BAB III
PELAKSANAAN KERJA

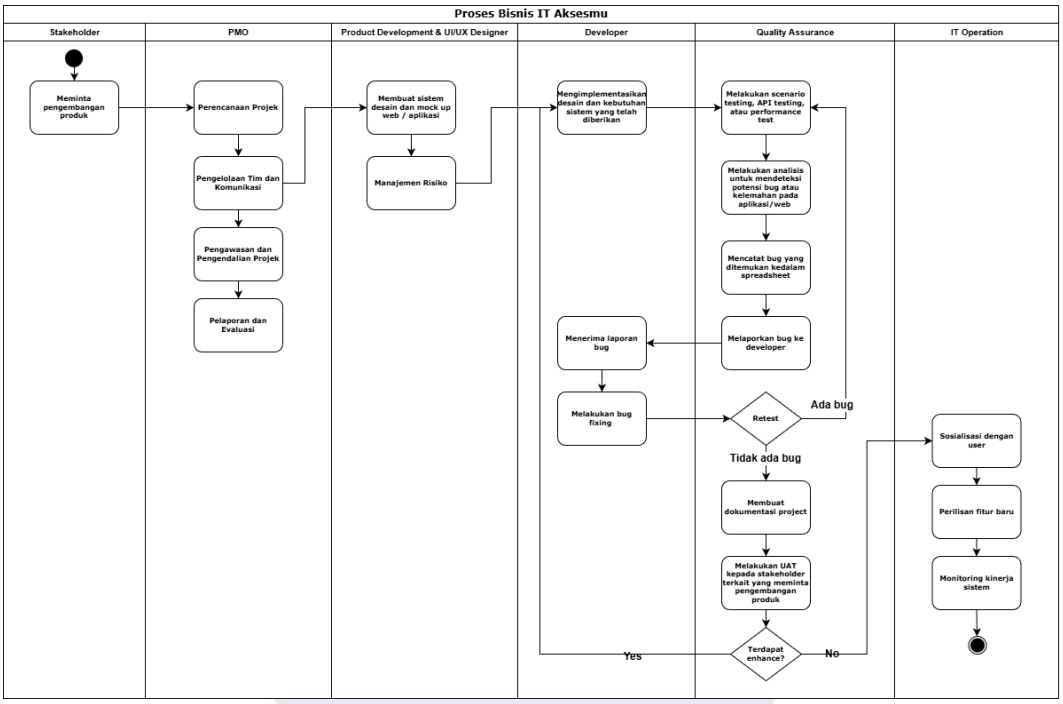
3.1 Kedudukan dan Koordinasi

3.1.1 Kedudukan

Posisi mahasiswa magang di PT Sumber Trijaya Lestari (Aksesmu) sebagai *Quality Assurance* yang berada pada departemen *IT Tech Product*. Mahasiswa magang berada di bawah bagian yang mengawasi pengembangan produk teknologi perusahaan. Mahasiswa magang bertanggung jawab dan berkolaborasi secara langsung di bawah pembimbing serta berinteraksi secara aktif dengan berbagai fungsi inti tim pengembangan produk.

Sebagai *Quality Assurance Internship*, kontribusi mahasiswa magang sangat krusial dalam memastikan kualitas aplikasi Aksesmu. Mahasiswa magang berpartisipasi aktif dalam siklus pengembangan sistem, bekerja sama secara dekat dengan tim *Developer* untuk memastikan perbaikan *bug* dilaksanakan dengan lengkap, serta berkolaborasi dengan tim *Product* untuk memverifikasi bahwa setiap fitur yang dibuat telah memenuhi kebutuhan fungsionalitas dan kriteria penerimaan yang ditentukan.

3.1.2 Koordinasi



Gambar 3. 1 Bagan Alur Koordinasi

Berdasarkan dari gambar 3.1 bisnis proses IT diatas, projek-projek yang terdapat dalam IT Aksesmu didapatkan dari keluhan atau evaluasi *stakeholder* ketika menggunakan aplikasi atau website Aksesmu. *Stakeholder* akan menjelaskan keluhan kepada *Project Management Officer* (PMO) ketika menggunakan aplikasi atau *website* Aksesmu. PMO akan merangkum projek-projek yang akan dijalankan oleh tim IT dan mendiskusikan projek tersebut kepada *product development* dan *UI/UX designer* untuk mencari solusi bagi aplikasi yang lebih baik. *UI/UX designer* akan membuat desain yang akan dirancang nantinya. Ketika sudah didiskusikan *product development* akan membuat dokumen presentasi mengenai projek yang akan didevelop yang berisi perencanaan rancangan database, desain, dan sebagainya.

Pada saat *developer* sudah mengimplementasikan ke aplikasi *staging*, *Quality Assurance* akan membuat skenario yang akan digunakan untuk pengujian secara manual dan otomatis sehingga kualitas aplikasi atau website sesuai standar yang diperlukan. Jika terdapat *bug*, *Quality Assurance* akan mencatat *bug* atau masalah yang ditemukan dari aplikasi atau website yang

sudah dibuat. Masalah-masalah tersebut akan dilaporkan kembali kepada *developer* dan *developer* akan melakukan *bug fixing* terhadap masalah yang ditemukan. Jika sudah, QA akan melakukan pengujan dari awal hingga akhir, dan jika tidak terdapat *bug* atau masalah *Quality Assurance* bisa melakukan *User Acceptance Testing* (UAT) dengan tim IT dan *stakeholder* yang meminta pengembangan produk. UAT adalah pengujian yang dilakukan oleh pengguna akhir, yang dapat berupa karyawan atau karyawan perusahaan yang berinteraksi secara langsung dengan sistem [11].

Akhirnya, sistem menuju tahap Validasi dan Operasi. Setelah mengatasi *bug*, *developer* menyelesaikan dokumentasi proyek. QA selanjutnya mengorganisir UAT bersama *stakeholder* untuk memastikan bahwa fitur yang dibuat benar-benar sesuai dengan kebutuhan bisnis. Setelah UAT disetujui tanpa permintaan perubahan signifikan, tanggung jawab berpindah ke *IT Operation*. Tim ini bertanggung jawab untuk melakukan Sosialisasi dengan pengguna, melaksanakan Perilisan fitur baru, serta melaksanakan *monitoring* kinerja sistem secara terus-menerus untuk memastikan sistem beroperasi dengan stabil dan efisien setelah peluncuran.

3.2 Tugas yang Dilakukan

Berisi tabel hal-hal yang mahasiswa magang lakukan selama menjalankan program.

Tabel 3. 1 Detail Pekerjaan yang Dilakukan

No.	Kegiatan	Tanggal Mulai	Tanggal Selesai
1.	**Surat Jalan v 0.75**	18 Juni 2025	4 Juni2025
1.a	Analisa sistem desain dan memahami struktur data	18 Juni 2025	18 Juni 2025
1.b	Membuat skenario pengujian Surat Jalan v 0.75	18 Juni 2025	19 Juni 2025

1.c	Pengujian projek Surat Jalan v 0.75 pada aplikasi Aksesmu Management System (AMS)	19 Juni 2025	4 Agustus 2025
1.d	Mencatat *bug* pada laporan pengujian	19 Juni 2025	4 Agustus 2025
1.e	Melaporkan *bug* kepada *developer*	19 Juni 2025	4 Agustus 2025
1.f	Membuat dokumentasi	19 Juni 2025	1 Agustus 2025
1.g	Presentasi *User Acceptance Testing* (UAT) dengan *user*	19 Juni 2025	3 Agustus 2025
1.h	Simulasi pengiriman menggunakan B-Log pada SPG Cikokol	4 Agustus 2025	4 Agustus 2025
2.	**Rencana Realisasi Anggaran Kas (RRAK)**	29 Juli 2025	29 Agustus 2025
2.a	*Meeting* dengan *department finance* dan *operation* untuk membahas *enhancement* website Rencana Realisasi Anggaran Kas (RRAK)	29 Juli 2025	29 Juli 2025
2.b	Membuat skenario pengujian Rencana Realisasi Anggaran Kas (RRAK)	29 Juli 2025	31 Juli 2025
2.c	Pengujian fitur realisasi pada website Rencana Realisasi Anggaran Kas (RRAK)	29 Juli 2025	28 Agustus 2025
2.d	Mencatat *bug* pada laporan pengujian	29 Juli 2025	28 Agustus 2025
2.e	Melaporkan *bug* kepada *developer*	29 Juli 2025	28 Agustus 2025
2.f	Membuat dokumentasi	29 Juli 2025	28 Agustus 2025
2.g	Sosialisasi dengan user mengenai *enhancement* website Rencana Realisasi Anggaran Kas (RRAK)	29 Agustus 2025	29 Agustus 2025
3.	**Permintaan Barang Manual**	11 Agustus 2025	10 Oktober 2025
3.a	Analisa sistem desain dan memahami struktur data	11 Agustus 2025	11 Agustus 2025

3.b	Membuat skenario pengujian Permintaan Barang Manual	11 Agustus 2025	15 Agustus 2025
3.c	Pengujian fitur Permintaan Barang Manual pada aplikasi Aksesmu Management Dashboard (AMD)	11 Agustus 2025	9 Oktober 2025
3.d	Mencatat *bug* pada laporan pengujian	11 Agustus 2025	9 Oktober 2025
3.e	Melaporkan *bug* kepada *developer*	11 Agustus 2025	9 Oktober 2025
3.f	Membuat dokumentasi	11 Agustus 2025	9 Oktober 2025
3.g	Presentasi *User Acceptance Testing* (UAT) dengan *user*	11 Agustus 2025	10 Oktober 2025
4.	***Failed Registration Log***	10 Oktober 2025	17 Oktober 2025
4.a	Membuat skenario pengujian *Failed Registration Log*	10 Oktober 2025	10 Oktober 2025
4.b	Pengujian fitur *Failed Registration Log* pada aplikasi Aksesmu Management System (AMS)	10 Oktober 2025	15 Oktober 2025
4.c	Mencatat *bug* pada laporan pengujian	10 Oktober 2025	15 Oktober 2025
4.d	Melaporkan *bug* kepada *developer*	10 Oktober 2025	15 Oktober 2025
4.e	Membuat dokumentasi	15 Oktober 2025	17 Oktober 2025
5.	**Pengkinian Data Bayar Tunda**	20 Oktober 2025	22 Oktober 2025
5.a	Analisa sistem desain dan memahami struktur data	20 Oktober 2025	20 Oktober 2025
5.b	Membuat skenario pengujian Data Pengkinian Bayar Tunda	20 Oktober 2025	20 Oktober 2025
5.c	Pengujian fitur Pengkinian Data Bayar Tunda pada aplikasi Aksesmu	20 Oktober 2025	22 Oktober 2025
5.d	Melaporkan *bug* kepada *developer*	20 Oktober 2025	22 Oktober 2025
5.e	Membuat dokumentasi	20 Oktober 2025	22 Oktober 2025

6.	***Bulk data order***	23 Oktober 2025	14 November 2025
6.a	*Bulk data order*	23 Oktober 2025	14 November 2025

Tabel 3.1 berisi sebagai struktur rincian dan kerangka waktu yang komprehensif yang berisi tanggal mulai dan tanggal selesai untuk setiap kegiatan yang dilakukan selama periode magang. Secara keseluruhan, tabel ini berisi siklus pengembangan aplikasi dan website Aksesmu, di mana aktivitas proyek terlama yang tercatat adalah Rencana Realisasi Anggaran Kas (RRAK) yang berjalan hampir satu bulan penuh dari 29 Juli 2025 hingga 29 Agustus 2025. Analisa sistem desain pada Surat Jalan v 0.75 dan Pengkinian Data Bayar Tunda merupakan aktivitas yang memiliki durasi yang sangat singkat dengan membutuhkan waktu satu hari. Pekerjaan yang dilakukan mencakup siklus pengembangan perangkat lunak yang lengkap, mulai dari tahap perencanaan hingga tahap penyerahan.

Proyek Surat Jalan v 0.75 dimulai dari 18 Juni 2025 sampai 4 Agustus 2025. Projek ini merupakan proyek yang paling terstruktur, meliputi seluruh tahapan dari analisa sistem desain, pembuatan skenario, pengujian pada aplikasi Aksesmu Management System (AMS), pencatatan dan pelaporan *bug*, dokumentasi, hingga Presentasi *User Acceptance Testing* (UAT) dan simulasi pengiriman. Selain itu, proyek Permintaan Barang Manual yang dimulai dari 11 Agustus 2025 sampai 10 Oktober 2025. Projek ini memiliki rentang waktu pengujian terpanjang di Aksesmu Management Dashboard (AMD) dengan mencakup tahapan analisa, pembuatan skenario, pengujian, pelaporan *bug*, dokumentasi, dan Presentasi UAT.

Proyek-proyek lainnya memiliki fokus dan durasi yang lebih singkat. Proyek Pengkinian Data Bayar Tunda yang dimulai dari 20 Oktober 2025 sampai 22 Oktober 2025 adalah salah satu proyek tercepat dengan durasi hanya tiga hari, sementara *Bulk data order* merupakan kegiatan terpisah yang dilaksanakan pada periode akhir, berlangsung dari 23 Oktober 2025 hingga 14 November 2025. Konsistensi tahapan kerja yaitu pembuatan skenario, pengujian, pencatatan *bug*, pelaporan *bug*, dan dokumentasi terlihat pada sebagian besar proyek, termasuk RRAK dan *Failed Registration Log* yang menunjukkan sebuah alur kerja yang standar dan terorganisir.

3.3 Uraian Pelaksanaan Kerja

Quality Assurance (QA) memegang peranan esensial dalam menjaga kualitas produk agar tetap konsisten terhadap standar operasional perusahaan. Fokus utama dari fungsi ini adalah melakukan mitigasi atau pencegahan defek pada aplikasi maupun situs web melalui rangkaian proses pengujian yang terstruktur [12]. Dalam ekosistem pengembangan aplikasi digital, lingkup tanggung jawab penjaminan kualitas mencakup perancangan sistem pelaporan yang terintegrasi, penyediaan dasbor pemantauan untuk evaluasi, pengelolaan alur kerja validasi data, hingga otomatisasi prosedur audit [13]. Implementasi sistem digital tersebut diharapkan mampu mengoptimalkan efisiensi dan akuntabilitas kerja, di mana seluruh data penilaian tersimpan secara terverifikasi sehingga dapat diaudit kembali oleh para pemangku kepentingan (*stakeholder*) secara langsung [14].

Selama periode magang di Aksesmu, kontribusi yang dilakukan sebagai QA mencakup serangkaian prosedur teknis yang dimulai dari penyusunan skenario pengujian positif dan negatif pada *spreadsheet* dengan merujuk pada sistem desain milik tim *product development*. Skenario tersebut kemudian digunakan sebagai landasan dalam melakukan pengujian aplikasi dan situs web guna memastikan efektivitas sistem. Selain itu, dilakukan pula validasi data melalui pengecekan basis data menggunakan DBeaver, intervensi pada database *staging*, serta pengujian API menggunakan Postman. Pengelolaan data lebih lanjut dilakukan dengan menggunakan Redis Insight melalui mekanisme *get data* dan *post data*. Seluruh temuan kendala teknis atau *bug* dicatat secara sistematis pada *spreadsheet* yang dibarengi dengan pembuatan dokumentasi hasil pengujian yang komprehensif. Tahap akhir dari kegiatan ini melibatkan pelaporan hasil kerja kepada atasan, koordinasi intensif dengan tim *IT Operation* selama proses pengujian, serta pelaksanaan *User Acceptance Testing* (UAT) bersama *stakeholder* terkait setelah seluruh tahapan pengujian dan dokumentasi dinyatakan tuntas.

3.3.1 Proses Pelaksanaan

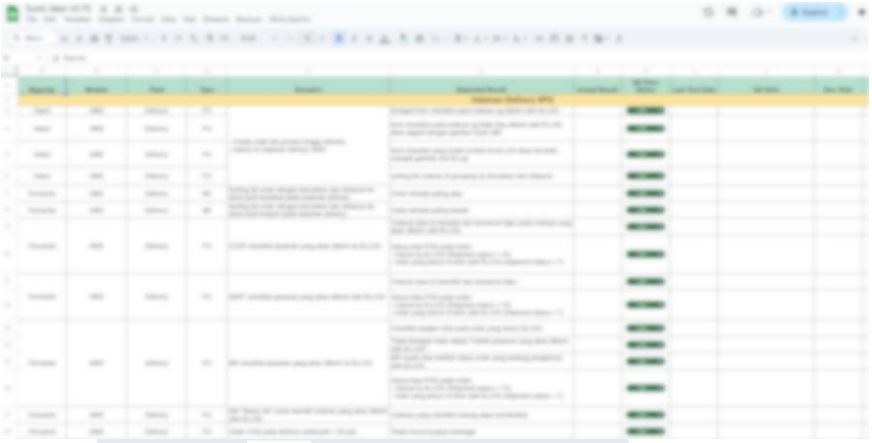
3.3.1.1 Surat Jalan v 0.75

a. Analisa sistem desain dan memahami struktur data

Gambar 3. 2 Memahami Sistem Desain

Berdasarkan gambar 3.2 dalam memastikan pemahaman teknis dan fungsional yang menyeluruh sebelum pengembangan, tahap pemahaman sistem desain untuk fitur Surat Jalan versi 0.75 adalah langkah penting. Pada tahap pertama, memahami *scoop* proyek dan peraturan bisnis. Pada tahap kedua, memahami struktur database untuk menghubungkan entitas data Surat Jalan dan hubungannya. Tahap ketiga adalah memeriksa tampilan fitur surat jalan untuk memverifikasi antarmuka pengguna. Hal ini terutama berkaitan dengan tampilan peringatan keamanan dan mekanisme *checklist* manual oleh *Chief of Stock Point* (COSP). Pesanan *Cash on Delivery* (COD) sangat penting dan membutuhkan dukungan dari *Member Relation* (MR) dan terakhir, menggambarkan alur aplikasi secara menyeluruh untuk menggambarkan urutan tindakan yang dilakukan pengguna, mulai dari pemilihan pesanan hingga pencetakan Surat Jalan. Hal ini memastikan alur kerja yang efisien dan sesuai dengan perubahan otorisasi yang baru.

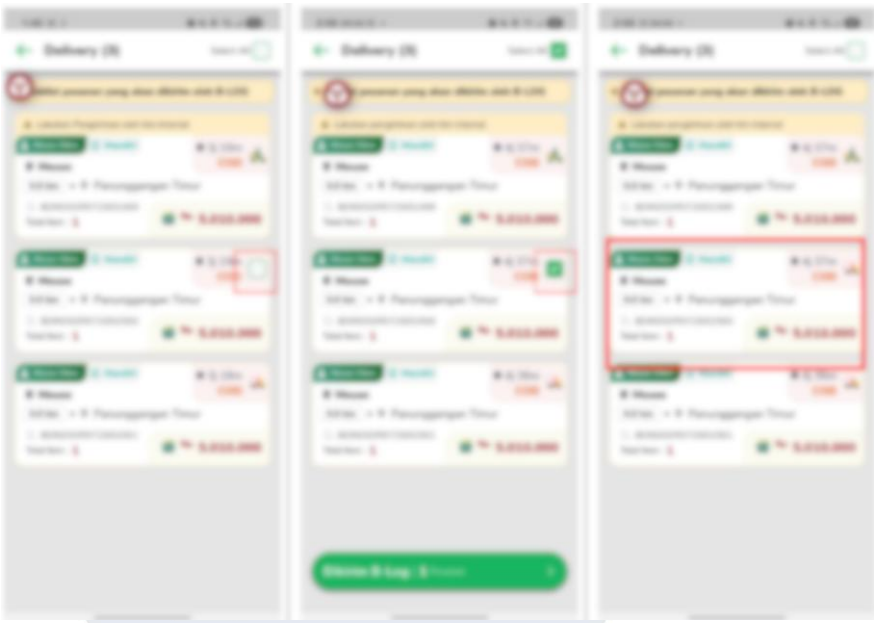
b. Membuat skenario *pengujian* Surat Jalan v 0.75



Gambar 3. 3 Membuat Skenario

Berdasarkan gambar 3.3 dalam memastikan bahwa fungsionalitas dan logika bisnis pada fitur Surat Jalan v 0.75, tahap membuat skenario Surat Jalan sangat penting. Tahap ini mencakup pembuatan sejumlah besar kasus uji. Perumusan skenario positif dan negatif termasuk semua kombinasi penting, seperti pengiriman melalui B-Log atau MR, pengujian otorisasi COSP dalam pemilihan manual, dan pengujian sistem peringatan untuk pesanan COD bernilai tinggi yang memerlukan pendampingan MR. Hasil dari proses ini adalah dokumen terperinci yang berisi skenario ujian dan hasil yang diharapkan, yang berfungsi sebagai panduan utama untuk melaksanakan pengujian sistem. QA telah menyelesaikan seluruh pembuatan dan validasi pada total 52 skenario uji yang mencakup alur pengiriman dengan hasil akhir tingkat kelulusan 100 persen karena seluruh skenario sudah berstatus OK.

c. Pengujian projek Surat Jalan v 0.75 pada aplikasi Aksesmu Management System (AMS)



Gambar 3. 4 Pengujian

Berdasarkan gambar 3.4 tujuan dari tahap pengujian adalah untuk menerapkan skenario uji secara langsung dalam memverifikasi secara menyeluruh fungsionalitas fitur Surat Jalan versi 0.75. Tahap ini terutama berfokus pada validasi fungsionalitas inti, verifikasi alur otorisasi, dan penanganan risiko. Dalam proses pengujian ini, COSP memilih pesanan yang akan dikirim dengan petugas dan B-Log dengan menggunakan tombol *checklist* yang tersedia, dan secara khusus memverifikasi sistem peringatan keamanan yang menunjukkan bahwa pesanan COD telah melebihi batas nilai tertentu dan memerlukan pendampingan MR. Proses pelaksanaan pengujian di lingkungan staging memerlukan waktu eksekusi yang bervariasi antara 2 hingga 10 menit untuk setiap satu skenario pengujian.

d. Mencatat *bug* pada laporan pengujian

Gambar 3. 5 Mencatat *Bug*

Berdasarkan gambar 3.5 setelah proses pengujian, tahap mencatat *bug* mencatat setiap ketidaksesuaian dengan hasil yang diharapkan. Pencatatan ini sangat penting untuk memastikan bahwa masalah yang ditemukan baik terkait dengan proses pemilihan pesanan manual oleh COSP, validasi database, dan mekanisme peringatan pendampingan MR dapat diperbaiki dan diperbarui oleh tim pengembang sebelum dirilis. Ini menjamin kualitas akhir fitur Surat Jalan. Berdasarkan hasil pengujian yang telah dilakukan, tim penguji mencatatkan total sebanyak 29 temuan *bug* yang teridentifikasi dalam daftar temuan *bug* untuk modul pengiriman tersebut, dengan pembagian prioritas yang terdiri dari 11 *high*, 9 *medium*, dan 9 *low*. Beberapa temuan utama mencakup masalah teknis seperti respon halaman pengambilan data yang bernilai *null*, status proses pengantaran oleh B-Log yang tidak muncul saat pengiriman sedang berlangsung, hingga kendala pada fungsi pemilihan semua pesanan di mana sistem hanya berhasil memilih 10 pesanan saja.

e. Melaporkan *bug* kepada *developer*

Langkah penting dalam siklus pengembangan fitur adalah melaporkan *bug*, yang memastikan kualitas dan stabilitas produk. Selain memberikan informasi tentang masalah, proses ini harus dilakukan secara sistematis dan informatif agar pengembang dapat mereplikasi, menemukan sumber masalah, dan memperbaikinya dengan cepat. Laporan *bug* yang efektif minimal harus mencakup deskripsi singkat masalah, prosedur pasti untuk mereplikasi *bug*, hasil yang diharapkan, dan hasil yang sebenarnya. Dalam memberikan konteks yang lengkap, juga penting untuk menyertakan tangkapan layar atau rekaman video. Seluruh 29 temuan *bug* yang telah dilaporkan kepada tim pengembang kini sudah berstatus OK karena seluruhnya telah berhasil diperbaiki dan diselesaikan secara tuntas.

f. Membuat dokumentasi

Gambar 3. 6 Membuat Dokumentasi

Berdasarkan gambar 3.6 dalam siklus pengembangan fitur Surat Jalan v 0.75, tahap dokumentasi adalah tahap sangat penting. Pada tahap ini, semua elemen teknis dan fungsional didokumentasikan secara formal. Panduan langkah-langkah COSP untuk memilih pesanan yang akan dikirim menggunakan B-Log dan petugas dokumentasi surat jalan versi 0.75 yang mencakup latar belakang, alur sistem, aturan bisnis, serta dokumentasi basis data. Dalam proses ini, tim juga merangkum hasil pengujian terhadap 29 temuan *bug* yang telah berstatus ok, termasuk perbaikan pada masalah teknis seperti respon halaman pengambilan data yang bernilai kosong, kendala status pengantaran oleh B-Log, hingga fungsi pemilihan pesanan yang sebelumnya terbatas. Seluruh skenario pengujian penting seperti

penggunaan B-Log untuk peran COSP dan SMR di Toko SPG, prosedur bukti pengiriman, serta aturan khusus untuk pesanan pembayaran di tempat senilai 25 juta rupiah ke atas telah terdokumentasi dengan lengkap sebagai acuan operasional sistem.

g. Presentasi *User Acceptance Testing* (UAT) dengan *user*

Presentasi *User Acceptance Testing* (UAT) dengan *user* adalah tahap krusial yang berfungsi sebagai validasi akhir untuk memastikan fitur yang telah dikembangkan seperti alur pengiriman menggunakan B-Log yang akan disimulasikan langsung di lokasi *Stock Point* (SPG) Cikokol dan memastikan telah memenuhi kebutuhan bisnis dan diterima sepenuhnya oleh pengguna akhir. Dalam tahap ini, presentasi UAT ini dilakukan di depan beberapa anggota departemen seperti *General Affair* dan *Operation* untuk mendapatkan *feedback* langsung. Dari UAT tersebut ditemukan adanya kebutuhan revisi seperti penyesuaian *blacklist* akses jalan yang tidak bisa dilalui saat pengiriman dan titik lokasi tujuan yang tidak sesuai dalam pengiriman menggunakan B-Log. Tujuannya adalah untuk mengumpulkan umpan balik formal dari *stakeholder* dan *user* guna memastikan bahwa solusi tersebut akurat, terkini, dan siap untuk diluncurkan ke lingkungan produksi, sekaligus menjadi referensi bagi *developer* atau *stakeholder* terkait logika bisnis baru.

h. Simulasi pengiriman menggunakan B-Log pada SPG Cikokol

Gambar 3. 7 Simulasi pengiriman menggunakan B-Log pada SPG Cikokol

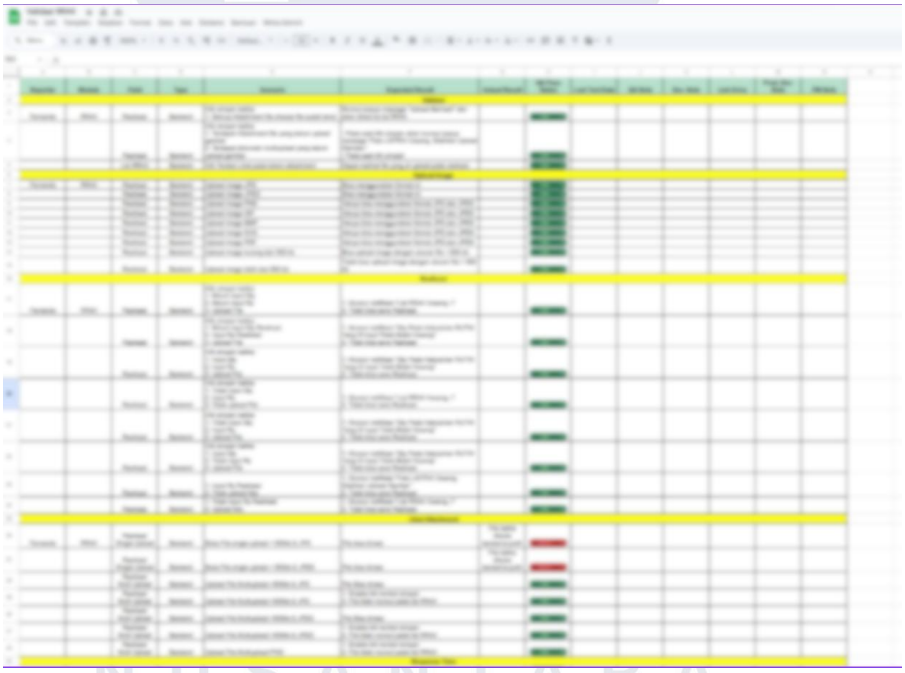
Berdasarkan gambar 3.7 merupakan tahap simulasi implementasi akhir berguna untuk menguji kesiapan fitur Surat Jalan v 0.75 secara langsung di salah satu SPG Stock Point di Cikokol. Pada awalnya, tim pengembang bekerja sama dengan COSP dan sopir B-Log yang berada di SPG Cikokol untuk membantu mengelola seluruh alur pengiriman menggunakan B-Log, menggunakan fitur baru seperti memverifikasi mekanisme peringatan keamanan, memilih pesanan secara manual, mencetak *waybill*, dan menyelesaikan proses serah terima. Sebelum fitur ini secara resmi diluncurkan di seluruh lokasi, tujuannya adalah untuk memastikan bahwa perubahan yang dilakukan mampu mengatasi pengiriman di lapangan dan bahwa staf SPG dapat menjalankan alur kerja baru dengan lancar dan efisien.

3.3.1.2 RRAK (Rencana Realisasi Anggaran Kas)

a. *Meeting* dengan department *finance* dan *operation* untuk membahas *enhancement* website Rencana Realisasi Anggaran Kas (RRAK)

Meeting ini membahas masalah yang dihadapi oleh *department finance* dan *operation* untuk membahas *enhancement* website RRAK. Projek ini berguna untuk memastikan bahwa data dan dokumen pendukung yang diunggah dalam sistem telah sesuai dengan standar administrasi dan teknis yang berlaku. Pada bagian realisasi pengguna diminta untuk input anggaran dan bukti fisik yang berisi anggaran yang diajukan. Validasi input gambar ini bertujuan untuk meminimalisir risiko pemalsuan dokumen, dan meningkatkan akuntabilitas pelaporan data.

b. Membuat skenario pengujian Rencana Realisasi Anggaran Kas (RRAK)

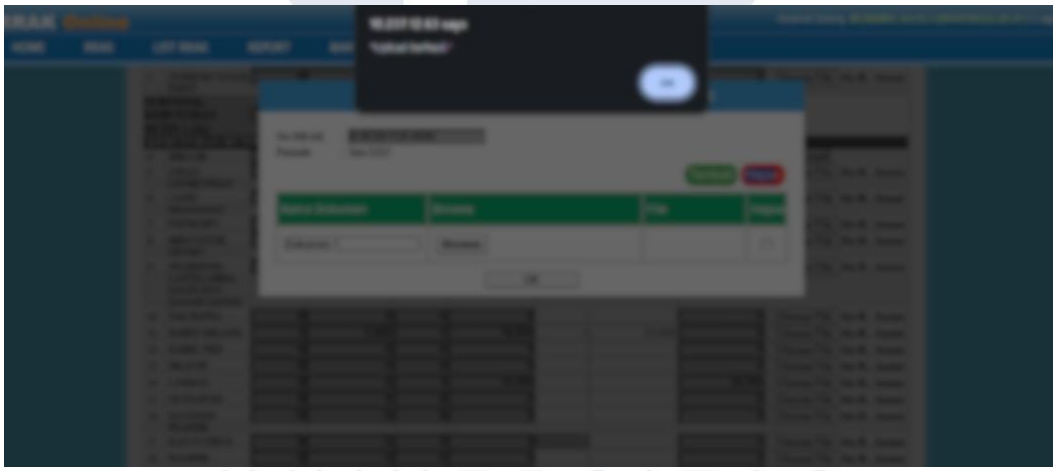


Gambar 3. 8 Membuat Skenario

Berdasarkan gambar 3.8 *Rencana Realisasi Anggaran Kas* (RRAK) ini berkonsentrasi pada tahapan validasi yang bertujuan untuk memastikan bahwa dokumen anggaran dan realisasi yang dimasukkan ke sistem adalah akurat dan sah. Proses validasi sangat penting untuk mengurangi kemungkinan dokumen

dipalsukan dan meningkatkan akuntabilitas, terutama untuk file bukti fisik seperti gambar. Kebijakan bisnis yang ketat digunakan untuk validasi, termasuk format file hanya jpg atau jpeg, ukuran file maksimal 500 kb, dan validasi nilai realisasi Rupiah yang harus lebih dari nol. Dalam memastikan kualitas, seluruh proses ini mencakup pembuatan skenario, dokumentasi pemeriksaan kualitas, dan pengujian pemeriksaan. Proses ini memverifikasi alur unggah, *preview* gambar, verifikasi manual oleh admin, dan pencatatan aktivitas di *log* sistem untuk memastikan bahwa setiap pengajuan anggaran dan pelaksanaan telah memenuhi standar administrasi dan teknis yang berlaku. Projek ini memiliki total 39 skenario pengujian yang mencakup fungsionalitas validasi, unggah gambar, realisasi, serta hak akses peran pengguna dengan hasil akhir tingkat kelulusan 100 persen karena seluruh skenario sudah berstatus ok.

c. Pengujian fitur realisasi pada website Rencana Realisasi Anggaran Kas (RRAK)



Gambar 3. 9 Pengujian

Berdasarkan gambar 3.9 tahap ini adalah pelaksanaan skenario uji untuk memverifikasi fungsionalitas validasi RRAK yang telah dikembangkan. Pengujian ini berfokus pada eksekusi semua skenario yang telah dirancang untuk memastikan bahwa aturan bisnis diterapkan dengan benar. Proses ini memverifikasi alur *user interface* saat pengguna mengunggah dokumen realisasi, memastikan sistem memberikan notifikasi kesalahan yang tepat, dan menguji fungsionalitas melihat gambar untuk memfasilitasi verifikasi manual. Pengujian ini vital untuk menjamin

bahwa proses validasi input data berjalan sesuai harapan sehingga meminimalkan risiko kesalahan dan menjaga akuntabilitas pelaporan anggaran. Proses pelaksanaan pengujian di lingkungan *staging* memerlukan waktu eksekusi yang bervariasi antara 2 hingga 20 menit untuk setiap satu skenario pengujian.

d. Mencatat *bug* pada laporan pengujian

Gambar 3. 10 Mencatat *Bug*

Berdasarkan gambar 3.10 tahap mencatat *bug* pengujian RRAK adalah proses dokumentasi yang sistematis terhadap semua *defect* atau penyimpangan yang ditemukan selama pelaksanaan Pengujian QA. Berdasarkan hasil pengujian yang telah dilakukan pada proyek RRAK, tim penguji mencatatkan total sebanyak 6 temuan *bug* yang teridentifikasi dalam daftar temuan *bug* proyek tersebut dengan pembagian prioritas yang terdiri dari 1 *high*, 1 *medium*, dan 4 *low*. Beberapa temuan tersebut mencakup kendala pada tampilan file yang menjadi putih saat dibuka setelah diunggah secara tunggal, serta isu pada fitur unggah file di mana ukuran file lebih dari 500kb masih dapat tersimpan dalam lampiran.

e. Melaporkan *bug* kepada *developer*

Melaporkan *bug* adalah bagian penting dari siklus pengembangan fitur karena memastikan kualitas dan stabilitas produk. Proses ini harus dilakukan secara sistematis dan informatif selain memberikan informasi masalah sehingga pengembang dapat mereplikasi, menemukan sumber masalah, dan memperbaikinya dengan cepat. Laporan *bug* yang efektif minimal harus mencakup deskripsi masalah singkat, metode pasti untuk mereplikasi *bug*, hasil yang diharapkan, dan hasil yang sebenarnya. Dalam memberikan konteks yang lengkap, tangkapan layar atau rekaman video serta informasi lingkungan seperti jenis browser, versi sistem

operasi, dan payload API sangat penting. Seluruh 6 temuan *bug* yang telah dilaporkan kepada tim pengembang kini sudah berstatus ok karena seluruhnya telah berhasil diperbaiki dan diselesaikan secara tuntas.

f. Membuat dokumentasi

Gambar 3. 11 Membuat Dokumentasi

Gambar 3.11 merupakan tahap membuat dokumentasi RRAK untuk fase pengarsipan final yang mencakup semua aspek pengembangan dan pengujian Validasi RRAK. Dokumentasi ini berfungsi sebagai *knowledge base* yang komprehensif, meliputi ringkasan aturan bisnis validasi file, alur sistem, hasil akhir *pengujian* QA, dan *summary* dari *bug* yang telah diperbaiki. Dokumentasi ini juga memuat tampilan antarmuka pengguna seperti tampilan realisasi RRAK yang berfungsi sebagai panduan visual. Tujuan utama dari dokumentasi ini adalah memastikan tersedianya materi referensi yang akurat dan terstruktur bagi tim *developer,* tim operasional, dan admin yang bertanggung jawab atas proses validasi

RRAK sehingga pemeliharaan dan pelatihan pengguna dapat dilakukan secara efektif.

g. Sosialisasi dengan user mengenai *enhancement* website Rencana Realisasi Anggaran Kas (RRAK)

Sosialisasi dengan user mengenai RRAK ini bertujuan untuk mengajarkan kepada *end user* mengenai cara penggunaan aplikasi. Sosialisasi dilaksanakan melalui zoom dengan COSP. Fitur realisasi pada RRAK membantu COSP dalam mencatat anggaran dan meningkatkan akuntabilitas perusahaan untuk menghindari risiko pemalsuan data.

3.3.1.3 Permintaan Barang Manual

a. Analisa sistem desain dan memahami struktur data

Gambar 3. 12 Memahami Sistem Desain

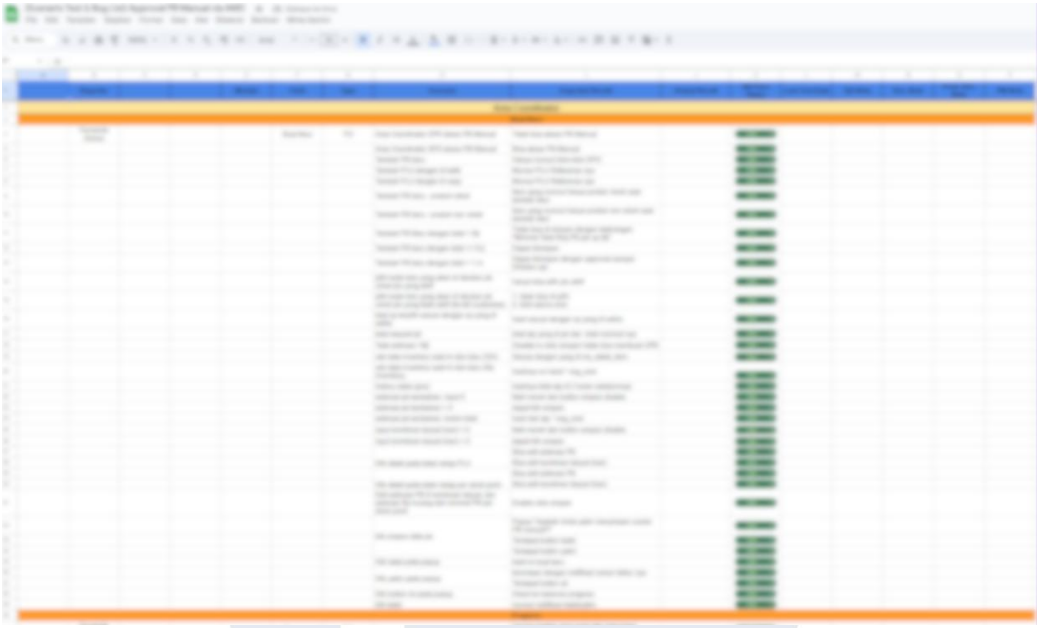
Pada gambar 3.12 merupakan tahap memahami desain sistem Permintaan Barang Manual. Tahap ini merupakan langkah awal penting dalam merancang solusi untuk mengatasi inefisiensi dan kesalahan manusia dalam proses PB yang sebelumnya berbasis Excel dan manual. Perancangan form digital PB Manual untuk alokasi item di SP, termasuk struktur database yang mendukung perhitungan estimasi yang akurat dan minim risiko. Selain itu, pada tahap perancangan alur kerja *Approval* PB Manual dilakukan melalui Aksesmu Management Dashboard (AMD)

yang akan menggantikan proses Berita Acara Pelaksanaan (BAP) dan e-mail yang tidak efisien. Ini mencakup pemetaan aliran sistem, tampilan antarmuka untuk Fitur Pengawasan Status Persetujuan, dan integrasi data otomatis dari PB yang disetujui langsung ke sistem *Distribution Center System* (DCS). Hal ini menghilangkan kemungkinan entri manual yang terlewatkan.

Masalah yang dihadapi Aksesmu adalah perhitungan estimasi Permintaan Barang (PB) manual SP masih menggunakan Excel dan perhitungan manual, hasil estimasi berpotensi mengalami *human-error* dan kurang efisien. Proses *approval* PB dilakukan via BAP dan e-mail untuk setiap faktur PB, sehingga memakan waktu tunggu *approval* yang cukup panjang dan kesulitan pelacakan status *approval* PB. Ketika proses *approval* PB manual telah selesai, proses agar data PB masuk ke dalam sistem DCS masih dilakukan secara manual, berpotensi terlewat.

Kemudahan dalam menganalisis dampak dari permintaan barang dan proses *approval* dan monitoring PB yang lebih efektif sangat dibutuhkan oleh perusahaan. Solusi dari masalah ini adalah form digital PB manual untuk alokasi item di SP, fitur untuk memantau status *approval* PB Manual, dan fitur *Approval* PB manual untuk memudahkan *approval* tanpa BAP.

b. Membuat skenario pengujian Permintaan Barang Manual

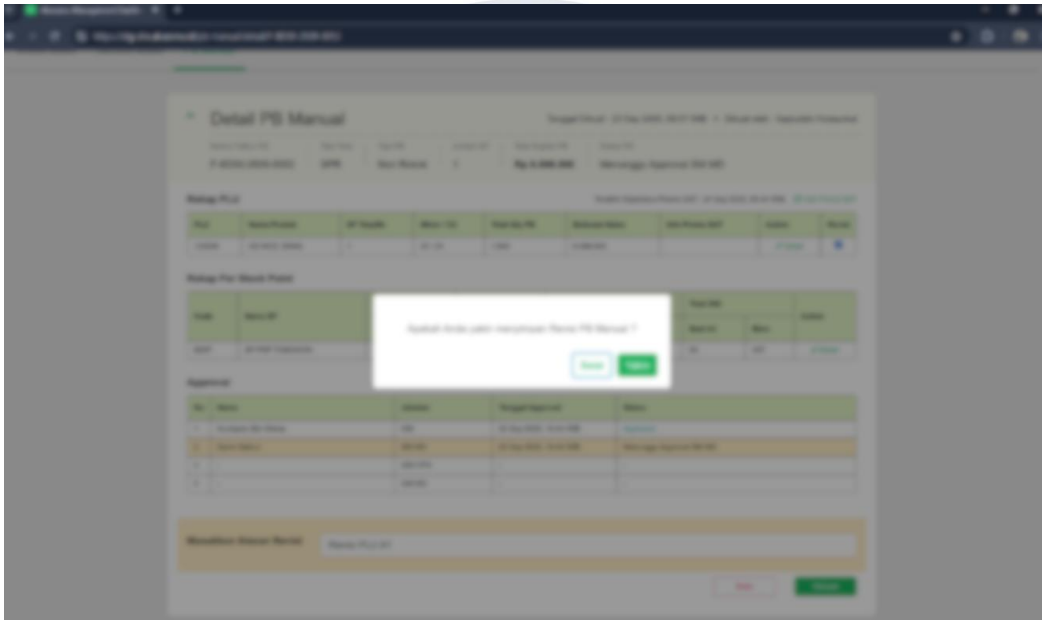


Gambar 3. 13 Membuat Skenario

Gambar 3.13 merupakan tahap membuat skenario yang berisi proses merumuskan serangkaian kasus uji komprehensif untuk memverifikasi fungsionalitas *Approval* PB *Manual* via Aksesmu Management Dashboard (AMD). Skenario uji ini dirancang untuk memastikan bahwa solusi yang dikembangkan efektif dalam mengatasi risiko *human error* dan alur *approval* yang lama. Proses ini mencakup perumusan skenario untuk validasi input pada Form Digital PB Manual, pengujian alur persetujuan yang disederhanakan via AMD, serta verifikasi fungsionalitas Fitur *monitoring* status PB yang baru. Secara spesifik, skenario ini merinci bahwa Stock Point Grosir (SPG) memerlukan *approval* dari *Operation Manager* (OM) dan OM SPG, sementara Stock Point Retail (SPR) hanya memerlukan *approval* dari OM. Jika nominal 5 juta maka akan *approval* akan di acc oleh OM, OM SPG, *Supplier & Product Specialist, Ssp Operation general Manager, dan Merchandising General Manager*. Jika nominal 1 milliar maka akan *approval* akan di acc oleh OM, OM SPG, *Supplier & Product Specialist, Ssp Operation general Manager, Merchandising General Manager, dan Operations director*. Setiap skenario mendefinisikan *test case* dan *expected result* untuk menguji logika bisnis hak akses pengguna dan yang paling penting verifikasi

integrasi data otomatis PB yang disetujui ke sistem DCS. QA telah menyelesaikan seluruh pembuatan dan validasi pada total 572 skenario pengujian dengan hasil akhir tingkat kelulusan 100 persen karena seluruh skenario sudah berstatus ok.

c. Pengujian fitur Permintaan Barang Manual pada aplikasi *Aksesmu Management Dashboard* (AMD)

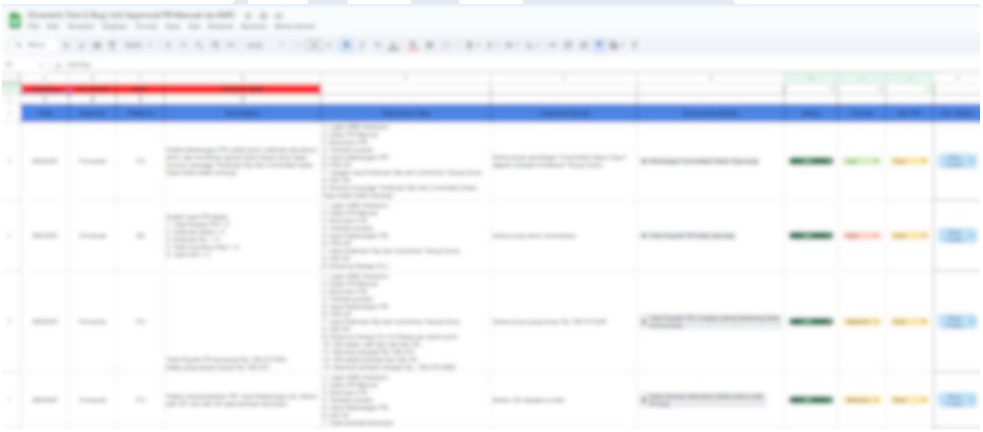


Gambar 3. 14 Pengujian

Gambar 3.14 merupakan tahap pengujian pelaksanaan skenario uji untuk memverifikasi fungsionalitas dan logika bisnis pada fitur *Approval* PB Manual via AMD. Pengujian ini difokuskan untuk memastikan akurasi perhitungan estimasi, efektivitas alur *approval* yang baru, serta kemudahan pelacakan status. Secara spesifik, pengujian ini merinci bahwa *Stock Point Grosir* (SPG) memerlukan *approval* dari *Operation Manager* (OM) dan OM SPG, sementara *Stock Point Retail* (SPR) hanya memerlukan *approval* dari OM. Jika nominal 5 juta maka akan *approval* akan di terima oleh OM, OM SPG, *Supplier & Product Specialist, Ssp Operation general Manager, dan Merchandising General Manager.* Jika nominal 1 milliar maka akan *approval* akan di acc oleh *OM, OM SPG, Supplier & Product Specialist, Ssp Operation general Manager, Merchandising General Manager, dan Operations director*. Proses pengujian mencakup validasi input pada form digital, pengujian hak akses *user* yang melakukan *approval*, dan verifikasi notifikasi yang

muncul, termasuk konfirmasi saat proses *approval* selesai seperti yang terlihat. Pengujian ini krusial untuk memastikan bahwa setelah disetujui, data PB secara otomatis terintegrasi ke sistem DCS tanpa memerlukan entri manual, sehingga mengurangi risiko *human error* dan mempersingkat waktu tunggu *approval*. Proses pelaksanaan pengujian di lingkungan *staging* memerlukan waktu eksekusi yang bervariasi antara 2 hingga 10 menit untuk setiap satu skenario pengujian.

d. Mencatat *bug* pada laporan pengujian



Gambar 3. 15 Mencatat *Bug*

Gambar 3.15 merupakan tahap mencatat *bug* dan proses dokumentasi wajib terhadap semua penyimpangan fungsional yang terdeteksi selama pengujian fitur *Approval* PB *Manual* via AMD. Hasil pengujian yang telah dilakukan pada proyek permintaan barang manual, QA mencatatkan total sebanyak 55 temuan *bug* yang teridentifikasi dalam daftar temuan *bug* proyek tersebut dengan pembagian prioritas yang terdiri dari 13 *high*, 33 *medium*, dan 9 *low*. Beberapa temuan yang dicatat meliputi kendala pada fungsionalitas pengisian jumlah barang, kesalahan dalam proses validasi dokumen, serta beberapa kendala tampilan pada antarmuka pengguna saat melakukan pemilihan produk secara manual.

e. Melaporkan *bug* kepada *developer*

Melaporkan *bug* adalah bagian penting dari siklus pengembangan fitur, itu memastikan kualitas produk dan stabilitasnya. Selain memberikan informasi masalah, proses ini harus dilakukan secara sistematis dan informatif. Ini dilakukan agar pengembang dapat mereplikasi, menemukan sumber masalah, dan memperbaikinya dengan cepat. *Bug* yang ditemukan berupa tampilan dan API, serta laporan *bug* yang efektif minimal harus mencakup deskripsi masalah singkat, teknik tepat untuk mereplikasi *bug*, hasil yang diharapkan, dan hasil yang sebenarnya. Tangkapan layar atau rekaman video serta informasi lingkungan seperti jenis browser dan versi sistem operasi sangat penting untuk memberikan konteks yang lengkap. Seluruh 55 temuan *bug* yang telah dilaporkan kepada tim pengembang kini sudah berstatus OK karena seluruhnya telah berhasil diperbaiki dan diselesaikan secara tuntas.

f. Membuat dokumentasi

Gambar 3. 16 Membuat Dokumentasi

Gambar 3.16 merupakan tahap membuat dokumentasi pengarsipan final yang mencakup semua aspek pengembangan fitur *Approval* PB Manual via AMD. Dokumentasi ini sangat penting karena memuat spesifikasi perbedaan alur *approval* berdasarkan jenis toko. Secara spesifik, dokumentasi ini merinci bahwa Stock Point Grosir (SPG) memerlukan *approval* dari *Operation Manager* (OM) dan OM SPG, sementara Stock Point Retail (SPR) hanya memerlukan *approval* dari OM. Jika nominal 5 juta maka akan *approval* akan di acc oleh OM, OM SPG, *Supplier & Product Specialist, Ssp Operation general Manager, dan Merchandising General Manager*. Jika nominal 1 milliar maka akan *approval* akan di acc oleh OM, OM SPG, *Supplier & Product Specialist, Ssp Operation general Manager, Merchandising General Manager, dan Operations director*. Selain itu, dokumentasi mencakup detail fungsional Form Digital PB Manual, alur integrasi data ke sistem DCS, dan summary hasil Pengujian QA untuk memastikan semua

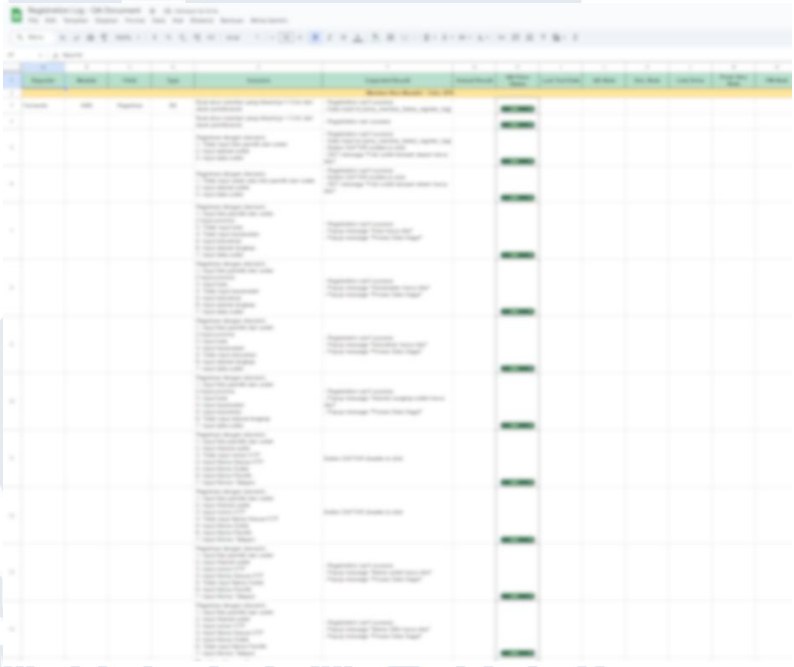
stakeholder memahami prosedur yang benar dan alur persetujuan yang disesuaikan dengan kebutuhan setiap area.

g. Presentasi *User Acceptance Testing* (UAT) dengan *user*

Presentasi dengan *department merchandising* (MD), *operation, dan logistik* mengenai cara pemakaian website permintaan barang manual. Terdapat beberapa fitur yang dihilangkan seperti validasi nominal, tidak menampilkan promo dari perusahaan, menambahkan email group MD pada tahap auto email ke senior manajer MD, Input jumlah tidak bisa *copy paste*, dan menambahkan kolom pada template logistik.

3.3.1.4 *Failed Registration Log*

a. Membuat skenario pengujian *Failed Registration Log*

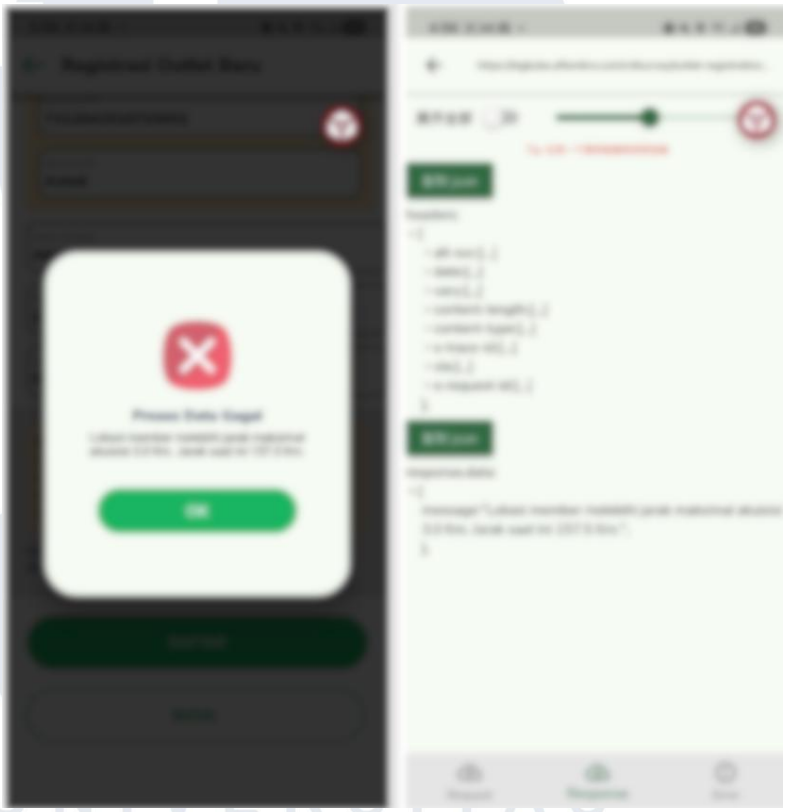


Gambar 3. 17 Membuat Skenario

Gambar 3.17 merupakan tahap membuat skenario proses serangkaian kasus uji yang sangat spesifik untuk memverifikasi aturan bisnis batasan jarak maksimal. Skenario uji ini dirancang untuk memastikan bahwa validasi jarak dan pencatatan log kegagalan hanya diimplementasikan dengan benar pada jenis toko yang memiliki batasan jarak. Kasus uji yang dirumuskan meliputi skenario positif seperti

registrasi di bawah batas jarak yang harus diterima, dan skenario negatif seperti registrasi melebihi batas jarak yang harus ditolak dan data kegagalannya dicatat ke dalam tabel *log*. Selain itu, skenario wajib memverifikasi bahwa jenis toko lain selalu diproses tanpa memicu validasi jarak atau pencatatan *log* kegagalan, sehingga memastikan efisiensi layanan sesuai dengan model bisnis yang berlaku.

b. Pengujian fitur *Failed Registration Log* pada aplikasi *Aksesmu Management System* (AMS)



Gambar 3. 18 Pengujian

Gambar 3.18 merupakan tahap pengujian pelaksanaan skenario uji untuk memverifikasi fungsionalitas validasi jarak maksimal *outlet*. Pengujian ini difokuskan untuk memastikan bahwa aturan bisnis batasan jarak diterapkan secara selektif. Proses *pengujian* melibatkan input data registrasi dengan kondisi jarak yang berbeda-beda. Hal ini bertujuan untuk memverifikasi bahwa registrasi yang melebihi batas jarak memicu penolakan sistem dan menampilkan pesan *error* yang

tepat kepada pengguna seperti yang diilustrasikan. Selain itu, pengujian juga memverifikasi bahwa sistem berhasil mencatat kegagalan tersebut ke dalam *log* sistem dan memastikan bahwa proses registrasi untuk jenis toko yang dikecualikan dari batasan jarak dapat diproses tanpa gangguan. QA telah menyelesaikan seluruh pembuatan dan validasi pada total 34 skenario pengujian dengan hasil akhir tingkat kelulusan 100 persen karena seluruh skenario sudah berstatus OK. Skenario yang diuji meliputi alur registrasi member non-mandiri untuk toko SPR dan SPG, yang mencakup validasi foto pemilik, pengisian alamat lengkap, hingga pengujian pesan peringatan jika nomor telepon tidak sesuai format atau data KTP tidak lengkap. Selain itu, dilakukan pula pengujian pada registrasi member mandiri untuk memastikan sistem memberikan notifikasi yang tepat saat terjadi kegagalan registrasi.

c. Mencatat *bug* pada laporan pengujian

Gambar 3. 19 Mencatat *Bug*

Gambar 3.19 merupakan tahap mencatat *bug* proses pendokumentasian semua ketidaksempurnaan atau penyimpangan yang ditemukan selama pengujian fitur Log Registrasi Gagal. Berdasarkan hasil pengujian yang telah dilakukan pada proyek *failed registration log* di sistem AMS, tim penguji mencatatkan total sebanyak 6 temuan *bug* yang teridentifikasi dalam daftar temuan *bug* proyek tersebut dengan kategori prioritas 6 *medium*. Beberapa temuan tersebut mencakup kendala pada *backend* di mana registrasi member non-mandiri dengan jarak lebih dari 3 km atau menggunakan koordinat GPS palsu masih dapat dilakukan, serta data yang tidak masuk ke dalam log kesalahan registrasi pada kondisi tertentu.

d. Melaporkan *bug* kepada *developer*

Siklus pengembangan fitur terdiri dari laporan *bug* yang memastikan kualitas produk dan stabilitasnya. Proses ini harus dilakukan secara sistematis dan informatif selain memberikan informasi masalah. Dengan cara ini, pengembang dapat mereplikasi, menemukan masalah, dan memperbaikinya secara instan. Laporan *bug* yang efektif untuk tampilan dan API juga harus mencakup deskripsi masalah singkat, metode yang tepat untuk mereplikasi *bug*, hasil yang diharapkan, dan hasil yang sebenarnya. Dalam memberikan konteks yang lengkap, tangkapan layar atau rekaman video serta informasi tentang lingkungan Anda, seperti jenis browser dan versi sistem operasi, sangat penting. Seluruh 6 temuan *bug* yang telah dilaporkan kepada tim pengembang kini sudah berstatus OK karena seluruhnya telah berhasil diperbaiki, termasuk perbaikan pada validasi data alamat *outlet* dan sinkronisasi data ke dalam log sistem.

e. Membuat dokumentasi

Gambar 3. 20 Membuat Dokumentasi

Gambar 3.20 merupakan tahap membuat dokumentasi tahap pengarsipan final yang mencakup semua aspek pengembangan dan pengujian fitur Log Registrasi Gagal. Dokumentasi ini berfungsi sebagai *knowledge base* yang krusial, berfokus pada detail aturan bisnis batasan jarak maksimal. Dokumentasi secara spesifik memuat perbedaan perlakuan antara jenis toko di mana satu jenis toko memiliki validasi jarak maksimal dengan aturan batas 3 kilometer yang memicu

penolakan dan pencatatan log kegagalan, sementara jenis toko lainnya memiliki validasi jarak yang ditiadakan dan registrasi selalu dilanjutkan. Dokumentasi ini juga merangkum alur sistem, hasil pengujian, dan ringkasan *bug* yang telah diperbaiki, memastikan semua *stakeholder* memahami implementasi logika validasi registrasi yang berlaku.

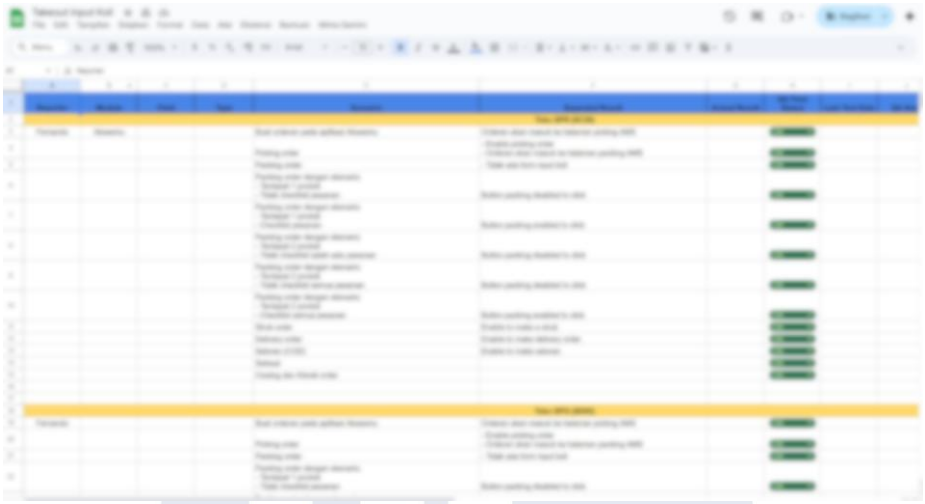
3.3.1.5 Pengkinian Data Bayar Tunda

a. Analisa sistem desain dan memahami struktur data

Gambar 3. 21 Memahami sistem desain

Gambar 3.21 merupakan tahap memahami desain sistem Pengkinian Data Bayar Tunda yang berfokus pada perancangan logika tampilan dan prioritas notifikasi di halaman *home* untuk mendukung kepatuhan data dan *loan collection*. Pemahaman sistem mendalami mekanisme pengecekan status pengguna OBA dari sumber data pinjaman untuk menentukan prioritas notifikasi. Fokus utama desain adalah membuat alur keputusan yang memastikan notifikasi tunggakan selalu diprioritaskan jika ada pinjaman jatuh tempo. Namun, jika pengguna juga wajib melakukan pengkinian data, sistem harus memaksa semua klik notifikasi untuk mengarahkan pengguna ke tampilan Pengkinian Data. Selain itu, sistem harus memastikan bahwa notifikasi ini hanya tampil jika perangkat yang digunakan terbaca sebagai *primary device*, dan tidak tampil jika perangkat tersebut terbaca sebagai *secondary device*.

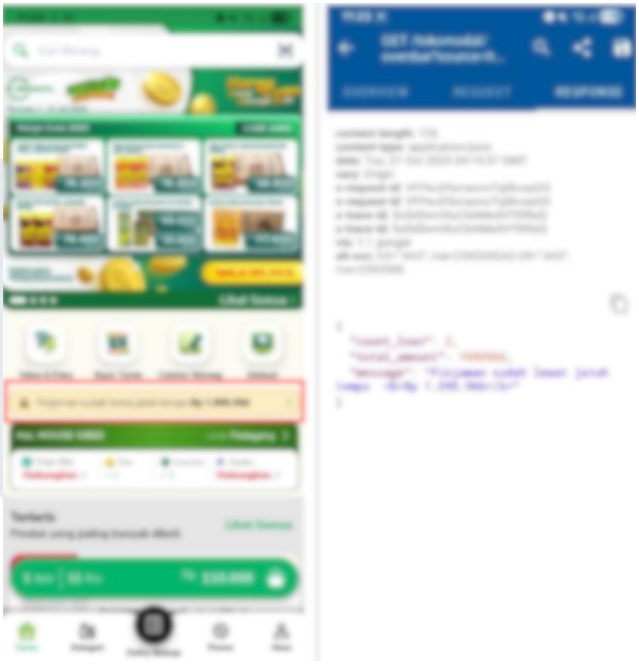
b. Membuat skenario pengujian Data Pengkinian Bayar Tunda



Gambar 3. 22 Membuat Skenario

Gambar 3.22 merupakan tahap membuat skenario yang berisi proses merumuskan serangkaian skenario uji komprehensif untuk memverifikasi logika prioritas notifikasi pada fitur Pengkinian Data Bayar Tunda. Skenario uji ini dirancang untuk memastikan kepatuhan data debitur dan mendukung upaya penagihan pinjaman. Perumusan skenario berfokus pada pengujian berbagai kombinasi status pinjaman dan kewajiban pengkinian data. Hal ini mencakup pengujian prioritas notifikasi tunggakan terhadap notifikasi pengkinian data dan skenario yang memverifikasi pengalihan paksa pengguna OBA ke tampilan pengkinian data saat ia wajib melakukan pengkinian data meskipun notifikasi tunggakan yang tampil. Selain itu skenario wajib memverifikasi bahwa notifikasi hanya muncul pada *primary device* dan tidak muncul pada *secondary device* untuk menjamin akurasi dan efisiensi notifikasi. QA telah menyelesaikan seluruh pembuatan dan validasi pada total 16 skenario pengujian dengan hasil akhir tingkat kelulusan 100 persen karena seluruhnya berstatus ok. Skenario yang diuji meliputi validasi munculnya notifikasi pinjaman jatuh tempo, logika klik pada notifikasi untuk mengarahkan pengguna ke halaman pengkinian data, verifikasi masuk menggunakan *primary device*, serta fungsionalitas tombol bayar tunda pada berbagai kondisi akun pengguna.

c. Pengujian fitur Pengkinian Data Bayar Tunda pada aplikasi Aksesmu



Gambar 3. 23 Pengujian

Gambar 3.23 merupakan pengujian yang difokuskan pada validasi logika prioritas tampilan notifikasi di halaman *home* serta pengalihan paksa pengguna. Proses pengujian melibatkan pengecekan berbagai kondisi status pinjaman pengguna OBA yang dikombinasikan dengan status kewajiban pengkinian data. Selain itu, pengujian juga memverifikasi bahwa notifikasi terkait hanya muncul pada *primary device* dan memastikan sistem menampilkan data pesanan dengan benar, untuk menjamin bahwa alur notifikasi berfungsi efektif dalam mendukung upaya *loan collection* dan pembaruan data pengguna. Proses pelaksanaan pengujian di lingkungan *staging* memerlukan waktu eksekusi yang bervariasi antara 2 hingga 5 menit untuk setiap satu skenario pengujian.

d. Melaporkan *bug* kepada *developer*

Setelah proses pengujian fitur Pengkinian Data selesai, sangat penting untuk melaporkan *bug* kepada *developer*. Jika ada kegagalan yang ditemukan, seperti ketika notifikasi tentang pengkinian data tidak muncul di halaman home untuk OBA terdaftar, meskipun respon server telah menunjukkan bahwa pembaruan data harus dilakukan. Berdasarkan hasil pengujian yang telah dilakukan pada proyek pengkinian bayar tunda, tim penguji mencatatkan total sebanyak 1 temuan *bug* yang teridentifikasi dalam daftar temuan *bug* proyek tersebut dengan kategori prioritas 1 *medium*. Temuan ini mencakup kendala pada tampilan notifikasi pengkinian data yang tidak muncul pada halaman beranda meskipun respon sistem sudah menunjukkan status aktif.

e. Membuat dokumentasi

Gambar 3. 24 Membuat Dokumentasi

Gambar 3.24 merupakan tahap membuat dokumentasi pengarsipan final yang mencakup semua aspek fungsional dan teknis dari fitur Pengkinian Data Bayar Tunda. Dokumentasi ini merinci logika prioritas notifikasi pada halaman *home*, yaitu memastikan notifikasi tunggakan selalu diprioritaskan, namun mengarahkan pengguna OBA ke halaman pengkinian data jika data wajib diperbarui. Dokumentasi ini juga memuat spesifikasi tampilan antarmuka dan respon sistem, seperti detail pesanan dan respon data yang dilihat selama pengujian. Tujuan dokumentasi ini adalah untuk menyediakan *knowledge base* yang akurat mengenai mekanisme validasi *primary device* dan alur pengalihan paksa, yang penting untuk pemeliharaan sistem dan pelatihan *support* terkait *loan collection* dan kepatuhan data debitur.

3.3.1.6 *Bulk Data Order*

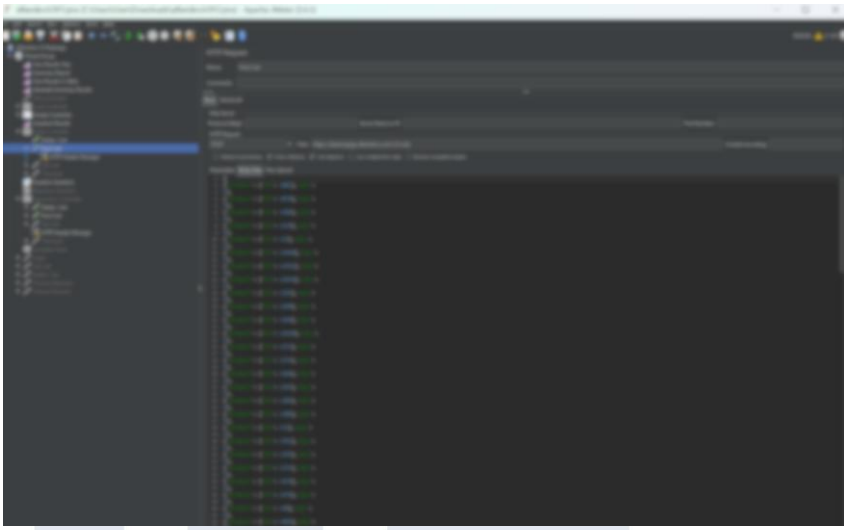
Penggunaan *Apache JMeter* dalam skenario *bulk data order* bertujuan untuk memvalidasi stabilitas aplikasi Aksesmu saat menangani pembuatan pesanan dalam volume besar secara simultan melalui antarmuka pemrograman aplikasi atau *API*. Berdasarkan hasil eksekusi skenario pengujian performa, sistem menunjukkan kemampuan pemrosesan data yang stabil dengan rata-rata waktu respons sebesar 450 milidetik per transaksi. Interpretasi dampaknya adalah peningkatan efisiensi operasional melalui penghematan waktu tim serta terjaminnya keandalan sistem dalam memproses transaksi massal secara cepat dan stabil dengan tingkat kesalahan minimal.

Tabel 3. 2 Perbandingan Efisiensi Waktu Eksekusi Pesanan

Jumlah Pesanan	Metode Manual	Metode Otomatis (Jmeter)
1 Order	40 detik	0.45 detik
10 Order	400 detik (6.6 menit)	1 detik
50 Order	2.000 detik (33.3 menit)	5 detik

Tabel 3.2 menunjukkan perbandingan efisiensi waktu eksekusi pesanan yang sangat signifikan antara metode manual dan penggunaan otomatis jmeter melalui lapisan *API*. Data tersebut memaparkan bahwa untuk memproses 50 pesanan, metode manual memerlukan waktu hingga 2.000 detik atau setara dengan 33,3 menit, sedangkan melalui otomatisasi proses yang sama dapat diselesaikan hanya dalam waktu 5 detik saja. Hal ini membuktikan bahwa penggunaan otomatisasi mampu meningkatkan efisiensi waktu eksekusi transaksi secara *end-to-end* secara drastis dibandingkan entri data melalui antarmuka pengguna.

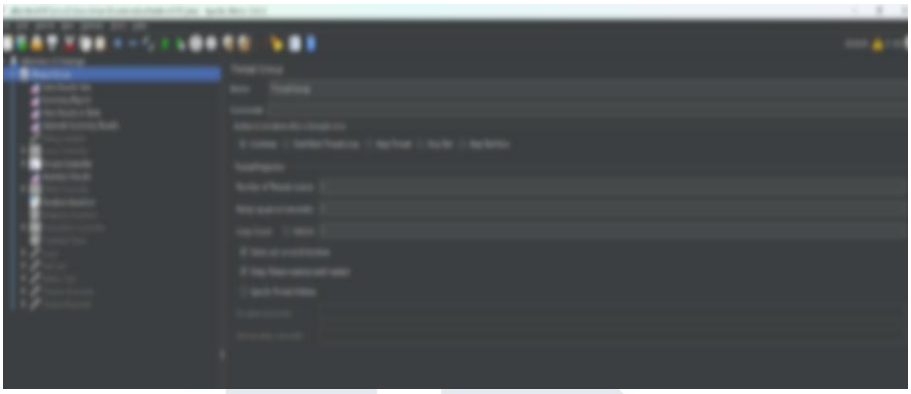
a. *Bulk Data Order*



Gambar 3. 25 *Bulking Data Order*

Gambar 3.25 merupakan tahap pengujian otomatis yang bertujuan untuk mempercepat siklus pengujian fitur pemesanan aplikasi Aksesmu. Apache JMeter adalah aplikasi untuk skenario pengujian berbasis *Application Programming Interface* (API) yang memungkinkan simulasi aktivitas pengguna tanpa menggunakan *User Interface* (UI). Tujuan utama *bulk data* adalah untuk membuat orderan dengan jumlah banyak [15]. Hal ini memberikan efisiensi waktu dalam melakukan pengujian bagi QA. engalihkan fokus pengujian ke lapisan API dapat meningkatkan frekuensi dan efisiensi pengujian regresi dan mengurangi waktu eksekusi transaksi *end-to-end.*

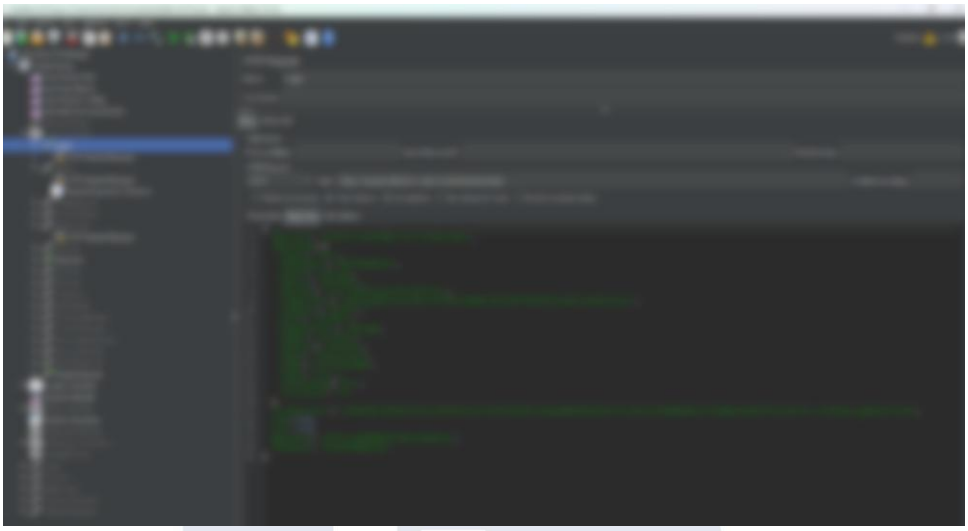
Pengujian Otomatis seluruh proses transaksi pesanan, mulai dari *login* pengguna hingga pembayaran adalah bagian dari proyek ini. Implementasi teknisnya menggunakan serangkaian permintaan HTTP yang mereplikasi setiap langkah. Ini termasuk memilih barang, menambahkan barang ke keranjang, dan melakukan pembayaran. Proses yang sangat penting adalah permintaan *Post Cart*, yang dimaksudkan untuk mengirimkan data *payload* produk dalam bentuk array JSON sekaligus. Dengan desain ini, banyak *item* dapat dimasukkan ke dalam keranjang dalam satu panggilan API. Hal ini adalah bagian penting dari fungsionalitas pembelian massal yang dimaksudkan untuk mempercepat dan mempermudah proses pengujian massal.



Gambar 3. 26 *Threads Group*

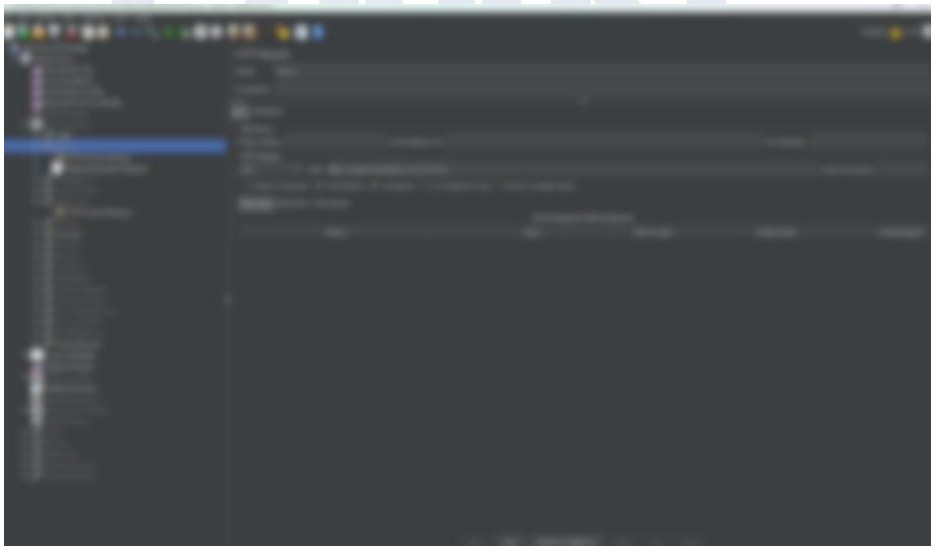
Gambar 3.26 mengilustrasikan tahap awal dalam penyusunan rencana pengujian (*test plan*) yang mencakup urutan komponen pengujian untuk mensimulasikan beban kerja pada server. Di dalam *thread group*, terdapat parameter teknis utama yang mengatur skenario pengujian, yaitu *Number of Threads* dan *Ramp-up Period*. *Number of Threads* berfungsi untuk menentukan jumlah pengguna virtual (*virtual users*) yang akan disimulasikan, sedangkan *Ramp-up Period* merepresentasikan durasi waktu yang diperlukan sistem untuk mengaktifkan seluruh *thread* tersebut secara bertahap. Sebagai contoh, jika konfigurasi menetapkan 10 *threads* dengan *Ramp-up Period* selama 10 detik dan *loop count* bernilai 1, maka sistem akan menjalankan satu *thread* setiap interval 1 detik. Penyesuaian pada nilai durasi dan jumlah *threads* ini sangat krusial untuk menciptakan simulasi beban yang akurat sesuai dengan kebutuhan pengujian.

Langkah berikutnya dalam konfigurasi ini adalah pengintegrasian *HTTP Header Manager* untuk mendefinisikan parameter autentikasi dan metadata aplikasi. Hal ini mencakup pengisian nilai *Authorization* dari API, tipe konten (*content type*), versi aplikasi, sistem operasi (*OS version*), jenis perangkat, hingga *firebase token*. Selain itu, penggunaan *HTTP request sampler* diperlukan untuk menentukan detail teknis seperti protokol, alamat IP (*IP address*), nomor *port*, serta metode dan jalur (*path*) yang akan diuji. Berbagai metode permintaan HTTP dapat diimplementasikan sesuai kebutuhan, mulai dari operasi standar seperti *GET*, *POST, DELETE, PUT*, dan *PATCH*, hingga metode yang lebih spesifik seperti *PROPFIND, MKCOL, MOVE, COPY, LOCK*, dan *SEARCH*.



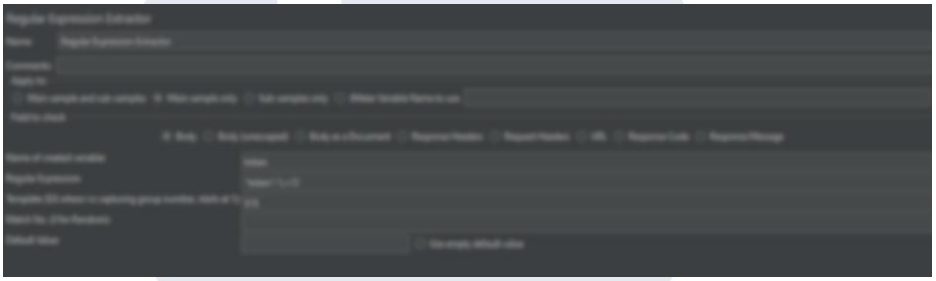
Gambar 3. 27 *HTTP Request Login*

Gambar 3.27 merupakan permintaan HTTP dikirim ke *endpoint* yang memiliki otorisasi untuk memulai proses login dan memulai autentikasi sesi. Permintaan tersebut membawa *payload* JSON yang mengandung detail perangkat seperti device id, os, brand, dan firebase_token. Detail ini diperlukan oleh server untuk memvalidasi perangkat klien dan memastikan keamanannya. Langkah pertama penting dalam skenario pengujian kinerja adalah permintaan login *POST* ini untuk mendapatkan token sesi atau kunci otorisasi, yang akan digunakan oleh semua permintaan transaksi berikutnya dalam alur pengujian.



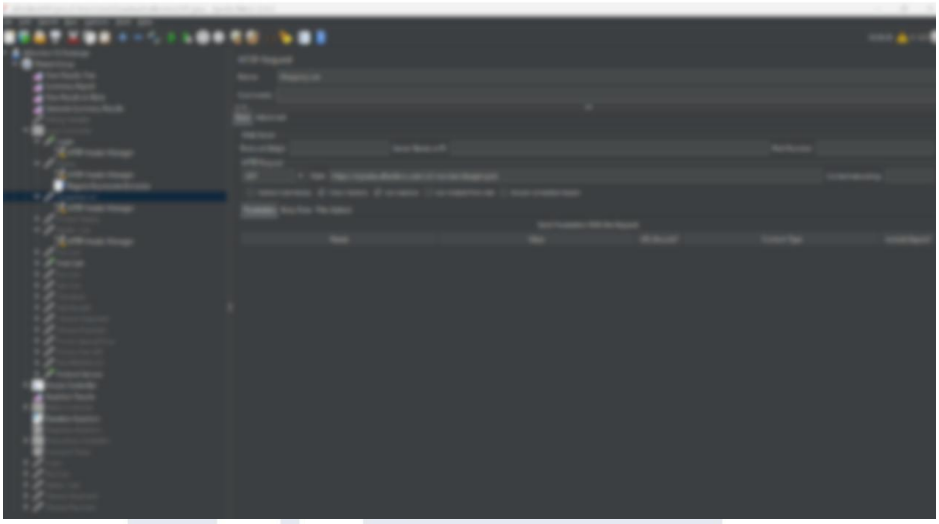
Gambar 3. 28 *HTTP Request Home*

Gambar 3.28 adalah permintaan *home* ini dikonfigurasi sebagai *HTTP Request* dengan metode *GET* dan menunjukkan tujuannya adalah untuk mengambil atau memuat konten dari server. Permintaan ini ditujukan ke *endpoint* yang bertanggung jawab untuk menampilkan konten halaman utama. Permintaan ini tidak membawa *body data* atau parameter lainnya karena menggunakan metode *GET*. Permintaan ini biasanya dilakukan setelah proses login berhasil dalam alur pengujian untuk menghitung kecepatan pemuatan semua elemen dinamis yang membutuhkan sesi pengguna yang sudah tervalidasi.



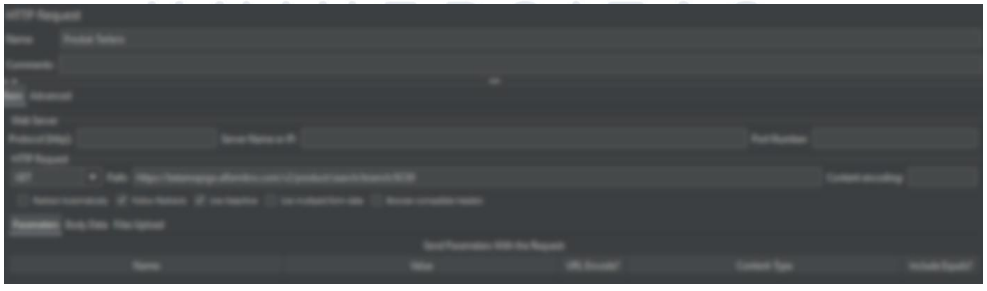
Gambar 3. 29 *Regular Expression Extractor Home*

Gambar 3.29 adalah *Post-Processor* elemen *Regular Expression Extractor* ini ditempatkan setelah permintaan login dan diatur untuk mengekstrak nilai token otorisasi dari *body* respons JSON server. Dengan menggunakan ekspresi reguler "token":"(.+?)", ekstraktor ini secara khusus menangkap nilai token dan menyimpannya dalam variabel JMeter yang disebut token. Tujuan utama korelasi data ini adalah untuk mendapatkan token sesi yang valid. Setelah itu, permintaan seperti *Home* dan *Put Cart* dapat digunakan dengan menggunakan *HTTP Header Manager* untuk memastikan bahwa pengguna telah terautentikasi dan dapat mengakses fungsi aplikasi yang membutuhkan sesi.



Gambar 3. 30 *HTTP Request Shopping List*

Gambar 3.30 merupakan bagian *HTTP shopping list* yang dikonfigurasi sebagai permintaan *GET* dan dikirim ke endpoint layanan daftar belanjaan pengguna. Mengambil atau menampilkan daftar barang yang telah ditambahkan atau disimpan oleh pengguna yang sedang login adalah fungsinya. Permintaan ini, seperti permintaan *Home*, tidak mengirimkan *body data* sama sekali, sebaliknya, bergantung pada token otorisasi yang dikumpulkan dari langkah login sebelumnya. Token ini harus disematkan ke *HTTP Header Manager* yang terkait untuk memastikan server mengembalikan daftar belanja pengguna yang benar. Permintaan ini memeriksa kemampuan server untuk menangani kueri database khusus pengguna yang berkaitan dengan daftar belanja atau daftar keinginan.



Gambar 3. 31 *HTTP Request* Produk Terlaris

Gambar 3.31 merupakan *HTTP request* yang disebut "Produk Terlaris" ini dikonfigurasi sebagai permintaan *GET* dan berfungsi untuk mengambil data dari

endpoint pencarian produk. Tujuan permintaan ini adalah untuk mencari produk terlaris atau produk katalog tertentu, dengan fokus pada filter lokasi/cabang khusus. Ini tidak membawa *body data*, sebaliknya dalam memverifikasi akses data, perlu menyertakan token otorisasi yang diperoleh dari proses login *HTTP Header Manager* terkait (*Authorization*: Bearer ${token}). Permintaan ini menguji kemampuan server untuk melakukan kueri database produk yang kompleks, seringkali dengan filter yang ditempatkan di tempat tertentu.

Gambar 3. 32 *HTTP Request Delete Cart*

Gambar 3.32 adalah permintaan HTTP yang disebut *Delete Cart* ini menggunakan metode *DELETE*, menunjukkan bahwa tujuan utamanya adalah menghapus isi keranjang belanja OBA. Permintaan ini dikirim ke *endpoint* yang bertanggung jawab untuk menghapus keranjang belanja. Ini tidak membawa *body data* karena ini adalah operasi *DELETE* yang diidentifikasi menggunakan sesi pengguna yang terautentikasi. Kesuksesan permintaan ini sangat bergantung pada pengiriman token otorisasi yang sah—yaitu header *Authorization*: Bearer ${token}—ke *HTTP Header Manager* yang relevan. Ini digunakan untuk menentukan pengguna mana yang harus dihapus dari keranjangnya. Dalam memastikan bahwa alur transaksi berikutnya dimulai dengan kondisi keranjang yang kosong, sehingga hasilnya konsisten dan dapat diulang, langkah ini sangat penting dalam skenario pengujian kinerja.

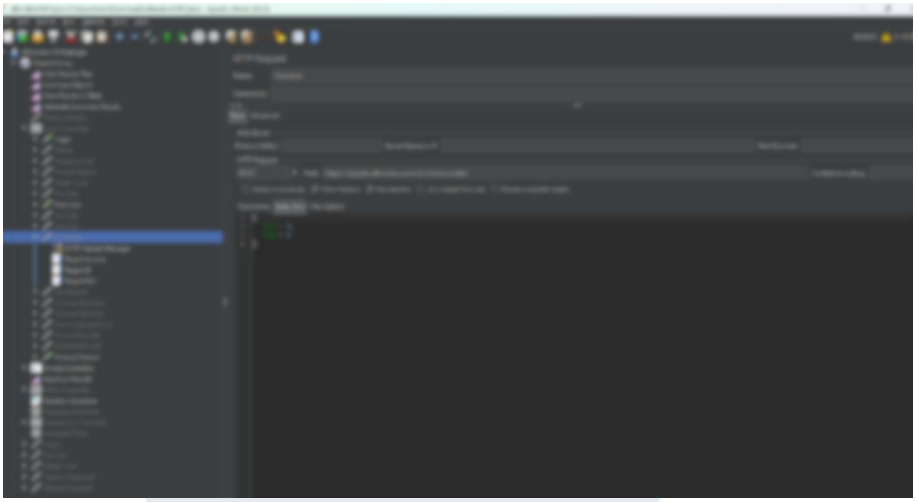
Gambar 3. 33 *HTTP Request Put Cart*

Gambar 3.33 merupakan *HTTP Request Put Cart* yang menggunakan metode *PUT* dan tujuan utamanya adalah untuk menambahkan atau memperbarui barang ke keranjang belanja pengguna. Permintaan ini dikirimkan ke *endpoint* keranjang belanja dan mengirimkan *body data* dalam format JSON yang berisi jumlah dan ID produk. Dalam memastikan konsistensi tes, permintaan ini harus didahului oleh langkah menghapus keranjang. Kesuksesan tes sangat bergantung pada pengiriman *header* yang valid dan menentukan sesi pengguna mana yang mengisi keranjang.

Gambar 3. 34 *HTTP Request Post Cart*

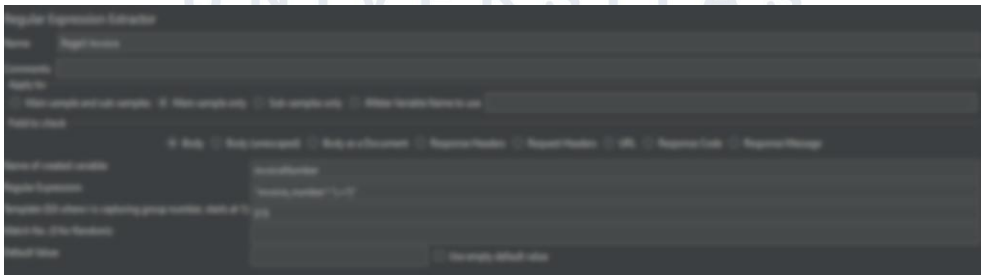
Gambar 3.34 merupakan *HTTP Request Post Cart* dengan metode yang menggunakan *POST*. Tujuan utamanya adalah untuk mengirimkan atau menambahkan berbagai item ke keranjang belanja pengguna. Permintaan ini ditujukan ke *endpoint* keranjang belanja dan mengirimkan *body data* dalam format array JSON. Dalam array ini, detail dua entri produk yang berbeda bersama dengan

jumlah masing-masing. Tahap *POST* ini digunakan untuk memasukkan beberapa item sekaligus, berbeda dengan metode *PUT* yang biasanya memperbarui satu item. *Header* yang valid akan mengidentifikasi pengguna mana yang menambah item ke keranjang.



Gambar 3. 35 *HTTP Request Checkout*

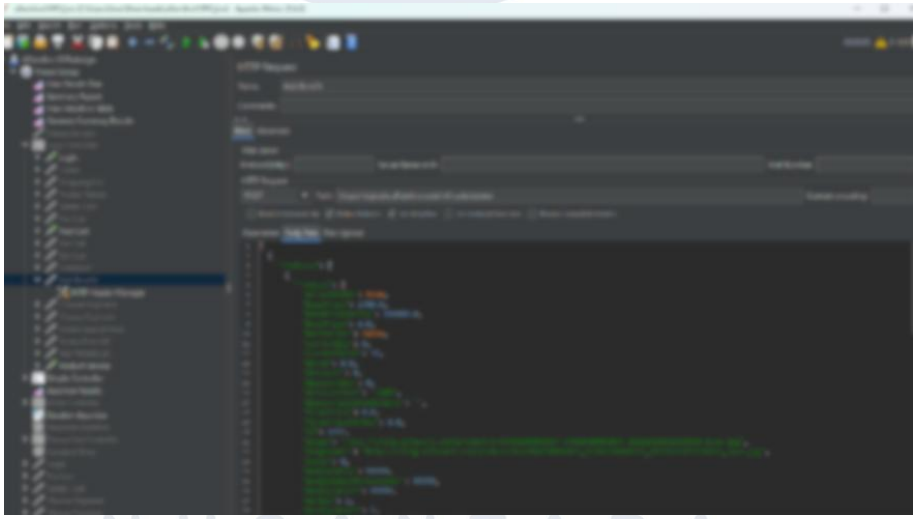
Gambar 3.35 merupakan *HTTP Request Checkout* yang menggunakan metode *POST*, dan tujuan utamanya adalah untuk memulai atau memulai proses *checkout* atau membuat pesanan baru. Permintaan ini dikirim ke *endpoint* yang memulai *checkout* dan membawa *body data* dalam format JSON yang mencakup data konfigurasi dasar seperti *flag* atau pilihan tertentu yang dibutuhkan server untuk memproses transaksi. *HTTP Request Checkout* ini adalah langkah terakhir dalam alur simulasi belanja, dan keberhasilannya bergantung pada keranjang belanja yang sudah terisi dan otorisasi pengguna yang sudah valid melalui header.



Gambar 3. 36 *Regex Checkout*

Gambar 3.36 merupakan tahap setelah login dilakukan, data perangkat diambil oleh *Regular Expression Extractor*, yang kemudian menyimpan token sesi sebagai tiket masuk. Dengan menggunakan *header authorization*, *HTTP Header Manager* memasukkan token ini ke semua permintaan berikutnya. Setelah otorisasi berhasil, dimulai menjelajah konten seperti *Home*, *Shopping List*, dan Produk Terlaris untuk menampilkan halaman utama dan detail katalog. Dalam memastikan hasil pengujian yang konsisten, *Delete Cart* dilakukan sebelum memulai transaksi untuk memastikan keranjang kosong.

Dalam menambahkan barang dan jumlah yang diinginkan, lalu dilanjutkan dengan *Put Cart* atau *Post Cart* selama proses pengisian keranjang belanja. Ada sejumlah *Regular Expression Extractor* yang digunakan dalam langkah terakhir. *Regex Invoice* mengambil nomor faktur yang dibuat server, *Regex ID* mengambil ID internal transaksi atau keuntungan, dan *Regex PLU* mengambil kode produk tertentu. Data yang diekstrak ini sangat penting untuk proses akhir, seperti mengkonfirmasi pembayaran atau mengirimkan barang.



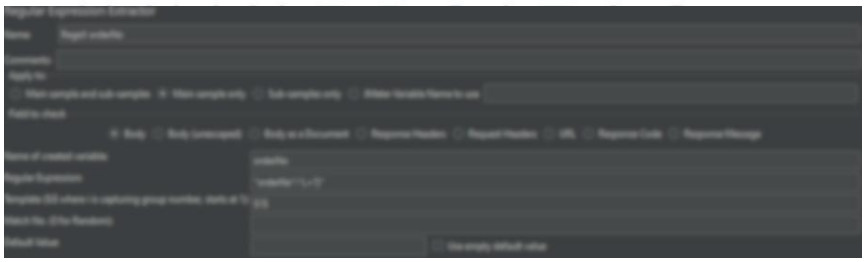
Gambar 3. 37 *HTTP Request Add Benefit*

Dalam memasukkan keuntungan yang dipilih ke dalam keranjang belanja yang sudah terisi, *HTTP Request Add Benefit* ini menggunakan metode *POST*. Permintaan ini dikirim ke *endpoint* promo khusus dan membawa *payload* JSON yang sangat rinci yang mencakup detail lengkap produk yang akan diberikan

keuntungan atau hadiah, termasuk ID produk, kode produk, harga, dan nilai promo. Alur *checkout* yang bergantung pada korelasi data membutuhkan permintaan ini setelah pembayaran. Itu harus dilakukan dengan menggunakan nilai ID produk dan kode yang diekstrak dari respons server sebelumnya untuk mengidentifikasi dan memvalidasi keuntungan mana yang diterapkan.

Gambar 3. 38 *HTTP Request Choose Shipment*

Dengan menggunakan metode *POST*, *HTTP Request* yang disebut *Choose Shipment* ini bertujuan untuk mengirimkan pilihan metode pengiriman yang telah dipilih pengguna ke server. Permintaan ini ditujukan ke *endpoint* penetapan pengiriman dan membawa *body data* dalam format JSON yang mencakup koordinat lokasi seperti *Latitude* dan *Longitude*, serta nilai numerik yang menunjukkan metode pengiriman yang dipilih. Langkah penting dalam alur *checkout* ini adalah membutuhkan token otorisasi *HTTP Header Manager* yang valid untuk membedakan sesi pengguna. Permintaan ini harus dilakukan setelah proses *Checkout* berhasil dimulai.

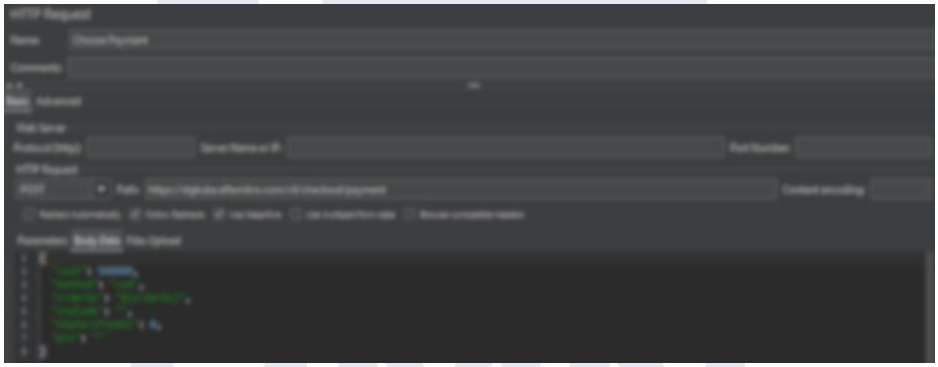


Gambar 3. 39 *Regex Choose Shipment*

Pertama, *Regex orderNo* melakukan korelasi data dengan mengambil nilai Nomor Pesanan yang dibuat atau divalidasi oleh server setelah metode pengiriman

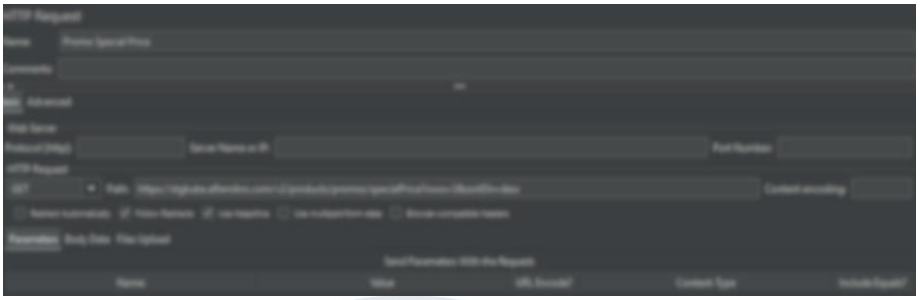
diatur. Elemen ini mencari *key* "orderNo" di bagian tanggapan permintaan dan menyimpan nilai yang ditemukan ke dalam variabel ${orderNo}. Nomor pesanan berbeda dari nomor faktur dan dibutuhkan untuk melacak atau memverifikasi pesanan pada tahap pembayaran berikutnya.

Kedua, *Regex cashes* dibuat untuk mendapatkan informasi penting tentang metode pembayaran tunai yang mungkin tersedia bagi pengguna. *Extractor* ini mencari tombol "uang tunai" dan mengumpulkan daftar semua nilai atau angka di dalam array yang dikembalikan oleh server. Dalam permintaan *Choose Payment* berikutnya, data yang dikumpulkan ini digunakan dan disimpan dalam variabel ${cashes}. Ini menjamin bahwa simulasi alur checkout dapat secara dinamis memilih dan menggunakan metode pembayaran tertentu, yang telah dihitung dan divalidasi oleh server setelah harga pengiriman ditetapkan.



Gambar 3. 40 *HTTP Request Choose Payment*

Permintaan *Choose Payment* ini menggunakan metode *POST* yang merupakan langkah terakhir sebelum transaksi benar-benar selesai. Permintaan ini ibarat meletakkan uang tunai di meja kasir. Fungsinya adalah memberi tahu server metode pembayaran apa yang dipilih dan berapa jumlah uang yang diserahkan. Data yang dikirimkan berisi jumlah uang tunai sebanyak lima ratus ribu dan nomor pesanan yang sudah didapat dari korelasi data sebelumnya. Keberhasilan permintaan ini memastikan server memvalidasi pembayaran dan mengunci pesanan tersebut menjadikannya transaksi yang sah dan selesai.



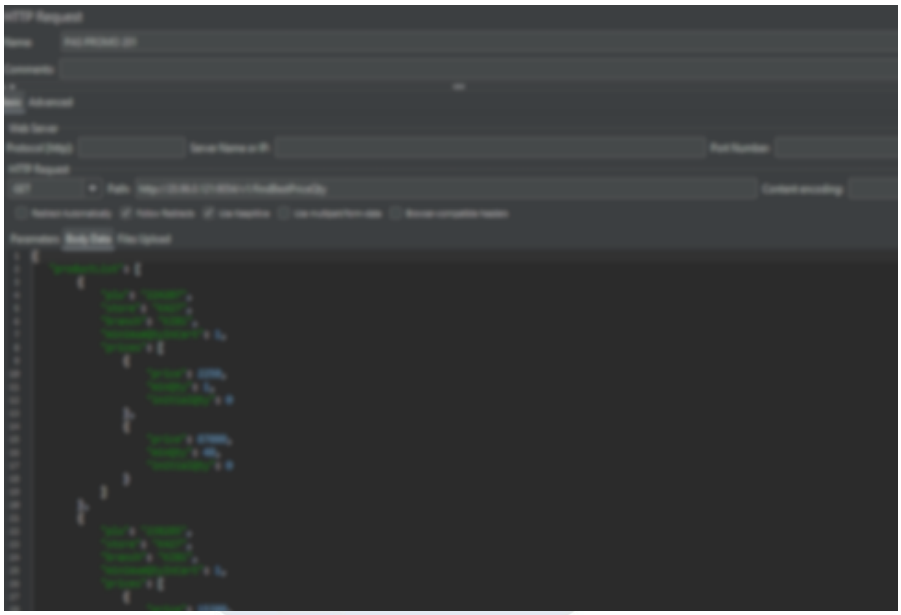
Gambar 3. 41 *HTTP Promo Special Price*

Dalam mengambil atau menampilkan daftar promosi harga khusus yang berlaku, *HTTP request* yang disebut *Promo Special Price* ini menggunakan metode *GET*. Permintaan ini ditujukan ke *endpoint* promosi harga khusus dan memiliki parameter yang dapat diatur di URL untuk mengatur tampilan data. Parameter ini termasuk jumlah baris yang diminta dan pengurutan hasilnya secara *descending*. Hal ini adalah permintaan pengambilan data, token otorisasi yang valid diperlukan dari *HTTP Header Manager* terkait untuk memastikan bahwa data promosi yang tepat dapat diakses. Permintaan ini mensimulasikan pengguna melihat katalog promosi dan memeriksa kemampuan server untuk memproses dan mengurutkan data promosi yang rumit.

Gambar 3. 42 *HTTP Request Promo Free Gift*

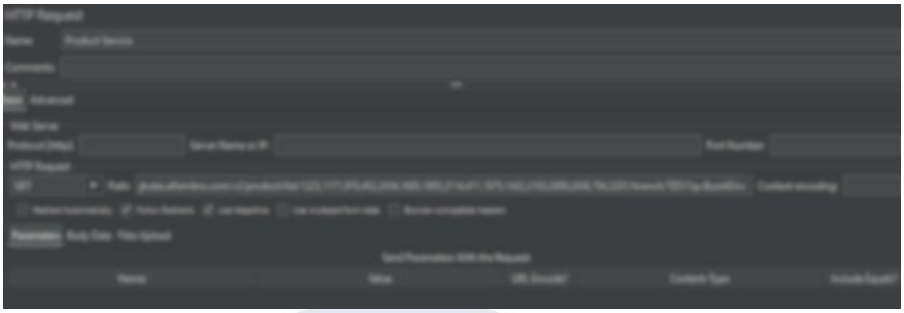
HTTP Request Promo Free Gift menggunakan metode *GET*. Metode ini memungkinkan Anda mengambil atau menampilkan daftar promosi hadiah gratis yang tersedia. Permintaan ini ditujukan ke *endpoint* produk yang berfokus pada promosi hadiah gratis. *Endpoint* tersebut memiliki parameter di URL untuk mengatur jumlah baris dan mengurutkan daftar hadiah secara menurun. Karena ini melibatkan pengambilan data, token otorisasi yang sah dari *HTTP Header Manager*

terkait diperlukan untuk memastikan bahwa data promosi yang tepat dapat diakses. Permintaan ini mensimulasikan pengguna melihat katalog hadiah promosi dan menguji kemampuan server untuk memproses dan mengurutkan data promosi yang rumit.



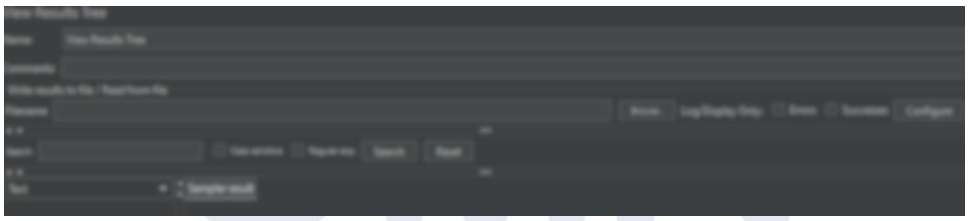
Gambar 3. 43 *HTTP PAS PROMO 201*

HTTP Request PAS Promo 201 ini menggunakan metode *GET*. Hal ini dapat mengambil data promosi tambahan dari endpoint tertentu yang. Permintaan ini menyertakan *body data* dalam format array JSON saat menggunakan metode *GET* dan ini digunakan untuk mengirimkan daftar panjang produk yang harganya ingin divalidasi atau dicari promosi. Permintaan ini, seperti permintaan pengambilan data promo lainnya, memerlukan token otorisasi yang valid dari *HTTP Header Manager* untuk menguji kemampuan server untuk menangani permintaan harga promosi yang sangat besar.



Gambar 3. 44 *HTTP Request Product Service*

HTTP Request Product Service ini menggunakan metode *GET*. Metode ini dapat digunakan untuk mengambil data detail dari daftar produk tertentu. Bukan hanya menampilkan satu produk, permintaan ini mengambil detail dari berbagai ID produk secara bersamaan, yang dimasukkan langsung ke dalam path URL. Selain ID produk, ada parameter yang mengatur hasil, seperti validasi ketersediaan di cabang dan pengurutan hasil secara menurun. Dalam memenuhi permintaan ini, diperlukan token otorisasi dari *HTTP Header Manager*. Ini digunakan untuk menguji kinerja server dalam memproses query yang sangat besar, misalnya untuk memverifikasi ketersediaan dan detail harga untuk berbagai produk sekaligus di lokasi cabang tertentu.



Gambar 3. 45 *View Result Tree*

View Result Tree adalah komponen esensial dalam pengujian performa yang berfungsi untuk menampilkan *output* berdasarkan permintaan *thread* yang dieksekusi. Alur otomasi yang disimulasikan oleh serangkaian *HTTP Request* ini berfokus pada pengujian pemrosesan *bulk data* dan kinerja server saat menangani query data yang kompleks dan panjang dalam satu permintaan. Permintaan seperti *Promo Special Price* dan *Promo Free Gift* menguji pengambilan dan pengurutan daftar promosi yang panjang, sementara *Product Service* menguji validasi detail dan ketersediaan banyak ID produk sekaligus yang dimasukkan dalam path URL.

Selain itu, PAS PROMO 201 menguji kemampuan server untuk memproses daftar produk massal yang dikirim dalam body untuk mengambil promosi. Setelah eksekusi, komponen *View Results Tree* digunakan untuk menganalisis *Sampler Result* untuk ketahanan server dan Respon *Data* untuk memverifikasi keakuratan semua data promosi dan produk massal yang dikembalikan. Alur *outomation* yang disimulasikan oleh serangkaian *HTTP Request* ini berkonsentrasi pada pengujian pemrosesan data massal dan meningkatkan kinerja server saat menangani permintaan data yang rumit dan panjang dalam satu permintaan. Permintaan seperti Promo Harga Khusus dan Promo Barang Gratis menguji pengumpulan dan pengurutan daftar promosi yang luas, sementara layanan produk menguji detail dan ketersediaan berbagai ID produk yang dimasukkan dalam path URL.

3.3.2 Kendala yang Ditemukan

Selama menjalankan tanggung jawab selama magang sebagai *Quality Assurance* di Aksesmu, terdapat beberapa tantangan seperti:

1) Perubahan kebutuhan yang berdampak pada hasil, seperti pengembangan yang seringkali menyebabkan perubahan sistem desain awal yang terjadi di tengah pengembangan. Kondisi ini sering muncul selama siklus pengembangan aplikasi menggunakan metode agile yang mengakibatkan ketidaksesuaian antara desain sistem dan pengujian yang telah dilakukan sebelumnya.
2) Skenario yang dirancang kadang-kadang kurang terstruktur dengan baik, yang mengakibatkan penundaan dalam publikasi hasil pengujian dan komunikasi antar anggota tim.
3) Tantangan dalam mengenali *edge cases* pada beberapa skenario pengujian yang tidak tercakup dalam desain sistem dan UI/UX yang disediakan, sehingga menyebabkan penemuan *bug* yang tidak terduga yang memerlukan penanganan lebih lanjut.

3.3.3 Solusi atas Kendala yang Ditemukan

Tindakan yang diambil saat menghadapi masalah selama pelaksanaan magang mencakup:

1) Mengatur perubahan kebutuhan dengan melaksanakan komunikasi dalam tim pengembang dan produk agar memahami perubahan sejak awal dan memperbarui skenario pengujian secara teratur agar tetap sejalan dengan perubahan desain sistem.
2) Meningkatkan mutu skenario pengujian supaya lebih terorganisasi dan mudah dimengerti. Selain itu, melakukan tinjauan bersama tim sebelum pengujian dimulai agar dapat memastikan skenario yang disusun.
3) Memperluas cakupan pengujian eksplorasi, termasuk *edge cases* yang mungkin terlewat dalam desain awal.

4) Meningkatkan kerjasama dalam tim dengan menyelenggarakan sesi diskusi agar semua anggota memahami perubahan yang ada dan mempercepat proses perbaikan serta peluncuran produk.