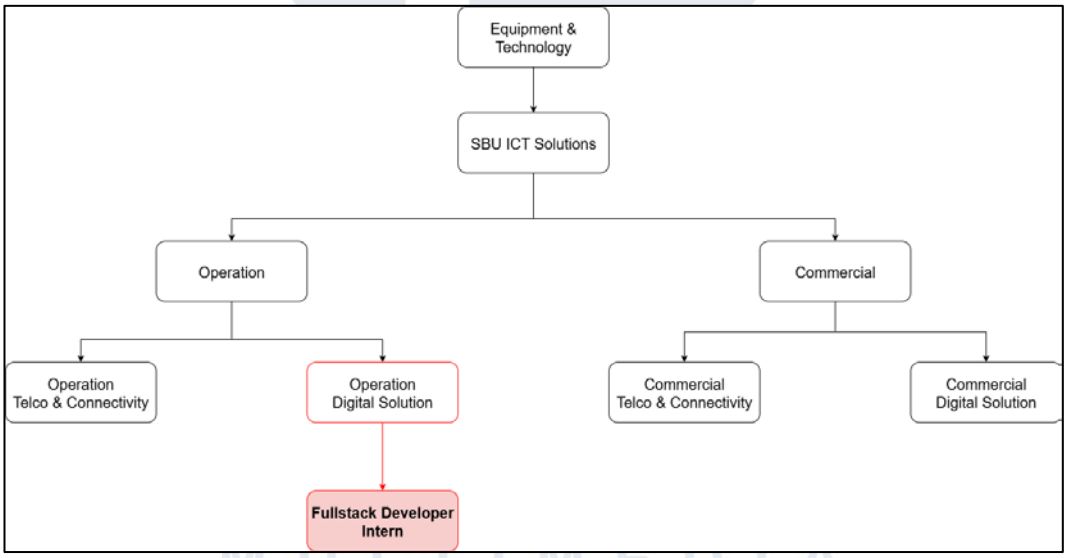
# BAB III

# PELAKSANAAN KERJA

## 3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kegiatan magang di PT IAS Support Indonesia, posisi dan tanggung jawab yang diberikan, telah disesuaikan dengan bidang keahlian serta kebutuhan perusahaan. Dalam melaksanakan kegiatan magang, sistem koordinasi kerja yang terstruktur bersama pembimbing lapangan dan tim terkait juga dilakukan, sehingga seluruh kegiatan dapat berjalan dengan efektif dan sesuai dengan arahan perusahaan. Pemahaman yang dalam mengenai posisi dalam organisasi serta alur koordinasi yang diterapkan penting untuk memastikan kelancaran dan efisiensi setiap tugas dan proyek yang diberikan.

### 3.1.1 Kedudukan




Gambar 3.1 Kedudukan Magang

Selama pelaksanaan kegiatan magang di PT IAS Support Indonesia, magang dilaksanakan pada unit *Strategic Business Unit* (SBU) ICT Solution, tepatnya di bagian *Operation Digital Solution* sebagai *Fullstack Developer Intern*. Unit tersebut berada di bawah koordinasi *Division Head Operation Digital Solution*, yang pada masa magang dijabat oleh Bapak Harival Tivani

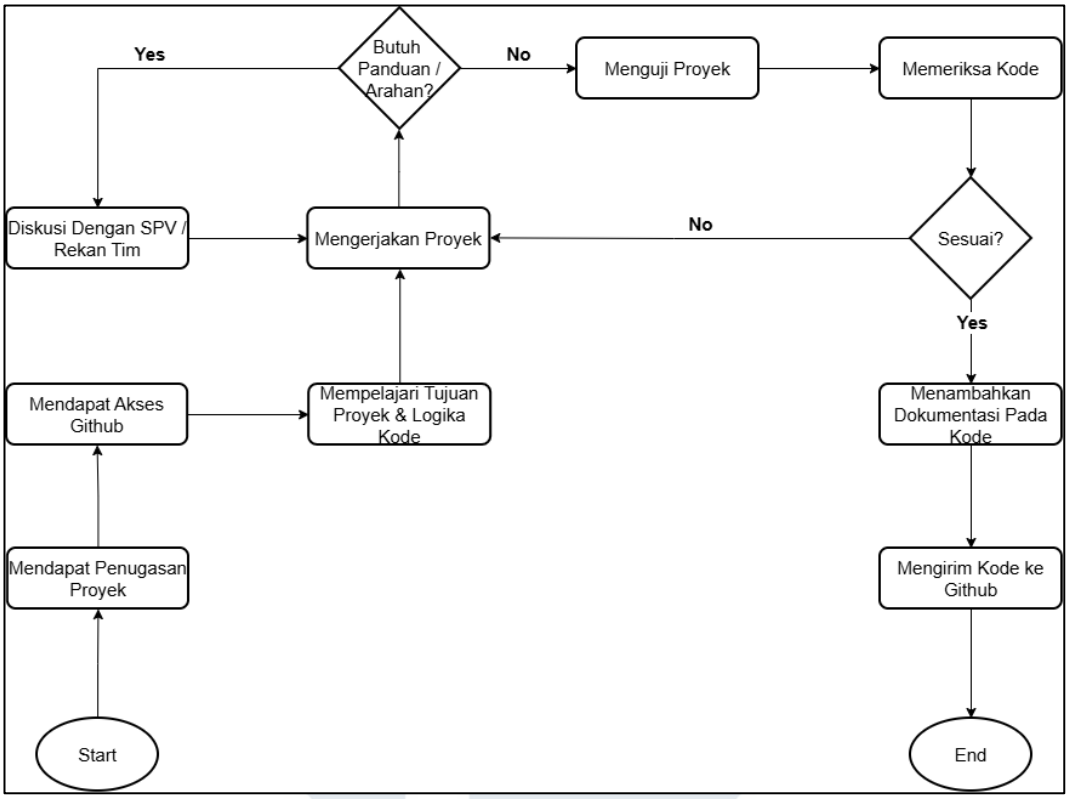
selaku pembimbing lapangan. Sebagai Fullstack Developer Intern, beberapa tanggung jawab utama meliputi, antara lain:

1. Mengembangkan dan menguji perangkat lunak sesuai dengan kebutuhan proyek.
2. Membantu dalam proses pengkodean, debugging, dan pemeliharaan aplikasi.
3. Berkolaborasi dengan tim pengembang dalam merancang dan mengimplementasikan solusi teknis.
4. Mempelajari serta menerapkan teknologi terbaru yang relevan dengan pengembangan perangkat lunak.
5. Menyusun dokumentasi kode dan laporan hasil kerja secara berkala.
6. Melaksanakan tugas lain yang relevan sesuai dengan arahan pembimbing magang.

### 3.1.2 Koordinasi

Kegiatan magang di PT IAS Support Indonesia dilaksanakan langsung dibawah bimbingan Division Head Operation Digital Solution. Koordinasi dilakukan secara fleksibel sesuai dengan kebutuhan dan progres proyek yang sedang dikerjakan tanpa memiliki jadwal yang pasti. Tujuan utama dari koordinasi ini adalah untuk memastikan setiap aktivitas pengembangan sistem berjalan sesuai dengan arahan dan standar yang diterapkan di lingkungan perusahaan.

Komunikasi dilakukan lebih sering secara langsung, namun whatsapp juga digunakan sebagai media komunikasi harian yang digunakan untuk diskusi teknis, penyampaian arahan singkat, serta koordinasi terkait pengerjaan proyek dan penyelesaian kendala teknis saat tidak bisa disampaikan secara langsung. Sementara itu, pengelolaan proyek dilakukan melalui platform GitHub yang berfungsi sebagai media kolaborasi kode, dokumentasi, serta pelaporan hasil pekerjaan yang telah diselesaikan.




Gambar 3.2 Bagan Alur Koordinasi

Pada Gambar 3.2, alur koordinasi kerja diawali ketika penugasan proyek diterima dari pembimbing lapangan. Setelah mendapatkan akses repositori GitHub, tujuan proyek dan logika kode dipelajari terlebih dahulu sebagai pendalaman pemahaman proyek. Setelah tahap tersebut, proyek sudah mulai bisa dikerjakan. Jika diperlukan panduan tambahan, diskusi dilakukan dengan supervisor atau rekan tim untuk mendapatkan arahan teknis yang lebih jelas dan prioritas tugas yang harus dikerjakan. Kemudian pengujian dan pemeriksaan kode dilakukan, serta hasil pekerjaan dipastikan sesuai dengan standar yang telah ditetapkan.

Setelah proses sebelumnya selesai, kode akan dievaluasi terlebih dahulu, jika kode telah dinyatakan sesuai oleh supervisor, dokumentasi kode ditambahkan didalam kode, kemudian hasil kerja diunggah ke repositori GitHub untuk dilakukan *deployment*. Jika kode belum sesuai, arahan akan diberikan terkait bagian yang tidak sesuai dan konsultasi dilakukan apabila menghadapi kendala teknis selama pengembangan sistem.

## 3.2 Tugas yang Dilakukan

Pada pelaksanaan magang, Berbagai proyek berlangsung secara paralel dan bertahap telah dilakukan sesuai prioritas perusahaan. Seperti yang tertera pada Tabel 3.1, pada minggu-minggu awal, fokus pekerjaan berada pada pengembangan Tenant Management System, dimulai dari pembuatan *reminder invoice*, penyesuaian data master, hingga penyempurnaan tabel, *form*, serta penambahan fitur aktivasi layanan. Memasuki minggu keempat, pengembangkan modul pada E-Office dilakukan dengan menyesuaiankan *field* isi surat dan optimalisasi modul inbox. Pekerjaan inti dimulai ketika proyek prioritas Employee Lounge System diberikan, di mana seluruh kegiatan pengembangan sistem dilakukan disini, mulai dari analisis kebutuhan, perancangan alur sistem dan basis data, serta mengembangkan fitur inti hingga tahap uji coba dan *deployment*.

Tabel 3.1 Detail Pekerjaan yang Dilakukan

| No. | Minggu | Proyek | Keterangan |
|---|---|---|---|
| 1. | 1 | Tenant Management System | Membuat reminder invoice |
| 2. | 1 | Tenant Management System | Presentasi progres & penambahan fitur |
| 3. | 1 | Tenant Management System | Menambahkan data master |
| 4. | 1-2 | Tenant Management System | Menyesuaikan tabel dan *form* |
| 5. | 2-3 | Tenant Management System | Membuat fitur aktivasi layanan |
| 6. | 4 | E-Office | Mengembangkan *field* isi surat |
| 7. | 4-5 | E-Office | Menyesuaikan modul inbox |
| 8. | 5 | Employee Lounge System | Menganalisis kebutuhan sistem |
| 9. | 5 | Employee Lounge System | Mendesain alur sistem & database |

| | | | |
|---|---|---|---|
| 10. | 6-8 | Employee Lounge System | Mengkode & membangun sistem |
| 11. | 8 | Employee Lounge System | *Deploy*, *test*, & *review* sistem |
| 12. | 9-10 | Tenant Management System | Membuat fitur *ticket* |
| 13. | 11 | Smart Utility | Mengembangkan UI client |
| 14. | 11-12 | Smart Utility | Mengembangkan UI admin & Menambahkan data master |
| 15. | 12 | Bus Management System | Menambah logika validasi tiket bus |
| 16. | 13 | Bus Management System | Menyesuaikan tampilan & tema warna |
| 17. | 13 | Employee Lounge System | Memperbaiki *bug booking* & menyesuaikan syarat *booking* |
| 18. | 14 | Bus Management System | Membuat tabel *historical booking* & *booking detail* |
| 19. | 14 | Bus Management System | Memperbarui dashboard |
| 20. | 15 | Bus Management System | Mengintegrasi API Damri Shuttle |
| 21. | 15 | Bus Management System | Memperbaiki fitur OTA |
| 22. | 15-17 | Bus Management System | Membuat & memperbaiki API |
| 23. | 16 | Employee Lounge System | Menyesuaikan *export template* excel *shift* dan *import*-nya |
| 24. | 18 | Autogate Report System | Membuat sistem |

Setelah sistem tersebut berjalan, Tenant Management System kembali dilanjutkan dengan membangun fitur *ticketing*. Pada minggu berikutnya, pengembangan Smart Utility dilakukan dengan mencakup pembuatan antarmuka untuk sisi klien maupun admin serta penambahan data master. Kontribusi berlanjut pada Bus Management System, mulai dari pembuatan fitur validasi tiket, penyesuaian tampilan serta tema, perbaikan *bug* dan *dashboard*, hingga integrasi API Damri serta penyempurnaan modul OTA. Pembuatan tabel historical untuk

proses *booking* juga telah dilakukan. Selain itu, beberapa perbaikan dilakukan kembali pada Employee Lounge System, khususnya pada proses *import* dan *template shift*. Pada tahap akhir magang, Autogate Report System diberikan sebagai proyek tambahan. Seluruh rangkaian tugas ini menunjukkan bahwa pekerjaan tidak hanya sekadar *coding*, tetapi mencakup analisis kebutuhan, perancangan sistem, integrasi API, *debugging*, hingga optimalisasi modul proyek.

## 3.3 Uraian Pelaksanaan Kerja

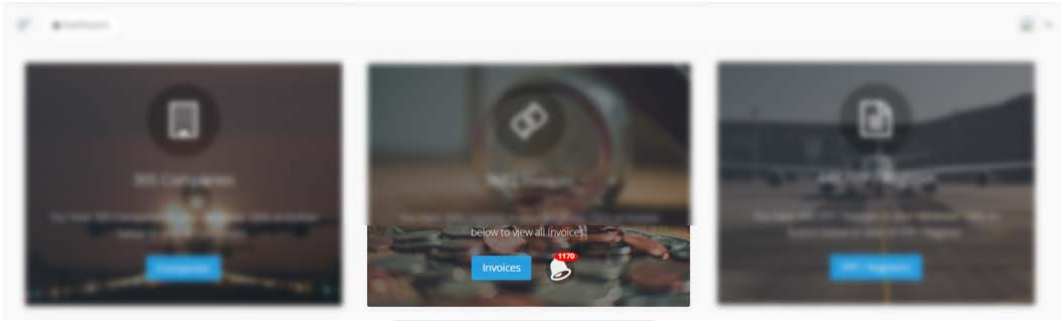
Penjelasan disajikan mengenai rangkaian kegiatan yang dilaksanakan selama periode kerja magang. Setiap aktivitas yang dilakukan dijelaskan secara sistematis untuk memberikan gambaran menyeluruh mengenai proses pengerjaan, peran yang dijalankan, serta kontribusi yang diberikan terhadap proyek yang dikerjakan. Bukti visual juga disertakan untuk memperkuat deskripsi pelaksanaan kerja.

### 3.3.1 Proses Pelaksanaan

Proses pelaksanaan tugas secara lebih rinci dijelaskan pada bagian ini. Setiap proyek atau fitur yang dikerjakan dijelaskan berdasarkan alur pengerjaannya, mulai dari pemahaman kebutuhan, pelaksanaan teknis, hingga hasil yang dicapai. Penyajian dibuat terstruktur agar dapat menggambarkan proses kerja secara jelas serta menunjukkan kemampuan teknis yang diterapkan selama kegiatan magang.

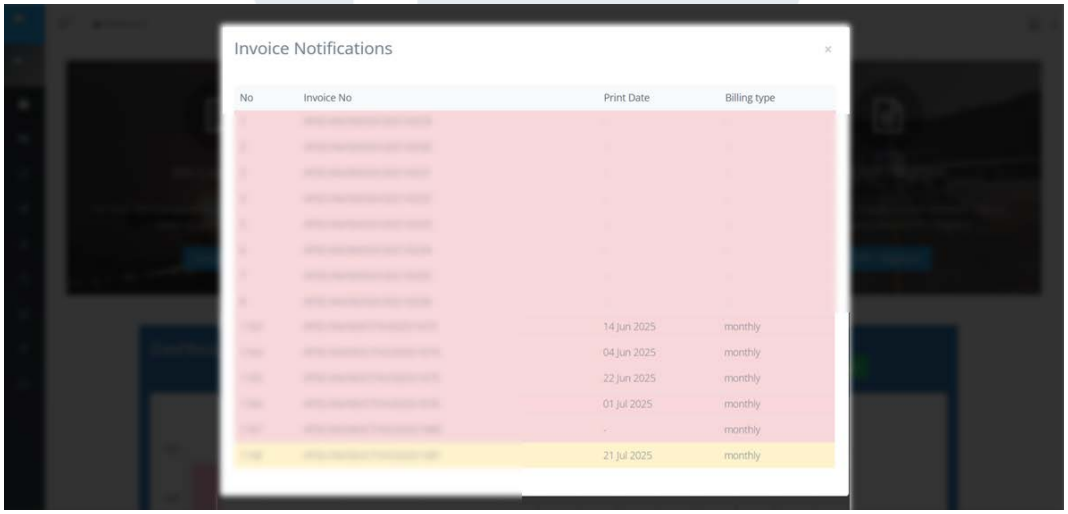
#### 3.3.1.1 Tenant Management System - Membuat *reminder invoice*

Pengerjaan dimulai dengan melanjutkan kebutuhan yang sebelumnya belum terselesaikan pada periode magang sebelumnya. Projek ini menggunakan laravel versi 8 dan php versi 8 dengan *package* admin panel Voyager. Pada modul Tenant Management System, salah satu tugas utama adalah menyelesaikan fitur *reminder invoice*.




Gambar 3.3 Tampilan *reminder invoice*

Seperti yang ditunjukkan pada Gambar 3.3, ikon lonceng di *dashboard* menunjukkan jumlah *invoice tenant* yang belum dibayarkan. Ikon tersebut memiliki angka yang menandakan jumlah invoice yang belum dibayar. Saat ikon lonceng diklik, sistem menampilkan sebuah modal seperti pada Gambar 3.4.




Gambar 3.4 Tampilan *list reminder invoice*

Didalam modal ini, daftar *invoice* yang masih menunggu pembayaran akan ditampilkan disini. Setiap *entry* menampilkan nomor *invoice*, *print date*, dan *billing type*. Nomor *invoice* jadi penanda unik setiap transaksi. *Print date* menunjukkan kapan *invoice* diterbitkan ke *tenant*. *Billing type* memberi tahu apakah tagihan itu bersifat bulanan atau tahunan.

```php
$dueDate = \Carbon\Carbon::parse($item->due_date);
$now = \Carbon\Carbon::now();
$daysLeft = $dueDate->diffInDays($now, false); // negatif jika masih di masa depan

$bgColor = '';
if ($daysLeft < 0 && abs($daysLeft) <= 14) {
    $bgColor = '#fff3cd'; // kuning
    // $text = 'Akan jatuh tempo';
} elseif ($daysLeft >= 0) {
    $bgColor = '#f8d7da'; // merah
    // $text = 'Sudah jatuh tempo';
}
```

Gambar 3.5 Kode klasifikasi tampilan

Pada Gambar 3.5, ada potongan kode yang mengatur pengelompokan status *invoice* berdasarkan tenggat pembayarannya. Invoice yang jatuh tempo kurang dari dua minggu ditandai warna kuning. Kalau sudah lewat jatuh tempo, warnanya merah. Invoice yang statusnya sudah dibayar akan otomatis hilang dari daftar ini. Sistem klasifikasi warna seperti ini memudahkan pengguna untuk memprioritaskan invoice mana yang harus segera ditangani.

#### 3.3.1.2 Tenant Management System - Presentasi progres & penambahan fitur

Setelah seluruh kebutuhan utama diselesaikan seluruhnya, pertemuan diadakan untuk mempresentasikan progres pengerjaan Tenant Management System. Pada sesi ini, dijelaskan fitur-fitur yang telah dikembangkan beserta demonstrasi alur pendataan *tenant* dari awal sampai akhir. Presentasi ini menjadi evaluasi awal sebelum sistem melanjutkan tahap pengembangan berikutnya.

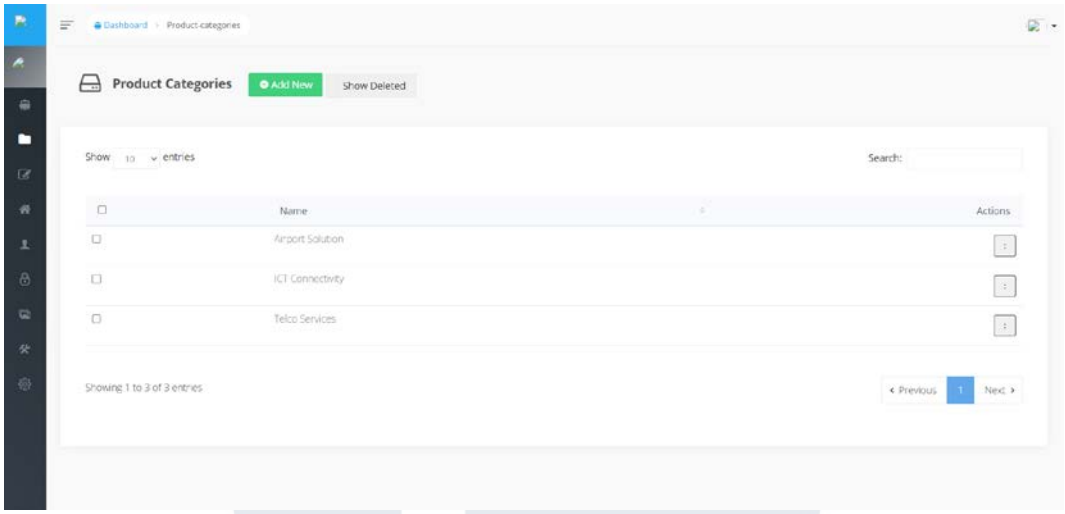
Gambar 3.6 Presentasi progres

Gambar 3.6 menunjukkan situasi rapat saat pemaparan progres berlangsung. Hasil diskusi menunjukkan fitur yang sudah dibuat sudah sesuai dengan kebutuhan awal. Meski begitu, beberapa permintaan pengembangan tambahan diajukan oleh peserta rapat. Penambahan yang dibutuhkan yaitu kebutuhan data baru untuk pengelompokan produk, penyesuaian pada modul aktivasi layanan, serta penambahan fitur tiket pengaduan.

#### 3.3.1.3 Tenant Management System - Menambahkan data master

Penambahan data master penting untuk mendukung pengelompokan produk berdasarkan kategori pada sistem Tenant Management. Di sistem ini, ada tiga kategori utama: *Airport Solution*, *ICT Connectivity*, dan *Telco Services*. Ketiga kategori ini

berfungsi sebagai pondasi pengelompokan produk, sehingga data menjadi lebih terstruktur dan lebih mudah dicari.



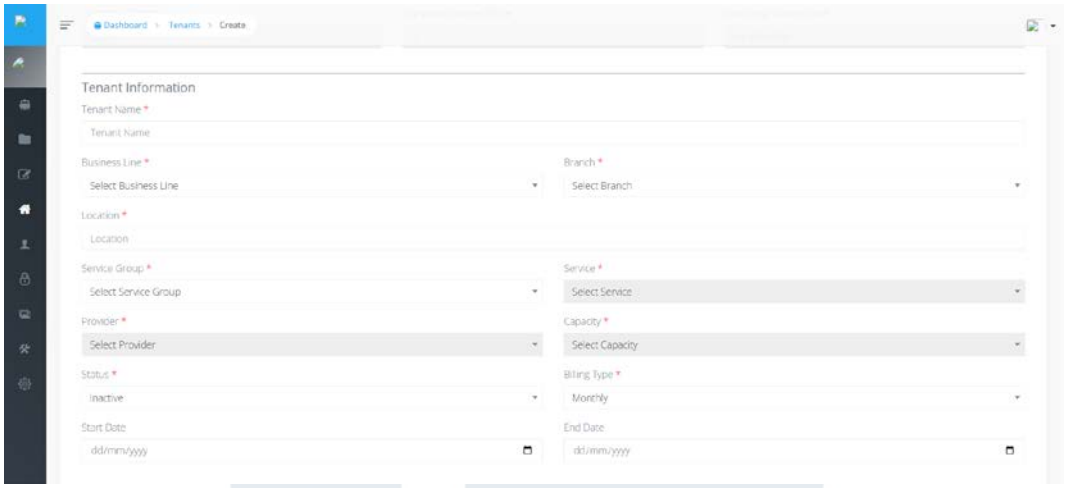

Gambar 3.7 Data master kategori produk

Gambar 3.7 memperlihatkan modul data master yang telah dibuat beserta fungsi CRUD. Modul ini memudahkan proses menambah, mengedit, dan menghapus data kategori sehingga membuat pengelolaan menjadi jauh lebih efisien. Implementasi data master juga menjaga konsistensi informasi di semua modul yang berkaitan dengan pendataan produk.

#### 3.3.1.4 Tenant Management System - Menyesuaikan tabel dan *form*

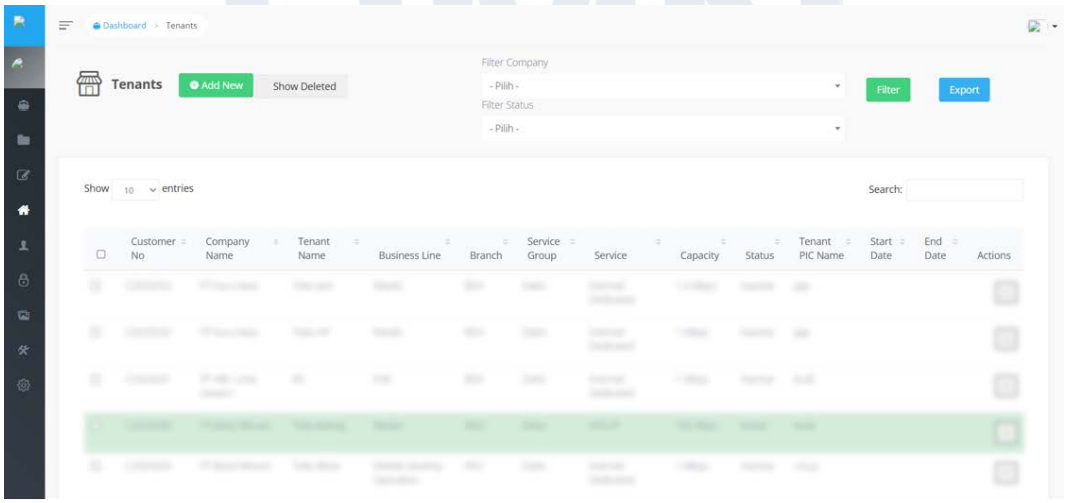
Setelah menambah data master kategori produk, penyesuaian dilakukan ke beberapa *form* supaya struktur data tetap konsisten. *Form* yang sebelumnya hanya memuat daftar layanan, sekarang mempunyai *field service group* dan *field* ini langsung terhubung ke data kategori produk. Alur input setelah perubahan ini menjadi lebih

terstruktur, sehingga pengguna pun lebih mudah memilih layanan yang sesuai dengan kelompoknya.




Gambar 3.8 Tampilan form yang disesuaikan

Gambar 3.8 menunjukkan bagaimana layanan yang muncul otomatis difilter berdasarkan kelompok layanan yang telah dipilih. Selain itu, simbol (*) ditambahkan untuk menunjukkan bahwa *field* tersebut harus diisi. Penyesuaian ini ditujukan untuk mengurangi kekeliruan pengguna saat mengisi *form* karena petunjuknya lebih jelas.

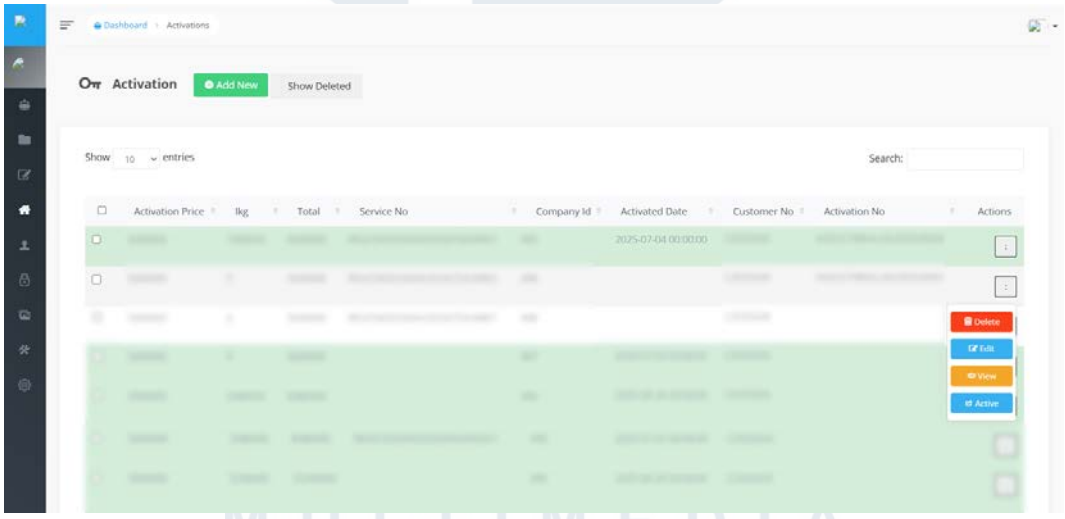



Gambar 3.9 Tampilan tabel yang disesuaikan

Selain itu, penyesuaian dilakukan pada tampilan tabel, seperti yang ditunjukkan pada Gambar 3.9. Kolom *service group* juga ditambahkan untuk memperlengkapi data yang disimpan. Perubahan yang sama juga dilakukan pada tabel lain yang berkaitan dengan kategori produk. Ini menjamin bahwa struktur data konsisten di seluruh modul.

#### 3.3.1.5 Tenant Management System - Membuat fitur aktivasi layanan

Fitur aktivasi layanan dibuat untuk meresmikan bahwa *tenant* sudah siap menggunakan layanan tersebut. Karena fitur ini mencatat status setiap layanan yang sudah diaktifkan, proses pemantauan layanan yang aktif menjadi lebih sistematis. Selain itu, fitur ini membantu admin untuk memastikan bahwa setiap layanan hanya digunakan setelah melewati proses verifikasi yang sesuai.




Gambar 3.10 Tombol-tombol aksi di menu aktivasi layanan

Menu aktivasi, seperti yang ditunjukkan pada Gambar 3.10, memiliki beberapa tombol aksi yang digunakan untuk mengontrol proses aktivasi. Tombol ‘View’ menampilkan data layanan dalam bentuk teks, sedangkan tombol ‘Active’ mengubah status layanan menjadi aktif. Hanya super admin yang dapat menggunakan tombol

‘Edit’ dan ‘Delete’. Ini dilakukan untuk membatasi akses agar perubahan data tetap diawasi.

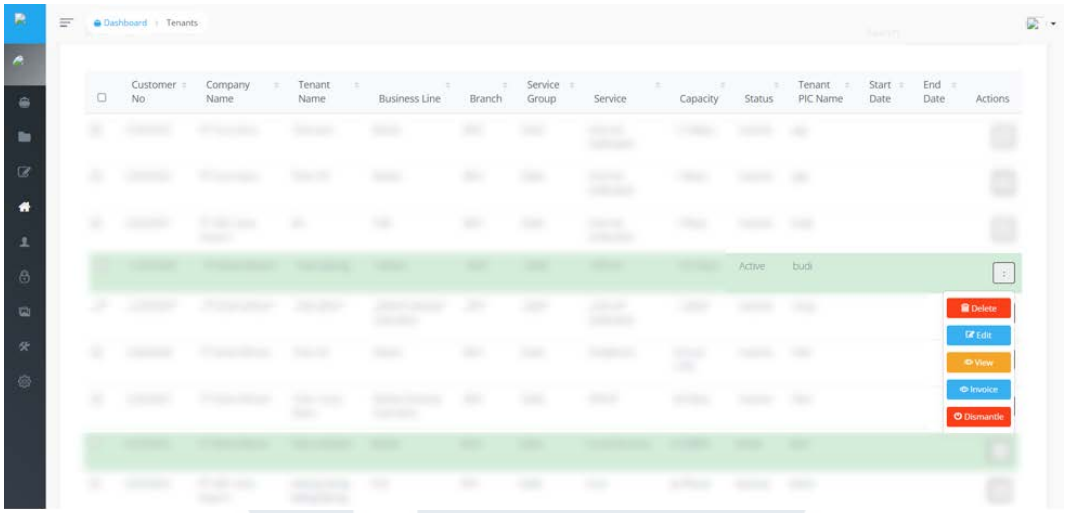
Gambar 3.11 Tampilan format surat berita acara

Format surat berita untuk acara aktivasi layanan ditunjukkan pada Gambar 3.11. Informasi seperti tanggal aktivasi, nama penyedia layanan PT IASS, identitas *tenant*, dan detail layanan yang akan diaktifkan akan menjadi isi dalam surat ini. Setelah itu, dokumen dicetak dan dikirimkan kepada tenant sebagai bukti bahwa layanan telah dinyatakan siap digunakan.


Gambar 3.12 Modal aktivasi layanan

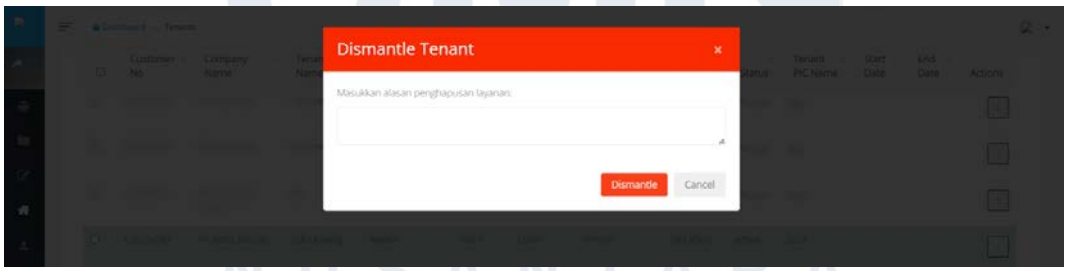
Proses aktivasi dapat dilakukan melalui sistem setelah surat berita acara ditandatangani oleh pihak penerima layanan. Untuk mengunggah *file* PDF surat tersebut, tombol ‘Active’ akan memunculkan modal, seperti yang ditunjukkan pada Gambar 3.12.

Jika dokumen yang benar telah diunggah, layanan dapat diaktifkan dan baris data yang terkait akan berwarna hijau sebagai penanda status aktif.




Gambar 3.13 Tombol dismatle pada menu *tenant*

Selain aktivasi, menu *tenant* memiliki fitur untuk menonaktifkan layanan. Halaman ini menampilkan daftar *tenant* bersama dengan layanan yang mereka gunakan, seperti yang ditunjukkan pada Gambar 3.13. Tombol ‘Dismantle’ tidak akan muncul untuk layanan yang belum aktif, tetapi akan muncul untuk layanan yang sudah aktif, memungkinkan penonaktifan.



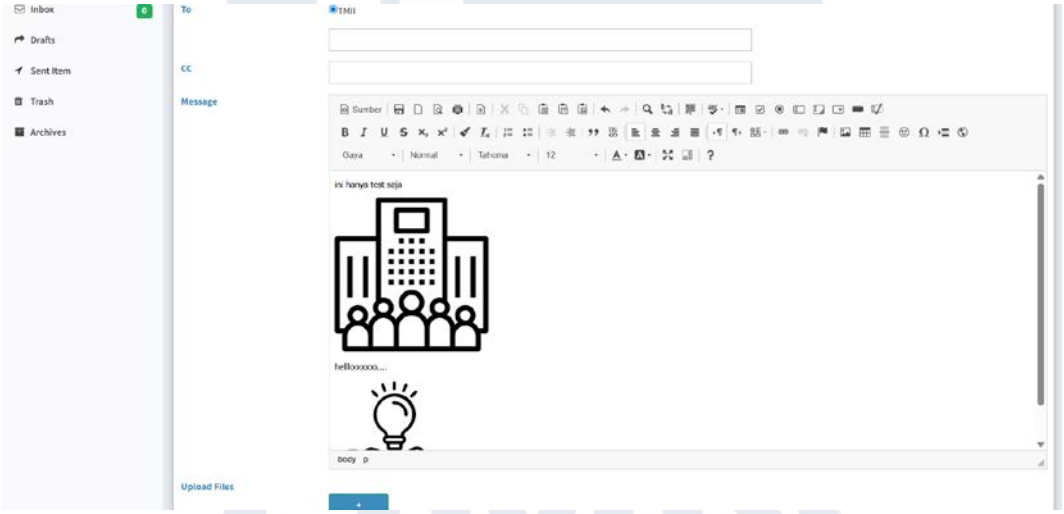

Gambar 3.14 Modal *dismantle* layanan

Ketika tombol ‘Dismantle’ diklik, modal muncul, seperti yang ditunjukkan pada Gambar 3.14. Pengguna harus menyatakan alasan penonaktifan, seperti kontrak yang telah berakhir, keputusan *tenant* untuk menghentikan layanan, atau perubahan pada paket

layanan. Status layanan akan berubah menjadi nonaktif setelah proses ini selesai dan alasan disimpan.

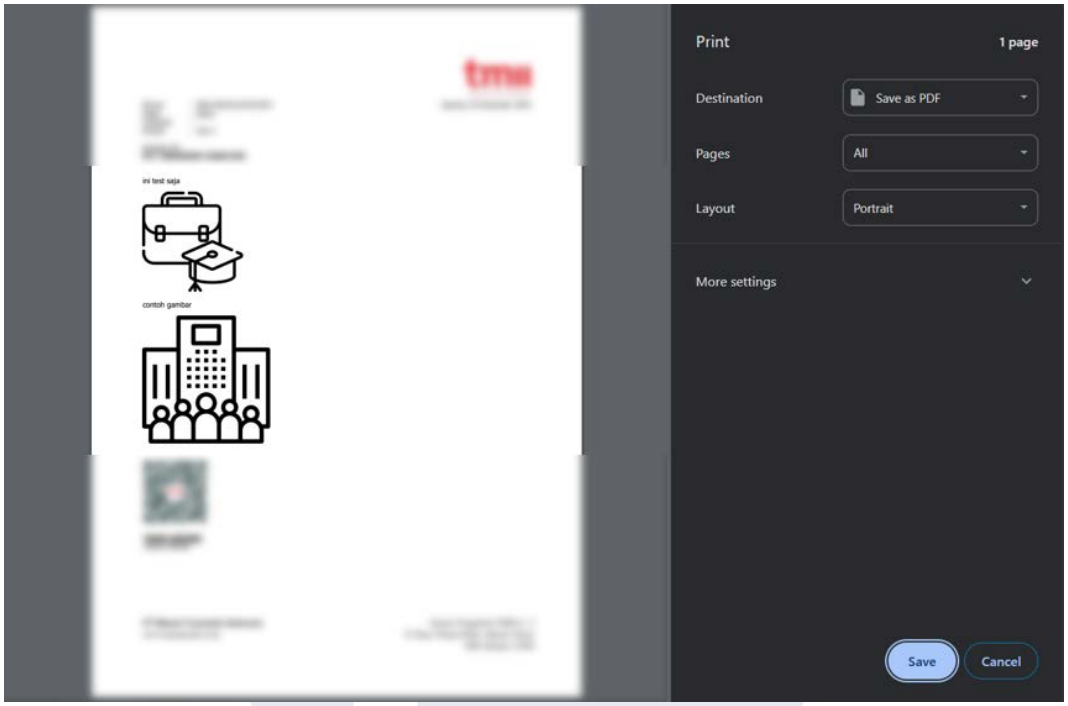
#### 3.3.1.6 E-Office - Mengembangkan field isi surat

Pada modul E-Office, salah satu kebutuhan baru datang dari pihak kedua proyek yang menginginkan agar isi surat dapat disisipkan gambar. Untuk memenuhi permintaan tersebut, dilakukan penyesuaian pada *field message* agar mendukung penyisipan gambar langsung di dalam teks isi surat. Proyek ini dikembangkan menggunakan CodeIgniter 3.1, sehingga seluruh penyesuaian harus tetap kompatibel dengan struktur sistem yang sudah berjalan.




Gambar 3.15 *Field message* yang dikembangkan

Versi teks editor yang digunakan, CKEditor, diperbarui untuk menyelesaikan penyesuaian. Editor diperbarui ke versi 4.22 karena versi awal 4.6 tidak mendukung penyisipan gambar. Seperti yang terlihat pada Gambar 3.15, gambar dapat ditambahkan ke dalam isi surat versi ini dengan menggunakan fitur *drag and drop*. Ini membuat proses penyusunan dokumen lebih mudah dan efisien.



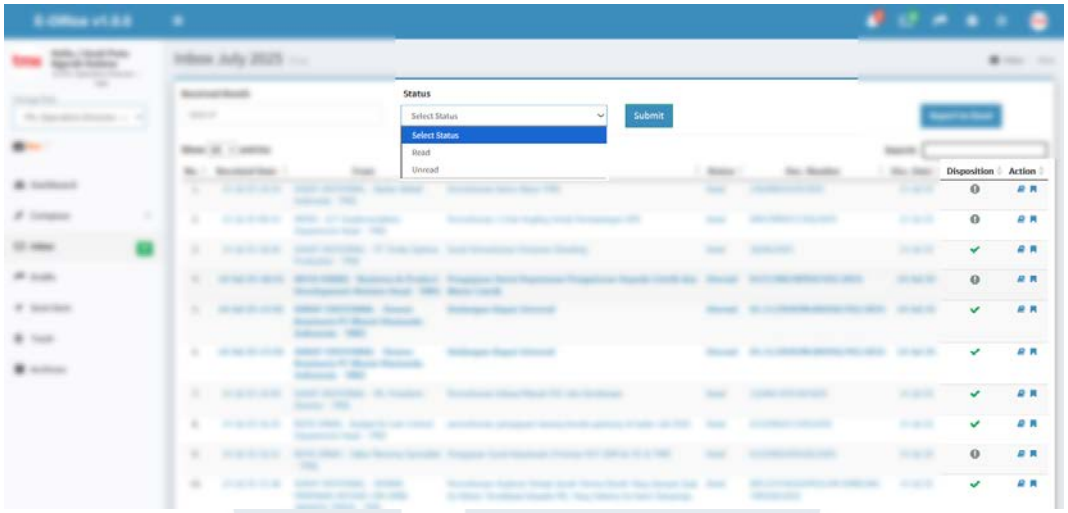

Gambar 3.16 Tampilan surat yang disisipkan gambar

Seperti yang ditunjukkan pada Gambar 3.16, setelah pembaruan diterapkan, gambar dapat ditempatkan di dalam isi surat seperti teks biasa. Pada awal, tengah, dan akhir paragraf, gambar ditempatkan dengan lebih mudah. Setelah menerapkan ini dengan sukses, kebutuhan pengguna terpenuhi, dan proses penyusunan surat menjadi lebih jelas dan sesuai dengan standar yang diharapkan pihak klien.

#### 3.3.1.7 E-Office - Menyesuaikan modul inbox

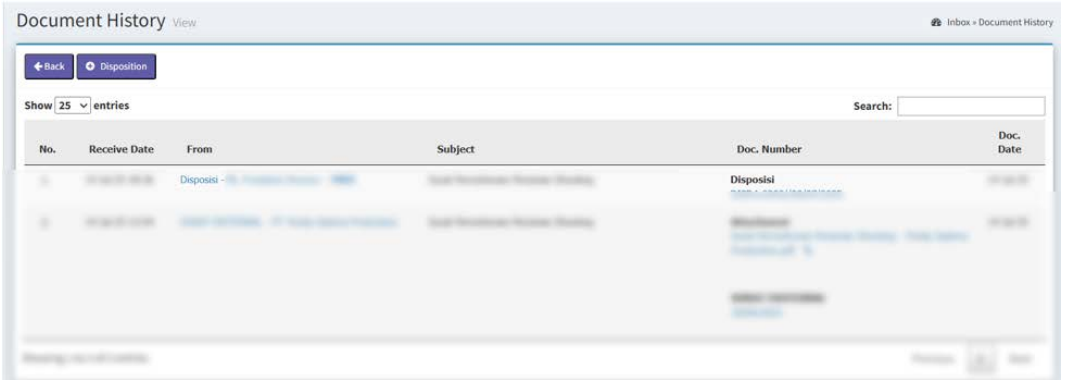
Pada modul inbox, ditemukan masalah ketika sebuah disposisi dikirimkan. Sistem menampilkannya seperti surat baru, sehingga daftar inbox terlihat penuh dan terkesan seperti spam. Untuk mengatasi hal ini, disposisi perlu ditampilkan sebagai bagian dari surat yang terkait, bukan sebagai surat terpisah. Selain itu filter status surat juga dibutuhkan pada inbox. Penyesuaian ini bertujuan

membuat tampilan inbox lebih rapi dan mudah dipahami oleh pengguna.




Gambar 3.17 Tampilan daftar inbox

Daftar inbox kemudian disempurnakan dengan menambahkan kolom *disposisi*. Kolom ini berfungsi menunjukkan apakah suatu surat memiliki disposisi atau tidak. Pada Gambar 3.17, jika surat tersebut memiliki disposisi, kolom akan menampilkan ikon centang. Jika tidak, kolom akan menampilkan tanda seru. Mekanisme ini membuat pengguna dapat langsung mengenali status disposisi tanpa harus membuka setiap surat satu per satu. Lalu filter dapat menyortir surat yang sudah dibaca atau yang belum dibaca.




Gambar 3.18 Tampilan lokasi disposisi

Pada halaman detail inbox yang ditampilkan pada Gambar 3.18, disposisi sekarang ditampilkan sebagai bagian yang dikelompokkan bersama surat utamanya. Dengan struktur semacam ini, tampilan inbox menjadi lebih ringkas dan tidak membingungkan, karena setiap disposisi ditautkan langsung ke dokumen yang relevan.

#### 3.3.1.8 Employee Lounge System - Menganalisis kebutuhan sistem

Dikarenakan harus diselesaikan dalam waktu kurang dari 1 bulan, proyek *Employee Lounge System* harus menjadi prioritas. Berbeda dengan proyek sebelumnya yang hanya mengubah atau menambah fitur, proyek ini dibangun dari awal. Karena lebih cepat, mudah digunakan, dan dapat digunakan lintas *platform*, sistem dibuat dalam bentuk web.


Gambar 3.19 Employee Lounge T3

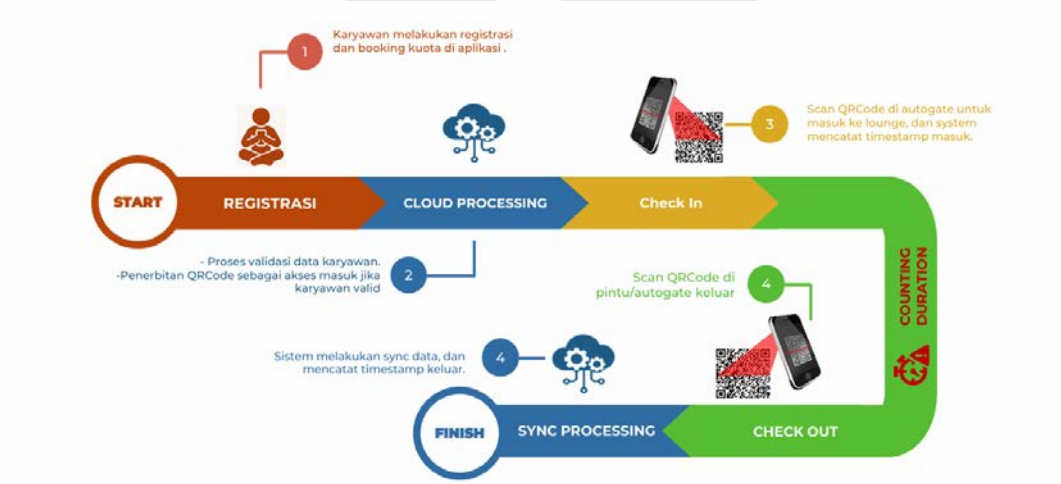
Employee Lounge T3 merupakan fasilitas yang direquest oleh Injourney Airport dan berlokasi di Bandara Soekarno Hatta Terminal 3 domestik. Lounge ini disediakan sebagai tempat beristirahat bagi karyawan selama jam istirahat pada *shift* mereka, sehingga mereka memiliki ruang yang nyaman dan tertata. Di dalamnya tersedia berbagai fasilitas yang dapat digunakan langsung oleh karyawan, seperti kamar, loker, toilet, *smoking room*, dan kantin.


Gambar 3.20 Kebutuhan *Employee Lounge System*

Sistem ini memiliki tiga proses inti seperti yang ditampilkan pada Gambar 3.20. Pertama, fitur *monitoring* ruangan yang memungkinkan sistem melihat ketersediaan ruang, jumlah kapasitas yang tersedia, serta berapa banyak orang yang sedang berada di dalam ruangan. Kedua, *access gate* yang berada di luar ruang lingkup sistem karena merupakan bagian dari perangkat keras yang sudah tersedia. Ketiga, *access control* berupa akses masuk ke ruangan menggunakan kode QR yang dapat dipindai untuk membuka akses.

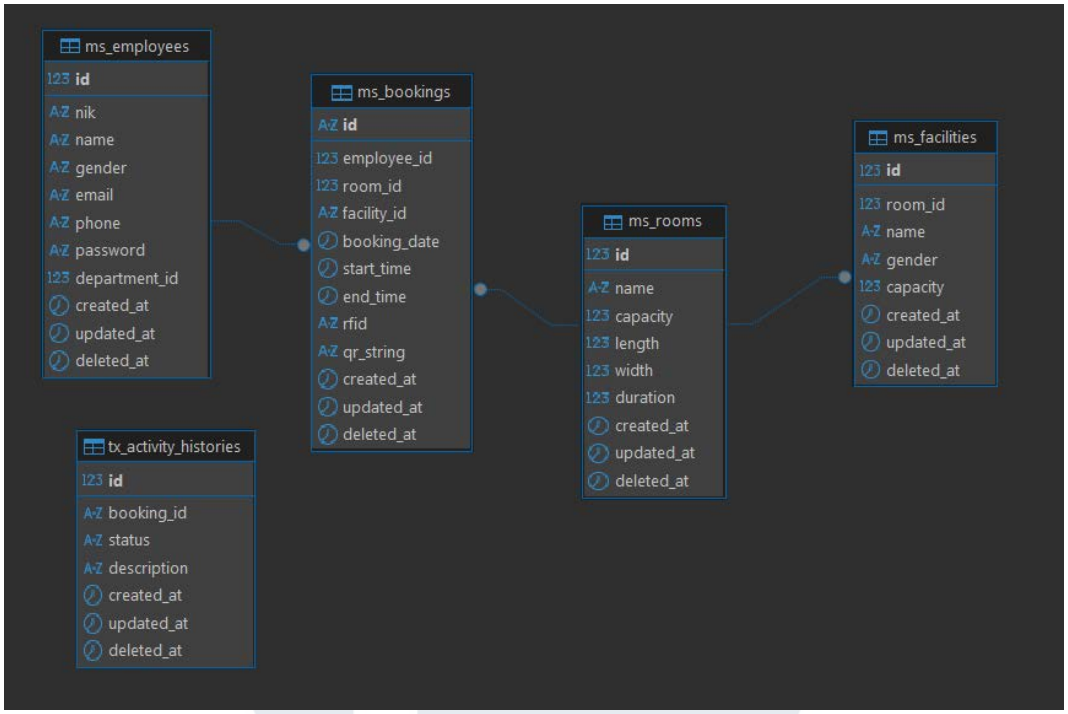
#### 3.3.1.9 Employee Lounge System - Mendesain alur sistem & database

Berdasarkan kebutuhan yang sudah dianalisis, alur sistem dirancang untuk mendukung proses pemesanan, akses masuk, dan pencatatan aktivitas secara otomatis. Perancangan ini dibuat untuk meminimalkan proses manual yang berpotensi menyebabkan kekeliruan data maupun ketidakteraturan alur kerja.




Gambar 3.21 Alur sistem yang dirancang

Alur pemesanan ruangan dapat dilihat pada Gambar 3.21. Proses dimulai ketika karyawan melakukan pemesanan melalui website. Pengguna harus *login* terlebih dahulu, kemudian mengisi formulir pemesanan sesuai ruangan yang diinginkan. Setelah data diverifikasi dan dinyatakan valid, sistem akan menghasilkan kode QR yang menjadi akses masuk ke lounge. Saat tiba di *autogate*, karyawan dapat memindai QR tersebut untuk memasuki ruangan. Ketika selesai menggunakan fasilitas *lounge*, karyawan melakukan *checkout* dengan QR yang sama. Waktu masuk dan keluar dicatat otomatis oleh sistem sebagai bagian dari proses *monitoring*.




Gambar 3.22 Design *Database*

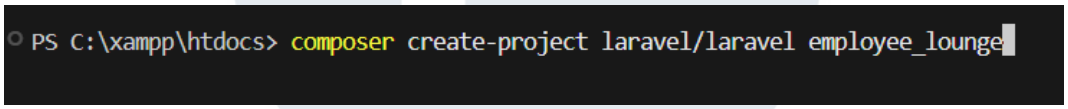
Struktur relasi utama yang diperlukan untuk sistem digambarkan pada Gambar 3.22. Data karyawan, termasuk NIK, nama, dan informasi departemen, disimpan dalam tabel ‘ms_employees’. Berikutnya, data master ruangan disimpan dalam tabel ‘ms_rooms’. Tabel ini mencakup informasi dasar seperti kapasitas, batas waktu penggunaan, dan informasi lain yang diperlukan untuk proses reservasi. Tabel ini terhubung ke ‘ms_facilities’, yang berisi daftar fasilitas ruangan, seperti loker dan fasilitas pendukung lainnya.

Selanjutnya, tabel ‘ms_bookings’ digunakan sebagai pusat proses pemesanan untuk menghubungkan semua tabel tersebut. Tabel ini menunjukkan hubungan antara karyawan, ruangan, dan fasilitas apa pun yang dipilih saat pemesanan dibuat. Selain itu, tabel ‘tx_activity_histories’ dapat digunakan untuk menyimpan semua riwayat aktivitas, termasuk waktu pemesanan, check-in, dan check-out, serta proses pembatalan reservasi. Struktur seperti ini

memungkinkan sistem untuk menjaga integritas data dan memproses alur reservasi secara lebih terorganisir.

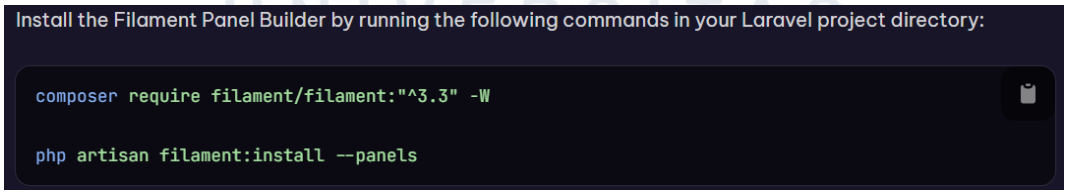
#### 3.3.1.10 Employee Lounge System - Mengkode & membangun sistem

Sistem *Employee Lounge* dibangun dengan Laravel Filament untuk sisi admin dan Tailwind CSS untuk sisi klien. Untuk memenuhi kebutuhan proyek yang memiliki tenggat waktu yang singkat, Filament adalah *toolkit package* admin resmi Laravel, yang memungkinkan pembuatan panel admin dengan cepat dan fleksibel. Pengembangannya dimulai dengan membangun proyek Laravel dengan composer.




Gambar 3.23 *Command* membuat laravel

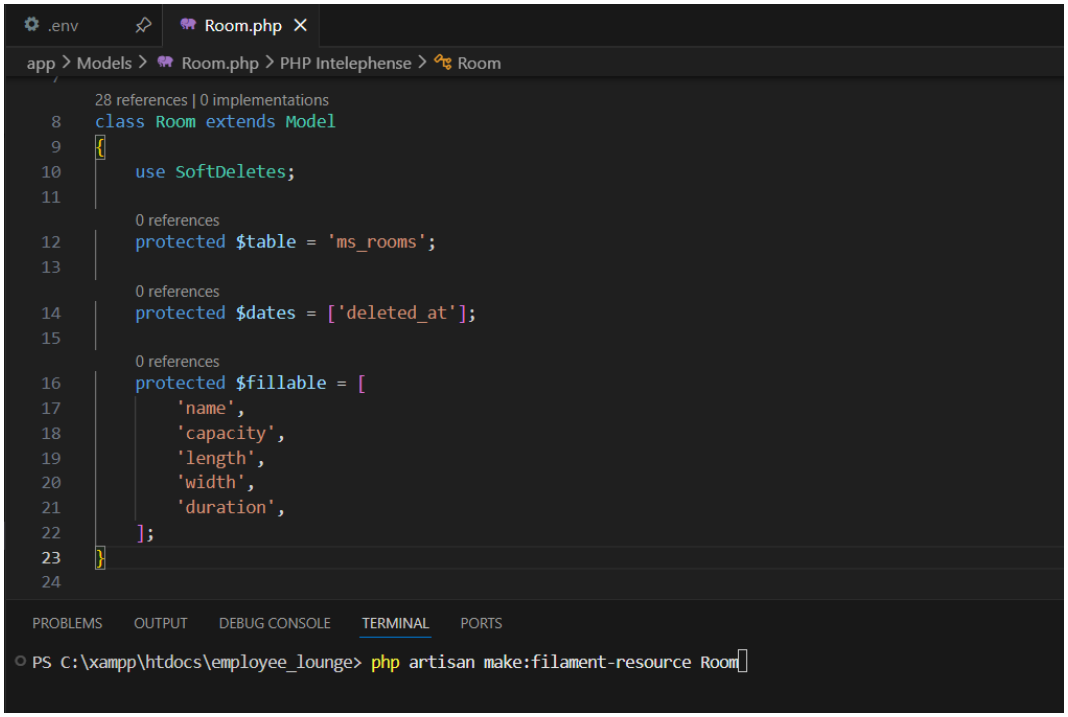
Karena versi Laravel yang spesifik tidak ditentukan, perintah yang ditunjukkan pada Gambar 3.23 akan menginstal versi terbaru dari Laravel. Setelah Laravel terpasang, langkah selanjutnya adalah membaca instruksi resmi Filament untuk proses instalasi dan konfigurasi awal. Instruksi ini dapat ditemukan di situs web resmi Filament. Dokumentasi ini memberikan penjelasan detail yang memudahkan proses *setup*, menjadikannya panduan utama.




Gambar 3.24 Dokumentasi filament
Sumber: Dokumentasi Filament (2025) [8]

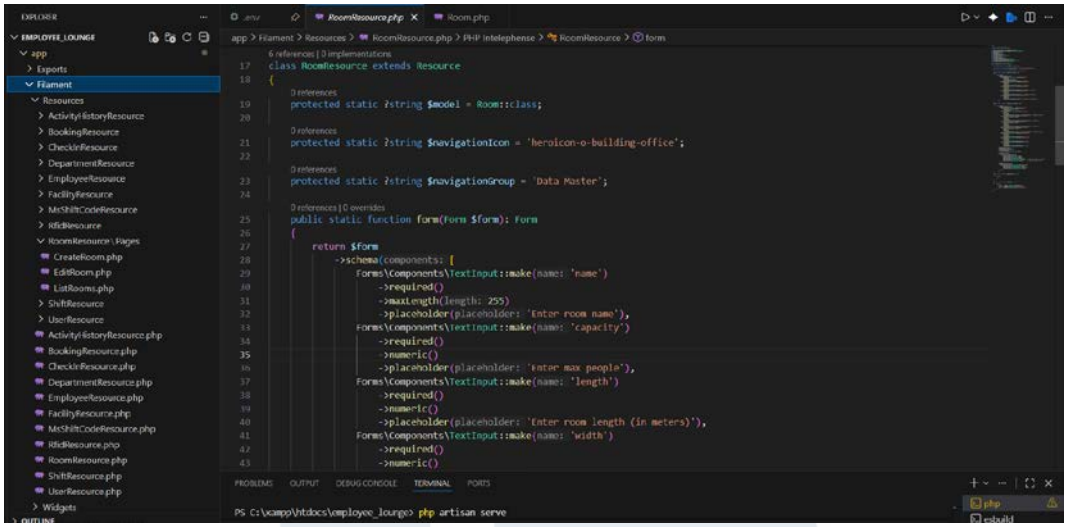
Dokumentasi Filament yang sangat lengkap dan mudah dipahami dapat dilihat pada Gambar 3.24. Web panduan ini mencakup petunjuk untuk instalasi, pembuatan tabel, formulir,

tampilan, dan pembuatan *widget* yang dapat disesuaikan. Dokumentasi ini membantu proses implementasi menjadi lebih fokus dan mengurangi kesalahan teknis.



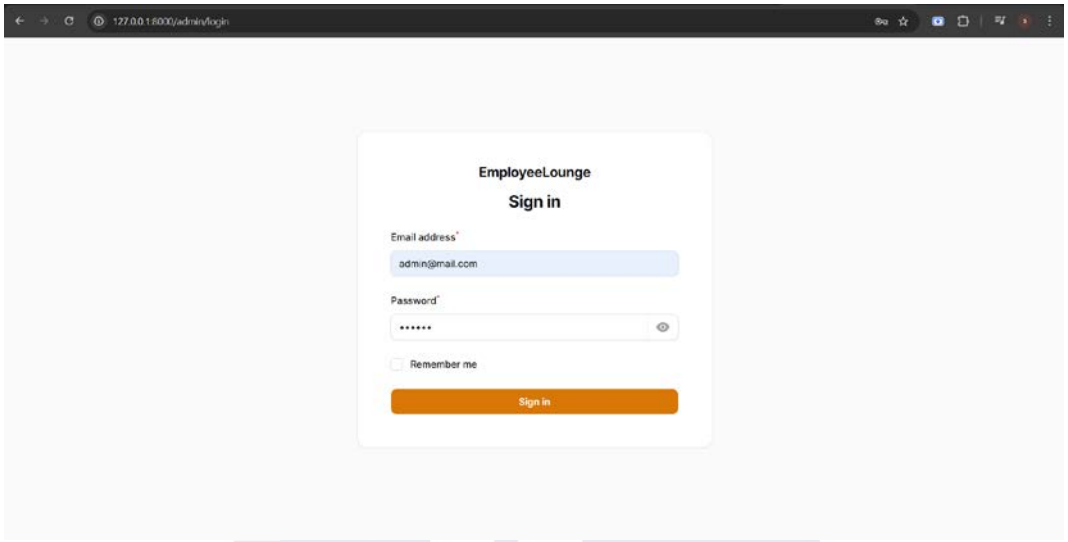

Gambar 3.25 Membuat *resource* filament

Untuk membangun tabel, *form*, dan *view*, langkah pertama adalah menyiapkan struktur tabel di *database* melalui *migration* atau *query* langsung. Setelah itu dibuat model yang mendefinisikan tabel dan kolom yang digunakan sebagai representasi data di Laravel. Selanjutnya, seperti pada Gambar 3.25, *resource* untuk model dapat dibuat melalui *command line* sehingga otomatis menghasilkan halaman daftar, *form create*, *form edit*, dan halaman detail yang siap digunakan dalam panel admin.



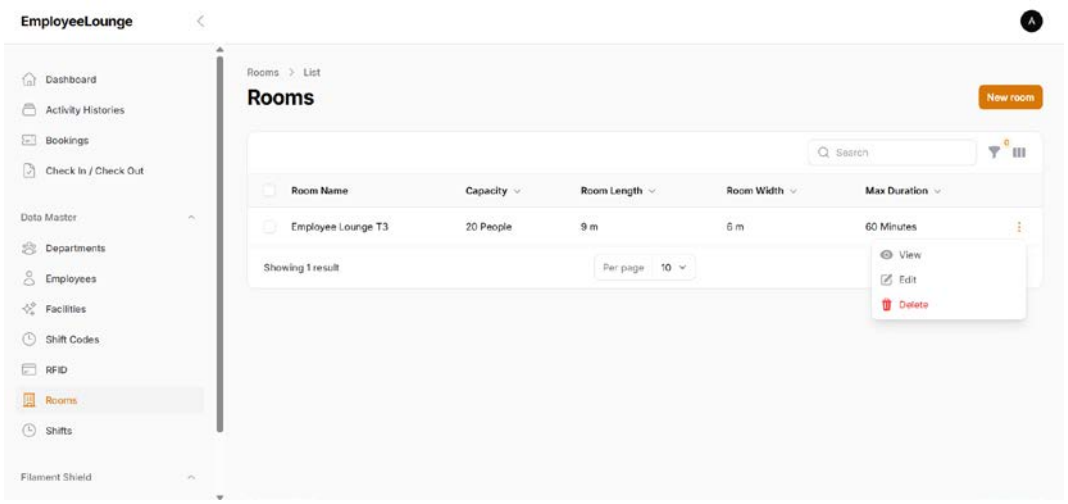

Gambar 3.26 Struktur folder dan kode

Hasil pembuatan sumber daya pada filament ditunjukkan pada Gambar 3.26. Filament secara otomatis membuat *file-file* yang diperlukan dan menempatkannya di dalam folder khusus Filament setelah perintah dijalankan. Ini memastikan struktur proyek tetap teratur. *File* ‘RoomResource.php’, pusat konfigurasi modul Ruangan, berfungsi untuk mengatur *field* di tabel dan *form* ruangan. Untuk menjalankan sistem, proses eksekusi dilakukan dengan menggunakan perintah “php artisan serve”, seperti yang dilakukan untuk menjalankan aplikasi Laravel lainnya.



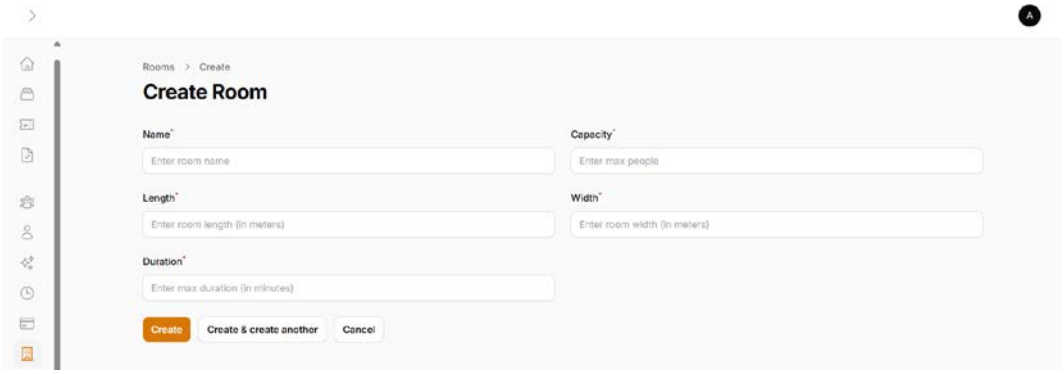

Gambar 3.27 Tampilan login

Tampilan login untuk sisi admin yang sederhana dan mudah dipahami ditunjukkan pada Gambar 3.27. Tampilan yang sederhana ini memungkinkan administrator masuk ke sistem dengan mudah karena hanya menampilkan item yang benar-benar diperlukan. Selama proses instalasi Filament, *path* "/admin" digunakan untuk mengakses halaman ini, dan akun administrator pertama dibuat secara otomatis.



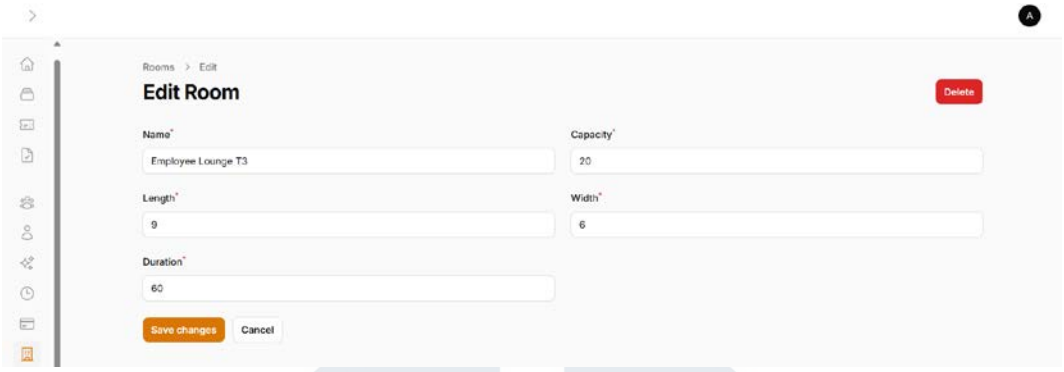

Gambar 3.28 tampilan modul room

Daftar tabel ruangan yang sudah terdaftar di dalam sistem merupakan tampilan awal modul Ruangan, seperti yang ditunjukkan pada Gambar 3.28. Tombol pada bagian atas berfungsi untuk membuat data ruangan baru, yang memungkinkan admin menambah entri dengan cepat. Setiap baris data memiliki tombol ‘View ‘dan ‘Edit’ yang dapat digunakan untuk melakukan perubahan data. Mekanisme *soft deletes* memungkinkan data tidak hilang secara permanen, sehingga dapat dipulihkan apabila terjadi kesalahan. Jika tidak dikustomisasi, tampilan ini berlaku untuk semua model yang telah dibuat.



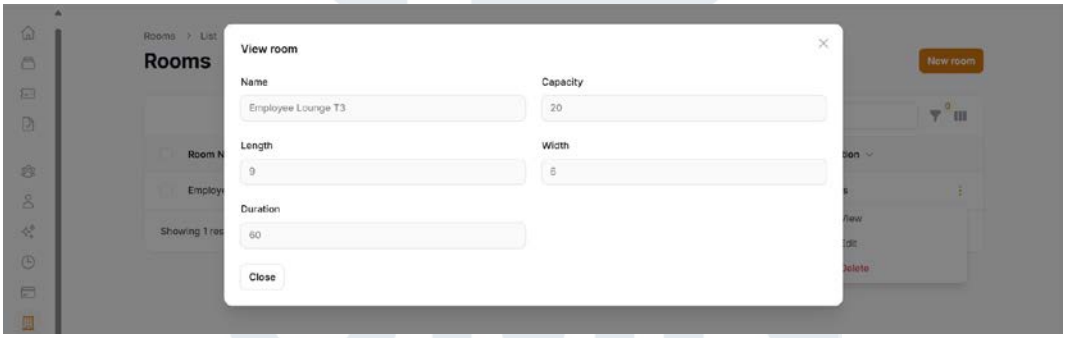

Gambar 3.29 Tampilan *form create*

Contoh *form* yang digunakan untuk membuat data baru pada modul Ruangan dapat dilihat pada Gambar 3.29. Filament sudah menyiapkan semua kolom yang diperlukan secara otomatis, jadi admin dapat memulai proses input tanpa menulis atribut nama atau identitas secara manual. Karena setiap *field* sudah terhubung langsung ke struktur *database* melalui konfigurasi di ‘RoomResource.php’, mekanisme ini mempermudah proses penyimpanan data.




Gambar 3.30 Tampilan *form edit*

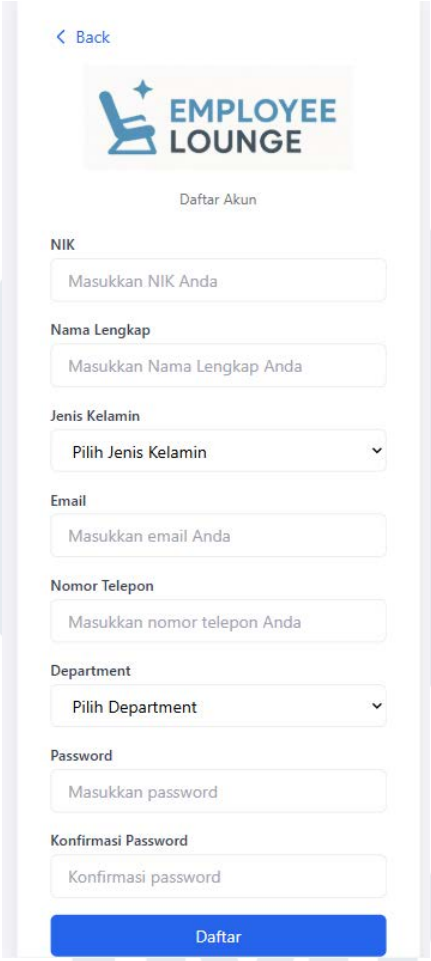
*Form* yang digunakan untuk mengeditan data yang sudah disimpan ditunjukkan pada Gambar 3.30. Ketika formulir dibuka, semua kolom akan otomatis terisi dengan data yang telah dimasukkan sebelumnya, sehingga admin tidak perlu mengetik ulang data yang tidak berubah. Metode ini mengurangi kemungkinan kesalahan pengisian dan mempercepat proses koreksi data.




Gambar 3.31 Tampilan *view detail*

Gambar 3.31 menunjukkan *view* yang digunakan untuk melihat detail data ruangan. Tampilan ini hanya menampilkan informasi tanpa opsi untuk mengubahnya, jadi cocok untuk digunakan ketika administrator hanya ingin memverifikasi data tertentu. Dengan memisahkan fungsi melihat data dan fungsi mengedit data, kehadiran halaman *view* membantu menjaga

kerapihan modul. Seperti yang ditunjukkan pada *form*, tabel, dan halaman *view* di atas, hasil bawaan Filament tidak diubah.



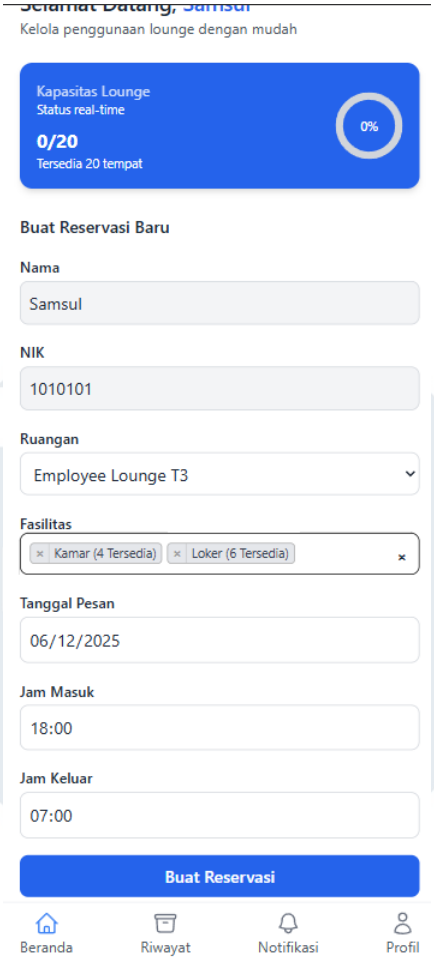

Gambar 3.32 Tampilan Registrasi Karyawan

Kode dibuat pada sisi klien tanpa filament. Sebagian besar kode Laravel menggunakan PHP, dan gaya dibuat dengan Tailwind CSS. Halaman registrasi yang dapat diakses karyawan digunakan untuk membuat akun baru. Karena sistem belum terhubung langsung ke data karyawan perusahaan, halaman ini diperlukan. Form registrasi meminta nama lengkap, jenis kelamin, email, nomor telepon, departemen, dan password yang akurat, seperti yang ditunjukkan pada Gambar 3.32. Setelah proses registrasi berhasil,

karyawan dapat menggunakan sistem untuk memesan ruangan di *lounge* karyawan.

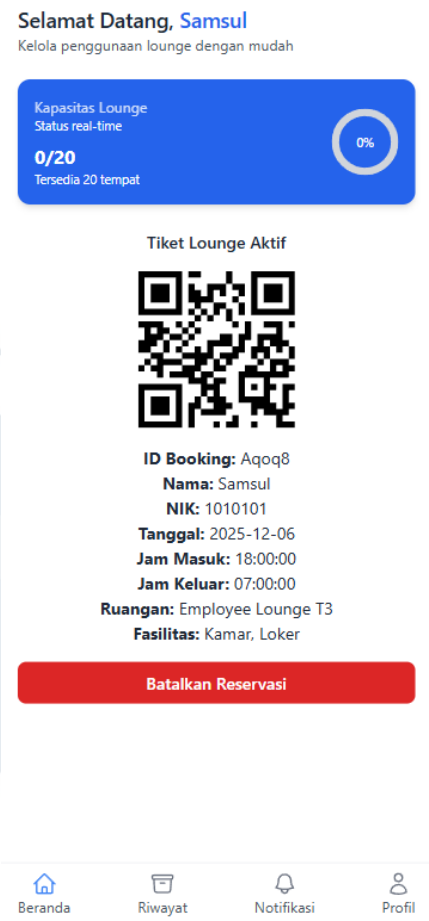

Gambar 3.33 Tampilan Login

*Form login*, seperti yang ditunjukkan pada Gambar 3.33, tersedia selain halaman registrasi. Hanya NIK dan *password* yang telah didaftarkan sebelumnya diperlukan untuk mengisi formulir ini. Jika karyawan memasukkan NIK atau *password* yang salah atau tidak terdaftar, sistem akan menolak proses login, sehingga tidak dapat memesan *lounge*.



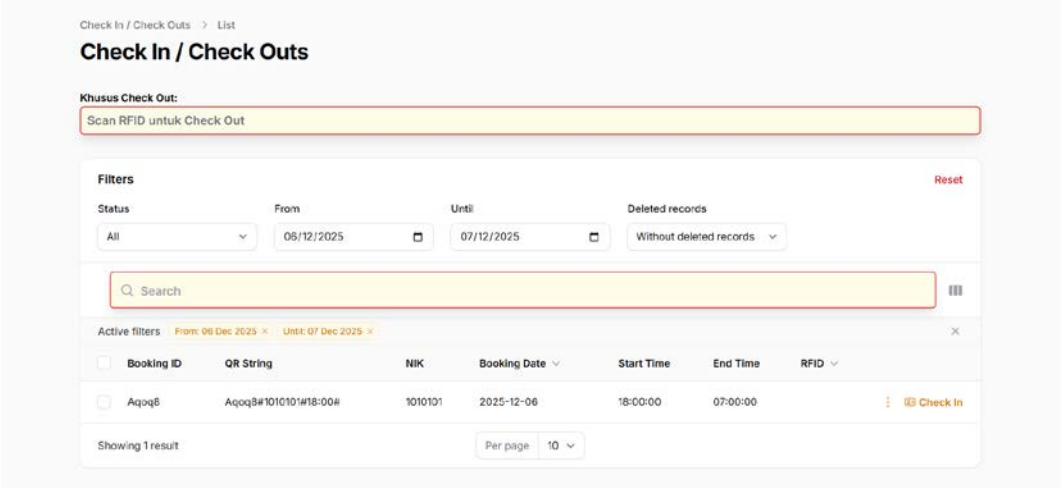

Gambar 3.34 Tampilan beranda

Seperti yang ditunjukkan pada Gambar 3.34, karyawan akan diarahkan ke halaman beranda setelah login berhasil. Karyawan dapat melihat kapasitas *lounge* secara *real-time* dan mengakses *form* reservasi pada halaman ini. Nama dan NIK telah terisi otomatis sesuai dengan akun login. Namun, saat ini hanya *Employee Lounge* T3 yang tersedia. Sesuai ketersediaan, fasilitas dapat dipilih lebih dari satu atau tidak sama sekali, seperti loker dan kamar. Selain itu, tanggal pemesanan akan otomatis menyesuaikan jadwal kerja karyawan.



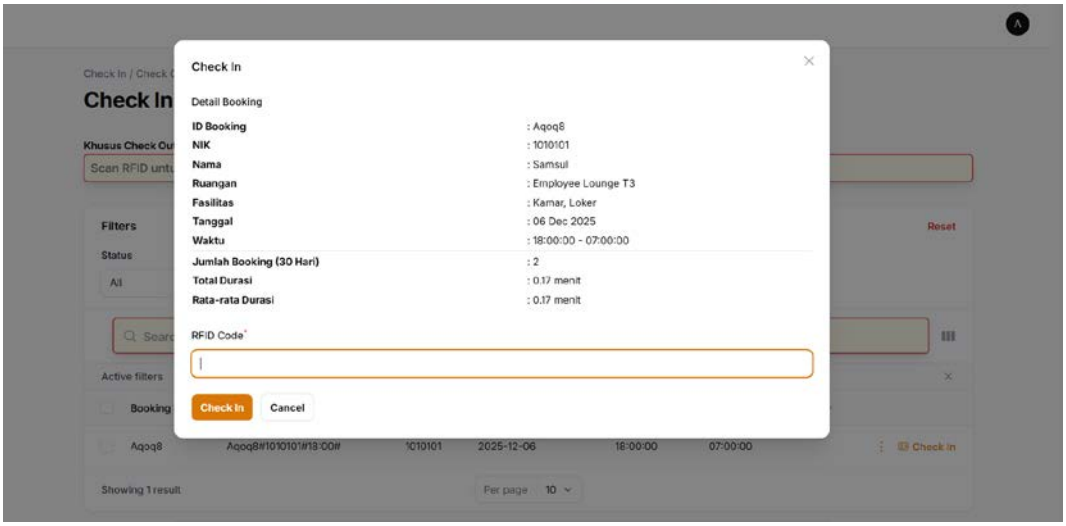

Gambar 3.35 Tampilan selesai reservasi

Karyawan akan menerima kode QR, seperti yang ditunjukkan pada Gambar 3.35, setelah proses reservasi selesai. Saat memasuki *lounge*, kode QR ini digunakan untuk melakukan *scan*. Halaman ini menampilkan kembali detail pemesanan selain QR untuk memastikan datanya cocok. Pada bagian bawah juga ada tombol pembatalan reservasi. Jika karyawan ingin membatalkan pemesanan, mereka harus mengisi alasan mereka.




Gambar 3.36 Tampilan *Check In/Check Out* pada admin

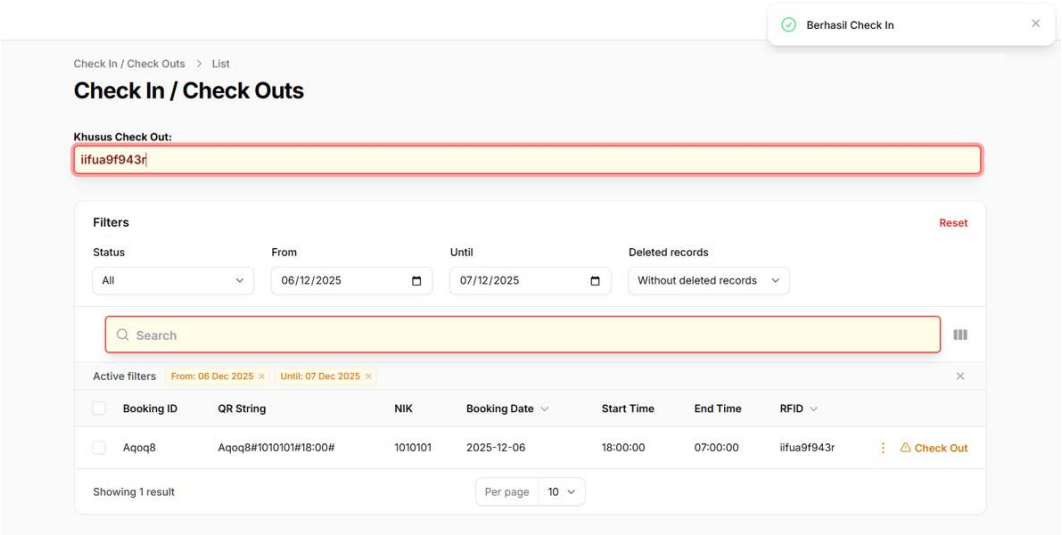
Data pesanan akan muncul di sisi admin dan resepsionis, seperti yang ditunjukkan pada Gambar 3.36, setelah karyawan melakukan reservasi. NIK, tanggal, dan jam pemesanan ditunjukkan, sehingga resepsionis dapat memastikan bahwa data tersebut benar. Saat karyawan tiba di *lounge* dan melakukan pemindaian kode QR, resepsionis akan melihat tampilan khusus Filament, yang mencakup field *Check In/Out*, fitur pencarian, filter, dan data pada tabel.




Gambar 3.37 Tampilan modal *Check In* pada admin

Seperti yang ditunjukkan pada Gambar 3.37, sebuah modal akan muncul setelah mengklik tombol ‘Check In’. Modal tersebut

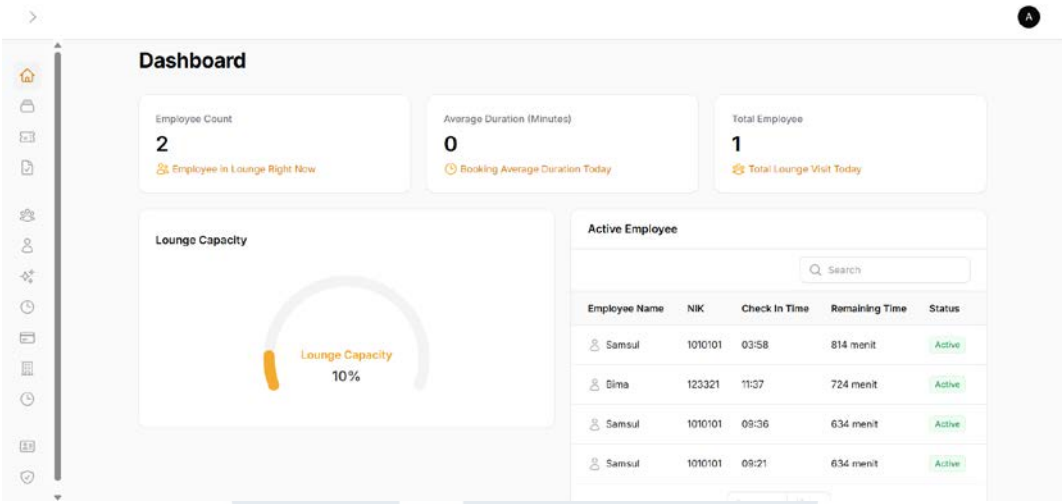
menampilkan detail karyawan dan riwayat pesanan, termasuk jumlah kunjungan, durasi rata-rata, dan total waktu penggunaan *lounge*. *Field* RFID digunakan untuk mencatat kartu RFID yang dipinjamkan karyawan. Setelah kode RFID dimasukkan, status pemesanan akan berubah dari “booking” menjadi “check-in”, dan karyawan akan mendapatkan kartu akses.




Gambar 3.38 Tampilan *Check In/Check Out* pada admin

Seperti yang ditunjukkan pada Gambar 3.38, notifikasi akan muncul di bagian kanan atas dan kolom RFID akan terisi otomatis sesuai kode kartu yang diberikan setelah proses verifikasi selesai. Untuk menyelesaikan proses *check-out*, karyawan hanya perlu mengembalikan kartu RFID kepada resepsionis ketika mereka keluar dari *lounge*. Resepsionis dapat melakukan *checkout* karyawan dengan dua cara: melalui tombol ‘Check Out’ yang menampilkan modal pemindaian RFID; atau dapat langsung memindai modal

RFID pada *field* ‘Khusus Check Out’ untuk mengubah status karyawan menjadi *checkout* otomatis.




Gambar 3.39 Tampilan *dashhboard* admin

Seperti yang ditunjukkan pada Gambar 3.39, Sistem *Employee Lounge* juga menyediakan *dashboard* yang menampilkan ringkasan informasi penting. *Dashboard* menampilkan jumlah karyawan yang sedang berada di *lounge*, durasi rata-rata penggunaan *lounge*, dan jumlah karyawan yang melakukan *check-in* pada hari tersebut. Grafik di baris kedua menunjukkan persentase kapasitas *lounge*, dan di bagian paling kanannya terdapat tabel yang menunjukkan daftar karyawan yang sedang *check-in*, menunjukkan sisa waktu penggunaan dan statusnya, apakah masih aktif atau sudah melewati waktu *check-out* (*expired*).

#### 3.3.1.11 Employee Lounge System - *Deploy, test, & review* sistem

Setelah proses *development* selesai, sistem langsung dicek dan di-*deploy* oleh supervisor sehingga bisa digunakan secara *live*. Setelah itu, sistem diperkenalkan kepada pengguna dan dilakukan sesi penjelasan mengenai alur penggunaan. Pada Gambar 3.40 terlihat momen ketika alur sistem dijelaskan kepada *user*, yaitu resepsionis dan *security* sebagai petugas yang bertanggung jawab

terhadap operasional *lounge*. Dalam sesi tersebut dilakukan penjelasan fitur, demonstrasi proses *check in* dan *check out*, serta tanya jawab dan review dari pihak pengguna.

Gambar 3.40 Momen menjelaskan alur sistem dan *review* oleh *user*

Proses testing dilakukan secara langsung oleh resepsionis sebagai pengguna utama. Pengujian berfokus pada kelancaran alur *check in* dan *check out* karena proses tersebut merupakan inti dari layanan di *employee lounge*. Resepsionis mencoba seluruh fitur sambil menggunakannya dalam situasi nyata, lalu mereka juga menjelaskan cara penggunaan kepada karyawan lain yang ingin melakukan reservasi.

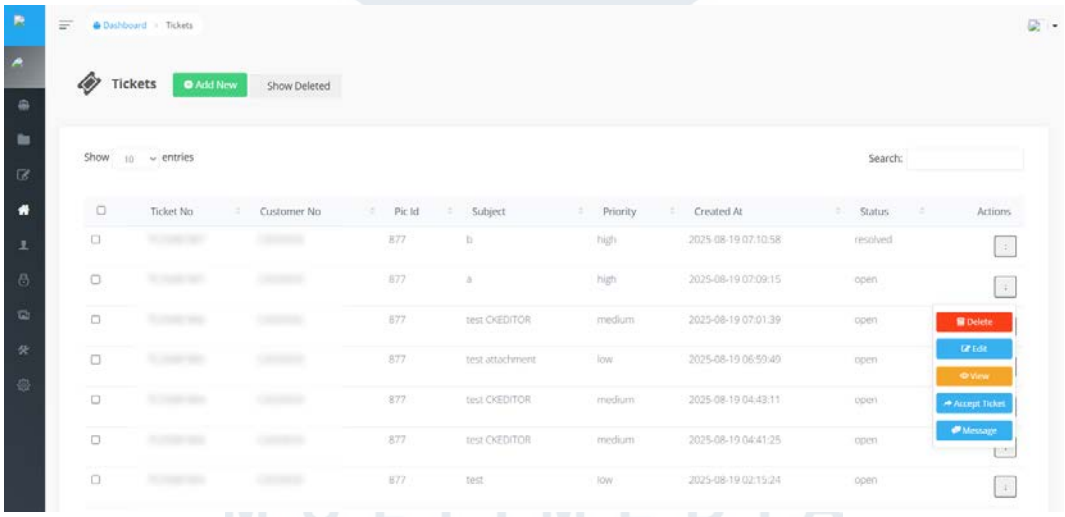
Hasil uji coba menunjukkan bahwa sistem mudah digunakan dan memiliki alur yang jelas. *User* sempat memberi masukan agar proses *check out* dibuat lebih sederhana, dan perbaikan tersebut langsung diterapkan pada sesi uji berikutnya. Testing berlangsung

beberapa kali selama kurang lebih 2 minggu akhir periode projek hingga sistem stabil dan dinyatakan layak digunakan.

Untuk menghindari masalah teknis, monitoring dilakukan setelah sistem diaktifkan dan digunakan. Untuk memastikan layanan tetap berjalan lancar, aduan seperti *bug* atau masalah kecil segera diperbaiki. Dengan kombinasi proses ini, sistem dapat digunakan dengan baik dan diterima dengan baik oleh pengguna.

#### 3.3.1.12 Tenant Management System - Membuat fitur *ticket*

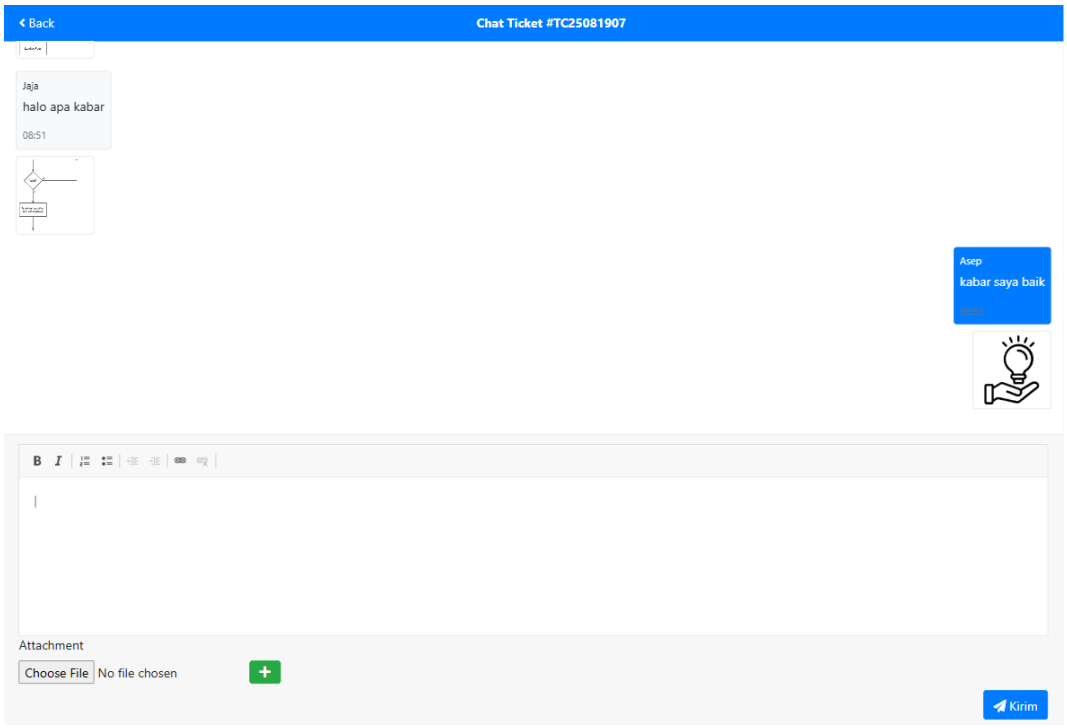
Ketika terjadi masalah dengan layanan yang digunakan, PIC harus dapat melaporkannya melalui fitur tiket pengaduan. Dengan fitur ini, proses penanganan keluhan akan lebih terorganisir karena setiap keluhan dapat dikelola secara langsung dalam sistem tanpa menggunakan aplikasi pesan eksternal. Akibatnya, alur komunikasi antara PIC dan admin menjadi lebih terstruktur dan tercatat.




Gambar 3.41 Tampilan modul tiket admin

Daftar tiket yang diajukan oleh PIC *tenant* ditampilkan di tampilan admin sistem, seperti yang ditunjukkan pada Gambar 3.19. Dengan mengklik ‘Accept Ticket’, admin dapat menerima dan memproses tiket, sehingga statusnya berubah menjadi *in progress*. Saat tiket berada pada status tersebut, percakapan tidak dapat

dilakukan. Tampilan obrolan akan muncul ketika manajer menekan tombol Pesan. Ini memungkinkan admin menanggapi laporan langsung. Admin dapat menutup tiket ketika dianggap sudah terselesaikan dengan mengklik tombol ‘resolved’.

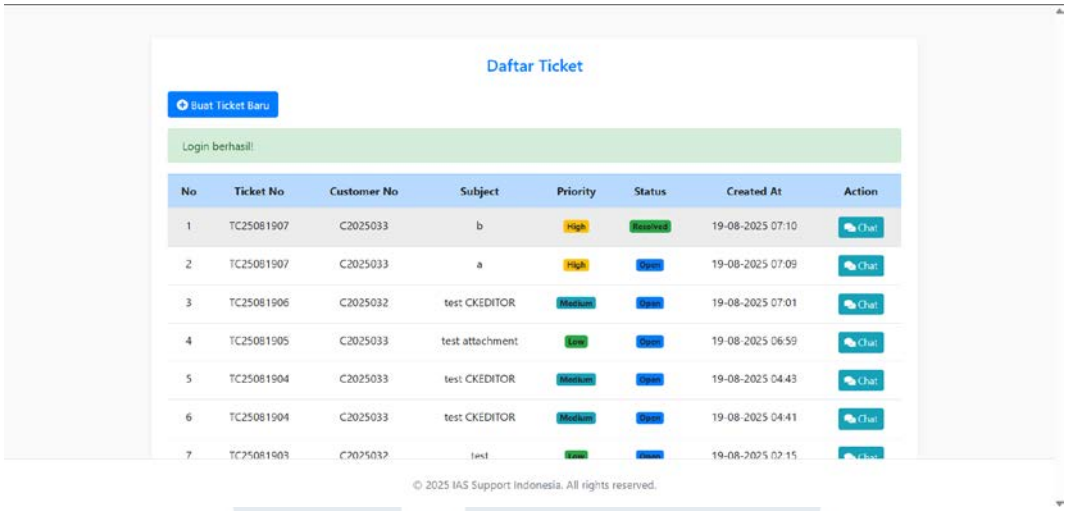



Gambar 3.42 Percakapan sisi admin

Sebagai *editor* teks, admin dapat menulis pesan dengan nyaman menggunakan CKEditor di bagian percakapan. Selain itu, sistem memungkinkan pengiriman lampiran seperti foto atau bukti terkait keluhan, sehingga komunikasi menjadi lebih jelas dan mencegah interpretasi yang salah. Ini adalah fitur yang membantu manajer memahami konteks masalah dengan lebih baik.

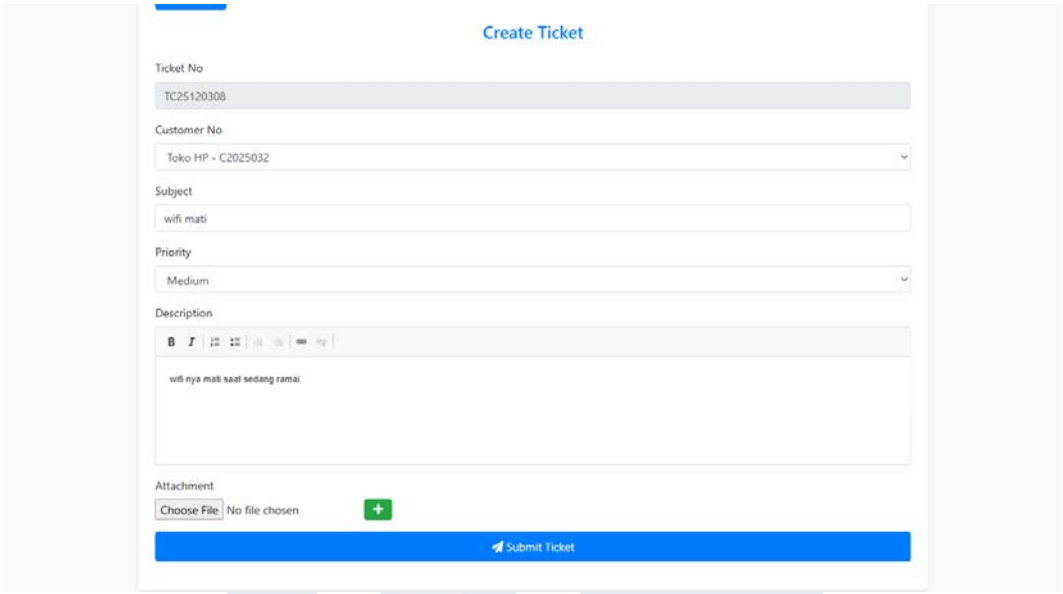
Tampilan percakapan pada sisi PIC memiliki fungsi yang sama dengan admin seperti Gambar 3.42. Perbedaannya hanya terletak pada posisi dan warna *bubble chat* agar pengguna dapat membedakan pesan yang dikirim maupun diterima. Dengan desain

ini, percakapan menjadi lebih nyaman dibaca dan tidak membingungkan selama proses penyelesaian keluhan berlangsung.




Gambar 3.43 Tampilan daftar ticket tenant

Pada sisi PIC, pengguna harus melakukan login terlebih dahulu menggunakan akun yang telah didaftarkan oleh admin. Setelah berhasil masuk, PIC akan melihat daftar tiket yang pernah diajukan, lengkap dengan nomor tiket, *customer number*, *subject*, prioritas, status, serta waktu pembuatan tiket. Tampilan seperti Gambar 3.21 ini memudahkan PIC untuk memantau perkembangan laporan yang sedang diproses.




Gambar 3.44 Tampilan Form buat ticket baru

Pada Gambar 3.22, Saat ingin membuat tiket baru, PIC dapat memilih *customer number* yang mewakili layanan tenant yang sedang mengalami kendala. Subjek dan prioritas juga dapat ditentukan, mulai dari *low*, *medium*, *high*, hingga *urgent*. Pada bagian deskripsi, PIC bisa menjelaskan masalah secara rinci serta menambahkan lampiran untuk memperkuat penjelasan, sehingga admin memiliki informasi yang cukup untuk melakukan verifikasi awal.

#### 3.3.1.13 Smart Utility - Mengembangkan UI client

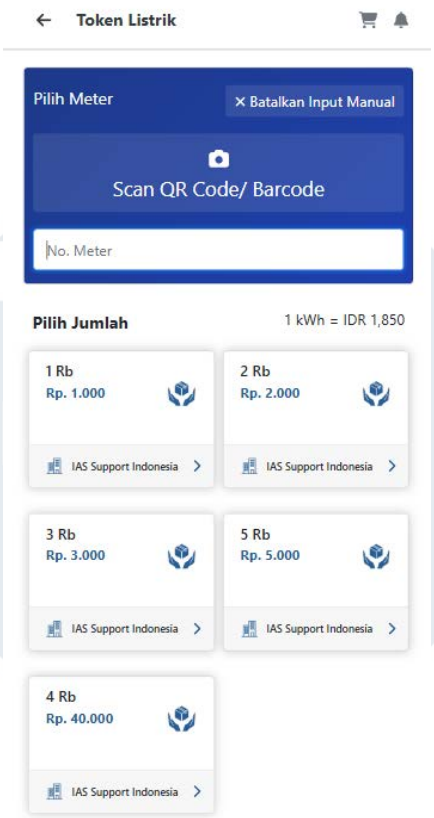
Pengembangan sistem Smart Utility adalah proyek berikutnya. Sistem ini akan mengatur pembelian listrik dan air *tenant* di sekitar Bandara Soekarno-Hatta. Sistem ini memudahkan pengelola bandara untuk melacak penggunaan listrik dan air setiap *tenant* secara langsung. Vue.js digunakan untuk tampilan sisi klien *mobile*. Ini membuat antarmuka lebih responsif, kontemporer, dan

mudah digunakan untuk tenant yang membutuhkan akses cepat melalui smartphone pengguna.


Gambar 3.45 Tampilan *dashboard mobile client*

Tampilan *dashboard* yang telah dikembangkan ditunjukkan pada Gambar 3.45. *Widget* pertama menampilkan sisa saldo pengguna dalam bentuk satuan air atau sisa kWh jam listrik. Di sebelah kanan, ada *dropdown* yang memungkinkan pengguna memilih *tenant* lain yang dikelolanya. Bagian tengah menampilkan pembelian terakhir, bersama dengan ringkasan analitik yang mencakup tujuh pembelian terakhir yang rata-rata, tertinggi, dan terendah. Di bagian bawah, pengguna dapat dengan cepat mengakses

riwayat pembelian, statistik penggunaan, layanan dukungan, dan halaman pembelian.



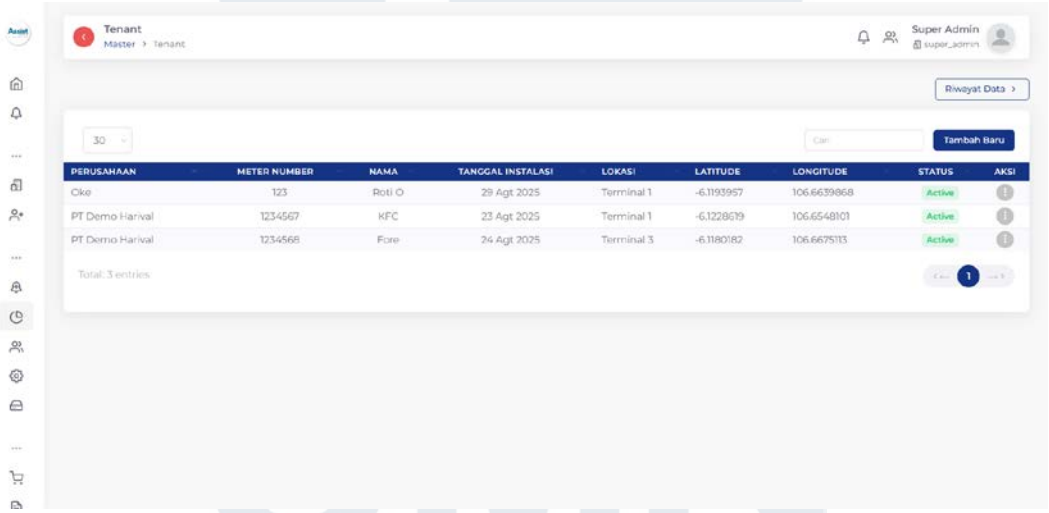

Gambar 3.46 Tampilan daftar produk

Daftar produk, seperti yang ditunjukkan pada Gambar 3.46, juga diubah tampilannya. Pada bagian atas terdapat kolom untuk memasukkan nomor meteran, yang dapat dipindai menggunakan kamera atau dimasukkan secara manual melalui *keyboard*. Ini memberikan pengguna dua opsi input yang berbeda. Agar informasi biaya mudah diakses, harga satuan produk ditampilkan di luar *widget* utama. Selain itu, daftar produk diatur ulang menjadi tampilan dua kolom per blok, daripada tampilan satu kolom yang memanjang sebelumnya. Ini membuat tampilan lebih rapi dan nyaman dilihat

pada *smartphone*. Pengalaman pengguna saat membeli listrik dan air menjadi lebih mudah dan efisien berkat perubahan ini.

#### 3.3.1.14 Smart Utility - Mengembangkan UI admin & Menambahkan data master

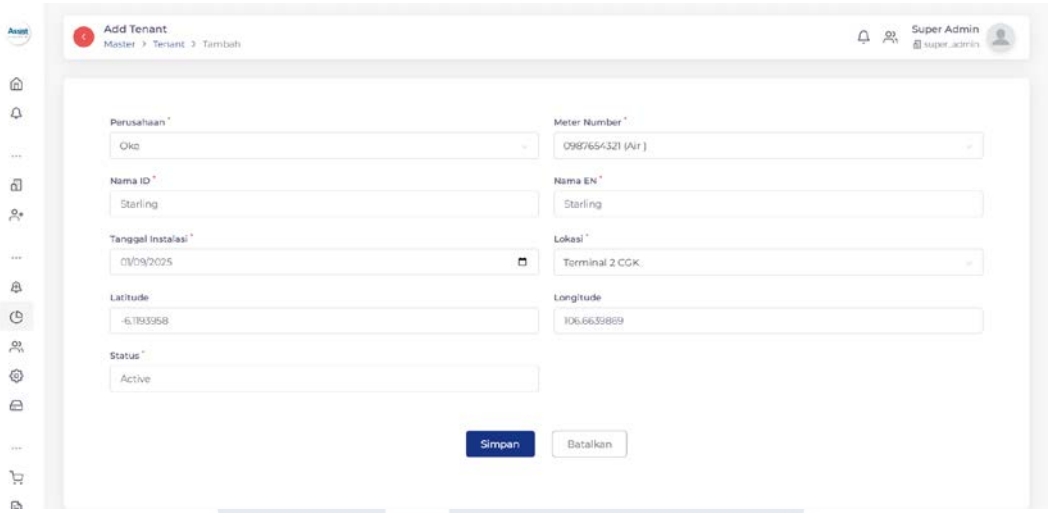
Pengembangan dilakukan menggunakan Laravel versi 9 untuk sisi admin, yang juga berfungsi sebagai *back-end* baik untuk admin maupun klien. Untuk memastikan bahwa seluruh proses pengelolaan utilitas berjalan lancar dan terorganisir, beberapa data master perlu ditambahkan pada tahap ini. Proses pengelolaan *tenant*, perangkat, dan visualisasi lokasi dan status layanan didasarkan pada data master ini.




Gambar 3.47 Data master *tenant*

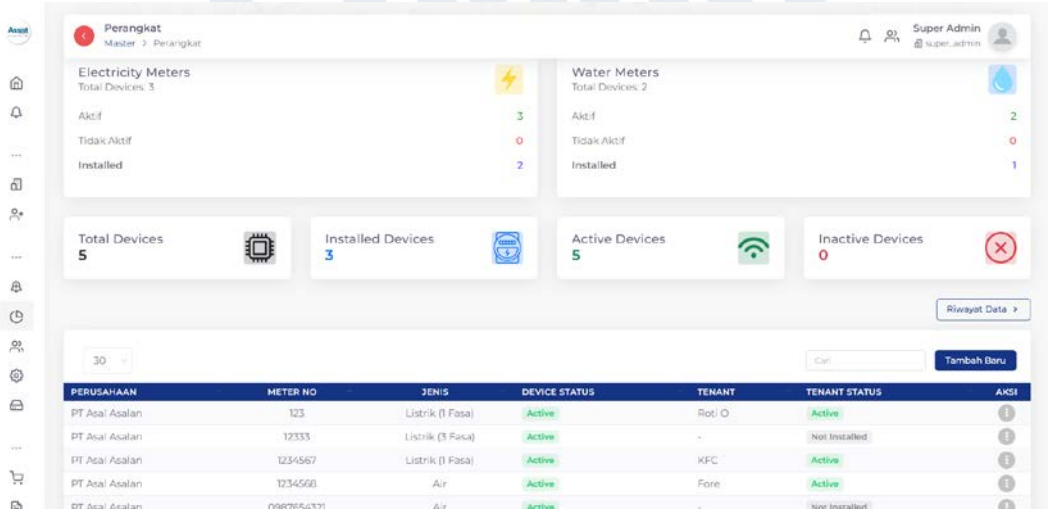
Dikarenakan tenant adalah pengguna listrik atau air, data master *tenant* sangat penting. Perusahaan induk *tenant*, nomor meteran, nama *tenant*, tanggal instalasi, lokasi, *latitude*, *longitude*, dan status aktif ditampilkan dalam tabel pada Gambar 3.47. Struktur

ini memberi admin kemampuan untuk melacak *tenant* dan menghubungkannya dengan perangkat yang digunakannya.



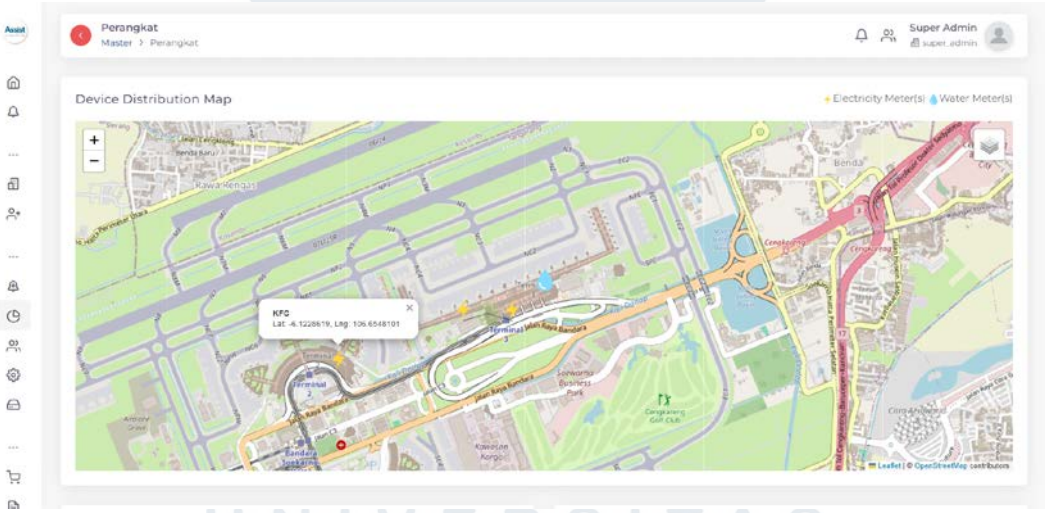

Gambar 3.48 *Form* input *tenant* baru

*Form* khusus disediakan untuk menambahkan data baru, seperti yang ditunjukkan pada Gambar 3.48. Nama *tenant* dibuat dalam dua bahasa, lalu *field* perusahaan dan nomor meter disajikan dalam bentuk *dropdown* untuk menghindari kesalahan input. Data tambahan seperti tanggal instalasi, lokasi, *latitude*, *longitude*, dan status memastikan bahwa *tenant* terdaftar dengan benar.




Gambar 3. 49 Data master perangkat

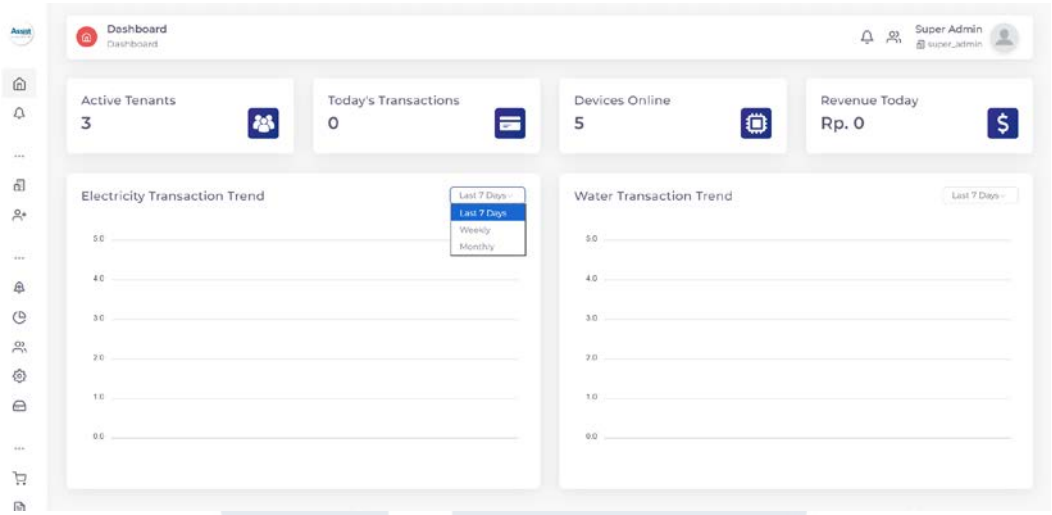
Selain itu, data master perangkat dikembangkan, seperti yang ditunjukkan pada Gambar 3.49. Perangkat meteran listrik dan air yang akan dipasang oleh *tenant* dicatat dalam data ini. Di bagian atas, widget menunjukkan jumlah perangkat total, serta perangkat yang aktif, nonaktif, dan yang sudah terpasang. Perangkat tidak harus sudah terinstal jika statusnya aktif; sebaliknya, itu menunjukkan bahwa perangkat siap digunakan. Tabel perangkat lengkap terletak di bawahnya, yang mengandung data *tenant* yang diambil dari tabel master *tenant*. Tampilan disediakan dengan struktur yang sama dengan *form tenant* untuk menambahkan perangkat baru. Parameter yang digunakan termasuk nomor perangkat, perusahaan penyedia layanan, jenis perangkat (listrik atau air), dan statusnya.




Gambar 3.50 Visualisasi lokasi *tenant* dan layanannya

Selain itu, sistem memungkinkan visualisasi peta, seperti yang ditunjukkan pada Gambar 3.50, melalui *library* Leaflet. Dengan menggunakan data *latitude* dan *longitude tenant* yang telah dimasukkan sebelumnya, visualisasi ini memungkinkan admin untuk melihat secara langsung di mana setiap *tenant* berada. Agar lebih mudah membedakan, simbol listrik dan tetesan air ditambahkan ke ikon peta. Dengan peta ini, pengelolaan utilitas

berbasis lokasi dan tidak hanya berbasis tabel menjadi lebih mudah untuk dipantau di lapangan.




Gambar 3.51 Tampilan *dashboard* admin

Gambar 3.51 menunjukkan *dashboard* admin yang menampilkan informasi seperti pendapatan harian, jumlah *tenant* aktif, transaksi harian, dan perangkat yang aktif. Grafik transaksi listrik dan air ditampilkan secara terpisah dan dapat difilter berdasarkan hari, minggu, atau bulan. Ini memudahkan manajemen untuk mengidentifikasi tren transaksi.

#### 3.3.1.15 Bus Management System – Menambah logika validasi tiket bus

*Bus Management System* adalah proyek berikutnya yang menggunakan *stack* utama Laravel, Vue.js, dan CodeIgniter. Sistem ini sudah cukup lama berjalan, jadi perlu beberapa penyesuaian tampilan dan logika untuk membuatnya sesuai dengan operasional

terbaru. Salah satu penyesuaian adalah penambahan logika validasi tiket yang berkaitan dengan proses *boarding* penumpang.


Gambar 3.52 Tampilan validasi tiket

Gambar 3.52 menunjukkan tampilan fitur validasi tiket yang sudah tersedia dalam sistem. Validasi dapat dilakukan dengan memindai kode QR pada tiket bus penumpang atau memasukkan kode tiket secara manual melalui kotak input. Meskipun fitur ini sudah berfungsi, pengecekan batas waktu validasi belum diterapkan, yang berarti penyalahgunaan tiket masih dapat terjadi. Penyesuaian ini juga memasukkan efek suara dan getaran untuk menunjukkan validasi berhasil atau gagal, dan untuk membuat notifikasi gagal lebih jelas bagi petugas.

```php
if ($ticketData->status_code == 2 && $boardingExpired >= $currentTime) {
    $this->db->update(
        't_trx_booking_detail',
        array('status' => 3, 'updated_by' => $createdBy, 'updated_on' => $currentTime),
        array('ticket_code' => $ticketData->ticket_code)
    );
    $res = array(
        'code'    => 0,
        'message' => 'Sukses, tiket anda sudah divalidasi',
        'data'    => $ticketData
    );
}
```

Gambar 3.53 Kode logika validasi

Untuk memastikan bahwa tiket hanya dapat divalidasi jika statusnya sudah dibayar dan waktu *boarding* belum melewati batas waktu yang ditentukan, ada logika tambahan yang diperlukan. Potongan kode yang menangani kondisi tersebut ditunjukkan pada Gambar 3.53. Setelah memeriksa apakah waktu *boarding* masih belum habis, sistem memeriksa status tiket, yang menunjukkan bahwa telah dibayar. Sistem memperbarui status tiket menjadi 3, jika kedua kondisi terpenuhi. Ini menunjukkan bahwa penumpang sudah masuk proses *boarding*.

```php
$cekReffNo = $this->trans->reffNumberValidation(req: $req);
if($cekReffNo){
    $res = array(
        'code'    => 400,
        'message' => 'Reff Number sudah pernah digunakan',
        'data'    => null
    );
}
else {
```

Gambar 3.54 Fungsi validasi *reff number*

Selain itu, angka reff juga ada validasinya, seperti yang ditunjukkan pada Gambar 3.54. Kode Reff number digunakan untuk menyimpan identifikasi pembelian dari transaksi yang dilakukan oleh aplikasi lain yang bekerja sama dengan penyedia tiket bus. Untuk menjaga integritas dan konsistensi data, pengujian ini

diperlukan untuk memastikan bahwa angka reff yang dimasukkan tidak duplikat dengan data yang sudah tersimpan.

#### 3.3.1.16 Bus Management System - Mengubah tampilan & tema warna

Setelah merger, pengelolaan sistem beralih dari APSD ke IASS. Untuk menyesuaikan dengan perubahan struktur ini, tampilan harus disesuaikan dengan identitas visual perusahaan yang baru. Sebagai bagian dari penyesuaian ini, elemen tampilan diperbarui dan tema warna disesuaikan dengan standar IASS.


Gambar 3.55 Tampilan login

Tampilan login setelah pembaruan ditunjukkan pada Gambar 3.55. Logo telah diganti, dan sebagai identitas sistem saat ini, ada keterangan “by IAS Support". Tampilan login juga dibuat responsif sehingga mudah diakses dari smartphone. Untuk mematuhi pedoman visual perusahaan, tema warna sistem juga diubah.

#### 3.3.1.17 Employee Lounge System - Memperbaiki *bug booking* & menyesuaikan syarat *booking*

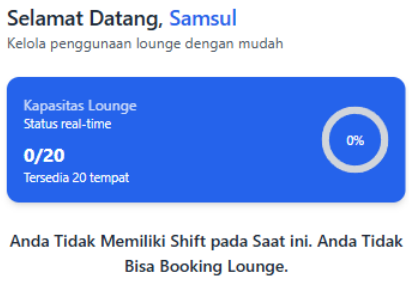
Dikarenakan laporan bahwa ada masalah dengan proses *booking lounge*, penyesuaian sistem dilakukan langsung. Masalah muncul bagi karyawan yang memiliki *shift* lintas hari, misalnya dari

jam 20.00 hingga 08.00. Saat waktu melewati pukul 00.00, formulir reservasi terus muncul meskipun karyawan telah mengirimkannya beberapa kali.

```
// shift datetime
if($shift){
    $shiftStart = Carbon::parse(time: "{$shift->start_date} {$shift->start_time}")->subMinutes(value: 60); //set start shift
    $shiftEnd = Carbon::parse(time: "{$shift->end_date} {$shift->end_time}"); //set end shift
}
// bookingan employee
$booking = Booking::where(column: 'employee_id', operator: $employee->id)
    ->whereBetween(DB::raw(value: "TIMESTAMP(booking_date, start_time)"), [$shiftStart, $shiftEnd])
```

Gambar 3.56 Kode *booking lounge*

Bagian kode yang menangani proses *booking lounge* ditunjukkan pada Gambar 3.56. Logika pengecekan waktu *shift* disesuaikan. Sebelum ini, sistem hanya membandingkan hari dan waktu secara terpisah. Akibatnya, ketika hari berganti, kondisi menjadi tidak valid. Untuk mengatasi hal ini, tanggal dan waktu pada data *shift* dan reservasi diubah menjadi tipe *datetime*. Ini memungkinkan sistem untuk memproses kondisi pergantian hari dengan benar.



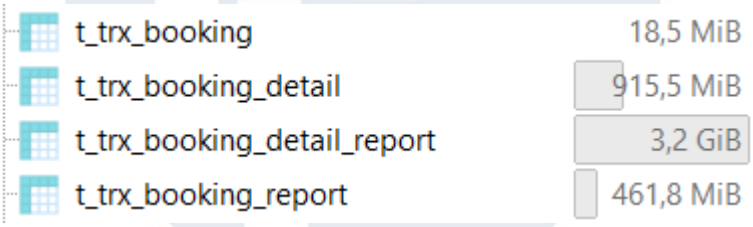

Gambar 3.57 Tampilan beranda

Gambar 3.57 menunjukkan tampilan beranda. Pesanan "Employee Lounge Sedang Penuh, Anda tidak dapat melakukan reservasi." ditampilkan di sistem ketika *lounge* dalam kondisi penuh. Pesan "Anda Tidak Memiliki Shift pada Saat Ini. Anda Tidak Bisa Booking Lounge." ditampilkan jika karyawan tidak memiliki *shift* pada hari tersebut. *Form* reservasi hanya ditampilkan ketika

karyawan memiliki *shift* aktif, *lounge* tersedia, dan belum pernah melakukan reservasi pada *shift* tersebut.

#### 3.3.1.18 Bus Management System - Membuat tabel *historical booking & booking detail*

Data harus disimpan terpisah dari data lama dan baru untuk menjaga sistem berjalan ringan. Solusinya adalah dengan membuat tabel *historical* yang berfungsi untuk menyimpan data *booking* dan detailnya. Tabel ini tidak lagi digunakan secara aktif dalam operasi harian. Agar proses transaksi dan penarikan data lebih cepat, data yang paling baru disimpan di tabel utama.



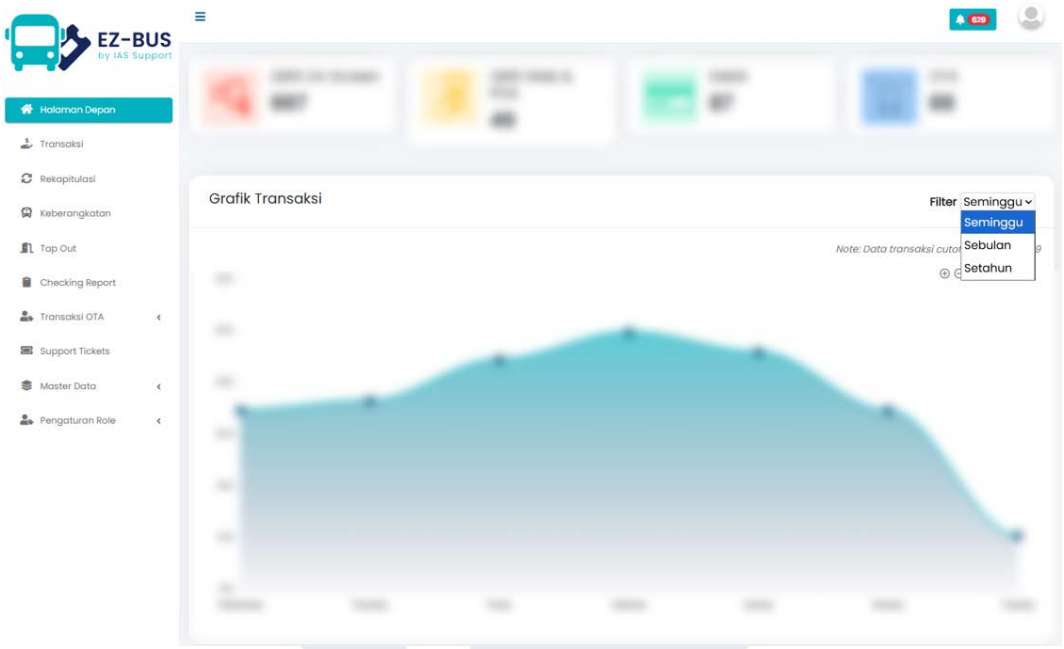

Gambar 3.58 Tabel biasa dan *historical*

Tabel *historical* dan tabel utama berbeda, seperti yang ditunjukkan pada Gambar 3.58. Tabel utama hanya menyimpan informasi dari satu bulan sebelumnya, sementara informasi dari waktu yang lebih lama ditransfer ke tabel *historical*. Untuk membedakan tabel *historical*, akhiran nama '_report' digunakan. Dengan pemisahan ini, proses pertanyaan sistem menjadi jauh lebih efisien. Selain itu, saat sistem membutuhkan data lama, sistem dapat langsung mengambilnya dari tabel *historical* tersebut.

#### 3.3.1.19 Bus Management System - Memperbarui dashboard

*Dashboard Bus Management System* harus diperbarui untuk memenuhi kebutuhan operasional terbaru. Untuk membuat informasi lebih mudah dibaca oleh admin dari IASS dan PO bus, pemaparan diubah selain tampilan. Selain itu, proses penarikan data

telah dioptimalkan untuk mengurangi jumlah waktu yang dibutuhkan untuk memuat halaman.



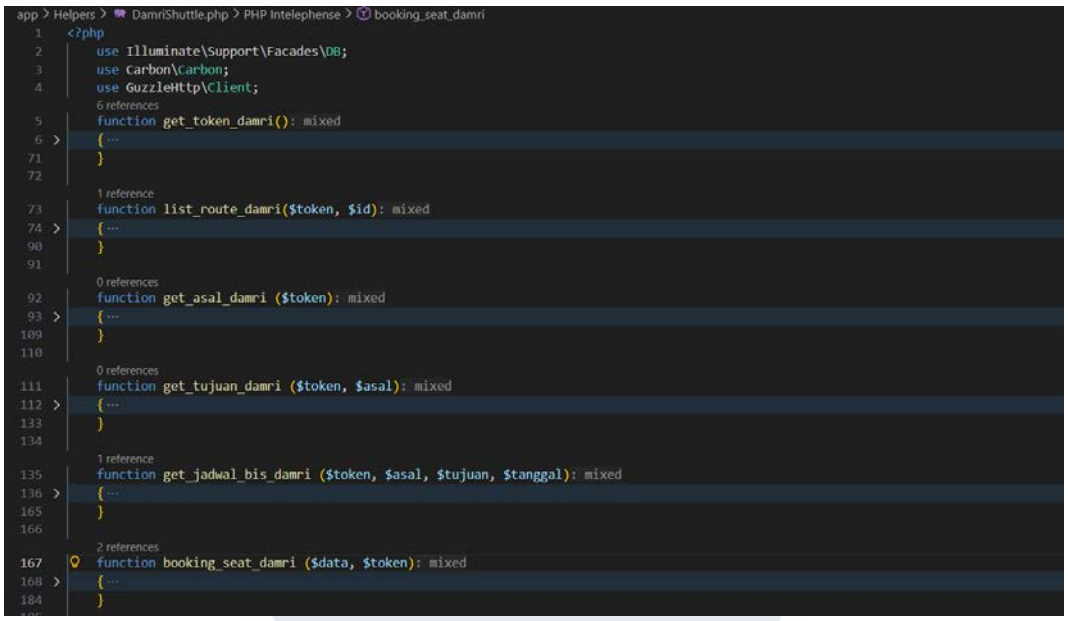

Gambar 3.59 Tampilan *dashboard* admin

Tampilan *dashboard* admin yang telah diperbarui ditunjukkan pada Gambar 3.59. Pada bagian atas, admin dapat dengan cepat memahami pola penggunaan sistem karena terlihat ringkasan jumlah transaksi berdasarkan metode pembayaran yang paling umum. Untuk mempermudah analisis tren, grafik total transaksi di bagian bawah dapat difilter berdasarkan rentang waktu tertentu, seperti satu minggu, satu bulan, atau satu tahun.

#### 3.3.1.20 Bus Management System - Mengintegrasi API Damri Shuttle

Integrasi dengan PO bus memerlukan koneksi langsung antara sistem Bus Management dan sistem yang dimiliki masing-masing PO. Proses integrasi dilakukan melalui API yang disediakan oleh pihak PO, disertai dokumentasi yang menjelaskan alur pemanggilan serta format data yang harus digunakan. Salah satu PO

yang diintegrasikan adalah Damri Shuttle, yang menyediakan akses API untuk kebutuhan pertukaran data operasional.



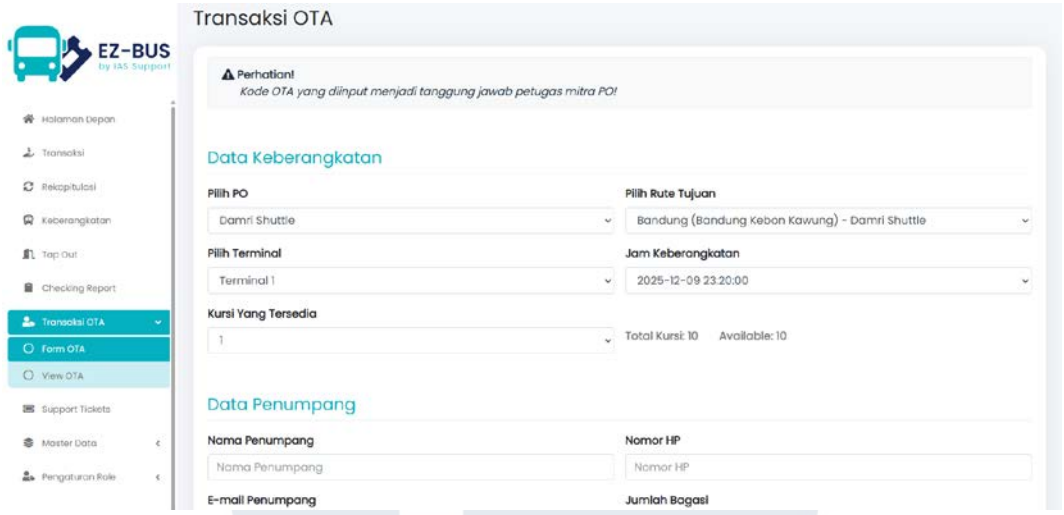

Gambar 3.60 Kode penggunaan API Damri Shuttle

Gambar 3.60 menampilkan kode yang digunakan untuk mengakses API Damri Shuttle. Kode ini ditempatkan pada bagian *helper* dan berperan sebagai penghubung untuk berbagai proses, termasuk memperoleh token login, mengambil data rute, jadwal, asal, dan tujuan. Selain menerima data, *helper* tersebut juga menangani permintaan pengiriman data seperti proses pemesanan, pembayaran, dan pembatalan, sehingga sistem Bus Management dapat berkomunikasi dengan sistem Damri Shuttle secara optimal.

#### 3.3.1.21 Bus Management System - Memperbaiki fitur OTA

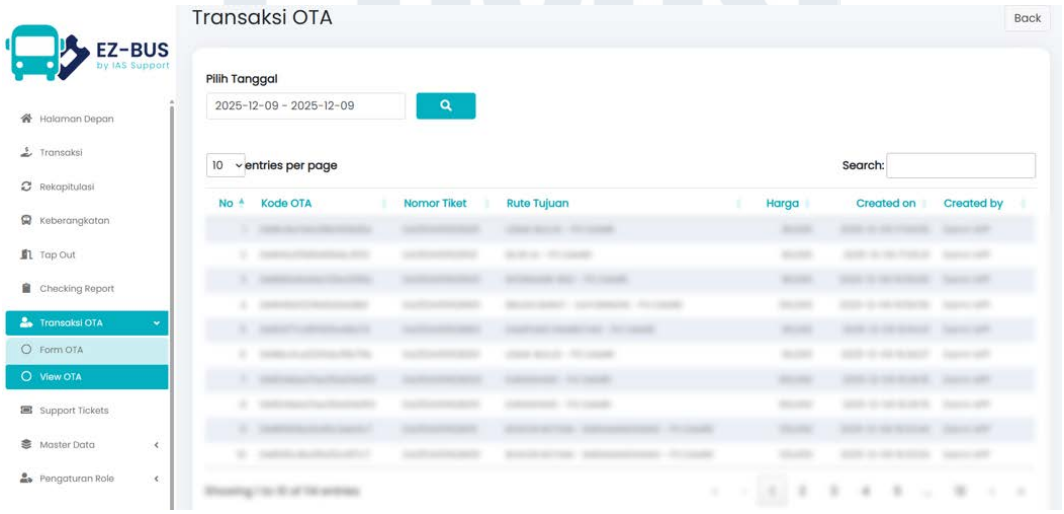
Untuk menerima transaksi tiket dari aplikasi pihak ketiga, fitur *Online Travel Agency* (OTA) diadakan. Sebelum ini, tampilan OTA masih menggabungkan tabel dan *form* dalam satu halaman, yang membuatnya kurang efisien saat digunakan. Pada pembaruan ini, halaman dibagi menjadi bagian terpisah. Ini dilakukan agar

proses input dan pengelolaan data lebih mudah dan tidak membingungkan pengguna.




Gambar 3.61 *Form* OTA

*Form* OTA yang digunakan untuk memasukkan data pembelian secara manual ditunjukkan pada Gambar 3.61. Nama pesan, rute, terminal keberangkatan, waktu keberangkatan, dan nomor kursi adalah bagian penting dari formulir. Pada bagian bawah juga tersedia *form* untuk mengisi detail penumpang, sehingga seluruh informasi dapat dicatat secara lengkap.




Gambar 3.62 Tabel OTA

Gambar 3.62 menampilkan tabel OTA yang digunakan untuk menyajikan seluruh data pembelian, baik yang berasal dari integrasi sistem maupun yang dimasukkan secara manual. Tabel ini memuat informasi seperti nomor referensi OTA, nomor tiket, rute tujuan, harga, tanggal transaksi, serta sumber aplikasi. Selain itu, tersedia fitur filter tanggal untuk memudahkan pencarian data berdasarkan periode tertentu.

#### 3.3.1.22 Bus Management System - Membuat & memperbaiki API

Pembuatan dan perbaikan API yang digunakan oleh pihak eksternal untuk berkolaborasi dengan Bus Management System juga dilakukan. Beberapa API perlu disesuaikan ulang karena perubahan kebutuhan, dan proses bisnis pada sistem. Selain itu, API lama yang sudah berjalan cukup lama ikut di-*review* kembali untuk memastikan logika, keamanan, dan efisiensinya tetap layak digunakan. Penyesuaian tersebut diperlukan agar proses integrasi dengan pihak eksternal dapat berjalan tanpa menimbulkan kesalahan.

```
0 references | 0 overrides
public function getListRoutePricebyPO(): void {
    // Validate JWT Token
    $this->jwt->validate();

    $this->form_validation->set_data(data: $_GET);
    $this->form_validation->set_rules(field: 'po_id', label: 'PO ID', rules: 'trim|required');

    $poId    = $this->input->get(index: 'po_id');
    $result     = $this->info->getListRoutePricebyPO(poId: $poId);

    if ($result) {
        $res = array(
            'code'    => 0,
            'message' => 'Sukses',
            'data'    => $result
        );
    }
    else {
        $res = array(
            'code'    => 111,
            'message' => 'Gagal, Data Informasi PO tidak tersedia',
            'data'    => null
        );
    }

    echo json_encode(value: $res);
}
```

Gambar 3.63 API harga rute bus

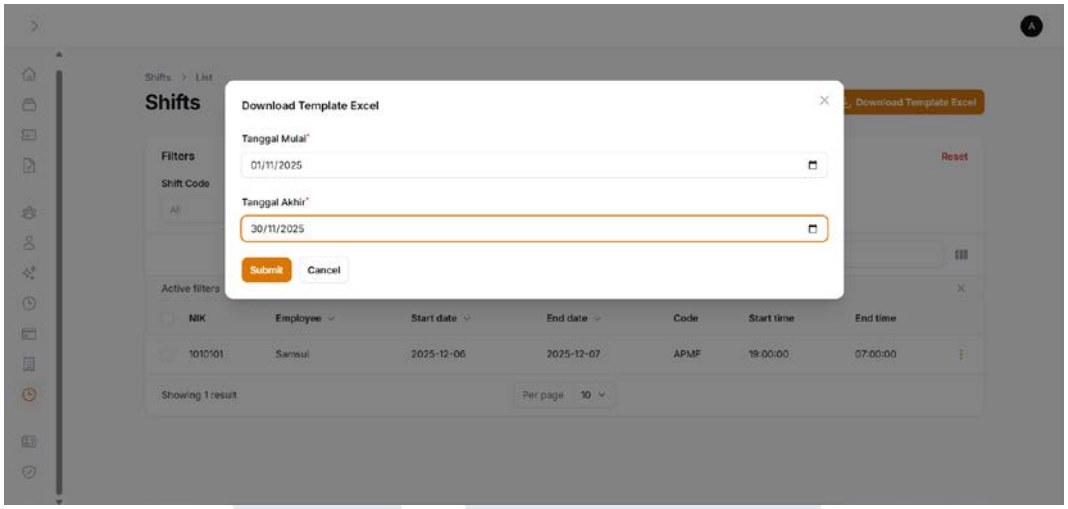
API yang telah dikembangkan untuk menampilkan informasi harga rute bus ditunjukkan pada Gambar 3.63. API ini membutuhkan parameter *po_id* untuk mengambil data yang sesuai. Nilai *po_id* tersebut digunakan untuk memfilter rute bus milik PO yang bersangkutan, kemudian sistem mengembalikan daftar rute beserta harga masing-masing. Dengan metode ini, pihak eksternal dapat mengakses data harga secara langsung untuk mendukung kebutuhan integrasi mereka.

Selain API tersebut, ada juga API lainnya seperti API daftar rute berdasarkan tipe bus, API sisa kursi bus untuk mengecek ketersediaan seat, API *checkpoint* bus, API penarik jadwal bus, dan beberapa API penunjang lainnya. Semua API ini diperbarui atau dibuat ulang sesuai kebutuhan agar kolaborasi antar sistem berjalan lebih stabil dan tidak menyusahkan siapa pun di kemudian hari.

#### 3.3.1.23 Employee Lounge System - Menyesuaikan *export template* excel *shift* dan *import*-nya

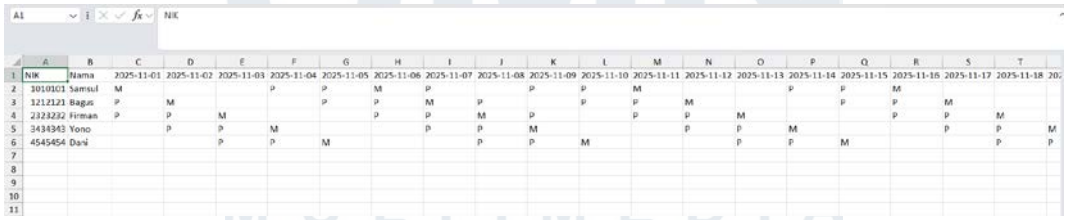
Dalam Employee Lounge Management muncul kebutuhan baru berupa penyesuaian format untuk proses import *shift*. Format sebelumnya tidak selaras dengan struktur Excel yang biasanya digunakan oleh tim pengelola *shift*, sehingga perlu disusun ulang agar lebih cocok dan mudah diisi. Penyesuaian ini juga bertujuan

mengurangi kesalahan input yang kerap terjadi akibat ketidaksesuaian kolom.




Gambar 3.64 *Download template* excel

Gambar 3.64 menunjukkan modal yang muncul setelah tombol ‘Download Template Excel’ pada modul shift ditekan. Modul ini menampilkan daftar *shift* untuk seluruh karyawan. Pada modal tersebut pengguna perlu mengisi tanggal mulai hingga tanggal akhir periode shift, biasanya dari awal bulan hingga akhir bulan. Setelah tanggal di-*submit*, sistem akan menghasilkan *file* Excel yang sudah siap.



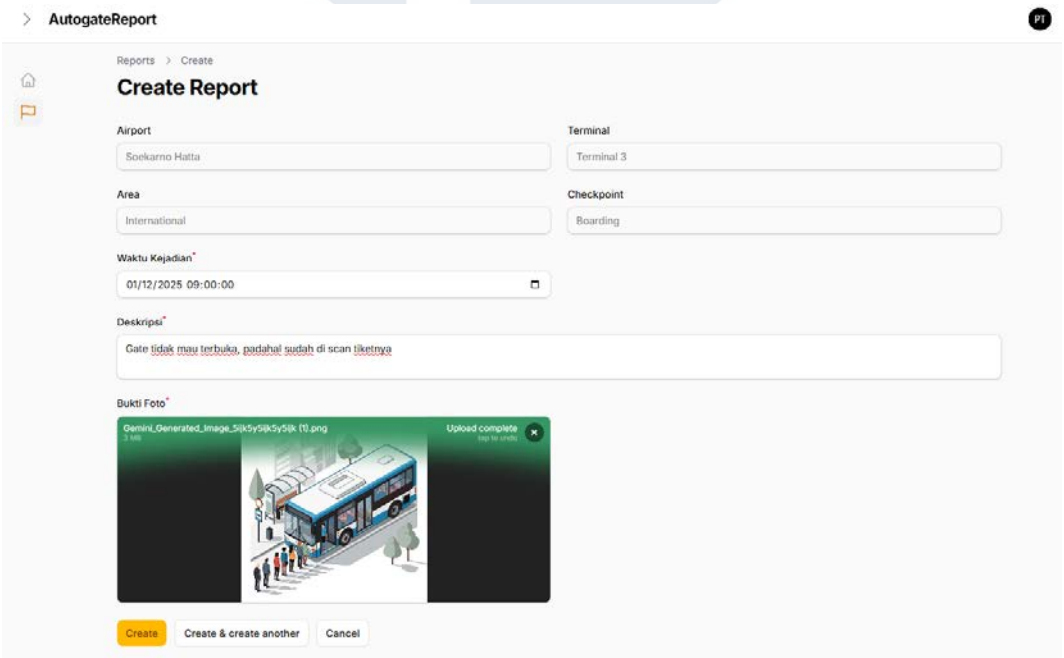

Gambar 3.65 Excel untuk di *import*

Gambar 3.65 memperlihatkan *file* Excel hasil *export* tadi. Hasil *export* hanya berisi header, sementara isinya diisi secara manual oleh bagian pengurus shift. Kolom NIK diisi dengan nomor induk karyawan, kolom Nama berisi nama karyawannya, dan kolom tanggal berfungsi sebagai header harian sesuai dengan *range* yang di

*export* sebelumnya. Jika suatu sel berisi kode *shift*, misalnya ‘P’ untuk shift pukul 07.00 hingga 19.00, maka informasi tersebut akan diproses saat *import* dilakukan. Jika sel dibiarkan kosong, hari itu dianggap libur. Untuk melakukan *import*, pengguna hanya perlu menekan tombol ‘Import Excel’, memilih *file*, lalu mengirimkannya.

#### 3.3.1.24 Autogate Report System – Membuat sistem

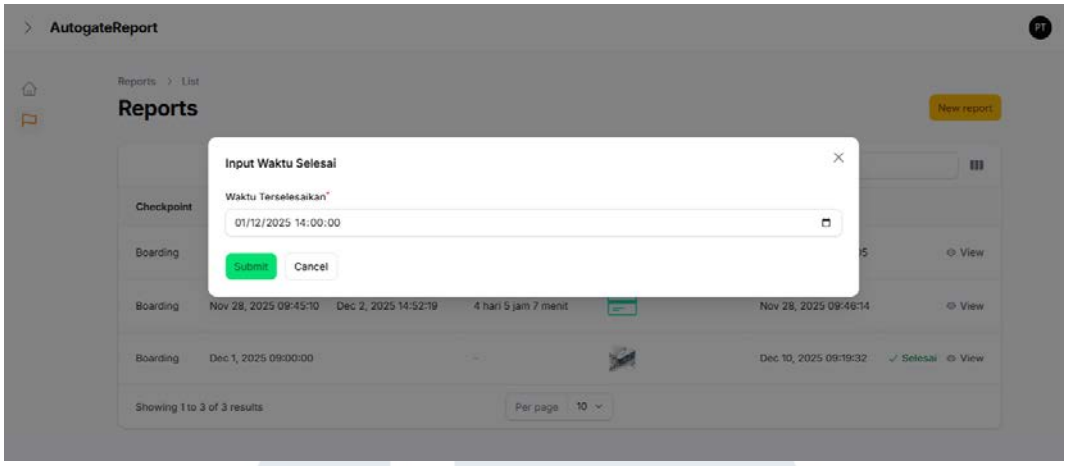
Projek terakhir sekaligus yang paling singkat adalah pembuatan sistem pelaporan untuk kendala yang terjadi pada perangkat *autogate*. Sistem ini dibuat menggunakan Filament 4. Kebutuhan ini muncul karena adanya pemasangan *autogate* baru yang di area *check-in*, *security checkpoint* dan *boarding* pesawat. Karena perangkatnya masih baru, potensi masalah tentu cukup tinggi, sehingga dibutuhkan sistem pelaporan yang ringkas dan mudah digunakan oleh petugas di lapangan.




Gambar 3.66 *Form* pelaporan

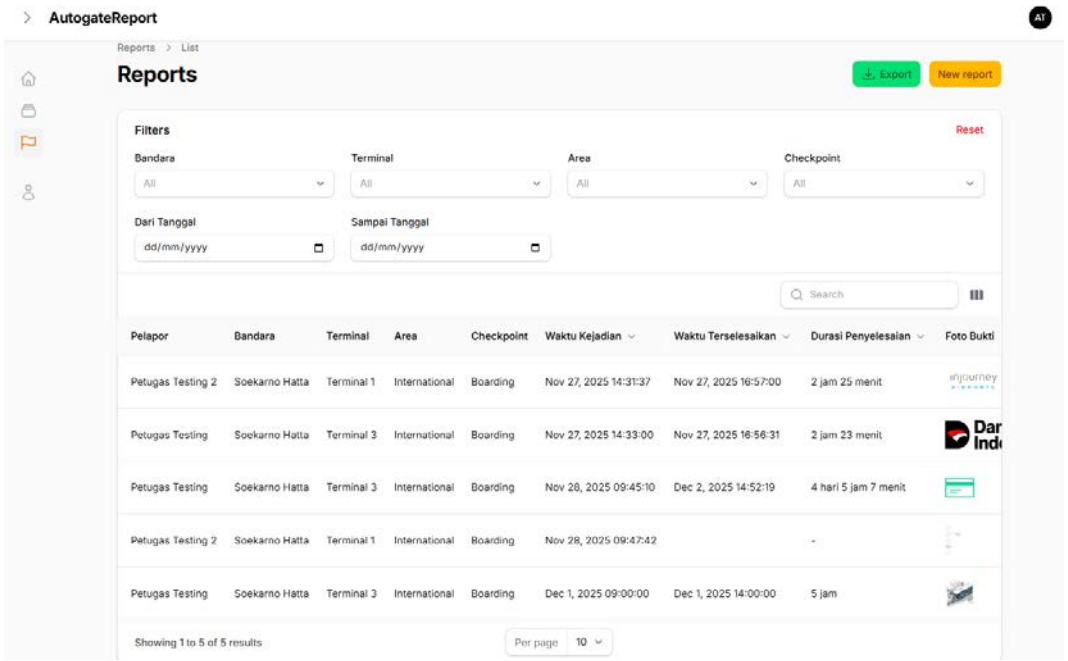
Gambar 3.66 menampilkan *form* pelaporan *autogate* yang digunakan petugas untuk mencatat kendala yang terjadi. *Field* seperti *Airport*, Terminal, Area, dan *Checkpoint* sudah otomatis

terisi sesuai dengan data penempatan petugas ketika akun mereka dibuat. Untuk mencatat kapan masalah mulai terjadi, dapat dilakukan dengan mengisi *field* waktu kejadian. Penjelasan detail masalah bisa dilakukan di kolom deskripsi, sementara bukti foto dengan batas maksimal 5 MB berfungsi sebagai bukti pendukung.




Gambar 3.67 Masalah terselesaikan

Petugas bisa mengubah status laporan bahwa masalah sudah selesai melalui tombol ‘Selesai’, seperti terlihat pada Gambar 3.67. Saat tombol tersebut diklik, modal yang meminta input waktu penyelesaian masalah akan muncul. Setelah diisi dan di-*submit*, durasi penanganan dapat dihitung sesuai dengan selisih waktu penyelesaian dengan waktu kejadian.




Gambar 3.68 *View* sisi admin

Gambar 3.68 menunjukkan tampilan pelaporan dari sisi admin. Admin dapat melihat semua laporan yang masuk dari petugas, lengkap dengan nama pelapor dan lokasi tugasnya. Informasi yang ditampilkan mencakup waktu kejadian, waktu penyelesaian (jika sudah diisi), durasi penyelesaian, foto bukti, dan tanggal pelaporan. Filter berdasarkan lokasi atau tanggal laporan juga bisa dilakukan pada daftar laporan untuk memudahkan *monitoring* dan evaluasi.

### 3.3.2 Kendala yang Ditemukan

Selama kegiatan kerja magang, terdapat beberapa kendala baik secara teknis maupun non-teknis yang menghambat proses penyelesaian proyek. Kendalanya yaitu:

1. *Transfer knowledge* yang kurang mendalam. Penyampaian materi atau pengenalan proyek hanya mencakup gambaran umum tanpa penjelasan yang lebih mendalam. Penjelasan tidak diberikan secara rinci mengenai detail teknis maupun logika bisnis yang kompleks dari sistem yang sedang dikembangkan. Hal ini mengakibatkan waktu yang dibutuhkan untuk memahami konteks masalah dan alur kerja sistem menjadi lebih lama.
2. Tidak adanya penggunaan sistem manajemen proyek. Pemberian tugas dan instruksi pengerjaan dilakukan sepenuhnya secara lisan tanpa penugasan tertulis yang terpusat. Kondisi ini menyulitkan proses pelacakan pekerjaan yang sedang dan harus dikerjakan, terutama ketika beban kerja berasal dari beberapa proyek yang berjalan bersamaan, serta pemantauan perkembangan tugas dan *deadline* yang pasti untuk setiap fitur yang dikerjakan.
3. Tidak adanya standarisasi dalam pengembangan. Proses pengembangan sistem di lingkungan kerja belum memiliki standarisasi yang baku. Hal ini mencakup gaya penulisan kode, penamaan variabel, struktur folder, dan penggunaan *library* dan *package*. Ketidakhadiran standar ini berdampak

pada inkonsistensi struktur kode, sehingga mempersulit dalam membaca, memahami, maupun memperbaiki kode yang sudah ada.

### 3.3.3 Solusi atas Kendala yang Ditemukan

Untuk mengatasi berbagai kendala sebelumnya, diterapkan beberapa solusi dan upaya yang digunakan guna memastikan kelancaran alur kerja dan menjaga kualitas hasil pengembangan sistem. Solusi-solusi tersebut adalah sebagai berikut:

1. Aktif bertanya dan melakukan eksplorasi mandiri. Komunikasi intensif dilakukan dengan cara aktif bertanya kepada pembimbing lapangan saat menemukan ketidakjelasan terkait modul, alur sistem, maupun logika bisnis. Selain itu, dilakukan permintaan akses terhadap dokumentasi atau arsip proyek yang relevan, serta eksplorasi mandiri dari berbagai sumber referensi. Hal ini untuk membuat pemahaman terhadap projek lebih terarah dan mengurangi potensi kesalahan dalam proses pengembangan.
2. Mencatat penugasan secara mandiri dan pelaporan berkala. Setiap instruksi dan pembagian tugas yang disampaikan secara lisan dicatat dalam catatan pribadi untuk menghindari adanya instruksi yang terlewat. Selanjutnya, laporan progres pekerjaan disampaikan secara terus-menerus kepada pembimbing lapangan untuk memastikan setiap tugas terpantau dan sesuai dengan ekspektasi waktu penyelesaian. Hal ini untuk menjaga transparansi dan memberikan gambaran yang lebih jelas mengenai progres pekerjaan, meskipun tidak tersedia sistem manajemen proyek formal.
3. Menganalisis dan menyesuaikan dengan kode yang sudah ada. Standar kode dapat dijaga dengan mempelajari struktur, pola, dan gaya penulisan yang sudah diterapkan pada bagian proyek sebelumnya. Dengan memahami karakteristik kode yang telah ada, penulisan kode baru yang dikembangkan memiliki pola, dan gaya penulisan yang serupa dengan kode lama, sehingga konsistensi sistem tetap terjaga meskipun tanpa dokumen standar tertulis. Hal ini membantu menjaga keseragaman proyek dan mempermudah proses pemeliharaan kode di tahap berikutnya.