BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani kegiatan magang di PT Bumi Serpong Damai Tbk, posisi yang diemban adalah sebagai Web and Application Developer pada divisi pengembangan aplikasi internal. Penempatan ini disesuaikan dengan latar belakang pendidikan di bidang sistem informasi, khususnya pada pengembangan aplikasi web dan mobile. Fokus utama selama masa magang adalah menangani berbagai bug (kesalahan sistem) yang muncul pada aplikasi, baik dari sisi fungsionalitas maupun antarmuka, serta mengembangkan fitur tambahan berdasarkan kebutuhan pengguna.

Dalam struktur organisasi tim, posisi berada di bawah tim support dan maintenance yang memiliki tanggung jawab utama untuk memastikan seluruh sistem internal berjalan secara stabil dan efisien. Tim ini terdiri atas 12 orang anggota, yang mencakup pengembang aplikasi, pengembang mobile, Quality Assurance (QA), serta koordinator tim. Setiap anggota memiliki peran spesifik dan saling melengkapi dalam mendukung proses pengembangan dan pemeliharaan aplikasi yang digunakan oleh berbagai divisi di perusahaan.

Setiap hari kerja dimulai dengan pembagian tugas yang diarahkan langsung oleh koordinator tim. Tugas yang diberikan bersifat variatif, mulai dari memperbaiki bug pada halaman tertentu, memperbarui tampilan antarmuka, hingga menyesuaikan fitur agar lebih sesuai dengan alur bisnis perusahaan. Selain itu, beberapa permintaan perubahan atau pengembangan fitur tambahan juga berasal langsung dari feedback pengguna akhir melalui helpdesk internal.

Untuk mendukung kelancaran penyelesaian tugas, koordinasi aktif dengan seluruh anggota tim dilakukan secara rutin. Komunikasi berlangsung melalui berbagai saluran internal seperti Microsoft Teams, WhatsApp Group, dan pertemuan

informal di ruang kerja. Diskusi ini mencakup pembahasan teknis, klarifikasi kebutuhan pengguna, pengecekan alur logika aplikasi, hingga berbagi solusi atas kendala yang dihadapi dalam proses pengembangan. Pendekatan kolaboratif ini terbukti sangat membantu dalam mempercepat penyelesaian tugas dan mencegah kesalahan yang berulang. Koordinasi yang dilakukan tidak hanya terbatas pada sesama pengembang, tetapi juga melibatkan tim QA dalam proses validasi dan pengujian aplikasi. QA memiliki peran penting dalam memastikan bahwa setiap pembaruan atau perbaikan sistem telah memenuhi standar kualitas yang ditetapkan. Diskusi dengan QA menjadi bagian penting sebelum aplikasi dipindahkan ke tahap produksi, guna memastikan semua fitur berfungsi dengan baik tanpa menimbulkan bug baru yang tidak terdeteksi.

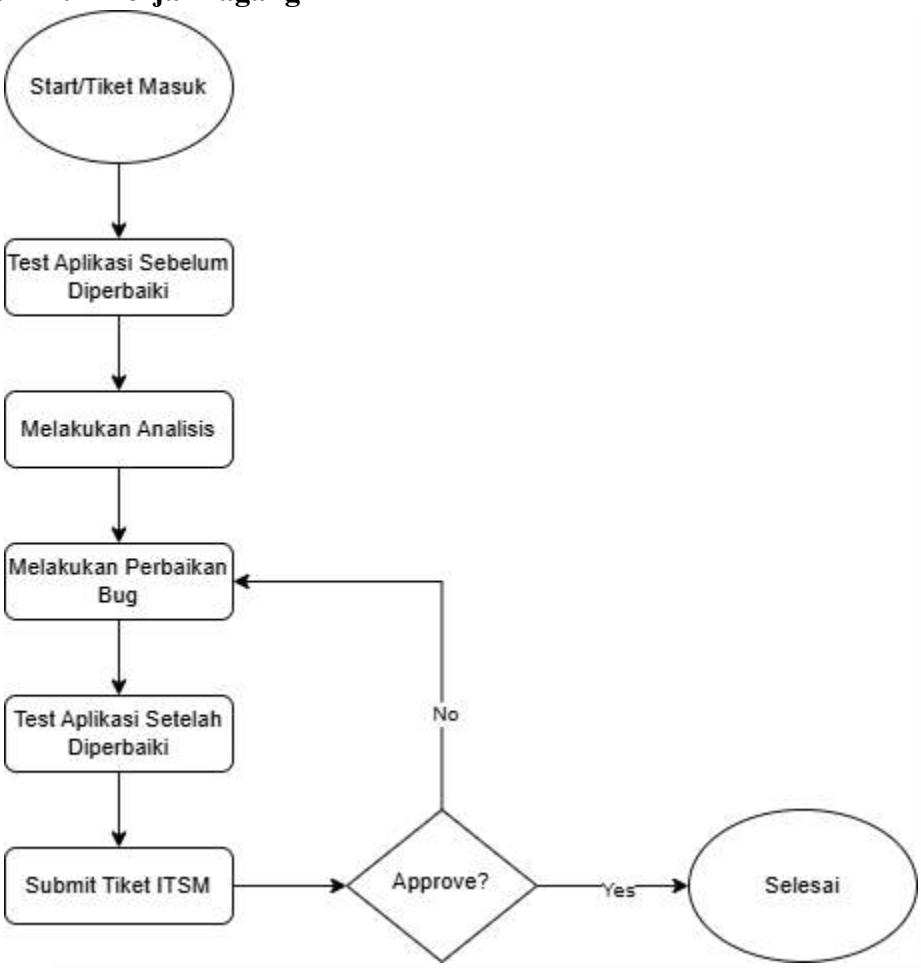
Setiap perubahan yang dilakukan terhadap sistem wajib terdokumentasi secara jelas dan rinci. Dokumentasi meliputi identifikasi permasalahan, langkah-langkah perbaikan atau pengembangan, hasil pengujian, serta konfirmasi dari pihak terkait. Dokumentasi ini dimasukkan ke dalam sistem pelacakan tugas internal yang digunakan untuk keperluan pemantauan progres dan audit. Proses ini memberikan pengalaman langsung dalam menerapkan praktik terbaik dalam pengelolaan proyek teknologi informasi.

Keberadaan dalam tim support memberikan banyak pengalaman dalam memahami alur kerja pengembangan dan pemeliharaan aplikasi secara menyeluruh. Proses magang tidak hanya memperdalam kemampuan teknis, tetapi juga membentuk pemahaman terhadap pentingnya kolaborasi tim, komunikasi profesional, serta kemampuan mengatur prioritas pekerjaan. Setiap anggota tim memberikan kontribusi dan saling mendukung demi mencapai hasil kerja yang optimal. Selain tugas teknis, keterlibatan dalam diskusi tim juga memberikan wawasan tentang strategi kerja, perencanaan waktu, serta cara menyampaikan solusi teknis dalam bahasa yang dapat dimengerti oleh rekan kerja dari latar belakang non-teknis. Proses ini melatih kemampuan berpikir kritis dan adaptif dalam lingkungan kerja profesional yang dinamis.

Pengalaman berinteraksi langsung dengan tim QA, pengembang lain, dan pihak pengguna memperkuat pemahaman bahwa kerja tim menjadi elemen utama dalam keberhasilan suatu proyek pengembangan aplikasi. Ketika komunikasi berjalan efektif, setiap tantangan teknis dapat diselesaikan dengan lebih efisien. Selain itu, rasa tanggung jawab terhadap hasil kerja meningkat karena setiap tugas yang dijalankan berdampak langsung pada kenyamanan dan produktivitas pengguna aplikasi.

Secara keseluruhan, peran dalam struktur tim support selama masa magang memberikan pengalaman yang menyeluruh, baik dari sisi teknis maupun manajerial. Interaksi intensif dengan anggota tim, koordinasi lintas fungsi, serta pelibatan dalam berbagai proses pengembangan menjadikan masa magang ini sebagai periode pembelajaran yang sangat berharga. Pengalaman ini tidak hanya meningkatkan kompetensi teknis dalam pengembangan aplikasi, tetapi juga memperkuat keterampilan komunikasi, kerja sama, dan problem solving yang akan sangat bermanfaat di dunia kerja profesional.

3.1.1 Alur Kerja Magang



Gambar 3.1 Alur Proses Kerja

Flowchart ini menggambarkan tahapan proses kerja dalam menangani bug yang dilaporkan melalui tiket masuk ke tim developer. Proses ini merupakan bagian dari aktivitas support yang dijalankan oleh Web and Application Developer di PT Bumi Serpong Damai Tbk.

1. Start / Tiket Masuk

   Proses dimulai ketika ada tiket laporan masuk yang berisi informasi mengenai adanya bug atau kesalahan pada aplikasi. Tiket ini biasanya diterima melalui sistem manajemen tiket seperti ITSM (IT Service Management).

2. Test Aplikasi Sebelum Diperbaiki

Langkah awal adalah melakukan pengujian terhadap aplikasi sesuai dengan laporan bug yang diterima. Tujuannya adalah untuk memverifikasi kebenaran laporan dan memahami kondisi aplikasi sebelum dilakukan perbaikan.

3. Melakukan Analisis

Setelah bug berhasil direproduksi, langkah selanjutnya adalah menganalisis penyebab kesalahan. Analisis ini mencakup penelusuran kode, pengecekan log error, serta pemahaman alur sistem untuk menemukan akar permasalahan.

4. Melakukan Perbaikan Bug

Setelah penyebab diketahui, dilakukan proses perbaikan (debugging) terhadap bagian kode atau konfigurasi sistem yang menyebabkan bug. Tahap ini dilakukan dengan hati-hati agar tidak menimbulkan error baru.

5. Test Aplikasi Setelah Diperbaiki

Setelah perbaikan dilakukan, aplikasi diuji kembali untuk memastikan bahwa bug sudah benar-benar terselesaikan dan sistem berjalan dengan normal tanpa efek samping (regression).

6. Submit Tiket ITSM

Jika perbaikan berhasil dan hasil pengujian memuaskan, maka tiket akan diperbarui atau ditutup melalui sistem ITSM. Di sini, status perbaikan akan dilaporkan sebagai “resolved” atau “closed”.

7. Approve?

Hasil pengujian akan ditinjau kembali. Jika bug dianggap belum terselesaikan atau masih ada masalah lain (jawaban: No), maka proses akan kembali ke tahap perbaikan bug untuk dilakukan perbaikan ulang. Jika sudah sesuai (jawaban: Yes), maka proses dilanjutkan.

8. Selesai

Proses penanganan bug dinyatakan selesai setelah tiket ditutup dan tidak ada lagi tindakan lanjutan yang diperlukan.

3.2 Tugas dan Uraian Kerja Magang

Tabel 3.1 Kegiatan dan Waktu Pengerjaan Kegiatan Selema Magang

Kegiatan	Waktu Pengerjaan
Learning OutSystem	24 Feb 2025 Sampai 28 Feb 2025
	3 Mar 2025 Sampai 7 Mar 2025
	24 Mar 2025 Sampai 28 Mar 2025
	26 Mei 2025 Sampai 30 Mei 2025
	2 Jun 2025 Sampai 5 Jun 2025
	16 Jun 2025 Sampai 20 Jun 2025
	23 Jun 2025 Sampai 30 Jun 2025
TASK 1 Filter Gedung Tidak Tampil Seluruhnya.	10 Mar 2025 Sampai 12 Mar 2025
	18 Mar 2025 Sampai 20 Mar 2025
TASK 2 Mengubah DropDown Biasa Menjadi DropDown Search di aplikasi web Shift HandOver pada Equipment.	7 Apr 2025 Sampai 9 Apr 2025
	14 Apr 2025 Sampai 15 Apr 2025
TASK 3 Select Kegiatan Page error ketika kembali ke halaman sebelumnya.	21 Apr 2025 Sampai 24 Apr 2025
TASK 4 Menambahkan Batas Maksimum Karakter pada Field Keterangan Gambar	28 Apr 2025 Sampai 29 Apr 2025
TASK 5 Penghilangan Tanda # pada Nomor Tiket di Halaman Request & Permit (Web & Mobile)	5 Mei 2025 Sampai 7 Mei 2025
TASK 6 Penambahan Wording pada Validasi Status Nonaktif ke Aktif (Request & Permit)	14 Mei 2025 Sampai 16 Mei 2025
	19 Mei 2025 Sampai 20 Mei 2025
TASK 7 Penyesuaian Wording Approval pada Mobile Apps untuk Akun BM & GS	10 Jun 2025 Sampai 13 jun 2025

3.2.1 Learning OutSystem

OutSystems merupakan platform pengembangan aplikasi berbasis *low-code* yang memfasilitasi proses pembuatan aplikasi dengan pendekatan visual dan komponen modular. Platform ini memungkinkan pengembangan aplikasi web maupun mobile secara lebih cepat dan efisien dibandingkan metode konvensional, serta mendukung integrasi dengan berbagai sistem eksternal. Dalam konteks kegiatan magang, pemahaman terhadap OutSystems menjadi hal fundamental

untuk menunjang pelaksanaan tugas-tugas teknis yang berkaitan dengan perbaikan dan pengembangan fitur aplikasi.

Proses pembelajaran OutSystems dilaksanakan secara mandiri (self-learning) dengan dukungan terbatas dari dokumentasi internal perusahaan. Tahapan pembelajaran dimulai dari pengenalan antarmuka pengguna (user interface), pemahaman terhadap struktur entitas data, hingga implementasi *logic flow* dan tindakan (*action*) dalam skenario aplikasi nyata. Materi pembelajaran diambil dari platform pelatihan resmi seperti *OutSystems Guided Paths*, komunitas daring, serta video tutorial yang tersebar di kanal edukatif. Proses ini dilakukan secara berkelanjutan sembari mengerjakan tugas-tugas riil yang diberikan oleh atasan pembimbing di perusahaan.

Selama proses belajar, beberapa tantangan teknis muncul, khususnya dalam memahami struktur *data model*, relasi antar entitas, serta mekanisme eksekusi *client-side* dan *server-side logic*. Ketidakterbiasaan terhadap pendekatan *low-code* menjadi hambatan awal, mengingat sebagian besar pengalaman sebelumnya berbasis *manual coding*. Untuk mengatasi hal tersebut, strategi pembelajaran adaptif diterapkan, yakni dengan langsung menerapkan teori yang dipelajari ke dalam penyelesaian bug dan modifikasi fitur pada proyek yang sedang berjalan.

3.2.1.1 Komponen Antarmuka Pengguna (UI) dalam OutSystems

OutSystems menyediakan berbagai komponen antarmuka pengguna (*User Interface*) yang dapat dimanfaatkan untuk membangun tampilan aplikasi web dan mobile secara efisien. Komponen UI dalam OutSystems bersifat modular dan dapat digunakan secara *drag-and-drop*, sehingga mempercepat proses desain serta meminimalkan kesalahan pengkodean. Setiap elemen dapat dikonfigurasi melalui properti yang telah disediakan, serta didukung dengan logika interaktif yang dapat dihubungkan dengan data maupun aksi pengguna.

Beberapa komponen UI yang sering digunakan antara lain Input Widget seperti *Input*, *Dropdown*, dan *Date Picker*, yang memungkinkan pengguna memasukkan data ke dalam sistem. Selain itu, terdapat komponen *Button* yang

berfungsi untuk mengeksekusi tindakan, *Table Records* dan *List* untuk menampilkan data secara terstruktur, serta *Popup* dan *Modal* yang digunakan untuk menyajikan informasi tambahan tanpa meninggalkan halaman utama. Komponen tersebut dapat dikustomisasi melalui *Style Classes* maupun CSS eksternal guna memenuhi kebutuhan desain perusahaan.

Dalam penerapannya selama kegiatan magang, pemanfaatan komponen UI menjadi hal yang krusial. Misalnya, pada penambahan fitur *dropdown searchable*, pemahaman terhadap struktur widget dan properti seperti Options, OnChange, dan Variable menjadi sangat penting. Demikian pula saat menangani bug pada tampilan halaman mobile, pemahaman terhadap hierarki layout dan penggunaan container seperti *Block* dan *Web Screen* menjadi faktor utama dalam penyelesaian tugas.

Dengan memahami struktur dan fungsi dari komponen UI yang tersedia di OutSystems, pengembangan antarmuka aplikasi dapat dilakukan secara lebih terarah, konsisten, dan sesuai dengan kebutuhan pengguna akhir. Hal ini juga memperkuat kemampuan analitis dalam merancang solusi antarmuka berbasis *low-code*, yang semakin dibutuhkan dalam lingkungan kerja modern.

3.2.1.2 Logika Aplikasi dalam OutSystems (Client dan Server Actions)

Dalam pengembangan aplikasi menggunakan OutSystems, logika aplikasi dikendalikan melalui dua jenis aksi utama, yaitu Client Action dan Server Action. Keduanya merupakan bagian penting dalam menentukan alur interaksi antara pengguna, antarmuka aplikasi, dan sistem backend. Pemahaman terhadap perbedaan dan fungsi masing-masing jenis aksi menjadi kunci dalam membangun aplikasi yang responsif, efisien, dan sesuai dengan kebutuhan pengguna.

Client Action digunakan untuk menjalankan logika yang diproses langsung di sisi pengguna (*client-side*), seperti validasi input, navigasi antar halaman, atau manipulasi tampilan yang tidak memerlukan komunikasi dengan server. Karena dijalankan di perangkat pengguna, aksi ini dapat meningkatkan kecepatan dan kenyamanan interaksi antarmuka. Di sisi lain, Server Action berfungsi untuk memproses logika yang berkaitan dengan database atau layanan eksternal. Aksi ini

akan dikirim ke server dan memerlukan waktu eksekusi tambahan karena adanya proses *request-response* melalui jaringan.

Selama kegiatan magang, penggunaan Client dan Server Actions menjadi elemen penting dalam menyelesaikan berbagai tugas. Sebagai contoh, ketika memperbaiki bug pada alur pengajuan lembur, diperlukan pemahaman tentang kapan suatu proses validasi cukup dijalankan di sisi klien, dan kapan harus melibatkan interaksi dengan database melalui Server Action. Dalam beberapa kasus, Server Action digunakan untuk mengambil data dari entitas tertentu, kemudian hasilnya ditampilkan secara dinamis melalui Aggregate dan diolah kembali menggunakan Client Action untuk memberikan umpan balik ke pengguna.

Dengan memahami konsep pemisahan tanggung jawab antara Client dan Server Actions, pengembangan aplikasi dapat dilakukan secara lebih efisien, mengurangi beban server yang tidak perlu, serta meningkatkan kecepatan respon aplikasi di sisi pengguna. Kemampuan ini juga memperkuat keterampilan teknis dalam merancang arsitektur aplikasi berbasis *low-code* yang scalable dan maintainable.

3.2.1.3 Pengelolaan Data Menggunakan Entity dan Aggregate dalam OutSystems

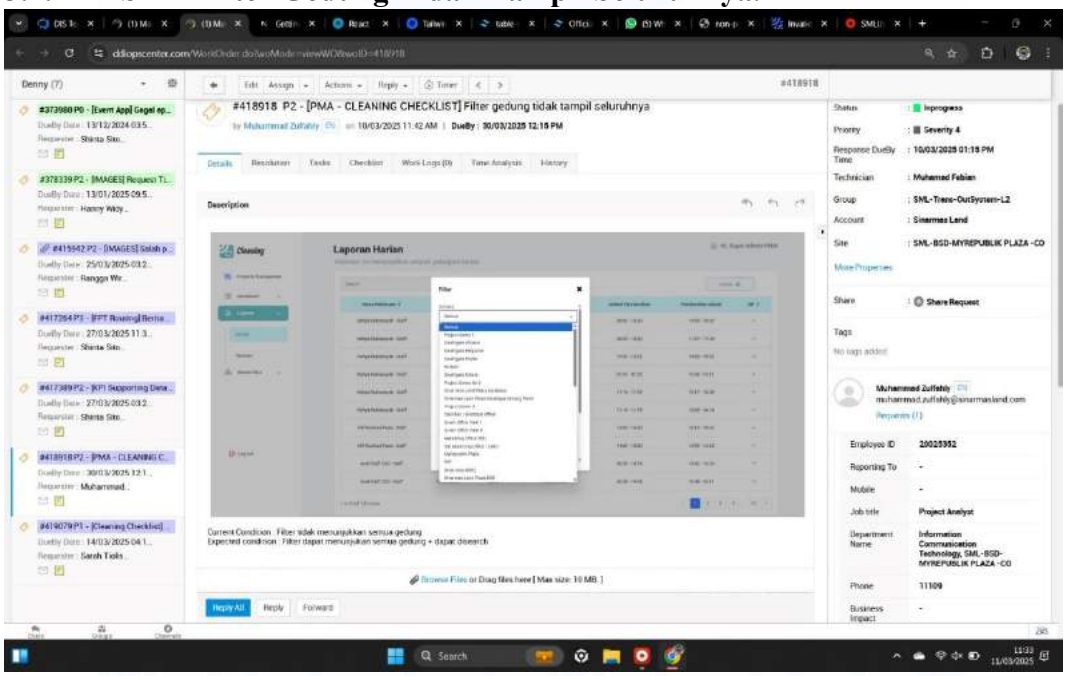
OutSystems menyediakan mekanisme pengelolaan data yang terstruktur dan terintegrasi melalui komponen Entity dan Aggregate. Entity merupakan representasi dari tabel dalam basis data yang digunakan untuk menyimpan dan mengelola informasi secara sistematis. Entity dapat bersifat *static* maupun *dynamic*, dan masing-masing memiliki atribut yang mendefinisikan jenis data yang disimpan. Penggunaan Entity memungkinkan pengembang untuk mendesain struktur data secara visual, serta mengatur relasi antar entitas seperti *one-to-many* atau *many-to-one* tanpa memerlukan skrip SQL manual.

Untuk mengambil dan memanipulasi data dari Entity, OutSystems menyediakan komponen Aggregate, yaitu alat kueri visual yang berfungsi seperti perintah SELECT dalam SQL. Aggregate memungkinkan pengguna untuk

melakukan *filtering*, *sorting*, *joining*, dan *projection* antar entitas dengan cara yang intuitif. Selain itu, hasil dari Aggregate dapat langsung dihubungkan dengan komponen antarmuka seperti tabel, list, atau formulir input, sehingga mempercepat integrasi data ke dalam tampilan aplikasi.

Selama pelaksanaan magang, pemahaman terhadap Entity dan Aggregate sangat krusial dalam menyelesaikan berbagai perbaikan bug dan pengembangan fitur. Contohnya, pada tugas penambahan fitur pencarian data berbasis dropdown, Aggregate digunakan untuk mengambil data dari entitas Equipment, kemudian ditampilkan dalam komponen Dropdown yang mendukung fitur pencarian. Di sisi lain, Entity juga dimodifikasi untuk menambahkan atribut baru yang dibutuhkan dalam validasi input pada modul tertentu. Proses ini melibatkan analisis struktur data yang ada serta penyesuaian logika aplikasi agar tetap konsisten dan tidak menimbulkan konflik data.

3.2.2 TASK 1 Filter Gedung Tidak Tampil Seluruhnya.

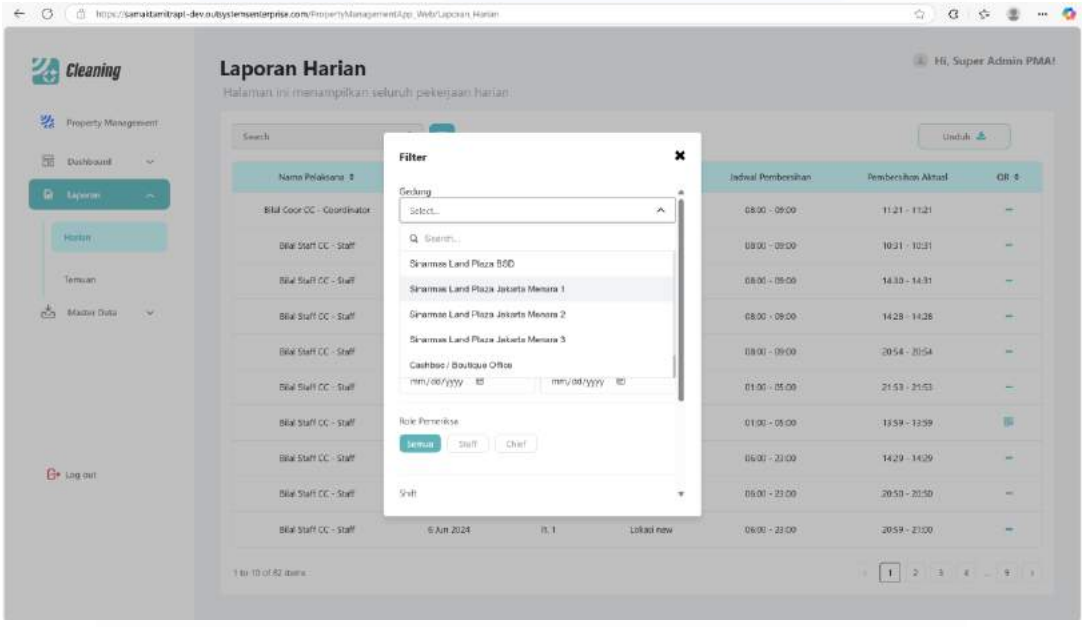


Gambar 3.2 Tiket PMA-CleaningChecklist

Merupakan dokumentasi awal dari tiket bug yang diterima terkait laporan harian pada modul Cleaning Checklist. Tiket ini berisi informasi tentang kendala filter gedung yang tidak berfungsi, menjadi referensi awal perbaikan fitur.

Deskripsi Task, Pada tugas pertama ini, saya mendapatkan tugas untuk mengembangkan sebuah fitur filter laporan harian yang digunakan oleh user untuk menyaring data pada halaman web aplikasi Property Management App (PMA). Fitur ini memungkinkan pengguna untuk memfilter data berdasarkan beberapa parameter seperti gedung, kategori, tower, lantai, lokasi spesifik, tanggal, dan role pemeriksa.

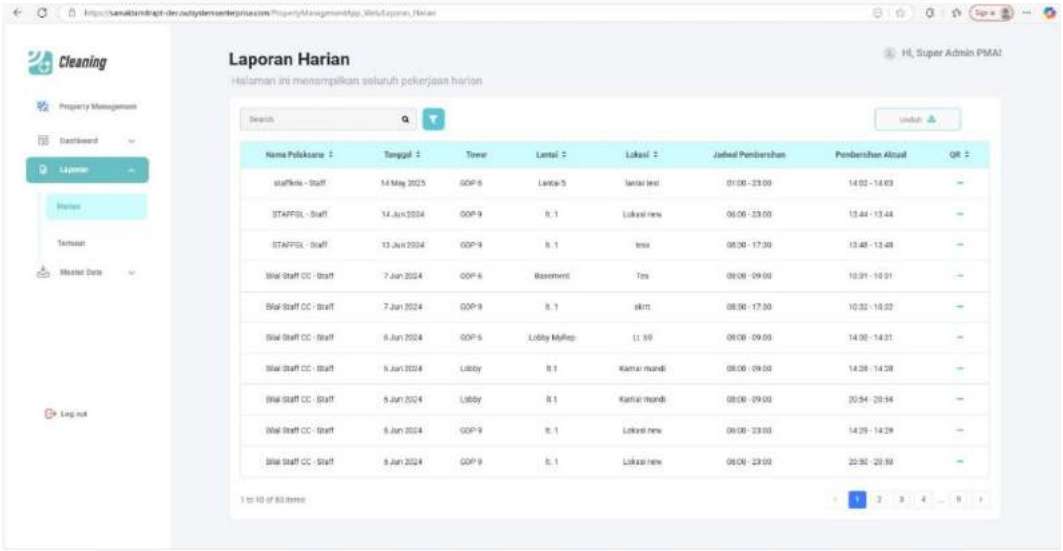
3.2.2.1 Analisis Aplikasi Website PMA-Cleaning Checklist



Gambar 3.3 Web Cleaning Checklist – Laporan Harian (Awal)

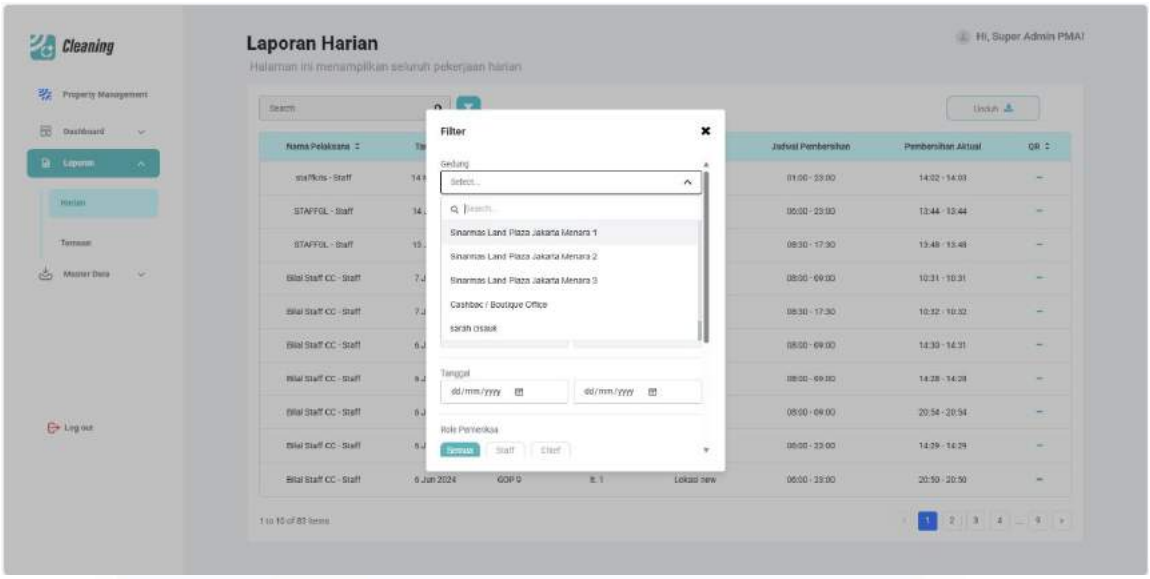
Gambar 3.3 menampilkan tampilan awal dari halaman laporan harian pada modul Cleaning Checklist sebelum fitur filter diterapkan. Semua data laporan ditampilkan sekaligus tanpa opsi untuk menyaring informasi berdasarkan parameter seperti gedung atau kategori. Akibatnya, pengguna harus menggulir secara manual untuk

menemukan data yang dibutuhkan. Hal ini menjadi kurang efisien, terutama ketika jumlah data yang ditampilkan sangat banyak. Dari gambar 3.3 terlihat bahwa antarmuka pengguna masih bersifat statis dan belum interaktif, sehingga memicu kebutuhan untuk ditingkatkan melalui penambahan fitur filter. Masalah ini kemudian menjadi fokus utama untuk diperbaiki dalam task pertama.Analisis Permasalahan, Sebelum fitur ini dibuat, halaman laporan harian hanya menampilkan seluruh data tanpa adanya kemampuan untuk menyaring data tertentu. Hal ini menyulitkan pengguna dalam mencari data spesifik, terutama ketika jumlah data yang ditampilkan sangat banyak. Oleh karena itu, dibutuhkan fitur filter yang responsif dan fleksibel.



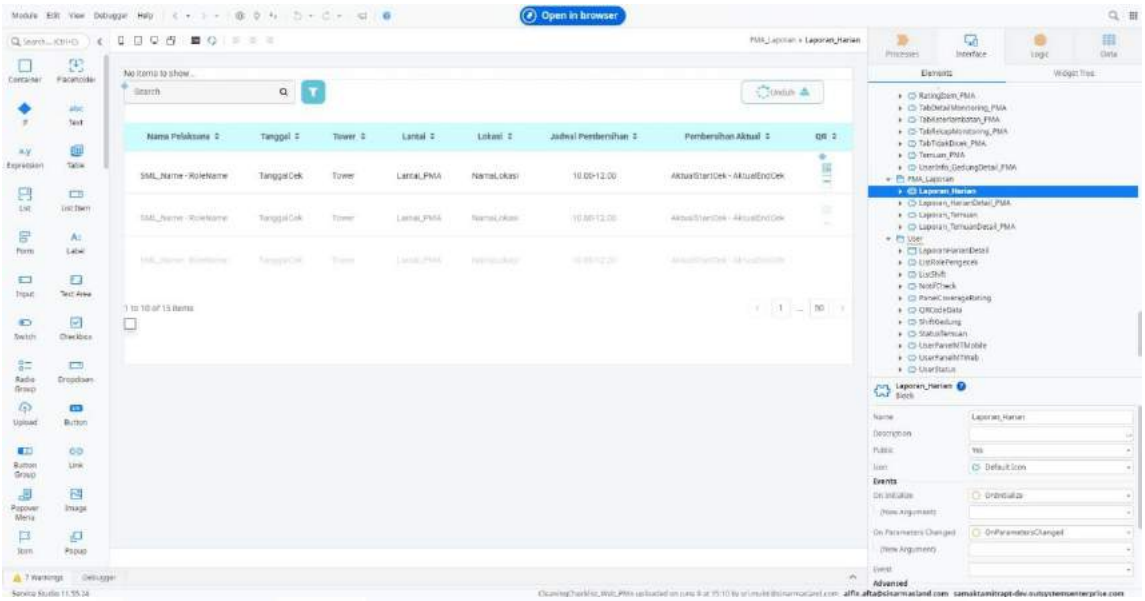
Gambar 3.4 Web Cleaning Checklist – Laporan Harian (Setelah Filter)

Gambar 3.4 menunjukkan hasil akhir halaman laporan harian yang telah dilengkapi dengan fitur filter. Fitur ini memungkinkan pengguna menyaring data berdasarkan gedung, kategori, dan parameter lain yang relevan. Perubahan ini secara signifikan meningkatkan efisiensi pengguna dalam mencari data tertentu, karena mereka tidak lagi harus melihat seluruh data yang ditampilkan sekaligus. Antarmuka menjadi lebih interaktif dan dinamis, memberikan pengalaman pengguna yang jauh lebih baik. Penambahan filter ini juga penting dalam proses audit dan review laporan harian, karena membantu memfokuskan data yang relevan dan kuat.



Gambar 3.5 Dropdown Search Filter Gedung

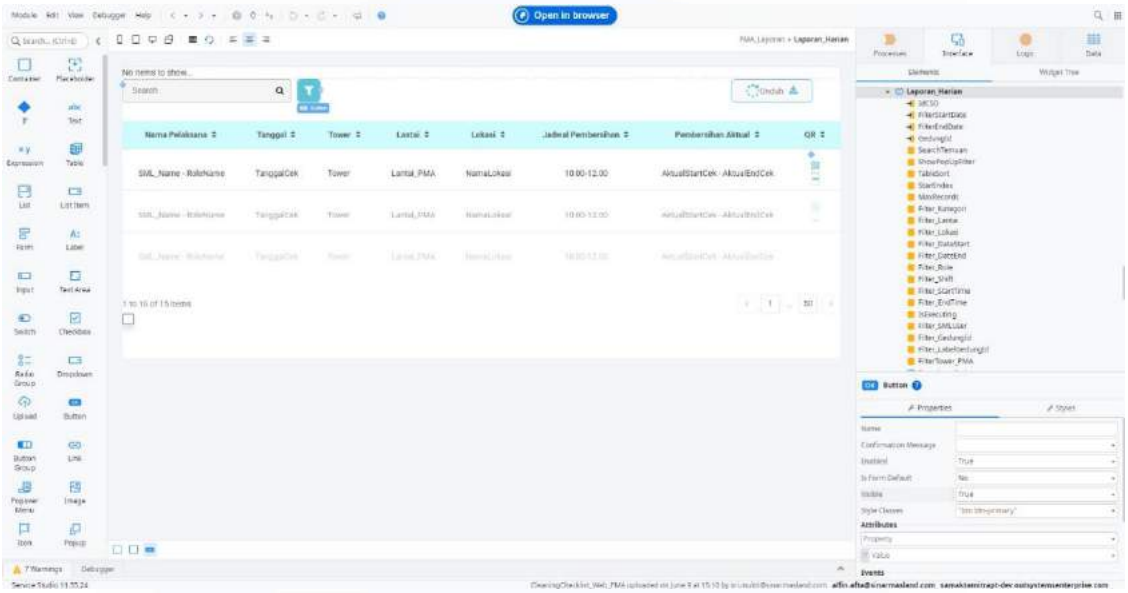
Gambar 3.5 menampilkan komponen DropdownSearch yang diterapkan pada bagian filter gedung. Komponen ini memungkinkan pengguna untuk mencari nama gedung secara langsung dengan mengetikkan huruf pertama atau nama gedung yang diinginkan. Fitur ini sangat membantu dalam mempercepat proses pencarian, terutama jika daftar gedung cukup panjang. Dibandingkan dropdown biasa yang hanya menampilkan daftar statis, DropdownSearch memberikan fleksibilitas dan efisiensi yang lebih tinggi. Gambar 3.5 mencerminkan keberhasilan dalam meningkatkan antarmuka pengguna (UI) dengan komponen yang modern dan fungsional. Implementasi ini juga menjadi solusi utama dari permasalahan awal yang dilaporkan dalam tiket bug.



Gambar 3.6 Tampilan OutSystems – Cleaning Checklist

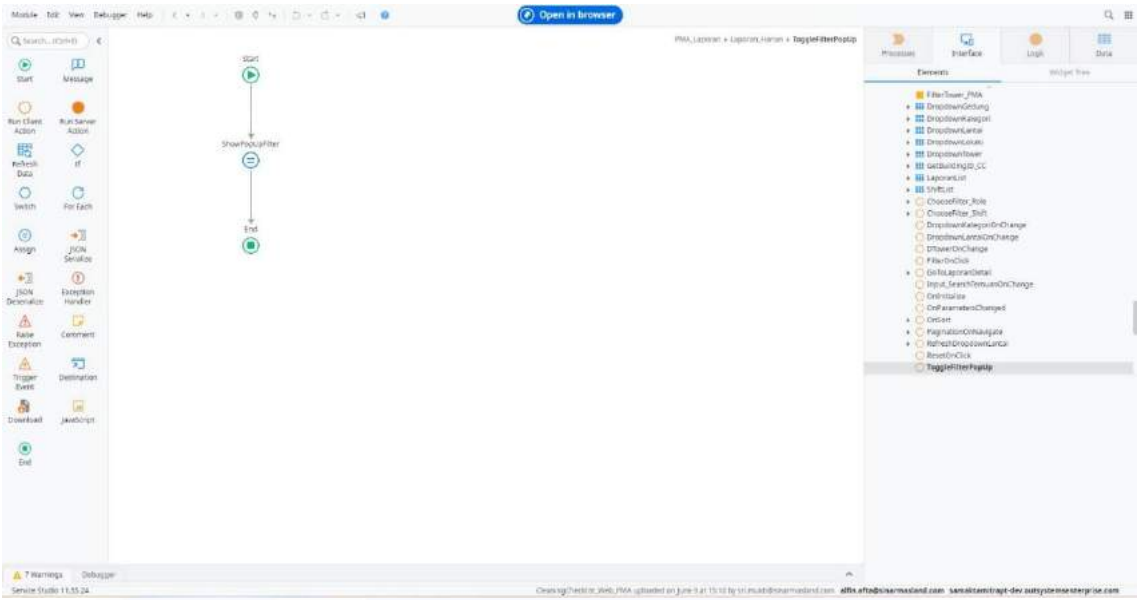
Gambar 3.6 menampilkan tampilan editor layar di platform OutSystems untuk modul Cleaning Checklist. Di dalamnya terlihat komponen-komponen visual seperti tabel laporan harian dan form filter yang disusun secara terstruktur. Desain ini menunjukkan bagaimana OutSystems memungkinkan pengembang membangun antarmuka dengan pendekatan drag-and-drop, sekaligus mengatur logika bisnis secara terintegrasi. Gambar 3.6 juga memberikan gambaran mengenai tata letak UI yang dikembangkan oleh penulis selama magang. Keberadaan popup filter, tombol aksi, dan tabel data yang ditampilkan secara dinamis mencerminkan penerapan prinsip desain modular dan responsif, sehingga tampilan halaman menjadi lebih intuitif dan mudah digunakan oleh user.

3.2.2.2 Analisis Dan Perbaikan Bug Menggunakan OutSystem – Cleaning Checklist



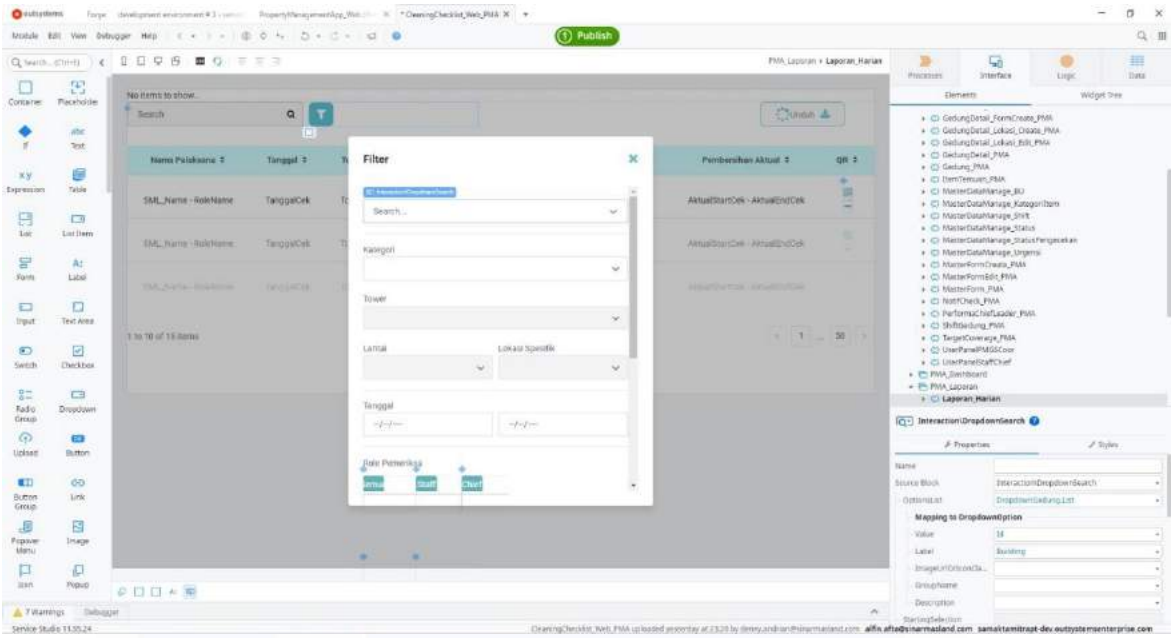
Gambar 3.7 Tombol Filter pada Halaman Cleaning Checklist

Gambar 3.7 menyoroti tombol filter yang menjadi pemicu utama munculnya popup filter pada halaman laporan harian Cleaning Checklist. Tombol ini merupakan komponen penting dalam antarmuka pengguna karena menghubungkan interaksi user dengan sistem logika di belakang layar. Dengan mengklik tombol ini, user dapat menampilkan form filter yang memungkinkan penyaringan data berdasarkan berbagai parameter seperti gedung dan tanggal. Penempatan tombol ini dirancang agar mudah diakses dan terlihat jelas di halaman. Gambar 3.7 memperlihatkan pentingnya desain interaktif dan keterpaduan antara elemen visual dan fungsionalitas, yang sangat berpengaruh terhadap pengalaman pengguna secara keseluruhan.



Gambar 3.8 Server Action – ToggleFilterPopUp

Gambar 3.8 menampilkan logika server action bernama ToggleFilterPopUp yang digunakan untuk mengatur aksi membuka dan menutup popup filter secara dinamis. Dalam OutSystems, server action ini akan dijalankan ketika user mengklik tombol filter. Aksi ini mengubah nilai boolean variabel untuk mengontrol visibilitas popup. Gambar 3.8 menunjukkan bagaimana backend logic dikaitkan langsung dengan antarmuka pengguna. Melalui logika ini, pengembang dapat memastikan bahwa popup hanya muncul ketika diperlukan, tanpa harus melakukan reload halaman secara keseluruhan. Penggunaan server action ini mencerminkan pendekatan efisien dan reaktif dalam pemrograman low-code di OutSystems.



Gambar 3.9 Query FilterGedung pada OutSystems

Gambar 3.9 menunjukkan query agregat bernama FilterGedung yang digunakan untuk mengambil data gedung dari database. Data ini kemudian digunakan sebagai sumber (data source) untuk komponen DropdownSearch di form filter. Query ini difilter berdasarkan kondisi tertentu agar hanya menampilkan gedung aktif atau sesuai parameter pencarian. Gambar 3.9 menegaskan pentingnya pengambilan data yang efisien dan tepat sasaran dalam aplikasi. Dengan menggunakan query agregat yang teroptimasi, proses loading data menjadi lebih cepat dan akurat. Hal ini memastikan bahwa pengguna hanya melihat pilihan gedung yang relevan, sehingga pengalaman pencarian menjadi lebih mudah dan fokus.

Pada Task 1, Memperbaiki fitur filter pada halaman laporan harian dalam modul Cleaning Checklist. Permasalahan diawali dari laporan bug Gambar 3.2 yang menunjukkan bahwa filter gedung tidak muncul di halaman laporan, menyebabkan pengguna kesulitan menyaring data berdasarkan lokasi kerja. Gambar 3.3 menunjukkan kondisi awal halaman tanpa filter, di mana semua data ditampilkan secara keseluruhan. Gambar 3.4 menampilkan tampilan setelah fitur filter ditambahkan, memberikan opsi kepada pengguna untuk memilih gedung tertentu. Gambar 3.5 menunjukkan komponen DropdownSearch yang ditambahkan agar

pengguna dapat mencari nama gedung dengan lebih cepat. Desain tampilan halaman yang ditunjukkan dalam Gambar 3.6 mengilustrasikan peletakan elemen filter yang telah diperbarui. Interaksi pengguna dengan tombol filter ditunjukkan pada Gambar 3.7, sedangkan logika server untuk membuka dan menutup form filter dapat dilihat pada Gambar 3.8. Terakhir, Gambar 3.9 menampilkan query untuk mengambil data gedung dari database, memastikan daftar yang muncul selalu akurat.

3.2.2.3 Langkah Pengerjaan

Berikut tahapan pengerjaan yang saya lakukan:

1. Membuat Popup Filter
    a. Menambahkan komponen Popup pada halaman Laporan_Harian.
    b. Menyusun layout filter dengan berbagai komponen input seperti Dropdown, Buildingid dan data lain yang sudah disediakan perusahaan
2. Mengatur Dropdown Search
    a. Menggunakan DropdownSearch untuk dropdown gedung agar user dapat mencari nama gedung dengan cepat.
    b. Mengatur properti OptionsList, Value, dan Label pada komponen tersebut agar data tampil sesuai.
3. Menghubungkan Filter ke Agregat Data
    a. Membuat logic untuk mem-filter data tabel berdasarkan input user di popup filter.
    b. Menambahkan parameter filter di query agregat (misalnya filter berdasarkan BuildingId, BuildingName, dan Role).
4. Pengujian (Testing)

a. Melakukan beberapa skenario pengujian seperti: filter berdasarkan satu parameter, kombinasi dua parameter, dan semua parameter.

b. Memastikan data yang ditampilkan sudah sesuai hasil filter.

3.2.2.5 Hasil yang Dicapai

Fitur filter berhasil ditambahkan ke halaman laporan harian. Pengguna kini dapat mencari data laporan lebih cepat dan akurat sesuai kebutuhan mereka. Fitur ini juga meningkatkan efisiensi penggunaan aplikasi dan mempermudah proses audit atau review harian.

3.2.3 TASK 2 Mengubah DropDown Biasa Menjadi DropDown Search di aplikasi web Shift HandOver pada Equipment.

Deskripsi Task, Pada task kedua ini, saya ditugaskan untuk memperbaiki dropdown equipment yang digunakan pada halaman penjadwalan *Preventive Maintenance* dalam aplikasi PMA. Permasalahan yang terjadi sebelumnya adalah dropdown tidak menampilkan data dan juga tidak mendukung pencarian (search).

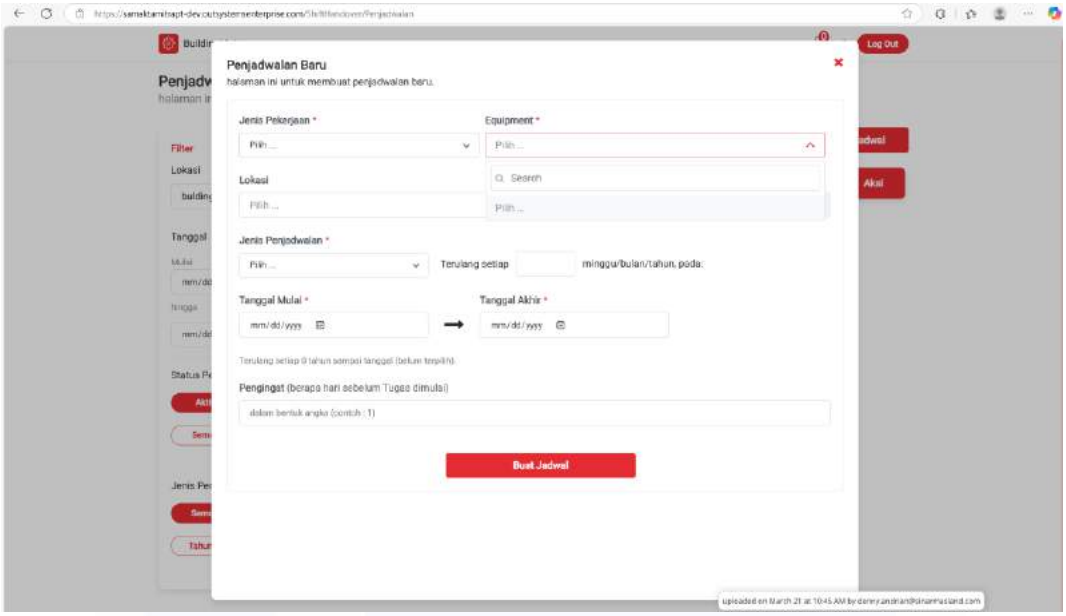
Current Condition (Sebelum Perbaikan)

1. Dropdown tidak menampilkan data equipment meskipun data tersedia di database.
2. Dropdown belum menggunakan fitur pencarian, sehingga menyulitkan user dalam memilih equipment tertentu.

Expected Condition (Setelah Perbaikan)

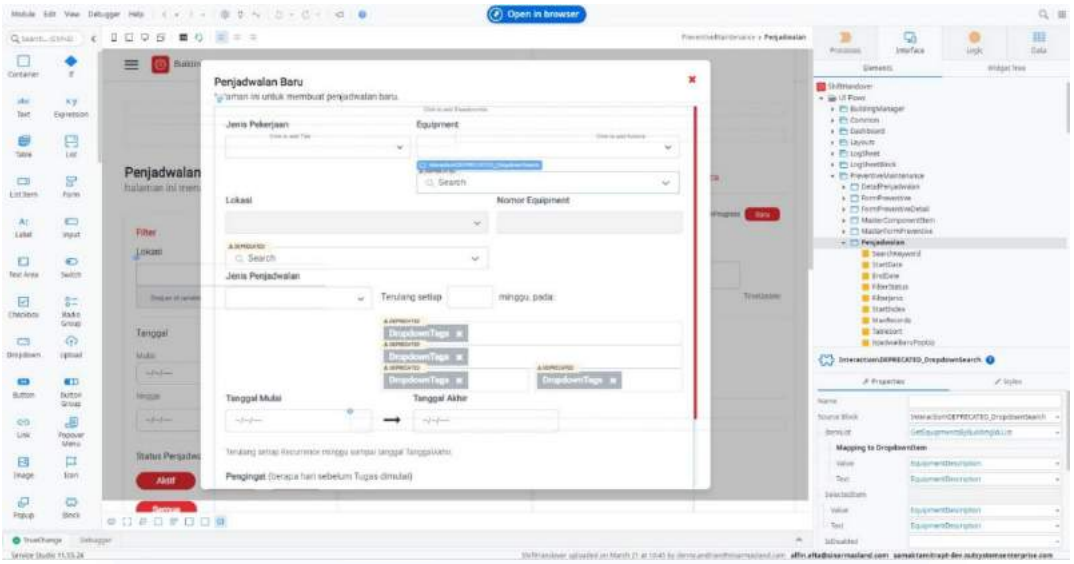
1. Dropdown dapat menampilkan data equipment berdasarkan gedung (building) yang dipilih.
2. Dropdown dapat di-search menggunakan input teks, agar user bisa cepat menemukan equipment yang dimaksud.

3.2.3.1 Analisis Aplikasi Web ShiftHandOver



Gambar 3.10 Dropdown Search pada Halaman ShiftHandOver

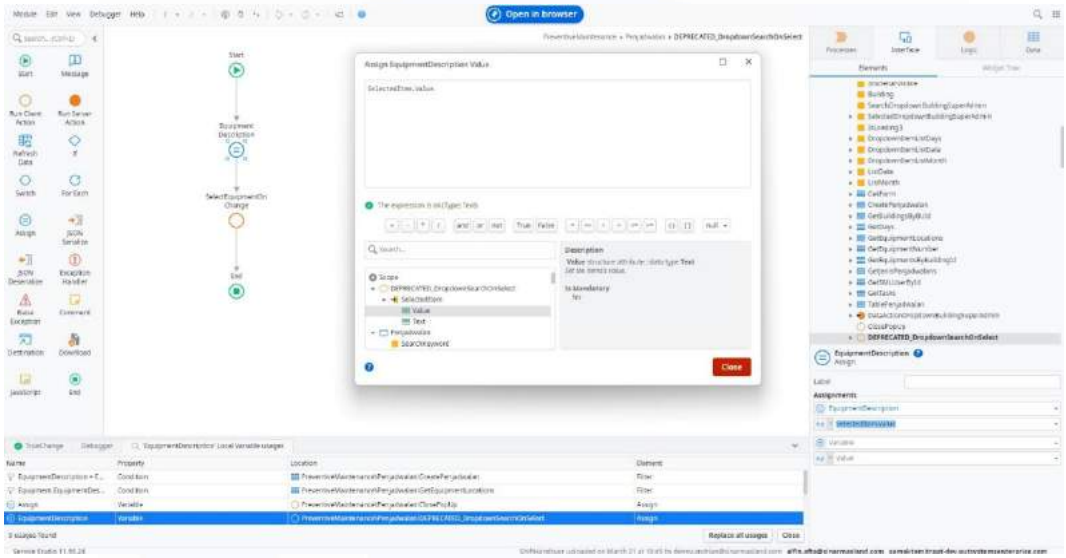
Gambar 3.10 memperlihatkan tampilan halaman ShiftHandOver setelah dropdown untuk pemilihan equipment diubah menjadi fitur DropdownSearch. Sebelumnya, dropdown tidak menampilkan data dan tidak bisa di-search, menyebabkan pengguna kesulitan saat memilih peralatan yang tepat. Setelah perbaikan, DropdownSearch memungkinkan pencarian data secara real-time berdasarkan input teks, sehingga user dapat menemukan item dengan cepat. Gambar 3.10 juga mencerminkan peningkatan signifikan dalam hal aksesibilitas dan efisiensi. Implementasi ini sangat bermanfaat terutama ketika daftar peralatan sangat panjang dan perlu difilter berdasarkan kebutuhan spesifik user dalam modul preventive maintenance.



Gambar 3.11 Komponen Dropdown Search Equipment di OutSystems

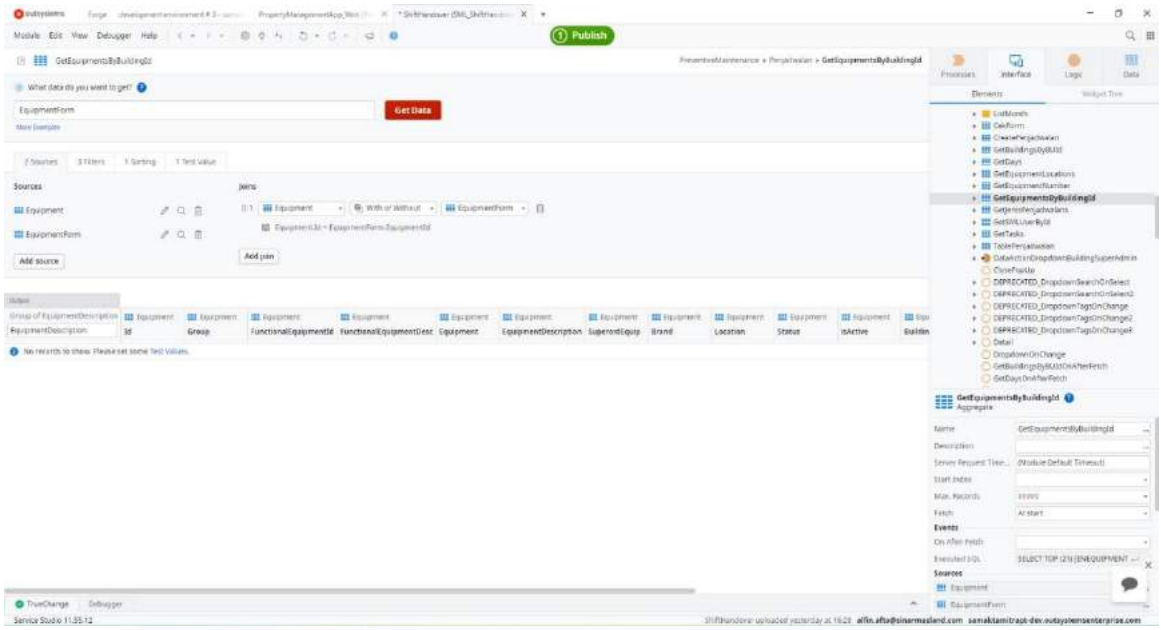
Gambar 3.11 menampilkan konfigurasi DropdownSearch yang digunakan untuk memilih equipment pada halaman ShiftHandOver. Komponen ini diatur dengan property OptionsList, Value, dan Label agar menampilkan data dari query GetEquipmentByBuildingId. DropdownSearch ini menggantikan dropdown statis yang sebelumnya tidak menampilkan data dengan benar. Kini pengguna dapat dengan mudah mencari nama peralatan berdasarkan input teks. Gambar 3.11 menunjukkan kemampuan OutSystems dalam menyusun UI dinamis yang terintegrasi langsung dengan data backend, sekaligus mencerminkan upaya peningkatan pengalaman pengguna dalam memilih item yang relevan secara cepat.

3.2.3.2 Analisis dan Perbaikan Bug Menggunakan OutSystem - ShiftHandOver



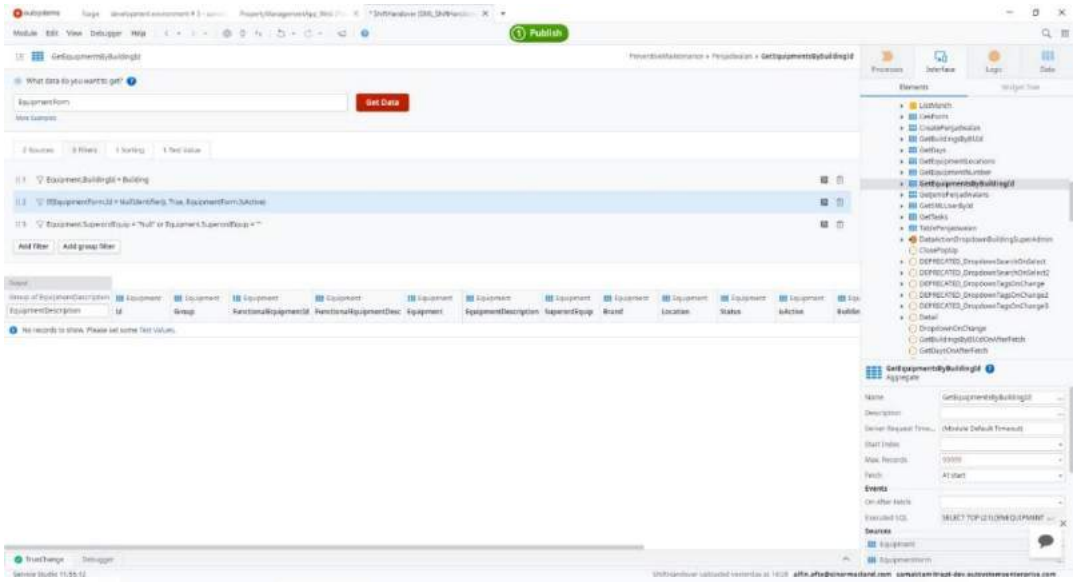
Gambar 3.12 Server Action – Assign Equipment Description

Gambar 3.12 menampilkan logika server action yang digunakan untuk menyimpan data hasil pilihan user dari DropdownSearch. Logic ini memastikan bahwa equipment yang dipilih akan disimpan dan digunakan untuk keperluan selanjutnya, seperti pengisian form atau pelaporan. Pengaturan ini juga menjaga integritas data yang dikirim ke backend. Proses ini penting karena mencegah kesalahan dalam pengiriman data dan memungkinkan alur kerja aplikasi berjalan lebih mulus. Gambar 3.12 memperlihatkan bagaimana logika backend yang sederhana namun krusial dapat menjaga fungsi form tetap stabil dan sesuai harapan.



Gambar 3.13 Aggregate GetEquipmentByBuildingId

Gambar 3.13 memperlihatkan query agregat yang digunakan untuk mengambil daftar equipment berdasarkan ID gedung yang dipilih oleh user. Query ini bersifat dinamis, yang berarti hasilnya akan berubah tergantung pada input BuildingId dari user. Dengan metode ini, daftar dropdown hanya akan menampilkan peralatan yang relevan di lokasi tersebut. Hal ini sangat penting untuk menyederhanakan pencarian dan menghindari error akibat pemilihan item yang tidak sesuai lokasi. Gambar 3.13 juga menunjukkan praktik terbaik dalam pemfilteran data berbasis relasi antar-entitas di OutSystems.



Gambar 3.14 Aggregate GetBuildingByBuildingId

Gambar 3.14 menunjukkan agregat query yang digunakan untuk mengambil informasi detail dari sebuah gedung berdasarkan ID-nya. Data ini digunakan untuk mengisi informasi lain di halaman atau sebagai acuan dalam memetakan relasi antar data. Penggunaan query ini memungkinkan pengembang mengambil data spesifik dan menghindari kebutuhan menampilkan semua data dalam satu waktu. Ini meningkatkan efisiensi sistem dan mengurangi beban aplikasi. Gambar 3.14 menunjukkan pemahaman logika relational database yang diimplementasikan secara visual melalui platform OutSystems untuk mendukung kebutuhan dinamis aplikasi.

Task ini berfokus pada perbaikan dropdown pada halaman ShiftHandOver. Pada awalnya, dropdown tidak dapat menampilkan data secara benar sehingga menyulitkan pengguna dalam memilih peralatan. Gambar 3.10 menampilkan tampilan awal halaman dengan dropdown bermasalah. Setelah dilakukan perbaikan, komponen DropdownSearch yang lebih fungsional seperti terlihat pada Gambar 3.11 berhasil diterapkan. Server Action untuk menyimpan deskripsi peralatan yang dipilih ditunjukkan dalam Gambar 3.12. Untuk mendukung dropdown ini, data peralatan diambil menggunakan query dalam Gambar 3.13

berdasarkan ID gedung yang dipilih, sedangkan informasi gedung yang terpilih ditampilkan menggunakan query yang ditunjukkan pada Gambar 3.14.

Analisis Permasalahan

Saya mulai dengan menganalisis struktur dropdown sebelumnya dan menemukan bahwa Dropdown masih menggunakan komponen standar, bukan searchable dropdown dan Data source belum dihubungkan dengan benar ke query GetEquipmentsByBuildingId, sehingga datanya tidak muncul.

3.2.3.3 Langkah Pengerjaan

1. Mengganti Komponen Dropdown
    a. Saya mengganti dropdown standar dengan block Interaction\DropdownSearch yang mendukung fitur pencarian (searchable).
    b. Komponen ini memungkinkan user mengetik untuk mencari nama equipment secara real-time.
2. Menghubungkan Data Source
    a. Saya mengatur OptionsList pada block DropdownSearch agar mengambil data dari GetEquipmentsByBuildingId.List.
    b. Melakukan pemetaan (mapping) field:
        a) Value → Id
        b) Label → EquipmentDescription
    c. Ini memastikan dropdown menampilkan label yang dapat dimengerti user, namun menyimpan ID sebagai nilai yang dikirim.
3. Testing dan Validasi
    a. Mengisi nilai test (Test Values) untuk gedung agar aggregate GetEquipmentsByBuildingId dapat mengeluarkan data.

b. Memastikan dropdown berhasil menampilkan data, dapat digunakan, dan mendukung search.

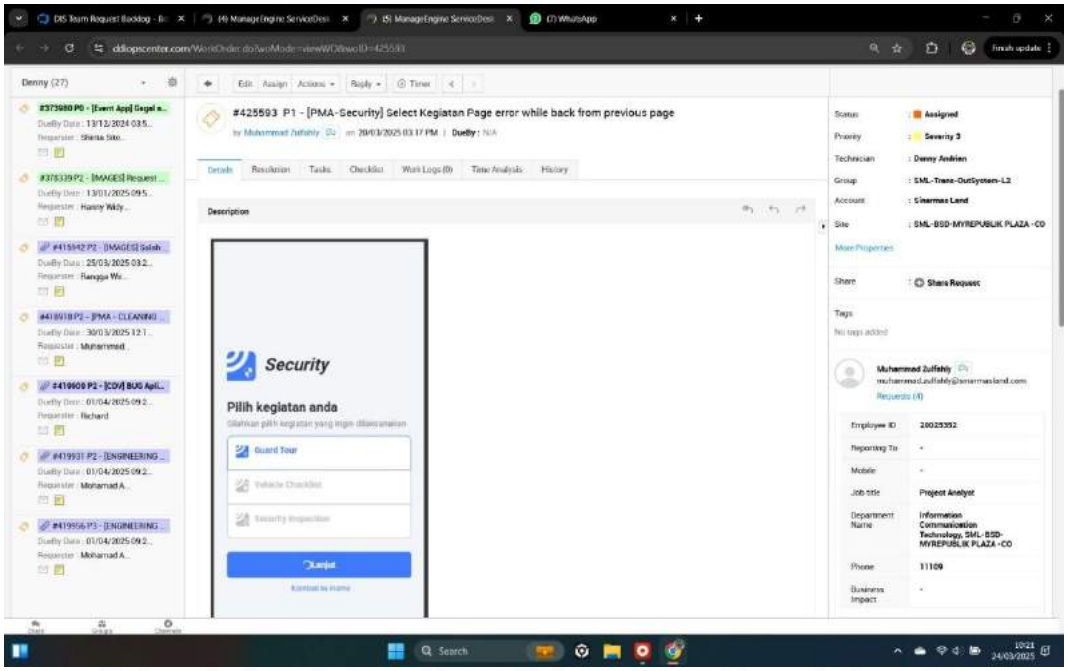
c. Menguji dropdown dengan berbagai skenario: saat gedung tidak dipilih, saat equipment tidak aktif, atau saat data kosong.

3.2.3.4 Hasil yang Dicapai

Dropdown equipment berhasil diperbaiki sehingga:

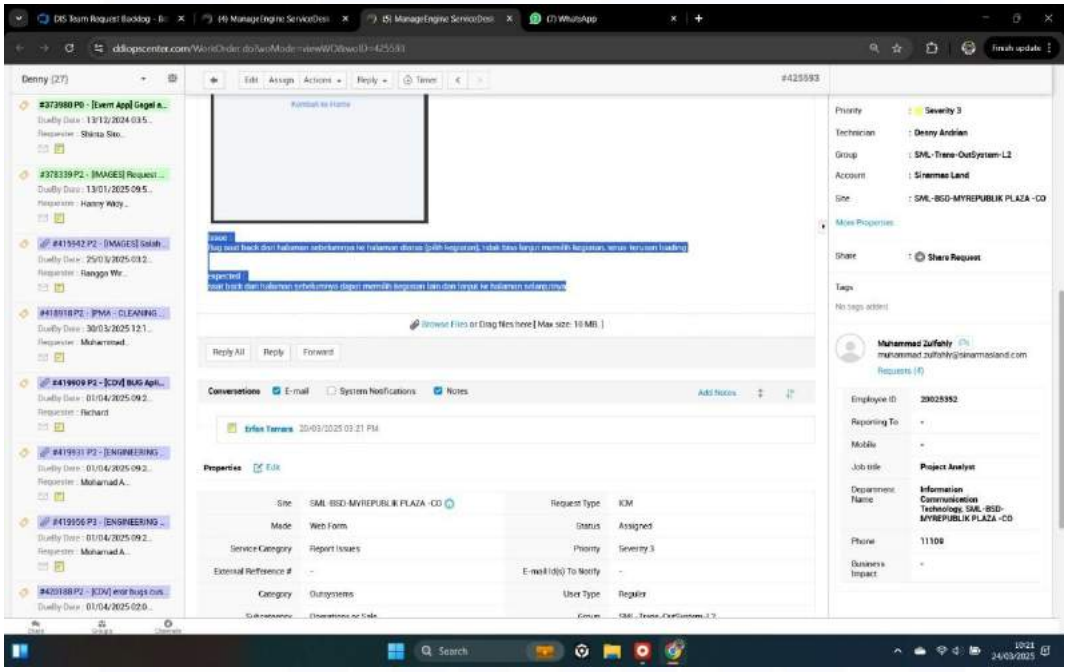
1. Data equipment muncul sesuai gedung yang dipilih.
2. Dropdown dapat di-search, membuat pengguna lebih cepat menemukan peralatan yang dimaksud.
3. UX (user experience) menjadi lebih baik, terutama saat data equipment sangat banyak.

3.2.4 TASK 3 Select Kegiatan Page error ketika kembali ke halaman sebelumnya.



Gambar 3.15 Tampilan Tiket PMA Security – Pilih Kegiatan

Gambar 3.15 menampilkan tiket bug yang masuk ke sistem PMA Security terkait halaman pemilihan kegiatan. Masalah utamanya adalah halaman tersebut tidak bisa digunakan lagi setelah pengguna menekan tombol back dari halaman aktivitas. Gambar ini menjadi acuan awal dalam analisis bug yang harus diperbaiki. Halaman Pilih Kegiatan merupakan bagian penting dalam modul Security App karena menentukan kegiatan yang akan dilaporkan. Gambar 3.15 menegaskan pentingnya pemrosesan state halaman dan validasi navigasi agar alur kerja pengguna tidak terganggu dan tetap berjalan lancar.



Gambar 3.16 Detail Masalah dan Ekspektasi – Pilih Kegiatan

Gambar 3.16 menjelaskan lebih lanjut permasalahan yang terjadi serta ekspektasi dari hasil perbaikannya. User berharap halaman Pilih Kegiatan tetap bisa digunakan dengan normal meskipun pengguna kembali dari halaman aktivitas sebelumnya. Penjelasan ini memudahkan tim developer memahami konteks dan reproduksi bug yang terjadi. Gambar 3.16 penting untuk mendokumentasikan kebutuhan pengguna dan menyelaraskan solusi teknis dengan harapan fungsional dari sisi pengguna akhir. Kejelasan deskripsi masalah juga menjadi dasar evaluasi apakah solusi yang diterapkan sudah sesuai dengan kebutuhan.

Deskripsi Task, Pada task ini, saya ditugaskan untuk memperbaiki bug pada halaman *Pilih Kegiatan* di modul Security dalam aplikasi mobile PMA. Bug terjadi saat user menekan tombol back (kembali) dari halaman aktivitas tertentu ke halaman pemilihan kegiatan, di mana halaman menjadi terus loading dan tidak bisa digunakan lagi untuk memilih kegiatan selanjutnya.

Detail Permasalahan:

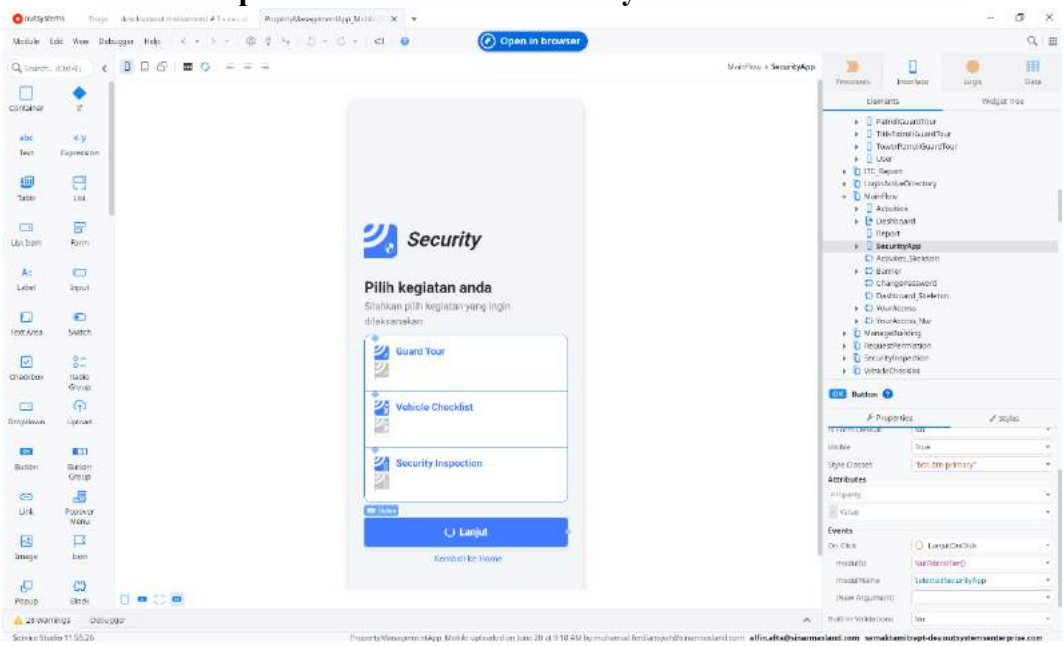
1. Issue:

   Saat kembali (back) ke halaman *Pilih Kegiatan*, halaman tidak bisa digunakan untuk memilih aktivitas lagi. Tombol “Lanjut” tidak merespons dan sistem terus-menerus loading.

2. Expected:

   Setelah menekan back, user tetap bisa memilih aktivitas lain dan menavigasi ke halaman sesuai tanpa masalah.

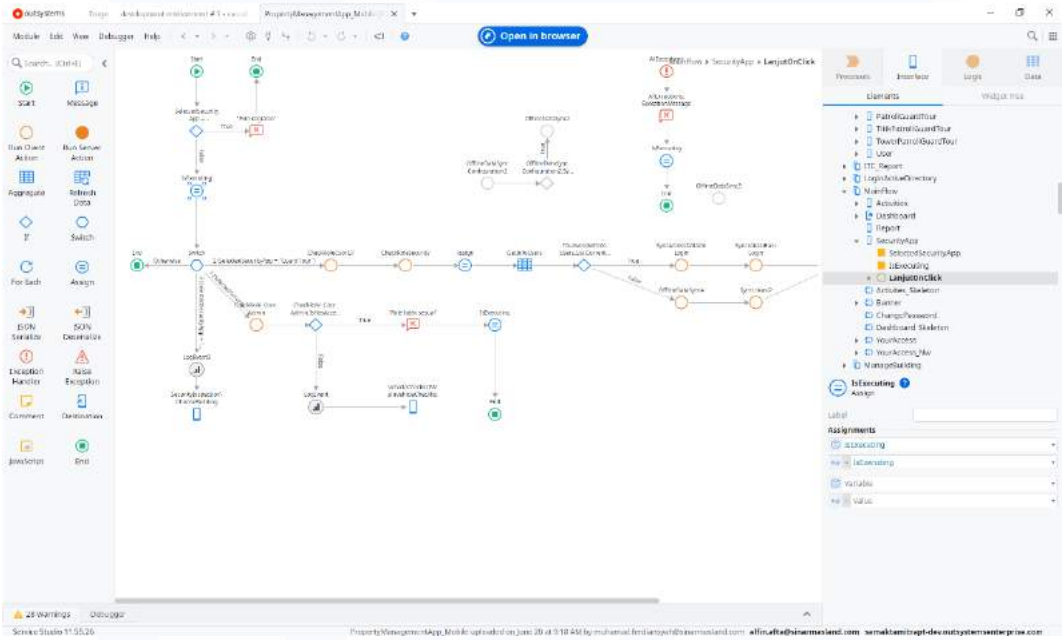
3.2.4.1 Analisis Aplikasi Mobile PMA-Security



Gambar 3.17 Tampilan Mobile App – Halaman Pilih Kegiatan

Gambar 3.17 menunjukkan tampilan antarmuka aplikasi mobile saat user berada di halaman Pilih Kegiatan. Ketika bug terjadi, halaman ini mengalami freeze atau tidak merespons, menyebabkan pengguna tidak dapat memilih ulang aktivitas. Visual ini memberikan ilustrasi langsung bagaimana bug berdampak pada pengalaman pengguna. Gambar 3.17 digunakan sebagai referensi penting dalam uji coba dan pengujian ulang setelah perbaikan dilakukan. Pemahaman antarmuka yang ditampilkan juga membantu tim teknis memastikan bahwa perubahan yang dilakukan tidak merusak tata letak atau fungsi visual lainnya.

3.2.4.2 Analsis Dan Perbaikan Bug Menggunakan OutSystem – PMA Security



Gambar 3.18 Server Action – Halaman SecurityApp (Mobile)

Gambar 3.18 menampilkan bagian server action pada halaman SecurityApp yang diperbaiki untuk mencegah error saat pengguna kembali ke halaman Pilih Kegiatan. Di sini, logika untuk reset variabel dan komponen visual ditambahkan agar halaman dapat dimuat ulang dari awal. Perbaikan ini sangat penting karena memastikan state halaman selalu konsisten, terutama pada aplikasi mobile yang rentan terhadap cache dan penyimpanan nilai sebelumnya. Gambar 3.18 menggambarkan solusi teknis

terhadap bug UI/UX yang muncul karena tidak adanya reset state saat navigasi ulang.

Task ketiga merupakan penanganan bug pada aplikasi mobile PMA Security, di mana pengguna tidak dapat melanjutkan pemilihan kegiatan saat menekan tombol kembali. Gambar 3.15 menunjukkan tiket bug dari sistem pelaporan internal. Gambar 3.16 menampilkan rincian masalah dan hasil yang diharapkan oleh pengguna. Gambar 3.17 memperlihatkan tampilan aplikasi saat bug terjadi. Analisis logika dilakukan pada server action seperti terlihat pada Gambar 3.18, yang kemudian diperbaiki dengan memperbaiki pengaturan variabel agar halaman dapat termuat ulang dengan benar.

Analisis Masalah, Saya memulai dengan menganalisis alur logika dan state management pada halaman SecurityApp, khususnya pada:

1. Variable yang menyimpan pilihan aktivitas (contoh: SelectedSecurityApp)
2. OnClick Event dari tombol “Lanjut”
3. Kemungkinan cache / variable state yang tidak direset ketika user kembali ke halaman ini

Saya menduga bahwa:

1. Variabel SelectedSecurityApp masih menyimpan nilai lama atau tidak direset.
2. Event LanjutOnClick tidak memproses ulang pilihan karena state tidak berubah.
3. Komponen UI tidak refresh ulang ketika halaman dibuka kembali via tombol *Back*.

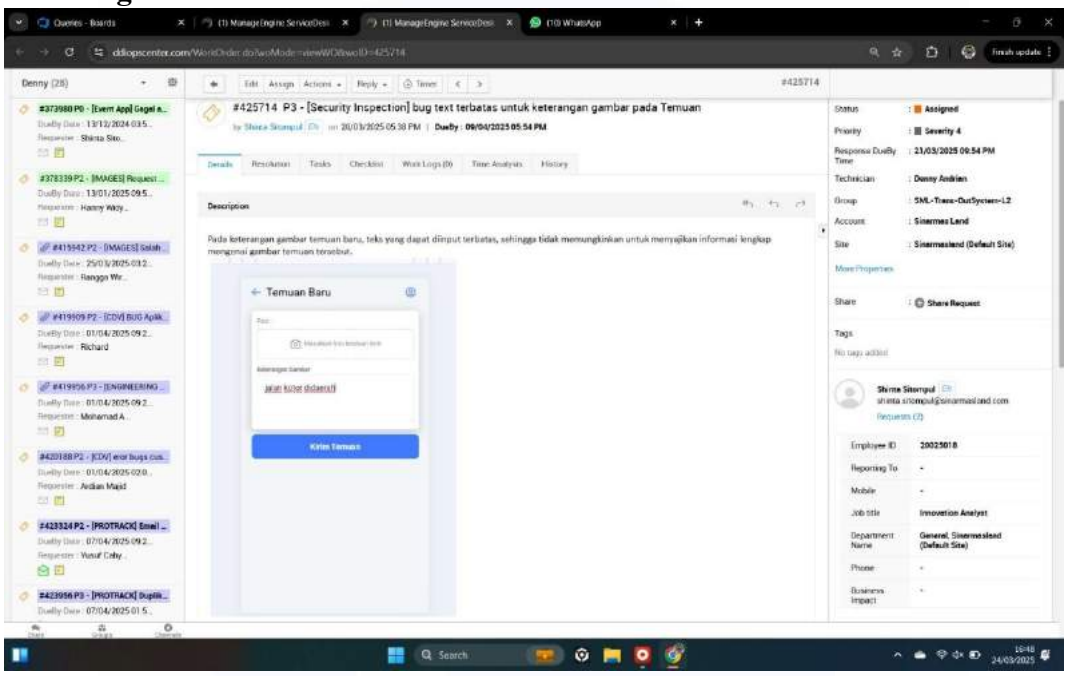
3.2.4.3 Langkah Perbaikan

1. Reset State Saat Masuk Halaman
    a. Saya menambahkan action pada OnReady atau OnInitialize halaman SecurityApp untuk mengosongkan nilai SelectedSecurityApp.
    b. Hal ini memastikan setiap kali halaman dibuka, user memulai dari kondisi awal.
2. Validasi Pilihan Sebelum Navigasi
    a. Pada LanjutOnClick, saya pastikan bahwa navigasi hanya dilakukan jika SelectedSecurityApp memiliki nilai yang valid.
    b. Menambahkan validasi visual (misalnya warning jika belum memilih aktivitas).
3. Menambahkan Logic Refresh
    a. Jika ada komponen yang tergantung pada input binding (seperti list item), dilakukan binding ulang agar UI tidak stuck pada loading loop.
4. Testing dan Simulasi Ulang Bug
    a. Saya mencoba menjalankan simulasi "masuk halaman → pilih aktivitas → lanjut → back → pilih aktivitas lain → lanjut" berulang kali untuk memastikan bug sudah hilang.

3.2.4.4 Hasil yang Dicapai

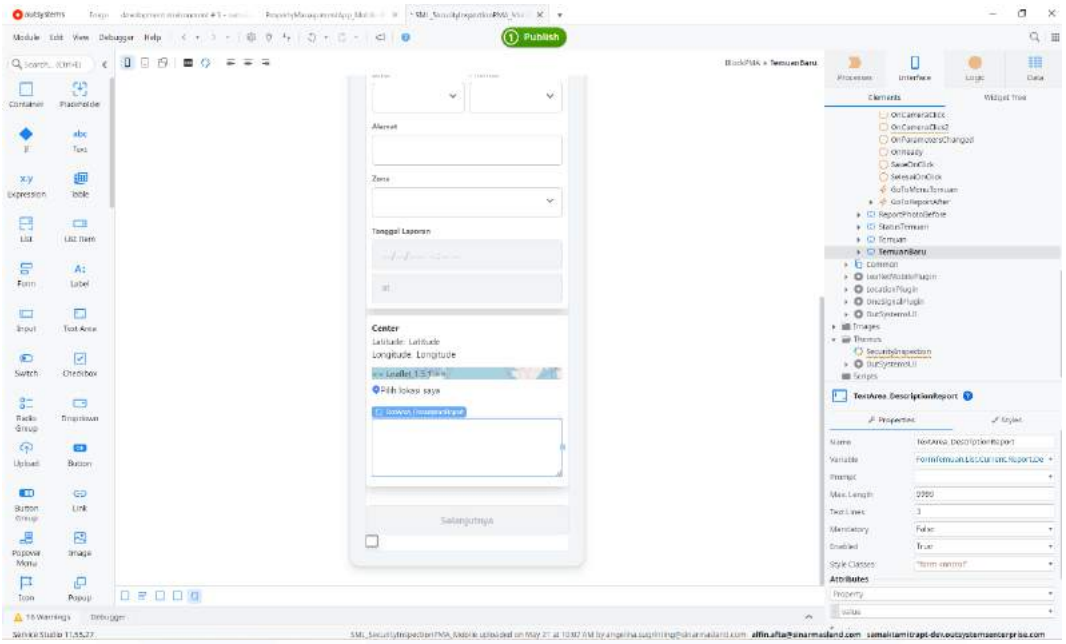
1. Bug tidak muncul lagi setelah tombol back ditekan.
2. User dapat kembali ke halaman *Pilih Kegiatan* dan memilih aktivitas lain tanpa error atau loading terus-menerus.
3. Alur kerja menjadi lebih lancar dan tidak mengganggu operasional petugas keamanan.

3.2.5 TASK 4 Menambahkan Batas Maksimum Karakter pada Field Keterangan Gambar



Gambar 3.19 Tiket PMA – Field Keterangan Gambar

Gambar 3.19 menampilkan tiket bug yang melaporkan adanya batasan karakter pada field “Keterangan Gambar” yang terlalu kecil, sehingga input pengguna terpotong. Masalah ini menjadi krusial karena menghambat penyampaian informasi penting oleh pengguna terkait gambar temuan di lapangan. Dokumentasi tiket ini menjadi dasar untuk menelusuri komponen TextArea yang digunakan dan mengidentifikasi pengaturan Max Length yang belum optimal. Gambar 3.19 sekaligus menjadi bukti adanya kebutuhan peningkatan fleksibilitas form input agar informasi yang dimasukkan user bisa ditampilkan secara lengkap.



Gambar 3.20 Komponen TextArea – DescriptionReport

Gambar 3.20 memperlihatkan komponen TextArea yang digunakan untuk mengisi keterangan gambar. Perubahan utama yang dilakukan adalah penyesuaian properti Max Length dari nilai rendah menjadi lebih besar (misalnya 9999 karakter). Hal ini memungkinkan user menulis penjelasan lebih lengkap tanpa khawatir informasi terpotong. Gambar 3.20 menggambarkan perbaikan sederhana namun berdampak besar terhadap kepuasan pengguna. Peningkatan ini menunjukkan pentingnya memperhatikan batas input field dalam desain aplikasi, terutama saat field tersebut digunakan untuk deskripsi naratif yang mendetail.

Dalam Task 4, penulis melakukan perbaikan pada form input keterangan gambar di halaman Temuan Baru. Sebelumnya, pengguna tidak dapat memasukkan deskripsi panjang karena batas karakter terlalu kecil. Gambar 3.19 menunjukkan tiket bug yang dilaporkan, sedangkan Gambar 3.20 memperlihatkan komponen TextArea setelah properti Max Length ditingkatkan untuk memungkinkan input teks yang lebih panjang. Analisis Masalah, Field menggunakan Text Area namun properti Max Length belum diatur dengan benar, sehingga membatasi input user.

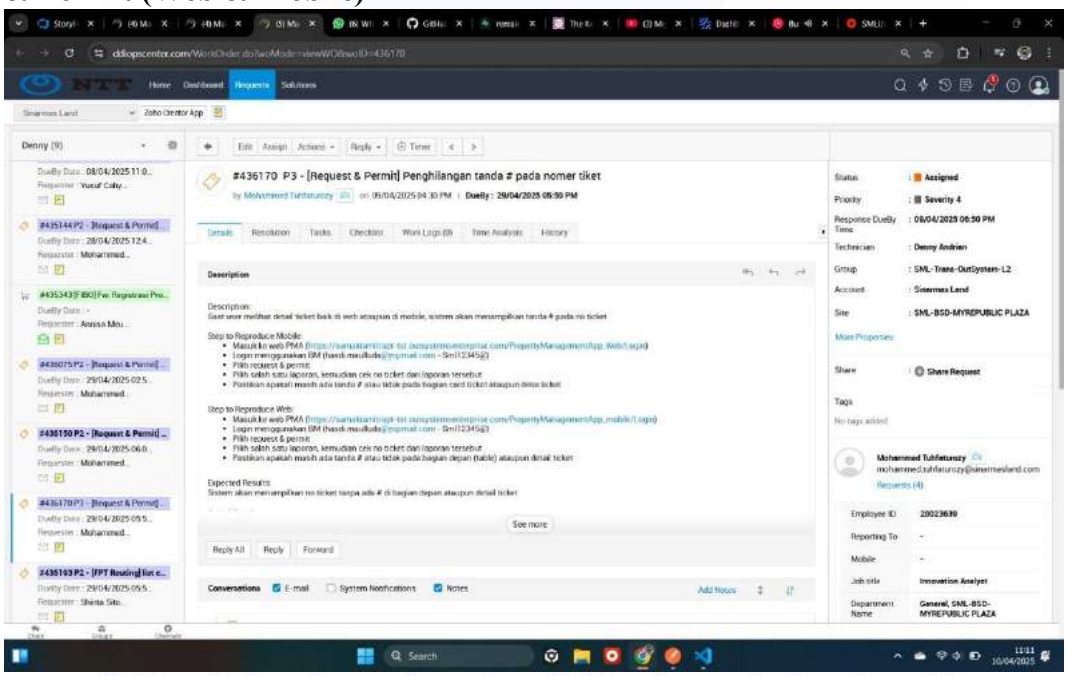
3.2.5.3 Langkah Perbaikan

1. Membuka komponen TextArea_DescriptionReport di halaman TemuanBaru.
2. Mengatur property Max. Length menjadi 9999 agar user bisa menginput deskripsi panjang.
3. Melakukan testing dengan input teks panjang untuk memastikan teks tidak lagi terpotong.

3.2.5.3 Hasil yang Dicapai

User kini dapat menulis keterangan gambar secara lengkap tanpa adanya pemotongan teks. Permasalahan pada form sudah terselesaikan.

3.2.6 TASK 5 Penghilangan Tanda # pada Nomor Tiket di Halaman Request & Permit (Web & Mobile)



Gambar 3.21 Tiket Request & Permit – Nomor Tiket Bertanda #

Gambar 3.21 menampilkan laporan bug mengenai adanya tanda “#” yang muncul di depan nomor tiket. Meskipun terlihat sepele, penambahan karakter ini menimbulkan kebingungan, terutama saat nomor tiket digunakan untuk keperluan administratif atau pencarian dalam sistem lain. Gambar ini menjadi dokumentasi penting yang menunjukkan bahwa aspek visual seperti formatting dapat berpengaruh besar terhadap keakuratan data. Tiket ini menjadi dasar untuk mencari tahu apakah simbol "#" berasal dari database, atau ditambahkan manual melalui komponen tampilan.

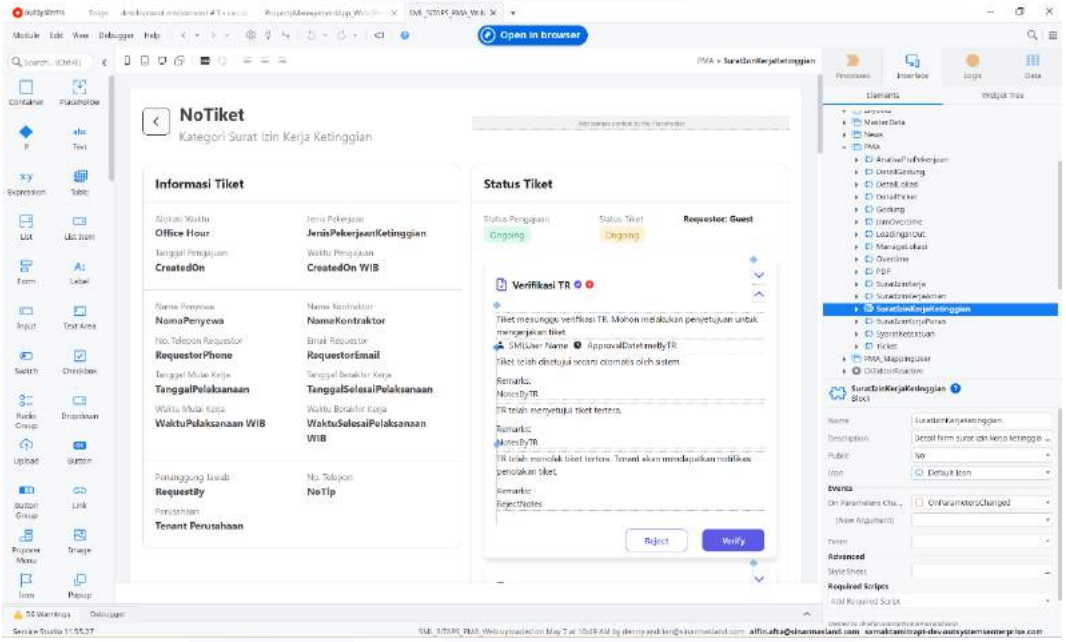
Current Condition (Sebelum Perbaikan):

1. Tanda # secara otomatis ditambahkan pada nomor tiket di halaman detail, baik saat diakses melalui web maupun mobile.
2. Tanda # muncul di bagian tampilan tabel (table) maupun field detail tiket.
3. Hal ini menimbulkan ketidaksesuaian format dan berpotensi menghambat pemrosesan internal yang membutuhkan format nomor tiket tanpa simbol tambahan.

Expected Condition (Setelah Perbaikan):

1. Nomor tiket ditampilkan tanpa tambahan karakter # baik di halaman web maupun mobile.
2. Format nomor tiket bersih, sesuai dengan nilai aktual dari database, untuk memudahkan pencarian atau validasi oleh sistem maupun user.

3.2.6.1 Analsis dan Perbaikan Bug Menggunakan OutSystem – Request & Permit



Gambar 3.22 Tampilan Surat Izin Kerja – Format Nomor Tiket

Gambar 3.22 menunjukkan hasil tampilan surat izin kerja setelah perbaikan dilakukan. Nomor tiket kini ditampilkan tanpa tanda “#”, sesuai dengan nilai asli yang tersimpan di database. Format tampilan yang bersih ini memudahkan user dalam mencatat atau mereferensikan nomor tiket dengan benar. Gambar 3.22 memperlihatkan dampak positif dari perbaikan sederhana yang meningkatkan kejelasan informasi, profesionalisme tampilan, serta menghindari potensi kekeliruan dalam proses validasi atau pencatatan nomor tiket.

Pada Task 5, pengguna melaporkan bahwa nomor tiket ditampilkan dengan tanda '#' yang tidak diinginkan, baik di halaman mobile maupun web. Gambar 3.21 memperlihatkan laporan bug tersebut. Setelah perbaikan, tampilan nomor tiket menjadi lebih rapi dan profesional sebagaimana terlihat pada Gambar 3.22. Setelah menelusuri struktur halaman detail tiket, saya menemukan bahwa:

1. Penambahan 45ymbol # dilakukan secara eksplisit melalui expression atau pemrosesan string di tampilan (frontend).

2. Nilai yang ditampilkan adalah hasil gabungan dari string “#” dan nilai tiket, bukan dari database secara langsung.

3.2.6.2 Langkah Perbaikan:

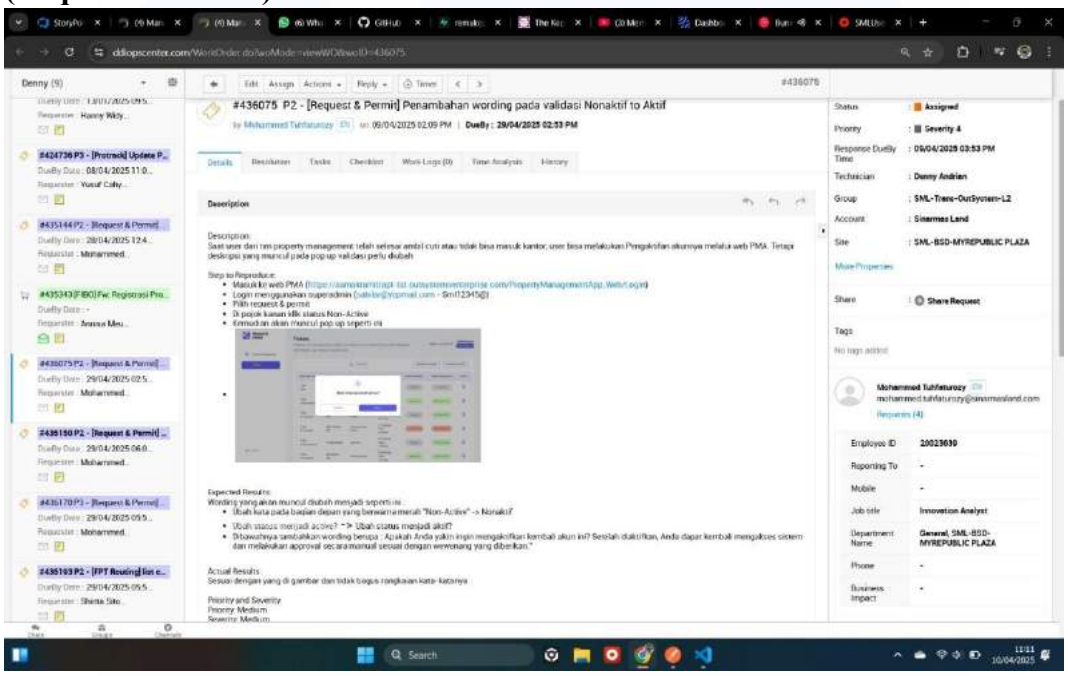
1. Menelusuri Lokasi Penambahan Tanda #:
    a. Saya mengecek widget expression atau text label di halaman Detail Ticket Web dan Mobile, khususnya di bagian yang menampilkan nomor tiket.
2. Menghapus Karakter Tambahan:
    a. Saya menghapus string "#" dari expression yang digunakan, sehingga hanya menampilkan variabel atau nilai dari ticket number saja.
    b. Hal ini dilakukan tanpa mengubah struktur data di backend, hanya memperbaiki tampilan yang muncul di UI.
3. Validasi di Web dan Mobile:
    a. Saya melakukan testing di kedua platform (web dan mobile) untuk memastikan bahwa tidak ada lagi karakter # yang muncul.
    b. Menguji beberapa tipe tiket untuk memastikan konsistensi di semua jenis modul Request & Permit.

3.2.6.3 Hasil yang Dicapai:

1. Tanda # berhasil dihilangkan dari tampilan nomor tiket pada halaman detail.
2. Tampilan menjadi lebih bersih dan sesuai dengan ekspektasi pengguna.
3. Pengguna dapat dengan mudah membaca dan mengidentifikasi nomor tiket tanpa kekeliruan.

4. Masalah di versi Web dan Mobile terselesaikan dengan solusi yang konsisten.

3.2.7 TASK 6 Penambahan Wording pada Validasi Status Nonaktif ke Aktif (Request & Permit)



Gambar 3.23 Tiket Request & Permit – Wording Validasi Status

Gambar 3.23 menampilkan bug terkait wording konfirmasi saat status user berubah dari “Nonaktif” ke “Aktif”. Pesan yang muncul sebelumnya kurang informatif dan dapat menimbulkan kebingungan. Oleh karena itu, perbaikan difokuskan pada pengubahan teks konfirmasi menjadi lebih jelas dan sopan. Gambar 3.23 merupakan dokumentasi penting karena menunjukkan bagaimana komunikasi antarmuka yang baik mempengaruhi pemahaman pengguna. Teks validasi yang dirancang dengan tepat membantu mengurangi kesalahan dan meningkatkan kepercayaan user terhadap sistem.

Task 6 berfokus pada pengubahan wording pada halaman aktivasi ulang akun. Sebelumnya, wording yang digunakan menimbulkan kebingungan karena tidak menjelaskan status akun secara eksplisit. Gambar 3.23 menampilkan tiket

perubahan wording agar pengguna memahami bahwa akun tersebut sebelumnya nonaktif dan telah berhasil diaktifkan kembali.

Pada task keenam ini, saya ditugaskan untuk memperbaiki wording (teks) validasi saat user ingin mengaktifkan kembali akun yang sebelumnya berstatus *Non-Active*. Permintaan berasal dari kebutuhan untuk memperjelas informasi dalam pop-up konfirmasi ketika user melakukan aktivasi akun melalui modul Request & Permit di aplikasi web PMA.

Current Condition (Sebelum Perbaikan):

1. Saat status akun user masih *Non-Active*, sistem menampilkan pop-up konfirmasi aktivasi yang belum memiliki wording yang jelas dan informatif.
2. Teks pada pop-up belum menjelaskan konsekuensi atau proses aktivasi dengan baik.
3. Label status juga masih menggunakan kata “Non-Active” yang dianggap kurang tepat secara bahasa.

Expected Condition (Setelah Perbaikan):

1. Label status diubah dari “Non-Active” menjadi Nonaktif agar sesuai dengan penggunaan bahasa Indonesia yang baik.
2. Pop-up konfirmasi menampilkan kalimat lengkap dan sopan sebagai bentuk validasi keputusan aktivasi user.
3. User diberikan informasi yang lebih ramah dan terarah sebelum sistem melakukan perubahan status.

3.2.7.1 Analsis Permasalahan

Setelah menelusuri halaman Tickets dan logic yang memunculkan pop-up, ditemukan bahwa:

1. Label status menggunakan hardcoded value “Non-Active” tanpa opsi terjemahan.
2. Wording pada pop-up terlalu singkat dan tidak menyampaikan informasi secara utuh kepada user.
3. Tidak ada penyesuaian bahasa antara label dan deskripsi validasi yang muncul, sehingga terasa tidak konsisten.

3.2.7.2 Langkah Perbaikan:

1. Mengubah Label Status:
    a. Saya mengubah teks pada label status dari Non-Active menjadi Nonaktif secara langsung di interface.
    b. Penyesuaian ini dilakukan untuk meningkatkan konsistensi penggunaan bahasa.
2. Memperbaiki Teks Validasi (Pop-up):
    a. Teks lama yang singkat diganti menjadi kalimat penuh seperti:

    “Apakah Anda yakin ingin mengaktifkan kembali akun ini? Setelah diaktifkan, Anda dapat kembali mengakses sistem dan melakukan pengajuan sesuai dengan wewenang yang diberikan.”

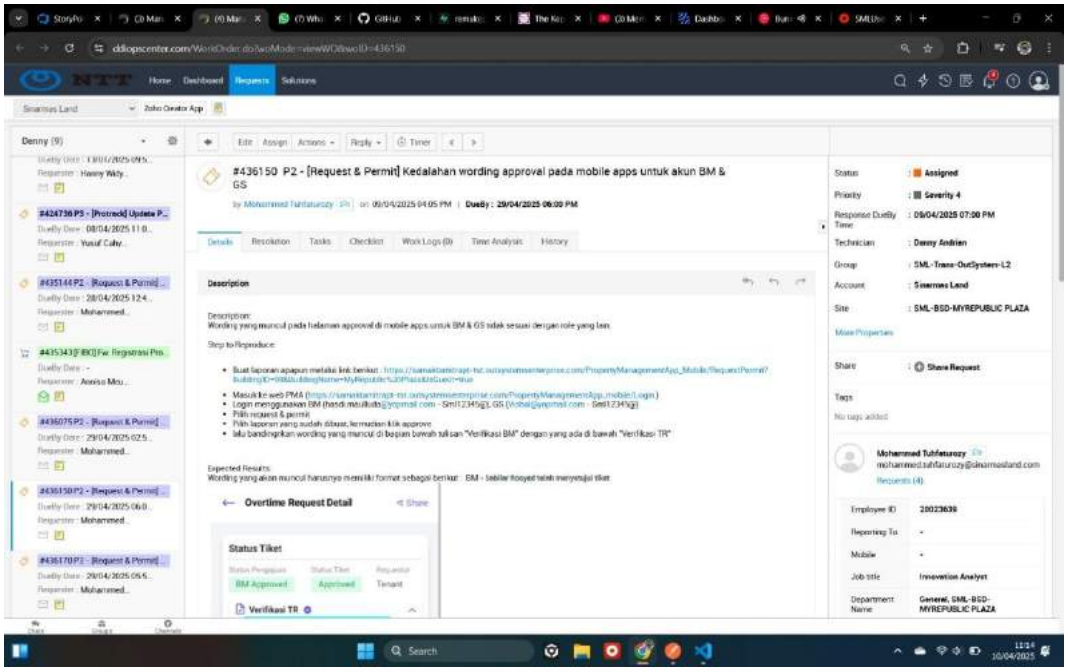
    b. Perubahan ini dilakukan di logic pop-up, pada bagian expression yang menampilkan prompt konfirmasi.
3. Testing & Validasi:
    a. Saya melakukan testing untuk beberapa user dengan status *Nonaktif*.

b. Memastikan bahwa pop-up muncul dengan wording baru, dan proses aktivasi berjalan normal setelah user menekan tombol konfirmasi.

3.2.7.3 Hasil yang Dicapai:

1. Label status berhasil diubah menjadi Nonaktif dengan penyesuaian bahasa yang lebih baik.
2. Wording pop-up konfirmasi kini menjadi lebih jelas, ramah, dan sesuai dengan konteks aktivasi.
3. User dapat memahami maksud konfirmasi aktivasi secara utuh sebelum mengambil keputusan.
4. Perubahan ini meningkatkan user experience serta menghindari potensi kesalahpahaman saat aktivasi akun dilakukan.

3.2.8 TASK 7 Penyesuaian Wording Approval pada Mobile Apps untuk Akun BM & GS

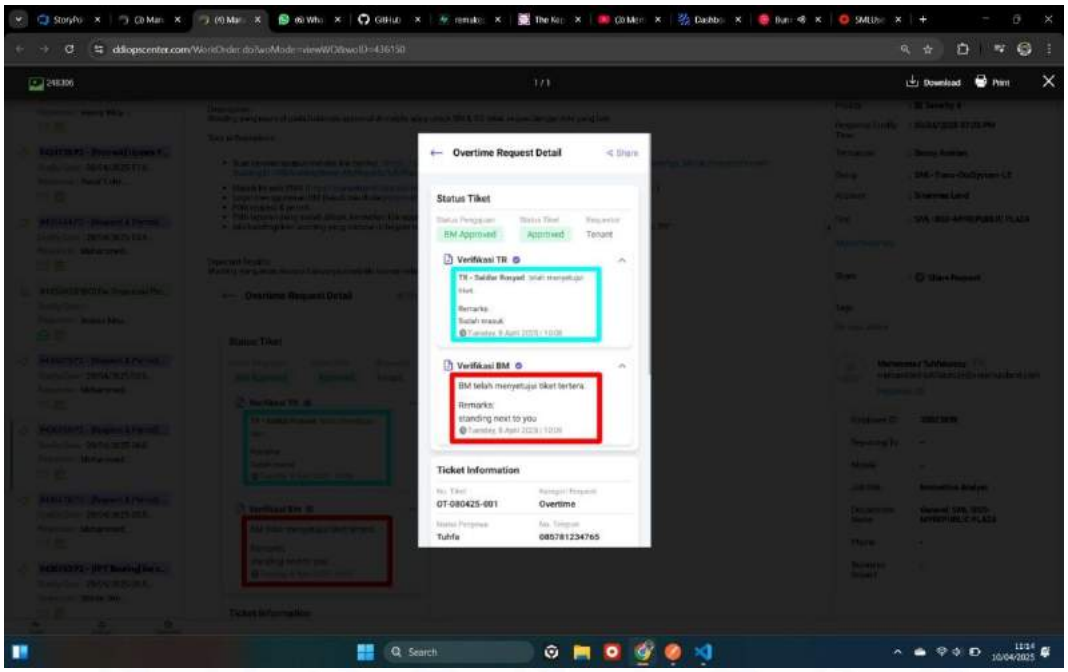


Gambar 3.24 Tiket Approval BM & GS – Wording Tidak Jelas

Gambar 3.24 memperlihatkan laporan bug terkait wording pada proses approval di aplikasi mobile untuk akun Building Manager (BM) dan General Service (GS). Masalahnya adalah sistem tidak mencantumkan nama user yang melakukan approval, sehingga tidak ada kejelasan siapa yang menyetujui tiket. Gambar 3.24 penting karena menunjukkan perlunya transparansi dan akuntabilitas dalam setiap proses verifikasi. Dengan perbaikan wording, proses approval menjadi lebih kredibel dan terdokumentasi dengan jelas.

Detail Permasalahan

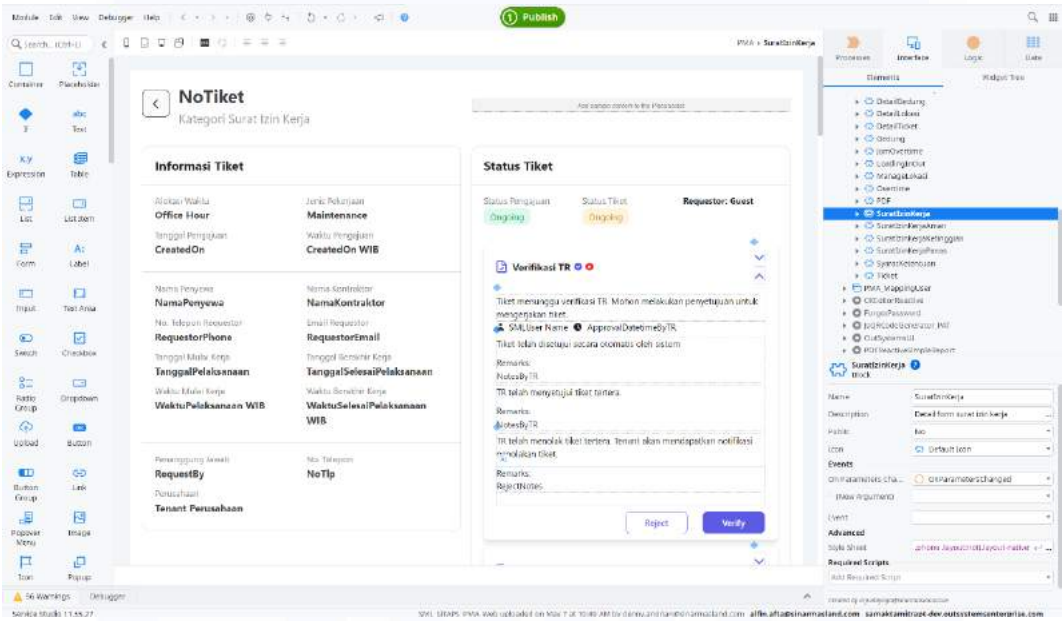
1. Wording approval yang muncul sama untuk semua role, tanpa menyebutkan nama user yang melakukan approval.
2. Contoh: Pada bagian verifikasi BM, sistem hanya menampilkan “BM telah menyetujui tiket tertera” tanpa informasi siapa yang melakukan tindakan.
3. Hal ini menyebabkan tidak ada kejelasan mengenai pihak yang menyetujui tiket pada tampilan mobile.



Gambar 3.25 Detail Tiket – Approval Role BM & GS

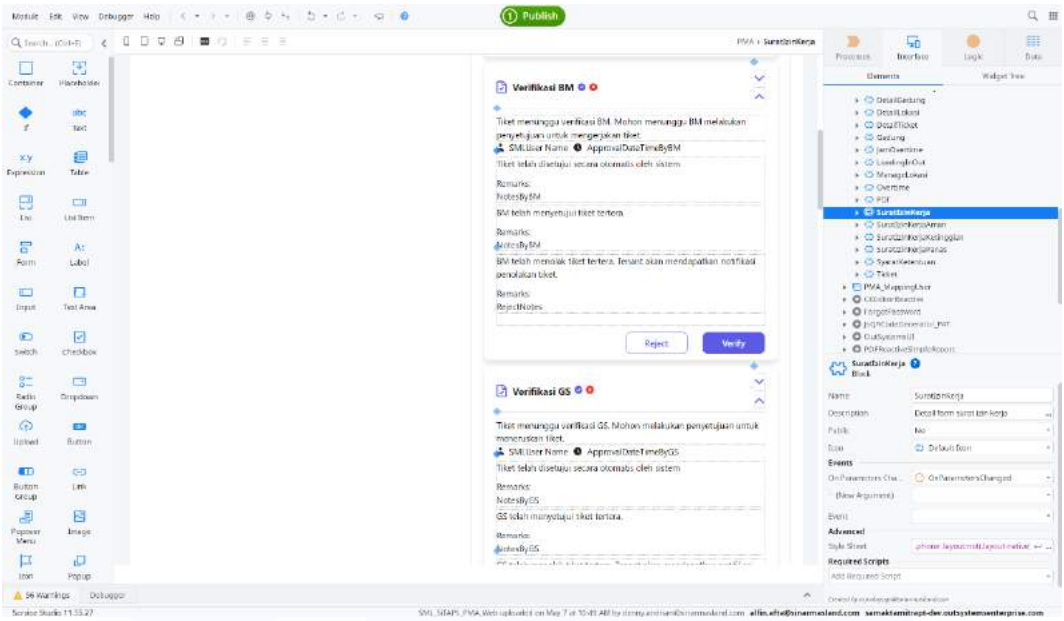
Gambar 3.25 menampilkan tampilan detail tiket setelah wording approval diperbaiki. Informasi yang ditampilkan kini mencantumkan nama pengguna dan peran yang menyetujui, misalnya: “BM – Sabillar Rosyad telah menyetujui tiket ini.” Hal ini membantu tim operasional mengetahui siapa yang bertanggung jawab dalam proses persetujuan. Gambar 3.25 menjadi bukti bahwa tampilan dan wording yang tepat tidak hanya memperindah UI, tetapi juga meningkatkan kejelasan alur kerja dan dokumentasi internal dalam sistem.

3.2.8.1 Analisis dan Perbaikan Bugs Menggunakan OutSystem – Request & Permit



Gambar 3.26 Tampilan Surat Izin Kerja – Format Approval

Gambar 3.26 menunjukkan tampilan surat izin kerja setelah wording approval diperbaiki. Nama user dan peran mereka tercantum secara otomatis di dalam format surat, sehingga meningkatkan profesionalisme dan kejelasan dokumentasi resmi. Gambar 3.26 juga mencerminkan pentingnya konsistensi antara aplikasi dan dokumen yang dihasilkan. Format seperti ini berguna bagi keperluan audit, validasi internal, maupun pelaporan resmi. Perbaikan ini memperkuat akuntabilitas proses dan memperbaiki kualitas komunikasi internal antar departemen.



Gambar 3.27 Verifikasi Dinamis untuk BM dan GS di Mobile App

Gambar 3.27 menampilkan hasil akhir sistem setelah perbaikan wording verifikasi untuk akun BM dan GS diterapkan. Verifikasi kini dilakukan secara dinamis, mencantumkan nama user dan perannya secara otomatis. Gambar 3.27 menunjukkan bahwa sistem telah disempurnakan untuk memberikan informasi yang lengkap dan kontekstual saat proses approval berlangsung. Dengan adanya fitur ini, pengguna dapat mengetahui secara pasti siapa yang melakukan tindakan, meningkatkan transparansi dan akuntabilitas dalam proses digital approval.

Pada Task 7, dilakukan perubahan wording untuk proses approval oleh user dengan role Building Manager (BM) dan General Supervisor (GS). Gambar 3.24 dan 3.25 menampilkan tiket dan detail proses perubahan. Perubahan wording bertujuan agar hasil approval mencantumkan nama user dan peran secara otomatis, sehingga dapat meningkatkan transparansi. Gambar 3.26 menunjukkan tampilan surat izin kerja setelah perubahan wording, sedangkan Gambar 3.27 memperlihatkan hasil akhir sistem yang kini menampilkan nama user secara dinamis saat proses verifikasi oleh BM dan GS.

Setelah menganalisis komponen tampilan dan logika business rule yang ada di mobile app:

1. Teks yang ditampilkan pada elemen Verifikasi BM tidak dinamis dan tidak mengacu pada data user yang melakukan approval.
2. Template teks ditulis statis di dalam UI, bukan mengambil nama user dari variable atau database.

3.2.8.2 Langkah Perbaikan

1. Identifikasi Komponen UI yang Terlibat

   Menemukan bagian pada halaman Overtime Request Detail mobile app tempat wording approval ditampilkan.

2. Modifikasi Format Wording

   a. Mengubah wording statis menjadi dinamis dengan menyisipkan nama user.

   b. Format baru: [Role] - [Nama User] telah menyetujui tiket tertera Contoh hasil: BM - Sabillar Rosyad telah menyetujui tiket tertera.

3. Update Expression di Tampilan

   Menggunakan expression atau assign logic untuk membentuk kalimat dinamis berdasarkan:

   a) Role user

   b) Nama user yang sedang login

   c) Tanggal dan jam approval

4. Uji Coba dan Validasi

    a. Melakukan testing menggunakan akun BM dan GS untuk memastikan wording muncul sesuai.

    b. Memastikan format yang tampil pada mobile mengikuti struktur

3.2.8.3 Hasil yang Dicapai

1. Wording pada approval di mobile app berhasil diperbaiki sesuai role masing-masing.
2. User kini dapat mengetahui siapa yang menyetujui tiket, meningkatkan transparansi proses approval.
3. Format kalimat menjadi lebih informatif dan akurat.
4. Validasi dilakukan pada berbagai role (BM & GS) dan hasilnya tampil sesuai ekspektasi.

3.3 Kendala yang Ditemukan

1. Kurangnya Pemahaman Awal tentang OutSystems
   Selama magang, salah satu kendala utama yang dihadapi adalah minimnya pemahaman mengenai platform OutSystems, karena sebelumnya belum pernah digunakan secara langsung dalam kegiatan akademik. Hal ini membuat proses adaptasi memerlukan waktu tambahan, terutama dalam memahami struktur modul, logika screen, dan penggunaan komponen-komponen UI seperti dropdown, switch, dan block reusable.
2. Keterbatasan Akses terhadap Dokumentasi Teknis Aplikasi
   Sebagian besar aplikasi yang dikelola telah dibangun cukup kompleks dan tidak memiliki dokumentasi teknis yang lengkap, seperti penjelasan flow logic, dependensi antar modul, atau struktur data relasional. Akibatnya, perlu dilakukan tracing secara manual terhadap proses logic dan query untuk memahami letak error maupun bug.

3.4 Solusi atas Kendala yang Ditemukan

1. Untuk mengatasi kendala ini, saya mulai dengan mempelajari dasar-dasar OutSystems melalui dokumentasi resmi dan tutorial yang tersedia di OutSystems Learn. Saya juga mengikuti beberapa video pembelajaran di YouTube dan forum komunitas OutSystems untuk memahami cara kerja komponen-komponen umum seperti dropdown, expression, dan logic pada screen. Selain itu, saya secara aktif bertanya kepada rekan tim dan supervisor saat menemui kesulitan teknis. Dengan pendekatan ini, saya bisa lebih cepat memahami struktur aplikasi dan menyelesaikan task-task yang diberikan secara mandiri.
2. Karena tidak tersedia dokumentasi teknis yang lengkap, saya mengambil inisiatif untuk melakukan eksplorasi mandiri terhadap struktur modul dan proses logic yang ada di dalam aplikasi. Saya memanfaatkan fitur *Debugger* dan *Data Preview* di OutSystems untuk memahami alur data dan mengetahui hasil dari masing-masing query atau action. Selain itu, saya membuat catatan pribadi berupa diagram alur dan penjelasan singkat setiap flow yang telah saya telusuri, agar mempermudah saya saat harus melakukan perubahan atau revisi di bagian yang sama di kemudian hari.