BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama masa magang di GLI, kegiatan difokuskan pada pengujian dan penjaminan kualitas aplikasi Alfagift sebagai bagian dari proses pengembangan dan pemeliharaan sistem. Peran yang dijalankan adalah sebagai QA, dengan tanggung jawab memastikan setiap fitur yang dikembangkan berfungsi sesuai dengan kebutuhan bisnis serta memberikan pengalaman terbaik bagi pengguna.

Kegiatan magang dilaksanakan secara *onsite* di bawah pengawasan langsung dari supervisor yang merupakan Manajer dari departemen Product Operations dan dimentori oleh karyawan dari tim tersebut. Dalam pelaksanaannya, pengujian dilakukan dengan menerapkan pendekatan *Black Box Testing*. Metode ini merupakan salah satu metode pengujian perangkat lunak yang membahas fungsionalitasnya dan tidak perlu melihat kode internal dari aplikasi yang diuji [7]. Pendekatan ini dilakukan melalui dua strategi utama: pertama, dengan melakukan eksplorasi mandiri terhadap fitur dan alur sistem untuk mengidentifikasi *bug*, *error*, maupun potensi masalah pada sisi fungsional dan tampilan; dan kedua, dengan melakukan validasi hasil pengujian melalui diskusi serta evaluasi rutin bersama supervisor dan tim pengembang.

Melalui pendekatan ini, tim QA berkontribusi dalam menjaga stabilitas, keandalan, dan konsistensi fitur aplikasi Alfagift, sehingga sistem yang dikembangkan dapat berfungsi secara optimal dan mendukung tujuan bisnis perusahaan dalam meningkatkan kepuasan pengguna.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kegiatan magang di PT Global Loyalty Indonesia, tugas yang dilakukan sebagai QA meliputi:

- Menganalisis *system requirement* dari fitur yang akan diuji coba.
- Melaksanakan pengujian *Full Cycle Rollout* untuk memastikan kualitas dan stabilitas sistem sebelum dirilis ke lingkungan produksi.
- Melaksanakan pengujian terhadap fitur-fitur baru pada aplikasi Alfagift guna

memverifikasi kesesuaian fungsionalitas dengan spesifikasi dan kebutuhan yang telah ditetapkan.

– Mendokumentasikan seluruh hasil pengujian, termasuk laporan hasil uji, temuan *bug*, serta tindak lanjut yang dilakukan selama proses pengujian.

– Menyusun skenario dan *test case* berdasarkan *requirement* dan skenario penggunaan aplikasi untuk memastikan cakupan pengujian yang komprehensif.

– Melakukan koordinasi dengan tim pengembang (*developer*) terkait isu atau *bug* yang ditemukan serta memverifikasi hasil perbaikan yang telah dilakukan.

Setiap tahap dilakukan secara iteratif untuk memastikan kualitas sistem yang konsisten dan sesuai standar perusahaan.

3.3 Uraian Pelaksanaan Magang

Kegiatan magang yang dilaksanakan di PT GLI merupakan bagian dari implementasi pembelajaran praktis di dunia industri. Fokus utama dari proyek ini adalah melakukan pengujian kualitas terhadap sistem dan fitur aplikasi Alfagift, guna memastikan fungsionalitas, keandalan, serta pengalaman pengguna berjalan sesuai standar yang ditetapkan.

Setiap aktivitas yang dilakukan selama pelaksanaan magang telah didokumentasikan secara sistematis dalam bentuk tabel, yang akan dijabarkan pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1 - 2	Perkenalan terkait lingkungan kerja GLI serta perkenalan *tools* yang digunakan oleh QA pada umumnya
3	Membuat skenario pada proyek penutupan akun
4	Menyiapkan lingkungan pengujian serta melakukan pengujian pada proyek penutupan akun
5	Melakukan *Full Cycle Rollout* Alfagift Beta
6	Membuat skenario dan melakukan pengujian terkait produk sama dengan voucer
7	Mempelajari promo-promo di dalam *database* serta ketentuan penggunanya untuk proyek selanjutnya
8	Melakukan *sprint planning* dan membuat skenario untuk pengujian *enhancement* voucer
9	Melakukan pengujian *enhancement* voucer
10	Mengikuti *sprint planning* serta membuat skenario Toko Event
11 - 12	Melakukan pengujian katalog dan pemesanan di Toko Event
13	Melakukan pengujian *Full Cycle Rollout* untuk versi yang terbaru
14	Membuat dokumentasi terkait pengujian yang sudah dilakukan
15 - 16	Mempelajari promo-promo terbaru di dalam *database* serta mempelajari Figma dan *storylist* untuk proyek selanjutnya

Selama proses pengerjaan tugas maupun proyek QA di perusahaan, berbagai perangkat atau *tools* dimanfaatkan yang mendukung pelaksanaan tugas tersebut, di antaranya sebagai berikut:

– Jira, perangkat lunak manajemen proyek dan *issue tracking* yang banyak digunakan dalam pengembangan perangkat lunak berbasis metode *Agile*. Dalam proses QA, Jira digunakan untuk mendokumentasikan *bug*, mencatat progres pengembangan, melakukan *sprint planning*, serta memantau alur kerja melalui *kanban board* atau *scrum board*. Penggunaan Jira memungkinkan setiap isu tercatat secara sistematis, sehingga mempermudah koordinasi antara QA, tim pengembang, dan tim terkait lainnya.

– Google Workspace, yang digunakan sebagai sarana kolaborasi digital yang

mencakup Google Docs, Google Sheets, dan layanan lainnya. Dalam kegiatan QA, alat ini mendukung pembuatan laporan pengujian, penyimpanan dokumen terkait kebutuhan pengujian, serta berbagi berkas antara anggota tim secara *real-time*. Penggunaan Google Workspace meningkatkan efisiensi komunikasi karena semua dokumen dapat diakses, diperbarui, dan ditinjau bersama secara sinkron.

– Apache SOLR, mesin pencarian berbasis Apache Lucene yang dirancang untuk menangani proses indeks dan pencarian data dalam skala besar. Dalam konteks QA, SOLR digunakan untuk memvalidasi hasil pencarian pada aplikasi, memastikan data yang di-*index* muncul sesuai relevansi, serta memastikan performa pencarian tetap optimal meskipun data terus bertambah. Pengujian sistem berbasis SOLR mencakup validasi *query*, pengukuran waktu respons, dan pengecekan konsistensi hasil pencarian.

– Redis, sistem *in-memory data structure store* yang biasanya digunakan sebagai *cache*, *message broker*, atau penyimpanan sementara. Dalam proses QA, Redis berperan penting dalam pengujian performa sistem karena banyak proses aplikasi bergantung pada data *cache* untuk mempercepat *response time*. Pengujian dilakukan dengan memverifikasi apakah data *cache* diperbarui dengan benar, bagaimana aplikasi berperilaku ketika *cache* kosong, dan apakah perubahan data tercermin pada aplikasi secara *real-time*.

– Figma, alat perancangan antarmuka pengguna (UI) dan pengalaman pengguna (UX) yang berjalan secara kolaboratif di web. QA menggunakan Figma untuk memahami rancangan tampilan dan alur fitur sebelum implementasi dilakukan. Dengan membandingkan desain di Figma dan hasil implementasi pada aplikasi, QA dapat memastikan kesesuaian elemen UI, gaya visual, konsistensi *layout*, dan alur interaksi pengguna. Proses ini membantu mengidentifikasi ketidaksesuaian sejak tahap awal.

– Swagger, atau dikenal sebagai *OpenAPI Documentation*, adalah dokumentasi interaktif yang digunakan untuk menampilkan dan menguji *endpoint API* yang disediakan oleh tim *backend*. Dalam kegiatan QA, Swagger digunakan untuk memvalidasi struktur *request* dan *response*, memastikan *status code* yang diterima sesuai standar, serta menguji berbagai skenario.

– MongoDB, adalah basis data NoSQL yang menyimpan data dalam bentuk dokumen (JSON). Untuk QA, MongoDB dipakai untuk meninjau data yang

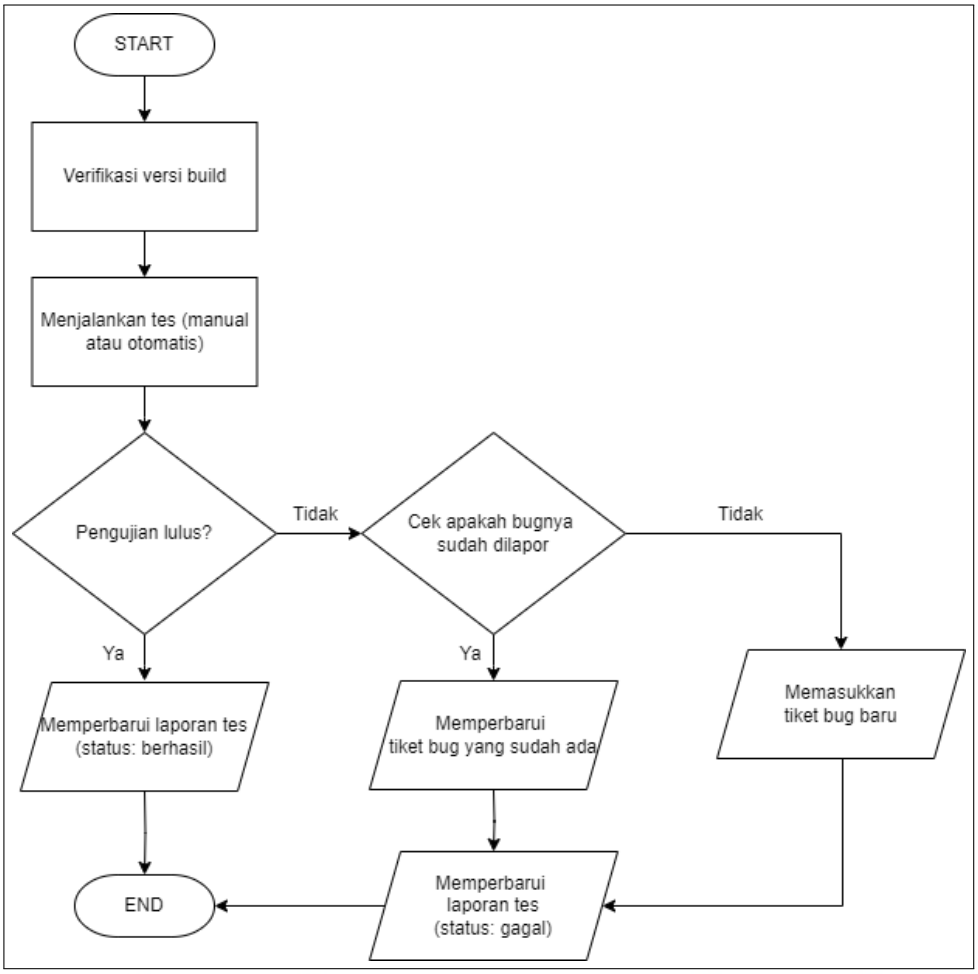
tersimpan setelah panggilan API dijalankan seperti memeriksa apakah data yang dibuat, diperbarui, atau dihapus sesuai dengan skenario uji. QA juga bisa menggunakan MongoDB untuk menyiapkan data uji, membersihkan *state* antar *test case*, dan melakukan pengecekan integritas data.

– DBeaver, alat sisi klien basis data yang memudahkan eksplorasi dan manajemen basis data (baik SQL maupun beberapa koneksi NoSQL). QA memakai DBeaver untuk melihat struktur tabel/koleksi, menjalankan kueri manual, mengekspor/impor data uji, dan cepat memverifikasi hasil perubahan di basis data tanpa harus bergantung pada UI aplikasi.

– Postman, alat pengujian API yang membantu QA berinteraksi langsung dengan *endpoint backend* tanpa harus melalui antarmuka aplikasi. Postman digunakan untuk mengirim permintaan HTTP (seperti GET, POST, PUT, dan DELETE), menguji respon server, memvalidasi struktur dan isi *payload*, serta memastikan setiap *endpoint* bekerja sesuai spesifikasi.

3.3.1 Alur Pengujian

Selama masa magang, proses pengujian dilakukan melalui beberapa tahapan yang saling terhubung. Setiap tahapan memiliki peran penting untuk memastikan bahwa fungsionalitas sistem berjalan sesuai dengan kebutuhan yang telah ditetapkan. Gambaran lengkap mengenai alur pengujian tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.1. Alur umum proses pengujian

Pada tahap awal, versi *build* aplikasi harus diverifikasi berdasarkan kebutuhan sistem atau fitur yang sedang diuji. Setelah versi aplikasi sudah benar akan melaksanakan eksekusi skenario uji pada aplikasi, baik melalui antarmuka pengguna (UI) maupun langsung pada layanan *backend* seperti API. Pengujian dilakukan dengan berinteraksi sesuai langkah-langkah pada *test case* untuk memverifikasi perilaku sistem.

Setelah *build* terverifikasi, proses pengujian dijalankan sesuai kebutuhan. Pengujian dapat dilakukan secara manual melalui antarmuka aplikasi, ataupun secara otomatis menggunakan skrip atau *tools* pendukung. Hasil eksekusi tes kemudian dievaluasi. Jika sistem berperilaku sesuai harapan dan seluruh skenario uji terpenuhi, maka pengujian dinyatakan lulus. Namun, apabila ditemukan ketidaksesuaian, *bug*, atau perilaku yang tidak sesuai dengan *requirement*, maka pengujian dianggap gagal.

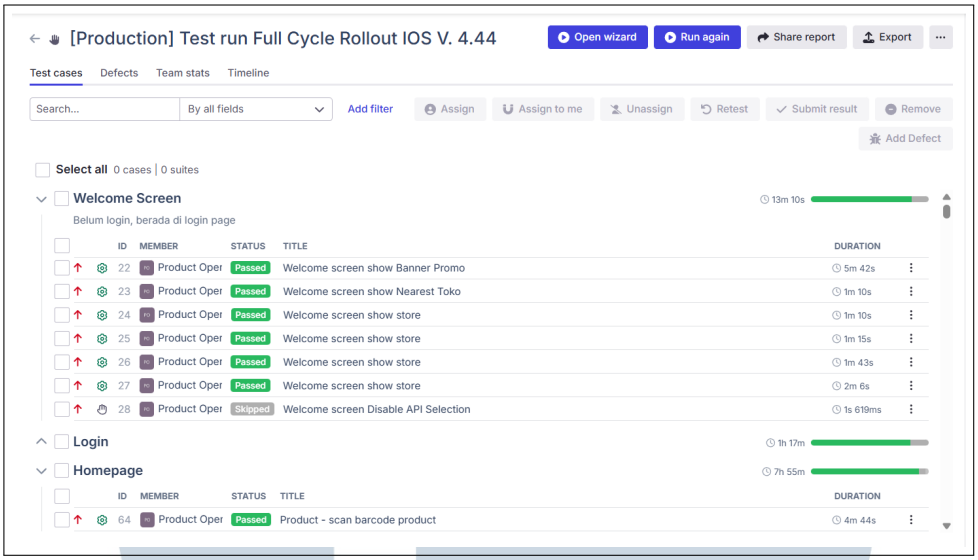
Jika pengujian dinyatakan lulus, pembaruan pengujian akan dilaporkan dengan status berhasil. Laporan ini berfungsi sebagai dokumentasi formal bahwa fitur telah divalidasi dan tidak ditemukan isu. Namun, apabila hasil tes gagal, perlu ditentukan apakah *bug* atau isu tersebut sudah pernah dilaporkan sebelumnya.

Jika *bug* sudah pernah dilaporkan, tiket *bug* yang sudah ada diperbarui dengan menambahkan bukti baru seperti *screenshot*, langkah reproduksi terbaru, ataupun detail lain yang relevan. Akan tetapi, jika *bug* belum pernah dilaporkan tiket *bug* baru akan dibuat untuk menginformasikan temuan tersebut kepada tim pengembang. Tiket ini berisi detail lengkap seperti deskripsi *bug*, langkah reproduksi, lingkungan pengujian, serta bukti pendukung.

Setelah penanganan *bug* selesai, penguji memperbarui laporan tes dengan status gagal, disertai catatan tambahan mengenai *bug* yang ditemukan dan referensi tiket yang telah dibuat atau diperbarui. Akhirnya, seluruh proses pengujian berakhir setelah laporan tes diperbarui. Tahapan ini memastikan bahwa setiap hasil pengujian terdokumentasi dengan jelas dan dapat ditindaklanjuti oleh tim terkait.

3.3.2 Full Cycle Rollout

Full Cycle Rollout (FCR) adalah tahapan pengujian aplikasi Alfagift secara komprehensif yang dilakukan sebelum rilis versi aplikasi terbaru ke *platform* distribusi. Proses ini mencakup pengujian pada perangkat Android maupun iOS, serta meliputi seluruh skenario penggunaan yang tersedia di dalam aplikasi. FCR dilakukan secara rutin dalam satu sampai tiga bulan sekali. Gambar 3.2 merupakan tampilan *test case* dari FCR yang dilakukan selama masa magang.

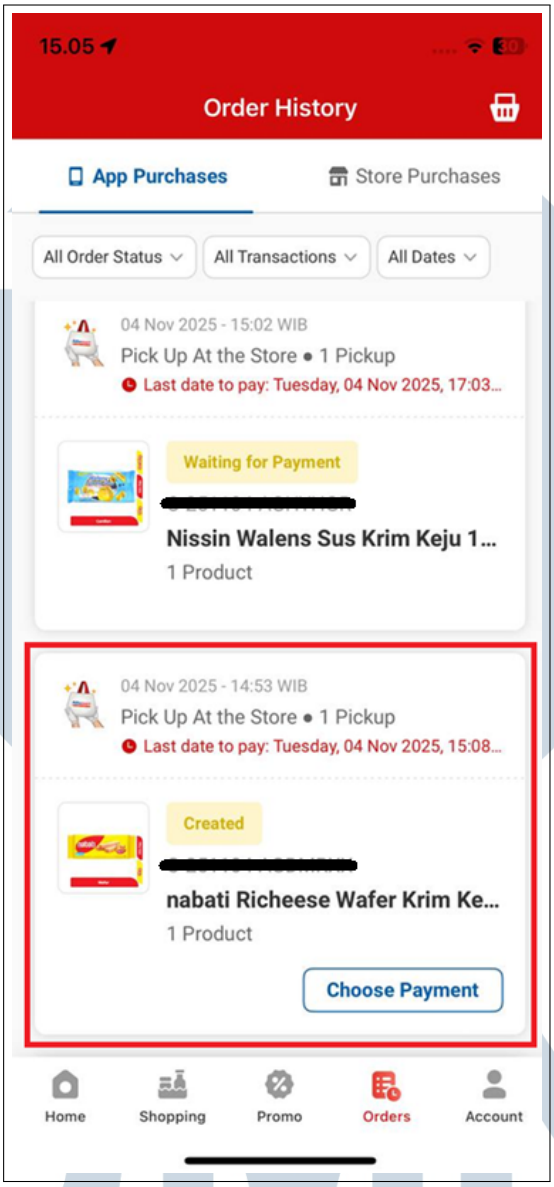


Gambar 3.2. Tampilan Pengujian *Full Cycle Rollout*

Pada Gambar 3.2, terlihat versi beta yang sedang diuji, FCR terdiri atas 554 *test case* yang terus diperbarui sesuai perkembangan fitur dan perubahan sistem. Cakupan pengujiannya meliputi seluruh fungsi aplikasi, mulai dari *Welcome Screen* hingga proses pemesanan produk secara daring.

Proyek ini merupakan kegiatan kolaboratif yang melibatkan seluruh peserta magang pada divisi Product Operations dengan pendampingan langsung dari karyawan tetap. Dari total 554 *test case* yang disiapkan dalam rangkaian FCR, seluruh skenario tersebut kemudian didistribusikan secara proporsional kepada setiap anggota tim untuk dieksekusi secara manual. Pembagian ini dilakukan agar keseluruhan cakupan pengujian dapat diselesaikan secara efisien dan menyeluruh.

Pembagian tugas secara merata diperlukan karena proses FCR memiliki batas waktu yang ketat, yakni maksimal tiga hari sejak pengujian dimulai. Dengan alokasi waktu yang terbatas tersebut, pengerjaan secara paralel menjadi langkah yang paling efektif untuk memastikan seluruh skenario dapat divalidasi tanpa mengurangi kualitas maupun ketelitian dalam proses peninjauan. Berikut adalah salah satu contoh pengetesan pembayaran ketika melakukan FCR.



Gambar 3.3. Contoh Pengujian Pemesanan

Pada skenario yang terlihat pada Gambar 3.3, setiap langkah verifikasi mulai dari pembuatan pesanan hingga konfirmasi pembayaran diperiksa untuk memastikan tidak terjadi kegagalan fungsi.

3.3.3 Pengujian Voucer

Pengujian voucer dilakukan untuk memastikan bahwa alur penukaran maupun penggunaan voucer tetap berfungsi dengan benar setelah adanya perubahan pada tampilan pesan kesalahan (*error message*). Perubahan ini memerlukan

proses pengujian ulang (*retesting*) untuk memverifikasi bahwa tampilan baru telah ditampilkan sesuai rancangan, serta memastikan bahwa perilaku sistem tidak mengalami regresi pada bagian lainnya.

A Pengaturan voucer

Sebelum voucer dapat digunakan dalam proses pengujian, tahap awal yang harus dilakukan adalah memverifikasi status voucer tersebut untuk memastikan bahwa voucer berada dalam kondisi aktif. Validasi ini penting karena voucer yang tidak aktif tidak dapat diproses oleh sistem dan dapat menghasilkan respons yang tidak sesuai dengan tujuan pengujian.

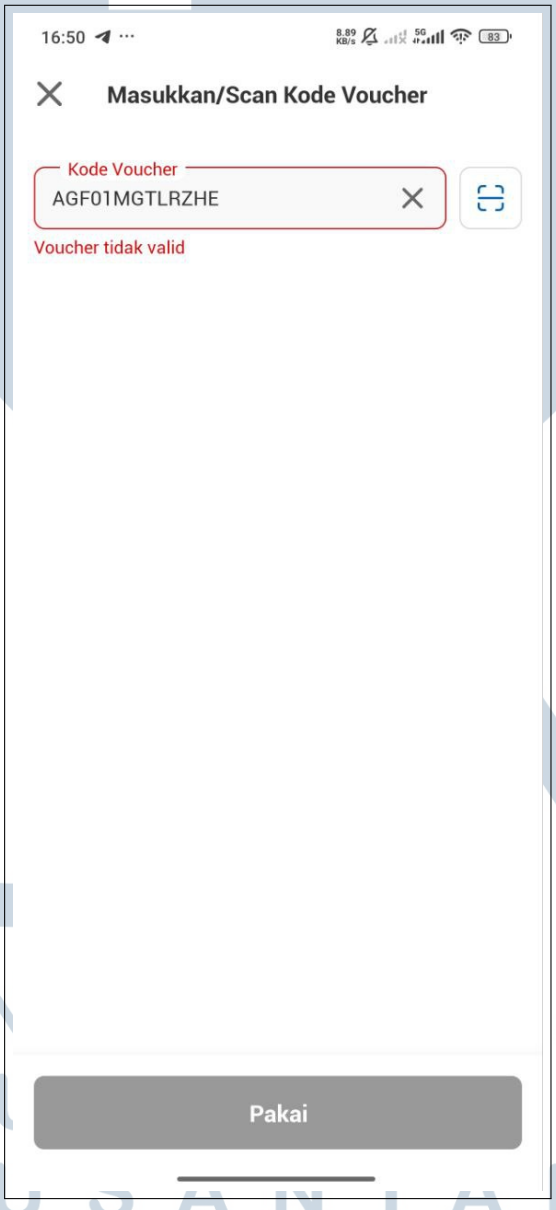
Lalu, status aktif pada voucer tidak ditentukan langsung dari tabel voucer itu sendiri, tetapi diatur melalui status *invoice* yang terkait. Oleh karena itu, sebelum pengujian dimulai, perlu dipastikan bahwa *invoice* yang menjadi acuan voucer telah berada pada kondisi aktif. Aktivasi *invoice* dilakukan dengan menyesuaikan atribut yang mengatur masa berlaku, status penerbitan, serta parameter lain yang memengaruhi keaktifan voucer. Setelah mengaktifkan *invoice*, voucer juga harus dipastikan aktif di tabel voucer dengan membuat nilai dari kode voucer yang digunakan menjadi *True*.

Setelah syarat voucer valid, *database product benefit* digunakan untuk memeriksa produk yang akan menjadi keuntungan di *invoice* yang digunakan sebagai pendukung aktivasi voucer. Dalam tahap ini, perlu dipastikan bahwa *invoice* tersebut memiliki manfaat atau nilai tambah produk yang masih aktif. Verifikasi ini memastikan bahwa voucer dapat diterapkan pada transaksi sesuai aturan yang telah ditentukan, terutama terkait hubungan antara voucer dan jenis produk yang memperoleh keuntungan tertentu.

Hal ini memastikan bahwa produk yang menjadi objek pengujian memiliki ketersediaan yang sesuai. Stok produk harus diperiksa terlebih dahulu, mengingat informasi stok biasanya disimpan dan dikelola melalui sistem Redis. Pengecekan ini dilakukan untuk memastikan bahwa voucer dapat diuji dalam kondisi yang representatif, sehingga hasil pengujian mencerminkan situasi yang sesuai dengan operasional sebenarnya.

B Perbandingan Tampilan Pesan Kesalahan

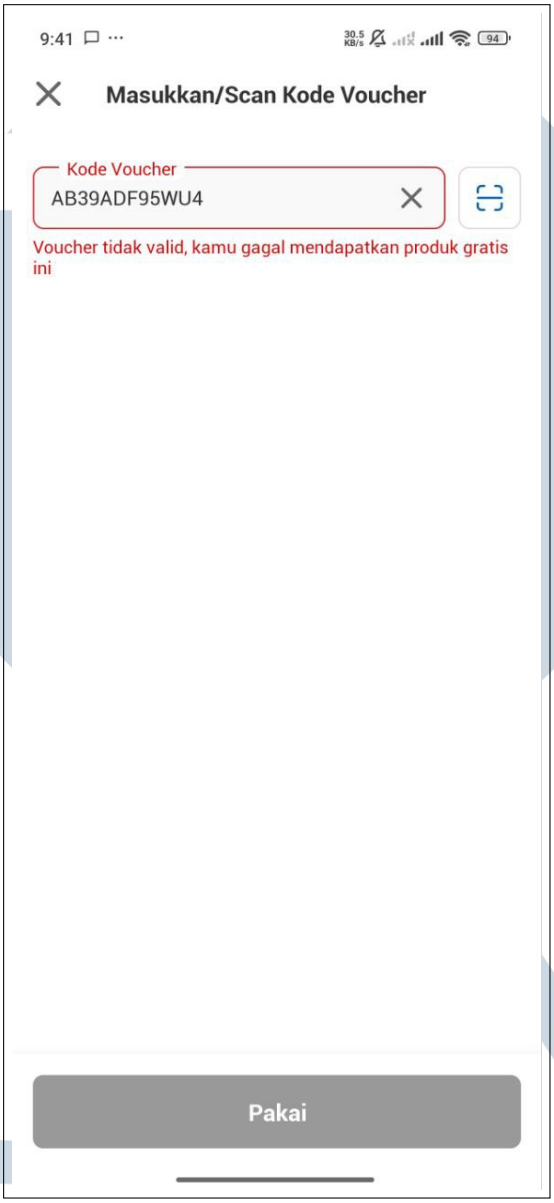
Dalam pengujian ini, skenario difokuskan pada kondisi ketika voucer tidak dapat digunakan, baik karena tidak memenuhi syarat, telah kedaluwarsa, maupun tidak sesuai dengan ketentuan transaksi. Setiap kondisi tersebut dieksekusi ulang untuk memastikan bahwa pesan kesalahan yang ditampilkan sesuai dengan desain terbaru.



Gambar 3.4. Tampilan Pesan Kesalahan Sebelum Perubahan

Gambar 3.4 menunjukkan tampilan pesan kesalahan sebelum dilakukan pembaruan. Pada versi ini, informasi terkait penyebab ketidakberlakuan voucer

masih belum ditampilkan secara jelas.



Gambar 3.5. Tampilan Pesan Kesalahan Setelah Dilakukan Pembaruan UI

Gambar 3.5 memperlihatkan tampilan terbaru setelah penerapan pembaruan antarmuka. Perbaikan dilakukan pada struktur pesan kesalahan agar informasi yang ditampilkan lebih jelas, informatif, dan selaras dengan desain visual aplikasi.

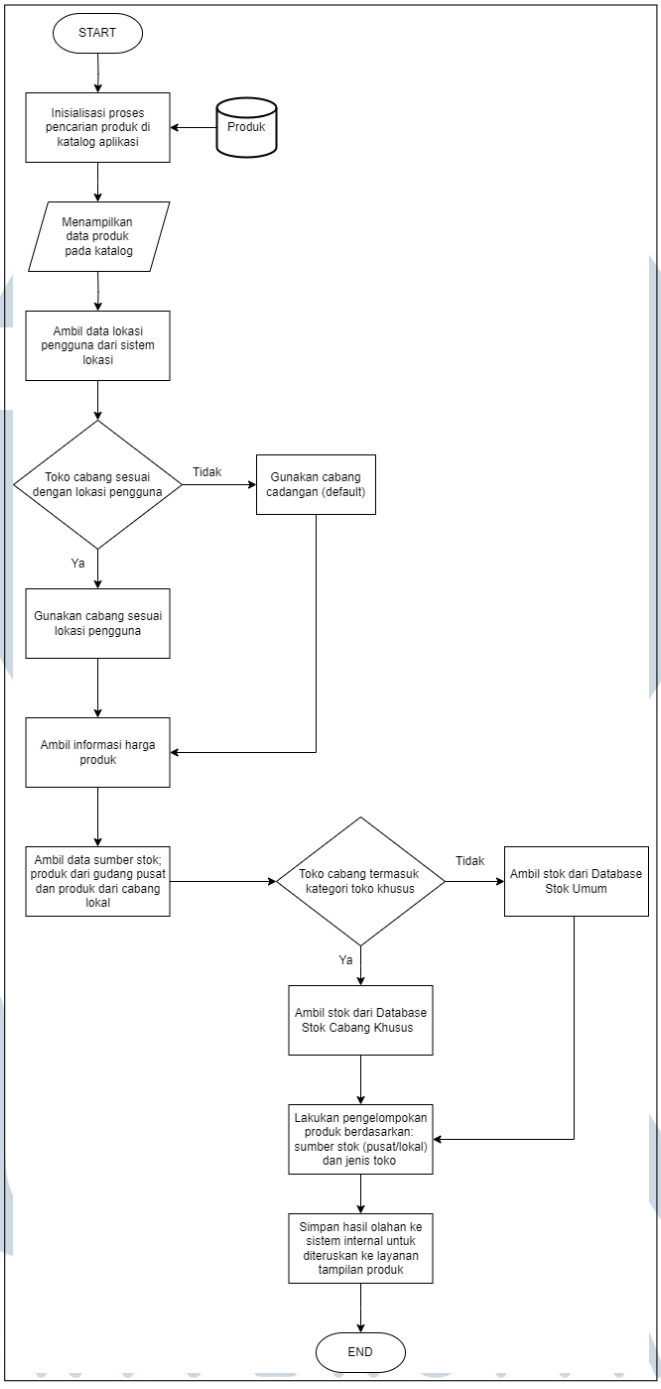
Setelah seluruh skenario dijalankan, hasil pengujian menunjukkan bahwa perubahan tampilan berhasil diterapkan dan seluruh pesan kesalahan ditampilkan sesuai dengan rancangan terbaru. Selain itu, tidak ditemukan adanya gangguan pada fitur lain yang terkait dengan proses validasi voucer.

3.3.4 Pengujian Toko Khusus

Toko Khusus merupakan sebuah fitur yang dirancang untuk memberikan proses pembelian yang lebih cepat dan efisien bagi pengguna. Dalam kondisi tertentu, sistem akan mengklasifikasikan suatu cabang sebagai Toko Khusus sehingga produk yang tersedia di cabang tersebut dapat diproses dan dibeli secara lebih instan. Mekanisme ini memungkinkan pengguna memperoleh pengalaman belanja yang lebih nyaman, karena sistem dapat memprioritaskan sumber stok tertentu dan menyederhanakan alur pemenuhan pesanan. Dengan adanya fitur ini, aplikasi dapat menampilkan ketersediaan produk yang lebih akurat sekaligus mempercepat proses transaksi dibandingkan toko dengan kategori umum.

A Diagram Alir Sistem Toko Khusus

Diagram alir berikut digunakan untuk menggambarkan alur proses sistem Toko Khusus dalam menentukan cabang, sumber stok, dan informasi harga produk yang ditampilkan pada katalog aplikasi Alfagift.



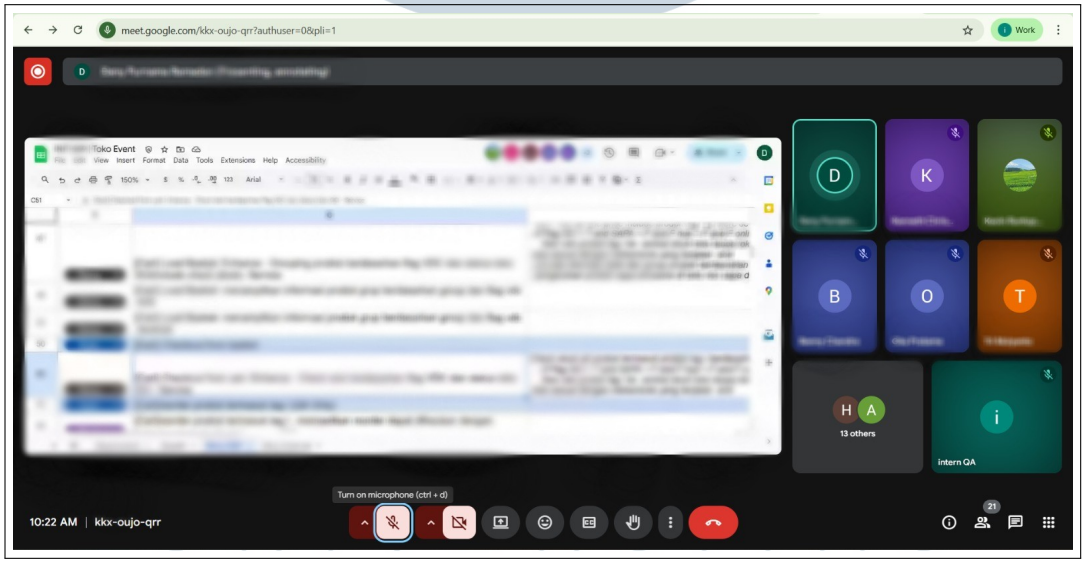
Gambar 3.6. Tampilan Diagram Alir Toko Khusus

Pada Gambar 3.6, proses pencarian dan penentuan stok produk pada katalog aplikasi dimulai dengan pengambilan data produk serta lokasi pengguna. Berdasarkan lokasi tersebut, sistem menentukan cabang yang akan digunakan. Cabang yang digunakan akan menyesuaikan lokasi ataupun cabang cadangan. Cabang ini kemudian menjadi acuan untuk pengambilan data harga dan sumber

stok. Sistem selanjutnya mengumpulkan data stok dari gudang pusat, cabang lokal, atau *database* stok khusus apabila cabang yang digunakan termasuk kategori Toko Khusus. Setelah data stok dan harga diperoleh, sistem melakukan pengelompokan produk sesuai sumber stok dan jenis toko. Hasil pengolahan tersebut kemudian disimpan dan diteruskan ke layanan tampilan produk agar pengguna menerima informasi yang relevan dan akurat.

B Persiapan Pengujian Toko Khusus

Ketika memulai suatu proyek pengujian baru, tim QA bersama tim pengembang terlebih dahulu melaksanakan kegiatan *sprint planning*. Tahap ini bertujuan untuk memperoleh pemahaman yang selaras mengenai tujuan, ruang lingkup, serta persyaratan yang harus dipenuhi dalam pengujian fitur Toko Khusus. Melalui proses perencanaan ini, setiap pihak yang terlibat dapat mengidentifikasi kebutuhan teknis, risiko awal, serta batasan yang mungkin muncul selama proses pengujian berlangsung. Gambar 3.7 menampilkan dokumentasi kegiatan *sprint planning* yang dilakukan pada tahap awal.

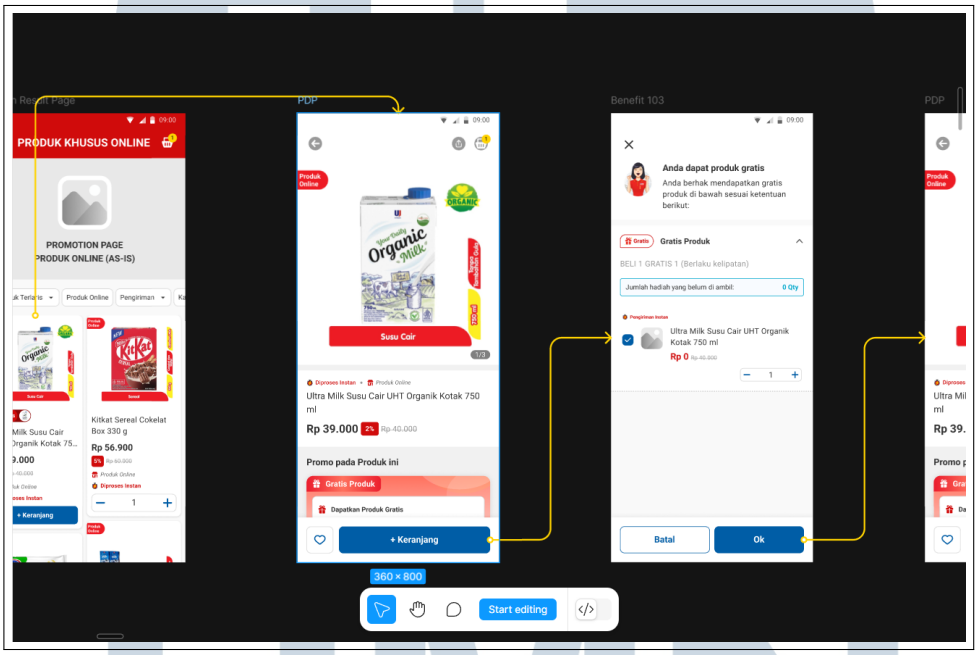


Gambar 3.7. *Sprint Planning* Pengembangan Fitur Toko Khusus

Setelah proses *sprint planning* diselesaikan, tim QA kemudian mulai menyusun skenario pengujian secara lebih mendetail. Penyusunan skenario ini dilakukan untuk memperjelas alur pengujian, menentukan kondisi prasyarat, serta mendefinisikan hasil yang diharapkan dari setiap langkah pengujian. Seluruh skenario dirancang berdasarkan *storylist* yang telah disusun tim pengembang dan

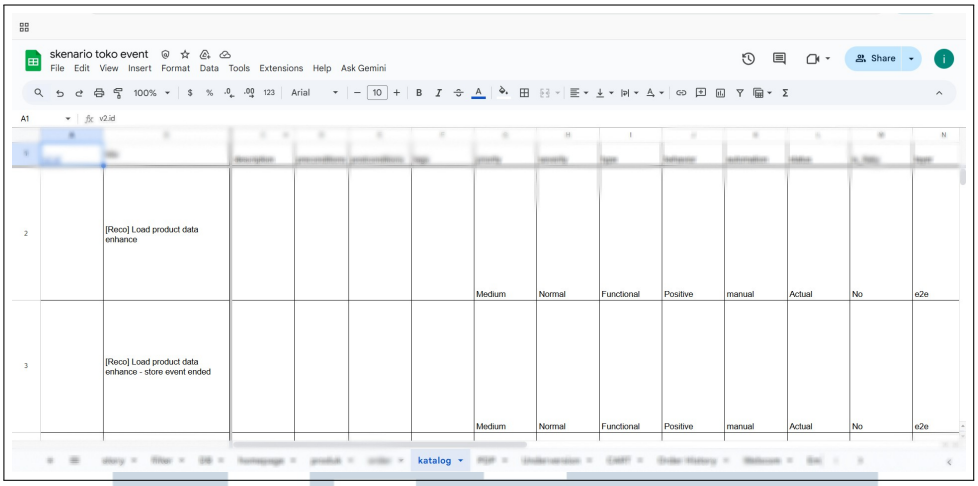
tim bisnis, sehingga setiap bagian pengujian dapat mencerminkan kebutuhan dan perilaku sistem yang sebenarnya.

Selain mengacu pada *storylist*, skenario pengujian juga disusun dengan mempertimbangkan rancangan antarmuka yang dibuat oleh tim desainer aplikasi. Desain tersebut memberikan gambaran konkret mengenai perilaku visual dan fungsional yang diharapkan, sehingga membantu tim QA dalam menyusun skenario yang lebih akurat dan sesuai implementasi. Contoh rancangan antarmuka (Figma) yang digunakan sebagai referensi ditunjukkan pada Gambar 3.8.



Gambar 3.8. Rancangan Antarmuka sebagai Acuan Penyusunan Skenario

Setelah keseluruhan kebutuhan dan rancangan dipahami dari *storylist* dan Figma, skenario pengujian mulai disusun di dalam Google Spreadsheets. Penggunaan Spreadsheets memungkinkan kolaborasi secara langsung antara sesama anggota tim QA, baik dalam penyusunan langkah-langkah pengujian maupun dalam proses verifikasi skenario yang telah disusun. Gambar 3.9 rancangan skenario yang dibentuk di Google Spreadsheets.



Gambar 3.9. Pembentukkan Skenario dengan Google Spreadsheets

Apabila seluruh skenario yang termuat di Google Spreadsheets telah dinyatakan sesuai dengan *storylist* dan rancangan Figma, dokumen tersebut akan diimpor ke dalam alat yang lain agar pengujian dapat dilaksanakan secara lebih terstruktur, terdokumentasi, dan mudah dilacak hingga tahap penyelesaian.

C Pelaksanaan Pengujian Toko Khusus

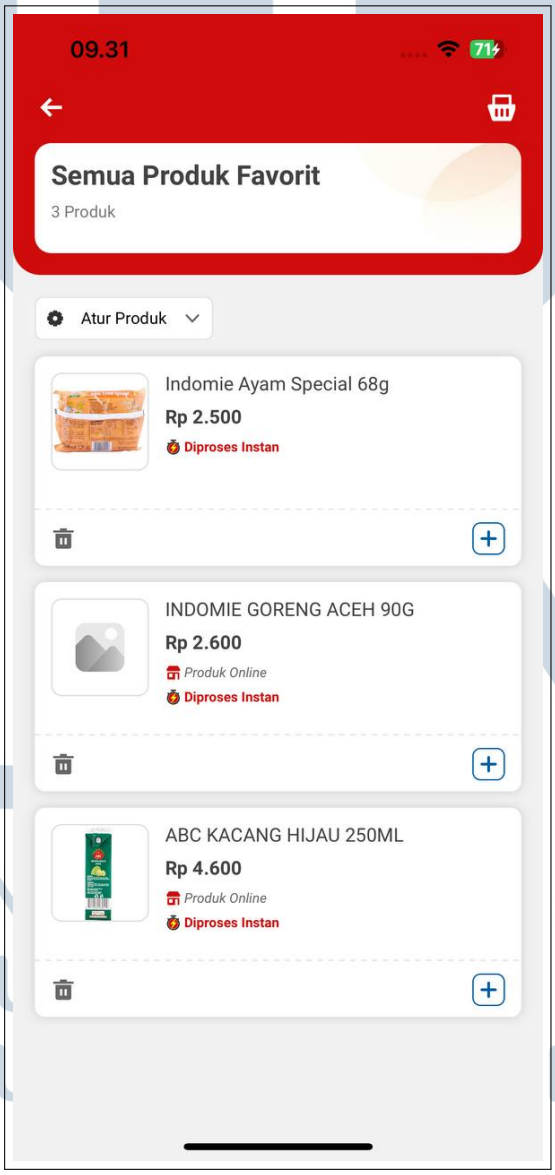
Setelah seluruh skenario pengujian disusun dan diimpor ke dalam alat lain, proses berlanjut pada tahap pelaksanaan pengujian. Tahap ini merupakan inti dari keseluruhan rangkaian QA karena memastikan bahwa setiap skenario yang sudah didefinisikan dapat dieksekusi dan divalidasi sesuai dengan hasil yang diharapkan. Setiap skenario dijalankan berdasarkan prioritas, tingkat kompleksitas, serta ketergantungannya terhadap fitur lain.

Selama proses ini, tim QA bertanggung jawab melakukan verifikasi terhadap setiap respons sistem, tampilan antarmuka, serta perilaku fitur Toko Khusus dalam skenario normal maupun kondisi ekstrem. Setiap langkah harus mengikuti prasyarat yang telah ditentukan agar data pengujian valid dan tidak bias. Dengan cara ini, seluruh aktivitas pengujian dapat direkam secara sistematis dan mudah ditelusuri apabila terjadi ketidaksesuaian. Sebelum menguji tampilan di katalog, toko yang digunakan untuk pengujian harus dilakukan intervensi di basis data toko pada MongoDB untuk menambahkan beberapa parameter demi menyesuaikan kebutuhan Toko Khusus.

Dalam praktiknya, ruang lingkup pengujian Toko Khusus dibagi ke dalam beberapa kelompok QA berdasarkan modul aplikasi. Pada kasus ini, fokus

pengujian berada pada bagian katalog yang berperan sebagai salah satu komponen utama dalam pengalaman belanja pengguna Alfagift. Struktur katalog mencakup berbagai bagian, seperti *Rekomendasi untuk Kamu, Produk Favorit, Shopping List, Belanja Rutin, Official Store, Official Store Search,* serta elemen lain yang mendukung penyajian produk sesuai kebutuhan pelanggan.

Salah satu contoh implementasi Toko Khusus yang diuji terdapat pada katalog *Produk Favorit*, yang menampilkan barang sesuai preferensi pengguna berdasarkan parameter toko aktif. Contoh tampilan tersebut ditunjukkan pada Gambar 3.10.



Gambar 3.10. Tampilan Katalog *Produk Favorit* di Toko Khusus

Setiap eksekusi skenario akan menghasilkan status akhir berupa *passed* atau *failed*. Apabila terdapat ketidaksesuaian antara hasil aktual dan hasil yang diharapkan, temuan tersebut dicatat sebagai *bug* dan dilaporkan kepada tim pengembang melalui alat seperti JIRA. Laporan *bug* harus dilengkapi dengan langkah reproduksi, kondisi lingkungan pengujian, ekspektasi, kondisi aktual, serta bukti pendukung seperti *screenshot* atau rekaman layar. Dengan dokumentasi yang lengkap, tim pengembang dapat melakukan analisis dan perbaikan lebih cepat serta presisi. Tim QA mengadopsi penggunaan JIRA untuk melaporkan *bug* agar lebih mudah untuk diawasi.

Apabila perbaikan telah selesai dilakukan oleh tim pengembang, proses dilanjutkan dengan tahap *retest* untuk memastikan bahwa isu telah teratasi dan tidak menimbulkan regresi pada modul lain. Siklus pengujian ini dapat berlangsung berulang kali hingga seluruh skenario mencapai status *passed* dan tidak ditemukan isu kritikal yang berpotensi mengganggu fungsi utama Toko Khusus.

Pelaksanaan pengujian juga diperkuat dengan penerapan pengujian *automation*. Penggunaan *automation* menjadi semakin penting karena pola skenario pada katalog memiliki struktur yang berulang, mulai dari memuat daftar produk, memvalidasi visibilitas produk, memastikan respons saat produk diklik, hingga memeriksa perubahan pada elemen UI. Efektivitas *automation* semakin terlihat karena struktur skenario katalog pada dasarnya berulang dan berfokus pada proses pemuatan halaman. Perbedaan antara kasus yang dites biasanya hanya berkaitan dengan syarat aktivasi Toko Khusus, sehingga otomatisasi pengujian dapat dijalankan dengan konsisten tanpa perlu melakukan variasi signifikan pada langkah-langkah utamanya.

Skrip *automation* dirancang untuk meniru langkah yang dilakukan penguji manual, mulai dari navigasi halaman katalog dan verifikasi konten. Dengan eksekusi yang lebih cepat, tim QA dapat memperoleh hasil regresi dalam waktu singkat dan memprioritaskan waktu yang digunakan untuk melakukan analisis lebih dalam terhadap hasil pengujian.

```
1 public class MobileTest {
2
3     public static void main(String[] args) throws Exception {
4
5         UiAutomator2Options options = new UiAutomator2Options();
6         options.setPlatformName("Android");
7         options.setDeviceName("Android Device");
8         options.setAppPackage("com.example.app");
```

```
        options.setAppActivity("com.example.app.MainActivity");

        AndroidDriver driver = new AndroidDriver(
                new URL("http://127.0.0.1:4723"), options);

        // Navigasi kategori
        driver.executeScript("mobile: scrollGesture", Map.of(
                "direction", "down", "percent", 0.7));
        driver.findElement(By.xpath(
                "//android.widget.TextView[@text='Kategori A']")).
    click();

        // Pencarian produk
        driver.findElement(By.id("com.example.app:id/btn_search"))
    .click();
        driver.findElement(By.id("com.example.app:id/input_search"
    ))
              .sendKeys("produk contoh");
        driver.pressKey(new KeyEvent(AndroidKey.ENTER));

        driver.quit();
    }
}
```

Kode 3.1: Contoh Implementasi Appium dan Java

Pengujian otomatis pada aplikasi Android dilakukan menggunakan Appium dengan bahasa pemrograman Java. Pada Kode 3.1, nama aplikasi disamarkan menggunakan paket ”com.example.app” untuk menjaga kerahasiaan.

Skrip melakukan tiga tahapan utama, pertama adalah inisialisasi Appium dengan konfigurasi perangkat dan aplikasi. Tanpa konfigurasi, aplikasi tidak akan bisa dijalankan. Lalu, tahapan selanjutnya yang dibahas adalah navigasi ke salah satu katalog menggunakan perintah *scroll*. Tahap terakhir dari kode tersebut adalah untuk pencarian produk melalui fitur pencarian di aplikasi. Kode ini menunjukkan proses dasar *automation* yang digunakan untuk menguji interaksi antarmuka aplikasi Android.

Seluruh temuan dan kemajuan pengujian kemudian didokumentasikan secara terstruktur melalui Google Docs. *Platform* ini dipilih karena memungkinkan kolaborasi antara anggota tim, sehingga setiap perubahan, catatan, atau pembaruan hasil pengujian dapat langsung dipantau bersama. Dokumen tersebut berisi daftar skenario, hasil harian, rangkuman isu, serta tindak lanjut yang perlu

dilakukan. Tidak hanya itu, Google Docs juga digunakan sebagai tempat penyimpanan bukti tambahan seperti *screenshot*, diagram alur, atau referensi lain yang berkaitan langsung dengan proses pengujian. Dokumentasi yang rapi membantu memudahkan evaluasi, koordinasi, dan penelusuran kembali apabila diperlukan verifikasi ulang di kemudian hari.

3.4 Kendala dan Solusi yang Ditemukan

Selama melaksanakan kegiatan magang, terdapat beberapa kendala yang dihadapi dalam melaksanakan tugas QA. Adapun kendala tersebut antara lain sebagai berikut:

1. Keterbatasan pengalaman dalam menggunakan berbagai *tools* yang berkaitan dengan proses QA, seperti Redis, Postman, dan perangkat pendukung lainnya.
2. Kompleksitas alur pengujian pada setiap fitur yang diuji, sehingga memerlukan pemahaman yang lebih mendalam terkait proses bisnis dan alur sistem.

Berdasarkan kendala yang ditemukan, beberapa solusi yang dilakukan selama pelaksanaan magang adalah sebagai berikut:

1. Melakukan pembelajaran mandiri melalui dokumentasi internal, referensi daring, serta sumber pengetahuan lain untuk memahami penggunaan fitur atau alat-alat pembantu pengujian yang sebelumnya belum dikuasai.
2. Memastikan pemahaman terhadap alur dokumen dan kebutuhan sistem dengan berkomunikasi secara aktif dengan supervisor, serta melakukan konfirmasi terhadap *user requirement* secara berkala untuk menghindari kesalahan interpretasi.