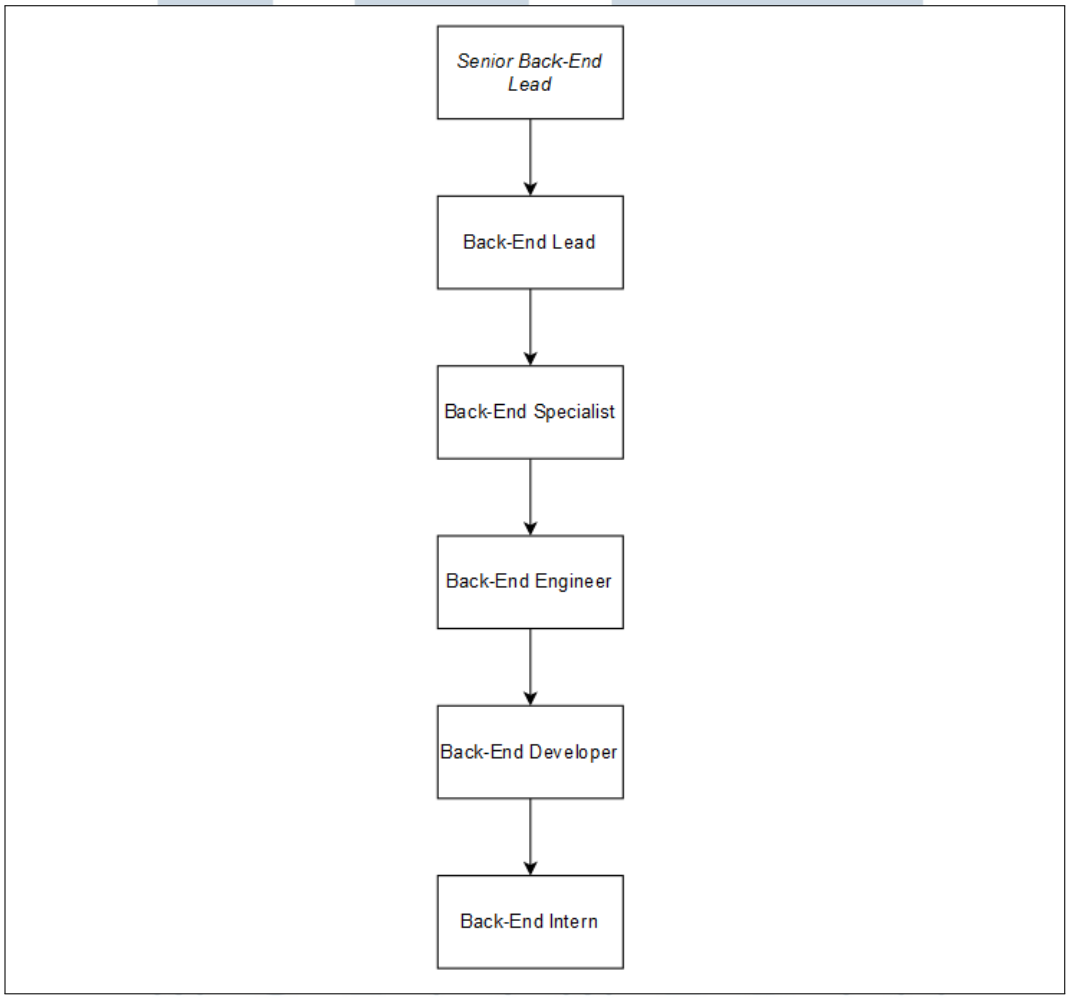
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Di PT Omni Digitama Internusa, Departemen *Back-End* terbagi menjadi beberapa tingkatan seperti yang tercantum pada Gambar 3.1, yang terdiri dari *Senior Back-End Lead*, *Back-End Lead*, *Back-End Specialist*, *Back-End Engineer*, *Back-End Developer*, dan *Back-End Intern*.



Gambar 3.1. Struktur Organisasi Departemen *Back-End* PT Omni Digitama Internusa

Departemen ini dipimpin oleh Bapak Rusmin Subiakto sebagai *Senior Back-End Lead*. Di bawah kepemimpinan Bapak Rusmin Subiakto, terdapat enam *Back-End Lead*, salah satunya adalah Bapak Christian William selaku *supervisor* selama

praktik pelaksanaan kerja magang. Bapak Christian William memimpin tiga orang *Back-End Engineer* dan satu *Back-End Intern*.

Selama praktik magang, penugasan diberikan dan dipandu oleh Bapak Christian William sebagai *supervisor* dan *Back-End Lead* dari tim. Daftar tugas atau *backlog* diatur dalam satu *platform* terdedikasi yaitu Jira, dan distribusi tanggung jawab diatur langsung oleh *lead* pada masing-masing tim. Penugasan selalu disertai dengan tautan tiket Jira bersamaan dengan sesi *onboarding* oleh Bapak Christian William terkait obyektif yang ingin dicapai pada penugasan tersebut. Pada beberapa kondisi, pelaksanaan *onboarding* diserahkan pada PIC (*Person In Charge*) terkait dengan *repository* atau *codebase* yang akan menjadi bagian penugasan, ataupun kepada *developer* yang pernah mengerjakan hal atau fitur serupa.

Dokumentasi terkait *codebase* ataupun *progress* penugasan dilakukan melalui platform *Notion* yang terdedikasi untuk masing-masing tim. Setiap harinya, Bapak Christian William selaku *lead* akan menyelenggarakan *daily meeting* bersama tim untuk mengumpulkan *progress* terkait hal yang dikerjakan pada hari tersebut untuk keperluan dokumentasi. Selain itu, setiap anggota juga wajib memperbaharui *progress* pekerjaan untuk masing-masing tiket pada satu *interface* yang terdedikasi untuk pemantauan *progress* tiket atau penugasan. Setiap penuntasan tiket atau penugasan, akan dilaksanakan *code review* oleh *lead* ataupun pihak terkait untuk membahas hasil pekerjaan, baik itu dari segi *improvement*, aspek *clean code*, ataupun kesalahan *logic* jika ada.

3.2 Tugas yang Dilakukan

Selama enam bulan praktik kerja magang, adapun tugas-tugas yang dilaksanakan sebagai *Back-End Intern* adalah sebagai berikut.

1. Mempelajari Alur dari Proyek Improvisasi Fitur Review Rating.
2. Pengembangan Projek
3. Integrasi dengan *Front-End*
4. Mendukung Proses Pengujian oleh Tim *Quality Assurance*
5. Mendukung Proses *Deployment*
6. Perbaikan *Bug* Pasca *Deployment*

3.3 Uraian Pelaksanaan Magang

Berikut adalah detail terkait kegiatan-kegiatan yang dilakukan selama praktik kerja magang.

Tabel 3.1. Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
1	• *Kick-off Meeting* Proyek • *Onboarding* penugasan oleh *supervisor* • Pengerjaan *debugging*, dan *self-test* tiket *bulk upsert* ulasan produk • Pengerjaan dan *self-testing* tiket *update status* ulasan produk.
2	• Pengerjaan dan *self-testing* tiket penyesuaian *flow* ulasan di halaman transaksi *offline* • Pengerjaan dan *self-testing* tiket penyesuaian *flow* ulasan di halaman transaksi *online* • *Alignment* dengan *Front-end Developer* • Pengerjaan API *Contract*
3	• *Onboarding requirement* tambahan proyek • Pengembangan dan *self-test endpoint* baru • Pengerjaan *unit test* untuk tiket *bulk upsert* ulasan produk • Pengerjaan *unit test* untuk tiket *update status* ulasan produk • Perbaikan *bug* saat integrasi dengan *front-end*
	Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
4	• Pengerjaan *unit test* untuk tiket endpoint baru • Perbaikan *bug* saat integrasi dengan *front-end* • Pengerjaan dan *self-test* tiket pembatasan jumlah percobaan ulasan produk • Pengerjaan dan *self-test* tiket penyesuaian perhitungan rata-rata nilai ulasan • Pengerjaan *feedback* dari tim *Quality Assurance* tiket eksternal untuk *handling fee* dan *reorder* wilayah Batam • Pengerjaan *feedback* dari tim *Quality Assurance* untuk proyek *Review Rating*
5	• Sesi *code review* oleh *supervisor* untuk proyek Review Rating • Pengerjaan *feedback* dari *code review* • Pengerjaan dan *unit-test* untuk modularisasi logika status ulasan transaksi *online* dan *offline* • Pelaksanaan *System Integration Testing* (SIT) • Pelaksanaan *User Acceptance Testing* (UAT)
6	• Pengujian ulang secara *end-to-end* pada *staging environment* • Penulisan *query* penyesuaian *flow* • Pemeriksaan menyeluruh (*regression*) terkait *impact deployment* proyek • Pengerjaan penyesuaian *regression testing* • *Deployment* dan *monitoring*

Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
7	• *Code review* penugasan *Shipper Phase 2* • *Monitoring* pasca *deployment* • Pengerjaan perbaikan *bug* pasca *deployment* • *Onboarding* pengerjaan tiket eksternal terkait integrasi *vue.ai* • Pengerjaan dan *self-test* tiket integrasi *vue.ai*
8	• API *testing* dan *end-to-end testing* hasil pengembangan penugasan tambahan *Shipper Phase 2* • *Monitoring* pasca *deployment* • Pengerjaan tiket eksternal terkait *handling fee adjustment* • Pengerjaan perbaikan *bug* pasca *deployment* terkait barang *refund* • Sesi *code review* oleh *supervisor* untuk tiket integrasi *vue.ai* • Pengerjaan dan *self-test Sqitch query* untuk migrasi database untuk proyek *Review Rating*
9	• API *testing* dan *end-to-end testing* pada hasil *bugfix* fitur *voucher* pada fitur *reorder*. • Pembuatan laporan hasil *bugfix* fitur *voucher* pada fitur *reorder* • *Onboarding* penugasan lanjutan dari penugasan *bugfix* (ditemukan isu atau *bug* terkait hasil *refund* dari *reorder* dimana *free item* tidak terhitung *refunded* untuk kasus *reorder manual* • *Debugging*, *tracing* pada penyebab isu *partial refund* pada *free item* pada fitur *reorder manual*
	Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
10	• Pengerjaan perbaikan *bug* terkait kurir *ownfleet* pada fitur *reorder* • Pengerjaan pengumpulan, pemeriksaan, dan improvisasi *query* terkait ulasan produk dan toko • Mendukung tim *Quality Assurance* dengan memperbaiki *bug* pada beberapa *staging environment*
11	• Pengerjaan pemeriksaan dan improvisasi *query* terkait ulasan produk dan toko • Pengujian ulang, dan perbaikan *feedback* migrasi *Sqitch* untuk proyek *Review Rating* • Pengerjaan *mock data injection* untuk tiket terkait fitur ulasan
12	• Pengerjaan *mock data injection* untuk tiket terkait fitur ulasan • Pemeriksaan *query* terkait fitur ulasan pada *database* berskala *production* • Mendukung tim *Quality Assurance* untuk tiket perbaikan *bug* barang *refund* untuk fitur ulasan
13	• Pengerjaan tiket integrasi *vue.ai* pada proses pembuatan pesanan pelanggan • Pengerjaan *unit test* pada *codebase* fitur ulasan
14	• Pengerjaan tiket perhitungan *handling fee* untuk fitur *reorder* • Pengerjaan perbaikan isu *SQL Injection* pada *legacy service* • Pengerjaan *unit test* pada *codebase* fitur ulasan
	Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
15	• *Code review* dan perbaikan pengerjaan tiket *handling fee* dan integrasi *vue.ai* • Pengerjaan perbaikan isu *SQL Injection* pada *legacy service* • Pengerjaan *unit test* pada *codebase* fitur ulasan
16	• Pengerjaan tiket migrasi Tiktok Shop pada fitur ulasan dan pemesanan ulang • Pengerjaan dokumentasi pada *codebase* fitur ulasan
17	• Pengerjaan tiket implementasi *voucher* ulasan untuk transaksi *offline* • Pengerjaan dokumentasi pada *codebase* fitur ulasan
18	• Pengerjaan tiket perbaikan *query* pada *batch* otomasi pembuatan *invoice* pesanan TP • Pengerjaan tiket improvisasi *query* pada *batch voucher* ulasan. • Pengerjaan dokumentasi pada *codebase* fitur ulasan
19	• *Support* perbaikan isu production terkait validasi input dari *front-end* pada fitur *ulasan* • Pengerjaan dokumentasi pada *codebase* fitur ulasan
20	• Pengerjaan pengumpulan dan analisis *query cost* pada seluruh bagian *codebase* pada fitur ulasan • Pengerjaan dokumentasi pada *codebase* fitur ulasan
21	• Pengerjaan pengumpulan dan analisis *query cost* pada seluruh bagian *codebase* pada fitur ulasan • Pengerjaan dokumentasi pada *codebase* fitur ulasan
22	• Riset terkait isu *post-production* terkait migrasi *charset* pada tabel terkait fitur ulasan • Pengerjaan pengumpulan dan analisis *query cost* pada seluruh bagian *codebase* pada fitur ulasan

Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
23	• *Code Review* dan perbaikan terkait seluruh tiket atau penugasan • Riset, perbaikan, dan pengujian terkait isu *post-production* terkait migrasi *charset* pada tabel terkait fitur ulasan
24	• *Code Review* dan perbaikan terkait seluruh tiket atau penugasan • Riset dan pengerjaan perbaikan terkait isu data tidak sinkron akibat *replication lag* pada fitur ulasan.

3.3.1 Mempelajari Alur dari Proyek Improvisasi Fitur Review Rating

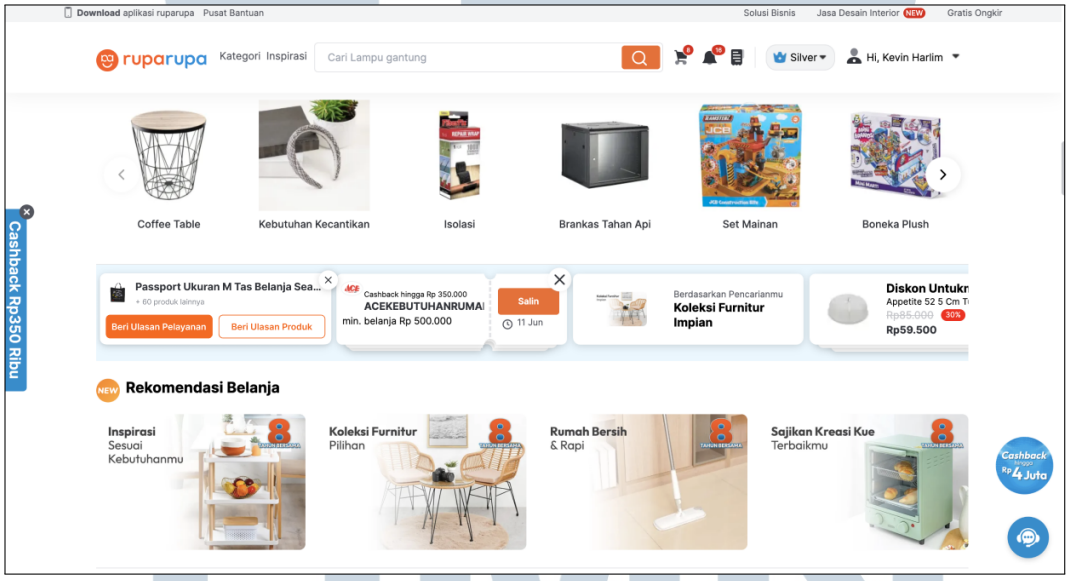
Fitur *Review Rating* atau fitur ulasan pada aplikasi *ruparupa* merupakan fitur yang dirancang untuk memungkinkan pelanggan yang melakukan pembelian produk untuk memberikan ulasan, baik itu pembelian secara *online* melalui aplikasi atau *web* maupun pembelian secara *offline* (langsung mengunjungi toko). Berdasarkan *Product Requirement Design* (PRD), Proyek Improvisasi *Review Rating*, terdapat dua tipe ulasan, yaitu ulasan terhadap produk yang dibeli dan ulasan terhadap pelayanan toko. Sebelumnya, fitur ulasan dan produk masih tergabung dalam satu *form*. Pemisahan ini ditujukan untuk membantu toko dalam memantau penilaian *Key Performance Index* (KPI) toko. Perubahan yang terdapat untuk ulasan produk adalah ulasan secara *bulk*, dan tidak ada perubahan dari segi *input*. Untuk ulasan toko, bentuk *feedback* yang dapat disampaikan adalah *rating* dengan bintang dari 1 sampai 5, dan penambahan *text box* (opsional) untuk penilaian dibawah 4 bintang agar bisa mengetahui alasan dan menjadi dasar untuk perbaikan pelayanan toko.

Improvisasi lain yang tercakup dalam proyek ini adalah penambahan *entry point* untuk ulasan, diantaranya *home page*, halaman daftar dan detail transaksi *online* maupun *offline*, dan halaman daftar dan detail ulasan. Selain itu, pengguna bisa memperbarui ulasan produk apabila memberikan ulasan pada salah satu produk dengan bintang dibawah empat dengan batas maksimal dua kali perbaikan. Apabila pelanggan belum memberikan ulasan produk, maka akan muncul tombol ”Beri Ulasan Produk”, begitu pun untuk ulasan toko akan muncul tombol ”Beri Ulasan Pelayanan” Apabila ulasan produk masih dapat diperbarui, maka akan muncul

tombol ”Perbarui Ulasan Produk”. Apabila pelanggan sudah memberikan ulasan diatas tiga bintang ataupun melewati jumlah batas maksimal perbaikan ulasan, maka akan muncul tombol ”Lihat Ulasan Produk”, begitupun ketika pelanggan sudah memberikan ulasan toko maka akan muncul tombol ”Lihat Ulasan Pelayanan” (ulasan toko tidak dapat diperbarui).

Dari segi operasional, proyek ini memungkinkan tim operasional untuk menyembunyikan ulasan produk pada PDP (*Product Detail Page*) yang sudah dibalas oleh *Customer Service*.

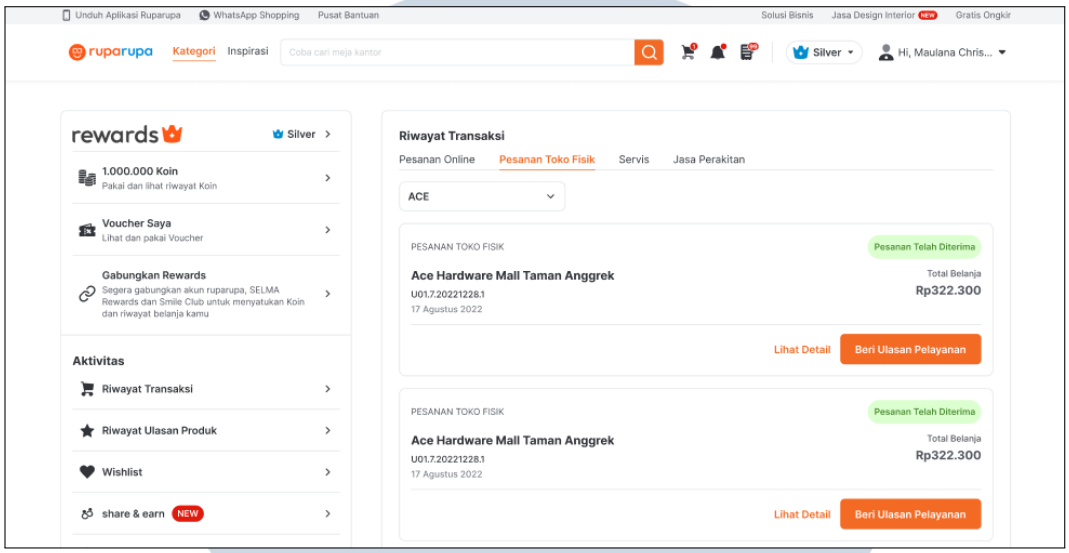
A. Ulasan Melalui *Home Page*



Gambar 3.2. *Entry point* ulasan melalui *home page*

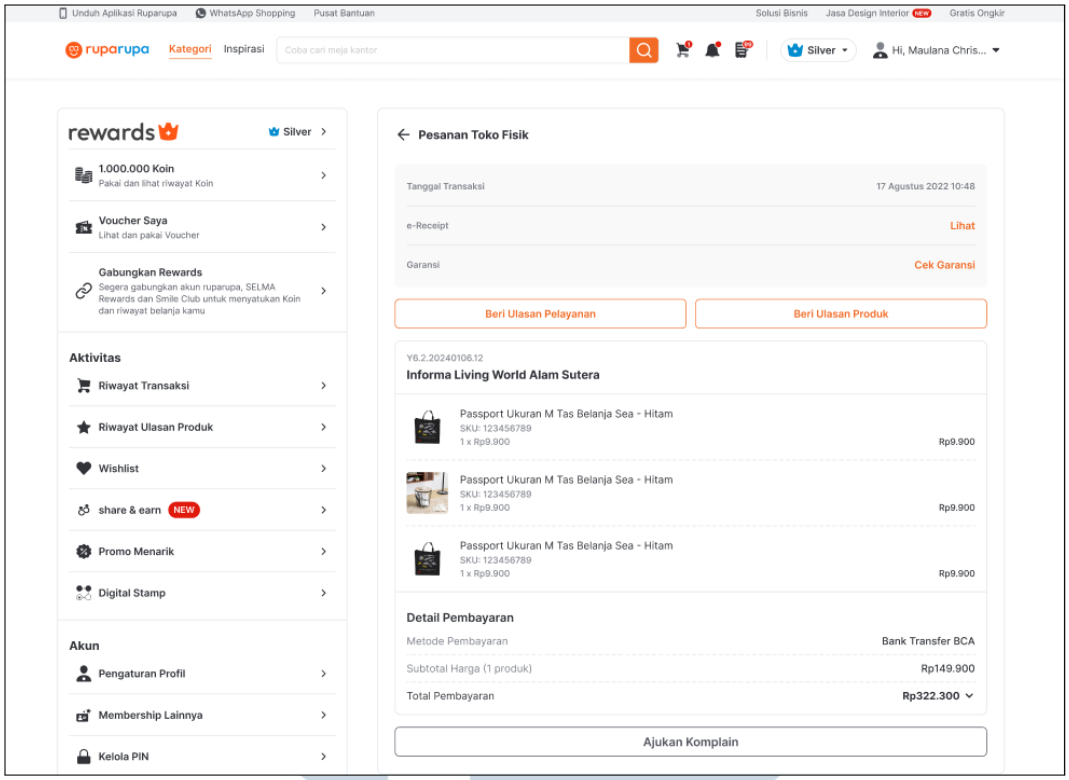
Gambar 3.2 menunjukkan tampilan *home page* yang menyertakan *mini-form* bagi pelanggan untuk memberikan ulasan. Pada halaman ini, hanya ditampilkan satu *invoice* yang diurutkan berdasarkan tanggal transaksi terbaru.

B. Ulasan Melalui Halaman Daftar dan Detail Transaksi



Gambar 3.3. *Entry point* ulasan melalui halaman daftar transaksi

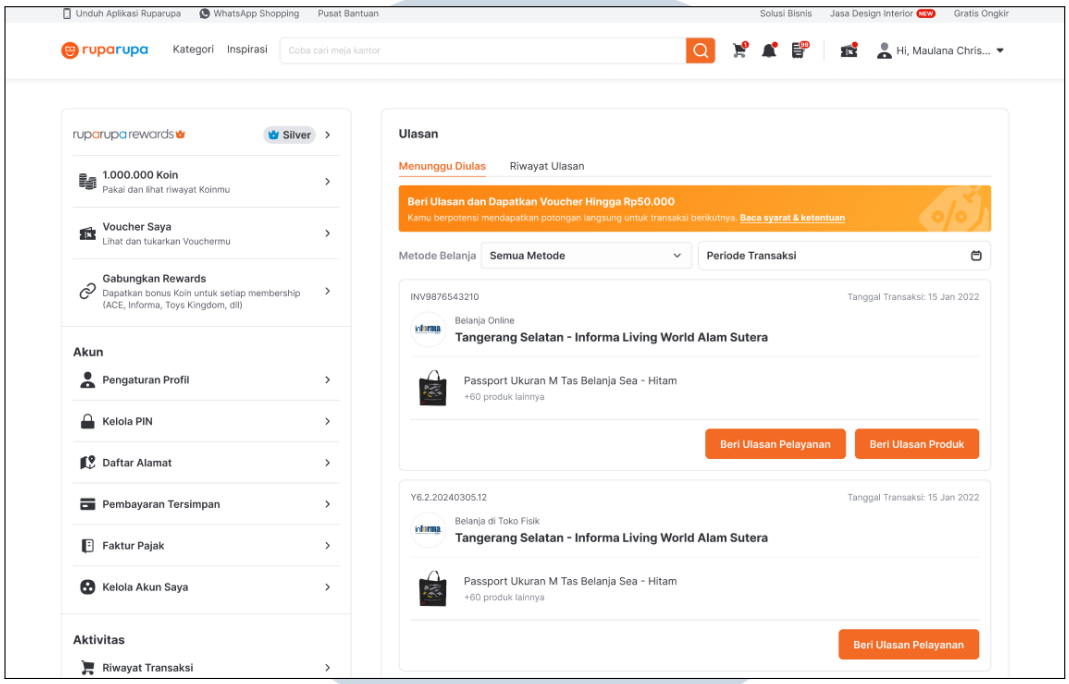
Gambar 3.3 menunjukkan tampilan halaman daftar transaksi yang menampilkan seluruh transaksi pelanggan, baik itu transaksi *offline*, *online*, sudah diulas, ataupun yang belum diulas. Pada dasarnya, tujuan utama halaman ini bukanlah untuk menampilkan data riwayat ulasan melainkan untuk menunjukkan riwayat transaksi, namun dilakukan improvisasi dengan menambahkan *entry point* ulasan pada halaman ini.



Gambar 3.4. *Entry point* ulasan melalui halaman detail transaksi

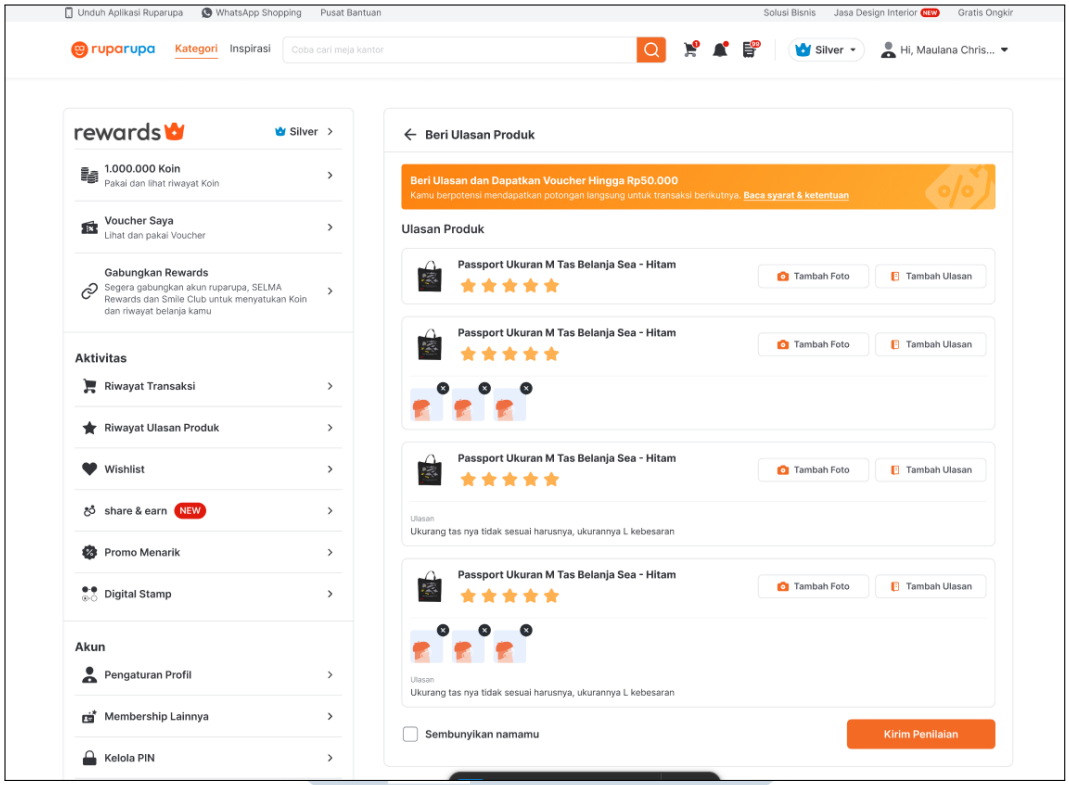
Gambar 3.4 menunjukkan tampilan halaman detail transaksi yang menunjukkan detail transaksi dari pelanggan.

C. Ulasan Melalui Halaman Daftar dan Detail Ulasan

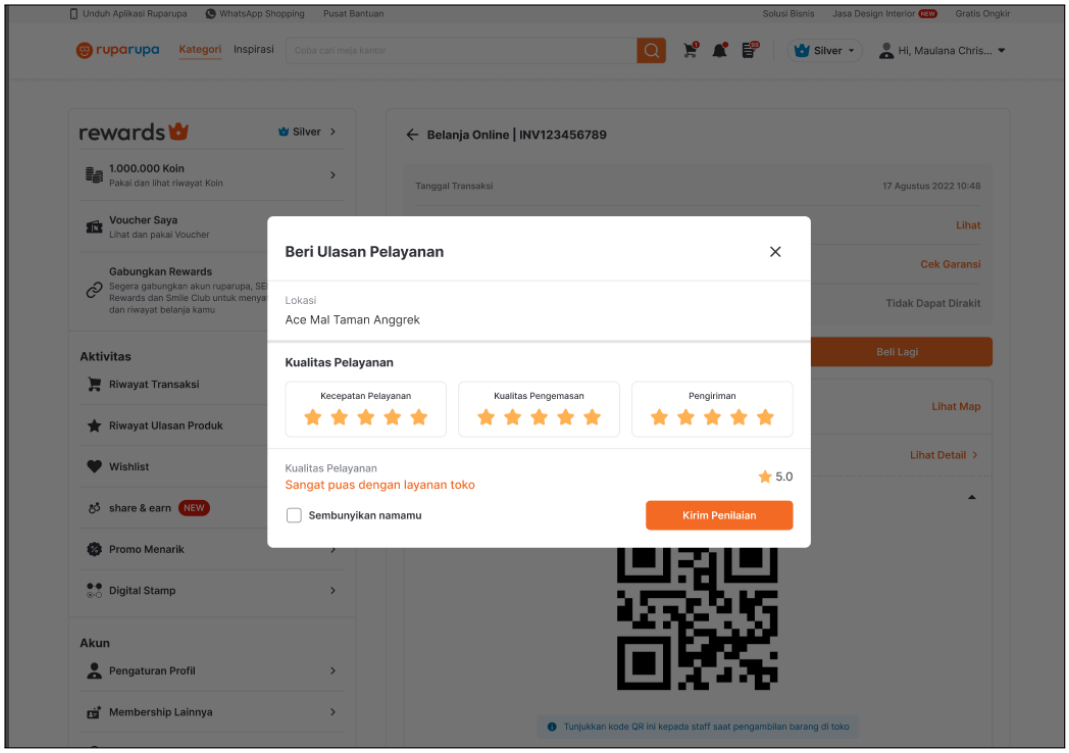


Gambar 3.5. *Entry point* ulasan melalui halaman ulasan

Gambar 3.5 menunjukkan tampilan halaman daftar ulasan pada *tab* ”Menunggu Ulasan”. *Tab* ini ditujukan untuk *invoice* yang belum diulas sama sekali (produk dan pelayanan toko), ataupun salah satu diantara kedua tipe ulasan tersebut belum diberikan, ataupun ketika keduanya sudah diulas namun ulasan produk masih dapat diperbarui.

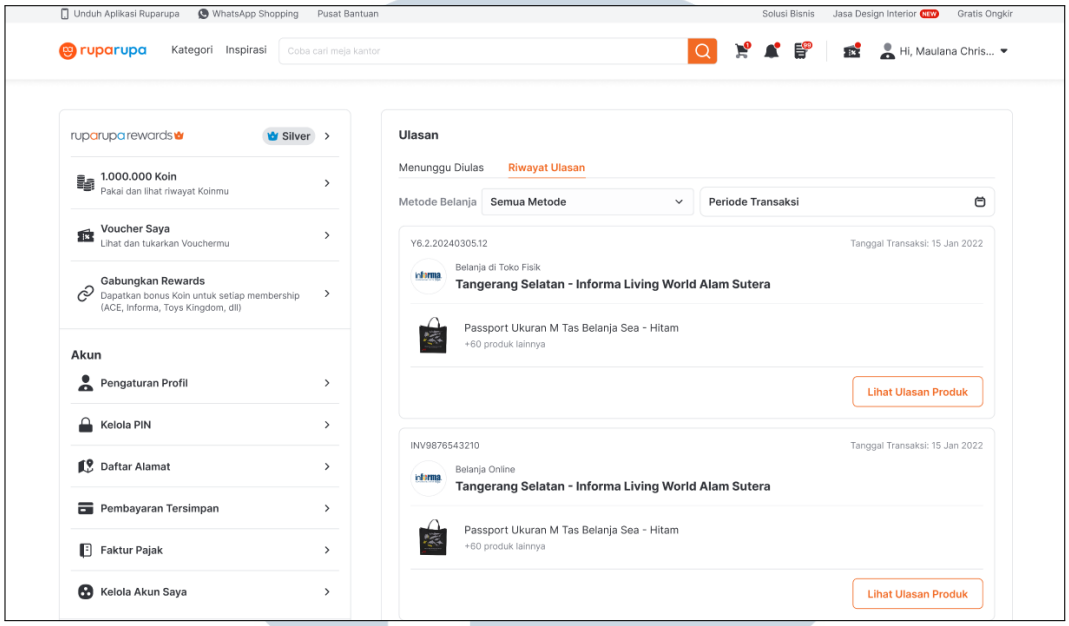


Gambar 3.6. *Popup* ulasan produk



Gambar 3.7. *Popup* ulasan pelayanan toko

Gambar 3.6 menunjukkan tampilan halaman detail ulasan produk, dan Gambar 3.7 menunjukkan tampilan halaman detail ulasan pelayanan.



Gambar 3.8. Tampilan halaman riwayat ulasan

Gambar 3.8 menunjukkan tampilan halaman riwayat ulasan pada *tab* ”Riwayat Ulasan”. *Tab* ini ditujukan untuk *invoice* yang pelayanan toko dan seluruh produknya sudah selesai diulas.

3.3.2 Pengembangan Proyek

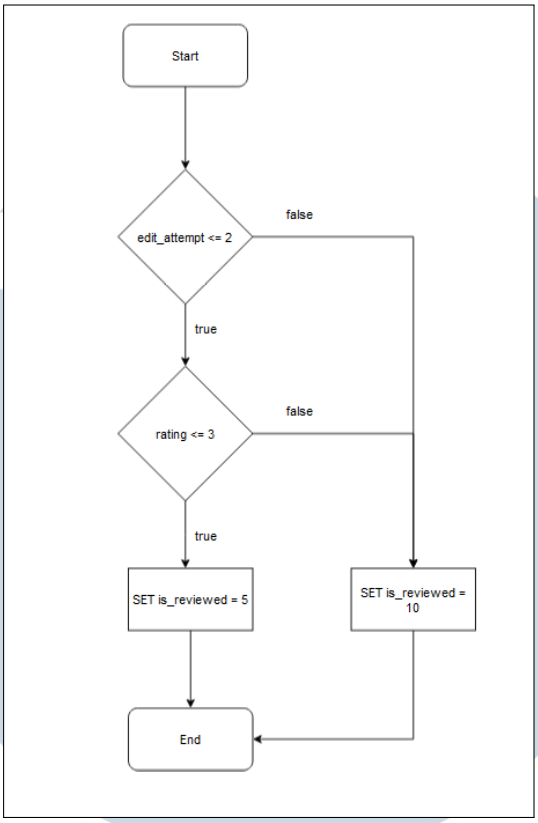
Pengembangan proyek terbagi menjadi beberapa *sub-task* yang harus dikerjakan untuk memenuhi seluruh *requirement* dari proyek.

A. Penyesuaian pada Halaman Transaksi

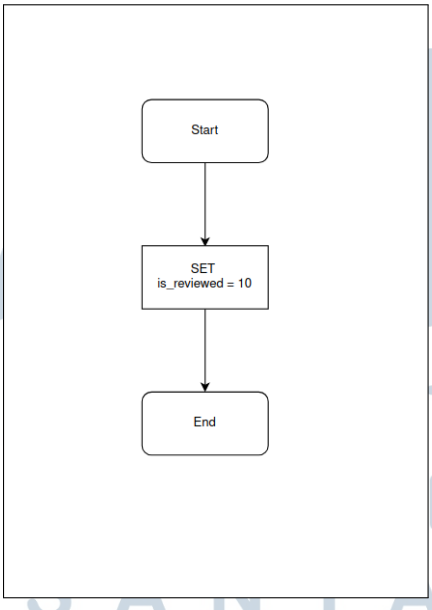
Penyesuaian yang dilakukan adalah menuliskan logika untuk mengolah dan menampilkan data status ulasan produk dan ulasan toko. Pada struktur basis data yang akan digunakan dan diimplementasikan untuk proyek ini, digunakan angka 0 untuk merepresentasikan bahwa suatu produk belum diulas, angka 5 menunjukkan ulasan produk dapat diubah, dan angka 10 menunjukkan ulasan produk sudah tidak dapat diubah. Untuk ulasan toko, hanya dapat dilakukan satu kali tanpa revisi. Pemetaan ketiga status ulasan produk ini didasarkan atas beberapa kriteria. Apabila pelanggan memberikan ulasan bintang di atas 3, maka ulasan tersebut

akan langsung dianggap final dan tidak dapat diubah kembali. Apabila pelanggan memberikan penilaian sama dengan atau di bawah 3 bintang, maka pelanggan diberikan kesempatan untuk melakukan revisi sebanyak dua kali. Transisi dari status 5 menuju 10 hanya akan terjadi apabila pelanggan mengubah jumlah bintang ulasan menjadi di atas 3 atau karena telah melewati batas jumlah revisi.

Pada lingkup ini, objek yang ditampilkan pada sisi pelanggan adalah status seluruh ulasan produk untuk setiap *invoice*. Maka dari itu, disusun sebuah pemetaan yang akan digunakan untuk menampilkan *summary*. Apabila salah satu status ulasan produk pada *invoice* adalah 5, maka *response* yang ditampilkan adalah status 5 yang merepresentasikan tombol ”Perbarui Ulasan Produk” pada sisi pelanggan. Apabila terdapat status ulasan 0 pada salah satu produk, maka *response* yang ditampilkan adalah status 0 yang merepresentasikan tombol ”Beri Ulasan Produk”. Apabila seluruh status ulasan produk adalah 10, maka *response* yang ditampilkan adalah 10 yang merepresentasikan bahwa ulasan *invoice* tersebut sudah memiliki status ”selesai”, dan yang ditampilkan adalah teks ”Lihat Ulasan Produk”. Untuk seluruh transaksi *online* maupun *offline*, digunakan satu modul yang sama untuk implementasi fitur ini. Gambar 3.9 dan Gambar 3.10 menunjukkan alur pemetaan status untuk ulasan produk dan pelayanan.



Gambar 3.9. Alur status ulasan produk



Gambar 3.10. Alur status ulasan pelayanan

Implementasi ini dilakukan dengan menambahkan kolom baru pada basis data *MySQL* untuk menyimpan jumlah revisi pada setiap produk, melakukan

modifikasi *query* dan logika validasi, serta menuliskan modul yang bersifat *reusable* pada seluruh *endpoint* terkait (*endpoint list* dan detail transaksi *online* maupun *offline*) untuk pemetaan status untuk *response* pada sisi pelanggan.

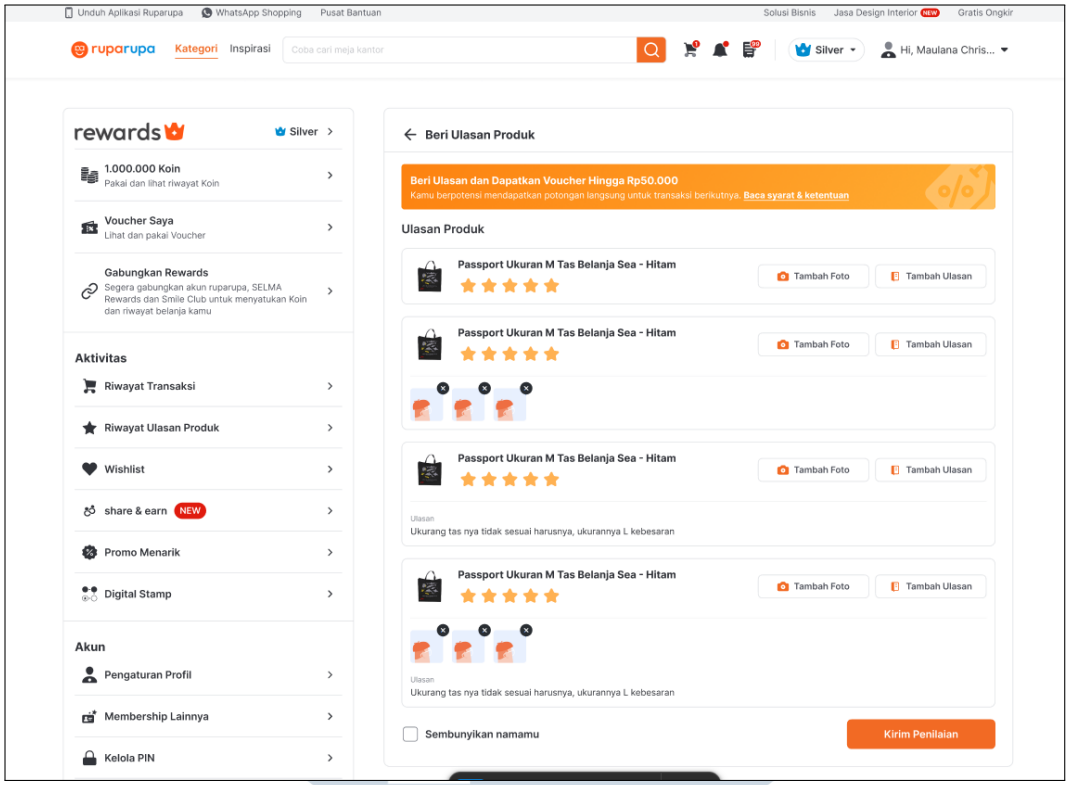
B. Pembuatan API *Endpoint* untuk Halaman Ulasan Pelanggan

Sebelum proyek *review rating* diusungkan, halaman ulasan pelanggan sudah memiliki *endpoint* khusus untuk memenuhi kebutuhan penyediaan penampilan data ulasan pelanggan. Namun, untuk mempertahankan prinsip *backward compatibility* atas alasan-alasan tertentu, *supervisor* mengusungkan pembuatan API *endpoint* baru untuk penyesuaian pada proyek ini.

Secara garis besar dari segi *response*, objektif dari *endpoint* ini adalah sama dengan *endpoint* lama, yaitu untuk menampilkan data riwayat ulasan maupun *invoice* yang belum diulas yang dapat diaplikasikan dengan berbagai *filter*, seperti *limit*, *offset*, tanggal transaksi, nomor transaksi, tipe transaksi, dan *SKU*. Modifikasi yang paling signifikan dari pembuatan *endpoint* ini adalah logika untuk menyesuaikan *output* status ulasan produk dan toko yang sama seperti logika pada halaman transaksi yang dilakukan langsung melalui *query* baru. Seluruh *entry point* untuk ulasan produk dan toko juga bergantung pada *endpoint* ini.

C. Implementasi *Bulk Insert* Ulasan Produk

Sebelum proyek ini diajukan, pelanggan hanya disediakan satu form untuk memberikan ulasan untuk setiap satu produk. Mekanisme ini membuat pelanggan harus melakukan lebih banyak *effort* atau aksi untuk melakukan ulasan terhadap semua produk dalam satu *invoice*. Oleh karena itu, proyek ini mengusungkan *form* yang memungkinkan pelanggan untuk dapat langsung memberikan ulasan pada seluruh produk dalam satu *invoice* yang sama, sehingga memudahkan dan menghemat waktu pelanggan. Gambar 3.11 menunjukkan contoh halaman yang berisikan *form* ulasan produk yang baru.



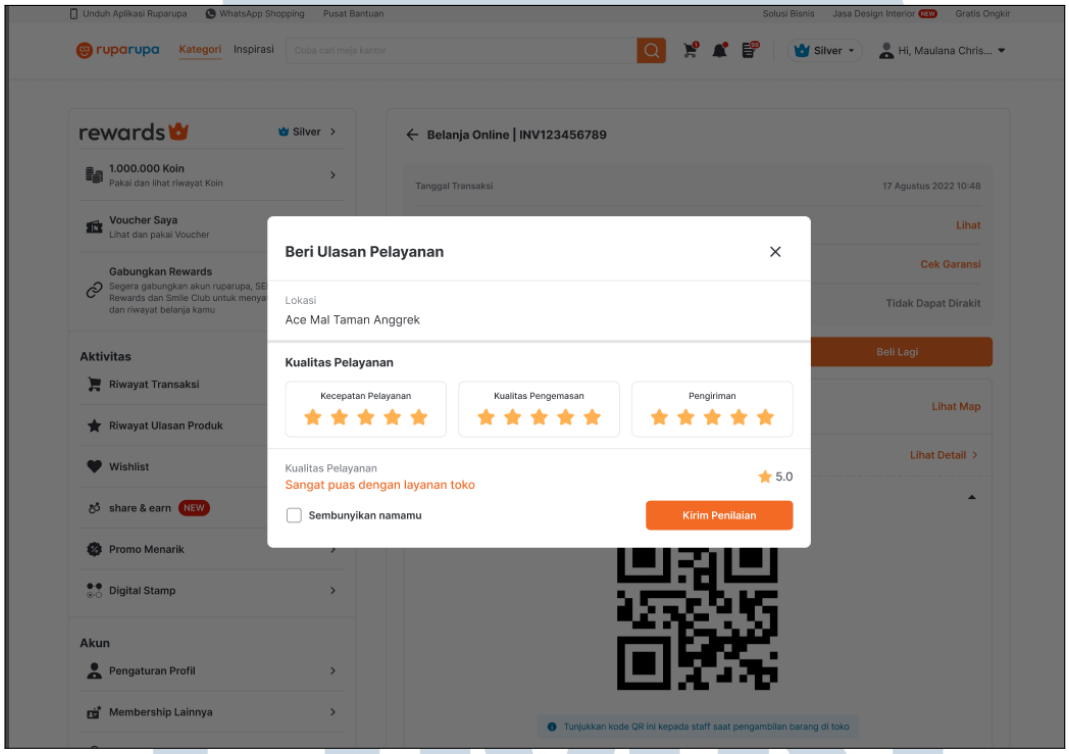
Gambar 3.11. *Form* ulasan produk

Implementasi untuk fitur ini menggunakan modul atau fungsionalitas yang sama dengan modul penyimpanan *single product* (*existing*), dengan menggunakan *array* sebagai pembeda untuk memungkinkan *request* ulasan data lebih dari satu produk untuk dapat disimpan pada *database*. Implementasi teknikal dilakukan dengan memanfaatkan *goroutine* pada bahasa pemrograman Golang untuk menerapkan prinsip *concurrency*, sehingga *request* ulasan yang masuk dapat diproses secara bersamaan dan dapat meningkatkan *performance* untuk data dengan skala yang besar.

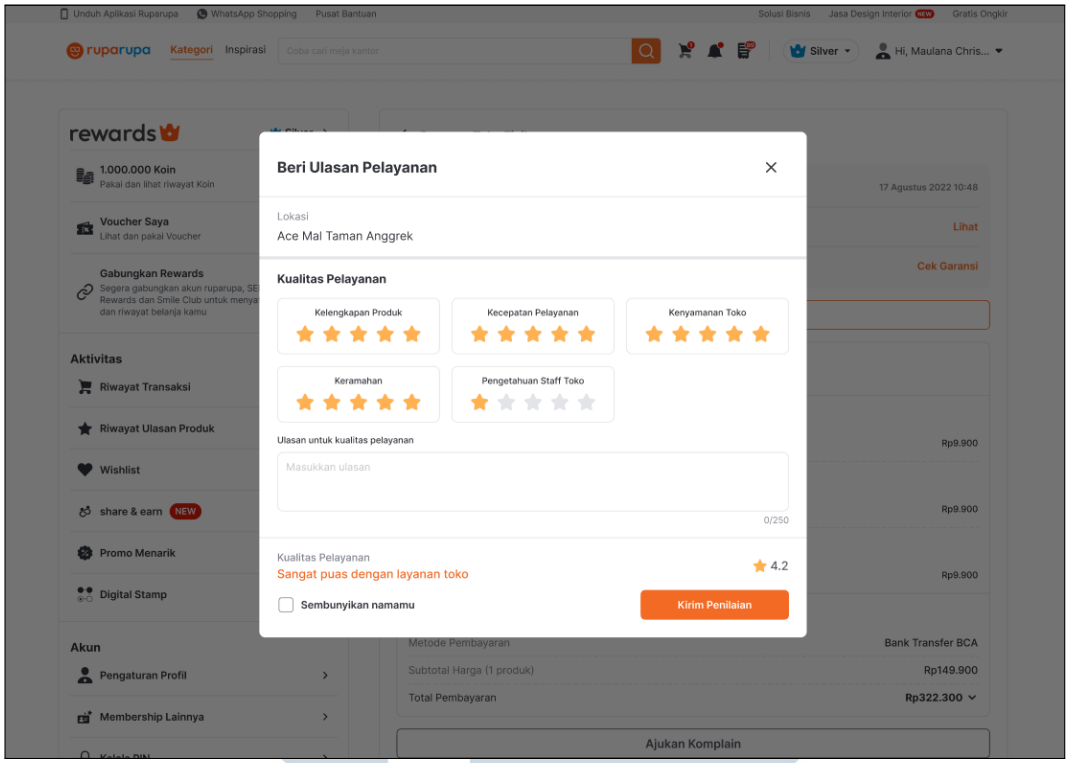
Untuk meminimalisir penggunaan *memory* ketika terdapat skenario dengan jumlah produk yang besar dalam satu *invoice*, maka diterapkan prinsip *concurrency* dengan *queue*, yaitu dengan inisialisasi sejumlah jalur antrian atau *worker* yang telah ditetapkan pada *environment variable*. Seluruh *request* ulasan produk akan disimpan pada satu variabel bertipe *channel*, lalu sejumlah *worker* diinisialisasikan secara asinkron dengan membawa *reference* pada *channel tersebut*. Setiap *worker* akan mengambil satu objek *request* dari *channel* tersebut untuk diproses, dan akan kembali mengambil objek *request* yang belum terproses secara bergilir hingga *channel* objek tersebut sudah tidak memiliki data untuk diproses kembali.

D. Penyesuaian Perhitungan Average Rating pada Ulasan Pelayanan

Perubahan yang terdapat pada proyek ini adalah menambahkan satu kriteria baru untuk ulasan pelayanan untuk transaksi *online* yaitu dari segi kecepatan pelayanan (*service speed*). Sebelumnya, kriteria penilaian hanya meliputi penilaian pengiriman (*shipment*) dan pengemasan (*packing*). Gambar 3.13 dan Gambar 3.12 menunjukkan contoh tampilan hasil penyesuaian.



Gambar 3.12. Tampilan *modal* ulasan toko untuk transaksi *online*



Gambar 3.13. Tampilan *modal* ulasan toko untuk transaksi *offline*

Untuk transaksi *offline*, kriteria yang digunakan adalah dari segi kecepatan pelayanan (*service speed*), lingkungan toko (*store environment*), pengetahuan *staff* (*staff knowledge*), kelengkapan produk (*product completeness*), dan keramahan (*hospitality*). Tidak ada perubahan kriteria untuk transaksi *offline*, namun *endpoint* yang digunakan untuk perhitungan sebelumnya digunakan secara bersamaan untuk kedua jenis transaksi, dan skenario ini belum ditangani, sehingga akan menyebabkan kesalahan perhitungan rata-rata karena sebelumnya untuk transaksi *online* hanya mempertimbangkan *rating* pengiriman dan pengemasan. Oleh karena itu, dilakukan perubahan untuk menyesuaikan dengan kebutuhan proyek.

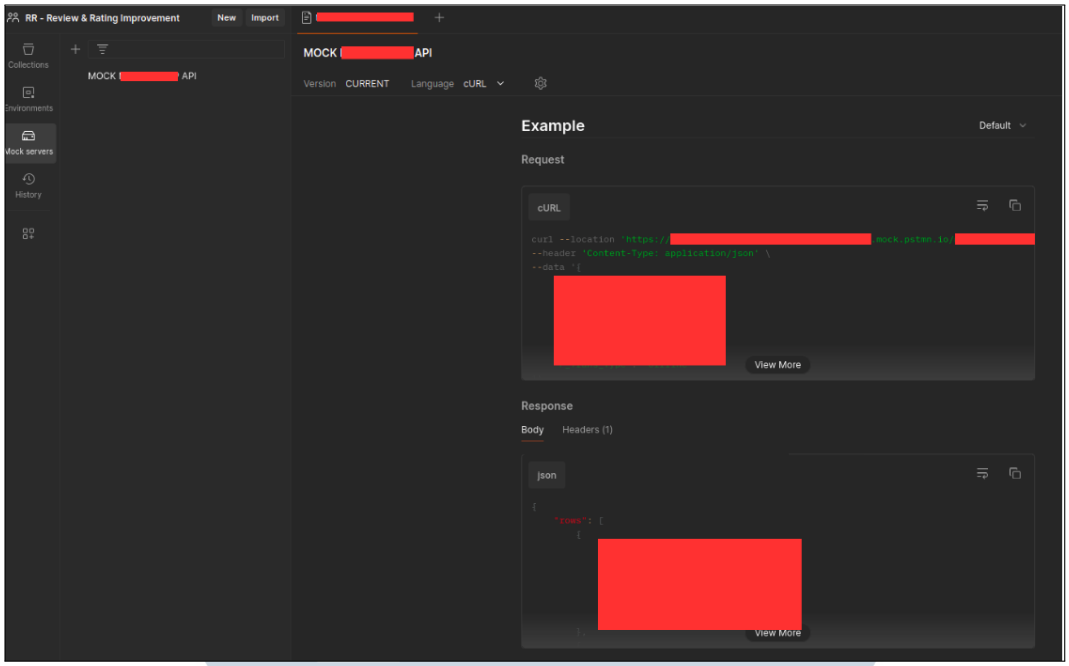
Impact lainnya dari penambahan *requirement* ini adalah pada struktur data pada basis data dan seluruh *endpoint* lainnya yang melibatkan proses perhitungan *average* ulasan, termasuk pada modul *analytics* maupun *endpoint* lainnya yang bersifat *customer-facing*. Oleh karena itu, dilakukan penyesuaian terkait logika pada *endpoint-endpoint* yang terkait beserta penulisan *query* DML (*Data Manipulation Language*) dan DDL (*Data Definition Language*) untuk menyesuaikan dengan *requirement* proyek.

3.3.3 Integrasi dengan Front-End

Seiring berjalannya proses pengembangan dari *back-end*, proses integrasi dengan tim *front-end* tetap berjalan melalui penulisan *API Contract*. *API Contract* berisi seluruh *endpoint* terkait proyek yang melewati proses modifikasi atau penyesuaian *response*, *payload* ataupun *parameter*, serta dokumentasi terkait cara mengakses API *endpoint* yang baru dikembangkan. Penulisan *API Contract* dilaksanakan melalui *Postman collection* yang dapat dibagikan. Melalui kontrak tersebut, proses integrasi oleh tim *front-end* dapat dilakukan dengan mudah, termasuk ketika terdapat penyesuaian yang bersifat mendadak dari segi *back-end*, sehingga perubahan tersebut dapat di-*notice* dan diakses secara *real-time*.

Data transaksi *offline* yang disimpan pada basis data milik Ruparupa tidak bersifat lengkap, karena proses penyimpanan utama data transaksi *offline* dilakukan pada API milik *head office* untuk menampung data transaksi dari berbagai unit bisnis. Untuk mengakses data tersebut, pihak *head office* menyediakan API untuk diakses oleh tim Ruparupa, namun mereka tidak menyediakan API untuk *environment* pengembangan atau *staging*, sehingga tim *developer* maupun *Quality Assurance* (QA) mengalami kesulitan untuk melakukan pengujian yang terkait dengan skenario transaksi *offline*).

Oleh karena itu, sebagai *back-end developer* dilakukan proses *mocking* data untuk melakukan proses *supplying* data, sehingga proses pengembangan dan pengujian dapat dilakukan dengan mudah. *Mocking* dilakukan dengan memanfaatkan *Postman Mock Server*, yaitu sebuah fitur Postman yang memungkinkan untuk membuat sebuah *server* palsu yang dapat menerima *request* dengan *pre-defined response* yang dapat dibuat oleh *developer*. Setelah proses pembuatan *mock server* dan *mock endpoint* beserta integrasi injeksi data dengan basis data Ruparupa selesai dilaksanakan, maka *base URL* yang digunakan oleh *staging environment* Ruparupa diubah dengan URL dari *mock server* tersebut, sehingga dapat diperoleh *response* yang sesuai. Gambar 3.14 menunjukkan contoh implementasi *mock server* untuk digunakan sebagai sumber data pengujian.



Gambar 3.14. Tampilan *modal* ulasan toko untuk transaksi *offline*

3.3.4 *Unit Testing*

Untuk meningkatkan kualitas dan *reliability* kode, maka penulisan *unit-testing* dilaksanakan pada seluruh *sub-task*, yang meliputi penulisan *endpoint* baru, penyesuaian perhitungan rata-rata ulasan, serta modul pemetaan status ulasan. Implementasi *unit-testing* memanfaatkan konsep *mocking*, yaitu mekanisme penulisan *pre-defined response* pada komponen eksternal yang dibutuhkan dari *service*, seperti koneksi *database*, layanan notifikasi ataupun pada *cloud service*. Hal ini dilaksanakan untuk menyesuaikan dengan prinsip *unit test* yang berfokus pada pengujian reliabilitas pada kode, untuk memastikan bahwa kode dapat menangani berbagai jenis skenario *input* ataupun proses, seperti proses bisnis ataupun skenario *error* yang dapat menyebabkan *service* bermasalah.

```
mockService.EXPECT().
    GetUser(ctx, userID).
    Return("", errors.New("user not found"))
```

Gambar 3.15. Contoh *mocking*

Gambar 3.15 menunjukkan contoh teknik *mocking*. Proses *mocking* dilakukan pada level *function* yang memiliki *external dependency*, dengan

menuliskan ekspektasi argumen ketika *test* dijalankan. Apabila fungsi tersebut tidak dipanggil sesuai dengan argumen yang diminta, maka *test* tersebut akan gagal. Fungsi *mock* akan memiliki *pre-defined response* yang akan dikembalikan setelah dipanggil oleh argumen untuk digunakan pada logika bisnis selanjutnya. Proses *mocking* dapat diimplementasikan dengan memanfaatkan *polymorphism* dan *abstraction* pada setiap komponen atau *dependency*, sehingga dapat terintegrasi baik untuk *integration* ataupun *unit testing*.

3.3.5 Mendukung Proses Pengujian oleh Tim Quality Assurance

Setelah proses pengembangan dan integrasi selesai dilaksanakan, dilakukan pengujian oleh tim QA sebelum dilaksanakan *System Integration Testing* (SIT) dan *User Acceptance Testing* (UAT). Proses *testing* dilaksanakan dengan menyusun dan mengumpulkan berbagai *test case*, lalu dilaksanakan pengujian secara *end-to-end* pada aplikasi, baik itu *web* maupun aplikasi *mobile*. Proses pengujian ini didukung oleh seluruh tim *developer* ketika terjadi *bug* atau kesalahan pada fitur atau *requirement* yang dikembangkan ataupun karena *issue* pada *environment staging*, ataupun ketika tim QA membutuhkan lebih banyak sampel data transaksi *offline*.

3.3.6 Mendukung Proses Deployment

Proses *deployment* dilakukan oleh tim infrastruktur dan *lead* dari tim *developer*. Sebelum proses ini, dirancang sebuah dokumen "*rundown deployment*" yang dibutuhkan oleh tim infrastruktur. Dokumen ini berisikan daftar *service* atau *repository* yang mengalami perubahan oleh akibat proyek yang disertakan dengan tautan *pull request* terkait, seluruh *query* DDL dan DML yang akan dieksekusi menjelang ataupun pasca *deployment*, serta detail langkah proses *deployment*.

3.3.7 Perbaikan Bug Pasca Deployment

Setelah *deployment* dilaksanakan, akan dijalankan fase *monitoring* dan *regression testing* pada fitur-fitur terkait. Hasil *regression* menunjukkan bahwa terdapat *edge case* minor yang belum terjangkau dari hasil proses penyesuaian proyek ini, yaitu pada proses *refund*, yang dimana pelanggan tetap dapat mengulas produk yang telah melalui proses *refund* atau pengembalian. Untuk menanggapi

kejadian ini, proses perbaikan dan penyesuaian segera dilaksanakan oleh tim *developer*.

3.4 Kendala dan Solusi yang Ditemukan

Kendala yang ditemukan adalah inkonsistensi dan banyak penggunaan tabel terkait ulasan yang sulit dipahami, khususnya pada tahap awal (*onboarding*) dan masa awal *development*. Berbagai konteks dan alur terhadap berbagai tipe tabel yang memiliki atribut terkait ulasan cukup sulit untuk dipahami, khususnya jika belum pernah bertanggung jawab atau memiliki pengalaman atas proyek ataupun fitur terkait. Solusi yang ditemukan adalah mencoba untuk memahami secara perlahan dan selalu memastikan ketidakpastian ataupun keraguan dengan *supervisor*, baik itu pada saat *onboarding* ataupun masa *development*.

Selain itu, fitur ulasan merupakan salah satu fitur yang meningkatkan *resource usage* pada sistem oleh karena jumlah data ulasan yang sangat besar yang ditambah dengan implementasi teknikal pada *database* yang membutuhkan banyak operasi *join* pada setiap *query*. Fitur ini sempat menyebabkan *database* mengalami insiden "*down*" untuk beberapa saat oleh karena *query* yang memiliki *cost* yang tinggi. Solusi yang ditemukan dan diterapkan adalah melakukan optimasi pada *query-query* ataupun proses yang dianggap menggunakan banyak *resouce*, menganalisis setiap perubahan ataupun modifikasi pada *query* dengan melakukan pemeriksaan pada *execution plan*, serta melakukan fase monitoring dan analisis terhadap proses yang meningkatkan utilisasi *resource* sehingga tindakan pencegahan dapat dilakukan sesegera mungkin.