salah satu *project*. Selain itu, kolom *Request ID* dalam tabel ini bersifat interaktif, yang memungkinkan *user* untuk mengakses halaman selanjutnya yang menampilkan laporan berita acara secara lebih rinci.



Gambar 3.10 Tampilan GUI Project Handover Report Complete

Pada Gambar 3.10 ditampilkan *Graphical User Interface* (GUI) dari halaman Inspection secara lebih detail. Tampilan ini disimulasikan menggunakan data *dummy* untuk merepresentasikan kondisi sistem ketika hasil pemeriksaan (*inspection*) memiliki status *Not Good* (NG). Dalam tampilan ini, inspector akan melakukan proses revisi terhadap data yang tidak sesuai. Setelah proses revisi selesai, sistem akan secara otomatis mengaktifkan tombol *Completed* yang berfungsi sebagai

indikator bahwa seluruh proses pemeriksaan telah diselesaikan dan data telah siap untuk ditindaklanjuti ke tahap berikutnya.

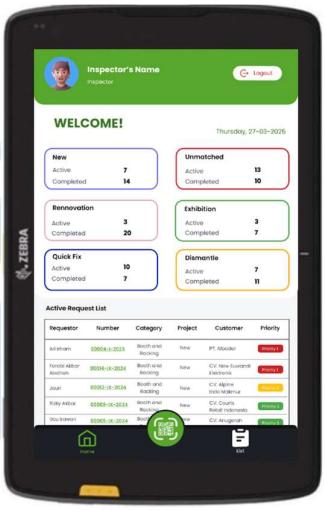


Gambar 3.11 Project Handover Report Not Completed

Pada Gambar 3.11 ditampilkan *Graphical User Interface* (GUI) dari halaman Inspection secara lebih detail. Tampilan ini disimulasikan menggunakan data *dummy* untuk merepresentasikan situasi ketika hasil pemeriksaan terhadap suatu *project* masih belum sepenuhnya diselesaikan oleh *inspector*. Dalam kondisi ini, ditemukan item dengan status *Not Good* (NG) yang belum dilakukan perbaikan atau revisi. Oleh

karena itu, sistem secara otomatis menonaktifkan tombol *Completed*, sebagai penanda bahwa proses *inspection* belum dapat dianggap selesai dan masih memerlukan tindakan lanjutan sebelum dapat ditutup secara formal.

Selanjutnya terdapat tampilan antarmuka atau GUI yang disusun secara tampilan mobile seperti pada Gambar 3.12 – Gambar 3.17 di bawah ini.

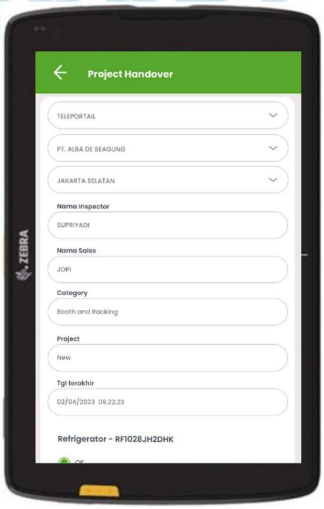


Gambar 3.12 Tampilan GUI Mobile Home

Pada Gambar 3.12 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan *mobile* dari halaman Home. Tampilan ini disimulasikan menggunakan data *dummy* yang memperlihatkan ringkasan status permintaan berdasarkan jenis *project*, seperti *New*, *Renovation*, *Quick Fix*, *Unmatched*, *Exhibition*, dan *Dismantle*, yang terbagi dalam status *Active* dan *Completed*. Selain itu, terdapat *Active Request List* yang menampilkan informasi seperti *requestor*, *number*, *category*, *project name*, *customer*, dan *priority*.

Gambar 3.13 Tampilan GUI Mobile Scan

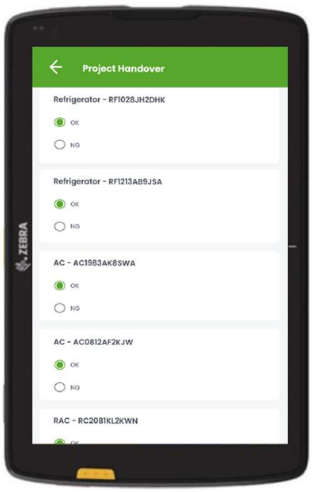
Pada Gambar 3.13 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan mobile dari halaman Scan. Tampilan ini memperlihatkan fitur *scan* yang dapat langsung mengarahkan *inspector* ke *forms* yang perlu diisi. Sebagai alternatif, *inspector* juga dapat memilih untuk tidak menggunakan fitur *scan* dan langsung menekan tombol *Input Forms* untuk mengisi *forms*.



Gambar 3.14 Tampilan GUI Mobile Form

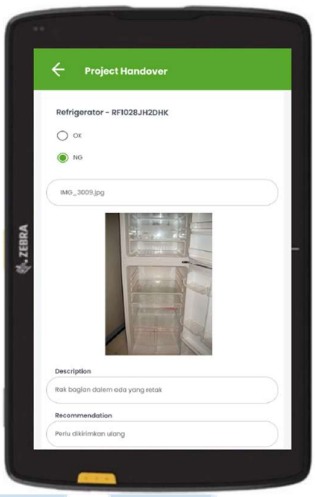
Pada Gambar 3.14 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan mobile dari halaman Forms Project Handover. Tampilan

ini disimulasikan menggunakan data *dummy* yang memperlihatkan fitur *forms* yang terdiri dari beberapa *input field* seperti nama *store*, nama *dealer*, dan lokasi yang diisi secara manual oleh *inspector*. Sementara itu, field seperti *Inspector Name*, *Sales Name*, *Category*, *Project*, dan *Tanggal Terakhir* terisi secara otomatis oleh sistem berdasarkan data yang telah tersimpan sebelumnya.



Gambar 3.15 Tampilan GUI Mobile Form 1

Pada Gambar 3.15 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan mobile dari halaman *Forms Project Handover*. Tampilan ini memperlihatkan fitur *forms* yang terdiri dari beberapa *input field* terkait produk dan tipe-nya serta pilihan status yang terdiri dari dua opsi, yaitu OK dan NG. *Form* ini digunakan oleh *inspector* untuk mengisi data item dan tipe item yang sedang diperiksa, kemudian menentukan status produk berdasarkan hasil inspeksi tersebut. Status OK menandakan bahwa produk telah memenuhi standar kualitas yang ditetapkan serta sudah *match* dengan apa yang requestor *request*, sementara status NG menunjukkan adanya ketidaksesuaian atau cacat yang perlu mendapat perhatian dan tindak lanjut.



Gambar 3.16 Tampilan GUI Mobile Form 2

Pada Gambar 3.16 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan mobile dari halaman *Forms Project Handover*. Tampilan ini memperlihatkan fitur *forms* ketika *inspector* memilih opsi NG untuk suatu item yang menunukkan adanya ketidaksesuaian atau cacat yang perlu mendapat perhatian dan tindak lanjut.

Gambar 3.17 Tampilan GUI Mobile Project Handover List

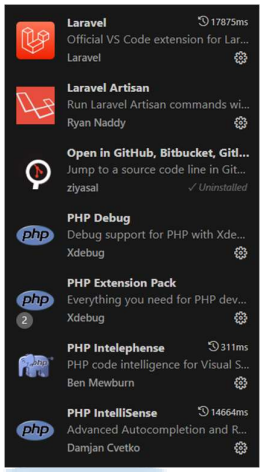
Pada Gambar 3.17 ditampilkan *Graphical User Interface* (GUI) dalam bentuk tampilan mobile dari halaman *Forms Project Handover*. Tampilan ini memperlihatkan tabel yang menampilkan *project handover*

list dengan beberapa kolom utama, yaitu *Inspection Date, Requestor, Category, Project, Customer*, dan *Status*. Kolom *Status* merepresentasikan tahapan dari proses inspeksi, apabila berstatus *Submitted*, berarti proses inspeksi telah selesai dilakukan namun belum diselesaikan (*completed*) dari sisi sistem website. Sementara itu, status *On Going* menunjukkan bahwa proses inspeksi masih berlangsung dan belum mencapai tahap akhir.

3.2.3 Minggu 4-5 (17 April - 23 April 2025): Setup Development Environment dan Eksplorasi Kode

persiapan lingkungan kerja dengan cara meng*install software* yang mendukung pengembangan sistem berbasis laravel. Pada tahap ini, digunakan perangkat kerja berupa laptop yang telah disediakan oleh pihak perusahaan untuk memastikan kesesuaian spesifikasi dan stabilitas dalam pengembangan. Untuk itu, dilakukan instalasi beberapa *software* inti yang bersifat wajib dalam mendukung *project* ini. *Software* tersebut antara lain mencakup Visual Studio Code sebagai (Integrated Development Environtment) IDE utama, PHP sebagai bahasa pemrogramman backend, Composer untuk *dependency manager*, Node.js sebagai runtime environment JavaScript yang berperan penting dalam mengelola *frontend*, termasuk pengelolaan asset seperti CSS dan JavaScript melalui Laravel, serta Laravel sebagai framework pengembangan. Selain itu, GitLab digunakan sebagai sistem pengelolaan versi *source code* yang terintegrasi dengan aktivitas kolaboratif pengembangan.

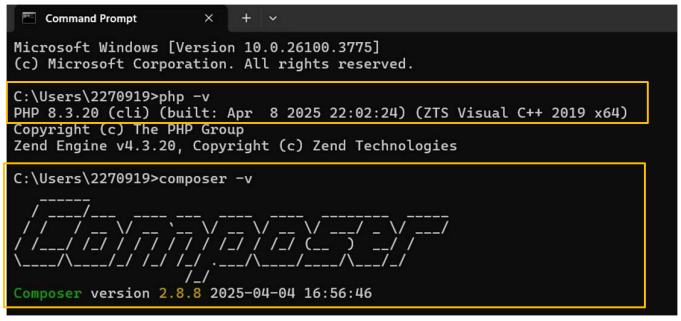
Instalasi Visual Studio Code dilakukan dengan meng*install* instalasi resmi melalui situs web Microsoft. Setelah proses instalasi, ditambahkan beberapa *extension* penting ditambahkan seperti PHP Intelephense dan Laravel Snippets untuk meningkatkan efisiensi dalam menulis dan membaca kode program seperti pada Gambar 3.18.



Gambar 3.18 Extension Visual Studio Code

Visual Studio Code dipilih karena memiliki fitur unggulan seperti penyorotan sintaks, pelengkapan otomatis, terminal bawaan, serta integrasi langsung dengan sistem version control berbasis Git. Melalui fitur-fitur tersebut, proses debugging, peninjauan logika, serta navigasi antar file *project* dapat dilakukan dengan cepat dan akurat. Kemampuan untuk menyesuaikan lingkungan kerja melalui ekstensi juga menjadikan Visual Studio Code sebagai salah satu alat bantu yang sangat mendukung *project* ini.

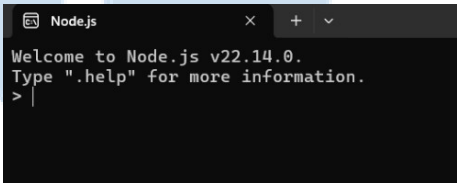
Untuk pemrosesan sisi server, dilakukan instalasi interpreter PHP secara langsung tanpa menggunakan bundel *software* seperti XAMPP. Installer resmi diunduh dari situs web php.net dan dikonfigurasi melalui environment path system agar dapat digunakan secara global melalui command line seperti pada Gambar 3.19.



Gambar 3.19 CMD PHP dan Composer

Dengan konfigurasi ini, proses inisialisasi *project* Laravel dan *dependency manager* melalui Composer dapat dilakukan tanpa hambatan. Composer digunakan untuk mengelola *library* eksternal yang dibutuhkan oleh Laravel, sehingga struktur *project* tetap modular dan konsisten. Langkah ini juga memungkinkan penggunaan server bawaan Laravel melalui perintah *php artisan serve*, yang digunakan selama tahap pengembangan dan pengujian lokal.

Node.js juga di*install* sebagai *runtime environment* JavaScript yang penting dalam pengelolaan *asset frontend project*. Instalasi Node.js dilakukan dengan mengunduh instalasi resmi dari situs resminya, kemudian dikonfigurasi agar dapat diakses melalui command line secara global seperti pada Gambar 3.20.



Gambar 3.20 Node.js

Node.js memfasilitasi pengelolaan *frontend* melalui NPM (*Node Package Manager*), yang sangat membantu dalam proses build dan kompilasi aset seperti CSS dan JavaScript menggunakan Laravel. Penggunaan Node.js ini menjadikan proses pengelolaan asset lebih terotomatisasi dan efisien, sehingga pengembangan antarmuka pengguna dapat berjalan dengan lancar dan performa aplikasi tetap optimal [13].

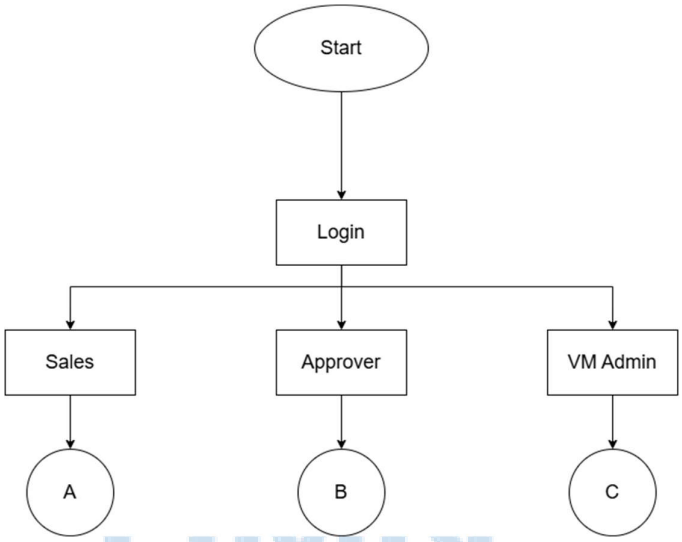
Setelah *software* utama berhasil diinstalasi, dilakukan eksplorasi terhadap struktur *project* Laravel yang telah tersedia di repository GitLab. *Project* kemudian di-clone melalui Git ke direktori lokal guna memfasilitasi pengembangan secara langsung di lingkungan kerja masing-masing. Struktur direktori standar Laravel seperti routes, app/Http/Controllers, resources/views, dan app/Models dipelajari untuk memahami relasi antar komponen dan alur pemrosesan data. Dengan

memahami arsitektur MVC yang diterapkan, pengembangan fitur baru dapat dilakukan dengan tetap menjaga konsistensi struktur kode.

Secara keseluruhan, pengembangan sistem dalam *project* ini dibagi ke dalam dua fase utama yang dirancang secara strategis agar pelaksanaan dapat dilakukan secara bertahap dan terstruktur. Pendekatan ini bertujuan untuk mengoptimalkan proses implementasi fitur inti terlebih dahulu, sebelum berlanjut pada fitur lanjutan yang lebih kompleks. Pada fase pertama, pengembangan difokuskan pada tiga komponen utama, yaitu fitur *Dealer Current Sell In* pada halaman Request, fitur *Approval*, dan *dashboard* visualisasi awal. Fitur *Dealer Current Sell In* dilengkapi dengan mekanisme *autofill* data yang diambil dari transaksi SAP, guna mempermudah *sales* dalam menginput data *request*. Sementara itu, pada fitur *Approval*, diimplementasikan sistem penentuan prioritas yang berfungsi untuk mengelompokkan *request* yang masuk berdasarkan tingkat urgensinya, sehingga proses pengambilan keputusan dapat dilakukan dengan lebih efisien dan tepat sasaran. Selain itu, juga dikembangkan tampilan dashboard awal yang menyajikan visualisasi data dalam bentuk grafik *pie* dan *bar*, dengan mengelompokkan informasi berdasarkan kategori *project* serta status pelaksanaan (OK/NG). *Dashboard* ini menjadi alat bantu awal dalam monitoring hasil pelaksanaan *project* secara agregat. Seluruh pengembangan pada fase pertama ditujukan untuk membangun pondasi sistem yang kuat, efisien, dan dapat diandalkan sebelum berlanjut ke pengembangan lanjutan pada fase berikutnya.

Fase kedua direncanakan mencakup pengembangan fitur tambahan dan penyempurnaan sistem secara menyeluruh. Ruang lingkup fase ini meliputi pembuatan tabel hasil pelaksanaan (*table result*) yang berfungsi untuk menampilkan data rekapitulasi secara detail, serta perluasan fungsi dashboard agar mencakup parameter tambahan yang lebih mendalam. Selain itu, pada fase ini juga direncanakan pengembangan fitur pelacakan

dan verifikasi yang terintegrasi dengan aplikasi *mobile*, sehingga proses validasi data dapat dilakukan secara langsung dari lapangan secara *real-time*. Pengembangan pada fase kedua ini bersifat melengkapi serta memperkuat sistem, khususnya dalam aspek *monitoring*, maintenance, dokumentasi, dan efisiensi operasional secara menyeluruh. Pembagian dua fase ini memberikan fleksibilitas dalam pengujian dan evaluasi berkelanjutan, serta memungkinkan setiap bagian sistem diuji secara independen sebelum dilakukan integrasi penuh.



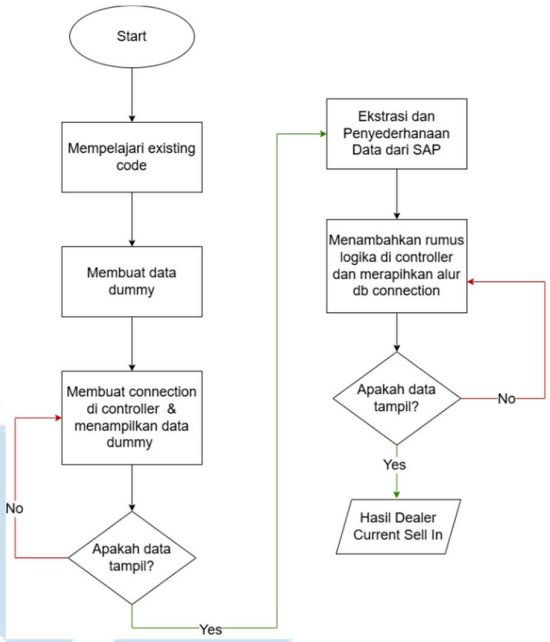
Gambar 3.21 Alur Roles

Pada website PGI Sales Portal terdapat beberapa jenis *roles* pengguna website. Namun, pada fase pertama pengembangan ini, fokus utama hanya diberikan pada tiga *roles*, yaitu *Sales, Approver,* dan *VM Admin*, sebagaimana ditampilkan pada Gambar 3.21.

3.2.4 Minggu 5 (24 April – 02 Mei 2025): Pengembangan Fitur Dealer Current Sell In

Setelah tahap perancangan antarmuka dan eksplorasi kode dasar diselesaikan, proses pengembangan berlanjut ke implementasi fitur pertama yang termasuk dalam fase pertama, yaitu *Dealer Current Sell In*.

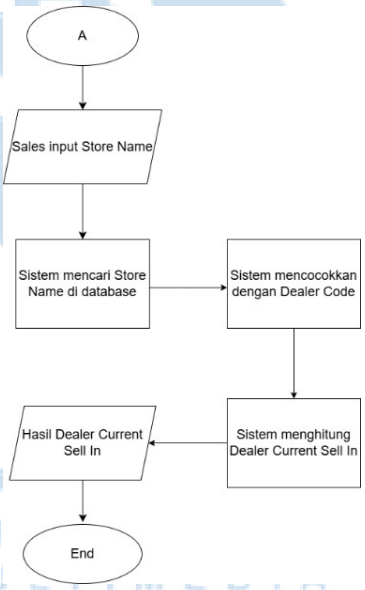
Berikut alur pengerjaan fitur *Dealer Current Sell In* yang terlihat pada Gambar 3.22.



Gambar 3.22 Flowchart Alur Pengerjaan Fitur Dealer Current Sell In

Gambar 3.22 memperlihatkan flowchart alur pengerjaan fitur *Dealer Current Sell In* yang dimulai dari tahap pemahaman terhadap kode yang sudah ada. Langkah ini bertujuan agar struktur dan alur kode yang dikembangkan tetap konsisten dengan sistem yang telah berjalan. Pada tahap ini juga dilakukan penelusuran terhadap alur logika program untuk memahami bagaimana data diproses dan mengalir antar bagian. Setelah memahami alur dasarnya, dibuatlah *data dummy* sebagai simulasi dari data riil. Tujuan pembuatan data ini adalah untuk menguji alur proses dan memastikan fitur dapat berjalan sebagaimana mestinya. Data *dummy* tersebut kemudian dihubungkan melalui *controller* dengan membuat koneksi ke database, lalu dicoba untuk ditampilkan pada *user interface* website. Jika data *dummy* tidak berhasil ditampilkan, maka dilakukan pengecekan ulang terhadap logika yang ada di dalam *controller*. Namun, apabila data berhasil muncul dengan baik, proses dilanjutkan ke tahap ekstraksi dan penyederhanaan data dari SAP. Di tahap ini, data *dummy*

dicocokkan dengan data riil dari SAP untuk memastikan kesesuaian dan akurasi. Selanjutnya, ditambahkan logika tambahan seperti rumus atau perhitungan yang dibutuhkan di dalam *controller*, serta dilakukan perapian alur koneksi data agar lebih efisien. Data hasil ekstraksi tersebut kemudian diuji kembali. Apabila data belum dapat ditampilkan dengan benar, maka perlu dilakukan pengecekan ulang baik pada logika di *controller* maupun pada bagian *JavaScript* yang berperan dalam proses penampilan data. Namun, jika seluruh data telah berhasil ditampilkan dengan baik dan sesuai harapan, maka fitur *Dealer Current Sell In* dapat dianggap telah berfungsi optimal dan siap untuk diterapkan dalam sistem.



Gambar 3.23 Flowchart Fitur Dealer Current Sell In Role Sales

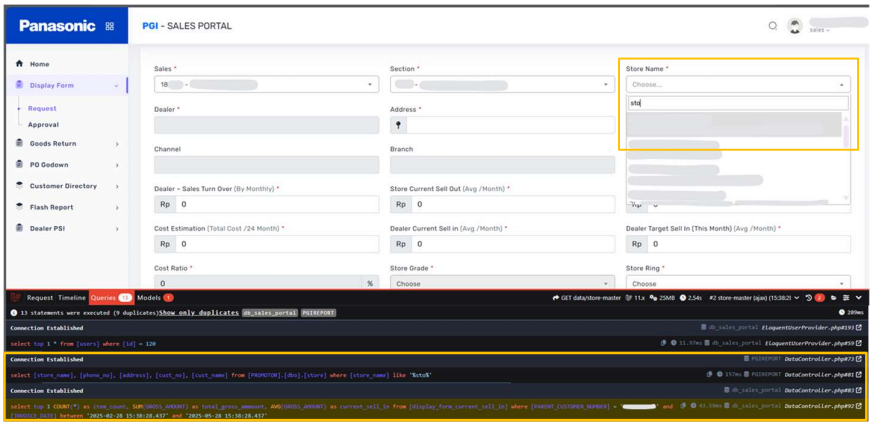
Gambar 3.23 memperlihatkan flowchart fitur *Dealer Current Sell In* pada *role sales* di website PGI Sales Portal, tepatnya di menu Display Form dengan submenu Request, terdapat sebuah form yang wajib diisi oleh pihak *requestor* atau *sales*. Form tersebut terdiri dari beberapa *field* yang harus diisi secara manual, serta beberapa *field* lain yang akan terisi secara otomatis apabila parameter tertentu terpenuhi. Salah satu contoh mekanisme *autofill* terjadi ketika *sales* menginput nilai pada field Store Name. Setelah data tersebut dimasukkan, sistem akan melakukan

pencarian data store yang sesuai di dalam database. Jika data ditemukan, sistem akan mencocokkannya dengan *Dealer Code* yang terkait dengan store tersebut. Setelah proses pencocokan selesai, sistem menghitung nilai *Dealer Current Sell In* menggunakan rumus yang telah ditentukan, yaitu berdasarkan rata-rata nilai *invoice* selama tiga bulan terakhir. Hasil perhitungan ini kemudian secara otomatis ditampilkan pada *field Dealer Current Sell In*.

	INVOICE_DATE	GROSS_AMOUNT	PARENT_CUSTOMER	PARENT_CUSTOMER_NUMBER	id
1	2025-03-22	5000000			1
2	2025-03-22	10000000			2
3	2025-03-22	8950000			3
4	2025-03-20	6213636			4
5	2025-03-20	9372727			5
6	2025-03-20	2118182			6
7	2025-03-20	7569874			7
8	2025-03-17	-453000			8
9	2025-03-17	4530000			9
10	2025-03-17	9240000			10
11	2025-03-17	7380000			11
12	2025-03-17	2075000			12
13	2025-03-17	1995000			13
14	2025-03-17	2500000			14
15	2025-03-17	7436000			15
16	2025-03-17	11154000			16
17	2025-03-17	25452000			17
18	2025-03-17	38178000			18
19	2025-03-17	18784000			19
20	2025-03-17	28176000			20
21	2025-03-17	15412000			21
22	2025-03-17	23118000			22
23	2025-03-11	29293635			23
24	2025-03-11	37140000			24
25	2025-03-11	7936365			25
26	2025-03-10	56808000			26
27	2025-03-10	85212000			27
28	2025-03-10	79104000			28
29	2025-03-10	118656000			29
30	2025-03-10	172500000			30

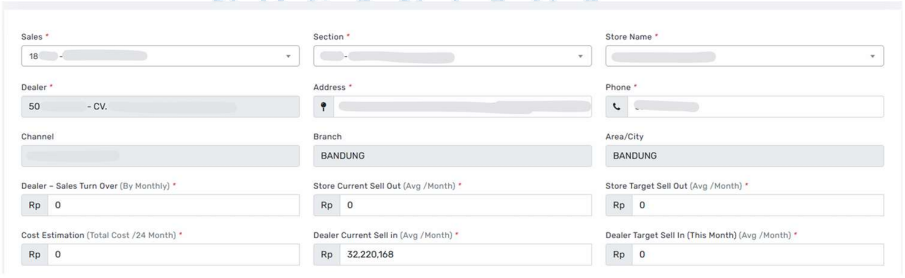
Gambar 3.24 Ekstraksi Data

Gambar 3.24 memperlihatkan cuplikan data hasil ekstraksi dari SAP yang telah disalin ke database lokal dalam bentuk tabel SQL. Data yang ditampilkan merupakan bagian yang telah disesuaikan dengan hanya mengambil kolom-kolom yang relevan terhadap kebutuhan sistem, antara lain *invoice_date*, *gross_amount*, *parent_customer*, dan *parent_customer_number*. Dengan membatasi hanya pada kolom-kolom yang diperlukan, proses pengolahan data menjadi lebih efisien dan fokus terhadap tujuan sistem.



Gambar 3.25 Mapping Data Antar Database

Pada Gambar 3.25 memperlihatkan proses *mapping* data yang dilakukan antar dua database berbeda dalam sistem. Pertama, sales memilih nama store pada *field* Store Name yang datanya berasal dari *database* pgireport. Setelah store yang dimaksud ditemukan, sistem akan mencocokkannya dengan dealer_code yang relevan. Dealer Code yang diperoleh selanjutnya digunakan untuk mengakses data transaksi penjualan dari database lain, yaitu db_sales_portal. Di dalam *database* ini, sistem akan mencari data invoice berdasarkan parent_customer_number di dalam *table* tersebut dan memfilter berdasarkan *invoice_date* dalam rentang tiga bulan terakhir. Nilai dari kolom *gross_amount* pada periode tersebut kemudian dijumlahkan dan digunakan sebagai dasar perhitungan untuk fitur *Dealer Current Sell In*.



Gambar 3.26 Tampilan Website Hasil Dealer Current Sell In

Pada Gambar 3.26 memperlihatkan tampilan website yang menunjukkan hasil dari perhitungan *Dealer Current Sell In*. Nilai ini

muncul secara otomatis setelah *requestor* ataupun *sales* memilih *Store Name*. Setelah pemilihan *Store Name* dilakukan, sistem akan menjalankan proses pencarian dan pencocokan data seperti yang telah dijelaskan sebelumnya, hingga akhirnya menghasilkan nilai rata-rata dari total invoice selama tiga bulan terakhir.

```php
public function findStoreMaster(Request $request)
{

    $store = DB::connection('pgireport')->table('PROMOTOR.dbo.store')S
        ->select('store_name', 'phone_no', 'address', 'cust_no', 'cust_name')
        ->where('store_name', 'like', "%{$request->search}%");

        ///Dealer Current Sell In

        $startDate = now()->subMonths(3);
        $endDate = now();

        $store=$store->get();
        $store=$store->map(function($storeItem) use ($startDate, $endDate){
        $currentSellIn = DB::connection('db_sales_portal')
            ->table('display_form_current_sell_in')
            ->select(
                DB::raw('COUNT(*)as item_count'),
                DB::raw('SUM(GROSS_AMOUNT) as total_gross_amount'),
                DB::raw('AVG(GROSS_AMOUNT) as current_sell_in'),
            )
            ->where('PARENT_CUSTOMER_NUMBER', $storeItem->cust_no)
            ->whereBetween('INVOICE_DATE', [$startDate, $endDate])
            ->first();

            $storeItem->item_count = $currentSellIn->item_count;
            $storeItem->total_gross_amount = $currentSellIn->total_gross_amount;
            $storeItem->current_sell_in = number_format($currentSellIn->current_sell_in, 0, ',', '.');

            return $storeItem;
        });
```

Gambar 3.27 Coding Controller untuk Perhitungan Dealer Current Sell In

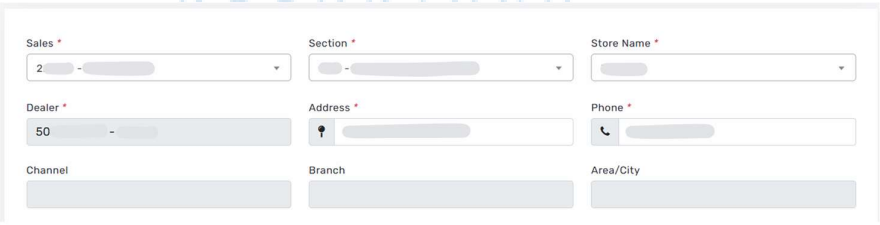
Pada Gambar 3.27 memperlihatkan potongan kode dari controller untuk perhitungan *Dealer Current Sell In* dengan penglogikaan yang telah dijelaskan sebelumnya.

Gambar 3.28 Coding JavaScript Dealer Current Sell In

Pada Gambar 3.28 memperlihatkan potongan kode dari JavaScript untuk proses penerapan *autofill* untuk *Dealer Current Sell In*.

3.2.5 Minggu 6 (05 Mei – 09 Mei 2025): Troubleshooting dan Pengembangan Fitur Priority

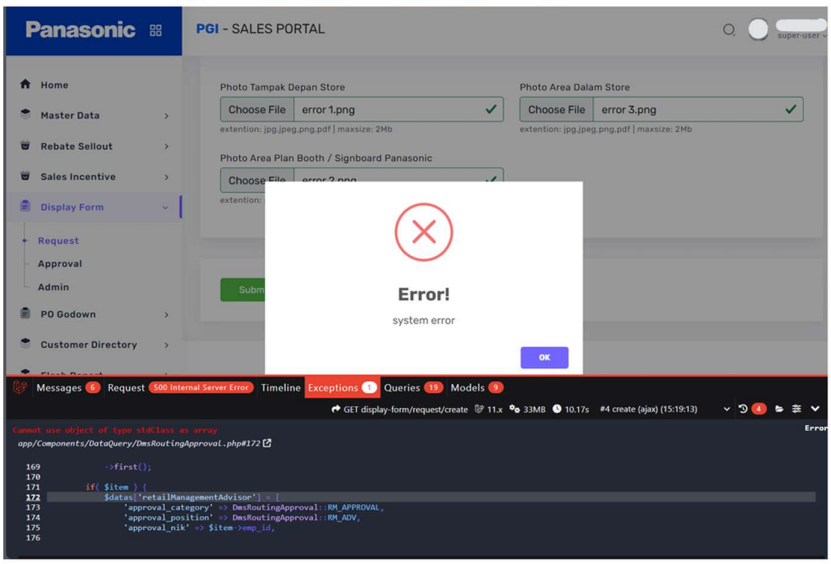
Setelah proses perhitungan Dealer Current Sell In selesai, tahapan berikutnya adalah pengembangan fitur *Priority*. Namun, sebelum memulai pengembangan fitur tersebut, ditemukan beberapa permasalahan pada sistem yang memerlukan proses *troubleshooting*. Permasalahan ini menyebabkan kegagalan dalam melakukan *submit* pada form request saat percobaan awal implementasi fitur *Priority*.



Gambar 3.29 Column yang terdisable

Pada Gambar 3.29 memperlihatkan tampilan website di mana field Channel, Branch, dan Area/City berada dalam kondisi *disabled*, sehingga tidak dapat diakses maupun menampilkan data yang seharusnya muncul

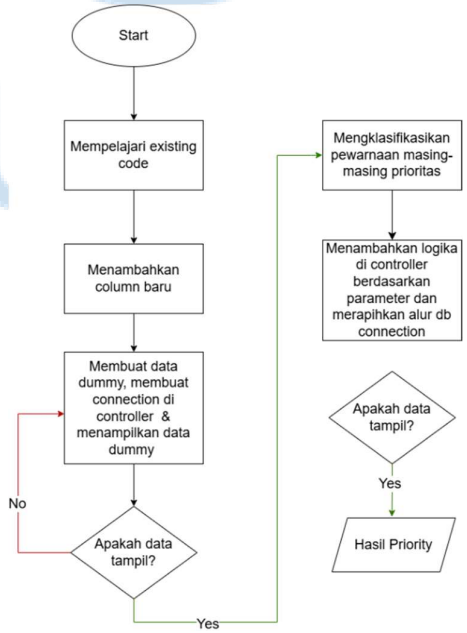
secara otomatis. Ketiga field ini idealnya berada dalam kondisi *enabled* karena informasi yang ditampilkannya diperlukan untuk melengkapi data form secara keseluruhan, sekaligus mendukung logika sistem yang telah dirancang. Permasalahan ini menjadi cukup signifikan untuk segera diselesaikan, mengingat field Area/City merupakan salah satu parameter utama dalam proses perhitungan nilai *Priority*. Setelah dilakukan penelusuran lebih lanjut, diketahui bahwa kondisi *disabled* pada field tersebut disebabkan oleh kegagalan sistem dalam memuat data dari database yang terkait. Permasalahan ini ternyata berasal dari konfigurasi file php.ini pada environment backend, di mana terdapat ekstensi PHP yang belum aktif karena masih ditandai dengan simbol titik koma (;) di awal baris. Setelah simbol tersebut dihapus dan server dijalankan ulang, ekstensi berhasil diaktifkan dan sistem kembali dapat mengambil serta menampilkan data dengan semestinya. Dengan demikian, permasalahan tersebut berhasil diselesaikan melalui proses *troubleshooting* konfigurasi server, sebelum pengembangan sistem dilanjutkan ke tahap berikutnya, yaitu implementasi fitur *Priority*.



Gambar 3.30 Tampilan Submit Error

Pada Gambar 3.30 memperlihatkan tampilan website pada saat melakukan proses submit pada form. Namun, proses tersebut gagal dan menampilkan pesan error dengan kode 500 Internal Server Error. Error ini menunjukkan bahwa terjadi gangguan pada sisi server, yang disebabkan oleh nilai *emp_id* yang terdeteksi *null* pada saat request diproses. Setelah dilakukan penelusuran, permasalahan tersebut diketahui berasal dari versi kode lokal yang belum sepenuhnya sinkron dengan versi terbaru di repository. Permasalahan ini berhasil diselesaikan dengan melakukan proses *git pull* untuk menarik pembaruan kode, sehingga sistem kembali berjalan normal dan proses submit dapat dilakukan tanpa kendala.

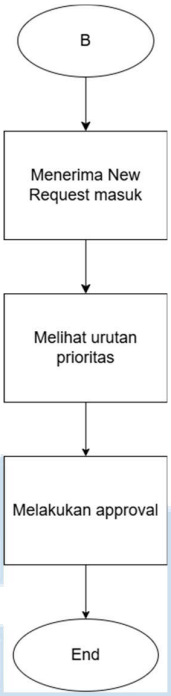
Setelah troubleshooting diselesaikan, proses pengembangan berlanjut ke implementasi fitur kedua yang termasuk dalam fase pertama, yaitu *Priority*. Berikut alur pengerjaan fitur *Priority* yang terlihat pada Gambar 3.31.



Gambar 3.31 Flowchart Alur Pengerjaan Fitur Priority

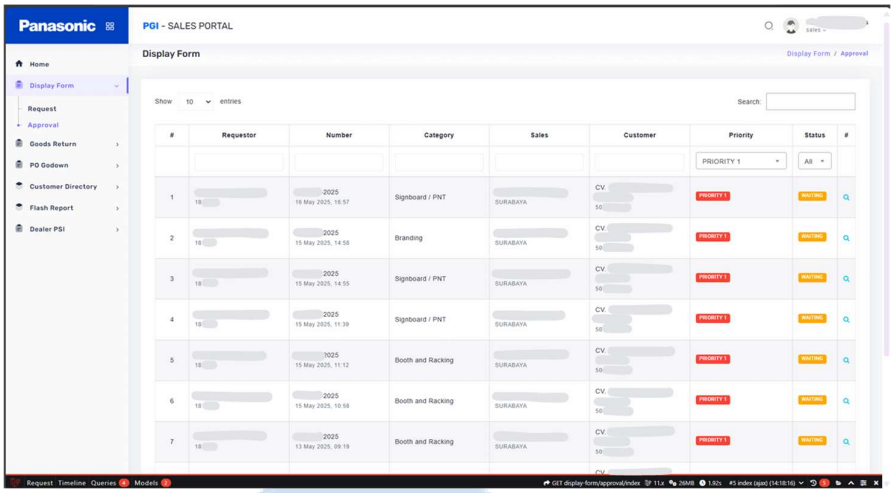
Gambar 3.31 memperlihatkan flowchart alur pengerjaan fitur *Priority* yang dimulai dari tahap pemahaman terhadap kode yang sudah ada.

Langkah ini bertujuan agar struktur dan alur kode yang dikembangkan tetap konsisten dengan sistem yang telah berjalan. Pada tahap ini juga dilakukan penelusuran terhadap alur logika program untuk memahami bagaimana data diproses dan mengalir antar bagian. Setelah memahami alur dasarnya, ditambahkanlah column priority sebagai column baru pada tampilan table di halaman approval. Sebagai langkah awal, tahap selanjutnya yaitu dibuatlah *data dummy* sebagai simulasi dari data riil. Tujuan pembuatan data ini adalah untuk menguji alur proses dan memastikan fitur dapat berjalan sebagaimana mestinya. Data *dummy* tersebut kemudian dihubungkan melalui *controller* dengan membuat koneksi ke database, lalu dicoba untuk ditampilkan pada *user interface* website. Jika data *dummy* tidak berhasil ditampilkan, maka dilakukan pengecekan ulang terhadap logika yang ada di dalam *controller*. Namun, apabila data berhasil muncul dengan baik, proses dilanjutkan ke tahap pengklasifikasian pewarnaan masing-masing prioritas. Selanjutnya, ditambahkan logika dalam penentuan parameter yang menentukan tiap-tiap prioritas yang dibutuhkan di dalam *controller*, serta dilakukan perapian alur koneksi data agar lebih efisien. Data hasil penentuan parameter tersebut kemudian diuji kembali. Apabila data belum dapat ditampilkan dengan benar, maka perlu dilakukan pengecekan ulang baik pada logika di *controller* yang berperan dalam proses penentuan logika data yang akan tampil. Namun, jika seluruh data telah berhasil ditampilkan dengan baik dan sesuai harapan, maka fitur *Priority* dapat dianggap telah berfungsi optimal dan siap untuk diterapkan dalam sistem.



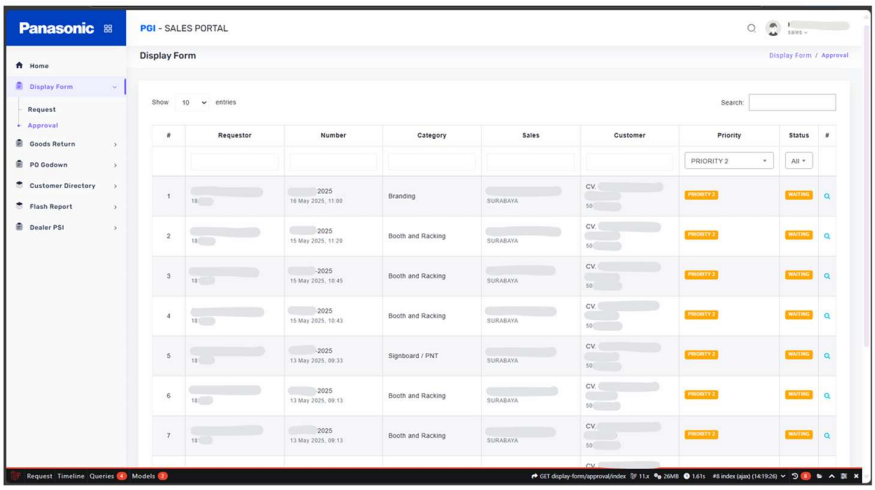
Gambar 3.32 Flowchart Fitur Priority Role Approver

Gambar 3.32 memperlihatkan *flowchart* alur kerja fitur *Priority* pada *role Approver*. Pada diagram tersebut, ketika seorang *approver* menerima sebuah *request* baru yang masuk, sistem akan menampilkan urutan prioritas dari *request* tersebut berdasarkan parameter yang telah ditentukan sebelumnya. Selanjutnya, *approver* dapat melakukan proses *approval* dengan mempertimbangkan urutan prioritas tersebut, sehingga proses dapat berjalan secara lebih terstruktur dan efisien.



Gambar 3.33 Tampilan Website Hasil Priority 1 dengan Status All

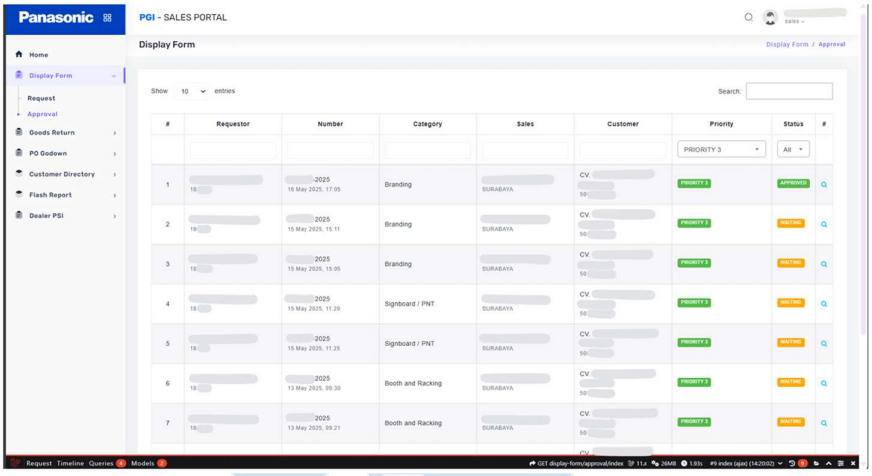
Gambar 3.33 memperlihatkan tampilan website ketika fitur *Priority* difilter pada nilai *Priority 1* saja. Pada kondisi tersebut, sistem hanya menampilkan data permintaan dengan tingkat prioritas *Priority 1*, tanpa membatasi status permintaan, sehingga seluruh status tetap ditampilkan secara menyeluruh (*All Status*).



Gambar 3.34 Tampilan Website Hasil Priority 2 dengan Status All

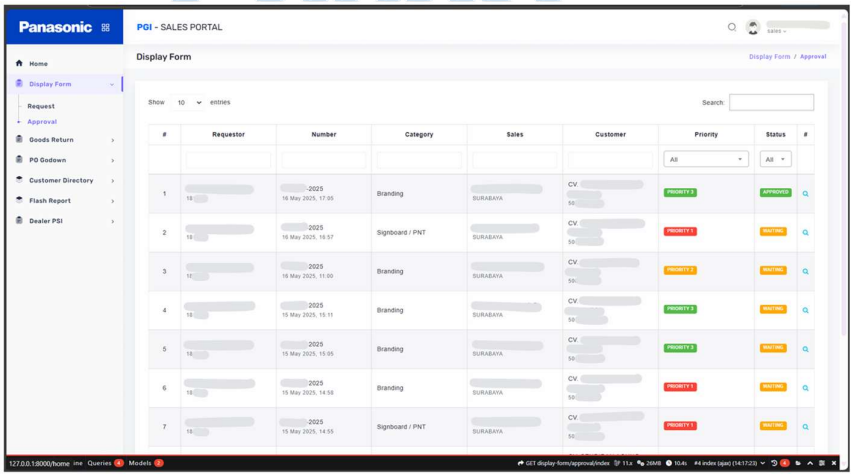
Gambar 3.34 memperlihatkan tampilan website ketika fitur *Priority* difilter pada nilai *Priority 2* saja. Pada kondisi tersebut, sistem hanya menampilkan data permintaan dengan tingkat prioritas *Priority 2*, tanpa

membatasi status permintaan, sehingga seluruh status tetap ditampilkan secara menyeluruh (*All Status*).



Gambar 3.35 Tampilan Website Hasil Priority 3 dengan Status All

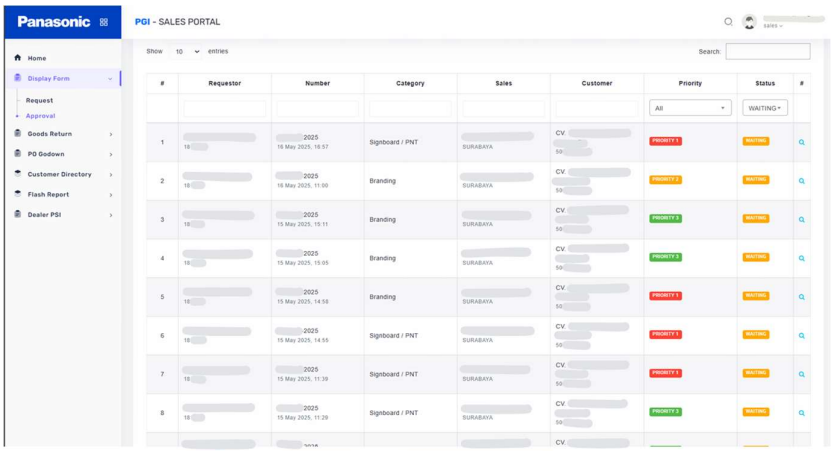
Gambar 3.35 memperlihatkan tampilan website ketika fitur *Priority* difilter pada nilai *Priority 3* saja. Pada kondisi tersebut, sistem hanya menampilkan data permintaan dengan tingkat prioritas *Priority 3*, tanpa membatasi status permintaan, sehingga seluruh status tetap ditampilkan secara menyeluruh (*All Status*).



Gambar 3.36 Tampilan Website Hasil Priority All dengan Status All

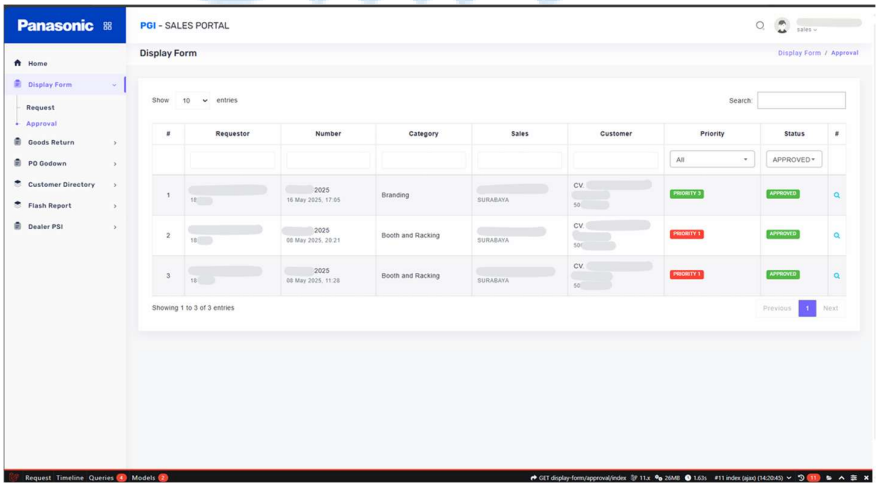
Gambar 3.36 memperlihatkan tampilan website ketika fitur *Priority* diset pada opsi *All*. Pada kondisi tersebut, sistem menampilkan seluruh

data permintaan tanpa membatasi tingkat prioritas, sehingga seluruh kategori *Priority* ditampilkan. Status permintaan tidak difilter, sehingga semua status tetap muncul secara menyeluruh (*All Status*).



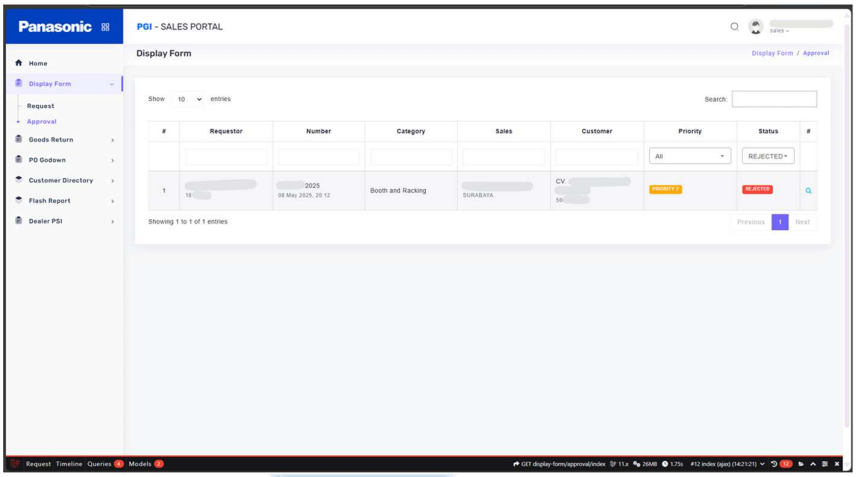
Gambar 3.37 Tampilan Website Hasil Priority All dengan Status Waiting

Gambar 3.37 memperlihatkan tampilan website ketika fitur *Priority* diset pada opsi *All*. Pada kondisi tersebut, sistem menampilkan seluruh data permintaan tanpa membatasi tingkat prioritas, sehingga seluruh kategori *Priority* ditampilkan. Status permintaan difilter pada nilai *Waiting*, sehingga sistem hanya menampilkan permintaan dengan status *Waiting* saja, sedangkan status lainnya tidak ditampilkan.



Gambar 3.38 Tampilan Website Hasil Priority All dengan Status Approved

Gambar 3.38 memperlihatkan tampilan website ketika fitur *Priority* diset pada opsi *All*. Pada kondisi tersebut, sistem menampilkan seluruh data permintaan tanpa membatasi tingkat prioritas, sehingga seluruh kategori *Priority* ditampilkan. Status permintaan difilter pada nilai *Approved*, sehingga sistem hanya menampilkan permintaan dengan status *Approved* saja, sedangkan status lainnya tidak ditampilkan.



Gambar 3.39 Tampilan Website Hasil Priority All dengan Status Rejected

Gambar 3.39 memperlihatkan tampilan website ketika fitur *Priority* diset pada opsi *All*. Pada kondisi tersebut, sistem menampilkan seluruh data permintaan tanpa membatasi tingkat prioritas, sehingga seluruh kategori *Priority* ditampilkan. Status permintaan difilter pada nilai *Rejected*, sehingga sistem hanya menampilkan permintaan dengan status *Rejected* saja, sedangkan status lainnya tidak ditampilkan.

```
<table class="table table-striped table-bordered table-hover" id="datatables">
    <thead>
    <tr>
        <th style="width: 5%" class="text-center">#</th>
        <th style="width: 15%" class="text-center">Requestor</th>
        <th style="width: 15%" class="text-center">Number</th>
        <th style="width: 15%" class="text-center">Category</th>
        <th style="width: 15%" class="text-center">Sales</th>
        <th style="width: 15%" class="text-center">Customer</th>
        <th style="width: 15%" class="text-center">Priority</th>
        <th style="width: 15%" class="text-center">Status</th>
        <th style="width: 5%" class="text-center">#</th>
    </tr>
```

Gambar 3.40 Coding Penambahan Column Priority

Gambar 3.40 memperlihatkan hasil coding dari blade.php untuk proses penambahan *column Priority*.

```
<td>
    <select class="form-control text-left form-search select2" id="f-priority">
        <option value="">All</option>
        <option value="PRIORITY 1" selected>PRIORITY 1</option>
        <option value="PRIORITY 2">PRIORITY 2</option>
        <option value="PRIORITY 3">PRIORITY 3</option>
    </select>
</td>
```

Gambar 3.41 Coding Penambahan Option Priority

Gambar 3.40 memperlihatkan hasil coding dari blade.php untuk proses penambahan *column Priority*.

created_at	updated_at	roi	add_sell_in	priority
2025-05-13 09:04:41.000	2025-05-13 09:04:41.000	185.52	2342424	PRIORITY 3
2025-05-13 09:05:46.000	2025-05-13 09:05:46.000	185.52	2342424	PRIORITY 3
2025-05-13 09:08:34.000	2025-05-13 09:08:34.000	17.82	23524242	PRIORITY 3
2025-05-13 09:10:42.000	2025-05-13 09:10:42.000	17.82	23524242	PRIORITY 3
2025-05-13 09:11:46.000	2025-05-13 09:11:46.000	17.82	23524242	PRIORITY 3
2025-05-13 09:13:24.000	2025-05-13 09:13:24.000	1.88	3453535	PRIORITY 2
2025-05-13 09:13:55.000	2025-05-13 09:13:55.000	1.88	3453535	PRIORITY 2
2025-05-13 09:19:43.000	2025-05-13 09:19:43.000	1859.84	3424573081	PRIORITY 1
2025-05-13 09:21:26.000	2025-05-13 09:21:26.000	1859.84	3424573081	PRIORITY 3
2025-05-13 09:30:19.000	2025-05-13 09:30:19.000	1859.84	3424573081	PRIORITY 3
2025-05-13 09:33:29.000	2025-05-13 09:33:29.000	1874	34506390997	PRIORITY 2
2025-05-15 10:43:36.000	2025-05-15 10:43:36.000	1275.78	23424234	PRIORITY 2
2025-05-15 10:45:22.000	2025-05-15 10:45:22.000	19199.54	3453453454	PRIORITY 2
2025-05-15 10:58:40.000	2025-05-15 10:58:40.000	1.86	23423	PRIORITY 1
2025-05-15 11:12:26.000	2025-05-15 11:12:26.000	14.35	242394272192	PRIORITY 1
2025-05-15 11:20:02.000	2025-05-15 11:20:02.000	1920	2352353	PRIORITY 2
2025-05-15 11:25:16.000	2025-05-15 11:25:16.000	355304.87	4353453453434	PRIORITY 3
2025-05-15 11:29:36.000	2025-05-15 11:29:36.000	19.65	35353344	PRIORITY 3
2025-05-15 11:39:30.000	2025-05-15 11:39:30.000	19200.06	3535354234	PRIORITY 1
2025-05-15 14:55:43.000	2025-05-15 14:55:43.000	28.76	3523423425	PRIORITY 1
2025-05-15 14:58:03.000	2025-05-15 14:58:03.000	197.64	6768698695	PRIORITY 1
2025-05-15 15:05:16.000	2025-05-15 15:05:16.000	1824.05	32523423453	PRIORITY 3
2025-05-15 15:11:11.000	2025-05-15 15:11:11.000	0.17	68678675	PRIORITY 3
2025-05-16 11:00:09.000	2025-05-16 11:00:09.000	18.64	33242534	PRIORITY 2
2025-05-16 16:57:52.000	2025-05-16 16:57:52.000	16.84	206272000	PRIORITY 1
2025-05-16 17:05:25.000	2025-05-16 17:05:25.000	1.89	2313461182	PRIORITY 3

Gambar 3.42 Data Priority

Gambar 3.42 memperlihatkan tampilan data pada tabel *Display Form* yang memuat keseluruhan data *request* form. Pada tahap awal, ditambahkan satu kolom baru untuk menyimpan nilai *Priority*. Proses pengujian dimulai dengan memasukkan data *dummy* tanpa parameter atau logika khusus sebagai percobaan awal, kemudian dilanjutkan dengan implementasi pengisian data berdasarkan parameter dan logika prioritas yang telah dirancang.

```
1 reference
const PRIORITY1 = 'PRIORITY 1';
1 reference
const PRIORITY2 = 'PRIORITY 2';
1 reference
const PRIORITY3 = 'PRIORITY 3';

1 reference
public static $priority = [
    self::PRIORITY1 => ['color' => 'danger', 'icon' => 'fa fa-clock-o'],
    self::PRIORITY2 => ['color' => 'warning', 'icon' => 'fa fa-clock-o'],
    self::PRIORITY3 => ['color' => 'success', 'icon' => 'fa fa-check'],
];
```

Gambar 3.43 Coding Implementasi Konstanta dan Mapping Visual Prioritas

Gambar 3.43 memperlihatkan potongan kode yang mendefinisikan konstanta prioritas serta pemetaan atribut visual berupa warna dan ikon untuk tiap level prioritas.

```
->addColumn('priority', function ($item): string {
    $priority = ApprovalStatus::$priority[$item->priority];
    return "<label class='badge badge-{$priority['color']}'>{$item->priority}</label>";
})
```

Gambar 3.44 Coding Implementasi Column Prioritas dengan Badge Berwarna

Gambar 3.44 memperlihatkan potongan kode yang berfungsi untuk menambahkan kolom bernama *priority* pada tabel data. Kode ini menggunakan array statis $priority di kelas ApprovalStatus sebagai acuan untuk memetakan nilai prioritas menjadi label berwarna khusus. Implementasi tersebut menghasilkan tampilan badge berwarna yang merepresentasikan tingkat prioritas secara visual, sehingga pengguna dapat dengan mudah mengidentifikasi tingkat urgensi data berdasarkan warna badge, seperti merah untuk prioritas tinggi, kuning untuk sedang, dan hijau untuk prioritas rendah.

```
if ($params['priority'] && $params['priority'] != '') {
    $model->where(column: 't2.priority', operator: $params['priority']);
}
```

Gambar 3.45 Coding Implementasi Filter Prioritas

Gambar 3.45 memperlihatkan potongan kode yang digunakan untuk memfilter data berdasarkan nilai prioritas yang dipilih oleh *approver*. Pada proses ini, sistem akan menambahkan klausa WHERE pada query apabila parameter *priority* tidak bernilai kosong, sehingga hanya data dengan prioritas yang sesuai akan ditampilkan.

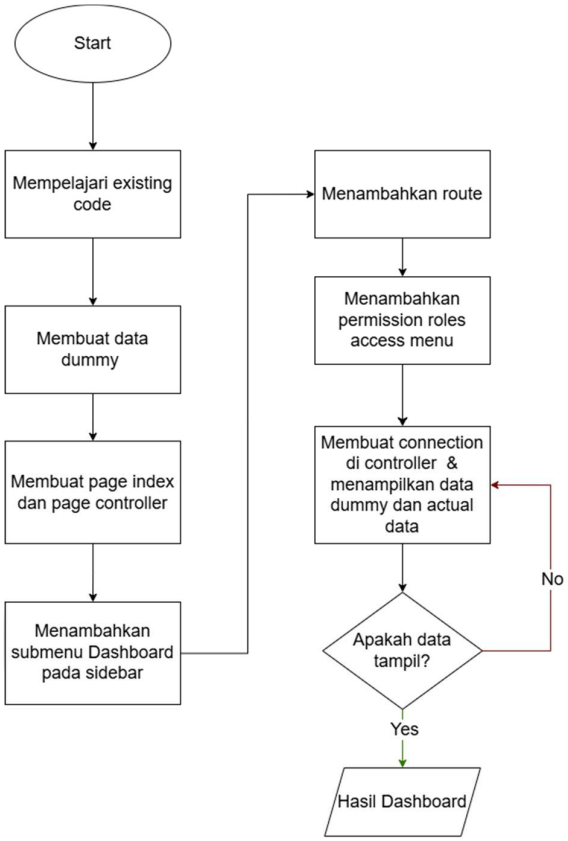
```php
$area = $request->post(key: 'area');
// $store_grade = $request->post('store_grade');
$rings = $request->post(key: 'store_ring');

$priority = '';

// Priority 1
$priority_1_areas = ['Sumatera Barat', 'Bengkulu', 'Jawa Timur', 'Lombok', 'Kalimantan Barat',
                     'Kalimantan Utara', 'Sulawesi Tengah', 'Sulawesi Tenggara', 'Ternate'];

if (in_array(needle: $area, haystack: $priority_1_areas) || $store_grade == 'A'|| $rings == 'VIP') {
    $priority = 'PRIORITY 1';
}
// Priority 2
elseif (!in_array(needle: $area, haystack: $priority_1_areas) && $store_grade == 'B' && $rings == 'Ring 1' || $rings == 'Ring 2'){
    $priority = 'PRIORITY 2';
}
// Priority 3
elseif (!in_array(needle: $area, haystack: $priority_1_areas) && $store_grade == 'C') {
    $priority = 'PRIORITY 3';
}
```

Gambar 3.46 Coding Penentuan Prioritas

Gambar 3.46 memperlihatkan potongan kode yang digunakan untuk menentukan level prioritas berdasarkan input yang diberikan oleh sales, seperti *area*, *store ring*, dan *store grade*. Kode ini dieksekusi ketika form disubmit, ditandai dengan adanya pengecekan terhadap metode request menggunakan if ($request->isMethod('post')). Pada tahap ini, sistem akan mengambil data input melalui metode POST, kemudian melakukan proses klasifikasi prioritas dengan struktur logika kondisional. *Area* tertentu, *store grade* yang tinggi (A), dan status *store ring* seperti “VIP”, “Ring 1”, “Ring 2” akan diklasifikasikan sebagai PRIORITY 1. Sementara kombinasi area lain dengan grade B dan ring 1 atau 2 akan dikategorikan sebagai PRIORITY 2, dan sisanya seperti grade C akan masuk ke dalam PRIORITY 3.

3.2.6 Minggu 7 (13 Mei – 16 Mei 2025): Pengembangan Dashboard Project Overview

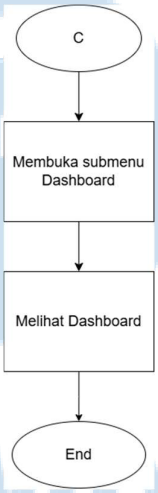
Setelah proses penambahan *column* dan logika *Priority* selesai, tahapan berikutnya adalah pengembangan fitur *Dashboard*. Berikut alur pengerjaan *Dashboard* yang terlihat pada Gambar 3.47.



Gambar 3.47 Flowchart Alur Pengerjaan Fitur Dashboard

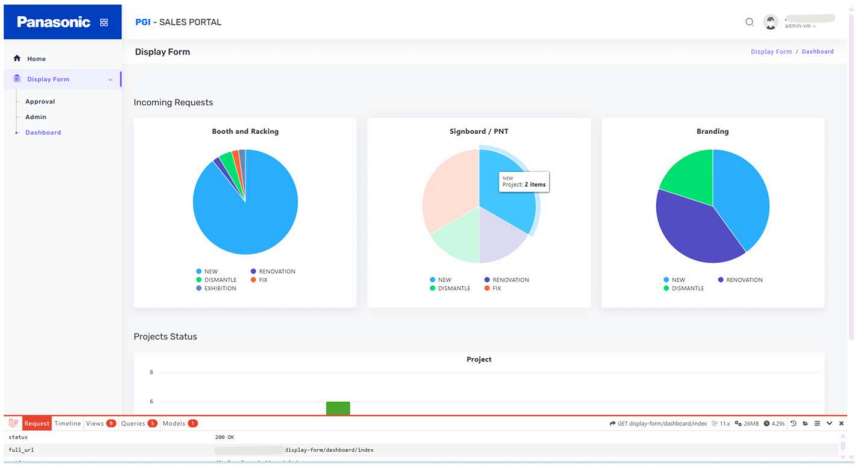
Gambar 3.47 memperlihatkan flowchart alur pengerjaan fitur Dashboard yang dimulai dari tahap pemahaman terhadap kode yang telah ada. Langkah ini bertujuan untuk memastikan bahwa struktur dan alur kode yang dikembangkan tetap konsisten dengan sistem yang sudah berjalan. Pada tahap ini juga dilakukan penelusuran alur logika program guna memahami bagaimana data diproses dan mengalir antar bagian sistem. Setelah memahami alur dasar tersebut, dibuat data *dummy* sebagai simulasi data riil. Pembuatan data ini bertujuan untuk menguji alur proses dan memastikan fitur dapat berjalan sesuai dengan yang diharapkan. Selanjutnya, dilakukan pembuatan halaman index dan controller sebagai langkah awal dalam pembuatan submenu baru, yaitu submenu *Dashboard* pada menu sidebar. Setelah itu, *route* ditambahkan agar submenu dapat diakses dengan benar. Kemudian, *permission roles* ditambahkan untuk mengatur akses hanya bagi *role* tertentu yang

berwenang mengakses submenu Dashboard. Jika seluruh tahapan ini berjalan dengan lancar, tahap berikutnya adalah menghubungkan data *dummy* dan data aktual melalui controller yang melakukan koneksi ke *database*, kemudian menampilkannya pada *user interface* website. Apabila data tidak berhasil ditampilkan, dilakukan pengecekan ulang terhadap logika pada controller. Namun, jika seluruh data berhasil ditampilkan dengan baik dan sesuai harapan, maka fitur Dashboard dianggap telah berfungsi secara optimal dan siap untuk diterapkan dalam sistem.



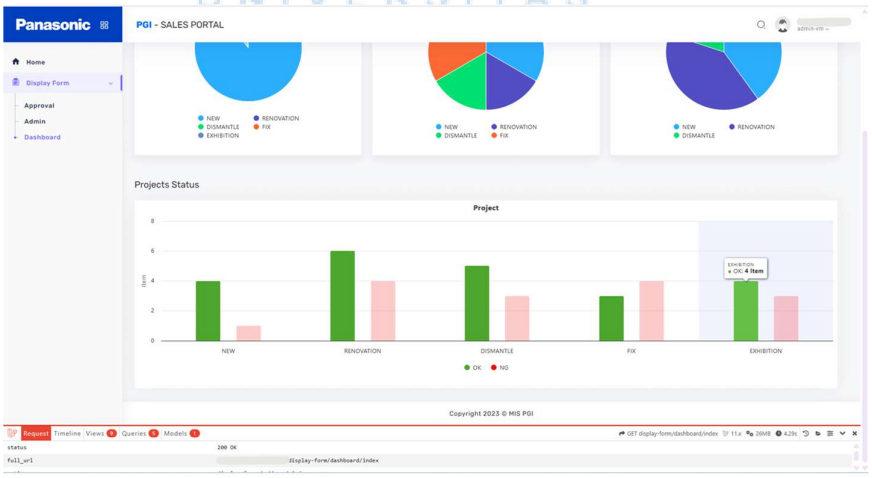
Gambar 3.48 Flowchart Fitur Dashboard Role VM Admin

Gambar 3.48 memperlihatkan flowchart alur kerja fitur *Dashboard* pada role VM Admin. Pada diagram tersebut, proses dimulai ketika VM Admin membuka submenu *Dashboard* yang tersedia pada sistem. Setelah submenu diakses, sistem akan menampilkan rangkuman visual berupa grafik yang berisi informasi terkait status *project*, baik dari segi hasil (OK/NG), maupun jenis *category project*. Informasi yang disajikan pada tampilan *Dashboard* ini membantu VM Admin dalam memantau progres dan kinerja project secara menyeluruh.



Gambar 3.49 Tampilan Website Hasil Dashboard

Gambar 3.49 memperlihatkan tampilan website yang menampilkan hasil dari fitur *Dashboard* yang telah dikembangkan. Pada tampilan tersebut, terdapat tiga pie chart yang masing-masing merepresentasikan kategori yang berbeda, yaitu *Booth and Racking, Signboard / PNT,* dan *Branding*. Setiap kategori memuat data *project* yang bervariasi sesuai dengan jenisnya, dan jumlah item yang ditampilkan dalam setiap grafik dihitung berdasarkan data request yang masuk ke dalam sistem. Visualisasi ini bertujuan untuk memberikan gambaran ringkas dan informatif terkait distribusi *project* yang sedang berjalan maupun telah diproses.



Gambar 3.50 Tampilan Website Hasil Dashboard 2

Gambar 3.50 memperlihatkan tampilan website yang menampilkan hasil dari fitur *Dashboard* yang telah dikembangkan. Pada tampilan tersebut, terdapat bar chart yang masing-masing merepresentasikan *project* yang berbeda, yaitu *New, Rennovation, Fix, Exhibition*. Setiap *project* memuat data OK dan NG yang bervariasi. jumlah OK dan NG yang ditampilkan dalam setiap grafik dihitung berdasarkan dari hasil proses verifikasi atau pengecekan terhadap setiap *project* yang diajukan.

	project	results_status	id
1	NEW	OK	1
2	NEW	NG	2
3	NEW	OK	3
4	NEW	OK	4
5	NEW	OK	5
6	RENOVATION	OK	6
7	RENOVATION	OK	7
8	RENOVATION	OK	8
9	RENOVATION	NG	9
10	RENOVATION	NG	10
11	RENOVATION	NG	11
12	RENOVATION	NG	12
13	RENOVATION	OK	13
14	RENOVATION	OK	14
15	RENOVATION	OK	15
16	FIX	OK	16
17	FIX	OK	17
18	FIX	OK	18
19	FIX	NG	19
20	FIX	NG	20
21	FIX	NG	21
22	FIX	NG	22
23	DISMANTLE	OK	23
24	DISMANTLE	OK	24
25	DISMANTLE	OK	25
26	DISMANTLE	OK	26
27	DISMANTLE	OK	27
28	DISMANTLE	NG	28
29	DISMANTLE	NG	29

Gambar 3.51 Data Dummy Dashboard Bar Chart

Gambar 3.51 memperlihatkan tampilan data pada tabel *Dashboard* yang berisi keseluruhan data *dummy* yang digunakan untuk keperluan visualisasi *bar chart*. Tabel tersebut memuat beberapa kolom penting, antara lain *nama project* dan *result_status*. Nilai dari *result_status* dikategorikan ke dalam dua klasifikasi, yaitu OK dan NG, yang masing-masing merepresentasikan hasil proses verifikasi atau pengecekan terhadap setiap *project* yang diajukan. Penyusunan data ini bertujuan untuk menguji alur visualisasi serta memastikan keakuratan pengelompokan data pada grafik yang akan ditampilkan.

Results Messages

	request_title	project
35	Booth and Racking	NEW
36	Booth and Racking	NEW
37	Booth and Racking	NEW
38	Booth and Racking	NEW
39	Booth and Racking	NEW
40	Booth and Racking	NEW
41	Booth and Racking	NEW
42	Booth and Racking	NEW
43	Signboard / PNT	NEW
44	Booth and Racking	RENOVATION
45	Booth and Racking	FIX
46	Booth and Racking	DISMANTLE
47	Booth and Racking	DISMANTLE
48	Booth and Racking	EXHIBITION
49	Signboard / PNT	NEW
50	Signboard / PNT	RENOVATION
51	Signboard / PNT	FIX
52	Signboard / PNT	DISMANTLE
53	Branding	NEW
54	Branding	RENOVATION
55	Branding	DISMANTLE
56	Branding	RENOVATION
57	Signboard / PNT	FIX
58	Branding	NEW

Gambar 3.52 Data Aktual Dashboard Pie Chart

Gambar 3.52 memperlihatkan tampilan data pada tabel *Display Form* yang berisi seluruh data *request* yang masuk ke dalam sistem. Tabel ini hanya menampilkan dua kolom utama, yaitu *project* dan *request_title*. Data dari kedua kolom tersebut digunakan oleh sistem untuk menghitung total jumlah *project* berdasarkan kategori yang ditentukan oleh nilai pada kolom *request_title*. Setiap nilai *request_title* mewakili satu kategori, dan di dalamnya terdapat beberapa *project* yang tercatat pada kolom *project*. Sistem kemudian menghitung total *project* untuk masing-masing kategori. Sebagai contoh, pada kategori *Booth and Racking*, dapat ditemukan sejumlah *project* dengan nama "New", dan total *project* ini akan dihitung untuk ditampilkan dalam visualisasi grafik pada *Dashboard*. Proses ini memastikan bahwa penyajian data dalam bentuk grafik benar-benar mencerminkan distribusi *project* sesuai kategori yang ada.

```
class MenuStructure
{
    7 references
    const DISPLAY_FORM = 'display-form';
    6 references
    const DISPLAY_FORM_REQUEST = 'display-form-request';
    11 references
    const DISPLAY_FORM_APPROVAL = 'display-form-approval';
    7 references
    const DISPLAY_FORM_ADMIN = 'display-form-admin';
    2 references
    const DISPLAY_FORM_DASHBOARD = 'display-form-dashboard';
```

Gambar 3.53 Coding Penambahan Submenu Dashboard 1

Gambar 3.53 memperlihatkan bagian awal dari proses penambahan submenu *Dashboard* dalam sistem, yaitu dengan mendeklarasikan beberapa konstanta di dalam *class MenuStructure*. Konstanta tersebut digunakan sebagai penanda atau identifikasi untuk masing-masing menu dan submenu yang ada di modul *Display Form*. Penambahan DISPLAY_FORM_DASHBOARD secara khusus menjadi langkah pertama dalam proses integrasi fitur Dashboard ke dalam sistem. Konstanta ini nantinya akan digunakan sebagai referensi utama pada bagian lain, seperti pemetaan menu di fungsi list(), pembuatan route, dan pengaturan hak akses.

```
public static function list(): array
{
    return [
        self::DISPLAY_FORM => ['title' => 'Display Form', 'icon' => 'form', 'url' => null, 'parent' => 1],
        self::DISPLAY_FORM_REQUEST => ['title' => 'Request', 'url' => route(name: 'display-form.request.index'), 'parent' => 0],
        self::DISPLAY_FORM_APPROVAL => ['title' => 'Approval', 'url' => route(name: 'display-form.approval.index'), 'parent' => 0],
        self::DISPLAY_FORM_ADMIN => ['title' => 'Admin', 'url' => route(name: 'display-form.admin.index'), 'parent' => 0],
        self::DISPLAY_FORM_DASHBOARD => ['title' => 'Dashboard', 'url' => route(name: 'display-form.dashboard.index'), 'parent' => 0],
```

Gambar 3.54 Coding Penambahan Submenu Dashboard 2

Gambar 3.54 memperlihatkan potongan kode dari fungsi list() yang terdapat pada class *MenuStructure* sebelumnya. Kode ini berfungsi untuk mendefinisikan daftar menu dan submenu yang akan muncul di sidebar sistem, khususnya pada menu *Display Form*. Masing-masing submenu direpresentasikan menggunakan key yang mengacu pada konstanta yang sudah didefinisikan sebelumnya. Secara keseluruhan, penambahan ini dilakukan sebagai langkah awal dalam proses integrasi fitur Dashboard

ke dalam sistem, supaya menu tersebut bisa muncul dan diakses dari sidebar utama oleh user dengan role tertentu.

```
1 reference
public static $group = [

    self::DISPLAY_FORM => [
        self::DISPLAY_FORM_REQUEST => self::DISPLAY_FORM_REQUEST,
        self::DISPLAY_FORM_APPROVAL => self::DISPLAY_FORM_APPROVAL,
        self::DISPLAY_FORM_ADMIN => self::DISPLAY_FORM_ADMIN,
        self::DISPLAY_FORM_DASHBOARD => self::DISPLAY_FORM_DASHBOARD
    ],
```

Gambar 3.55 Coding Penambahan Submenu Dashboard 3

Gambar 3.55 memperlihatkan dari kode program yang mendefinisikan struktur $group dalam *class* *MenuStructure*. Kode ini berfungsi untuk mengelompokkan submenu-submenu yang berada di bawah menu utama *Display Form*. Di dalam struktur ini, submenu seperti *Request*, *Approval*, *Admin*, dan *Dashboard* dikelompokkan sebagai bagian dari DISPLAY_FORM. Setiap submenu direpresentasikan dengan konstanta yang sebelumnya sudah didefinisikan di bagian atas class.

```
});
Route::group(attributes: ['prefix' => 'dashboard', 'as' => 'dashboard.'], routes: function (): void {
    $class = \App\Http\Controllers\DisplayFormDashboardController::class;
    Route::match(methods: ['GET', 'POST'], uri: 'index', action: [$class, 'index'])->name(name: 'index');
});
```

Gambar 3.56 Struktur Routing untuk Submenu Dashboard

Gambar 3.56 memperlihatkan potongan kode yang berfungsi untuk mendefinisikan *namespace* dan jalur akses dari submenu *Dashboard* pada modul Display Form. Dengan prefix berupa 'dashboard', maka setiap URL yang mengarah ke fitur ini akan diawali dengan /display-form/dashboard. Selanjutnya, method Route::match(['GET', 'POST'], ...) digunakan untuk menangani permintaan HTTP *GET* dan *POST* sekaligus, yang kemudian diarahkan ke method index() di dalam DisplayFormDashboardController. Pendekatan ini berguna ketika satu fungsi controller perlu merespons dua jenis permintaan tersebut, seperti saat menampilkan data awal sekaligus memproses filter dari input pengguna.

```
Breadcrumbs::for(name: 'display-form.dashboard.index', callback: function (BreadcrumbTrail $trail): void {
    $trail->push(title: 'Display Form', url: '#');
    $trail->push(title: 'Dashboard', url: route(name: 'display-form.dashboard.index'));
});
```

Gambar 3.57 Breadcrumb Navigasi untuk Halaman Dashboard

Gambar 3.57 memperlihatkan potongan kode berfungsi untuk membangun struktur navigasi bertingkat (*breadcrumb*) pada halaman Dashboard dalam modul *Display Form*. Kode ini menggunakan metode Breadcrumbs::for untuk mendefinisikan jalur navigasi yang dimulai dari "Display Form" sebagai label konteks utama, kemudian dilanjutkan ke "Dashboard" sebagai halaman yang sedang diakses

```
'admin-vm' => [
    MenuStructure::DISPLAY_FORM,
    MenuStructure::DISPLAY_FORM_APPROVAL,
    MenuStructure::DISPLAY_FORM_ADMIN,
    MenuStructure::DISPLAY_FORM_DASHBOARD
],
```

Gambar 3.58 Akses SubMenu Dashboard

Gambar 3.58 memperlihatkan potongan kode yang digunakan untuk mengatur hak akses submenu pada sistem, khususnya untuk fitur *Dashboard*. Dalam potongan kode tersebut, role admin-vm diberikan akses eksplisit terhadap beberapa submenu dalam menu *Display Form*, salah satunya adalah submenu *Dashboard*. Konfigurasi ini memastikan bahwa submenu *Dashboard* akan muncul di sidebar dan dapat diakses oleh pengguna dengan role tersebut.

```php
public function index(): View
{
    $boothDataRaw = DB::table(table: 'display_form')
        ->select(columns: 'project', DB::raw(value: 'count(*) as total'))
        ->where(column: 'request_title', operator: 'Booth & Racking')
        ->groupBy(groups: 'project')
        ->get()
        ->keyBy(keyBy: 'project');

    $brandingDataRaw = DB::table(table: 'display_form')
        ->select(columns: 'project', DB::raw(value: 'count(*) as total'))
        ->where(column: 'request_title', operator: 'Branding')
        ->groupBy(groups: 'project')
        ->get()
        ->keyBy(keyBy: 'project');

    $signboardDataRaw = DB::table(table: 'display_form')
        ->select(columns: 'project', DB::raw(value: 'count(*) as total'))
        ->where(column: 'request_title', operator: 'Signboard/PNT')
        ->groupBy(groups: 'project')
        ->get()
        ->keyBy(keyBy: 'project');

    // Definisikan kategori project untuk konsistensi
    $projects = ['NEW', 'RENOVATION', 'FIX', 'DISMANTLE', 'EXHIBITION'];

    // Buat array data lengkap dengan nilai 0 jika tidak ada
    $boothData = [];
    foreach ($projects as $p) {
        $boothData[] = $boothDataRaw->has(key: $p) ? $boothDataRaw[$p]->total : 0;
    }

    // Branding hanya ada New, Rennovation, Dismantle
    $brandingProjects = ['NEW', 'RENOVATION', 'DISMANTLE'];
    $brandingData = [];
    foreach ($brandingProjects as $p) {
        $brandingData[] = $brandingDataRaw->has(key: $p) ? $brandingDataRaw[$p]->total : 0;
    }

    // Signboard/PNT ada New, Rennovation, Dismantle, Exhibition
    $signboardProjects = ['NEW', 'RENOVATION', 'FIX', 'DISMANTLE'];
    $signboardData = [];
```

Gambar 3.59 Controller SubMenu Dashboard Pie Chart

Gambar 3.59 memperlihatkan potongan kode fungsi index() pada controller Submenu Dashboard. Fungsi ini bertugas untuk mengambil dan menyiapkan data yang akan ditampilkan pada tampilan utama dashboard. Data yang diolah berasal dari tabel display_form dan dikategorikan berdasarkan jenis permintaan seperti *Booth & Racking*, *Branding*, dan *Signboard/PNT*. Masing-masing data kemudian dihitung jumlahnya berdasarkan nilai *project* seperti *NEW*, *RENOVATION*, *FIX*, *DISMANTLE*, dan *EXHIBITION*. Hasil pengolahan ini nantinya dikirim ke *view* untuk divisualisasikan dalam bentuk grafik *pie chart*.

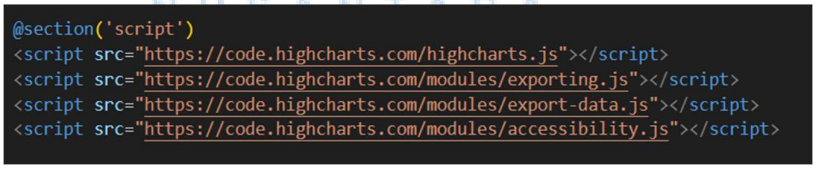
```php
// Ambil data bar chart (display_form_dashboard)
$barDataRaw = DB::table(table: 'display_form_dashboard')
    ->select(columns: 'project', 'results_status', DB::raw(value: 'count(*) as total'))
    ->groupBy(groups: 'project', 'results_status')
    ->get();

// Inisialisasi array untuk OK dan NG dengan 0
$barProjects = $projects;
$okData = array_fill(start_index: 0, count: count(value: $barProjects), value: 0);
$ngData = array_fill(start_index: 0, count: count(value: $barProjects), value: 0);

foreach ($barDataRaw as $row) {
    $index = array_search(needle: $row->project, haystack: $barProjects);
    if ($index !== false) {
        if (strtoupper(string: $row->results_status) == 'OK') {
            $okData[$index] = $row->total;
        } elseif (strtoupper(string: $row->results_status) == 'NG') {
            $ngData[$index] = $row->total;
        }
    }
```

Gambar 3.60 Controller SubMenu Dashboard Bar Chart

Gambar 3.60 memperlihatkan potongan kode pada controller Submenu Dashboard yang digunakan untuk mengolah data *bar chart* dari tabel *display_form_dashboard*. Prosesnya diawali dengan mengambil jumlah data berdasarkan kombinasi nama *project* dan status hasil (OK/NG). Setelah itu, data dimasukkan ke dalam array sesuai urutan projectnya, sehingga bisa langsung dipakai saat ditampilkan dalam grafik batang di halaman dashboard.

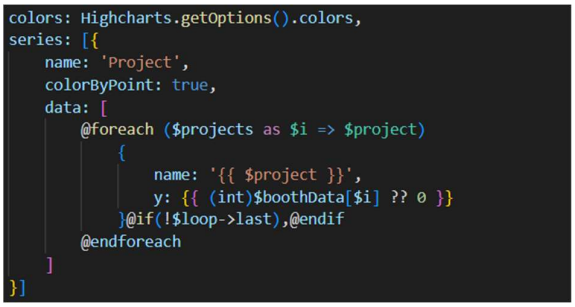
Gambar 3.61 Index Pie Chart

Gambar 3.61 memperlihatkan potongan kode tampilan pada Submenu Dashboard yang berfungsi menampilkan visualisasi data dalam bentuk grafik. Terdapat tiga *pie chart* untuk kategori Booth & Racking, Signboard/PNT, dan Branding, masing-masing ditampilkan dalam elemen tersendiri. Di bawahnya, terdapat satu *bar chart* yang digunakan untuk memvisualisasikan status hasil project (OK dan NG). Seluruh grafik ini dirender menggunakan *Highcharts* dan ditata dalam layout yang responsif.



Gambar 3.62 Script Index

Gambar 3.62 memperlihatkan potongan kode pada bagian @section('script') yang berfungsi untuk memuat *library* eksternal dari Highcharts. *Library* ini dibutuhkan untuk mendukung visualisasi grafik interaktif pada halaman dashboard.



Gambar 3.63 Pengambilan Data untuk Visualisasi Pie Chart

Gambar 3.63 memperlihatkan potongan kode yang digunakan untuk mengisi data grafik pie pada bagian Dashboard. Kode tersebut menggunakan struktur series dari Highcharts, di mana properti data diisi secara dinamis menggunakan perulangan @foreach. Setiap entri dalam array tersebut merepresentasikan sebuah *project* yang terdiri dari dua parameter, yaitu *name* (nama *project*) dan *y* (jumlah total *project* berdasarkan data yang telah diolah sebelumnya).

Gambar 3.64 Pengambilan Data untuk Visualisasi Bar Chart

Gambar 3.64 memperlihatkan potongan kode yang digunakan untuk mengisi data grafik bar pada bagian Dashboard. Kode tersebut menggunakan struktur series dari Highcharts, di mana setiap elemen series merepresentasikan satu kategori hasil, yaitu *OK* dan *NG*. Data yang ditampilkan telah diolah sebelumnya pada sisi controller, kemudian dikonversi dalam bentuk array dan dikirim ke *frontend* menggunakan fungsi @json.

3.2.7 Minggu ke 8-11 (19 Mei – 10 Juni 2025) : Mengurus Dokumentasi Pendaftaran Tempat Magang Serta Pembuatan Laporan Magang

Pada periode ini, aktivitas magang difokuskan pada dua aspek utama, yaitu pengurusan administrasi program Merdeka Belajar Kampus Merdeka (MBKM) dan penyusunan laporan kerja praktik sebagai bentuk pertanggungjawaban akademik. Tahap awal dimulai dengan melakukan pendaftaran secara daring melalui platform MBKM, yang dilanjutkan dengan penyusunan dan pengunggahan dokumen penting berupa *Letter of Acceptance* (LOA) dari pihak perusahaan dan deskripsi pekerjaan (*Job Description*) yang mencerminkan ruang lingkup tugas yang telah disepakati. Kedua dokumen tersebut merupakan persyaratan wajib yang harus diverifikasi oleh dosen penanggung jawab MBKM di tingkat program studi. Pada periode ini, aktivitas magang difokuskan pada dua aspek utama, yaitu pengurusan administrasi program Merdeka Belajar Kampus Merdeka (MBKM) dan penyusunan laporan kerja praktik sebagai bentuk pertanggungjawaban akademik. Tahap awal dimulai dengan melakukan pendaftaran secara daring melalui platform MBKM, yang dilanjutkan dengan penyusunan dan pengunggahan dokumen penting berupa *Letter of Acceptance* (LOA) dari pihak perusahaan dan deskripsi pekerjaan (*Job Description*) yang mencerminkan ruang lingkup tugas yang telah disepakati. Kedua dokumen tersebut merupakan persyaratan wajib yang harus diverifikasi oleh dosen penanggung jawab MBKM di tingkat program studi.

Secara paralel dengan kegiatan administratif, penyusunan laporan magang juga mulai dilaksanakan secara intensif. Proses penyusunan laporan dimulai dengan peninjauan kembali terhadap dokumentasi teknis, catatan *daily task*, serta referensi dari *project* yang telah diselesaikan selama periode magang. Dalam penyusunan laporan, mahasiswa juga menjalin komunikasi aktif dengan dosen pembimbing akademik guna memperoleh masukan yang konstruktif terhadap isi dan struktur penulisan. Penyusunan laporan ini menjadi bagian integral

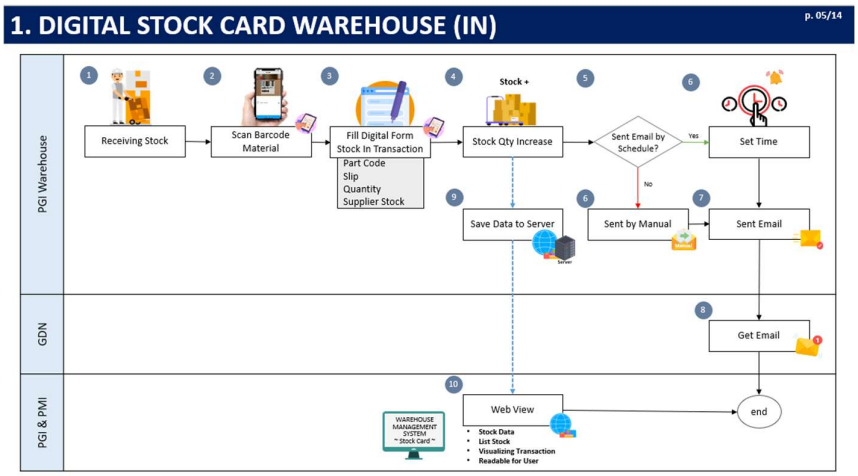
dalam proses refleksi terhadap pengalaman magang, yang tidak hanya menekankan pencapaian praktis tetapi juga pemahaman konseptual yang mendasarinya. Laporan tersebut berfungsi sebagai bukti nyata bahwa mahasiswa mampu mengintegrasikan kompetensi akademik yang diperoleh di bangku perkuliahan dengan tantangan dan dinamika kerja di lingkungan industri. Selain itu, laporan magang juga menjadi sarana untuk menilai sejauh mana mahasiswa dapat mengelola tanggung jawab profesional serta mengomunikasikan hasil kerjanya melalui dokumen formal yang sesuai dengan standar akademik.

3.2.8 Minggu ke 11-13 (10 Juni – 1 Juli 2025) : Pembuatan Flow System Digital Stock Card serta Konfigurasi Awal

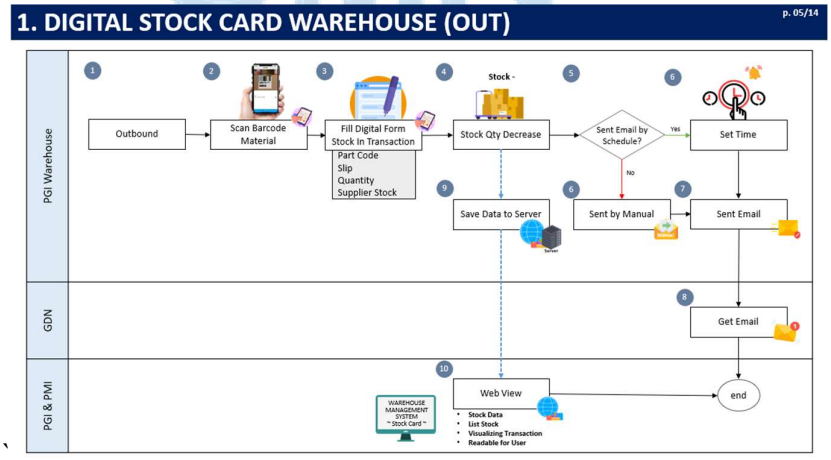
Selain mengembangkan sistem utama PGI Sales Portal, *project* tambahan juga dilaksanakan untuk mendukung peningkatan efisiensi di aspek operasional perusahaan. Salah satu *project* yang dikembangkan secara paralel adalah sistem Digital Stock Card, yaitu aplikasi *mobile* berbasis Flutter yang dirancang untuk mendigitalisasi proses pencatatan dan pelacakan stok barang secara *real-time*. Tujuan utama dari pengembangan sistem ini adalah untuk menggantikan metode pencatatan manual yang selama ini berpotensi menimbulkan keterlambatan dan kesalahan input. Sistem ini dirancang agar dapat digunakan secara langsung oleh tim gudang melalui perangkat *mobile*.

Langkah awal dalam pengembangan sistem dimulai dengan menyusun detail *flow system* sebagai representasi menyeluruh dari alur proses utama yang akan diimplementasikan ke dalam aplikasi. Diagram ini menggambarkan tahapan proses dalam sistem Digital Stock Card ketika stock masuk. Proses ini dimulai dari proses *Receiving Stock* di gudang, dilanjutkan dengan scan barcode material menggunakan aplikasi *mobile*. Setelah barcode terbaca, *user* akan diminta untuk mengisi digital form yang mencakup informasi seperti *Part Code*, nomor slip, jumlah unit barang, serta nama supplier. Formulir digital ini berfungsi sebagai

sumber data utama yang memicu penambahan jumlah stok barang di sistem. Seluruh data kemudian dikirim dan disimpan ke dalam server, serta ditindaklanjuti dengan pengiriman notifikasi email secara otomatis atau manual ke pihak terkait. Data yang tadi tersimpan di dalam server tersebut dapat diakses oleh unit terkait melalui *Web View* yang menyajikan data stok, visualisasi transaksi, serta daftar riwayat penggunaan yang telah terdigitalisasi. Penggambaran diagram tersebut tertuang pada Gambar 3.65.



Gambar 3.65 System Flow Stock In Digital Stock Card



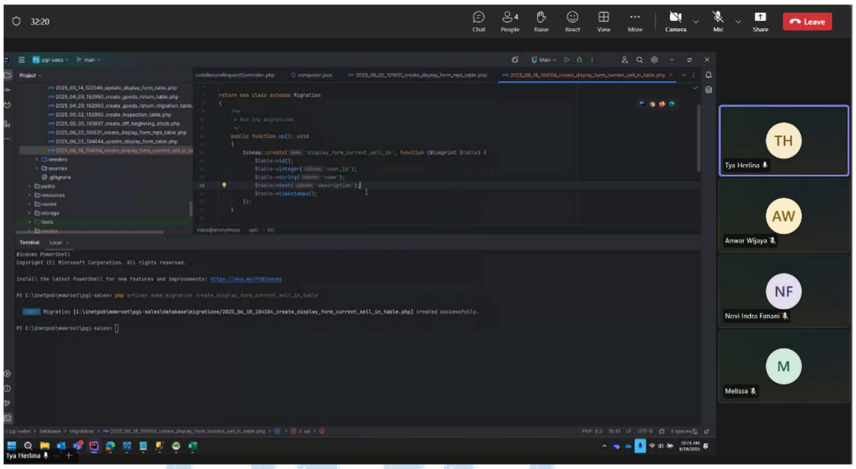
Gambar 3.66 System Flow Stock Out Digital Stock Card

Diagram pada Gambar 3.66 ini menggambarkan tahapan proses dalam sistem Digital Stock Card ketika stock keluar. Proses diawali dengan kegiatan *Outbound*, yaitu pengambilan barang dari gudang yang kemudian dilanjutkan dengan scan barcode material menggunakan perangkat mobile. Setelah barcode terbaca, *user* mengisi digital form transaksi *Stock Out* yang mencakup informasi penting seperti *Part Code*, nomor slip, jumlah barang yang dikeluarkan, serta asal supplier atau tujuan distribusi. Data yang diinput dalam form ini secara otomatis akan memicu penurunan kuantitas stok pada sistem, yang selanjutnya disimpan ke dalam server melalui koneksi API. Sistem kemudian akan memproses pengiriman notifikasi email secara otomatis berdasarkan jadwal tertentu, atau memungkinkan *user* untuk mengirimkannya secara manual. Data yang tadi tersimpan di dalam server tersebut dapat diakses oleh unit terkait melalui *Web View* yang menyajikan data stok, visualisasi transaksi, serta daftar riwayat penggunaan yang telah terdigitalisasi.

Untuk mendukung proses pengembangan, dilakukan instalasi dan konfigurasi lingkungan kerja yang diperlukan di perangkat pribadi dan juga perangkat kantor. Meskipun *project* Flutter yang digunakan merupakan *project existing*, pengaturan ulang environment tetap dilakukan agar pengembangan dapat berjalan lancar dan sesuai dengan standar yang telah digunakan oleh tim sebelumnya. Android Studio dipilih sebagai IDE utama karena mendukung Flutter secara optimal, dan Flutter SDK serta Dart SDK diinstal ulang untuk memastikan kompatibilitas. Proses ini menjadi bagian penting dalam memastikan bahwa sistem dapat dijalankan dengan stabil dan seluruh dependensi telah terpasang dengan benar.

Setelah proses instalasi dan konfigurasi lingkungan kerja selesai dilakukan, kegiatan dilanjutkan dengan pertemuan secara *online* bersama tim pengembang, yaitu Ibu Tya, yang terlibat dalam *project* PGI Sales Portal. Pertemuan ini bertujuan untuk membahas proses *database*

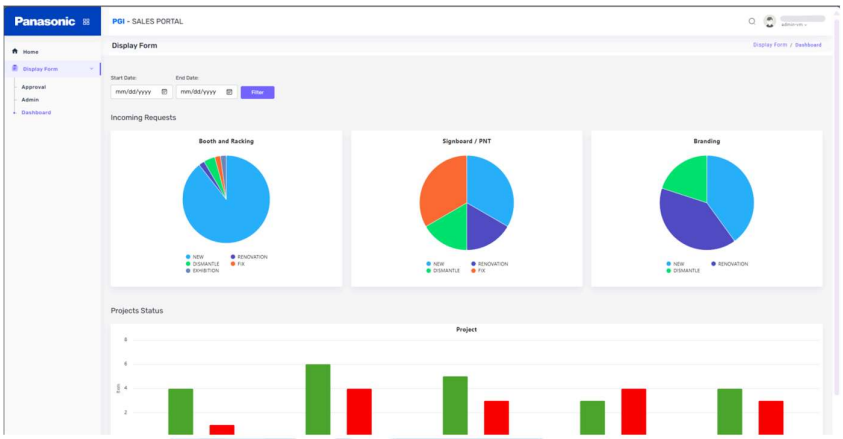
migration serta mendemonstrasikan tampilan website dan menjelaskan berbagai *function* yang telah dikembangkan sebelumnya. Dalam sesi tersebut turut hadir Bapak Anwar, selaku *Group Chief DX* di PT Panasonic Manufacturing Indonesia sekaligus pembimbing lapangan untuk *project PGI Sales Portal*, serta Bapak Indra yang menjabat sebagai *System Development*. Beliau memberikan sejumlah masukan dari sisi sistem internal dan integrasi. Dokumentasi dari pertemuan ini ditunjukkan pada Gambar 3.67.



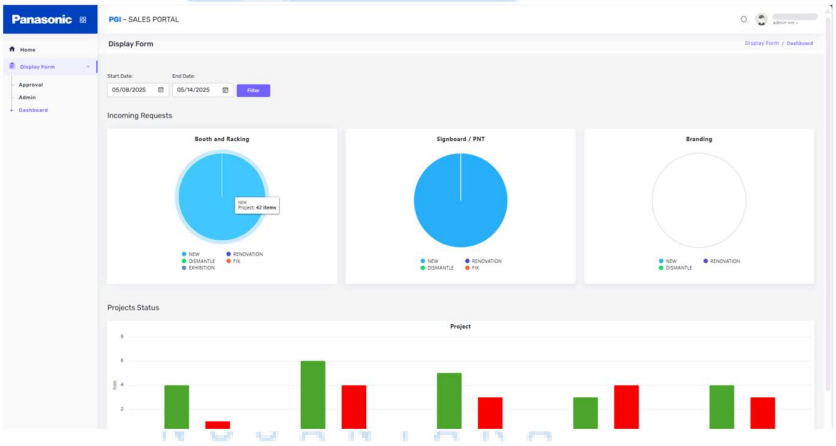
Gambar 3.67 Database Migration Meeting

Dalam sesi pemaparan tersebut, dijelaskan secara mendalam mengenai hasil pengembangan sistem yang mencakup perubahan struktur *database* sebagai bagian dari proses *migration*. Perubahan ini meliputi penyesuaian sejumlah *table* dan *field* yang dirancang untuk mendukung kebutuhan fitur baru serta memperbaiki efisiensi alur data dalam sistem. Penjelasan disampaikan dengan menunjukkan bagian-bagian *database* yang mengalami modifikasi, beserta penggunaannya dalam masing-masing modul sistem, serta fungsionalitas yang dihasilkan dari integrasi tersebut. Selain itu, ditampilkan *demo* website untuk memperlihatkan alur sistem secara langsung. Penjelasan ini bertujuan untuk memberikan gambaran menyeluruh mengenai alur sistem yang telah diimplementasikan, serta memastikan bahwa keseluruhan

pengembangan selaras dengan kebutuhan operasional dan struktur internal perusahaan. Terdapat saran dan rekomendasi terkait penambahan satu fitur kembali untuk tampilan pada halaman dashboard.



Gambar 3.68 Fitur Filter pada Dashboard Sebelum Diterapkan



Gambar 3.69 Fitur Filter pada Dashboard Setelah Diterapkan

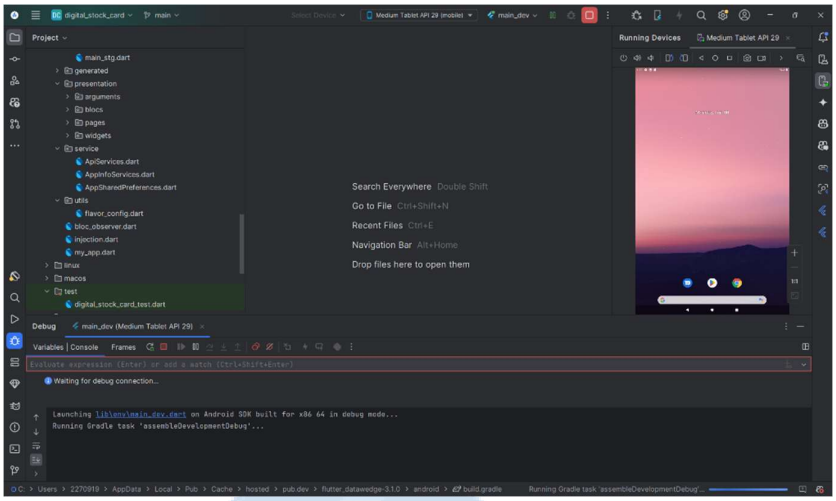
Berdasarkan hasil evaluasi dan diskusi yang dilakukan bersama tim pengembang, terdapat saran untuk menambahkan satu fitur tambahan pada halaman *dashboard*, yaitu fitur *filter* berdasarkan rentang tanggal, bulan, dan tahun. Fitur ini bertujuan untuk memberikan fleksibilitas kepada *user* dalam menampilkan data *project* sesuai periode waktu tertentu, sehingga analisis performa dan status *project* dapat dilakukan secara lebih spesifik dan kontekstual. Sebelum implementasi dilakukan,

tampilan *dashboard* hanya menampilkan seluruh data *project* tanpa batasan waktu, seperti ditunjukkan pada Gambar 3.68. Setelah fitur *filter* ditambahkan, pengguna dapat memilih tanggal awal dan tanggal akhir untuk memfilter data yang ditampilkan, sebagaimana terlihat pada Gambar 3.69.

```php
public function index(Request $request)
{
    $start_date = $request->start_date ?? null;
    $end_date = $request->end_date ?? null;

    $piechart = DB::connection('db_sales_portal')->table('display_form');
    if ($start_date && $end_date) {
        $piechart->whereDate('request_date', '>=', $start_date)
                 ->whereDate('request_date', '<=', $end_date);

    }

    $boothDataRaw = (clone $piechart)
        // ->table('display_form')
        ->select('project', DB::raw('count(*) as total'))
        ->where('request_title', 'Booth and Racking')
        ->groupBy('project')
        ->get()
        ->keyBy('project');
        // dump($boothDataRaw);

    $signboardDataRaw = (clone $piechart)
        ->select('project', DB::raw('count(*) as total'))
        ->where('request_title', 'Signboard / PNT')
        ->groupBy('project')
        ->get()        You, last month • fix code …
        ->keyBy('project');

    $brandingDataRaw = (clone $piechart)
        // ->table('display_form')
        ->select('project', DB::raw('count(*) as total'))
        ->where('request_title', 'Branding')
        ->groupBy('project')
        ->get()
        ->keyBy('project');



    $projects = ['NEW', 'RENOVATION', 'DISMANTLE', 'FIX', 'EXHIBITION'];
```

Gambar 3.70 Code untuk Fitur Filter

Gambar 3.70 memperlihatkan potongan kode yang mengimplementasikan fitur *filter* berdasarkan rentang waktu pada halaman *dashboard*. Pada bagian awal fungsi, sistem menerima input berupa start_date dan end_date dari objek *request*. Kedua nilai tersebut kemudian digunakan untuk menyaring data berdasarkan kolom request_date, dengan menggunakan klausa whereDate. Pemanggilan whereDate('request_date', '>=' ,$start_date) dan whereDate('request_date', '<=', $end_date) memastikan bahwa data yang diambil dari *database* hanya mencakup entri yang berada dalam interval tanggal yang ditentukan oleh pengguna.



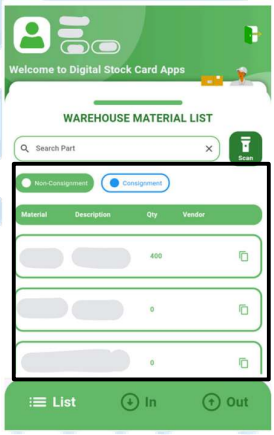
Gambar 3.71 Proses Debugging pada Android Studio

Setelah proses penambahan fitur *filter* pada halaman *dashboard* berhasil diselesaikan, tahap selanjutnya adalah melakukan proses *debugging* secara menyeluruh untuk memastikan bahwa fitur yang baru diintegrasikan dapat berjalan dengan baik tanpa menimbulkan konflik pada bagian sistem lainnya. Proses *debugging* ini dilakukan menggunakan *Android Studio* sebagai *environment* pengembangan utama, dengan memanfaatkan *emulator* untuk menjalankan aplikasi dalam mode *debug*. Fokus utama dari proses ini adalah memastikan bahwa integrasi kode baru tidak menimbulkan *error* yang berkaitan dengan alur data, kompatibilitas versi, maupun struktur direktori *project*. Selain itu, dilakukan juga penyesuaian terhadap konfigurasi *project* agar seluruh dependensi dan modul yang digunakan dapat dikenali dan dijalankan secara lokal tanpa kendala. Gambar 3.71 memperlihatkan antarmuka *Android Studio* saat proses *debugging* sedang berlangsung, menunjukkan status *build* dan koneksi aplikasi dengan *emulator Android.*

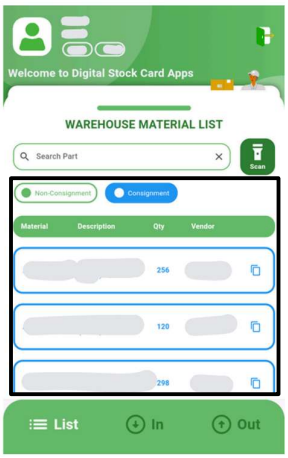
3.2.9 Minggu 14-15 (2 Juli – 9 Juli 2025): Pengembangan Modul Mobile Digital Stock Card

Setelah proses *debugging* selesai dilakukan dan aplikasi *Digital Stock Card* dapat dijalankan dengan stabil pada lingkungan pengembangan

lokal menggunakan *Android Studio*, tahapan berikutnya difokuskan pada penyempurnaan tampilan *user interface* (UI) aplikasi. Perbaikan ini mencakup penyelarasan desain visual serta penanganan masalah teknis yang sebelumnya ditemukan, seperti *icon* yang tidak muncul pada beberapa halaman. Permasalahan tersebut ditelusuri dan diketahui berasal dari kesalahan dalam penulisan *path* atau belum terdaftarnya file *assets* dengan benar dalam konfigurasi *project*. Penyelarasan dilakukan dengan merapikan struktur direktori, memperbarui file *pubspec.yaml*, serta menguji kembali tampilannya di berbagai perangkat. Sebagai bagian dari upaya meningkatkan pengalaman pengguna, dilakukan percobaan penambahan elemen visual baru pada salah satu fitur.



Gambar 3.72 Tampilan Layout Non-Consignment Toggle ketika diklik



Gambar 3.73 Tampilan Layout Consignment Toggle ketika diklik

Hasil penambahan elemen visual dapat dilihat pada Gambar 3.72 dan Gambar 3.73, yang menampilkan perbedaan layout saat *toggle* "Non-Consignment" dan "Consignment" diaktifkan. Perubahan tersebut tidak hanya menyesuaikan elemen warna dan gaya komponen agar lebih responsif, tetapi juga memperjelas status masing-masing kategori material yang ditampilkan dalam daftar. Pada kondisi *Non-Consignment*, layout menggunakan nuansa hijau dengan *border* minimalis dan jumlah stok yang ditampilkan dalam satuan penuh, sedangkan pada *Consignment*, desain diperbarui dengan penggunaan *outline* biru yang lebih kontras, penyesuaian *font weight*, serta penyelarasan posisi ikon untuk memudahkan pembacaan data.

```
BlocBuilder<ConsignmentToggleBloc, ConsignmentToggleState> (
  builder: (context, state) {
    final selected = state.selected;

    return Row(
      children: [
        InkWell(
          onTap: () {
            context.read<ConsignmentToggleBloc>().add(ToggleToNonconsignment());
            context.read<MaterialListBloc>().add(
                FilterConsignmentStatusEvent(selected: ConsignmentStatus.nonConsignment)
            );
          },
          child: Container (
            margin: const EdgeInsets.symmetric(...),  // EdgeInsets.symmetric
            padding: const EdgeInsets.symmetric(...),  // EdgeInsets.symmetric
            decoration: BoxDecoration(
              color: selected == ConsignmentStatus.nonConsignment
                  ? ColorStyle.primaryColor
                  : Colors.white,
              borderRadius:
                const BorderRadius.all(Radius.circular(20)),
              border: Border.all(
                width: 3, color: ColorStyle.primaryColor),  // Border.all
            ),  // BoxDecoration
            child: Row(
              children: [
                Icon(
                  Icons.circle,
                  color: selected == ConsignmentStatus.nonConsignment
                      ? Colors.white
                      : ColorStyle.primaryColor,
                ),  // Icon
                const SizedBox(
                  width: 6,
```

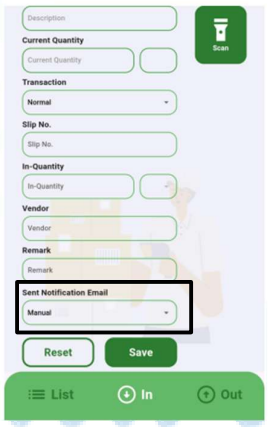
Gambar 3.74 Code untuk tampilan Consignment Toggle

Pada Gambar 3.74 memperlihatkan potongan kode dari Dart untuk proses penerapan tampilan *consignment toggle* pada *home page*.

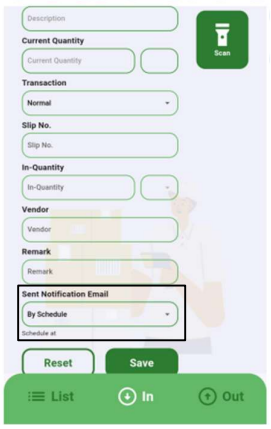
Sejalan dengan perbaikan di sisi tampilan, dilakukan juga proses penggantian API yang digunakan dalam aplikasi, dari API lama yang terhubung dengan sistem *Warehouse Management System* (WMS) ke

API baru yang dikembangkan secara khusus untuk mendukung sistem *Digital Stock Card*. Pergantian ini dilakukan untuk menyederhanakan alur komunikasi antar sistem, mempercepat respons aplikasi, serta memastikan kesesuaian struktur data dengan kebutuhan modul baru yang dikembangkan.

Sebagai pelengkap dari pengembangan fungsionalitas aplikasi, ditambahkan juga fitur *"Sent Notification Email"* pada tampilan aplikasi. Fitur ini memungkinkan pengguna untuk mengirim notifikasi secara otomatis maupun secara manual melalui email setelah melakukan tindakan tertentu dalam sistem. Berikut tampilannya pada Gambar 3.75 dan 3.76.

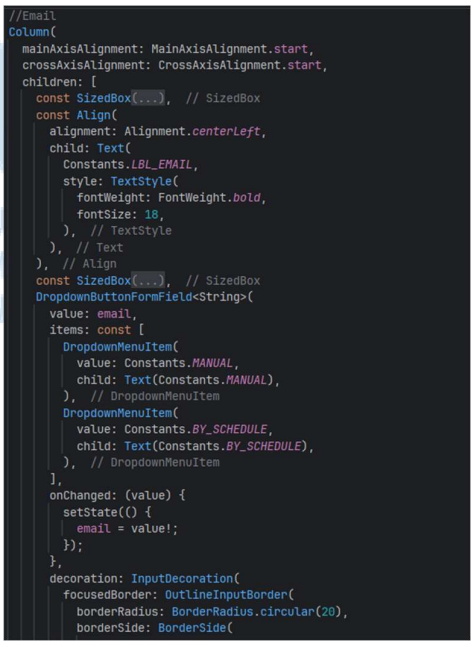


Gambar 3.75 Tampilan Layout Fitur Tambahan Sent Email by Manual



Gambar 3.76 Tampilan Layout Fitur Tambahan Sent Email by Schedule

Fitur *"Sent Notification Email"* dirancang untuk memberikan fleksibilitas dalam proses pengiriman notifikasi kepada pihak ketiga setelah terjadinya aktivitas tertentu di dalam sistem, seperti pencatatan stok masuk (*In-Quantity*) atau keluaran barang (*Out-Quantity*). Fitur ini terbagi menjadi dua mode pengiriman, yaitu *Manual* dan *By Schedule*, yang masing-masing ditampilkan pada Gambar 3.75 dan Gambar 3.76. Pada mode *Manual*, sistem akan langsung mengirimkan email sesaat setelah pengguna menekan tombol *Save*. Sementara itu, pada mode *By Schedule*, pengguna diberikan opsi untuk menjadwalkan waktu pengiriman notifikasi. Ketika opsi *By Schedule* dipilih, akan muncul sebuah *popup* pemilihan waktu yang memungkinkan pengguna untuk menentukan waktu pengiriman notifikasi, dengan batas waktu minimum yang ditentukan adalah lima menit dari waktu saat ini. Setelah pengguna memilih waktu, informasi tersebut akan ditampilkan secara otomatis pada bagian *Schedule at*, sebagai indikator kapan email akan dikirimkan oleh sistem.



```
//Email
Column(
  mainAxisAlignment: MainAxisAlignment.start,
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    const SizedBox(...),  // SizedBox
    const Align(
      alignment: Alignment.centerLeft,
      child: Text(
        Constants.LBL_EMAIL,
        style: TextStyle(
          fontWeight: FontWeight.bold,
          fontSize: 18,
        ),  // TextStyle
      ),  // Text
    ),  // Align
    const SizedBox(...),  // SizedBox
    DropdownButtonFormField<String>(
      value: email,
      items: const [
        DropdownMenuItem(
          value: Constants.MANUAL,
          child: Text(Constants.MANUAL),
        ),  // DropdownMenuItem
        DropdownMenuItem(
          value: Constants.BY_SCHEDULE,
          child: Text(Constants.BY_SCHEDULE),
        ),  // DropdownMenuItem
      ],
      onChanged: (value) {
        setState(() {
          email = value!;
        });
      },
      decoration: InputDecoration(
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(20),
          borderSide: BorderSide(
```

Gambar 3.77 Code untuk tampilan Fitur Sent Email

Pada Gambar 3.77 memperlihatkan potongan kode dari Dart untuk proses penerapan tampilan *fitur Sent Email* pada *Form Stock In* dan *Stock Out page*.

3.3 Kendala yang Ditemukan

Selama menjalani program praktik kerja magang, karyawan magang dihadapkan pada berbagai tantangan dan hambatan yang tidak dapat dihindari. Kesulitan tersebut muncul baik dalam pelaksanaan tugas-tugas harian yang bersifat rutin maupun dalam pengerjaan tugas *project* yang lebih kompleks. Salah satu *project* yang menjadi fokus utama selama masa magang adalah pengembangan sistem berbasis web pada *project* PGI Sales Portal. Berikut ini beberapa kendala atau hambatan yang dihadapi karyawan magang sebagai seorang *internship trainee* di perusahaan PT. Panasonic Gobel Indonesia yang ditempatkan pada PT. Panasonic Manufacturing Indonesia yaitu sebagai berikut:

1. Adaptasi terhadap lingkungan kerja dan sistem internal perusahaan

   Pada tahap awal pelaksanaan kegiatan magang, karyawan magang dihadapkan pada proses adaptasi terhadap budaya kerja, ritme aktivitas harian, serta prosedur komunikasi yang berlaku di lingkungan perusahaan. Ketidakterbiasaan dalam mengikuti alur kerja yang sistematis dan formal menjadi tantangan tersendiri, terutama bagi individu yang sebelumnya belum memiliki pengalaman di dunia industri manufaktur. Oleh karena itu, proses penyesuaian terhadap lingkungan dan etos kerja perusahaan perlu dilakukan secara bertahap agar dapat mendukung kelancaran pelaksanaan tugas yang diberikan.

2. Pemahaman terhadap alur teknis pengembangan sistem

   Dalam keterlibatan pada pengembangan *project* PGI Sales Portal, karyawan magang menghadapi tantangan dalam memahami alur teknis sistem yang digunakan di lingkungan perusahaan. Kendala utama ditemukan pada proses adaptasi terhadap struktur kode yang kompleks, dan pemahaman terhadap relasi antar-komponen.

Kondisi ini menuntut adanya pemahaman ulang terhadap konsep-konsep teknis yang relevan, serta penyesuaian terhadap standar pengembangan sistem yang telah ditetapkan. Proses pembelajaran dilakukan secara mandiri dan juga melalui bimbingan dari pihak internal perusahaan.

3. Penyesuaian terhadap penggunaan *framework* baru dalam proses pengembangan

   Salah satu tantangan teknis yang dihadapi selama kegiatan magang adalah keharusan untuk mempelajari dan menerapkan *framework* baru. *Framework* Laravel digunakan sebagai dasar pengembangan sistem internal perusahaan. Pemanfaatan Laravel memerlukan pemahaman mendalam mengenai konsep Model-View-Controller (MVC), struktur file *project*, *routing*, serta integrasi dengan sistem autentikasi dan database. Proses penyesuaian terhadap *framework* ini tidak hanya menuntut kemampuan teknis, tetapi juga keterampilan analitis dan ketelitian dalam membaca dokumentasi maupun kode yang telah dikembangkan oleh tim sebelumnya.

3.4 Solusi atas Kendala yang Ditemukan

Berdasarkan hambatan dan kesulitan yang dihadapi selama program praktik kerja magang pada perusahaan PT. Panasonic Manufacturing Indonesia, terdapat beberapa solusi yang diterapkan untuk mengatasi kendala yang dialami oleh karyawan magang sebagai seorang Internship Trainee, yaitu sebagai berikut:

1. Solusi atas kendala adaptasi terhadap lingkungan kerja dan sistem internal perusahaan

   Untuk mengatasi tantangan dalam beradaptasi dengan lingkungan kerja, dilakukan pendekatan secara bertahap melalui pengamatan langsung terhadap pola kerja karyawan tetap serta pembiasaan terhadap ritme aktivitas harian yang berlaku. Selain itu, arahan dari pembimbing lapangan dan rekan kerja turut membantu dalam proses

pemahaman terhadap struktur organisasi serta budaya komunikasi yang digunakan.

2. Solusi atas kendala pemahaman terhadap alur teknis pengembangan sistem

   Dalam mengatasi kendala teknis terkait pemahaman alur sistem, dilakukan eksplorasi terhadap dokumentasi *project* serta pengamatan terhadap implementasi kode yang telah tersedia. Proses ini didukung oleh diskusi dengan pembimbing tempat kerja magang atau dengan tim *developer* untuk memperoleh penjelasan yang lebih rinci mengenai struktur sistem dan logika bisnis yang diterapkan. Melalui pendekatan tersebut, pemahaman terhadap sistem secara keseluruhan dapat diperoleh secara bertahap, sehingga karyawan magang mampu mengikuti alur kerja teknis dengan lebih baik.

3. Solusi atas kendala penyesuaian terhadap penggunaan *framework* baru dalam proses pengembangan

   Untuk menghadapi tantangan dalam penggunaan *framework* Laravel, dilakukan proses pembelajaran mandiri melalui studi dokumentasi resmi Laravel, video tutorial, serta referensi pengembangan sistem sebelumnya. Selain itu, praktik langsung melalui studi kasus atau fitur sederhana turut diterapkan untuk memperkuat pemahaman terhadap konsep *Model-View-Controller* (MVC), *routing*, dan integrasi *database*.