BAB III

PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi Pelaksanaan Kerja

Bab ini membahas kedudukan dan koordinasi kerja penulis selama magang di bitbybit. Penulis menempati posisi *UI/UX Designer Intern* pada divisi *Product* dan berkoordinasi langsung dengan *Supervisor* serta tim lintas divisi. Bagian ini menjelaskan posisi dalam struktur organisasi, tanggung jawab yang diemban, mekanisme koordinasi tugas, serta metode kerja yang diterapkan.

3.1.1 Kedudukan Pelaksanaan Kerja

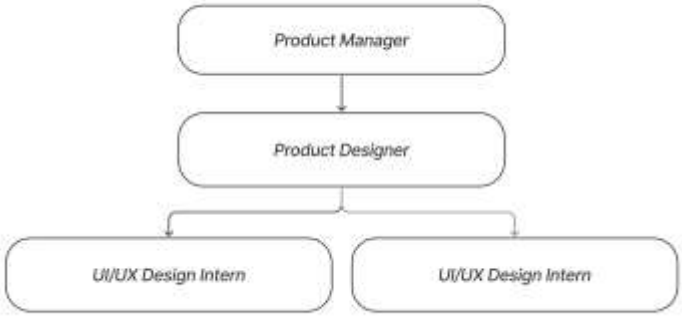
Selama periode magang dari 21 Juli 2025 hingga 21 Januari 2026, penulis menjabat sebagai *UI/UX Designer Intern* pada divisi *Product* di bitbybit, unit usaha dari PT Benar Benar Bagus. Magang dilaksanakan dengan sistem *hybrid*, di mana penulis bekerja dari kantor pada hari Selasa, Rabu, dan Jumat, serta bekerja dari rumah pada hari Senin dan Kamis. Penulis berada di bawah supervisi langsung *Product Manager* yang berperan sebagai pembimbing lapangan sekaligus koordinator utama dalam pembagian tugas dan evaluasi hasil kerja.

Dalam struktur organisasi perusahaan, divisi *Product* berkolaborasi erat dengan tim *Software Developer* dan *Business Development* untuk mendukung pengembangan produk. Sebagai *UI/UX Designer Intern*, penulis bertanggung jawab merancang desain *User Interface* dan *User Experience* untuk berbagai fitur yang sedang dikembangkan, baik dalam bentuk *feature improvement* maupun *feature development*. Selain itu, penulis juga berpartisipasi dalam kegiatan *testing* produk dan penyusunan dokumen pendukung seperti *handover file* dan *Product Requirement Document* (PRD) yang digunakan oleh tim *Software Developer* dalam proses implementasi.

3.1.2 Koordinasi Pelaksanaan Kerja

Koordinasi pelaksanaan kerja di bitbybit dilakukan secara terstruktur melalui sistem *agile* dengan metode *sprint* dua mingguan. Kebutuhan dan prioritas proyek berasal dari klien, *Co-Founder*, atau tim *Business Development* yang kemudian disampaikan kepada *Product Manager*. Pada awal *sprint*, *Product Manager* memimpin sesi *sprint planning* dan membagikan tugas kepada anggota tim *Product* serta *Software Developer* sesuai prioritas proyek. Selain itu, setiap pagi diadakan *daily standup* untuk menyampaikan rencana kerja harian, kendala, serta *update progress* dari masing-masing anggota tim.

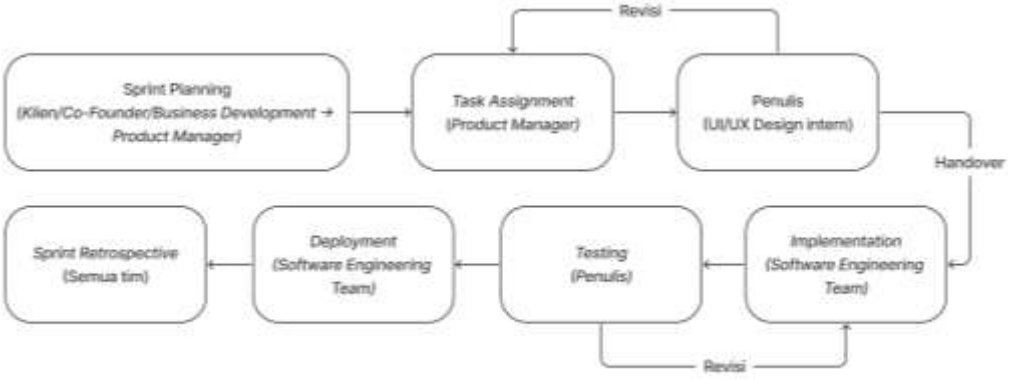
Selama *sprint* berlangsung, penulis juga mengikuti *weekly UI/UX sync* yang umumnya dilakukan secara *onsite* untuk membahas perkembangan desain. Dalam *weekly sync* ini, penulis menerima *review* dari *Product Manager* maupun *UI/UX Designer* lain dan mendiskusikan perbaikan yang diperlukan. Dalam hierarki tim *Product,* penulis berada di bawah supervisi *Product Manager* dengan struktur seperti gambar dibawah:



Gambar 3. 1 Bagan alur tim *Product*

Komunikasi sehari-hari menggunakan Google Chat untuk diskusi dan Jira sebagai *task management tool* untuk memantau status pekerjaan. Alur koordinasi dengan tim *Software Developer* dilakukan ketika desain siap untuk diimplementasikan, di mana *file* dan dokumen *Product Requirement Document* (PRD) diserahkan sebagai *handover*. Setelah tahap pengembangan, tugas biasanya *re-assign*ed kepada penulis untuk dilakukan

testing. Jika ditemukan *bug* atau ketidaksesuaian dengan desain, penulis akan melakukan *re-assign* kepada *Software Developer* terkait, hingga fitur dinyatakan siap untuk *deployment.*



Gambar 3. 2 Bagan Alur Koordinasi

Selain koordinasi internal dalam divisi *Product* dan *Software Developer,* penulis juga berinteraksi dengan divisi *Business Development* dalam proses *user research*. Tim *Business Development* yang sehari-hari menggunakan produk bitbybit, khususnya aplikasi bitChat, memberikan masukan terkait pengalaman pengguna dan kebutuhan fitur baru melalui *sync*. Penulis juga berkolaborasi dalam penyebaran kuesioner kepada *brand* di bawah PT Momen Indah Alami yang menggunakan produk bitbybit, khususnya tim *customer service*, untuk memperoleh *insight* lebih luas mengenai pengalaman penguna.

Di akhir setiap *sprint*, koordinasi ditutup dengan *backlog grooming*, *showcase frontend*, dan sesi *retrospective.* Pada tahap ini, tim secara kolektif membahas hal-hal yang berjalan baik, hambatan yang ditemui, serta strategi perbaikan yang perlu dilakukan pada *sprint* berikutnya. Koordinasi penulis bersifat kolaboratif lintas divisi, memungkinkan pengembangan produk lebih adaptif dan responsif terhadap kebutuhan *user*.

3.2 Tugas yang Dilakukan

Selama masa magang di bitbybit, penulis berperan sebagai *UI/UX Designer Intern* dalam divisi *Product.* Penulis terlibat dalam perancangan dan pengembangan fitur produk digital. Penulis menjalankan setiap tugas berdasarkan siklus kerja dua mingguan (*sprint*) dan dikoordinasikan melalui sistem Jira, di bawah bimbingan *Product Manager*.

Pekerjaan penulis melibatkan analisis kebutuhan pengguna (*user research*), perancangan alur dan desain UI/UX, pembuatan *Product Requirement Document* (PRD), serta koordinasi dengan tim *Software Developer* dalam proses implementasi. Beberapa tugas juga melibatkan kolaborasi dengan *intern* lain untuk kegiatan *user research* dan *ideation*. Berikut merupakan daftar kegiatan dan keterangan pekerjaan yang telah dilakukan selama masa magang:

Tabel 3. 1 Detail Pekerjaan yang Dilakukan Selama Kerja

Minggu	Tanggal	Proyek	Keterangan
2	28 Juli – 1 Agustus 2025	*Pin chat feature*	Membuat dan merancang fitur "*Pin Chat"* pada bitChat agar pengguna dapat menyematkan percakapan penting di atas daftar pesan.
2	30 Juli 2025	*bitChat Icon*	Mendesain ulang ikon aplikasi bitChat agar lebih konsisten dengan identitas visual produk.
3-6	4 Agustus – 29 Agustus 2025	*Increase AI Agent Activation Rate: AI Agent onboarding*	Melakukan penelitian dan perancangan ulang alur *onboarding AI Agent*, termasuk penyempurnaan tampilan dan interaksi dalam bitChat.
3-6	4 Agustus – 29 Agustus 2025	*Increase AI Agent Activation Rate: AI Studio polishing*	Melakukan analisis dan perancangan ulang *UI/UX* pada tampilan *AI Agent studio* agar lebih efisien dan mudah digunakan.

6	18 Agustus – 22 Agustus 2025	*Remove "Powered by bitbybit"*	Menambahkan fitur untuk menghapus *watermark* "*Powered by bitbybit*" pada halaman login WhatsApp.
5-6	18 Agustus – 29 Agustus 2025	*Push Users to Connect*	Merancang *banner* notifikasi untuk mendorong pengguna terhubung dengan saluran komunikasi yang sesuai.
7-8	1 September – 12 September 2025	*Translate Whole Ticket*	Menambahkan fitur penerjemahan otomatis untuk seluruh isi tiket dalam *menu live chat.*
7-10	1 September – 30 September 2025	*Improve bitChat React Native*	Melakukan analisis dan perancangan ulang UI/UX pada tampilan aplikasi bitChat agar lebih efisien dan mudah digunakan.
11-12	30 September – 10 October 2025	*Improvement on scheduled trigger automation*	Memperbaiki dan mengoptimalkan fitur automasi pada sistem *trigger* terjadwal di bitCRM.
13-14	13 Oktober – 24 Oktober 2025	*Improve payment method flow and Settings in Commerce*	Merancang ulang alur dan *User Interface* fitur *Custom Payment Method* untuk meningkatkan keterbacaan dan kemudahan penggunaan
13-14	13 Oktober – 24 Oktober 2025	*Improve order table in Commerce*	Mengoptimalkan tabel pesanan dengan memisahkan status dan menambahkan kolom baru untuk integrasi pengiriman.
15-16	27 October – 3 November 2025	*Prebuilt automation for birthday*	Mendesain fitur *pre-built automation* yang memungkinkan *brand* mengirimkan ucapan ulang tahun otomatis kepada pelanggan.

15-16	27 October – 3 November 2025	*Conversion API Improvements*	Memperbaiki tampilan dan *flow* pengaturan *Conversion API* agar memudahkan *brand* dalam melacak performa kampanye *marketing*.
17-18	10 November – 21 November 2025	*Order timeline improvement*	Meningkatkan tampilan *timeline* pesanan mengenai status dan riwayat pesanan pelanggan dengan parameter yang terbaru,
17-18	10 November – 21 November 2025	*Order tags settings*	Merancang fitur pengaturan *tag* untuk kategorisasi pesanan di *settings* untuk menambah, mengedit, dan menghapus *tag*.
19-20	24 November – 5 Desember 2025	*Review automation triggers & actions (including AI)*	Melakukan riset dan analisis sistem *automation triggers* dan *actions* untuk meningkatkan fleksibilitas automasi berbasis AI.

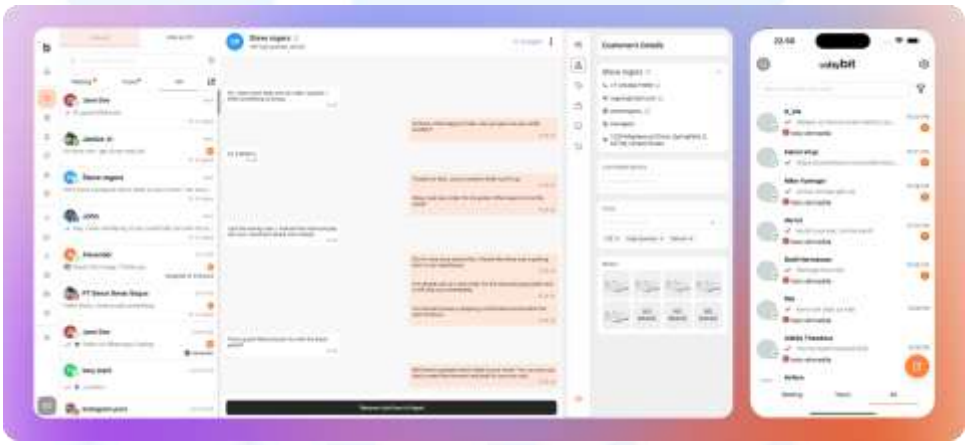
3.3 Uraian Pelaksanaan Kerja

Selama masa magang di bitbybit, penulis terlibat dalam berbagai proses perancangan *User Interface* dan *User Experience* yang berfokus pada pengembangan produk internal perusahaan. Tugasnya mencakup perancangan *User Interface* aplikasi, mengembangkan fitur atau anatarmuka yang sudah ada, hingga penyusunan dokumentasi produk. Satu proyek ditetapkan sebagai tugas utama, yaitu perbaikan *User Interface* dan pengalaman pengguna aplikasi bitChat *React Native*, sedangkan empat proyek lainnya bersifat pendukung, meliputi pembuatan fitur baru, mengembangkan *flow*, fitur atau antermuka yang sudah ada.

3.3.1 Proses Pelaksanaan Tugas Utama Kerja

Proyek utama yang penulis kerjakan selama masa magang di bitbyBit adalah *Improve BitChat React Native*, yaitu perancangan ulang tampilan serta peningkatan pengalaman pengguna pada aplikasi bitChat versi *React Native*. bitChat sendiri merupakan platform *Customer*

Relationship Management (CRM) berbasis WhatsApp, Instagram dan Facebook yang membantu *brand* menjaga komunikasi langsung dengan pelanggan. Melalui aplikasi ini, pengguna dapat mengelola pesan masuk, mengirim pesan siaran, dan mengintegrasikan *chatbot* untuk mempermudah interaksi. Namun, versi aplikasi *mobile* mengalami tingkat adopsi yang rendah di kalangan *customer service agent* karena keterbatasan fitur dan pengalaman penggunaan yang kurang optimal dibandingkan versi *website*. Oleh karena itu, proyek ini bertujuan untuk meningkatkan kepuasan pengguna dan memperluas penggunaan *bitChat React Native* melalui perbaikan *User Interface* serta penambahan fitur-fitur penting.

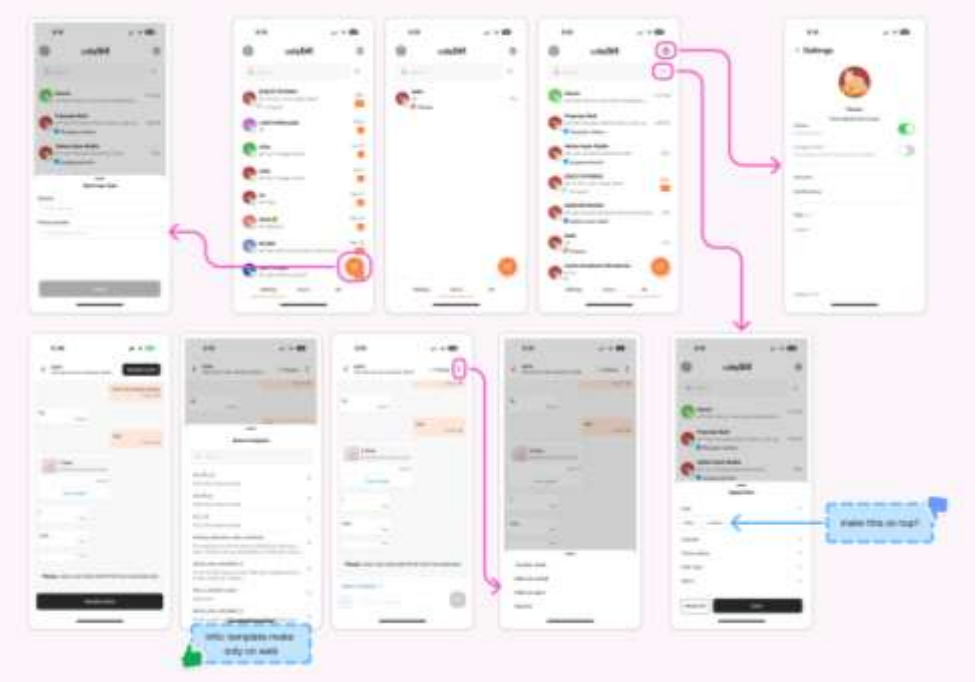


Gambar 3. 3 bitChat dan bitChat *React Native*
Sumber: bitbybit (2025)

Proyek ini dipilih sebagai tugas utama karena memiliki kompleksitas tinggi dan cakupan pekerjaan yang luas. Pengerjaan tidak hanya mencakup perbaikan *User Interface*, tetapi juga penambahan berbagai fitur baru yang sebelumnya hanya tersedia di versi *website*. Proyek ini juga menuntut kolaborasi lintas divisi, mulai dari pengumpulan *insight* pengguna di tim *Business Development*, koordinasi dengan tim *Product*, hingga komunikasi dengan tim *Software Developer*. Dalam proses pengerjaannya, penulis menerapkan tahapan perancangan sesuai dengan metode perancangan *Design Thinking*, yaitu *empathize, define, ideation, prototyping,* dan *testing*.

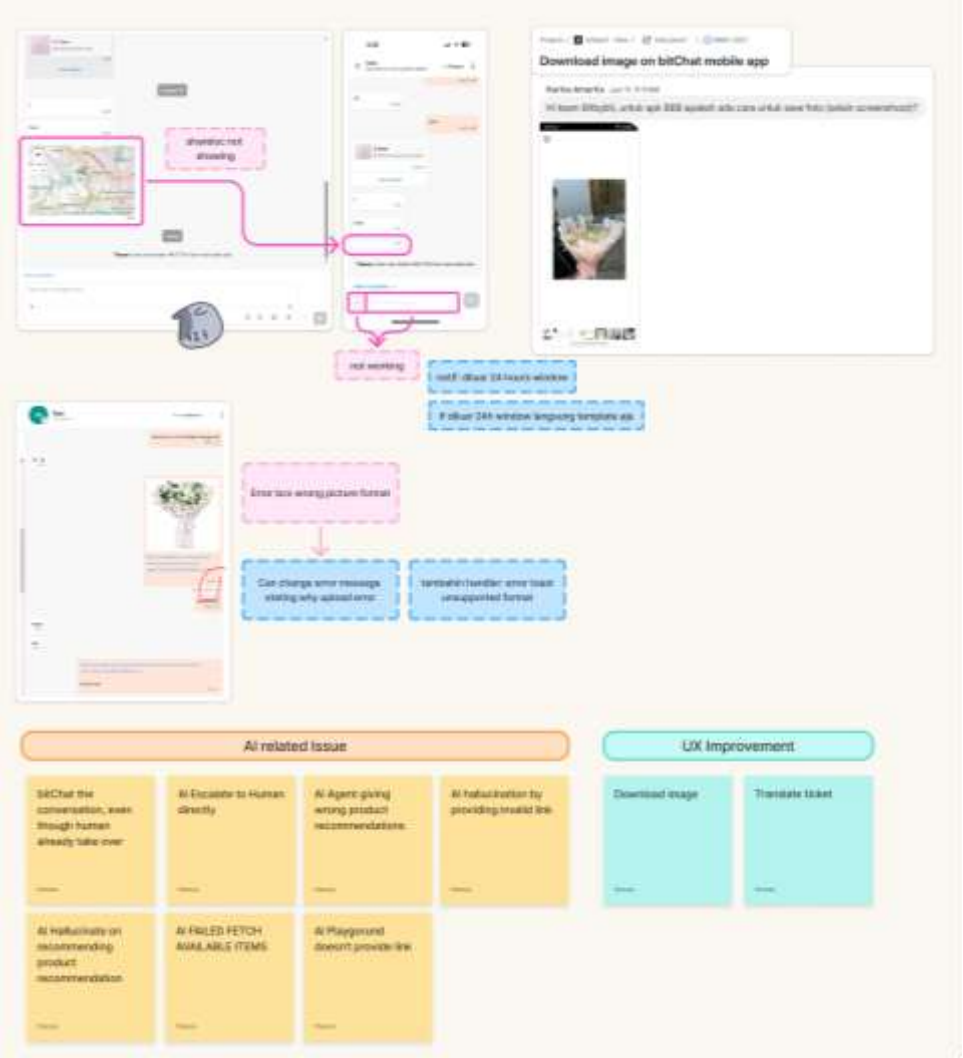
A. Empathize

Tahap awal perancangan dimulai dengan analisis aplikasi *bitChat React Native* sekarang. Eksplorasi dilakukan secara langsung terhadap *User Interface* dan *User Experience* untuk mengidentifikasi elemen yang berpotensi diperbaiki. Beberapa temuan awal meliputi inkonsistensi komponen antarhalaman, tata letak yang belum optimal, serta alur interaksi yang kurang efisien dalam pengelolaan percakapan pelanggan.



Gambar 3. 4 Analisa bitChat *React Native*

Selain analisis aplikasi, *backlog* pada Jira ditelusuri untuk menemukan daftar tugas atau isu terkait bitChat React Native yang belum terselesaikan. Percakapan internal pada *Google Chat* juga ditinjau untuk mencatat keluhan dari agen *customer service* di bawah naungan PT Momen Indah Alami. Keluhan tersebut mencakup kesulitan dalam navigasi, lambatnya *loading* aplikasi, serta kurangnya fitur yang memudahkan pekerjaan mereka.



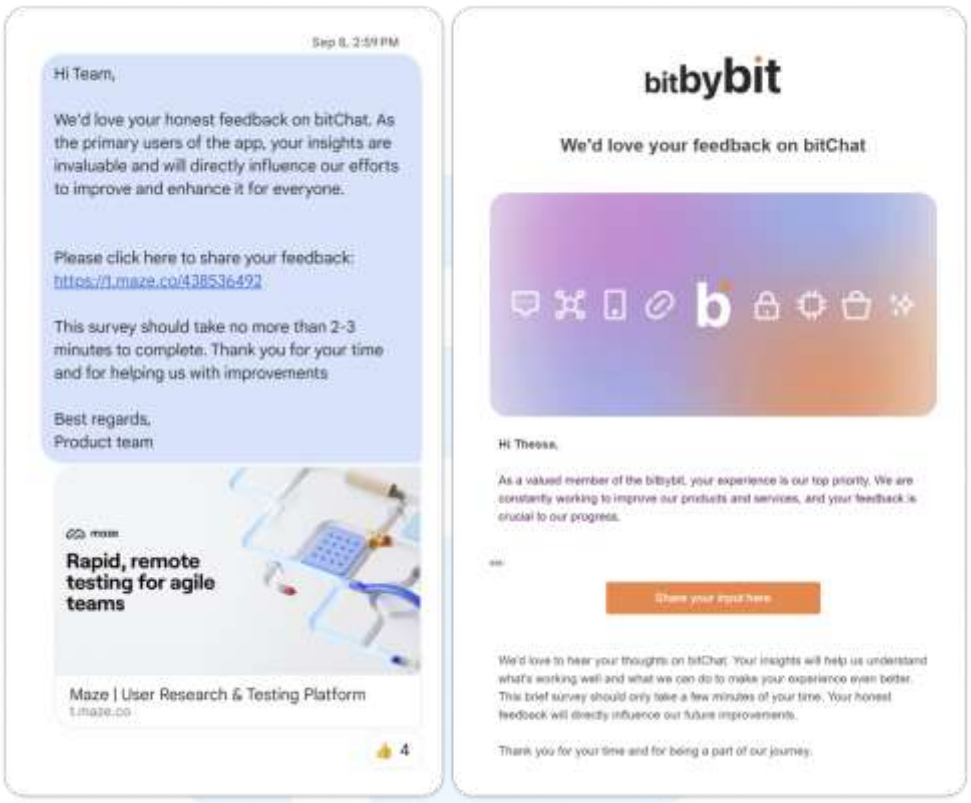
Gambar 3. 5 Hasil riset internal

Analisis kompetitor dilakukan untuk memahami pasar dan fitur unggulan aplikasi serupa di industri *Customer Relationship Management* (CRM). Aplikasi yang dianalisis meliputi Tidio dan LiveChat, yang merupakan platform *live chat* populer dengan basis pengguna luas. Analisis dilakukan dengan mengunduh aplikasi tersebut, mengeksplorasi fitur yang tersedia, dan membandingkannya dengan bitChat *React Native*.

Gambar 3. 6 *Competitor Analysis* bitChat RN

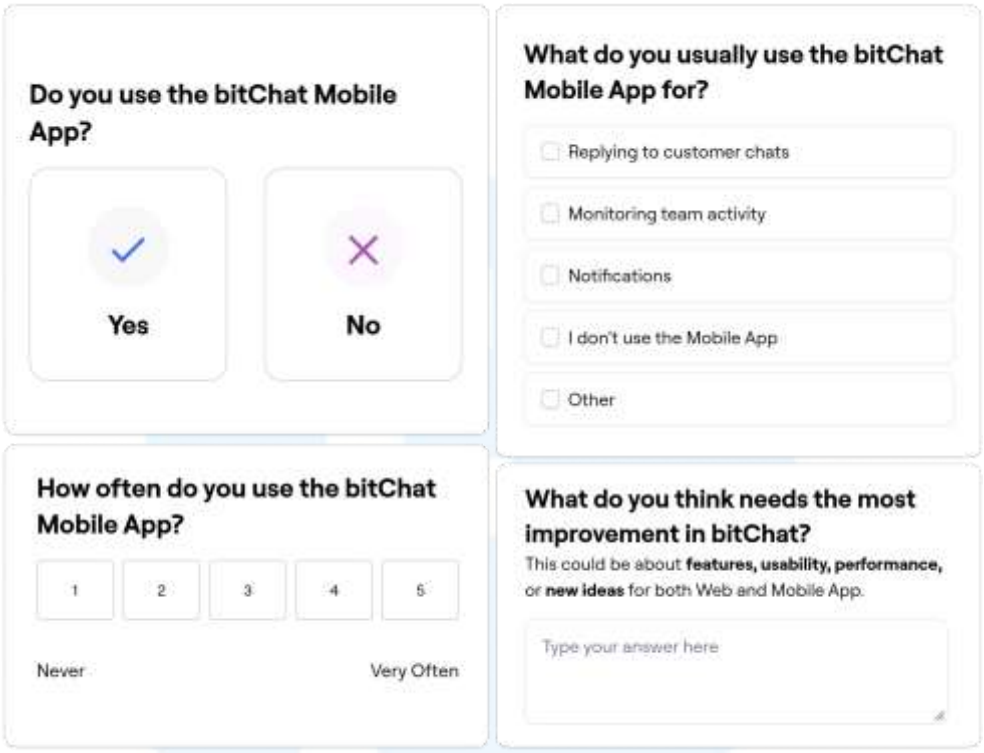
Dari analisis ini ditemukan beberapa fitur yang dapat diadopsi atau dijadikan inspirasi. Tidio memiliki *User Interface* yang bersih dengan ikon yang konsisten pada setiap fungsi, sesuatu yang belum sepenuhnya diterapkan di bitChat. Fitur seperti catatan, status tiket, dan *AI Help* membantu kolaborasi dan efisiensi. Berdasarkan diskusi dengan *Product Manager*, fitur serupa belum menjadi prioritas karena rendahnya penggunaan *AI assistant* pada versi *desktop*. LiveChat memiliki *User Interface* yang minimalis, ikon dan teks yang memperjelas navigasi. Fitur kolaborasi timnya juga lengkap. Hasil analisis kompetitor ini memberikan *insight* berharga tentang standar industri dan ekspektasi pengguna terhadap aplikasi *live chat*.

Untuk mendapatkan *feedback* langsung dari pengguna, penyebaran kuesioner dilakukan kepada agen *customer service*. Kuesioner dirancang untuk mengumpulkan data tentang pengalaman pengguna saat menggunakan bitChat *React Native*, fitur yang paling sering digunakan, kendala yang dihadapi, serta saran perbaikan. Penyebaran kuesioner dilakukan melalui *Google Forms* dan dibagikan kepada sekitar 700 agen *customer service* di bawah naungan PT Momen Indah Alami.



Gambar 3. 7 *E-mail campaign template* bitChat RN

Penulis dan rekan *UI/UX intern* mempresentasikan hasil kuesioner dalam sesi *UI/UX Sync* bersama tim *Product*. Dari delapan respons yang diterima, ditemukan bahwa hanya 29% pengguna yang pernah memakai aplikasi bitChat *React Native*, dan lebih dari 50% di antaranya jarang menggunakannya. Mayoritas keluhan berkaitan dengan kestabilan aplikasi, *bug*, dan pengalaman penggunaan yang membingungkan.



Gambar 3. 8 Pertanyaan kuesioner riset bitChat RN

Penulis melanjutkan proses riset dengan melibatkan pengguna internal bitbybit melalui kegiatan *Focus Group Discussion* (FGD) bersama tim *Business Development*. Kegiatan ini bertujuan memperdalam pemahaman terhadap tantangan dan kebutuhan pengguna internal yang berinteraksi langsung dengan klien setiap harinya. Dalam sesi diskusi ini, tim menyiapkan sejumlah pertanyaan terbuka seperti:

1. *What are the biggest frustrations you face when dealing with customers?*
2. *What is one thing about the app you find most frustrating?*
3. *What is one thing you find most useful about the app right now?*
4. *If you could add any feature to BitChat app to make your job easier, what would it be and why?*

Diskusi berlangsung secara interaktif dan menghasilkan banyak wawasan. Tim *Business Development* mengungkapkan bahwa salah satu hambatan terbesar adalah sulitnya mengelola pesan dalam jumlah besar dan memastikan pesan prioritas tidak terlewat. Beberapa peserta juga menyoroti

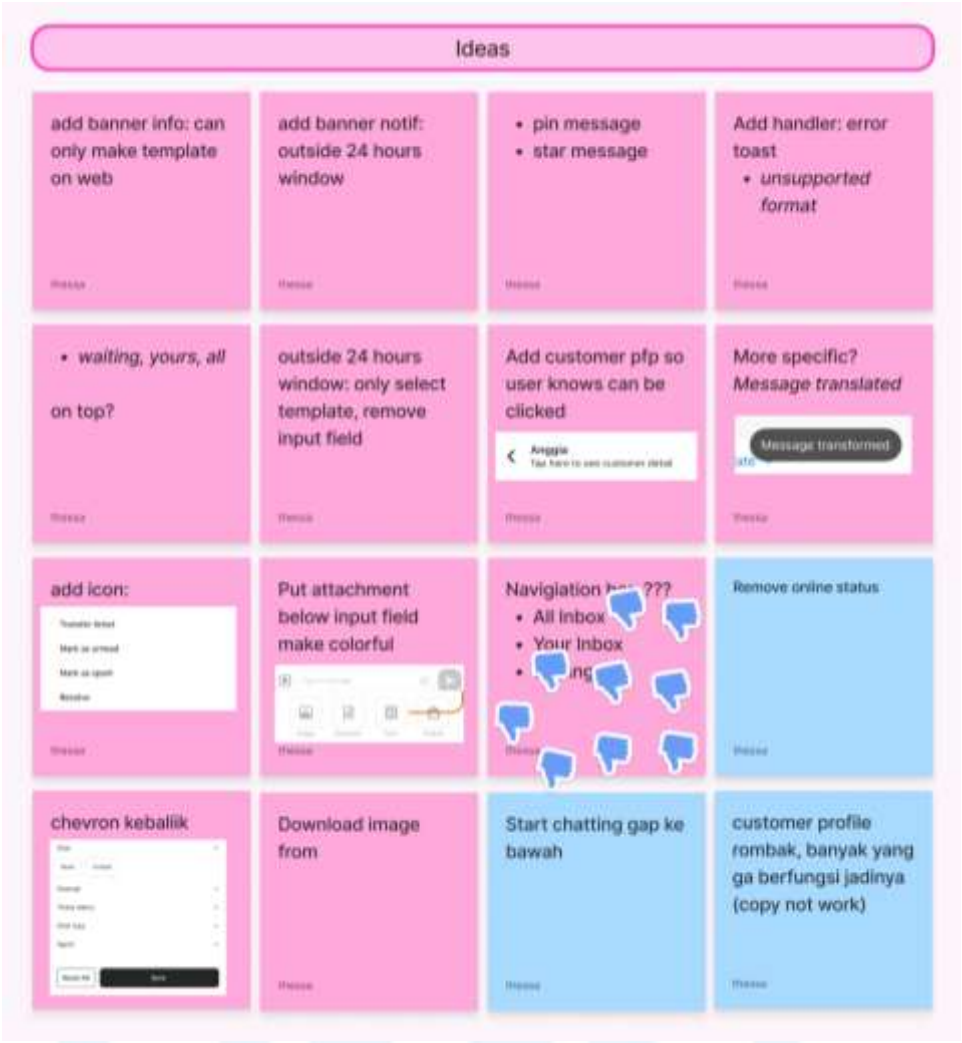
masalah konsistensi tampilan serta keterbatasan fitur pada versi *mobile* yang membuat mereka lebih memilih versi *website*. Selain itu, diskusi ini juga memberikan masukan berharga terkait pengembangan fitur baru seperti *priority badge* dan sistem pencarian yang lebih intuitif.

B. Define

Masalah utama pada aplikasi bitChat adalah *User Interface* (UI) yang masih sangat dasar dan belum dioptimalkan untuk perangkat *mobile*. Mayoritas komponennya masih merupakan salinan dari versi *desktop* tanpa menyesuaikan kebutuhan layar. Selain itu, terdapat sejumlah fitur penting dari versi *website* yang belum diimplementasikan, sehingga pengalaman pengguna terasa belum maksimal dan tidak seimbang dengan versi *desktop*.

Berdasarkan diskusi dengan *Product Manager* dan tim *Business Development*, aplikasi bitChat versi *mobile* dianggap bukan prioritas utama. Tim menilai aplikasi ini hanya sebagai pelengkap platform utama berbasis *website*. Namun, hal ini bertentangan dengan ekspektasi mereka sendiri yang tetap menginginkan ulasan positif dari aplikasi. Terdapat juga laporan dari klien bitbybit yang menyampaikan keluhan mengenai *error* di aplikasi *mobile*, menunjukkan bahwa aplikasi ini tetap penting dan perlu perhatian setara dengan versi *desktop* agar tetap terlihat profesional.

Tujuan utama dari perancangan ulang ini adalah untuk meningkatkan *usability* serta menyatukan *design system* antara versi *mobile* dan *desktop*. Dengan memperbaiki inkonsistensi visual, menambah fitur yang hilang dari versi web, dan mendesain *interface* khusus untuk aplikasi *mobile*, diharapkan dapat memberikan pengalaman yang lebih baik bagi pengguna. Dalam jangka panjang, langkah ini juga diharapkan dapat meningkatkan tingkat adopsi aplikasi serta menghasilkan ulasan positif dari pengguna bitChat *mobile*. Berikut adalah hasil *define* dari kuesioner dan FGD yang dilakukan sebelumnya dengan tim *Business Development:*



Gambar 3. 9 Hasil *define* setelah *FGD* bitChat RN

Untuk meningkatkan kejelasan dan kemudahan penggunaan, perlu ditambahkan *banner* informasi bahwa pembuatan *template* "*quick reply*" hanya dapat dilakukan melalui versi *desktop*. Selain itu, *banner* tambahan juga perlu muncul ketika pengguna mencoba mengirim pesan di luar jendela 24 jam agar pengguna tidak kebingungan ketika *input field* dinonaktifkan. Beberapa fitur tambahan seperti pin message, unduh gambar, terjemahan pesan, serta *error toast* yang perlu diterapkan. Sementara itu, tampilan *customer profile* perlu sepenuhnya didesain ulang karena masih berantakan akibat penggunaan komponen yang diadaptasi langsung dari versi *desktop*.

Gambar 3. 10 Permasalahan *bugs* sesuai FGD

Selain itu masih ditemukan sejumlah *bug* dan inkonsistensi yang mengganggu pengalaman pengguna dalam menggunakan aplikasi. Contohnya, tombol kembali pada perangkat Android tidak berfungsi ketika filter aktif, daftar agen tidak menampilkan *AI Agent*, serta fitur "*share location*" yang tidak muncul. Selain itu, perubahan perusahaan tidak tersimpan setelah aplikasi ditutup, halaman bantuan belum terhubung ke Mintlify (aplikasi berisi mesin dokumentasi kode), dan beberapa tampilan masih tidak konsisten secara visual.

Keterbatasan perancangan terutama disebabkan prioritas pengembangan dan keputusan internal tim. Berdasarkan masukan dari *Product Manager*, beberapa ide fitur dinilai belum relevan bagi sebagian besar klien bitChat yang merupakan bisnis kecil. Selain itu, hasil FGD juga memperlihatkan bahwa tim internal masih menganggap aplikasi ini sebagai pelengkap, sehingga pengembangan fitur baru menjadi terbatas.

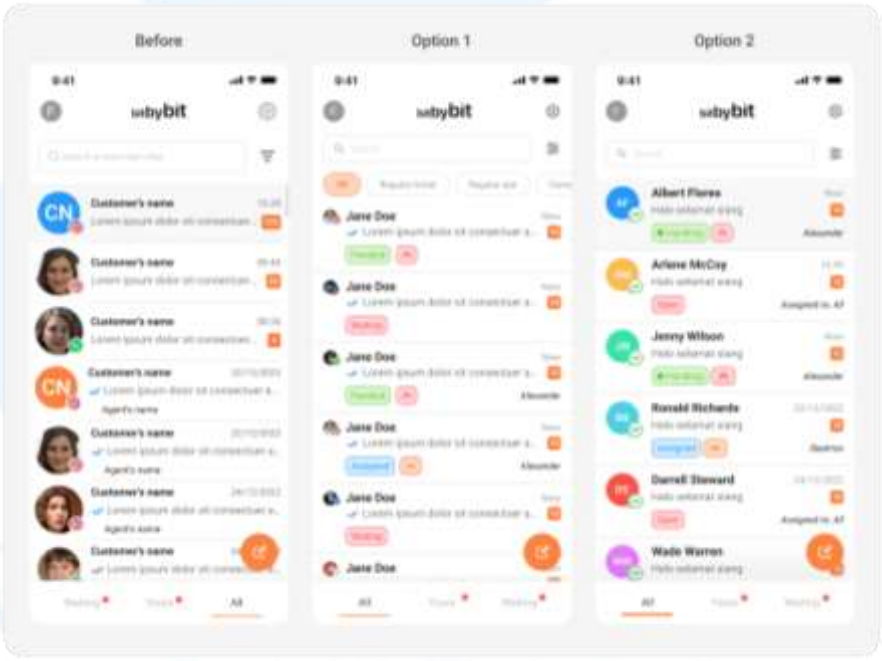
C. Ideate

Tahap *ideation* dimulai dengan melihat kembali rancangan *User Interface* bitChat *React Native* yang sudah ada serta hasil analisis kompetitor yang dilakukan pada tahap sebelumnya. Karena sebagian besar komponen desain sudah tersedia dari *design system* yang digunakan tim *Product*, proses *ideation* berfokus pada optimalisasi tata letak, penyusunan ulang hierarki visual, dan penyesuaian komponen agar lebih sesuai dengan karakteristik aplikasi *mobile*.

Dalam perancangan ini, penulis menggunakan Figma sebagai alat utama untuk menyusun rancangan awal dalam bentuk *high-fidelity* design. Tahapan *ideation* dilakukan secara iteratif, dimulai dengan membuat versi awal dari tampilan dan *flow* baru berdasarkan kebutuhan yang telah didefinisikan sebelumnya. Rancangan awal ini dibuat sebagai dasar eksplorasi visual dan fungsionalitas.

1. *Home User Interface*

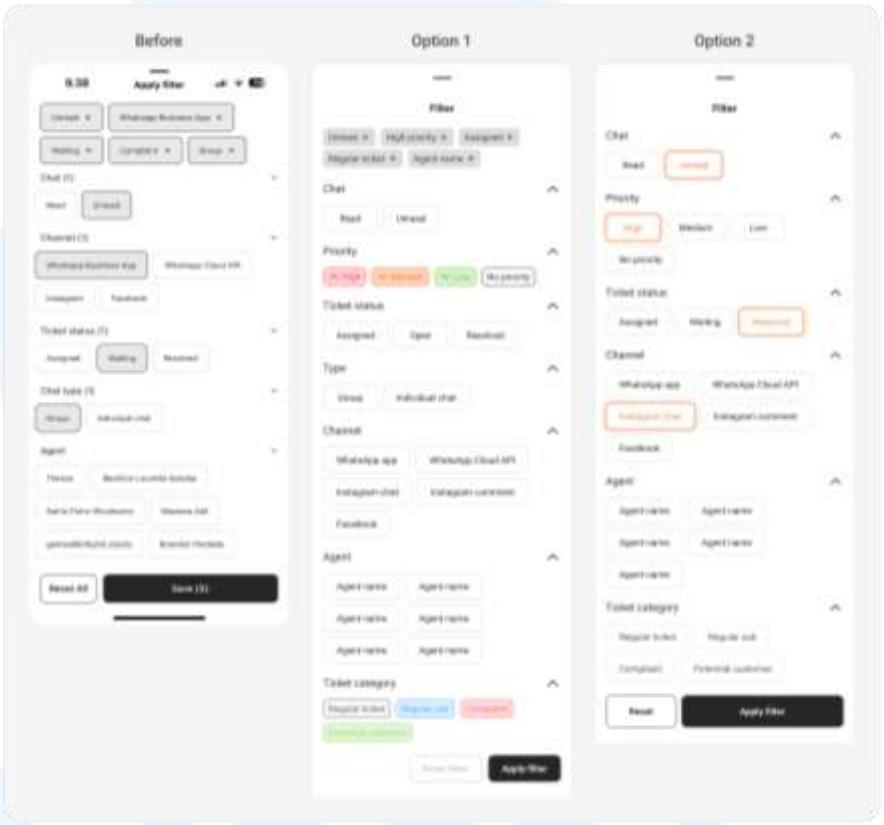
Tahap perancangan *Home User Interface* dimulai dengan menggunakan desain eksisting sebagai acuan utama. Selama tahap ini, rekan *UI/UX Designer Intern* lainnya mengerjakan *desktop version improvements*, sehingga penulis perlu menyesuaikan rancangan aplikasi *mobile* agar tetap konsisten. Integrasi fitur baru yang ditambahkan meliputi *priority badge* dengan empat status: *low, medium, high,* dan *no priority*, serta pembaruan tampilan untuk status tiket seperti *Handled*, *Waiting*, *Assigned*, dan *Blocked*.



Gambar 3. 11 *Progress* desain *home* 1

Iterasi awal menampilkan dua opsi tampilan dengan perbedaan ukuran foto profil pengguna. Berdasarkan *feedback* tim *Product*, opsi dengan foto profil lebih kecil dipilih karena pada

implementasi *WhatsApp API* gambar profil pengguna tidak muncul, sehingga elemen tersebut tidak perlu menjadi fokus visual utama dan kurang efisien dalam penggunaan ruang layar. Oleh karena itu, desain yang akhirnya dipilih merupakan opsi kedua.

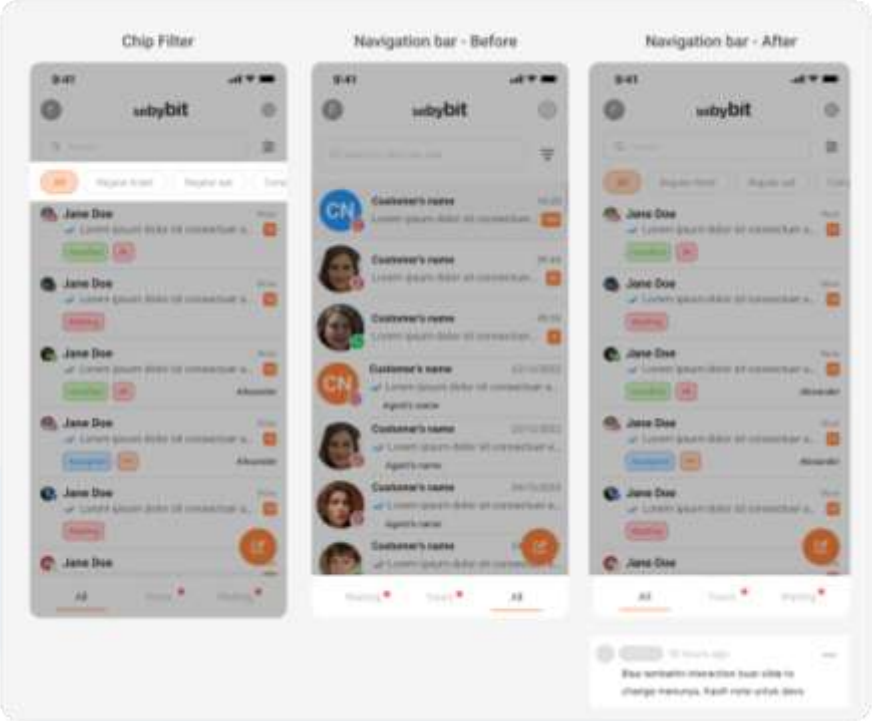


Gambar 3. 12 *Progress* desain *chat filter*

Untuk *filter chat option*, dua opsi diajukan kepada *Supervisor*. Opsi pertama berupa sistem *tagging* yang diperbarui dari versi sebelumnya dengan tampilan yang lebih rapi, sedangkan opsi kedua menghapus *tags* dan menggantinya dengan perubahan warna pada tombol yang dipilih. *Supervisor* memilih opsi kedua karena tampil lebih sederhana, namun memberikan *feedback* agar seluruh warna dan tinggi tombol diseragamkan agar konsisten.

Gambar 3. 13 *Progress* desain *applied filter*

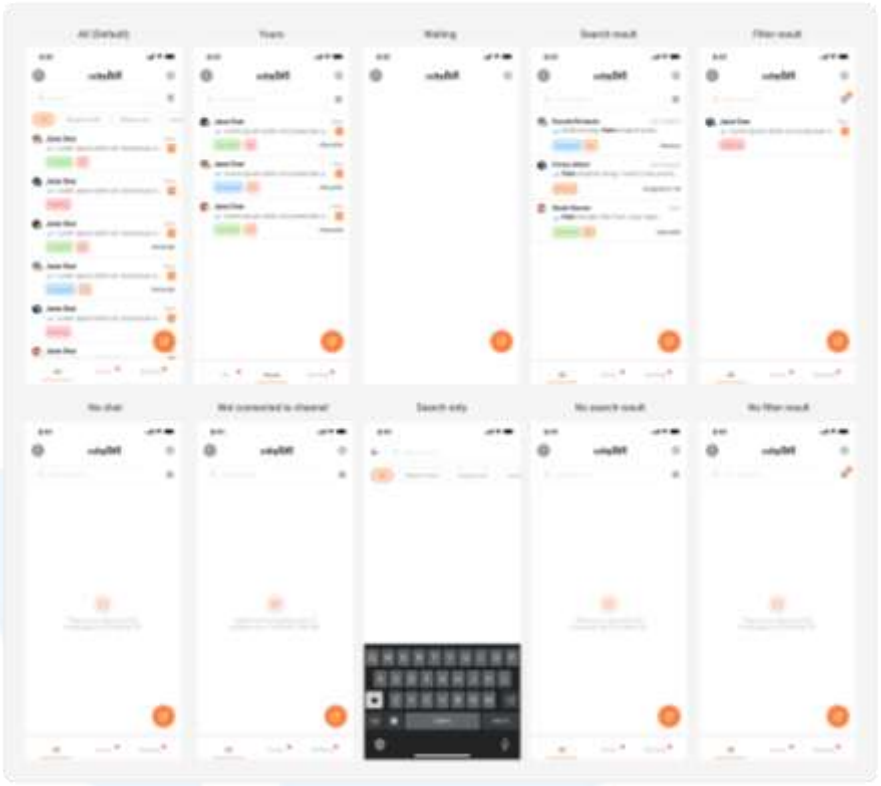
Selain itu, penulis juga memperbarui ikon *filter* dan merancang dua alternatif tampilan untuk kondisi ketika *filter* telah diterapkan. Opsi pertama mempertahankan gaya visual sebelumnya, sementara opsi kedua menampilkan tampilan yang lebih bersih menyerupai notifikasi. Hasil diskusi dengan *Supervisor*, opsi kedua dipilih karena tampil lebih minimalis menyerupai *notification state* tanpa mengganggu fokus utama.



Gambar 3. 14 *Progress* desain *home* 2

Penulis menambahkan *chip filter* yang ditempatkan di bawah kolom pencarian dan tombol *filter* untuk mempermudah pengguna dalam menyaring percakapan berdasarkan kategori tertentu. *Chip filter* ini bersifat dinamis, mengikuti kategori yang telah diatur oleh masing-masing perusahaan, sehingga setiap pengguna dapat memiliki tampilan *filter* yang berbeda sesuai kebutuhan operasionalnya. Fitur ini membantu mempercepat navigasi dalam mengelola percakapan berjumlah besar.

Penulis juga melakukan perubahan urutan navigasi, dari "*Waiting – Yours - All*" menjadi "*All - Yours - Waiting*" untuk meningkatkan alur penggunaan. *Supervisor* memberikan masukan tambahan agar penulis menambahkan catatan kepada tim *Developer* untuk menyertakan *slide interaction* saat berpindah antar menu. Penulis menambahkan catatan ini di *Product Requirement Document* di tahap akhir.



Gambar 3. 15 *Progress* desain *home* 3

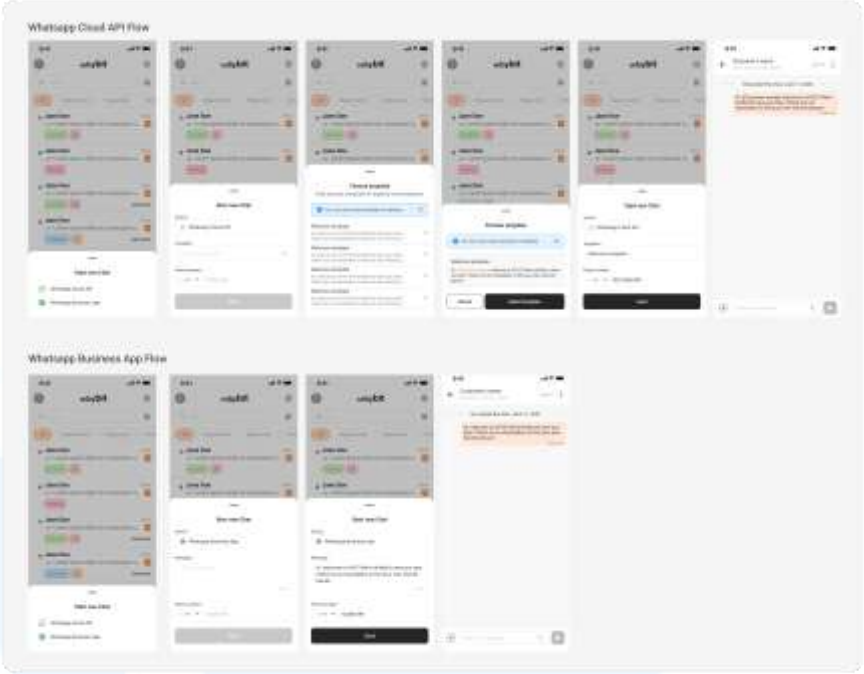
Setelah tampilan utama *Home UI* disetujui dan mendapatkan *feedback* positif, tahap berikutnya adalah pembuatan kondisi tampilan. Setiap *state* mencakup skenario seperti tampilan pada masing-masing menu navigasi (*All*, *Yours*, dan *Waiting*), kondisi saat pengguna melakukan pencarian, menerapkan *filter*, tidak ada *chat*, belum terhubung dengan *communication channel* seperti WhatsApp, saat pengguna sedang mengetik di kolom pencarian, tidak menemukan hasil pencarian, dan kondisi tanpa hasil *filter*. Semua *state* ini dijabarkan secara detail agar tim *Software Developer* memiliki panduan visual yang jelas dalam mengimplementasikan setiap kemungkinan.

Gambar 3. 16 *Home UI feedback Supervisor*

Selama proses revisi, terdapat masukan dari *Supervisor* mengenai tampilan *text field* saat berada pada *active state*. Desain saat ini belum memiliki kondisi *active* seperti pada versi *desktop*. Hasil diskusi menyepakati bahwa untuk saat ini, fitur tersebut belum perlu ditambahkan pada versi *mobile*.

2. *Start new chat flow*

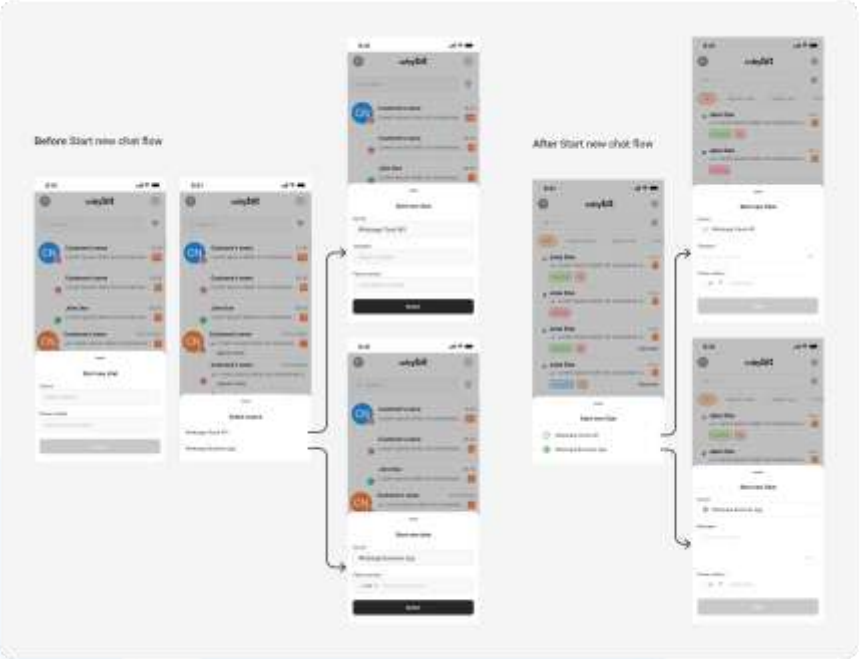
Tahap berikutnya adalah perancangan alur *Start New Chat*. Karena aplikasi ini berfungsi sebagai *Customer Relationship Management (CRM)* bagi agen layanan pelanggan, proses memulai percakapan tidak dapat dilakukan secara langsung seperti pada platform perpesanan pribadi umumnya. Sistem ini terintegrasi dengan beberapa *communication channel*, yaitu WhatsApp Business App, WhatsApp Cloud API, Instagram, dan Facebook. Namun, pada tahap *redesign* ini fokus diarahkan pada integrasi dengan WhatsApp Business App dan WhatsApp Cloud API karena merupakan *communication channel* utama yang paling sering digunakan oleh klien bitbybit.



Gambar 3. 17 *Progress* desain *start chat flow* 1

Dalam sistem CRM ini, agen layanan pelanggan harus memilih sumber (*source*) terlebih dahulu sebelum memulai percakapan. Pemilihan sumber ini penting karena setiap *communication channel* memiliki aturan dan batasan yang berbeda. Untuk WhatsApp Cloud API, agen tidak dapat langsung mengetik pesan baru saat membuka percakapan dengan pelanggan.

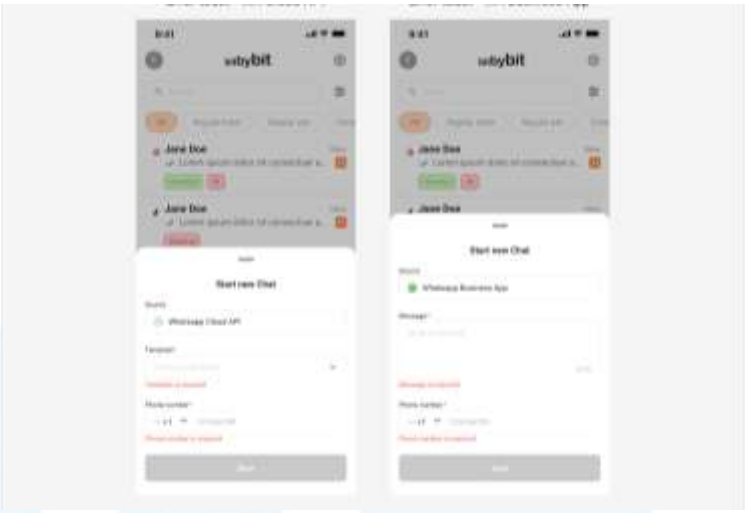
Pesan yang dikirim harus menggunakan *chat template* yang sebelumnya sudah dibuat atau disetujui melalui aplikasi *desktop*. Selain itu, *chat template* wajib mendapatkan persetujuan dari pihak Meta agar dapat digunakan secara resmi. Sebaliknya, untuk WhatsApp Business App, *Custiomer Service Agent* dapat langsung mengetik pesan secara bebas tanpa perlu menggunakan *chat template*.



Gambar 3. 18 Desain *Before* dan *After start new chat flow*

Perubahan utama yang diterapkan dalam alur ini adalah urutan tampilan *modal*. Pada desain sebelumnya, *modal* pertama yang muncul menampilkan *input field* untuk memilih sumber dan nomor telepon secara bersamaan. Penulis menemukan bahwa tampilan lanjutan setelahnya akan berbeda tergantung sumber yang dipilih. Oleh karena itu, desain baru mengganti alur tersebut.

Langkah pertama hanya berisi pilihan sumber, kemudian dilanjutkan dengan *modal* berikutnya yang menyesuaikan *input field* berdasarkan sumber yang telah dipilih. Selain itu, ditambahkan ikon di sebelah nama *communication channel* agar pengguna dapat mengenali sumber dengan lebih cepat, serta tanda asterisk (*) untuk menandai kolom wajib guna meningkatkan kejelasan dan mencegah kesalahan *input* dari pengguna.



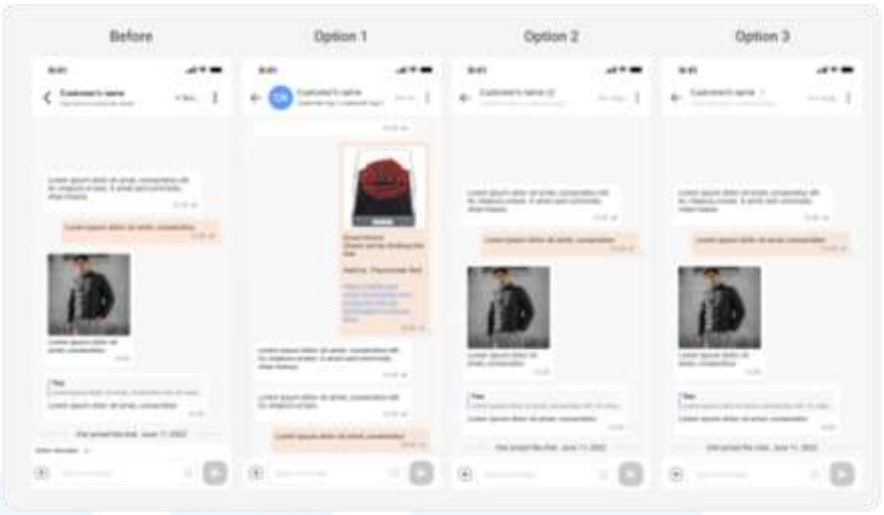
Gambar 3. 19 *Progress* desain *start chat flow* 1

Setelah mendapatkan *feedback* dari *Supervisor*, terdapat tambahan revisi terkait kondisi *error state* pada *modal*. Penulis menambahkan *error state* untuk setiap kolom yang belum terisi, sehingga pengguna dapat langsung mengetahui bagian mana yang masih harus dilengkapi. Selain itu, tombol "*send"* juga diatur agar dalam keadaan *disabled* hingga seluruh kolom wajib telah terisi.

3. *Chat User Interface*

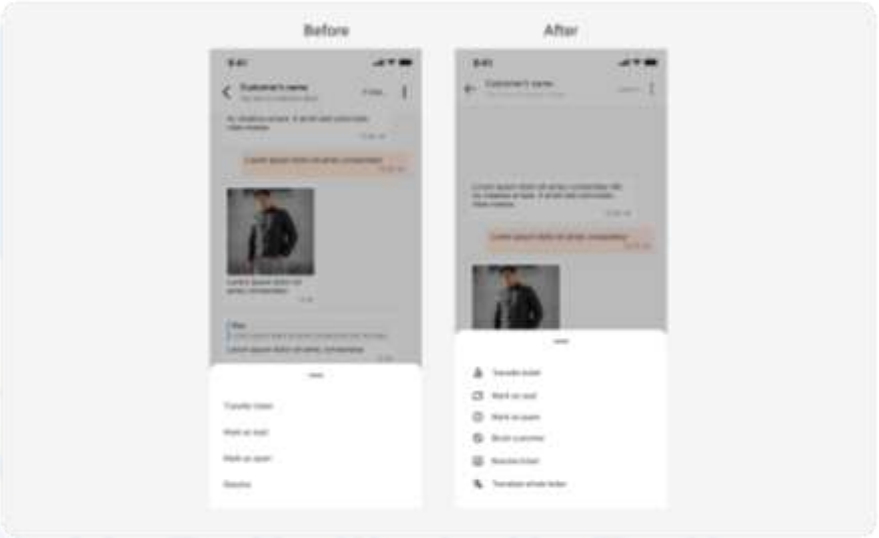
Pada bagian ini dirancang tampilan *chat* ketika *Customer Service Agent* memasuki ruang percakapan dengan pelanggan. Pertama-tama, bagian *header* didesain dengan beberapa pembaruan. Ikon panah "*back*" diganti dengan versi yang lebih besar agar memiliki area klik (*hitbox*) yang lebih luas. Di bawah nama pelanggan terdapat *caption* kecil bertuliskan "*Tap here to see customer's details."* Saat pengguna menekan nama atau *caption* tersebut, akan muncul halaman baru berisi detail pelanggan. Untuk bagian ini, penulis sempat memberikan beberapa opsi kepada tim:

1. Menambahkan foto profil dan mengganti *caption* di bawah nama pelanggan menjadi *customer tags.*
2. Tidak menggunakan foto profil, namun menambahkan ikon "*link out*" di sebelah nama pelanggan.
3. Mengganti ikon "*link out*" dengan ikon *chevron* ">".



Gambar 3. 20 *Proses* desain *header chat*

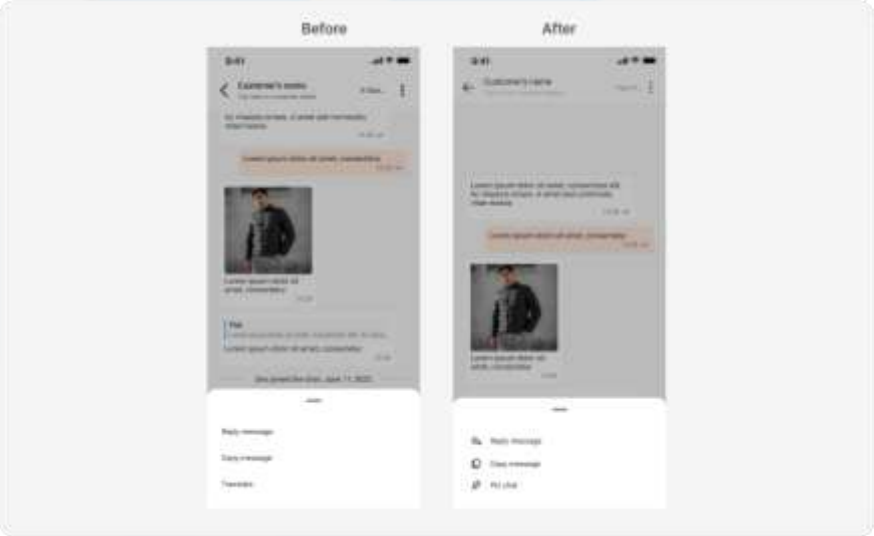
Setelah berdiskusi dengan tim, penulis memutuskan untuk tidak menggunakan foto profil karena pada WhatsApp API sering kali foto profil tidak muncul, sehingga hanya akan membuang ruang. *Supervisor* juga memberikan masukan agar cukup menggunakan teks "*Tap here to see details*," karena opsi tersebut dinilai paling jelas dan mudah dipahami dibandingkan menggunakan *tags*, ikon *link out*, atau *chevron* yang kurang intuitif.



Gambar 3. 21 Desain *Before* dan *After kebab menu modal*

Selanjutnya, penulis memperbarui tampilan *kebab menu* di pojok kanan atas dengan versi yang disesuaikan dengan *design*

system terbaru. Selain itu, penulis juga menghapus indikator titik hijau pada *status bar* yang menandakan *Customer Service Agent* sedang aktif. *Supervisor* menyarankan penghapusan elemen tersebut karena status aktif sudah dapat diasumsikan ketika *Customer Service Agent* sedang menangani *chat*.



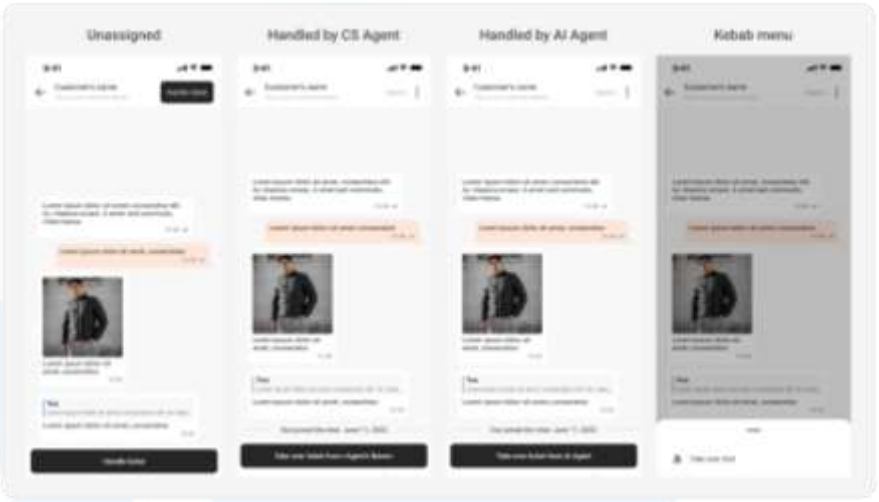
Gambar 3. 22 Desain *Before* dan *After long-press menu modal*

Pada *kebab menu,* sebelumnya hanya terdapat empat opsi, yaitu *transfer ticket, mark as unread, mark as spam,* dan *resolve.* Penulis menambahkan dua opsi tambahan, yaitu "*Block customer*" dan *"Translate whole ticket"* agar sesuai dengan versi *desktop.* Penulis juga menambahkan ikon pada setiap opsi agar tampil lebih menarik dan konsisten secara visual. Ikon baru ini juga dirancang langsung oleh penulis agar sesuai dengan *design system* saat ini.

Gambar 3. 23 Desain Ikon

Hal yang sama diterapkan pada *modal* yang muncul saat pengguna menekan lama (*long-press*) pada *chat bubble.*

Sebelumnya, opsi yang tersedia adalah *reply message, copy message,* dan *translate.* Kini, opsi *translate* diganti menjadi *pin chat* dan juga dilengkapi dengan ikon baru hasil rancangan penulis.



Gambar 3. 24 *State* tampilan chat

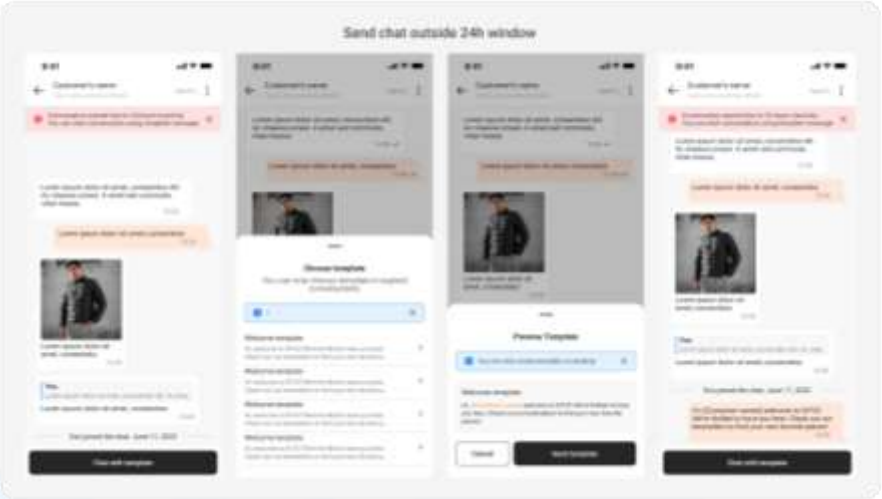
Selanjutnya tampilan *chat* dalam beberapa *state* berbeda:

(1) Ketika belum ditangani,

(2) Sedang ditangani oleh *Customer Service Agent* lain, dan

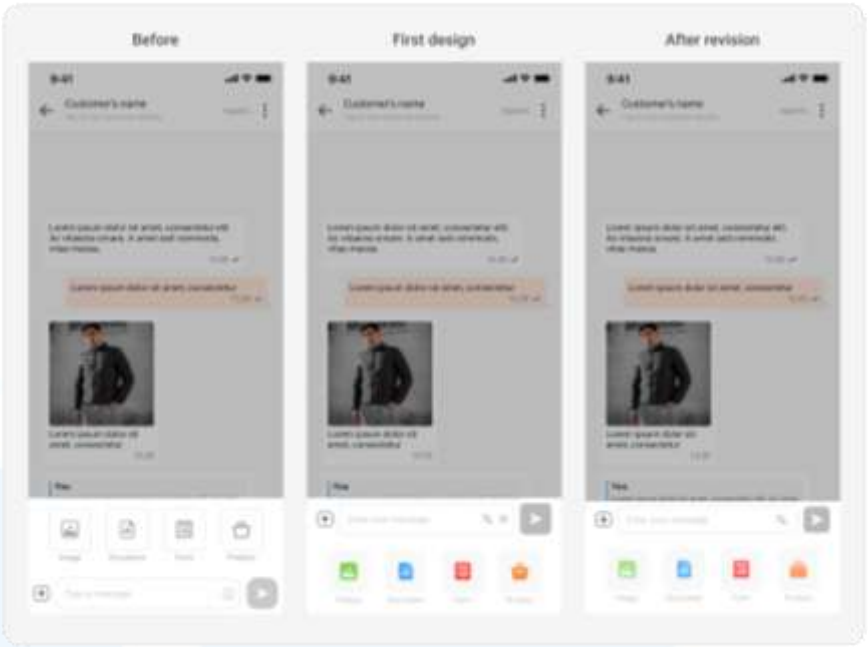
(3) Ketika sedang ditangani oleh *AI Agent.*

Saat *chat* belum ditangani siapa pun, bagian kanan atas berubah menjadi tombol "*Handle ticket*," dan area *input* di bawah juga diganti dengan tombol besar bertuliskan "*Handle ticket."* Ketika *chat* sedang ditangani oleh *Customer Service Agent* lain atau *AI Agent,* tombol tersebut berubah menjadi "*Takeover ticket from <Agent's Name>*" atau "*Takeover ticket from AI Agent*." Pada keadaan ini, *kebab menu* hanya berisi satu opsi "*Take over ticket,*" yang kini juga dilengkapi dengan ikon.



Gambar 3. 25 *Send chat outside 24hour window flow*

Untuk *state* ketika *chat* sudah melewati batas waktu 24 jam, sebelumnya tidak ada penanda khusus sehingga *input field* hanya tidak berfungsi, yang dapat membingungkan pengguna. Oleh karena itu, penulis menambahkan *banner* informasi bertuliskan "*Conversation expired due to 24 hours inactivity. You can start conversation using template message.*" Area *input* di bagian bawah juga diganti menjadi tombol "*Chat with template.*" Hal ini karena *Customer Service Agent* memang tidak dapat mengirim pesan ke pelanggan setelah 24 jam tanpa respons. Saat tombol ditekan, muncul *modal* berisi pilihan *template message,* dilengkapi *banner* agar *Agent* dapat mengirim *chat template* tersebut ke pelanggan.



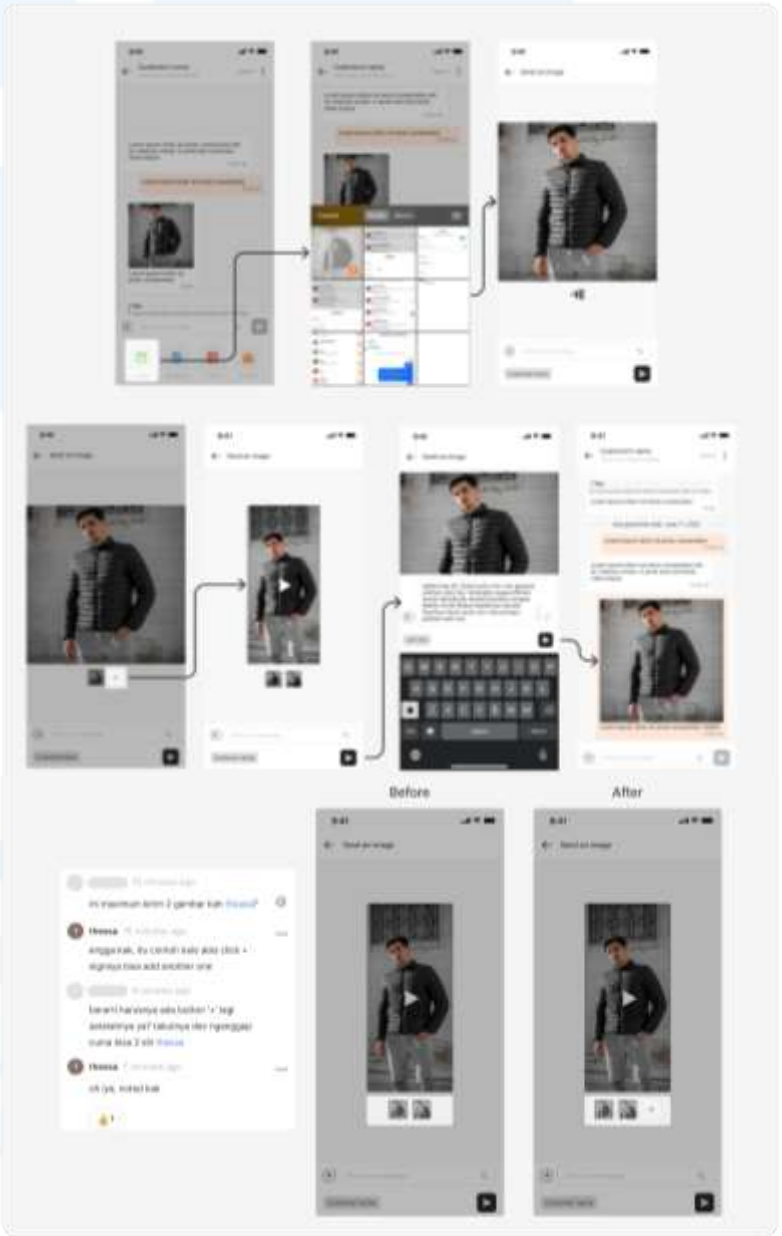
Gambar 3. 26 Proses desain *Attachments footer*

Terakhir, ikon *emoji* di area input diganti menjadi ikon *translate* untuk mengakomodasi fitur baru *Translate Reply Message.* Di sisi kiri terdapat tombol "+" yang, jika ditekan, akan menampilkan empat opsi lampiran: *image, document, form,* dan *product.* Pada desain sebelumnya, opsi lampiran muncul di atas *input field,* namun kini dipindahkan ke bawah agar lebih efisien dan selaras dengan arah interaksi pengguna.

Penulis juga memperbarui ikon-ikon yang digunakan agar sesuai dengan *design system* terbaru, sekaligus memberi warna berbeda pada tiap ikon agar tampilan tidak monoton dan lebih mudah dibedakan. *Supervisor* memberikan sedikit masukan bahwa warna yang digunakan terlalu cerah, sehingga penulis kemudian menyesuaikannya dengan warna yang lebih lembut dan seimbang.

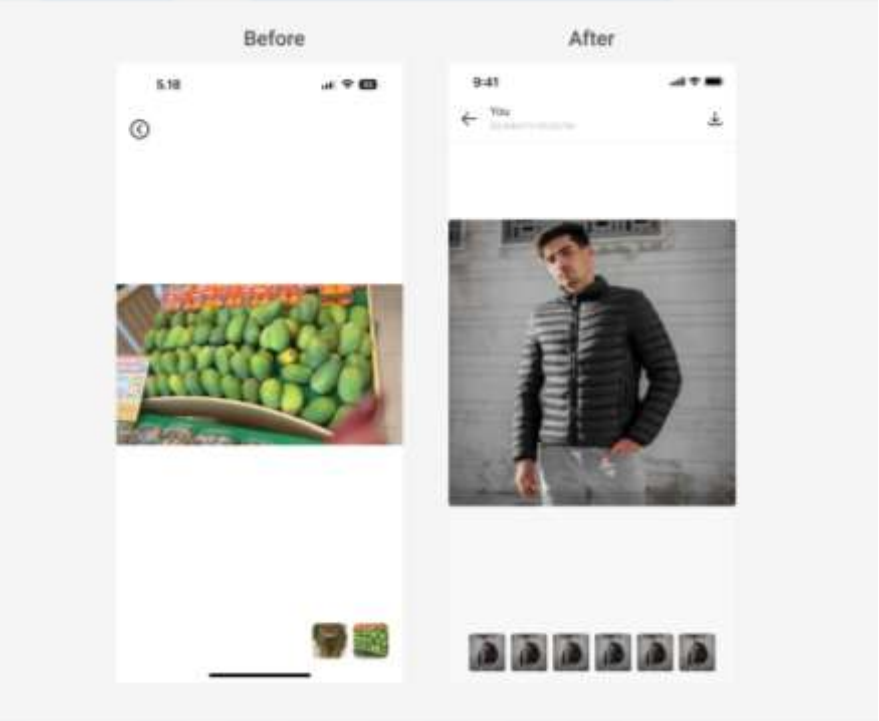
4. *Attachments*

Bagian ini membahas fitur *attachments* yang muncul ketika pengguna mengetuk tombol "+" di sebelah *input message field*. Opsi pertama adalah *image*, di mana pengguna dapat memilih beberapa foto atau video dari galeri perangkat mereka. Pengguna juga dapat menambahkan foto atau video tambahan melalui tombol "+" di sebelah *preview.*



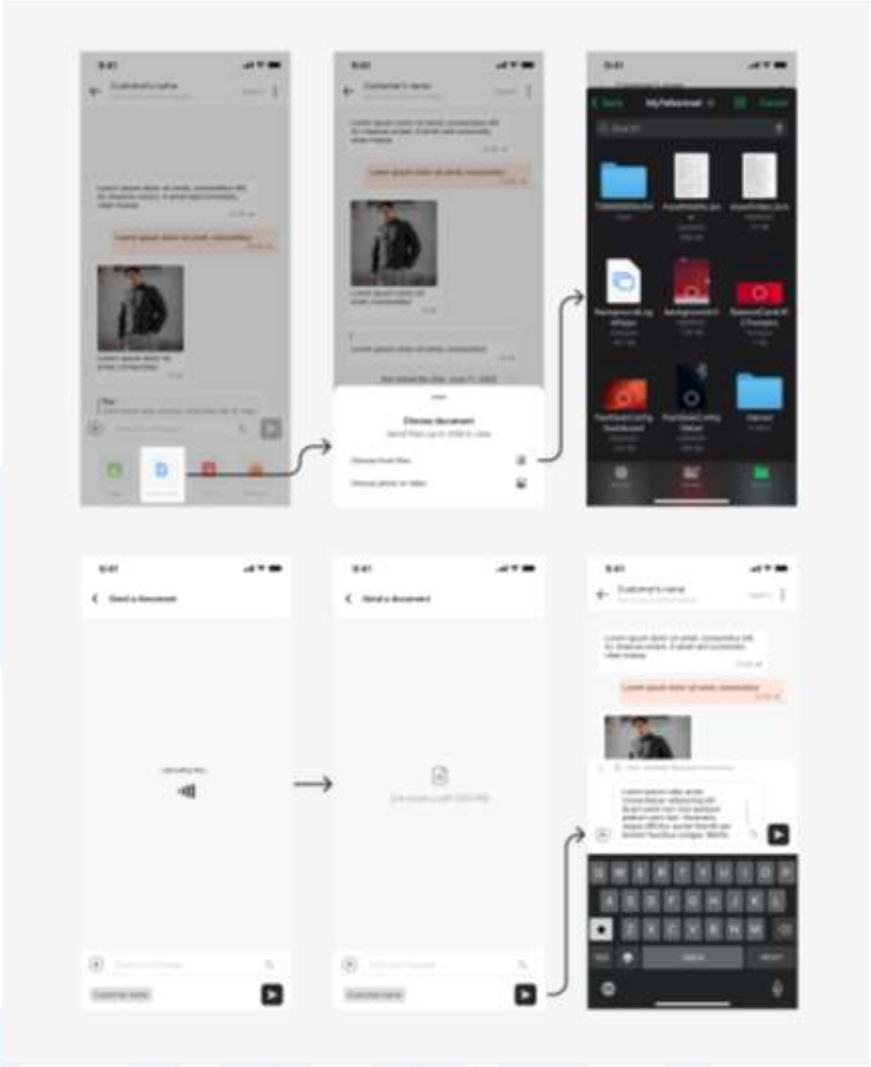
Gambar 3. 27 Desain *attachments*

Dari hasil *review, Supervisor* memberikan masukan terkait tampilan *preview* pengiriman gambar. Pada desain awal, tombol "+" tambahan belum muncul setelah dua gambar ditambahkan, sehingga dapat menimbulkan kesalahpahaman bahwa pengguna hanya dapat mengirim maksimal dua gambar. *Supervisor* menyarankan agar tombol "+" tetap muncul setiap kali pengguna menambahkan gambar agar alur interaksi lebih jelas.



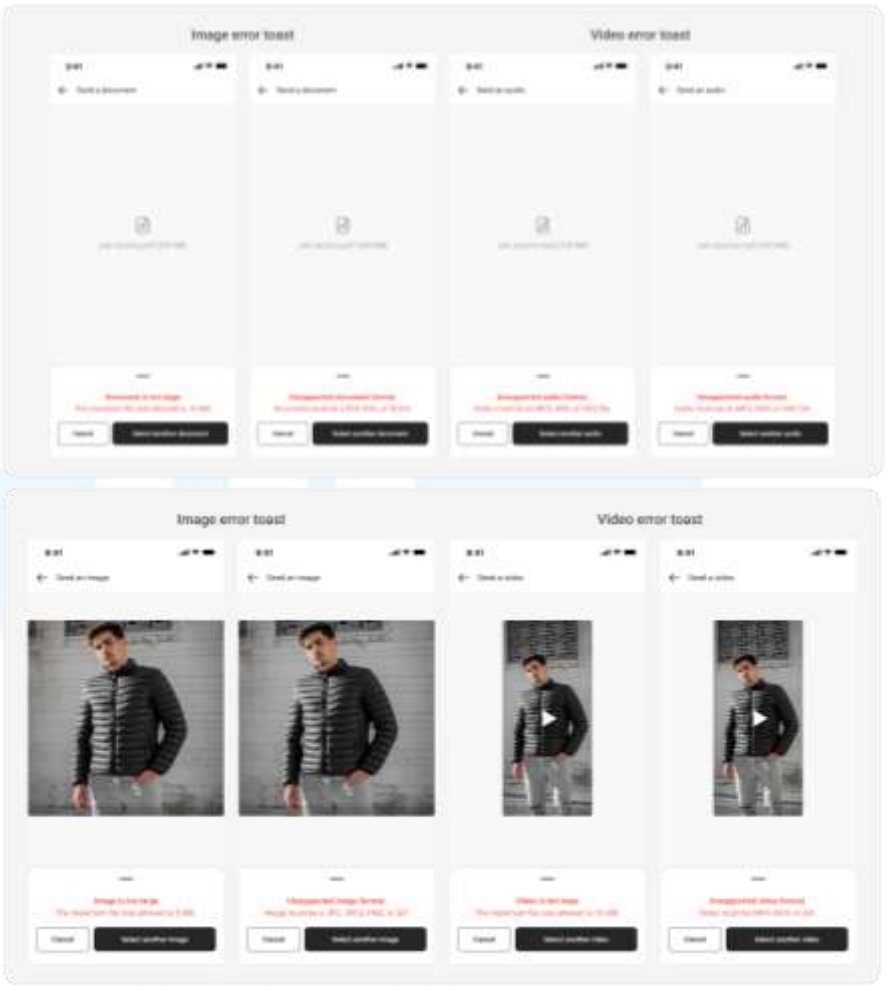
Gambar 3. 28 Desain *Before* dan *After* media

Penulis juga menambahkan fitur baru berupa ikon *download* di pojok kanan atas saat pengguna melihat gambar yang telah dikirim. Fitur ini memungkinkan pengguna untuk mengunduh gambar, serta memperbesar atau memperkecil tampilannya dengan lebih mudah. Bagian *header* pada tampilan media juga diperbarui agar tanggal dan waktu pengiriman gambar muncul sebagai keterangan di bawahnya.



Gambar 3. 29 Desain proses *Attachments documents*

Opsi kedua adalah *document.* Saat pengguna mengetuk ikon *"Documents,"* akan muncul *modal* baru yang menampilkan dua pilihan sumber *file*: dari galeri atau dari *file* perangkat. Setelah pengguna memilih dokumen, nama dan ukuran *file* akan ditampilkan untuk ditinjau sebelum dikirim. Pengguna juga dapat menambahkan pesan opsional bersama dokumen tersebut. Semua berkas dari alur ini dikirim sebagai lampiran dokumen, bahkan jika asalnya dari galeri foto atau video.

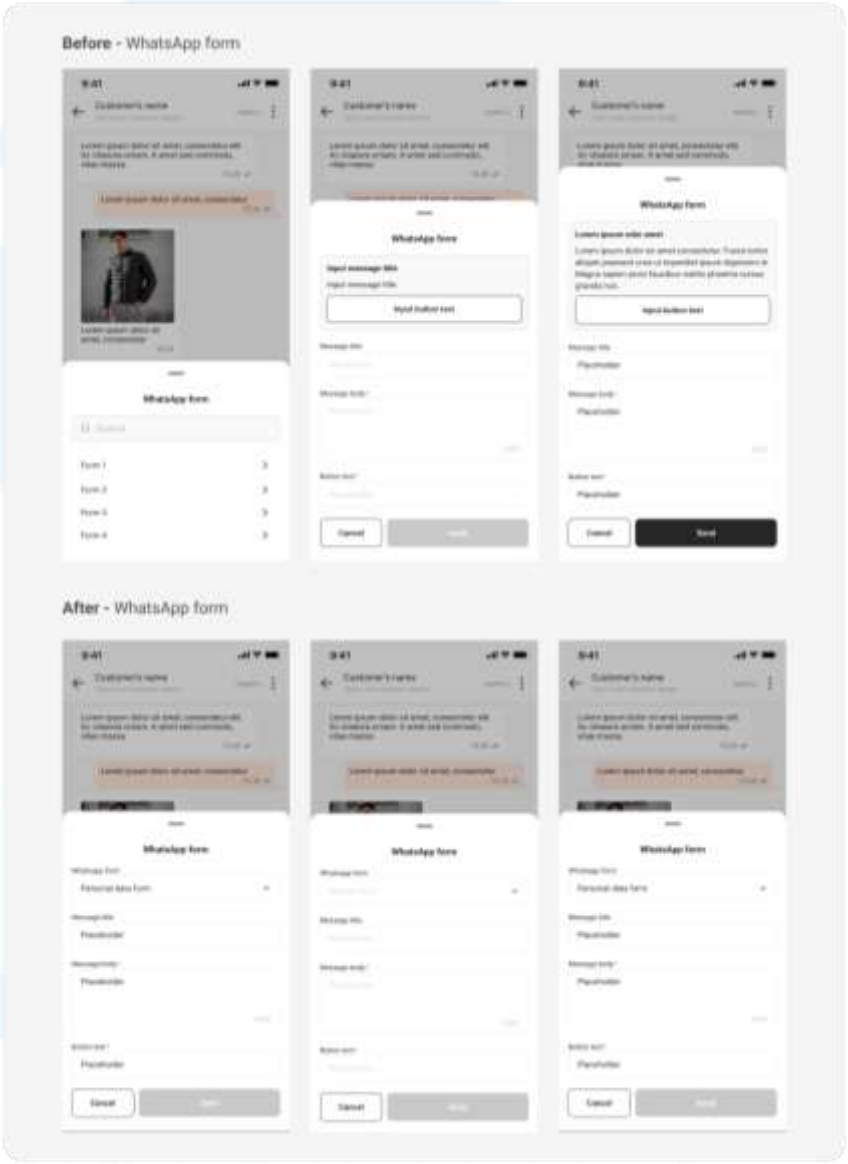


Gambar 3. 30 Desain *error toast* foto dan video

Terakhir, penulis menambahkan *error toast* untuk menampilkan pesan ketika *file* terlalu besar atau memiliki format yang tidak didukung, sehingga pengguna mengetahui alasan kegagalan pengiriman. Sebelumnya *error toast* ini hanya berupa banner kecil di bagian bawah layar. Berdasarkan uji coba penulis, banner ini seringkali ketutupan elemen lainnya. Maka itu, penulis mendesain ulang dengan menggunakan component *modal* dan menambahkan tombol "*Select another image/video"*.

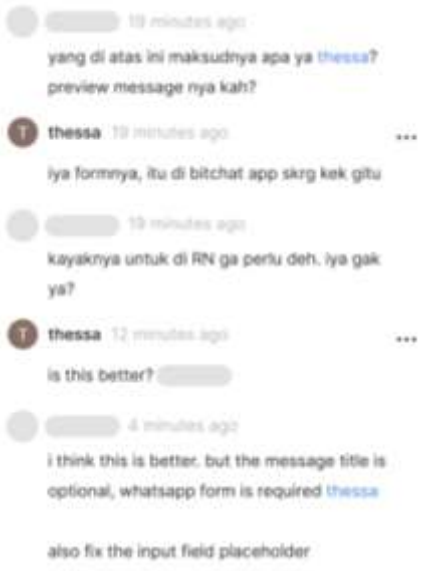
Opsi ketiga adalah WhatsApp *Form***.** Fitur ini memungkinkan *Customer Service Agent* untuk mengirim formulir WhatsApp yang sudah dibuat sebelumnya. Pengguna dapat mencari dan memilih formulir yang tersedia, mengisi kolom yang

diperlukan, serta melihat pratinjau formulir di bagian atas *modal* yang akan diperbarui secara langsung saat mereka mengetik. Setiap kolom wajib diisi memiliki tanda bintang merah (*), dan tombol "*Send*" akan tetap tidak aktif hingga seluruh kolom wajib terisi.



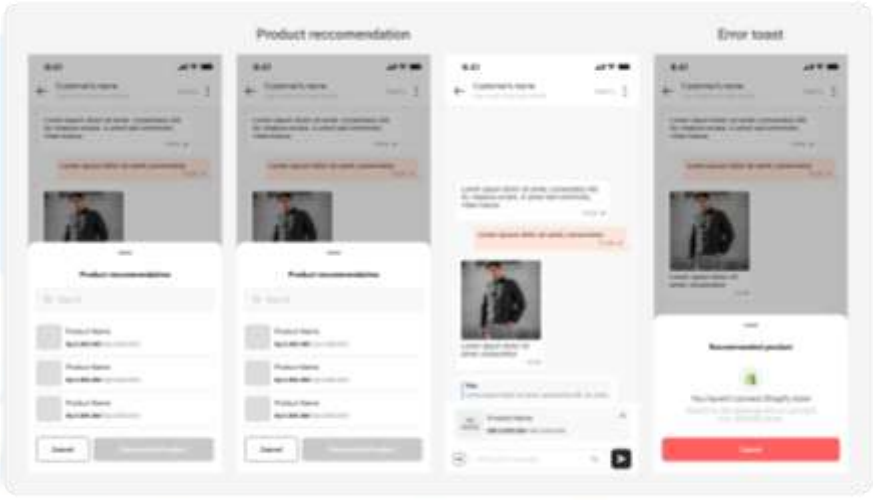
Gambar 3. 31 Desain *Before* dan *After* WhatsApp *form Attachments*

Error message juga muncul tepat di bawah kolom yang bermasalah. Setelah mendapatkan masukan dari *Supervisor,* penulis menghapus fitur *preview form* karena dianggap belum diperlukan di bitChat App. Penulis juga memperbaiki agar kolom judul pesan bersifat opsional serta menyesuaikan *placeholder* agar lebih jelas.



Gambar 3. 32 *Feedback Supervisor* WhatsApp *form*

Opsi terakhir adalah *product.* Dengan adanya layanan *commerce* di bitbybit, pengguna dapat menambahkan katalog produk mereka ke dalam sistem dan mengirim rekomendasi produk langsung kepada pelanggan. Saat pengguna mengetuk ikon *product*, mereka dapat mencari dan memilih produk dari toko yang sudah terhubung, lalu mengirimkannya dengan pesan tambahan opsional.



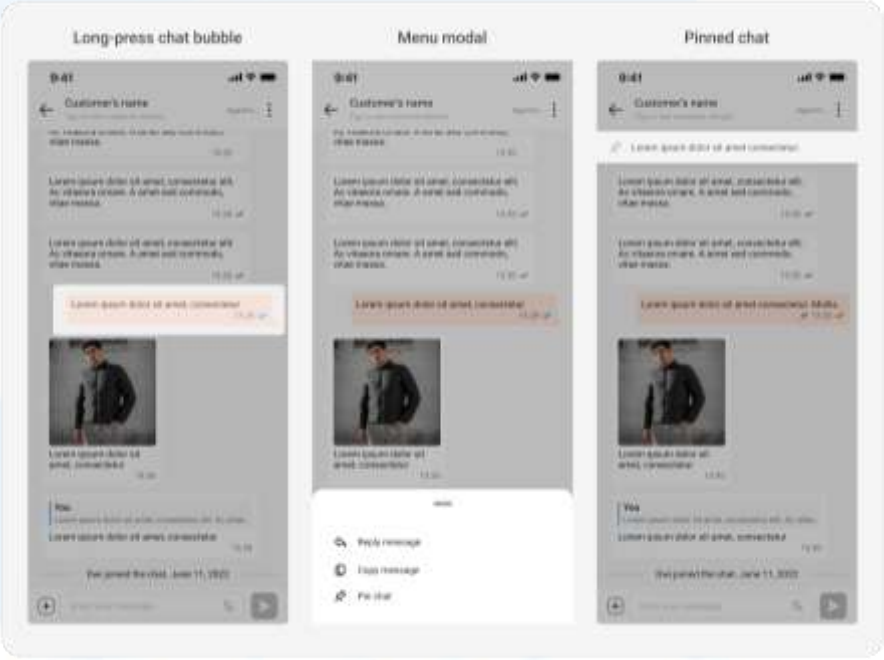
Gambar 3. 33 Desain *Before* dan *After* WhatsApp form

Tombol "*Send"* hanya aktif ketika setidaknya satu produk telah dipilih. Fitur ini juga akan memeriksa apakah toko pengguna sudah terhubung dengan platform *e-commerce* seperti Shopify

sebelum digunakan. Pesan kesalahan ini merupakan *modal* baru yang ditambahkan oleh penulis agar pengguna lebih mudah memahami penyebab *error.*

5. *Pin chat feature*

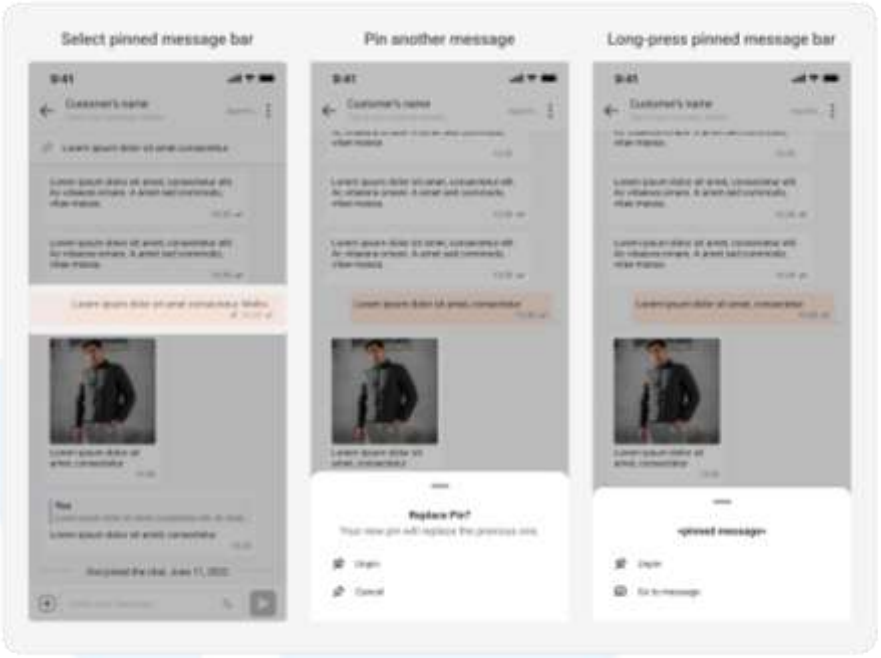
Pada bagian ini, penulis mengembangkan fitur *Pin Chat*, yang memungkinkan pengguna untuk menandai pesan penting agar mudah diakses kembali. Sebelumnya, fitur ini juga merupakan tugas pertama penulis saat awal magang di *bitChat* versi *desktop*. Oleh karena itu, proses pengerjaan pada versi ini berlangsung lebih cepat karena penulis sudah memahami kebutuhan sistem serta kesalahan yang sempat terjadi pada iterasi sebelumnya.



Gambar 3. 34 Proses desain *Pin message feature* 1

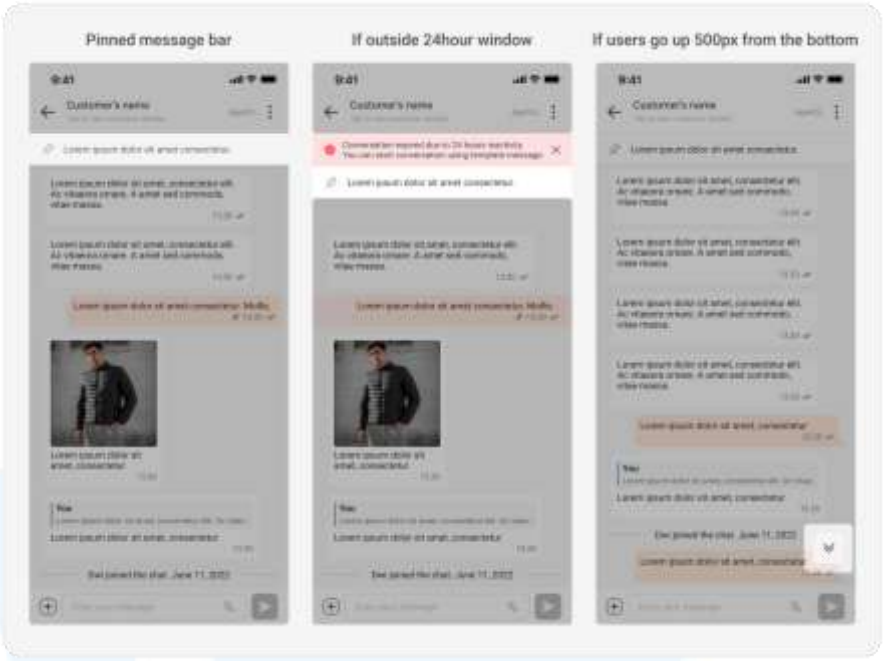
Fitur ini dikembangkan sebagai perluasan dari elemen yang telah dijelaskan pada bagian *Chat Interface*. Saat pengguna melakukan *long-press* pada *chat bubble*, akan muncul opsi baru bertuliskan *"Pin Chat."* Ketika opsi tersebut dipilih, pesan yang ditekan akan muncul di bawah *header* dalam bentuk *pinned message bar*. Jika pengguna menekan *bar* tersebut, tampilan percakapan akan

otomatis bergulir ke pesan yang telah di-*pin* dan menyorot pesan tersebut (*highlighted message*) agar mudah dikenali.



Gambar 3. 35 Proses desain *Pin message feature* 2

Selain itu, ketika pengguna melakukan *long-press* pada *pinned message bar*, akan muncul *modal* dengan judul yang menampilkan isi pesan yang dipasangkan. Dalam *modal* ini terdapat dua opsi, yaitu *"Unpin message"* dan *"Go to message."* Menambahkan interaksi baru berupa tombol dengan ikon dua *chevron* mengarah ke bawah yang muncul ketika pengguna menggulir layar ke atas sejauh jarak tertentu. Ketika tombol ini ditekan, tampilan akan langsung berpindah ke bagian paling bawah dari ruang percakapan. Apabila pengguna mencoba untuk men-*pin* pesan baru ketika sudah ada pesan yang di-*pin*, aka akan muncul *modal* dengan pesan konfirmasi "*Replace Pin? Your new pin will replace the previous one*" serta dua opsi: *Unpin* atau *Cancel*.

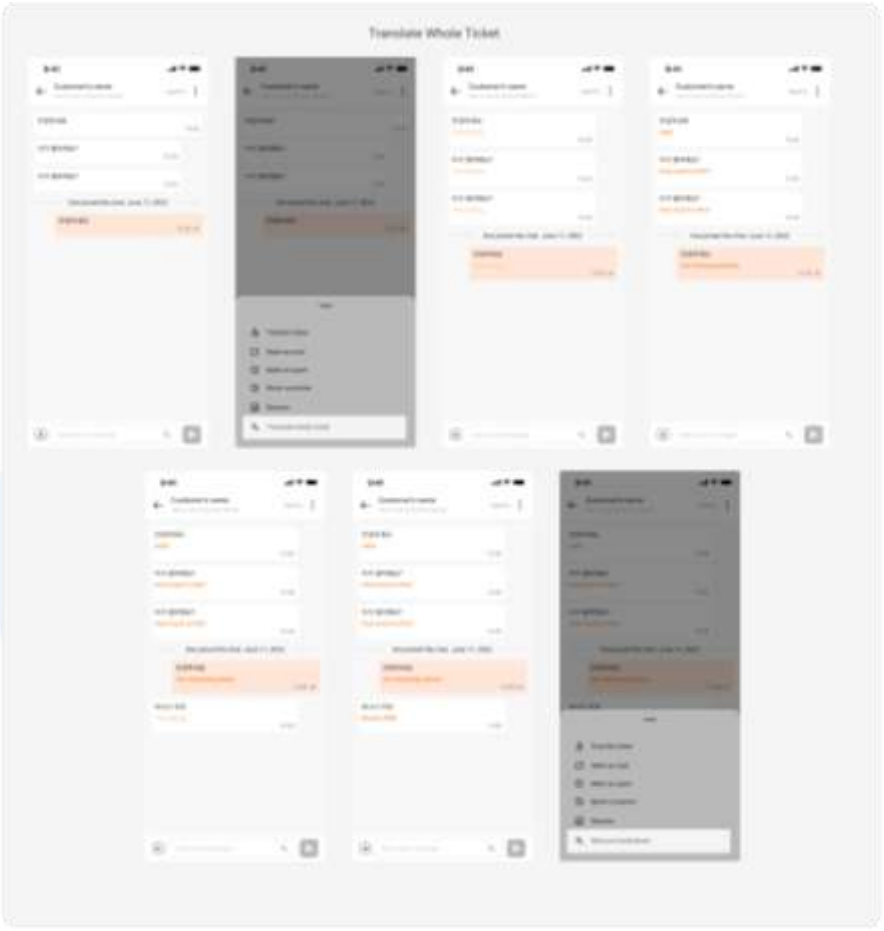


Gambar 3. 36 Proses desain *Pin message feature* 3

Untuk kondisi khusus, jika *Customer Service Agent* sebelumnya telah men-*pin* pesan dan ruang percakapan tersebut sudah melewati batas waktu 24 jam, maka *pinned message bar* akan muncul di bawah *warning banner*. Penempatan ini dipilih agar peringatan tetap terlihat jelas oleh pengguna tanpa mengganggu konteks pesan yang di-*pin.* Fitur ini telah mendapat *feedback* dari *Supervisor*, yang menyatakan bahwa hasil desain sudah sesuai dengan kebutuhan dan tidak memerlukan revisi lebih lanjut.

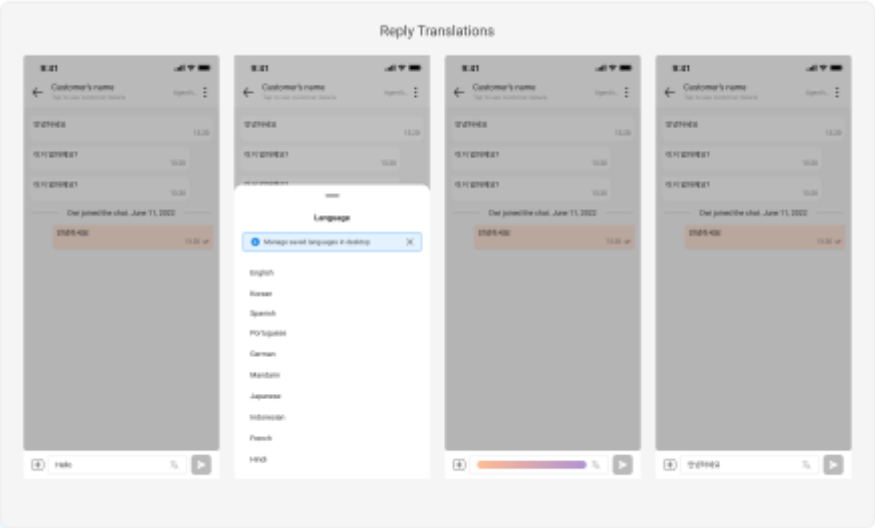
6. *Translate whole ticket feature*

Bagian ini membuat fitur *Translate Whole Ticket*, yaitu fungsi untuk menerjemahkan seluruh isi percakapan antara *customer* dan *customer service agent* secara otomatis. Fitur ini dikembangkan untuk mendukung kebutuhan tim internal *Business Development*, mengingat klien bitbybit berasal dari berbagai negara dengan bahasa yang berbeda. Penulis sebelumnya juga telah mengembangkan fitur serupa pada bitChat versi *desktop*, sehingga proses pengembangan pada versi *React Native* berlangsung lebih efisien karena telah memahami alur kerja dan kebutuhan sistem.



Gambar 3. 37 Proses desain *Translate Whole Ticket*

Untuk mengakses fitur ini, pengguna dapat membuka *kebab menu modal* dan memilih opsi *"Translate whole ticket."* Setelah diaktifkan, sistem akan menampilkan status *"translating"* hingga proses penerjemahan selesai. Hasil terjemahan kemudian muncul di bawah pesan asli dengan format *italic* dan warna oranye sebagai pembeda visual. Fitur ini juga memungkinkan pesan baru yang masuk untuk diterjemahkan secara otomatis. Ketika seluruh pesan telah diterjemahkan, opsi pada *menu modal* akan berubah menjadi *"Remove Translation"* agar pengguna dapat mengembalikan tampilan ke kondisi awal.



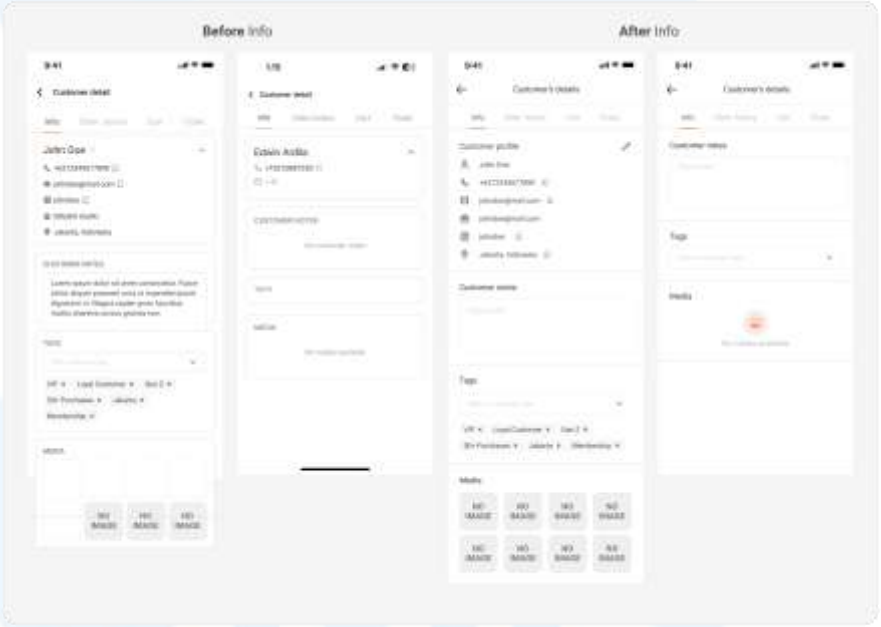
Gambar 3. 38 Proses desain *Reply Translate*

Selain itu, fitur *Reply Translation* yang memungkinkan *customer service agent* membalas pesan pelanggan dalam bahasa asli pelanggan tersebut. Pada bagian *input field*, ikon emoji diganti menjadi ikon terjemahan yang berfungsi sebagai *trigger* untuk mengaktifkan fitur ini. Ketika ikon tersebut diketuk, akan muncul *modal* yang menampilkan daftar bahasa yang tersedia untuk diterjemahkan. Pada versi *desktop*, pengguna dapat memilih bahasa dari daftar lengkap serta menyimpan beberapa bahasa favorit untuk akses yang lebih cepat.

Namun, pada versi *React Native*, pilihan bahasa yang muncul terbatas pada bahasa yang telah disimpan sebelumnya melalui versi *desktop*. Oleh karena itu, dalam *modal* terdapat *info banner* bertuliskan *"Manage saved languages in desktop."* Setelah pengguna memilih bahasa tujuan, sistem akan menampilkan *loading state* dengan animasi gradasi oranye–ungu, mengikuti panduan warna AI pada bitbybit *design system.* Setelah proses penerjemahan selesai, teks pada *input field* akan otomatis diubah ke bahasa yang dipilih sebelum dikirimkan kepada pelanggan.

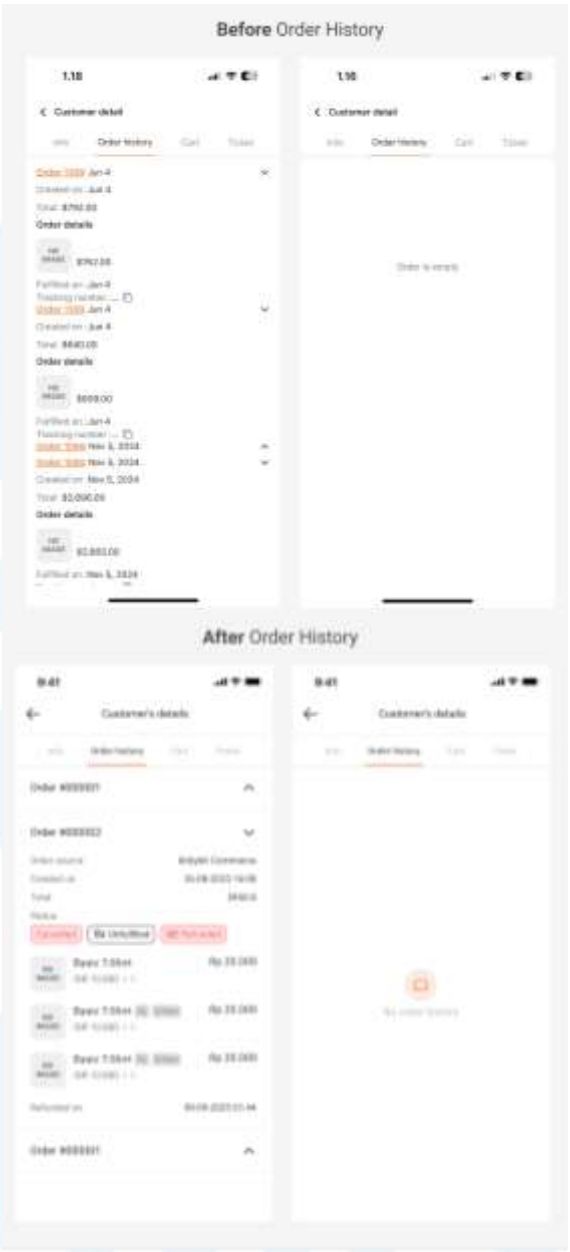
7. *Customer details*

Fitur *Customer Details Section* dapat diakses dengan menekan profil pelanggan pada aplikasi. Fitur ini menampilkan informasi detail mengenai pelanggan dan terbagi menjadi empat bagian utama: *Info, Order History, Cart,* dan *Ticket.* Sebelumnya, tampilan *Customer Details* masih menggunakan komponen dari versi *desktop*, sehingga belum memiliki komponen yang dirancang khusus untuk *React Native.* Berdasarkan masukan dari *Supervisor* dan tim *UI/UX Designer* lainnya, bagian ini perlu dirombak secara menyeluruh dengan memanfaatkan komponen dari *Commerce RN* agar lebih konsisten dan efisien.



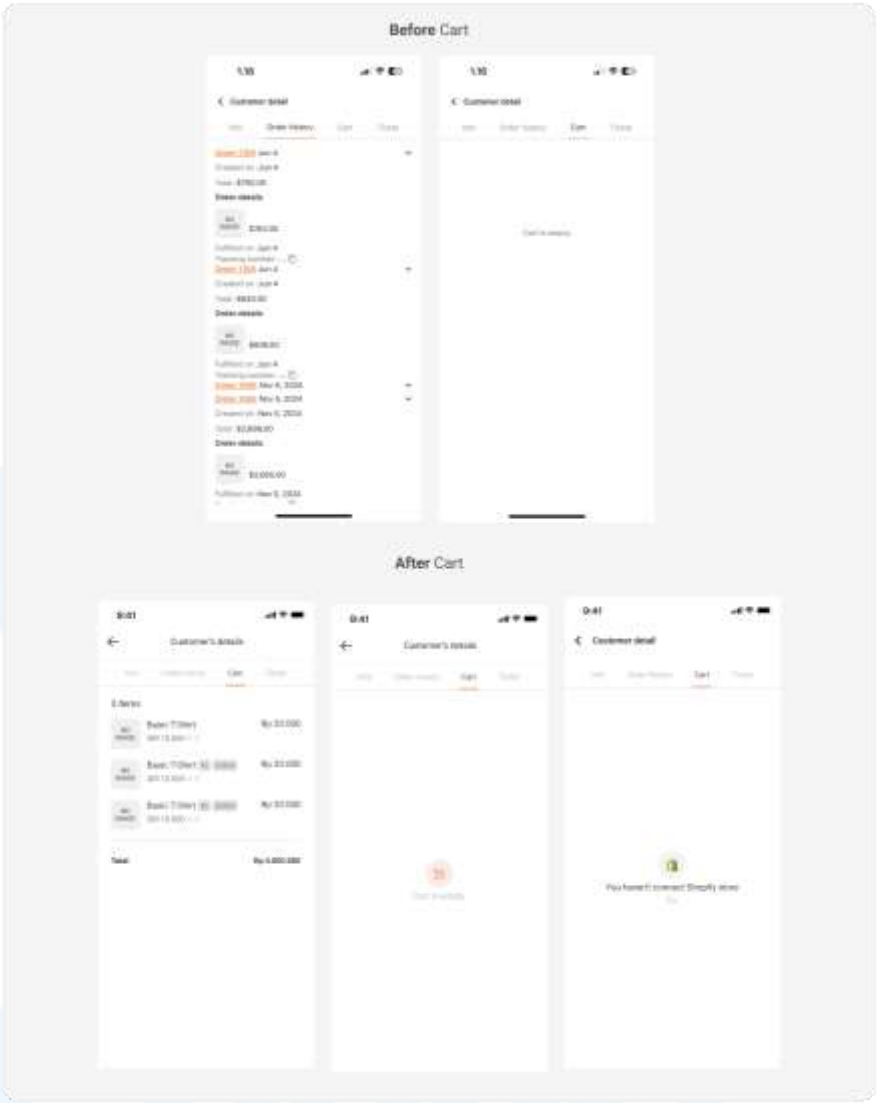
Gambar 3. 39 Proses desain *Customer Details*

Pada bagian *Info,* penulis melakukan penyesuaian tata letak agar tampil lebih rapi dan mudah dibaca. Seluruh ikon diperbarui menjadi versi *lined* (tidak terisi) untuk menyesuaikan dengan gaya visual baru. Ikon *copy* juga diperbesar agar memiliki *hitbox* yang lebih luas sehingga lebih mudah ditekan. Selain itu, *empty state* diperbarui dengan ikon berwarna oranye untuk menyamakan dengan sistem warna bitbybit.



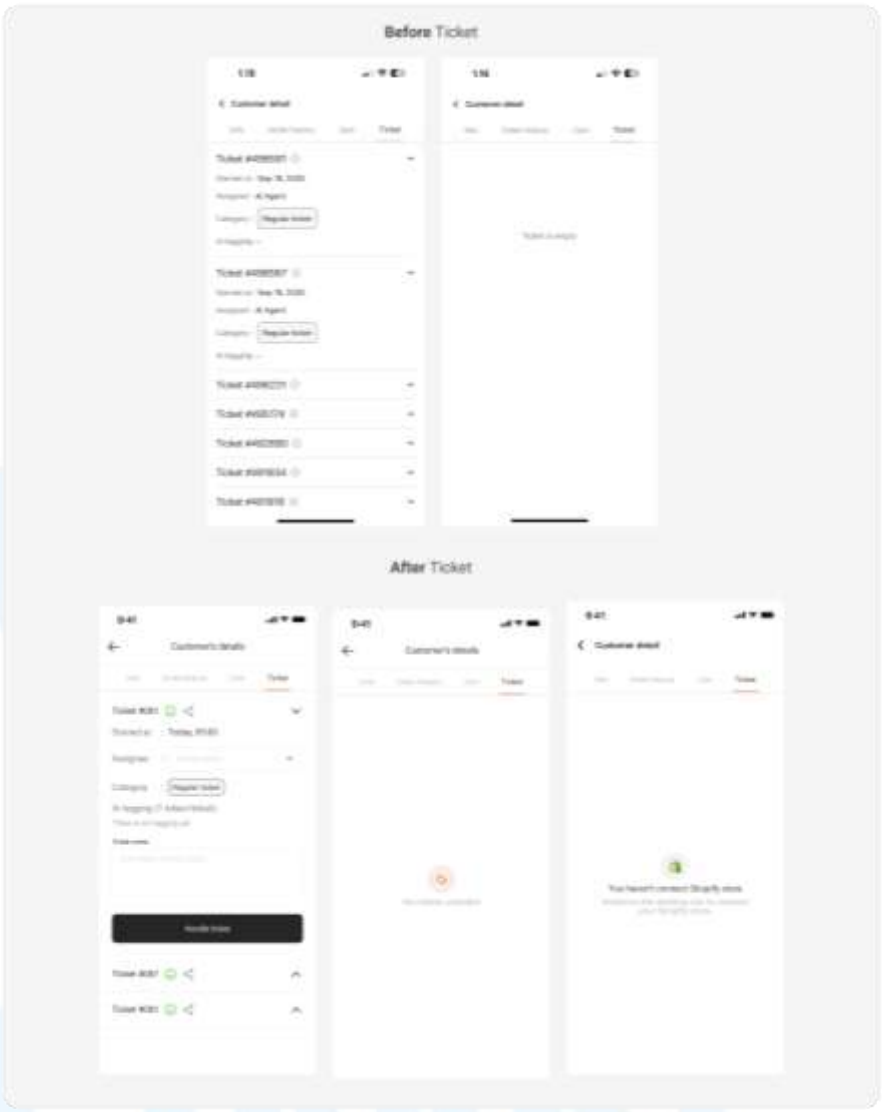
Gambar 3. 40 Proses desain *Customer Details* 2

Bagian *Order History* diadaptasi dari desain *Commerce RN* dan dimodifikasi agar sesuai dengan tampilan profil pelanggan. Penulis menambahkan versi *collapsed* untuk menampilkan daftar pesanan secara ringkas, serta memperbaiki jarak antar elemen seperti *padding* dan *spacing* yang sebelumnya kurang rapi. *Empty state* juga diperbarui menjadi teks *"No order history"* dengan ikon berwarna oranye agar konsisten dengan gaya visual keseluruhan.



Gambar 3. 41 Proses desain *Customer Details* 3

Sama seperti *Order History*, bagian *Cart* juga memanfaatkan desain dari *Commerce RN* dan disesuaikan dengan konteks *Customer Profile.* Berdasarkan saran dari *Supervisor*, penulis menambahkan fungsi agar total belanja tetap *stick* di bagian bawah layar saat isi keranjang terlalu panjang (*overflow*). *Empty state* diperbarui menjadi teks *"Cart is empty"* dengan ikon berwarna oranye. Selain itu, apabila pengguna belum menghubungkan toko *Shopify*-nya, akan muncul peringatan *"You haven't connected a Shopify store. Switch to the desktop site to connect your Shopify store."* beserta logo *Shopify* untuk memperjelas konteks.



Gambar 3. 42 Proses desain *Customer Details* 4

Terakhir, untuk bagian *Ticket*, penulis menggunakan komponen versi *desktop* yang sudah di sesuaikan untuk *mobile app.* Sama seperti sebelumnya, penulis menambahkan ikon dan memperbarui *copywriting* untuk *empty* dan *not connected state*. Apabila pengguna belum menghubungkan toko *Shopify*-nya, akan muncul peringatan *"You haven't connected a Shopify store. Switch to the desktop site to connect your Shopify store."* beserta logo *Shopify* untuk memperjelas konteks. Ini merupakan *User Interface improvement* baru ditambahkan penulis.

D. *Prototype*

Tahap *prototyping* pada perancangan ini tidak dilakukan secara manual menggunakan Figma seperti pada proses desain awal, melainkan melalui Figma Make. Pemilihan platform ini dilakukan karena Figma Make memungkinkan hasil desain langsung menghasilkan *code output* yang digunakan oleh tim *Software Developer*. Dengan demikian, proses implementasi desain ke dalam produk aktual menjadi lebih efisien, mengurangi risiko perbedaan hasil.

Namun, penggunaan Figma Make hanya diterapkan untuk fitur tertentu yang memiliki alur interaksi lebih kompleks atau tampilan yang membutuhkan ketelitian tinggi, seperti *Translate Whole Ticket* dan *Pin Chat Feature*. Hal ini dilakukan agar hasil visual dan interaksi pada fitur-fitur tersebut dapat diimplementasikan secara presisi sesuai rancangan, tanpa terlewat oleh tim *Software Developer*.

Untuk fitur lain dengan kompleksitas rendah hingga sedang, *prototype* disusun dalam bentuk *flow diagram* di Figma. Setiap layar dijelaskan melalui panah dan anotasi yang menggambarkan alur interaksi. Tahap ini juga berkaitan langsung dengan proses *handover,* yaitu transisi dari tim *Product* ke tim *Software Developer* sebagai langkah menuju tahap *testing.* Penulis bertanggung jawab memastikan seluruh rancangan dan alur interaksi dapat diimplementasikan secara tepat melalui *handover file*. Berikut adalah proses pembuatan *handover file*:

1. *User Interface* untuk *Homepage*

   Penulis menjelaskan perubahan antara tampilan *before* dan *after* pada bagian Home UI. Pembaruan mencakup *header,* penambahan *chip filter* dengan kondisi *default* dan *selected,* serta penyusunan ulang *layout* dengan tambahan *new priority badge.* Pada bagian ini penulis juga menjabarkan komponen yang baru serta variasi *state* seperti *default* dan *selected,*

Gambar 3. 43 *Prototype Home UI 1*

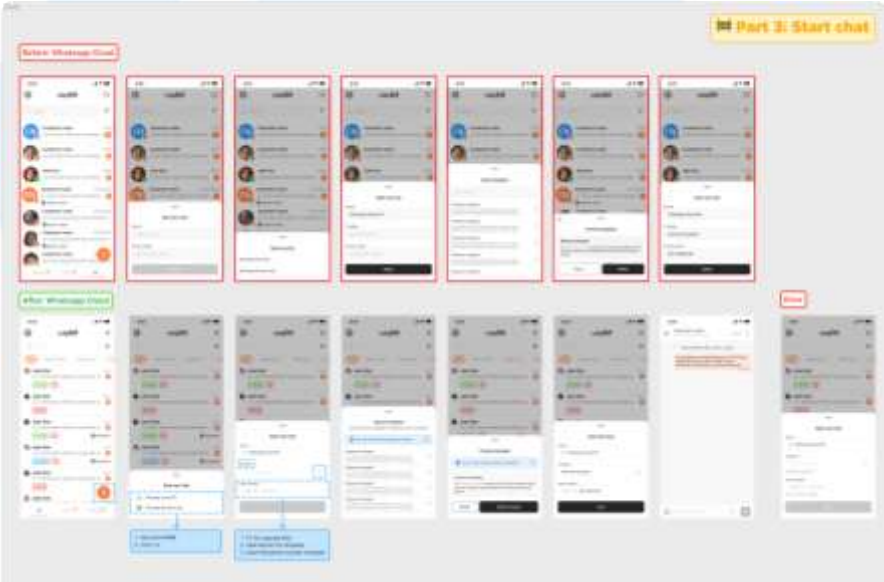
Untuk bagian *Chat Details,* penulis menggunakan dokumentasi dari rekan *UI/UX intern* lain yang mengerjakan komponen ini pada versi *desktop.* Pada bagian *Filter Modal,* penulis menampilkan perbandingan antara tampilan *before* dan *after,* serta menjelaskan kondisi *default* dan *applied state* beserta perubahan pada *button state* ketika dalam kondisi *default* atau *selected*.

Gambar 3. 44 *Prototype Home UI 2*

Penulis juga menampilkan pembaruan pada *empty state* dengan ikon baru, serta menjabarkan seluruh kondisi yang mungkin terjadi: *All, Yours Waiting, Empty,* dan *Not Connected.* Selain itu, dijelaskan pula alur ketika pengguna melakukan *search,* termasuk kondisi ketika hasil pencarian tidak ditemukan, pengguna menerapkan filter, atau terjadi *error state.*

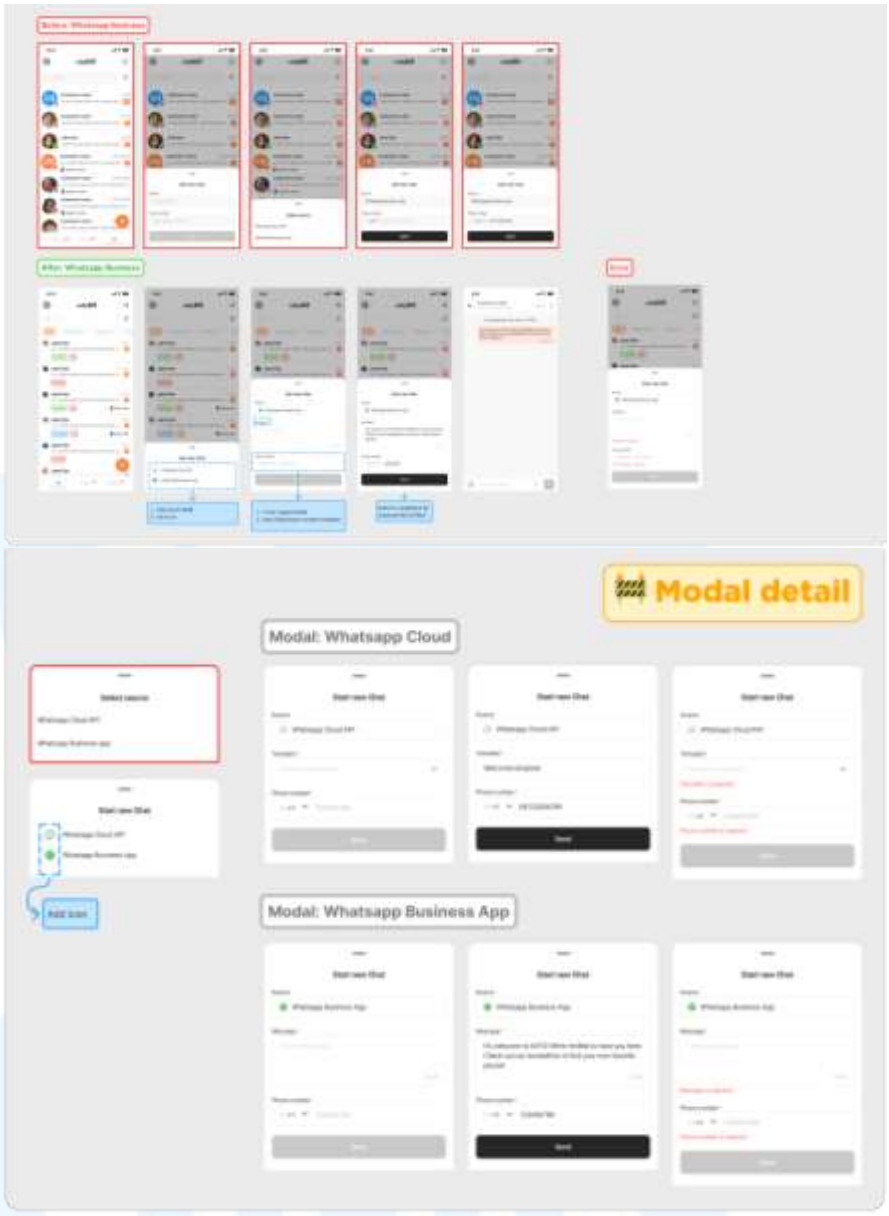
2. *User Interface* untuk alur *Start Chat*

Pada bagian *Start Chat Flow*, penulis menampilkan perbandingan *before* dan *after* dalam *handover file* untuk memperjelas perubahan yang dilakukan pada alur memulai percakapan baru. Tujuannya adalah agar tim *Software Developer* dapat dengan mudah mengidentifikasi perbedaan elemen yang perlu diperbarui dalam implementasi.



Gambar 3. 45 *Prototype Start Chat Flow* 1

Beberapa pembaruan utama meliputi penambahan langkah awal agar sistem terlebih dahulu menanyakan sumber tiket sebelum pengguna dapat memulai percakapan. Setiap bagian *input* kini dilengkapi ikon untuk memperjelas fungsi masing-masing kolom. Tanda bintang ditambahkan pada *input field* wajib untuk meningkatkan kejelasan. Ikon *chevron* ditambahkan pada *template selector* untuk menandakan bahwa elemen tersebut dapat diklik dan menampilkan daftar pilihan. Selain itu, *input field* nomor telepon kini menggunakan format *template* untuk mempermudah *input* dan mencegah kesalahan format.

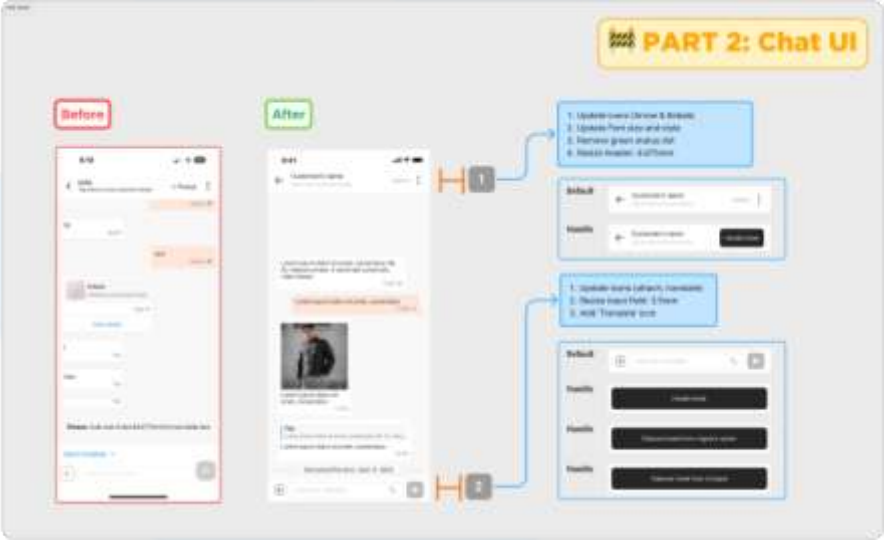


Gambar 3. 46 *Prototype Start Chat Flow 2*

Selain perubahan elemen utama, disertakan juga *breakdown modal* secara detail untuk membantu tim *Software Developer* memahami spesifikasi teknis komponen. Pada *modal* tersebut, jarak antar elemen diatur dengan *margin* sebesar 1rem dan *spacing* sebesar 1.25rem agar tampilan terasa seimbang serta sesuai dengan pedoman *design system*.

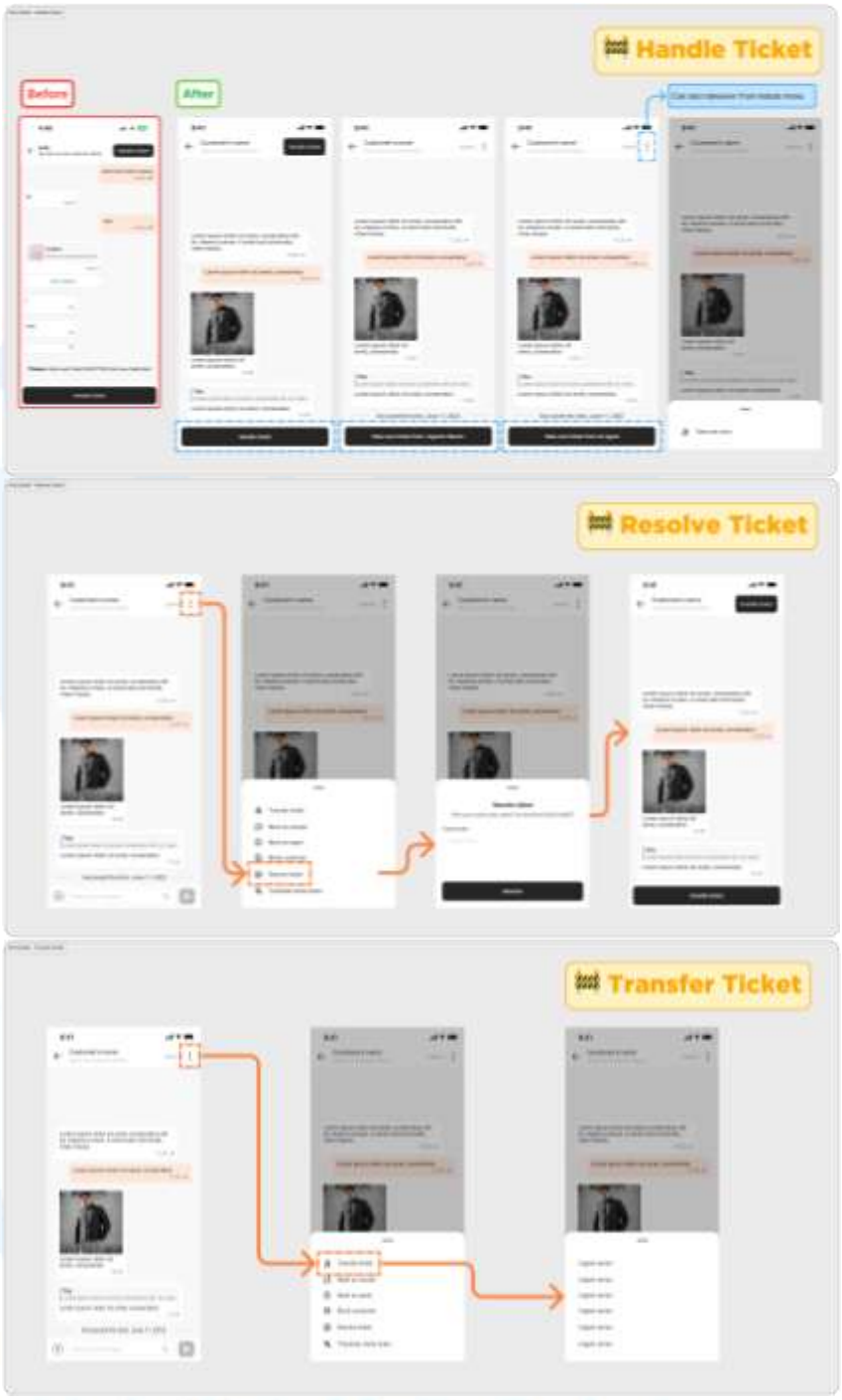
3. *User Interface* untuk halaman *Chat*

Pada bagian *handover file* untuk *Chat User Interface***,** penulis menyusun dokumentasi menampilkan perbandingan *before* dan *after* secara detail agar tim *Software Developer* dapat memahami perubahan elemen dengan lebih jelas.



Gambar 3. 47 *Prototype Chat UI 1*

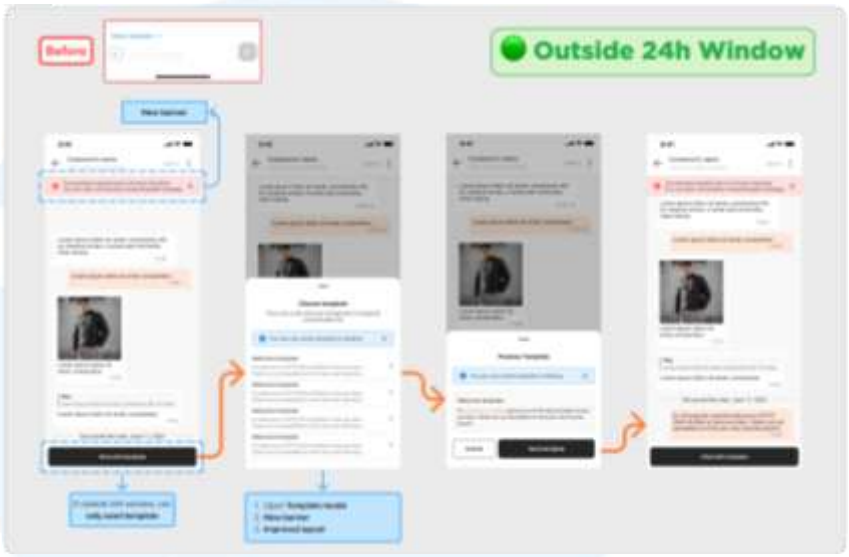
Pada bagian *header*, dilakukan beberapa pembaruan yaitu ikon "*panah"* dan "*kebab menu"*, pembaruan *font size* dan *font style*, penghapusan *green status dot*, serta *resize header* menjadi 3.875rem agar proporsional dengan elemen lainnya. Sementara pada bagian *footer*, dilakukan pembaruan ikon "*attach"* dan "*translate"*, dan *resize input field* menjadi 3.5rem.



Gambar 3. 48 *Prototype Chat UI 2*

Selain pembaruan tampilan, alur juga disusun untuk membantu tim *Software Developer* memahami interaksi pengguna di dalam fitur ini, antara lain: *Flow "Handle Ticket”*, menggambarkan proses agen ketika mulai menangani tiket baru;

Flow "Resolve Ticket", menjelaskan alur ketika tiket ditandai selesai oleh agen; *Flow "Transfer Ticket",* memperlihatkan proses pemindahan tiket ke agen atau tim lain.



Gambar 3. 49 *Prototype Chat UI 3*

Flow "Outside 24 Hour Window", menunjukkan kondisi ketika agen mencoba membalas pesan yang sudah melewati batas waktu 24 jam. Ketika kondisi ini terjadi, agen tidak bisa menghubungkan pelanggan. Penulis menambahkan indikator berupa *warning banner* dan tombol di paling bawah agar agen bisa komunikasi dengan pelanggan menggunakan *template.*

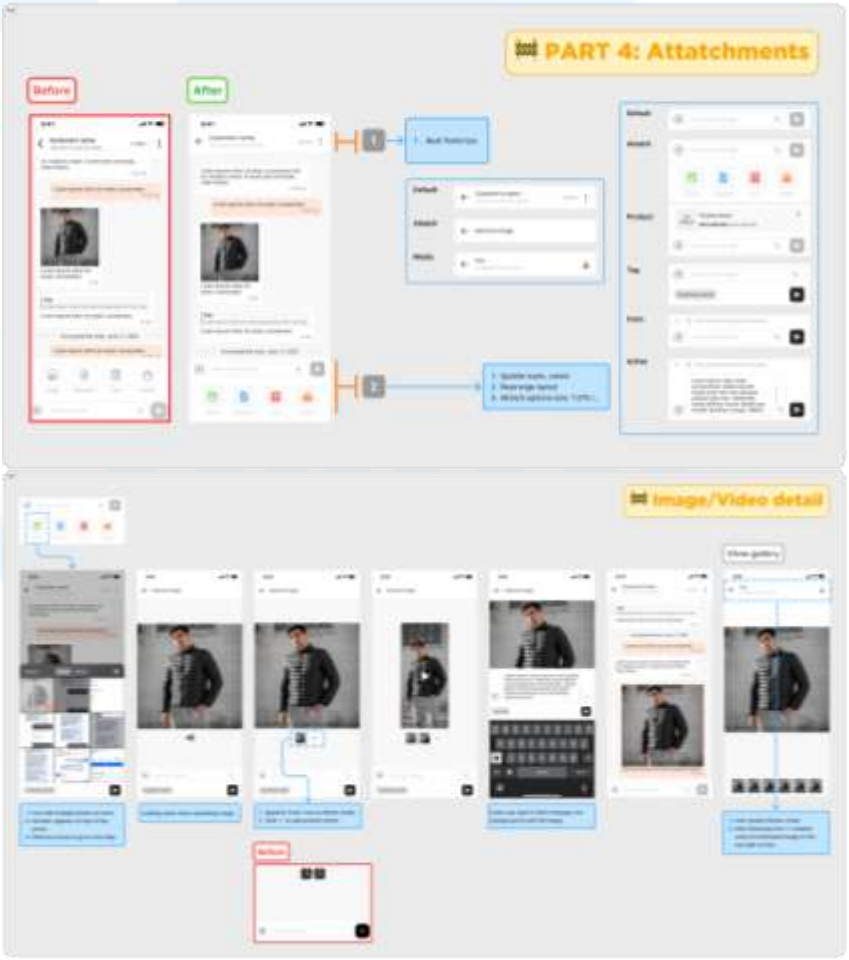
Gambar 3. 50 *Feedback Prototype Chat UI*

Berdasarkan masukan dari *Supervisor*, pada tampilan ini perlu diberikan penekanan tambahan tidak hanya pada *highlight*

banner, tetapi juga pada tombol *"Chat with Templat"* agar tim *Software Developer* memahami perilaku fitur tersebut dengan jelas. Selain itu, posisi tombol ini juga perlu disesuaikan karena terlihat terlalu menempel ke bagian bawah layar.

4. *User Interface* untuk fitur *Attachments*

Penulis menampilkan tampilan *before* dan *after* agar tim *Software Developer* dapat memahami perubahan secara menyeluruh. Pada bagian *header*, dilakukan beberapa pembaruan seperti mengubah untuk menjaga konsistensi, memperbarui teks menjadi "*Send an image/file/audio*" agar lebih informatif, serta menambahkan *media header* baru yang menampilkan tanggal dan ikon *download*.



Gambar 3. 51 *Prototype Attachments 1*

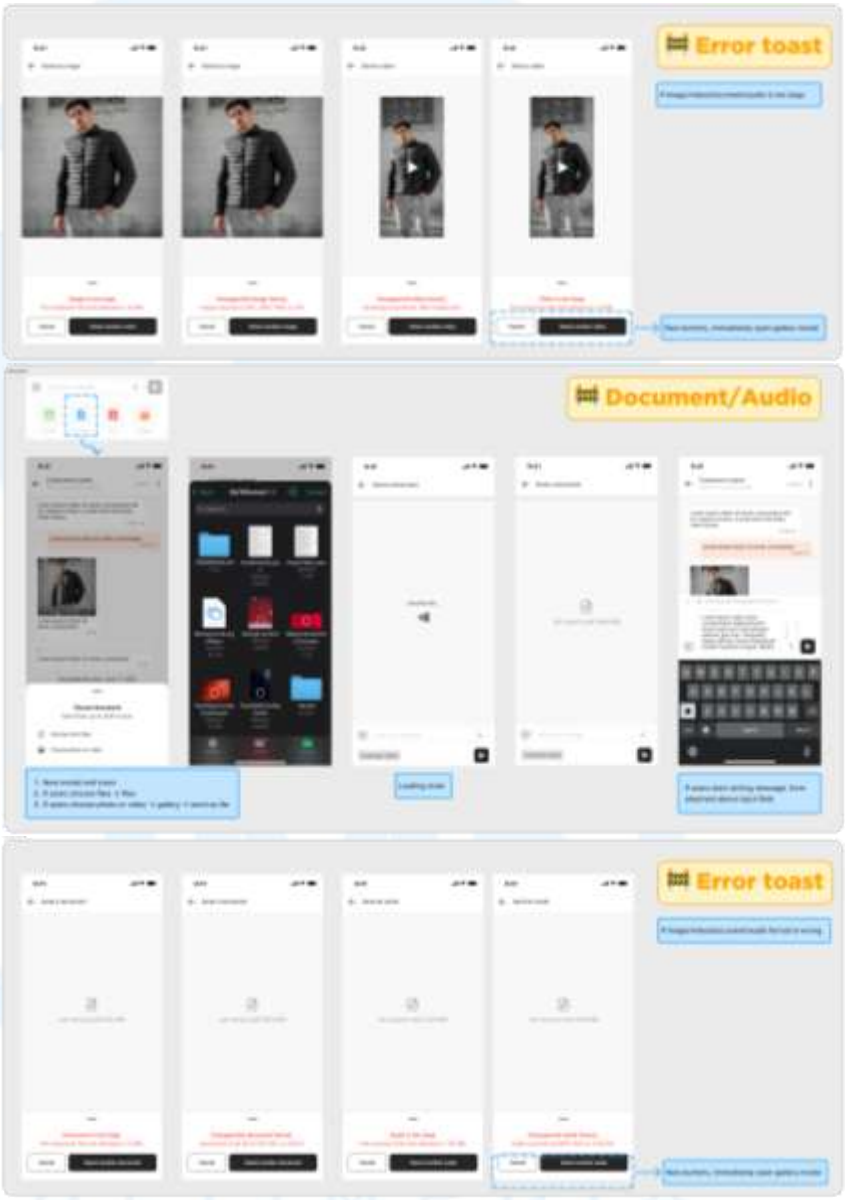
Pada bagian *footer*, dilakukan pembaruan ikon dan warna agar selaras dengan gaya baru, pengaturan ulang tata letak, serta penyesuaian ukuran. Selain itu, seluruh kemungkinan *footer states* dijabarkan dalam *handover* agar tim *Software Developer* dapat memahami setiap kondisi visual yang mungkin terjadi.

Untuk proses *attaching image* (foto atau video), penulis menjabarkan alur pengguna secara lengkap. Ketika pengguna menekan tombol "*Attach Image*," sistem akan membuka *modal gallery* dari ponsel. Setelah menekan ikon panah untuk melanjutkan, sistem menampilkan *loading state* sebelum media berhasil diunggah. Setelah proses unggah selesai, media muncul sebagai *mini preview* di bawah area *input* dengan ikon *trash* untuk menghapus media. Pengguna dapat menambahkan foto lain dengan menekan tombol "+" dan tetap dapat mengetik pesan meskipun terdapat *preview* gambar di latar belakang. Setelah terkirim, media akan tampil di *chat bubble* dengan *header* baru berisi nama pengirim, tanggal, dan waktu pengiriman, serta ikon *download* di kanan atas.

Gambar 3. 52 *Feedback Prototype Attachments*

Supervisor memberikan *feedback* dengan menanyakan fungsi *loading state*, sehingga penulis menambahkan keterangan tambahan agar tim *Software Developer* memahami konteksnya. Selain itu, penulis juga memperbarui pesan *error toast* seperti

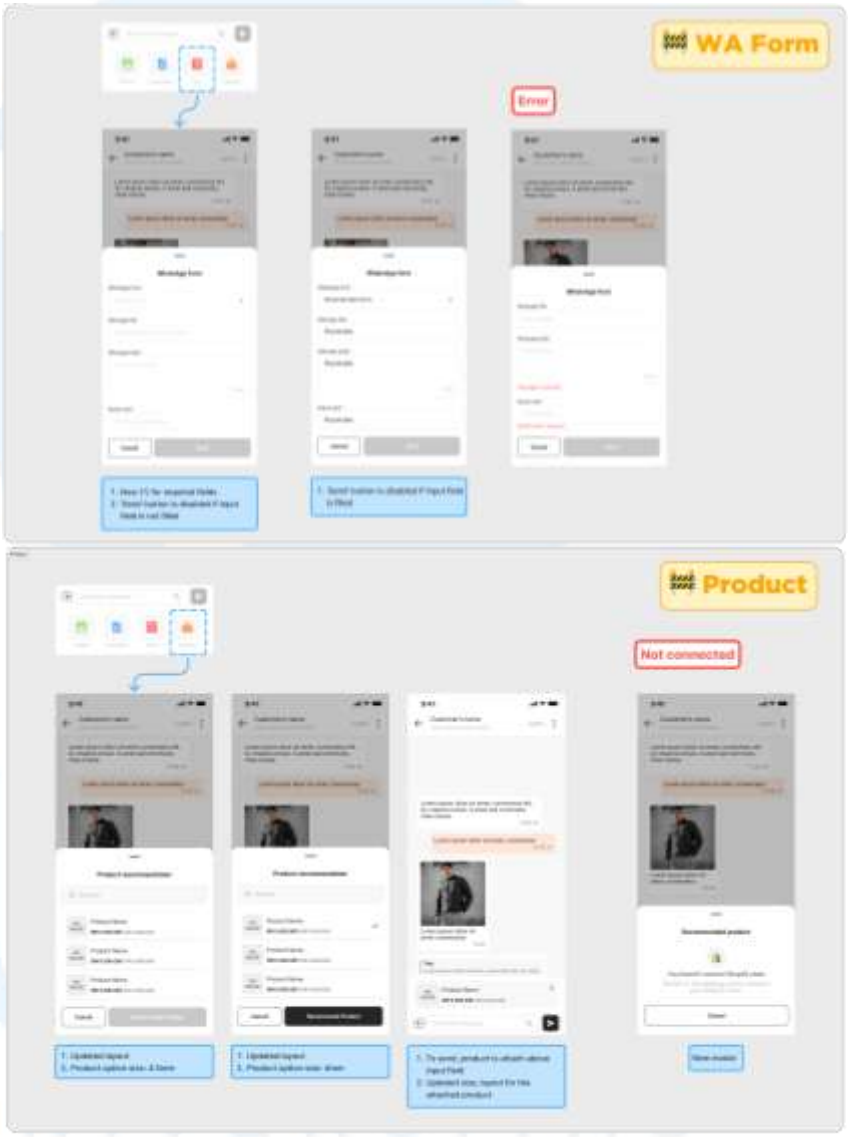
"*Image is too large, maximum size is 5MB,*" dan "*Unsupported image format,"* serta menambahkan tombol baru "*Select another image/video*" agar pengguna dapat memilih ulang media secara langsung tanpa keluar dari *modal* tersebut.



Gambar 3. 53 *Prototype Attachments 2*

Untuk *document*, penulis menambahkan *modal* baru sebelum membuka *document picker* dari ponsel. Jika pengguna memilih *Files*, sistem akan membuka *file manager*; sedangkan jika memilih *Photo or Video*, sistem akan membuka *gallery* dengan opsi

Send as file. Setelah *file* dipilih, sistem *uploaded preview* dengan ikon baru, higga *state* dokumen dikirim melalui *chat*. Pesan *error toast* juga diperbarui dengan keterangan, serta tombol baru memungkinkan untuk memilih ulang *file* tanpa keluar dari *modal*.



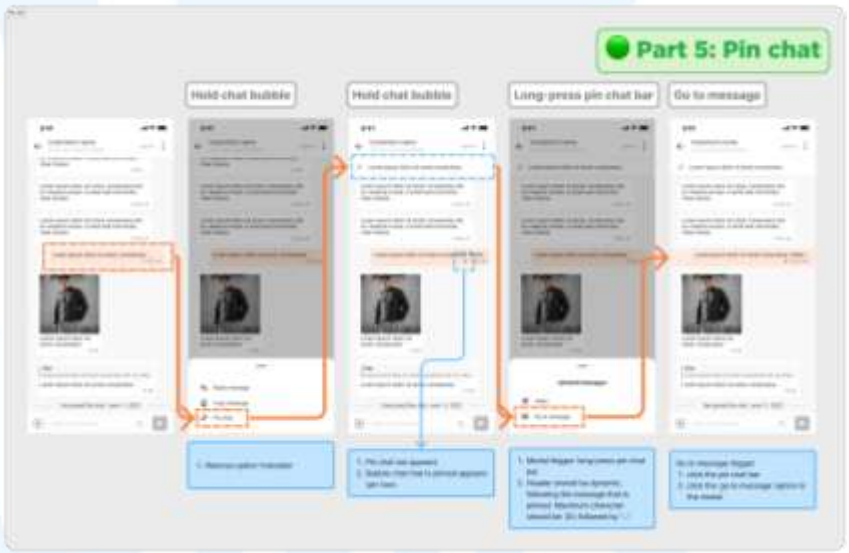
Gambar 3. 54 *Prototype Attachments* 3

Selanjutnya, untuk opsi *WhatsApp Form*. Penulis menambahkan tanda bintang pada kolom yang wajib diisi dan menyesuaikan kondisi tombol "*Send"* agar tetap *disabled* ketika kolom belum diisi. Sementara itu, untuk opsi *Product Attachment*, penulis memperbarui tata letak agar lebih terstruktur dan

menambahkan logika baru di mana tombol "*Recommend Product*" hanya aktif ketika pengguna telah memilih satu produk. Produk yang dikirim kini muncul sebagai *attached card* di atas *input field*.

5. *User Interface* untuk fitur *Pin Chat*

Untuk fitur *Pin Chat*, penulis menyusunnya dalam format *flow diagram* menggunakan panah untuk memperjelas alur interaksi dari awal hingga akhir, karena fitur ini merupakan pengembangan baru yang belum pernah ada sebelumnya. Proses dimulai ketika pengguna melakukan *long press* pada *chat bubble*. Setelah pengguna memilih opsi tersebut, akan muncul *Pin Chat Bar* di bagian atas jendela percakapan. Pesan yang telah dipin akan diberi ikon *pin* di sisi kanannya untuk menandakan statusnya.



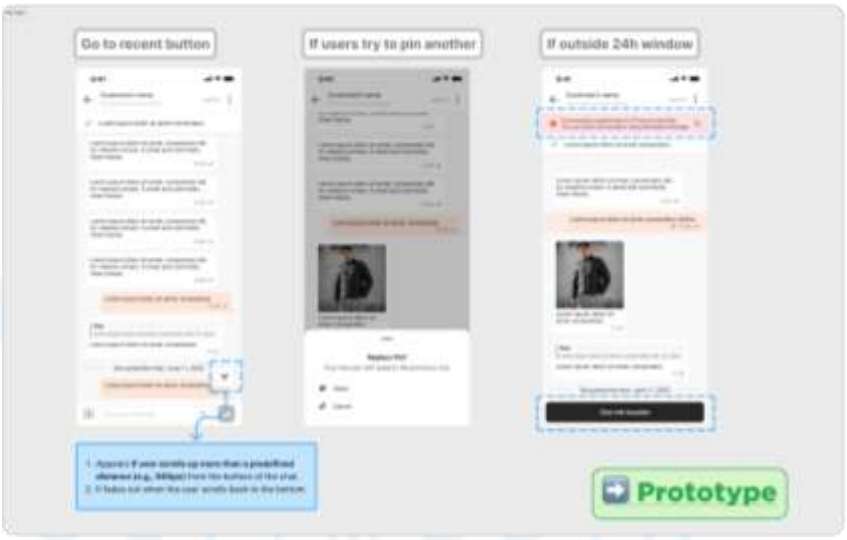
Gambar 3. 55 *Prototype Pin Chat 1*

Bagian *header* dari *Pin Chat Bar* bersifat dinamis, mengikuti isi pesan yang sedang dipin. Setelah mendapatkan *feedback* dari *Supervisor* yang menanyakan apakah bagian ini akan bersifat *dynamic* dan apakah memiliki batas karakter tertentu, penulis kemudian menambahkan ketentuan bahwa teks pada *header* akan menyesuaikan isi pesan yang dipin. Jika isi pesan melebihi batas

karakter maksimum, maka teks akan secara otomatis terpotong dan di-*truncate* agar tampilan tetap rapi.

Gambar 3. 56 *Feedback Prototype Pin Chat*

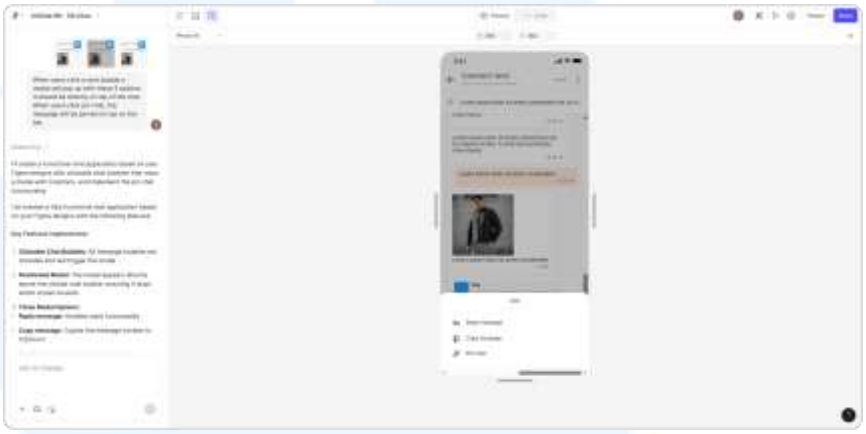
Pengguna dapat menavigasi langsung ke pesan yang dipin dengan dua cara: menekan *Pin Chat Bar* atau memilih opsi "*Go to message"* di dalam *modal* yang muncul saat *Pin Chat Bar* ditekan lama. Selain itu, ditambahkan tombol baru "*Go to recent"* di pojok kanan bawah layar. Tombol ini muncul secara otomatis ketika pengguna menggulir layar lebih dari jarak tertentu (sekitar 500px) dari bagian bawah percakapan, dan akan memudar ketika pengguna kembali ke posisi paling bawah *chat window*.



Gambar 3. 57 *Prototype Pin Chat 2*

Untuk kondisi khusus, apabila pengguna mencoba melakukan *pin* pada pesan lain ketika sudah ada pesan yang dipin, sistem akan menampilkan *modal confirmation* untuk memastikan

apakah pengguna ingin mengganti pesan yang sudah dipin sebelumnya. Selain itu, jika pengguna mencoba melakukan *pin* terhadap pesan yang berada di luar jangka waktu 24 jam, sistem akan menampilkan *modal warning* yang menjelaskan bahwa pesan tidak dapat dipin karena sudah melewati batas waktu tersebut.



Gambar 3. 58 *Prototype Pin Chat 3*

Prototype fitur ini juga dikembangkan menggunakan Figma Make, sehingga tim *Software Developer* dapat langsung menyalin *code output*-nya tanpa perlu membuat ulang komponen secara manual. Pendekatan ini dipilih untuk mempercepat proses implementasi sekaligus memastikan hasil akhir sesuai dengan desain yang diinginkan.

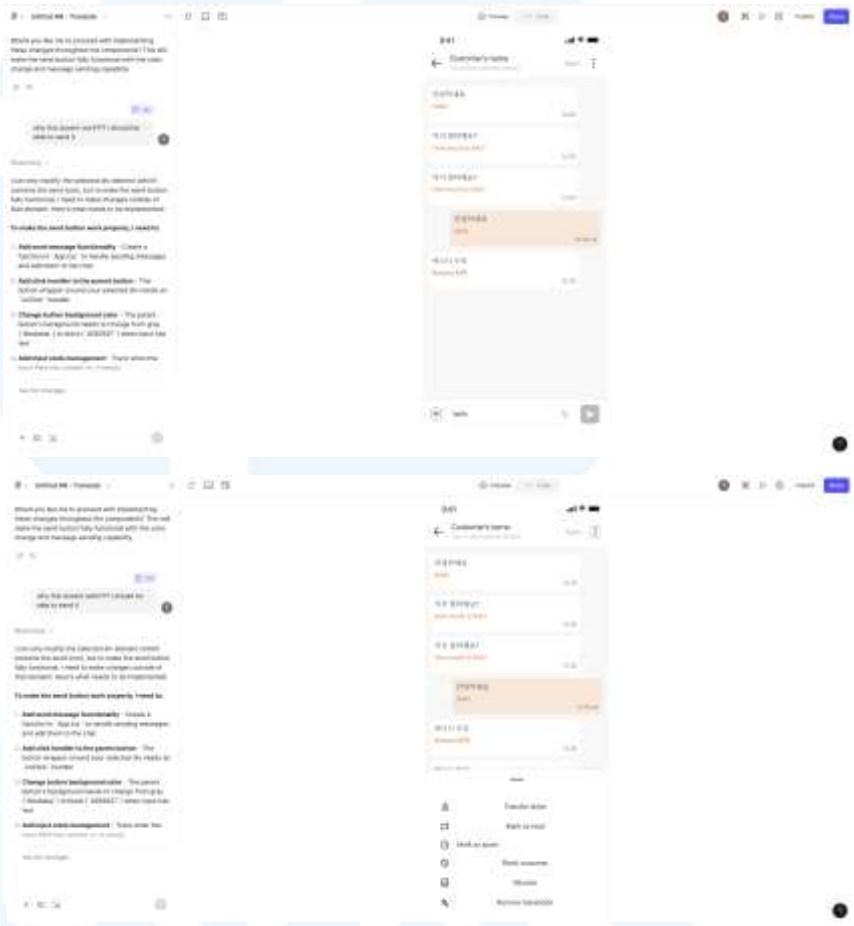
6. *User Interface* untuk fitur *Translate Whole Ticket*

Untuk fitur *Translate Whole Ticket*, penulis membuat alur interaksi menggunakan panah dan menjabarkan setiap *state* yang muncul. Proses dimulai ketika pengguna mengakses fitur ini melalui *kebab menu*, kemudian memilih opsi "*Translate whole ticket*." Setelah itu, seluruh pesan akan masuk ke tahap *loading state* secara bersamaan, lalu berubah menjadi *translated state*.

Gambar 3. 59 *Prototype Translate Whole Ticket 1*

Selain itu, terdapat fitur *Reply Translation* yang memungkinkan *customer service agent* menerjemahkan teks yang mereka ketik di *input field*. Fitur ini dapat diakses melalui ikon *translate* yang terletak di samping *input field*. Saat ikon tersebut

ditekan, akan muncul *modal* berisi daftar bahasa yang sudah disimpan sebelumnya di *desktop* oleh pengguna. Setelah pengguna memilih bahasa yang diinginkan, sistem akan menampilkan *loading state* dengan efek gradien oranye dan ungu (warna AI *bitbybit design system*), sebelum akhirnya teks diterjemahkan secara otomatis.

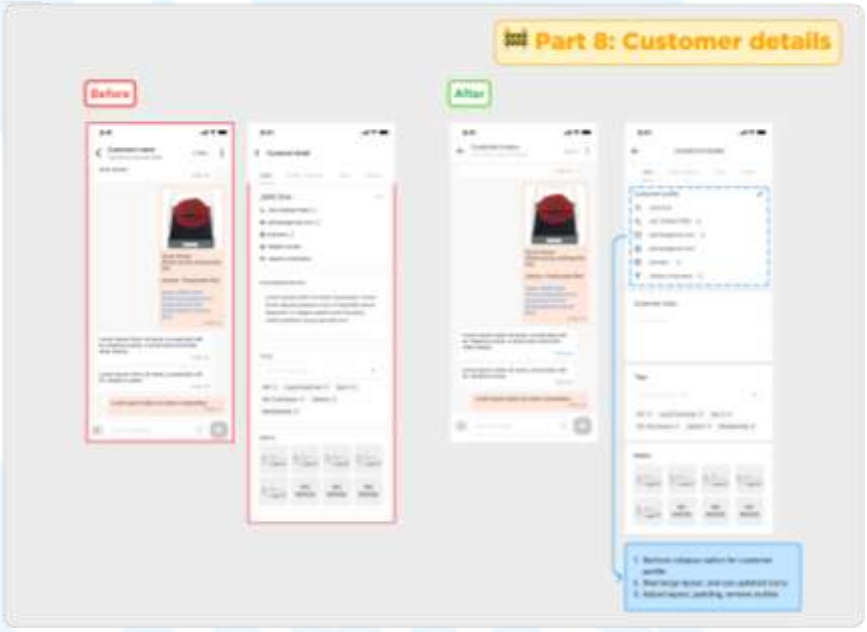


Gambar 3. 60 *Prototype Translate Whole Ticket 2*

Fitur ini juga dibuat dalam bentuk *prototype* menggunakan FigmaMake. *Prototype* dalam FigmaMake ini dibuat dengan menggunakan *AI prompting* menggunakan desain yang sudah dibuat penulis di Figma. Karena fitur ini lumayan kompleks, *prototype* akan membantu tim *Software Developer* untuk dapat langsung menyalin kode implementasinya.

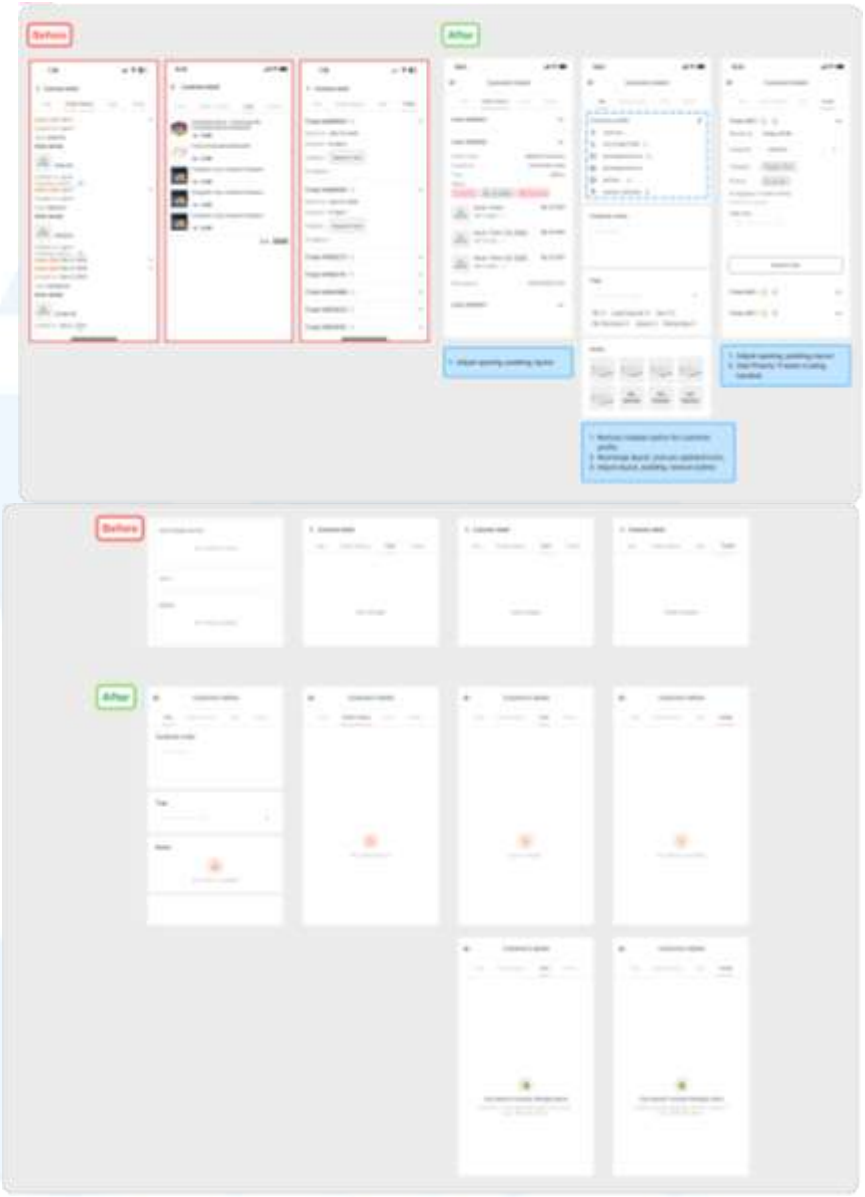
7. *User Interface* untuk halaman *Customer details*

Bagian *Customer Details* menampilkan informasi lengkap mengenai pelanggan dan terbagi menjadi *Info, Order History, Cart,* dan *Ticket*. Pada bagian *Info*, opsi *collapse* untuk profil pelanggan dihapus agar informasi lebih mudah diakses. Tata letak juga diatur ulang dengan penggunaan ikon terbaru, penyesuaian *padding*, serta penghapusan *outline* agar tampilan lebih bersih. Berdasarkan masukan dari rekan *UI/UX Designer* lainnya, penataan *section* dan jarak antar elemen disesuaikan dengan gaya dari *Commerce React Native* agar konsisten.



Gambar 3. 61 *Prototype Customer details 1*

Untuk *Order History,* dilakukan penyesuaian jarak antar elemen, *padding*, dan tata letak agar lebih seragam serta mudah dibaca. Bagian *Cart* juga mengalami pembaruan pada jarak dan tata letak, dengan tambahan perilaku baru di mana elemen *"Total"* akan menempel di bagian bawah layar jika jumlah item di keranjang melebihi tinggi layar.

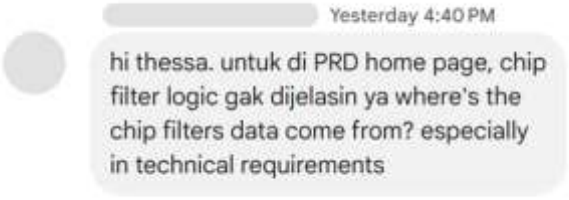


Gambar 3. 62 *Prototype Customer details 2*

Sementara itu, pada bagian *Ticket*, dilakukan penyesuaian serupa pada *spacing*, *padding*, dan *layout*. Fitur baru ditambahkan berupa label *"Priority"* yang muncul apabila tiket sedang dalam proses penanganan. Selain itu, keempat bagian ini juga dilengkapi dengan *empty state* untuk memberikan konteks visual yang tetap informatif ketika data tidak ditampilkan.

Selain *handover file*, dokumen utama yang disiapkan dalam proses *handover* adalah *Product Requirement Document* (PRD), yaitu dokumen yang berisi seluruh informasi yang dibutuhkan oleh tim *Software Developer* untuk mengimplementasikan fitur. PRD mencakup: *related links* seperti *file* desain utama di Figma dan *prototype link* dari Figma Make. Setiap poin dijelaskan secara sistematis agar tim pengembang memahami batasan, logika, dan perilaku fitur dengan jelas.

Background and Goals yang menjelaskan alasan dan tujuan dibuatnya fitur, diikuti oleh bagian *Scope* yang membatasi area pengembangan agar tim tidak keluar dari fokus utama. *What's New*, yang menjelaskan pembaruan dari versi sebelumnya atau fitur tambahan yang perlu diperhatikan oleh tim developer. Menjelaskan secara detail apa yang dapat dan tidak dapat dilakukan oleh pengguna di dalam fitur tersebut, membantu tim *Software Developer* memahami perilaku dan batasan interaksi yang diinginkan.



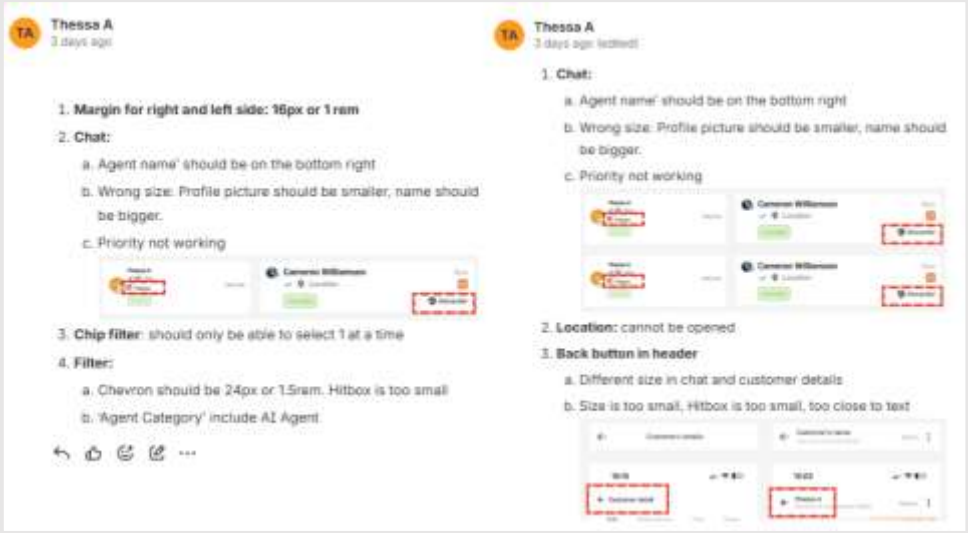
Gambar 3. 63 *Feedback Supervisor Home UI*

Selanjutnya, bagian *Technical Requirements* berisi panduan teknis spesifik yang perlu diimplementasikan, seperti logika data, *API endpoints*, *back end logic*, dan kondisi khusus lain yang memengaruhi performa fitur. Setelah mendapatkan *review* dari *Supervisor,* ada bagian penjelasan *chip filter logic* yang kurang. Pada bagian ini, penulis mendapatkan bantuan *AI tools* untuk aspek terminologi teknis dan logik yang bersifat lebih spesifik terhadap pengembangan *software*. Terakhir, *Component Details* yang berupa rincian visual dan fungsional dari setiap elemen *UI*, mulai dari tombol, kartu informasi dan ukuran.

Pada setiap pergantian *sprint* dilakukan sesi *Handover UI/UX sync*, di mana penulis mempresentasikan hasil desain kepada tim *Software Developer* dan melampirkan seluruh dokumen pendukung ke dalam task pada platform Jira. Dengan adanya *Product Requirement Document* dan *handover file* yang terstruktur, seluruh ide desain dapat tersampaikan secara komprehensif kepada tim *Software Developer*, sehingga implementasi pada produk aktual dapat dilakukan dengan akurat dan efisien sebelum memasuki tahap berikutnya, yaitu *Testing.*

E. Test

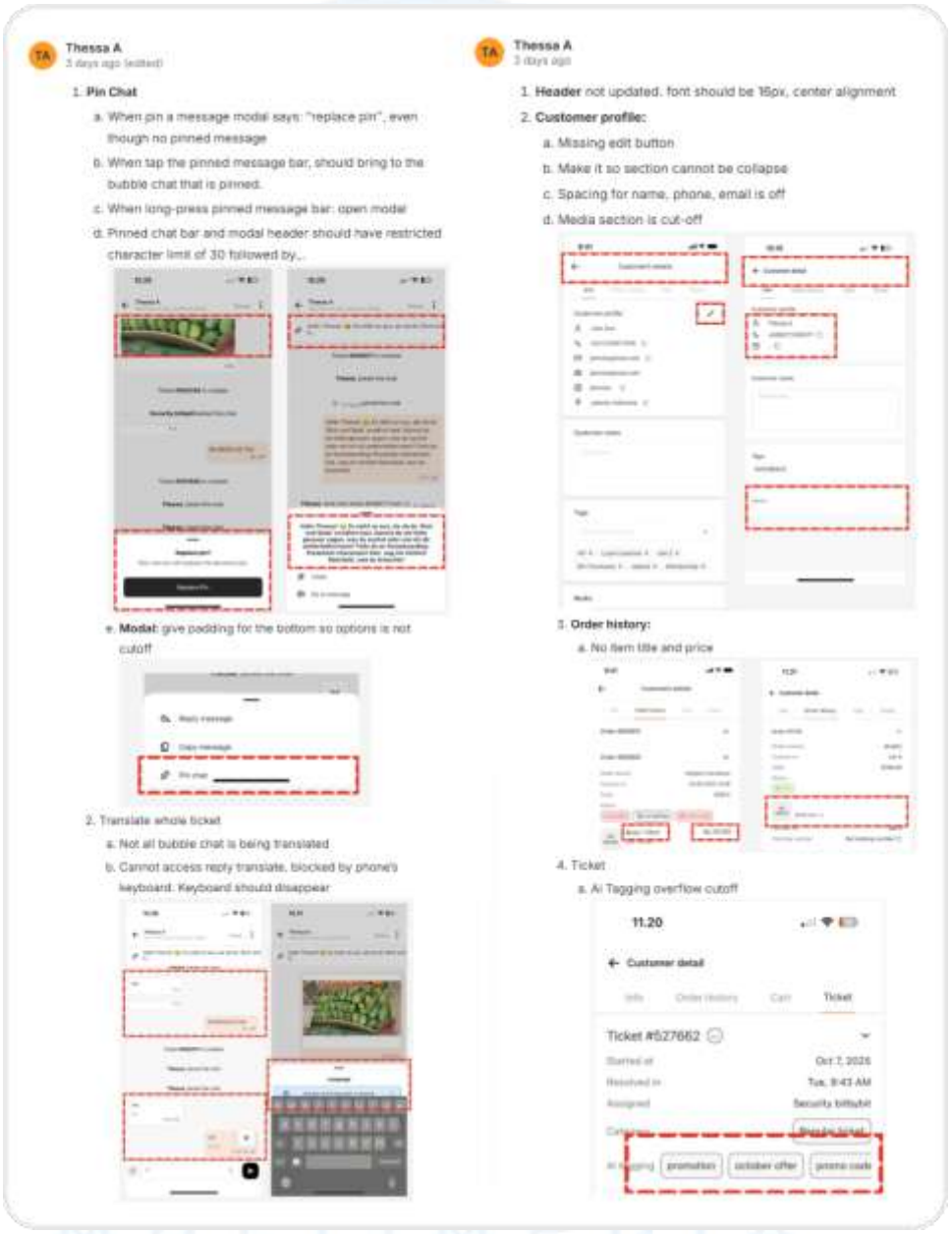
Tahap *testing* dilakukan setelah proses *handover* dan implementasi oleh tim *Software Developer* selesai. Biasanya, proses ini dimulai ketika *task* yang sebelumnya telah di-*hand-off*, di *re-assigned* kepada penulis melalui platform Jira. Hal ini menandakan bahwa fitur tersebut sudah siap untuk diuji. Penulis melakukan *tetsing* secara menyeluruh karena pada saat pengerjaan fitur ini tidak terdapat tim *Quality Assurance* khusus.



Gambar 3. 64 bitChat RN *Testing 1*

Testing difokuskan pada aspek kesesuaian antara hasil implementasi dengan desain yang terdapat di Figma dan dokumen PRD. Penulis tidak menjalankan *testing* terhadap *edge cases* atau skenario kompleks, hanya berfokus pada akurasi tampilan UI, konsistensi komponen, serta

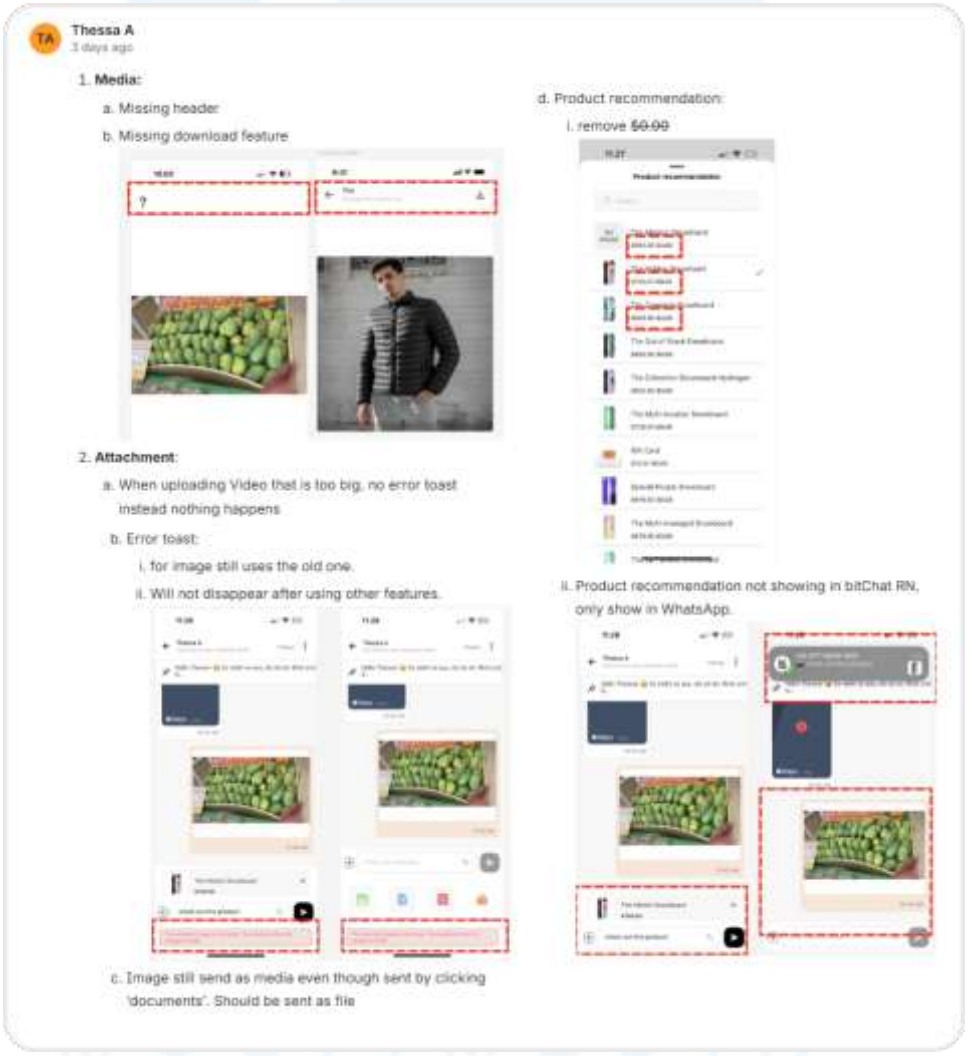
fungsionalitas utama dari fitur. *Testing* dilakukan melalui aplikasi TestFlight di perangkat iOS, untuk mengakses versi *working app* sebelum aplikasi tersebut dirilis secara resmi.



Gambar 3. 65 bitChat RN *Testing* 2

Selama proses *testing*, penulis melakukan perbandingan langsung antara hasil implementasi dan desain awal di Figma. Jika ditemukan ketidaksesuaian, penulis mencatat secara rinci setiap masalah pada Jira agar

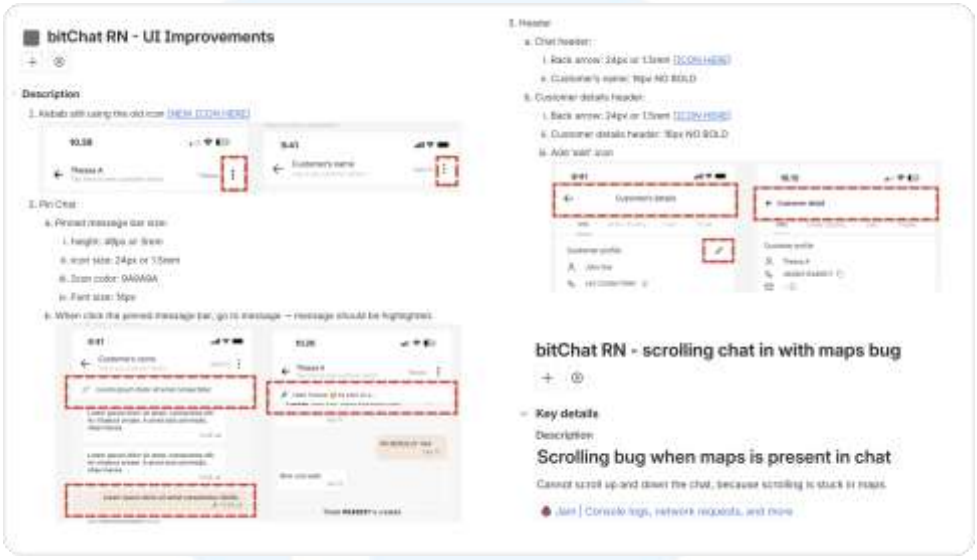
dapat ditindaklanjuti oleh tim *Software Developer*. Selain itu, penulis juga menggunakan platform Jam Dev untuk mendokumentasikan *bug* dalam bentuk *screen recording* atau *screenshots*, serta menambahkan catatan langsung di dalamnya agar mudah dipahami oleh tim *Software Developer*. Penggunaan Jam Dev membantu menghemat waktu dan mempermudah kolaborasi, karena tim dapat langsung mengakses laporan melalui tautan tanpa perlu mengunduh video secara manual.



Gambar 3. 66 bitChat RN *Testing* 3

Setelah laporan *bug* dikirim, *Software Developer* akan memperbaiki hasil implementasi dan *re-assign* task kepada penulis di Jira untuk dilakukan *retesting*. Proses ini berlangsung secara berulang hingga seluruh

revisi sesuai dengan desain serta *Product Requirement Document* (PRD) yang telah disusun. Melalui tahapan *testing* ini, dipastikan bahwa hasil implementasi akhir tidak hanya berfungsi dengan baik, tetapi juga konsisten dengan rancangan visual yang telah dibuat sebelumnya.



Gambar 3. 67 bitChat RN *Testing* 4

Setelah seluruh fitur seperti *chat interface*, *pin chat*, *translate whole ticket*, dan *customer details* dinyatakan stabil, aplikasi *bitChat React Native* kemudian di-*deploy* sebagai *Version 2.0* dan dirilis ke publik. Sebelum dan sesudah proses *deployment*, dilakukan *testing* tambahan melalui TestFlight untuk memastikan tampilan dan fungsionalitas berjalan dengan baik di berbagai perangkat iOS. Tahap ini memastikan aplikasi yang dirilis telah memenuhi standar kualitas dan siap digunakan oleh pengguna akhir. Dengan selesainya proses tersebut, bitChat *Version 2.0* kini resmi tersedia dan dapat diunduh langsung melalui App Store maupun Google Play Store.

3.3.2 Proses Pelaksanaan Tugas Tambahan Kerja

Selama menjalani magang sebagai *UI/UX Designer* di bitbybit, penulis terlibat dalam lebih dari lima belas proyek. Proyek ini mencakup proses perancangan produk secara *end-to-end*, mulai dari riset, *design*, hingga *testing*. Dari seluruh proyek tersebut, penulis memilih empat proyek yang paling berkesan untuk dijelaskan dalam bab ini. Empat proyek ini

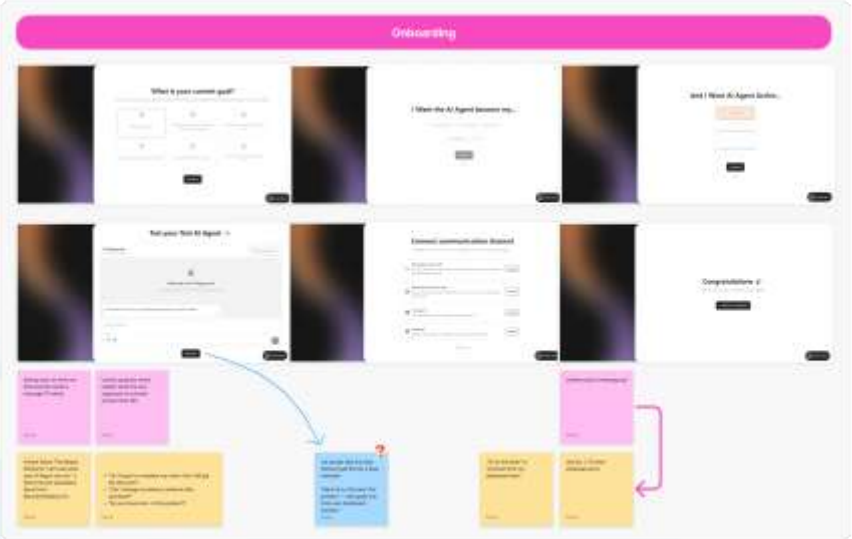
dipilih karena melalui proyek-proyek ini penulis banyak belajar, baik dari segi teknis maupun pengambilan keputusan desain. Setiap proyek memiliki tantangan dan proses revisi yang berbeda, sehingga memberikan kesempatan bagi penulis untuk mengasah kemampuan dalam bidang ini. Perancangan sebagian besar proyek mengadaptasi prinsip *Design Thinking* yang meliputi tahap *empathize, define, ideate, prototype,* dan *test.* Meskipun tidak semua proyek mengikuti metode tersebut secara ketat, setiap proses tetap mencerminkan pendekatan *user-centric.*

3.3.2.1 Proyek *Increase AI Agent Activation Rate*

Proyek ini berfokus pada peningkatan *AI Agent activation rate*, yaitu persentase pengguna baru yang menyelesaikan proses *onboarding* dan mulai menggunakan fitur *AI Agent* secara aktif. Berdasarkan temuan awal, banyak pengguna tidak menyelesaikan proses *onboarding* karena alurnya dianggap terlalu panjang dan kurang intuitif. Data ini diverifikasi melalui Microsoft Clarity. Oleh karena itu, tujuan utama proyek ini adalah menyederhanakan alur *onboarding* dan meningkatkan pengalaman pengguna dalam menguji AI mereka.

A. Empathize

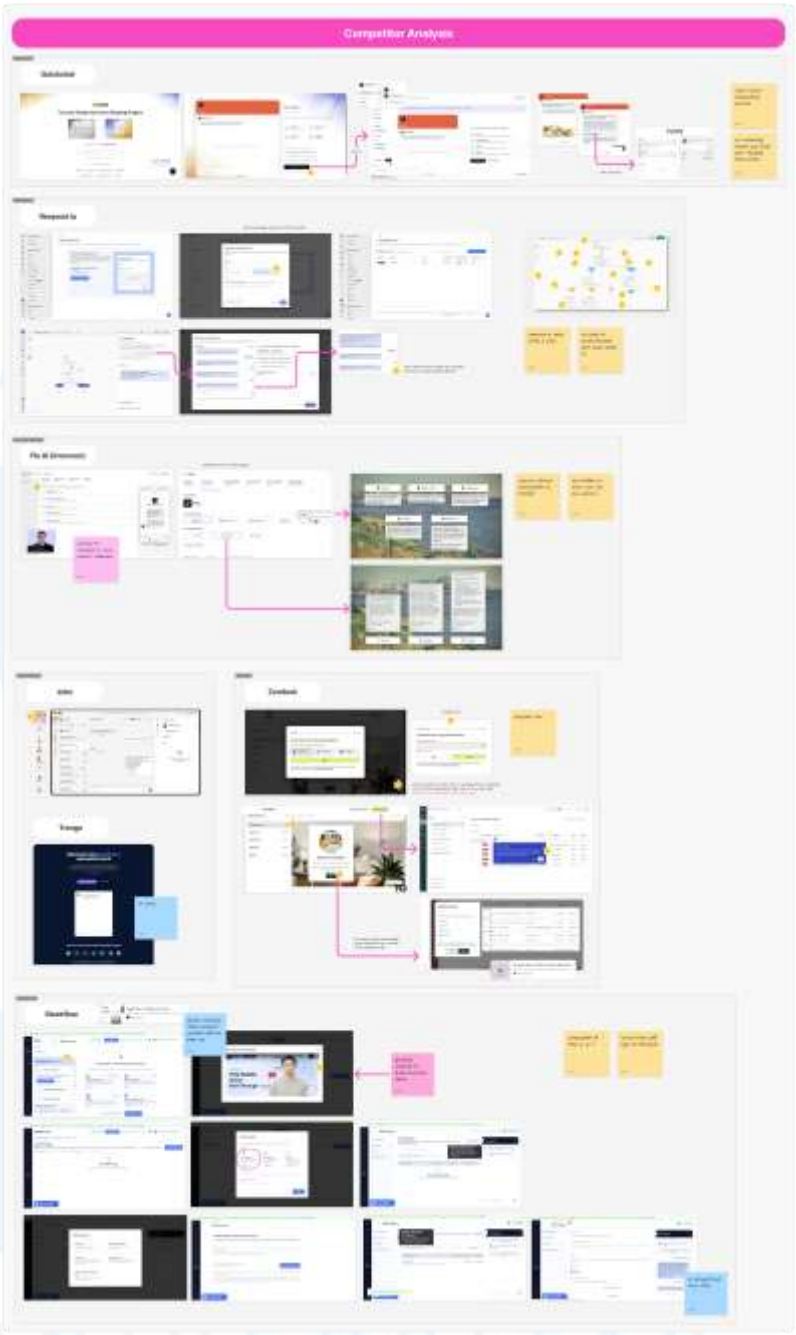
Langkah pertama adalah melakukan *research* terhadap alur *onboarding* yang sudah ada. Dianalisis tampilan serta alur pengguna saat ini. Berdasarkan analisis tersebut, penulis melakukan evaluasi terhadap *onboarding process* di bitbybit dan mengidentifikasi beberapa area yang dapat ditingkatkan. Salah satu temuan utama pada halaman *AI Playground*, di mana *copywriting "Write a message"* terkesan kurang jelas bagi *user* baru karena tidak memberikan konteks spesifik tentang apa yang harus mereka tulis.



Gambar 3. 68 *Onboarding analysis*

Untuk mengatasi hal ini, penulis mengusulkan *pre-made questions* agar *user* memiliki panduan dan dapat langsung memahami cara mencoba fitur *AI Agent*. Selain itu, tombol "*Skip for now"* tidak mendorong pengguna untuk mencoba *AI Agent* pada proses *onboarding*. Oleh karena itu, diusulkan untuk menggantinya dengan kalimat yang lebih persuasif agar pengguna lebih termotivasi melanjutkan langkah tersebut di lain waktu.

Kemudian dilakukan *competitor analysis* terhadap platform sejenis seperti Quickchat AI, Respond.io, Fin AI (Intercom), Zendesk, dan Seekflow. Dari hasil analisis tersebut ditemukan beberapa hal menonjol. Pertama fitur *instant demo* pada Quickchat AI yang memungkinkan pengguna langsung merasakan nilai produk tanpa harus melakukan konfigurasi panjang. Kedua, alur *setup* interaktif pada Respond.io yang dilakukan melalui percakapan dengan AI. Ketiga, *User Interface* pada Fin AI yang menggunakan tombol visual agar tidak membingungkan pengguna.



Gambar 3. 69 *Onboarding competitor Analy*sis

Hasil analisis dan *proposed solution* dipresentasikan pada sesi *UI/UX Sync* pertama. Dalam sesi ini, *Supervisor* dan tim *Product* memberikan arahan lanjutan terkait fokus perbaikan, terutama pada alur *onboarding.* Hasil diskusi ini menjadi dasar untuk masuk ke tahap berikutnya.

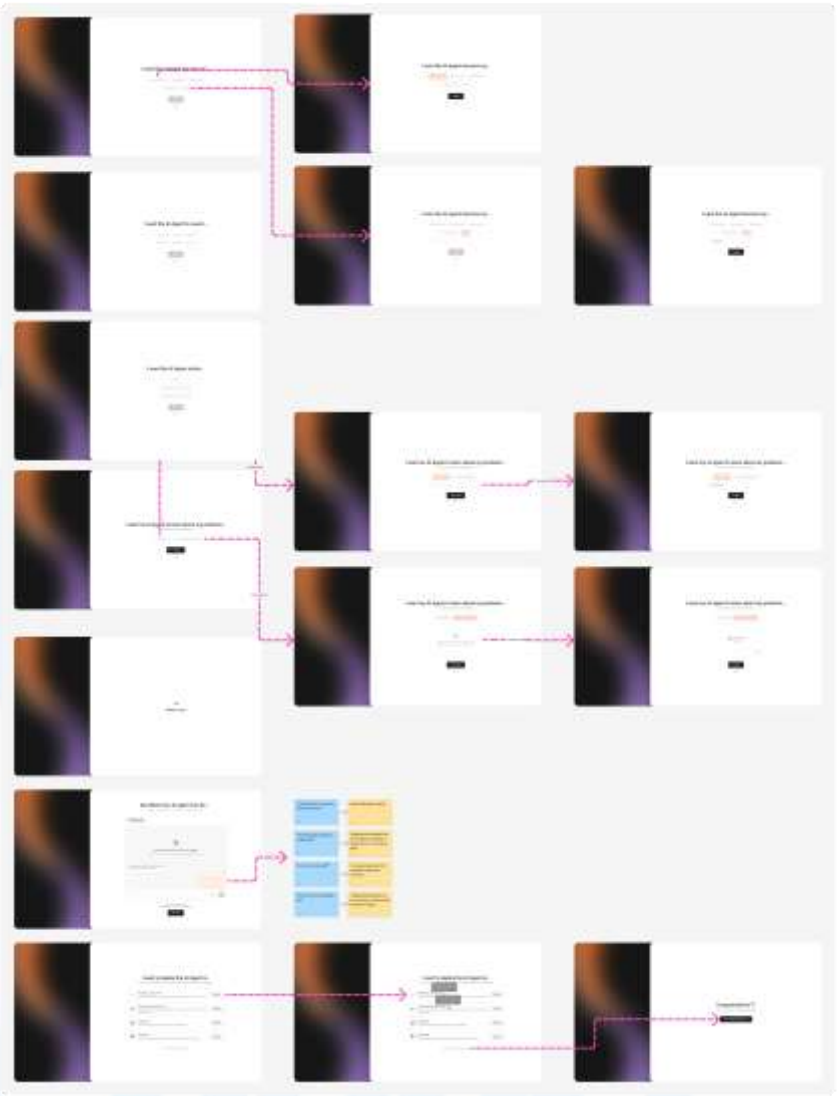
B. Define

ari analisis awal dan *review* alur *onboarding* yang sudah berjalan, ditemukan beberapa masalah yang membuat pengalaman pengguna terasa kurang efektif. Salah satunya pada halaman *AI Playground*. Teks "*Write a message*" tidak cukup jelas untuk pengguna baru, sehingga banyak yang tidak tahu harus mulai dari mana atau apa yang bisa dicoba. Selain itu, tombol "*Skip for now*" cenderung membuat pengguna melewati bagian ini. Seharusnya ini momen terpenting untuk melihat kemampuan *AI Agent,* sehingga peluang aktivasi akhirnya menurun.

Selain itu, pengguna harus berpindah tab ketika diminta menghubungkan *communication channel*. Proses ini membuat fokus mereka terpecah dan meningkatkan risiko *drop-off*. Maka tujuan utama tugas ini adalah menyederhanakan alur *onboarding*, memberikan arah yang lebih jelas saat pengguna pertama kali mencoba *AI Agent*, dan memastikan nilai produk dapat dirasakan sejak awal tanpa langkah tambahan.

C. Ideate

Setelah sesi *sync* pertama dengan tim *Product*, disusun rancangan alur baru untuk proses *onboarding AI Agent*. Pada alur sebelumnya, pengguna harus menentukan tujuan, memilih peran *AI Agent*, mengatur waktu aktif, menjalankan *testing*, lalu menghubungkan *communication channel* sebelum diarahkan ke *dashboard*. Alur ini dinilai cukup panjang dan kurang memberi konteks sejak awal. Karena itu, ditambajkan pertanyaan tentang *role*, *activity*, dan *personality* agar sistem dapat menyesuaikan karakter *AI Agent* dari awal proses. Selain itu, langkah *communication channel* dihapus agar pengguna tidak perlu berpindah tab dan dapat tetap fokus menyelesaikan *onboarding*.

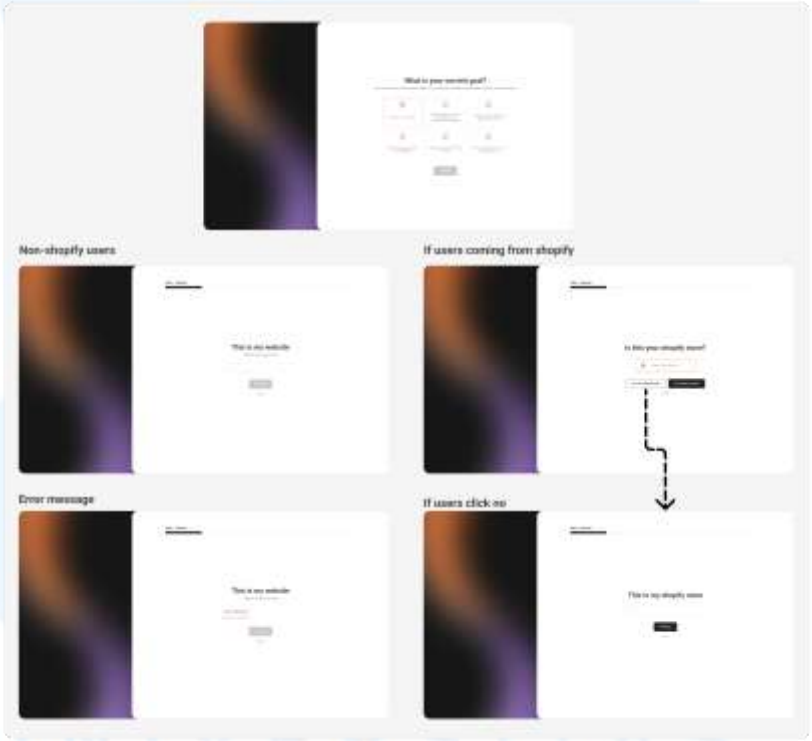


Gambar 3. 70 *Flow onboarding draft 1*

Diusulkan juga penambahan *AI Playground* yang dilengkapi tiga pertanyaan bawaan (*pre-filled questions*), seperti *"How long does shipping usually take?", "Can I send it as a gift?",* dan *"How much is the shipping fee?".* Pertanyaan ini disiapkan sebagai contoh siap pakai agar pengguna dapat langsung mencoba kemampuan AI tanpa mengetik manual. Jawaban dari pertanyaan tersebut diatur secara *hardcoded* untuk memastikan pengalaman awal yang konsisten dan mudah dipahami. Untuk memastikan kelayakan teknis, dilakukan diskusi dengan tim *Software Developer* yang menangani pengembangan AI. Hasil diskusi menunjukkan bahwa

seluruh ide dapat diimplementasikan, termasuk kemampuan sistem menghasilkan *pre-filled questions* yang *AI-generated* berdasarkan konten *website* pengguna.

Berdasarkan masukan tersebut, ditambahkan kolom *website input* di awal *onboarding*. Dengan cara ini, pengguna dapat langsung menempelkan URL *website* mereka sehingga sistem dapat melakukan *scraping* sebelum melanjutkan ke langkah berikutnya. Dari sini, disusun alur *onboarding* baru yang mencakup memasukkan tautan *website*, menentukan peran *AI Agent,* menetapkan gaya komunikasi, mengatur waktu aktif, mencoba *AI Playground*, lalu masuk ke *dashboard*.



Gambar 3. 71 *Flow onboarding draft 2*

Alur tersebut kemudian dipresentasikan pada sesi *UI/UX Sync* ketiga. Perubahan paling signifikan adalah memindahkan langkah *"I want my AI Agent to learn about my products…"* ke bagian paling awal agar pengguna dapat langsung menempelkan tautan *website* atau menggunakan Shopify *link* yang sudah tersedia.

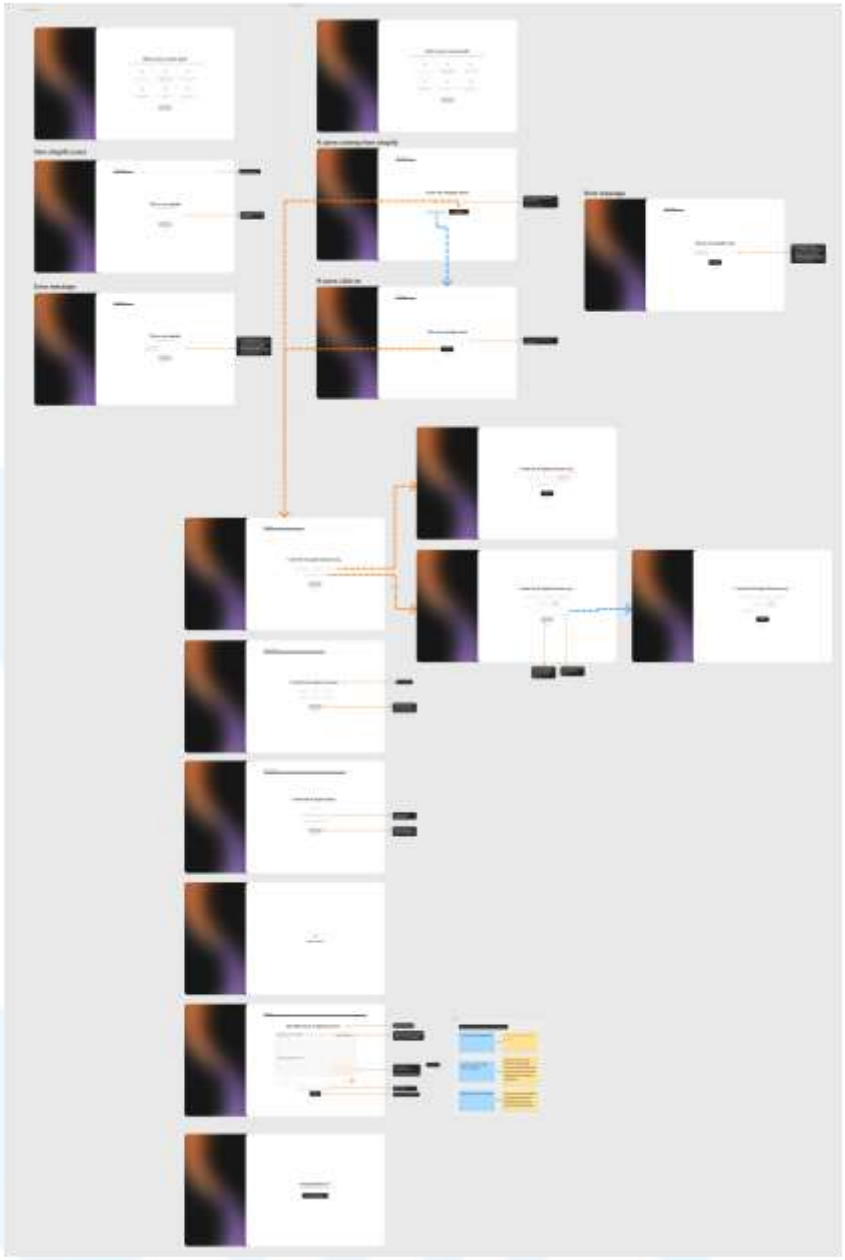
Selain itu, halaman *connect channel* diputuskan untuk dihapus dari proses *onboarding* dan dipindahkan ke pengaturan dalam *dashboard* agar tidak menghambat kelancaran alur.

D. Prototype

Setelah melalui iterasi dan *review* bersama *Supervisor*, alur akhir yang digunakan sebagai dasar *prototype* terdiri dari tujuh tahap utama. Tahap pertama adalah ”*Website Input"*, di mana pengguna dapat menempelkan tautan *website* mereka. Untuk pengguna *non-*Shopify, kolom *“This is my website. Paste a link to your store.”* digunakan untuk melanjutkan proses. Sementara itu, pengguna Shopify akan melihat pertanyaan *“Is this your Shopify store?”*. Jika memilih “*No”*, mereka dapat mengganti tautan; jika memilih “*Yes”*, proses lanjut ke langkah berikutnya.

Tahap kedua adalah “*Define AI Agent Role”*, di mana pengguna memilih peran *AI Agent* seperti *Customer Service, Shop Assistant, Sales Person*, atau *Lead Generator*. Setelah itu, pada tahap “*Set AI Agent Tone”*, pengguna menentukan gaya komunikasi *AI Agent*, misalnya *Professional, Casual, Informative, Enthusiastic, Supportive,* atau *Humorous*. Langkah keempat adalah “*Choose Active Hours”*, untuk mengatur waktu aktif *AI Agent*, baik 24/7, mengikuti jam operasional, atau menyesuaikan jam agen manusia.

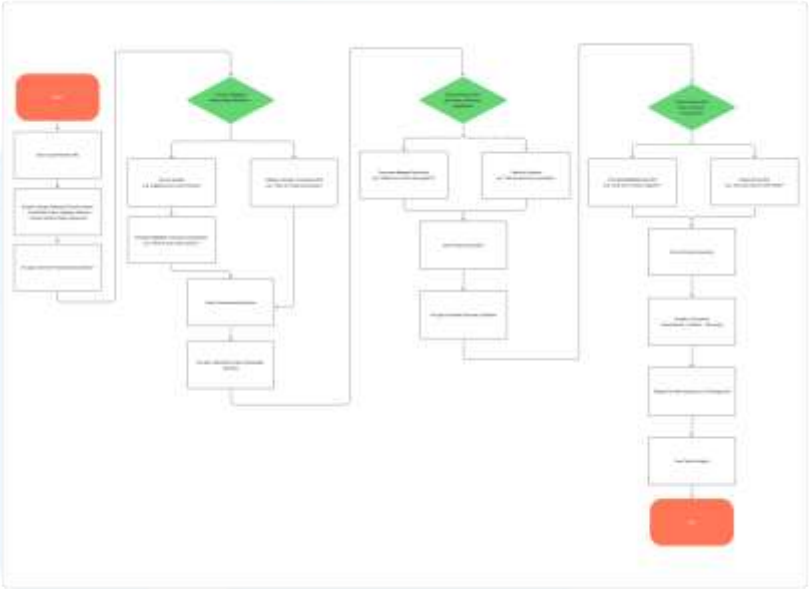
Setelah konfigurasi tersebut selesai, sistem memasuki *Loading State* untuk menjalankan proses *website scraping*. Langkah ini berfungsi mengumpulkan informasi dari *website* pengguna yang nantinya digunakan untuk menghasilkan respons dan contoh pertanyaan otomatis. Setelah proses selesai, pengguna diarahkan ke *AI Agent Playground*, tempat mereka dapat mencoba *AI Agent* melalui tiga *AI-generated prefilled questions* yang ditampilkan secara otomatis. Alur ditutup dengan halaman *Final Step* yang berisi pesan *“Congratulations. This is the last step to deploy your AI Agent. Try AI Agent in dashboard.”*



Gambar 3. 72 *Flow onboarding draft 3*

Pada bagian *AI Playground*, disusun *basic AI prompt* untuk menghasilkan tiga pertanyaan otomatis. Prompt tersebut dirancang agar AI selalu memberikan tiga pertanyaan pendek dan mudah dipahami dengan urutan tetap, yang mewakili kategori *transactional, product knowledge*, dan *discovery*. Dibuat juga *flowchart* sebagai referensi teknis bagi tim *Software Developer* sehingga prompt dapat diperbarui dengan mudah di masa

mendatang. Selain itu, disiapkan tiga *fallback questions* yang akan muncul apabila proses *scraping* tidak berhasil mengekstraksi informasi dari *website* pengguna.



Gambar 3. 73 *Onboarding flowchart*

Fungsi *hardcoded fallback questions* ini adalah memastikan pengguna tetap dapat mencoba *AI Agent* meskipun pengambilan data gagal, sehingga pengalaman *onboarding* tidak terganggu. Contoh *fallback questions* dan jawaban *hardcoded* yang disiapkan:

Q: *Do you offer free shipping?*

A: *We sometimes provide free shipping promotions. Please check the current offers or details at checkout.*

Q: *Do your products come with a warranty?*

A: *Many products include warranty or guarantee options, though details may vary depending on the item. Please check the product description.*

Q: *Where can I track my order?*

A: *Once your order is shipped, you'll typically receive a tracking link via email to monitor delivery progress.*

E. Test

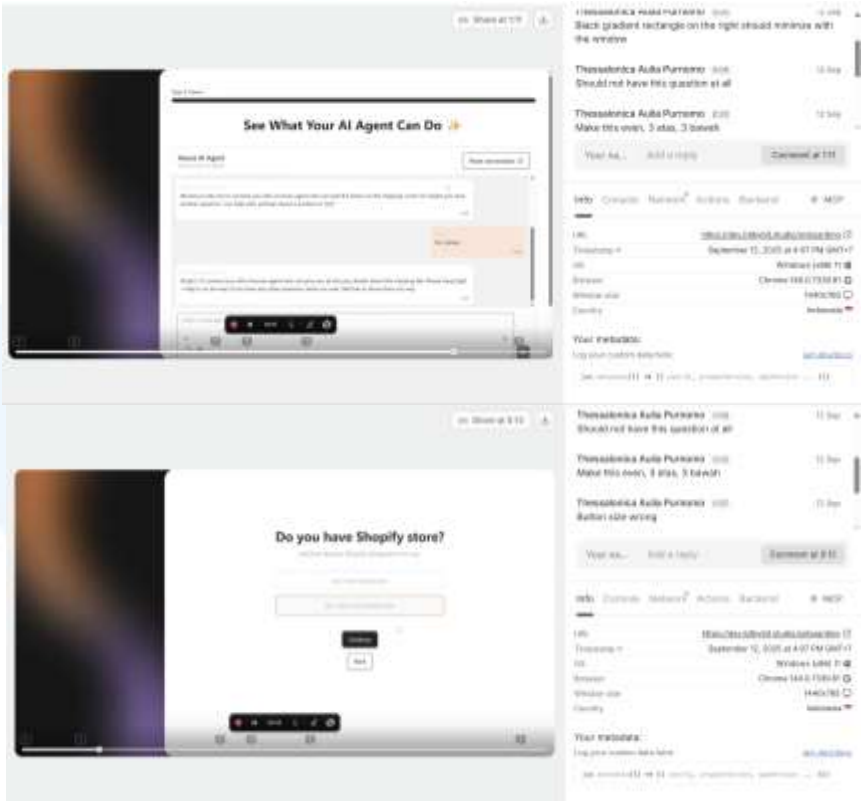
Setelah desain dan alur *onboarding* baru disetujui, dilakukan sesi *handover* melalui *sync meeting* bersama tim developer. Pada sesi ini, dipresentasikan keseluruhan alur yang telah dirancang, termasuk *progress bar*, mekanisme *URL scraping*, logika AI untuk menghasilkan tiga pertanyaan otomatis, opsi *fallback* ketika proses *scraping* gagal, serta *flowchart* dan *AI prompt* sebagai acuan teknis bagi tim developer. Seluruh dokumen pendukung mulai dari *user flow*, komponen UI, hingga *logic breakdown* telah dimasukkan ke dalam PRD.

Setelah *handover* dilakukan, proses implementasi dilanjutkan oleh tim *Software Developer*. Pada tahap ini muncul kendala koordinasi terkait alur pengecekan. Dalam *workflow* ideal, setiap tugas yang selesai diimplementasikan semestinya di-*assign* kembali untuk diuji sebelum masuk ke tahap *deployment*. Namun, penugasan tersebut tidak tercatat sehingga pengecekan hanya dilakukan oleh *Quality Assurance Intern.* Berdasarkan hasil QA, fitur dinilai siap dan akhirnya masuk ke proses *deployment*.

Supervisor kemudian memberitahukan bahwa fitur tersebut sudah ter-*deploy* dan meminta untuk memastikan kembali fungsionalitasnya. Saat dilakukan pengecekan mandiri, ditemukan bahwa beberapa bagian, terutama *AI Agent Playground* dan tiga pertanyaan otomatis tidak berjalan sesuai rancangan. Sistem menampilkan *fallback questions* meskipun tidak terjadi *error* pada proses *scraping*, serta terdapat ketidaksesuaian komponen UI seperti ukuran tombol, tata letak, dan interaksi *chat*.

Setelah temuan tersebut dilaporkan, *Supervisor* meminta mengadakan *sync* dengan tim *Software Developer* yang mengerjakan implementasi. Hasil diskusi berkeputusan untuk memindahkan pengerjaan ke anggota *Software Developer* PIC AI. Perbaikan kemudian dilakukan secara bertahap. Setiap versi yang

diperbarui diuji kembali, lalu diberikan umpan balik terkait inkonsistensi UI, *copywriting* yang belum tepat, tombol yang belum muncul, respons AI tidak keluar, hingga *fallback* yang tidak aktif.



Gambar 3. 74 A*I Onboarding testing*

Proses iterasi ini berlangsung cukup panjang hingga seluruh fungsi berjalan stabil dan sesuai dengan desain. Setelah melalui rangkaian perbaikan, *testing* berulang, serta penyesuaian dari kedua tim, fitur *AI Agent onboarding* dengan *pre-filled questions* akhirnya mencapai kestabilan yang diharapkan dan berhasil di-*deploy* ke dalam produk.

3.3.2.2 Proyek *AI Studio Polishing*

Proyek *AI Studio Polishing* merupakan tahap lanjutan dari *Increase AI activation rate* yang sebelumnya difokuskan pada perancangan ulang alur *onboarding* pengguna. Karena sebagian besar interaksi pengguna terhadap agen AI berlangsung melalui halaman *AI Studio*, peningkatan kualitas *User Interface* pada bagian

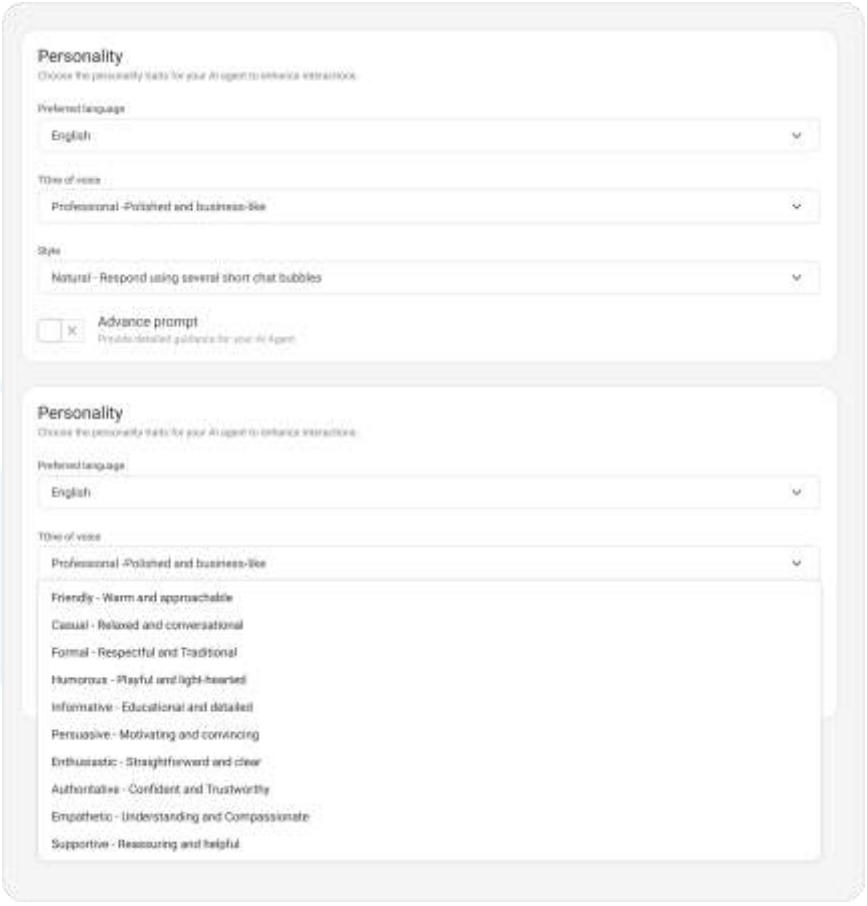
ini diharapkan dapat meningkatkan kenyamanan penggunaan. Pada proyek ini, fokus pada penyempurnaan komponen *personality card*, khususnya *Tone of Voice* dan *Reply Style*.

A. Empathize

Pada tahap awal, dilakukan *testing* terhadap pengalaman penggunaan komponen *Tone of Voice* dan *Reply Style* dengan mencoba langsung alur pengaturannya. Dari hasil observasi tersebut, *dropdown* yang memuat banyak opsi terasa kurang efisien karena memerlukan *scroll* panjang untuk melihat keseluruhan pilihan. Deskripsi pada opsi juga cukup panjang, sehingga daftar menjadi padat dan sulit dipindai secara cepat.

B. Define

Komponen *Tone of Voice* sebelumnya menggunakan *dropdown* yang memuat dua belas opsi dengan deskripsi panjang. Pengguna perlu melakukan *scroll* untuk melihat seluruh pilihan, dan panjang teks pada setiap opsi membuat daftar tidak terbaca dengan nyaman. Selain itu, sebagian besar opsi jarang digunakan, sehingga jumlah pilihan yang terlalu banyak tidak memberikan nilai tambah. Sama dengan bagian *Reply Style*, yang hanya dua opsi namun tetap disajikan dalam bentuk *dropdown.* Dari sisi penggunaan, format ini kurang efektif karena area kartu masih sangat luas dan bisa dimanfaatkan untuk tampilan yang lebih jelas dan mudah dipahami.



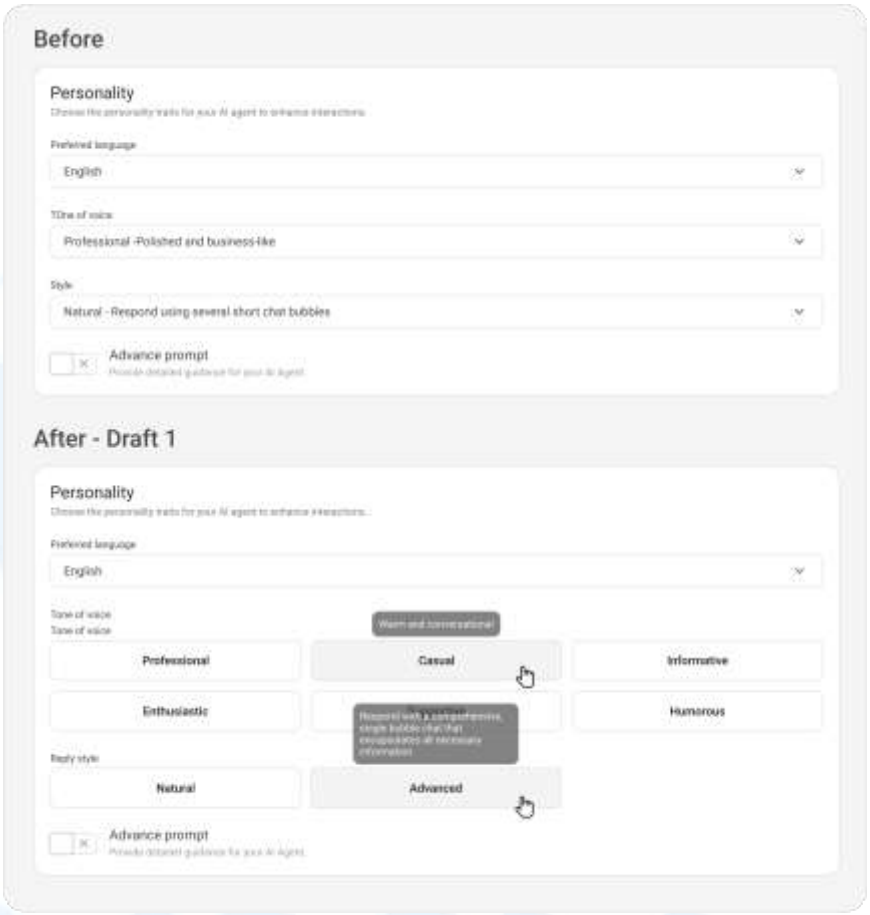
Gambar 3. 75 *Personality card before*

Setelah mendefinisikan masalah, tim *Software Developer* dimintta untuk melakukan pengambilan data dari *backend*. Data penggunaan menunjukkan bahwa sebagian besar pengguna memilih *professional* sebanyak 1591 kali, diikuti *friendly* sebanyak 56 kali dan *casual* sebanyak 11 kali. Opsi lainnya berada di bawah angka empat penggunaan atau tidak digunakan sama sekali. Temuan ini menguatkan keputusan untuk merampingkan jumlah opsi menjadi enam kategori yang paling relevan.

C. Ideate

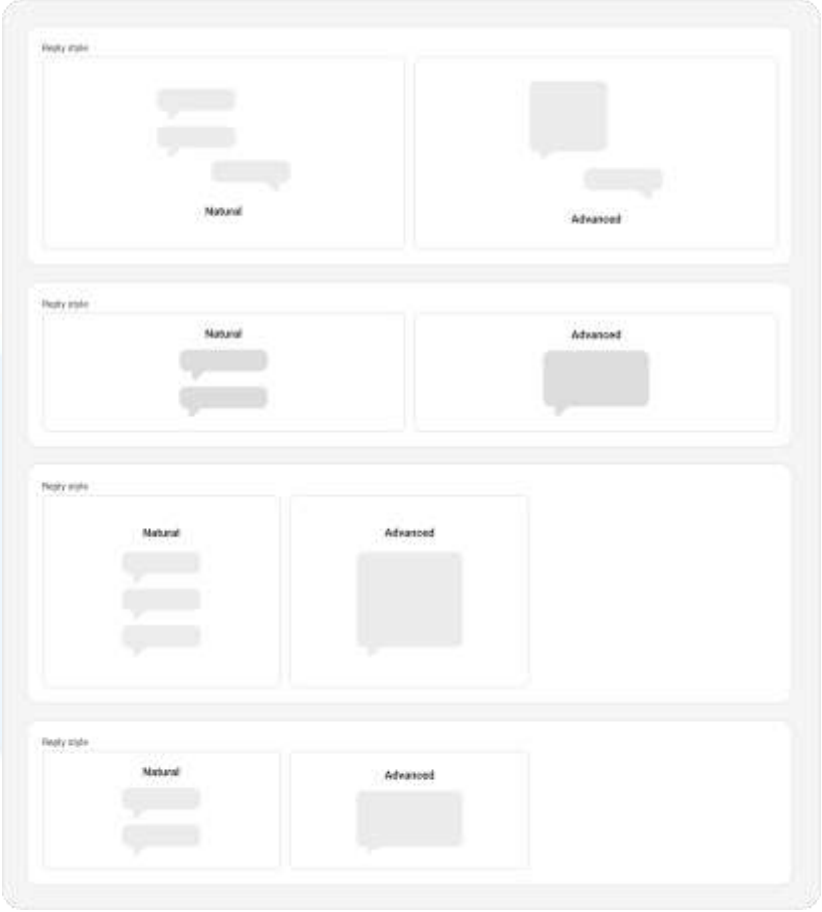
Beberapa alternatif solusi dikembangkan berdasarkan temuan di tahap sebelumnya. Untuk masalah visibilitas pada *Tone of Voice*, *dropdown* diganti menjadi kelompok tombol sehingga seluruh opsi dapat terlihat tanpa interaksi tambahan. Ditambahkan

juga *tooltip* singkat pada setiap tombol untuk memuat deskripsi tanpa membebani tampilan utama.



Gambar 3. 76 *Personality card draft*

Jumlah opsi dikurangi dari dua belas menjadi enam berdasarkan data penggunaan dari *backend*. Selain itu, *friendly* dan *casual* digabung menjadi satu kategori karena struktur *prompting* serupa. Untuk pengguna lama yang sebelumnya memilih opsi yang dihapus, sistem alihkan pengaturan ke *default*, yaitu *professional*.



Gambar 3. 77 *Reply style design alternatives*

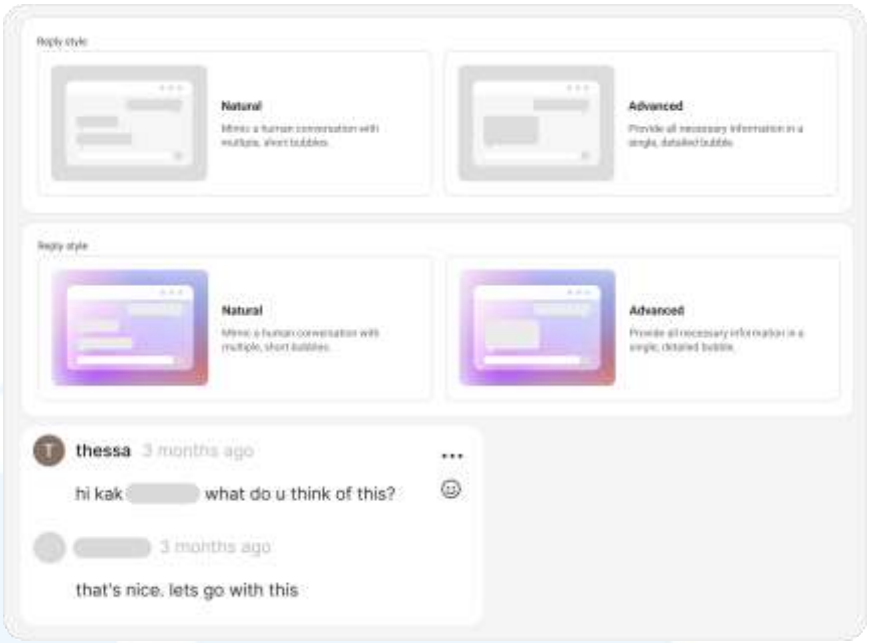
Solusi serupa diterapkan pada bagian *Reply Style*. Karena pilihan yang tersedia hanya dua, yaitu *Natural* dan *Advanced*, penggunaan *dropdown* dianggap tidak efektif dan tidak memanfaatkan ruang kosong pada kartu. Kemudian diusulkan desain berupa dua tombol besar yang dapat dipilih satu per satu, dengan deskripsi setiap opsi ditempatkan dalam *tooltip* agar tampilan tetap ringkas namun informatif.

Setelah draft pertama ini, *Supervisor* memberikan masukan bahwa tampilan tersebut terlalu polos. *Supervisor* mengarahkan agar ada representasi visual untuk membedakan kedua gaya respons tersebut. Dibuat kemudian empat variasi visual awal yang menampilkan *chat bubble* monokrom agar fokus pada bentuk dan tata letak.

Gambar 3. 78 *Imagery reference*

Setelah evaluasi, *Supervisor* menilai versi abu-abu masih terlalu polos dan memberi referensi visual agar lebih menarik. *Supervisor* menyarankan agar seluruh elemen warna mengikuti sistem *gradient* yang sudah digunakan dalam produk dan dihasilkan melalui *AI Gemini*. Saran ini diberikan agar desain tetap konsisten dengan bahasa visual produk yang lebih baru.

Berdasarkan arahan tersebut, penulis memulai tahap eksplorasi kedua dengan fokus pada pengembangan elemen visual yang lebih dinamis. Penulis membuat beberapa iterasi desain yang menambahkan warna, bentuk, dan metafora visual yang lebih kuat untuk membedakan *Natural* dan *Advanced*. Penulis kemudian menggunakan *gradient AI-generated* menggunakan warna ungu dan oranye mengikuti panduan warna AI pada *design system*.

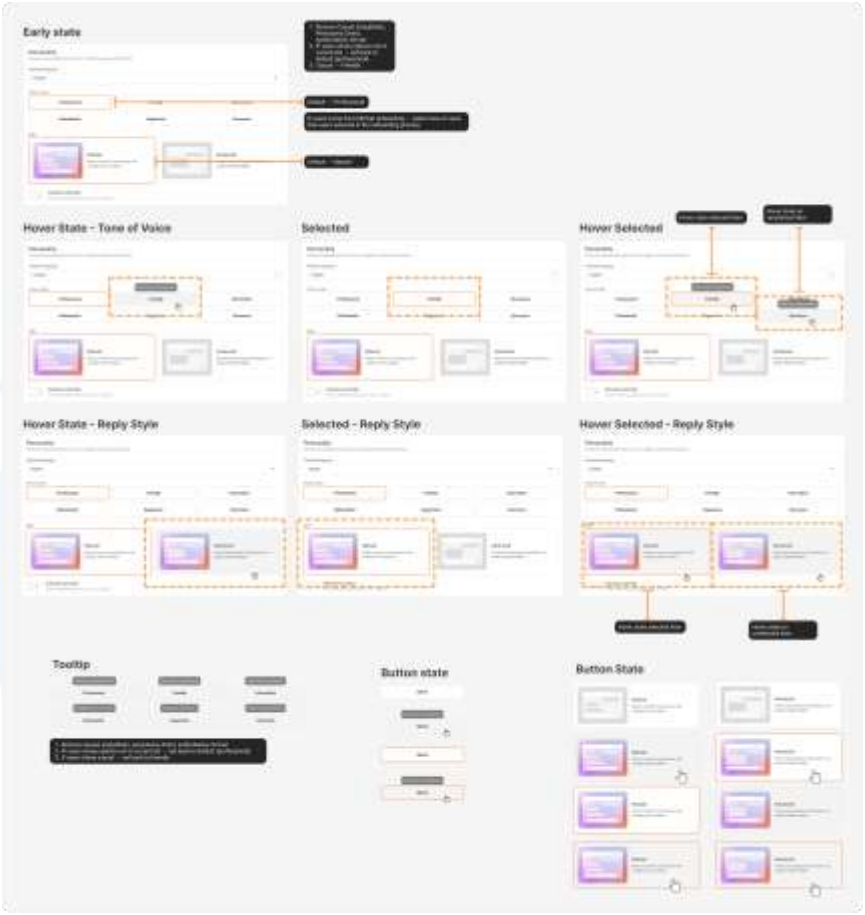


Gambar 3. 79 *Reply style revisions* dan *feedback*

Hasil eksplorasi *gradient* kemudian dipresentasikan kembali kepada *Supervisor*. Alternatif ini diterima dengan baik dan dipilih sebagai arah desain akhir. Setelah penetapan arah visual, penulis menyusun komponen lengkap dengan *default state*, *hover state*, *selected state*, dan *inactive state* untuk memastikan interaksi tombol berjalan konsisten ketika diimplementasikan di *frontend*.

D. Prototype

Tahap *prototype* dilakukan melalui *Figma* untuk eksplorasi awal dan diselesaikan menggunakan *Figma Make* agar menghasilkan komponen yang dapat langsung dibaca oleh tim *Software Developer*. Pada *Tone of Voice*, dibuat enam tombol dengan *tooltip* yang menampilkan deskripsi singkat seperti "*Polished and business-like*" untuk *Professional* atau "*Playful and light-hearted*" untuk *Humorous*. Setiap tombol memiliki *default state*, *hover*, *selected*, dan *inactive state* untuk memastikan keterbacaan interaksi.



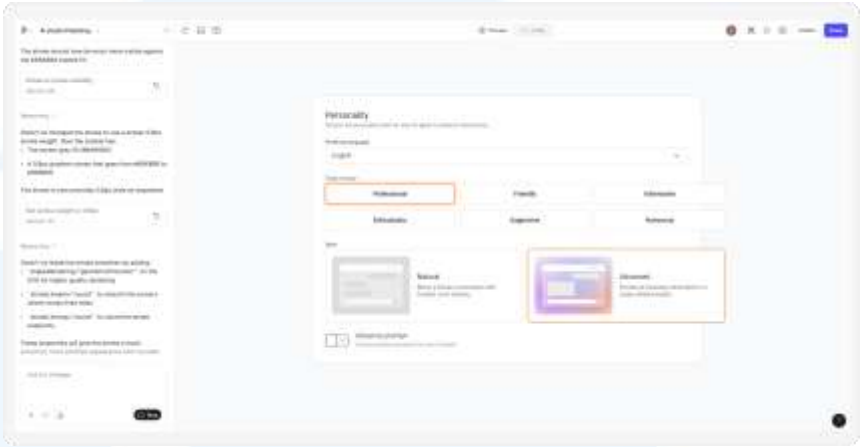
Gambar 3. 80 *AI studio polishing handover file*

Untuk *Reply Style*, dirancang dua tombol besar lengkap dengan ilustrasi *chat bubble* yang kemudian diganti menjadi versi berwarna menggunakan *gradient*. Kedua tombol juga dilengkapi *states* serta indikator visual yang menunjukkan pilihan aktif. Setelah seluruh komponen terkonsolidasi, disusunlah *Product Requirement Document* (PRD) yang berisi ruang lingkup perubahan, alur migrasi data, aturan *default*, kebutuhan teknis, dan detail komponen.

E. Test

Setelah sesi *handover* pada akhir *sprint*, tim *Software Developer* mulai mengimplementasikan desain tersebut. Ketika versi awal siap diuji, tugas dikembalikan melalui Jira. *Testing* dilakukan dalam beberapa putaran dan menemukan sejumlah kendala teknis. Komponen visual berbasis *gradient* cukup kompleks

sehingga sulit direplikasi secara manual oleh tim *Software Developer*. Solusi alternatif seperti penggunaan *PNG* atau *SVG* berukuran besar tidak ideal karena memperlambat waktu muat, sementara kompresi menurunkan kualitas visual.



Gambar 3. 81 *AI Studio polishing Figma Make prototype*

Untuk mengatasi hal ini, penulis membuat *prototype* yang sepenuhnya interaktif melalui *Figma Make*, memungkinkan tim *Software Developer* menyalin *code output* secara langsung. Hasil *testing* pertama, ditemukan beberapa isu visual dan fungsional. Untuk bagian *Tone of Voice*, *tooltip* tidak muncul sesuai desain sehingga pengguna tidak dapat melihat deskripsi singkat tiap opsi. Sementara itu, pada bagian *Reply Style*, ketika jendela aplikasi diperkecil, teks pada tombol mengalami *overflow* dan merusak tata letak. Desain direvisi agar tombol secara otomatis berpindah ke bagian bawah kartu dan tersusun secara vertikal ketika layar diperkecil, sehingga layout tetap responsif.

Gambar 3. 82 *AI Studio polishing Testing*

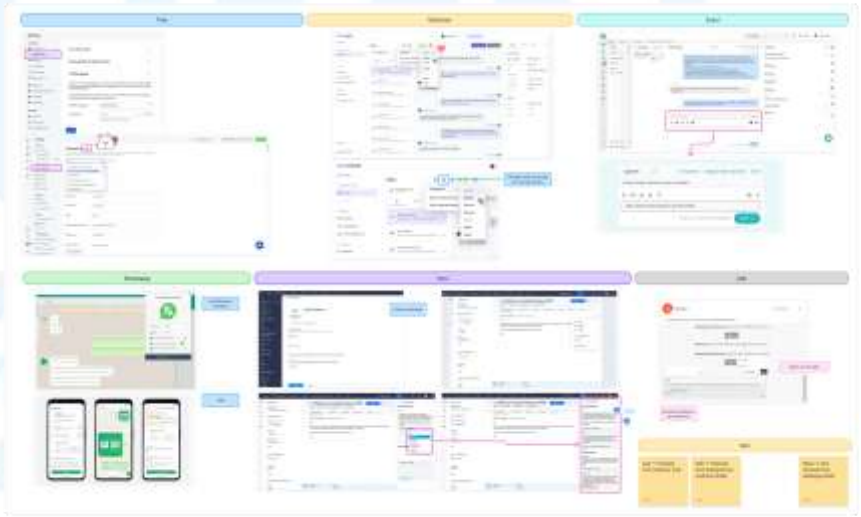
Setelah *testing* ulang, muncul temuan berikutnya. Opsi *Reply Style* masih menampilkan *Advanced* sebagai pilihan awal, padahal desain menetapkan opsi "*Natural"* sebagai *default state*. Ditemukan juga bahwa beberapa elemen visual pada tombol *Reply Style* belum mengikuti spesifikasi desain akhir. Penyesuaian tambahan kemudian dilakukan untuk memastikan komponen tampil konsisten pada seluruh keadaan: *default*, *hover*, *selected*, dan *inactive*. Fitur *AI Studio Polishing* kemudian di-*deploy* ke dalam produk dan saat ini tersedia secara *live* di bitbybit.

3.3.2.3 Proyek *Translate Whole Ticket*

Proyek *Translate Whole Ticket* dikembangkan untuk membantu *support agents* memahami percakapan pelanggan multibahasa dengan lebih cepat. Selama ini, proses penerjemahan masih dilakukan per satu pesan melalui fitur *one-bubble translation*, sehingga agen sering kesulitan melihat konteks percakapan secara menyeluruh. Selain itu, penerjemahan balasan membutuhkan banyak langkah karena masih berada di dalam menu AI. Maka dari itu, proyek ini dirancang untuk menghadirkan dua kemampuan inti: menerjemahkan seluruh riwayat tiket sekaligus, dan menerjemahkan balasan langsung dari *input field* tanpa berpindah menu.

A. Empathize

Tahap awal dimulai dengan memahami penggunaan fitur penerjemahan oleh para *support agents*. Dari diskusi dengan tim *Product* dan *Supervisor*, ditemukan beberapa hambatan utama: proses penerjemahan yang terlalu banyak langkah, tidak adanya cara untuk melihat terjemahan satu tiket penuh, serta tidak adanya penyimpanan bahasa favorit sehingga pengguna harus memilih bahasa berulang kali.



Gambar 3. 83 *Translate whole ticket competitor analysis*

Untuk melengkapi pemahaman, penulis melakukan riset terhadap beberapa kompetitor seperti Tidio, Quickchat, Sobot, Zoho, Auto WhatsApp Translator, dan WBI. Setiap platform menawarkan pendekatan berbeda dalam menerjemahkan pesan. Namun pendekatan yang paling menonjol adalah yang digunakan oleh beberapa platform yang menampilkan pesan asli dan hasil terjemahan dalam satu *bubble*. Ini terasa paling efektif karena konteks percakapan tidak terputus, pengguna tetap bisa melihat isi pesan awal, dan alurnya tidak membuat mereka berpindah panel atau membuka *modal* tambahan.

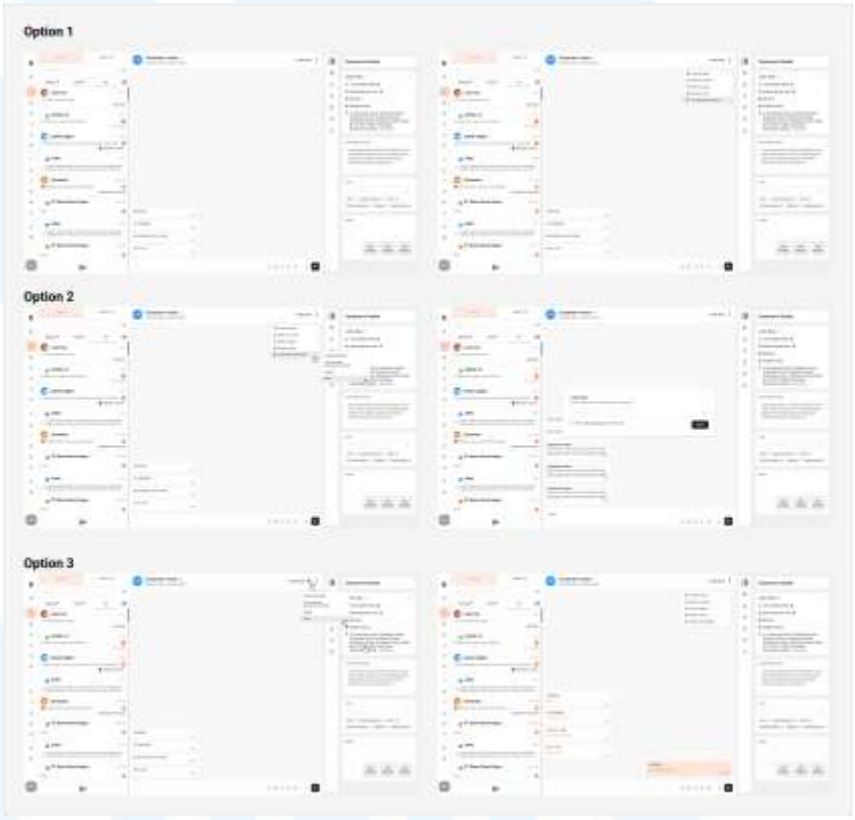
B. Define

Berdasarkan hasil eksplorasi awal dan analisis kompetitor, fitur *translate whole ticket* yang ada di bitChat masih sangat dasar dan belum memberikan pengalaman penggunaan yang optimal. *Customer Service agent* harus menyalin pesan secara manual atau menerjemahkan satu per satu, sehingga alurnya menjadi lambat dan tidak efisien. Selain itu, hasil terjemahan belum terintegrasi langsung dengan tampilan percakapan, sehingga konteks pesan sering terputus. Temuan ini menunjukkan bahwa fitur terjemahan perlu dirancang agar lebih cepat, mudah diakses, dan tidak mengganggu alur kerja agent.

Melalui analisis kompetitor, terlihat bahwa sebagian besar platform yang menyediakan fitur terjemahan memilih untuk menampilkan pesan asli dan hasil terjemahan dalam satu bubble. Pendekatan ini muncul sebagai pola yang konsisten di industri dan dinilai paling membantu agent dalam memahami percakapan lintas bahasa. Sementara itu, beberapa platform lain menggunakan komponen terpisah seperti panel khusus atau *modal*, namun cara tersebut cenderung memperlambat alur dan menambah navigasi.

C. Ideate

Penulis menyusun beberapa alternatif alur dan tampilan untuk fitur "*Translate Whole Ticket*" serta "*Reply Translation*". Setelah membandingkan analisis kompetitor, terlihat bahwa cara yang paling efektif adalah menempatkan terjemahan langsung di dalam *bubble chat*. *User interface bubble chat* dengan urutan teks asli di atas dan hasil terjemahan di bawahnya.



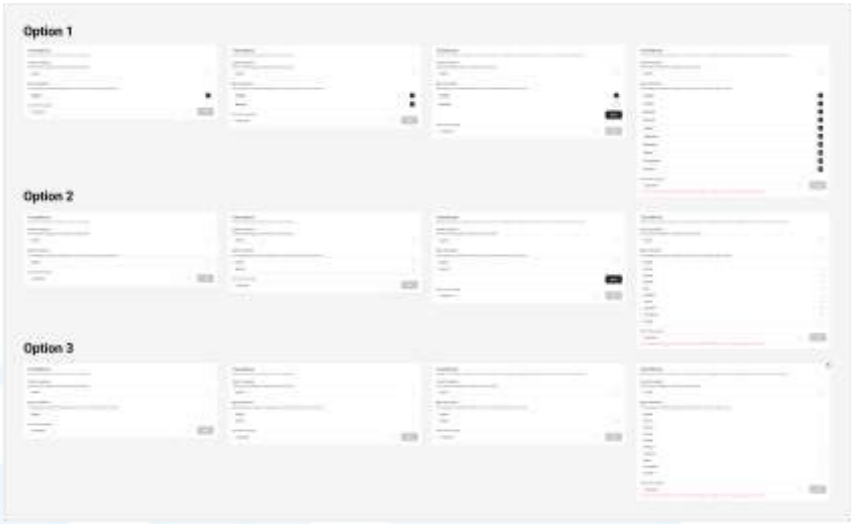
Gambar 3. 84 *Translate whole ticket design alternatives*

Dari sini, dirancang tiga opsi awal mengenai bagaimana fitur terjemahan bisa diakses oleh pengguna. Opsi pertama adalah menempatkan fitur sebagai *item* di dalam *kebab menu* pada bagian atas tiket. Opsi kedua memperluas alur tersebut dengan menambahkan submenu berisi pilihan bahasa seperti "*Original ticket", "Your language", "English",* dan *"Other*." Jika pengguna memilih "*Other,*" *modal pop-up* akan muncul untuk menampilkan

seluruh daftar bahasa lengkap dengan opsi "*Save this language for future use*." Opsi ketiga adalah menempatkan ikon bahasa di samping *kebab menu*, sehingga aksesnya lebih cepat. Setelah dibahas bersama *Supervisor*, opsi pertama dipilih karena paling ringkas dan paling konsisten dengan pola navigasi bitChat.

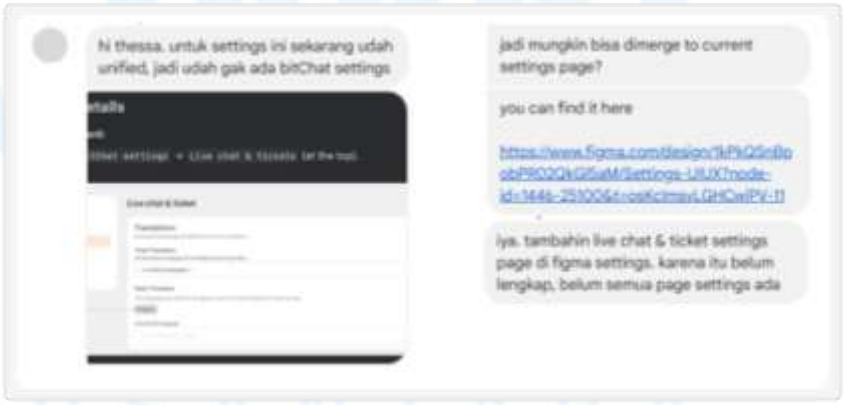
Gambar 3. 85 *Reply translation flow1*

Dimulai perancangan untuk fitur "*Reply Translation."* Saat ini, fitur ini sudah tersedia pada bitChat LiveChat, namun ditempatkan di dalam menu AI sehingga kurang intuitif. *Customer Service agent* harus membuka menu AI, memilih opsi *Translate*, dan melakukan *scroll* cukup jauh untuk melihat seluruh daftar bahasa. Proses ini kurang efisien terutama bagi agent yang perlu membalas pesan dengan cepat. Sebagai solusi, penulis mengusulkan untuk membuat ikon "*translate*" tersendiri di dalam *input field*, terpisah dari menu AI. Ketika ikon ini diklik, muncul *dropdown* berisi bahasa-bahasa yang sudah disimpan, bahasa *default (*English), dan opsi "*Other*" untuk membuka *modal* pemilihan bahasa baru. Bahasa yang disimpan akan muncul secara permanen di *dropdown* untuk mempercepat proses *reply translation.*



Gambar 3. 86 *Translation settings design alternatives*

Untuk mendukung kedua fitur tersebut, penulis juga merancang halaman "*Language*" di dalam *settings*. Halaman ini berfungsi sebagai pusat pengaturan seluruh bahasa yang dapat dipilih CS agent baik untuk *whole ticket translation* maupun *reply translation*. Dikembangkan beberapa variasi tampilan, termasuk versi dengan *checklist*, dengan ikon *checkmark*, atau dengan ikon tempat sampah. Penulis juga memikirkan berbagai kondisi seperti batas maksimal 10 bahasa, *error state*, dan *tag* bahasa akan muncul ketika pengguna menambahkan atau menghapus bahasa tertentu.



Gambar 3. 87 *Translation settings feedback*

Seluruh ide ini kemudian dipresentasikan pada sesi *UI/UX sync* bersama *Supervisor* dan tim *Product* untuk mendapatkan

masukan lebih lanjut. Beberapa catatan yang muncul antara lain: ikon pada *modal* masih terlalu besar, *padding* tidak konsisten dengan *design system*, dan penggunaan komponen lama yang seharusnya sudah diganti dengan card component yang berlaku sekarang. Selain itu, *Supervisor* mengingatkan bahwa halaman *settings* sudah "*unified*," sehingga halaman bitChat *Settings* harus dipindahkan agar mengikuti struktur *settings* terbaru.

D. Prototype

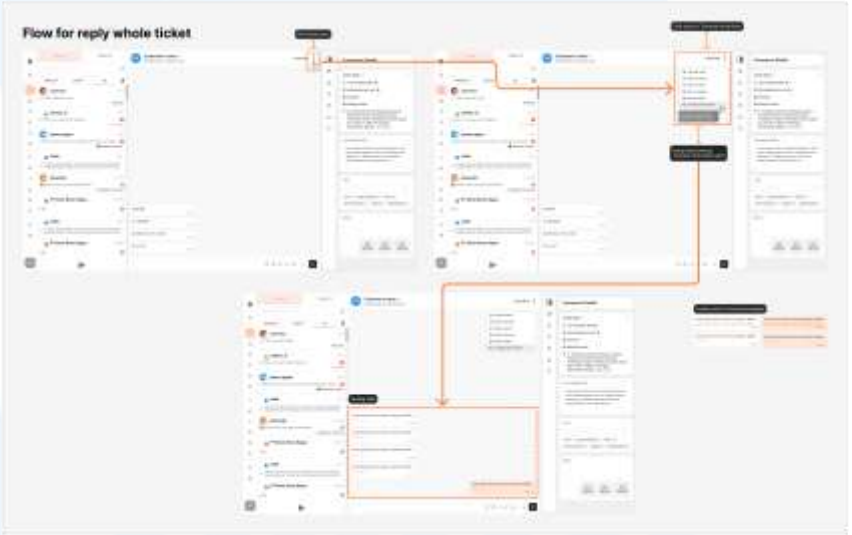
Setelah seluruh alur direvisi berdasarkan masukan dari *Supervisor*, desain mulai difinalisasi. Untuk fitur *Reply Translation*, ditambahkan ikon terjemahan baru di bawah *input field* sebagai pemicu fitur ini. Ikon tersebut dilengkapi *hover state* dan versi ikon yang sudah diperbarui agar konsisten dengan sistem visual bitChat.

Gambar 3. 88 *Reply translations flow 2*

Ketika ikon diklik, muncul *dropdown* dalam keadaan *empty state* yang berisi *English* sebagai bahasa *default* serta opsi *Others* untuk membuka daftar bahasa lengkap. Ketika pengguna memilih *Others*, muncul sebuah *modal* yang berisi *dropdown* seluruh bahasa, sebuah *checkbox* untuk menyimpan bahasa tersebut untuk

penggunaan berikutnya, serta *tooltip* kecil bertuliskan "*You can manage this in the bitChat settings*".

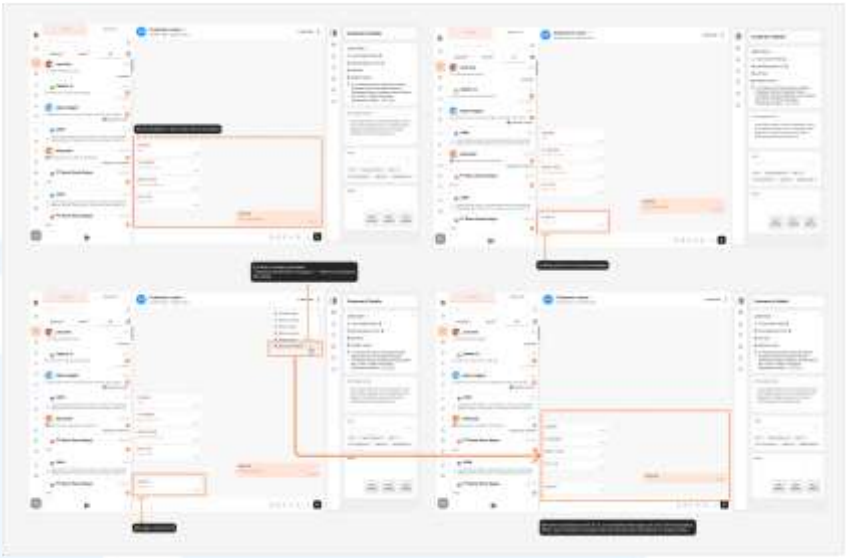
Tombol *Apply* dalam *modal* ini tetap tidak aktif sampai pengguna memilih satu bahasa. Setelah pengguna memilih bahasa dan mengaktifkan opsi penyimpanan, tombol *Apply* dapat ditekan dan sistem langsung menampilkan *loading state* berupa gradasi oranye dan ungu sesuai *design system* bitbybit. Begitu proses selesai, pesan di *input field* berubah menjadi versi terjemahan. Pada tahap ini, dibuat juga *component breakdown* untuk *modal*, *language item*, dan *state variations* agar implementasi lebih mudah dipahami oleh tim *Software Developer*.



Gambar 3. 89 *Translate whole ticket flow 3*

Untuk fitur *Translate Whole Ticket*, opsi terjemahan ditempatkan di *kebab menu* di kanan atas tampilan tiket. Ketika pengguna mengarahkan kursor ke opsi tersebut, muncul *tooltip* berisi penjelasan singkat bahwa fitur ini akan menerjemahkan percakapan ke bahasa *default* dan dapat diatur melalui bitChat *settings*. Setelah diklik, seluruh pesan dalam tiket memunculkan *loading state* bertuliskan "*translating...*" dalam warna oranye muda di bawah teks asli. Setelah proses selesai, setiap pesan menampilkan

dua bagian, teks asli di bagian atas warna hitam dan hasil terjemahan di bawah dengan warna oranye dan gaya miring.



Gambar 3. 90 *Translate whole ticket flow 4*

Untuk pesan baru yang masuk pada tiket yang sudah diterjemahkan, sistem secara otomatis memunculkan *loading state* sebelum menampilkan hasil terjemahan. Jika tiket pernah diterjemahkan sebelumnya dan datanya masih tersimpan, tampilan muncul tanpa proses *loading*. Ketika percakapan sudah dalam kondisi diterjemahkan, teks di *kebab menu* berubah menjadi "*Remove translation*". Opsi ini hanya menghilangkan tampilan terjemahan dari UI tanpa menghapus data terjemahan dari basis data. Supervisor meminta agar *loading state* memproses seluruh pesan secara paralel sehingga tidak muncul satu per satu. Semua perilaku ini kemudian diimplementasi dalam *prototype Figma Make* agar pengembang dapat mencoba alur lengkapnya dan menyalin *code output* untuk setiap komponen.

Gambar 3. 91 *bitChat settings handover file*

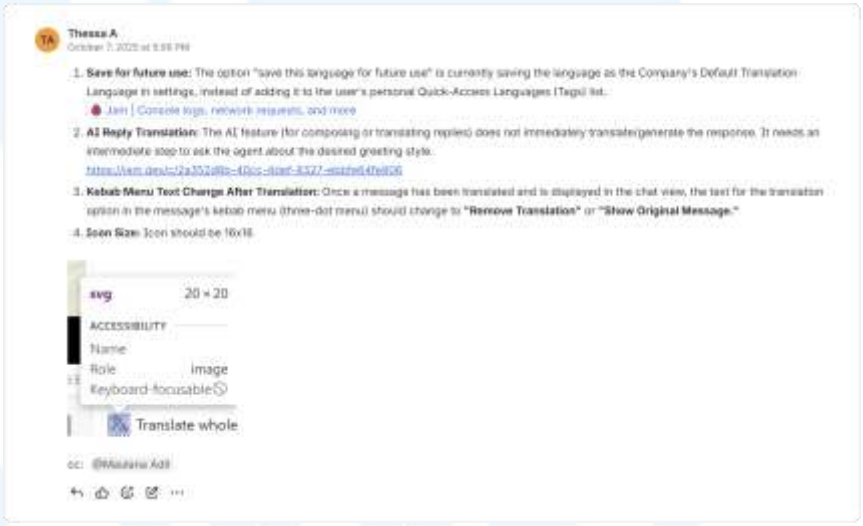
Pada bagian *Settings*, dirancang halaman khusus untuk mengelola bahasa terjemahan sesuai arahan *Supervisor*. Komponen *tags* digunakan untuk menampilkan bahasa yang digunakan pada *Reply Translation*, di mana *English* selalu menjadi *tag* permanen yang tidak dapat dihapus. Pengguna dapat menambah bahasa baru melalui *dropdown* "*Select language to add*" dan menekan tombol *Add*. Bahasa yang ditambahkan akan muncul sebagai *tag* baru dengan ikon "*x"* untuk menghapusnya. Ketika total bahasa mencapai sepuluh, muncul pesan khusus di bawah *dropdown* yang menjelaskan bahwa pengguna harus menghapus salah satu bahasa sebelum menambahkan yang baru.

Semua perubahan pada halaman ini memunculkan *banner* "*Unsaved chang*es" yang meminta pengguna untuk menyimpan atau membatalkan perubahan. Di bagian *Ticket Translation*, pengguna dapat memilih bahasa *default* untuk fitur *Translate Whole Ticket* melalui *dropdown* yang otomatis menggunakan bahasa negara perusahaan sebagai *default*. Setelah seluruh komponen selesai, penulis melanjutkan dengan penyusunan dokumen *handover* dan *Product Requirement Document*.

E. Test

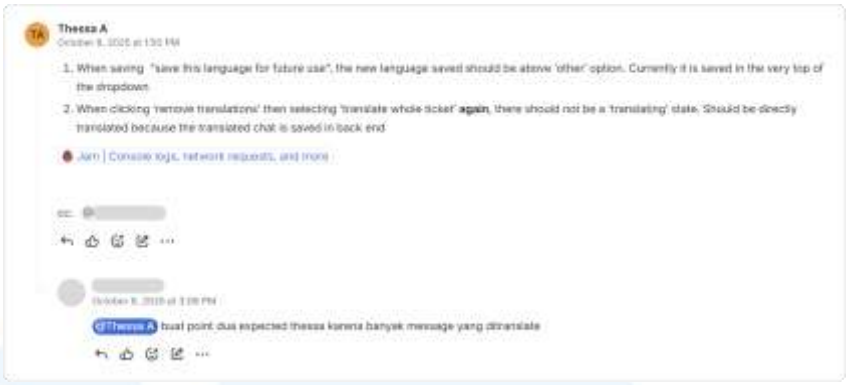
Setelah diserahkan kepada tim *Software Developer* melalui sesi *Handover Sync,* implementasi fitur berlangsung selama satu *sprint*. Ketika versi awal telah siap diuji, tugas ini kembali diberikan melalui Jira untuk menjalankan *testing*. Proses *testing* menemukan

beberapa isu yang perlu diperbaiki. Pada *Reply Translation*, *modal* bahasa sempat menampilkan tombol *Apply* dalam keadaan aktif meskipun belum ada bahasa yang dipilih, sehingga perlu penyesuaian pada kondisi aktif dan tidak aktif. Komponen *tooltip* belum muncul sesuai tata letak yang direncanakan. Selain itu, *loading state* awalnya tidak mengikuti gradasi warna yang telah ditentukan sehingga perlu diperbaiki agar konsisten dengan *design system*. Sementara itu, *dropdown* bahasa sempat tidak menampilkan *English* sebagai opsi paling atas sehingga diminta untuk menyesuaikan penampilannya.



Gambar 3. 92 *Translate whole ticket test 1*

Untuk fitur *Translate Whole Ticket*, *testing* awal menemukan bahwa *loading state* muncul secara paralel. Hal ini menyebabkan pengalaman pengguna terasa lambat dan tidak sesuai arahan *Supervisor*. Penulis kemudian memberikan catatan tambahan agar seluruh proses *loading* dipicu secara bersamaan. Selain itu, opsi di *kebab menu* belum berubah menjadi "*Remove translation*" setelah percakapan berhasil diterjemahkan sehingga penulis menambahkan catatan tentang perubahan teks menu berdasarkan konteks fitur.



Gambar 3. 93 *Translate whole ticket test 2*

Pada bagian *Settings*, *testing* menunjukkan bahwa pesan batas maksimal sepuluh bahasa muncul meskipun daftar belum mencapai jumlah tersebut. Kondisi pemicu pesan kemudian diperbaiki agar hanya tampil tepat ketika jumlah bahasa mencapai sepuluh. *Banner* "*Unsaved changes*" awalnya tidak muncul untuk beberapa jenis perubahan, misalnya ketika pengguna menghapus *tag* melalui ikon "*x*." Setelah diperbaiki, *banner* berhasil muncul untuk seluruh perubahan yang memengaruhi konfigurasi bahasa. Penulis juga memastikan bahwa perubahan yang disimpan dapat bertahan antar sesi sesuai persyaratan teknis arahan *Supervisor.*

Setelah seluruh perbaikan diterapkan oleh tim *Software Developer*, *testing* ulang menunjukkan bahwa alur interaksi sudah berjalan stabil, tampilan konsisten dengan desain, dan seluruh fungsi memenuhi persyaratan yang tercantum dalam PRD. Setelah seluruh revisi diselesaikan dan dinyatakan lengkap, fitur ini akhirnya *deploy* dan sudah *live* dapat langsung digunakan oleh pengguna.

3.3.2.4 Proyek *Commerce payment method improvement*

Proyek ini bertujuan untuk meningkatkan konsistensi, kejelasan, dan efisiensi pada alur pengelolaan metode pembayaran di dalam sistem. Proyek dimulai dari *brief Product Designer* yang menjadi PIC bitCommerce, yang melihat bahwa fitur *Custom Payment Method* belum dimanfaatkan secara optimal. Fitur ini menyediakan *fields* penting seperti *payment name, account holder*

name, account number, payment instructions, dan *media*, namun belum ada cara yang efisien bagi *user* untuk memanfaatkan data tersebut dalam berkomunikasi dengan *customer*.

Mencakup perancangan ulang komponen terkait pembayaran, perbaikan pengalaman pengguna di *halaman Create Order* dan *Settings*, serta optimasi proses pengiriman detail pembayaran melalui bitChat. *Brief* menekankan tiga area perkembangan: *redesign flow* pemilihan *payment method* untuk meningkatkan *readability* dan *usability CS Agent*, penambahan halaman *Order Settings* untuk *centralized payment method management*, dan perbaikan tampilan *payment method data* di *Payment card*. Seluruh kegiatan dalam proyek ini berfokus pada penyelesaian masalah yang ditemukan di lingkungan produksi, sehingga hasil akhirnya dapat langsung mendukung kebutuhan operasional tim dan pengguna.

A. Empathize

Pada tahap awal, penulis mencoba langsung alur yang berkaitan dengan pengelolaan metode pembayaran untuk memahami kendala yang muncul dari sudut pandang pengguna. Selama proses tersebut, penulis menemukan bahwa penambahan *payment method* baru hanya bisa dilakukan melalui halaman *Create Order*, sehingga alurnya terasa tidak efisien.

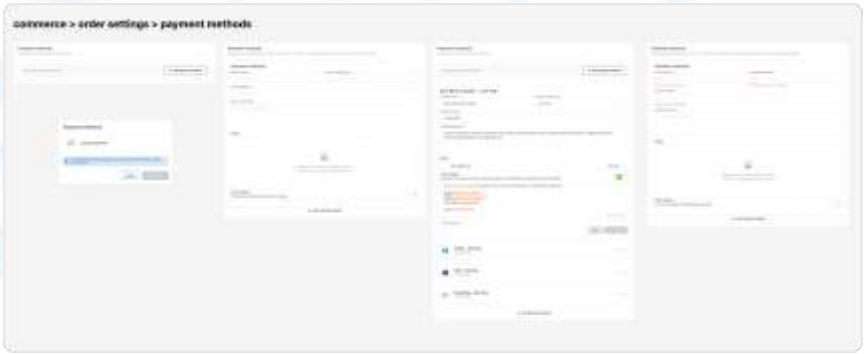
Selain itu, metode pembayaran yang sudah tersimpan tidak dapat dihapus atau diperbarui, sehingga daftar pada *dropdown* terus menumpuk dan menyulitkan pencarian. Saat mengirim payment details melalui bitChat, prosesnya juga masih sepenuhnya manual meskipun data sebenarnya sudah tersimpan di sistem. Observasi ini menunjukkan bahwa pengguna membutuhkan alur yang lebih ringkas, terpusat, dan otomatis agar pekerjaan dapat dilakukan dengan lebih cepat dan minim kesalahan.

B. *Define*

Penulis mulai memetakan inti masalah yang dihadapi pengguna terkait pengelolaan *payment method*. Berdasarkan penjelasan awal dari tim *Product*, tidak adanya pusat pengaturan pembayaran membuat pengguna harus mengisi ulang informasi yang sama setiap kali membuat *order*. Proses ini memperlambat pekerjaan dan meningkatkan risiko kesalahan *input*. Selain itu, *CS Agent* juga tidak memiliki cara cepat untuk mengirim informasi pembayaran melalui bitChat, sehingga komunikasi dengan pelanggan menjadi kurang efisien. Dari temuan tersebut, kebutuhan utama dapat dirumuskan sebagai penyediaan satu halaman terpusat untuk mengelola metode pembayaran, meningkatkan alur pemilihan *payment method* saat membuat *order*, serta menyediakan mekanisme pengiriman detail pembayaran secara cepat di bitChat melalui *chat template* yang terotomatisasi.

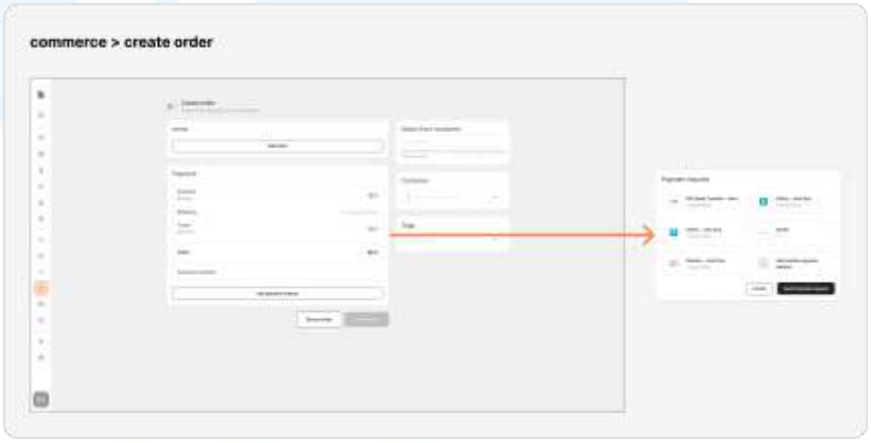
C. Ideate

Penulis mulai menyusun desain yang bisa menjadi solusi untuk permasalahan ini. Pada bagian *settings*, penulis merancang sebuah *section* baru pada jalur *commerce* > *order settings* > *payment methods* untuk menambah, mengubah, dan menghapus metode pembayaran di satu tempat. Penulis juga merancang *empty state*, *edit state*, dan *filled state* untuk memastikan alur pengelolaan data tetap jelas dan mudah dipahami.



Gambar 3. 94 *Order settings draft 1*

Untuk proses *edit*, *section* akan melakukan *expand* dan menampilkan beberapa *input field* seperti nama pembayaran, nama pemilik akun, nomor rekening, instruksi pembayaran yang bersifat opsional, serta unggahan media. Menambahkan fitur baru berupa pembuatan *chat template* dengan *variable*, memanfaatkan *modal* yang sudah tersedia dalam *design system* bitbybit. Selain itu, penulis mengusulkan mekanisme pendeteksian kata kunci, misalnya nama bank seperti BCA, sehingga logo bank dapat muncul secara otomatis setelah metode pembayaran disimpan. Setelah didiskusikan dengan *Supervisor*, cara ini memungkinkan untuk diimplementasikan.



Gambar 3. 95 *Commerce create order draft 1*

Pada halaman *create orders*, penulis mengevaluasi desain sebelumnya yang mengharuskan pengguna menggulir daftar yang panjang untuk memilih metode pembayaran. Sebagai solusi, penulis mengonsep ulang alurnya menjadi sebuah *modal* yang menampilkan seluruh metode pembayaran secara lebih terstruktur, dilengkapi logo, nama metode, nama pemilik akun, dan nomor rekening. *Modal* ini juga menyediakan opsi untuk menambah metode pembayaran baru menggunakan alur yang sama seperti pada *settings*.

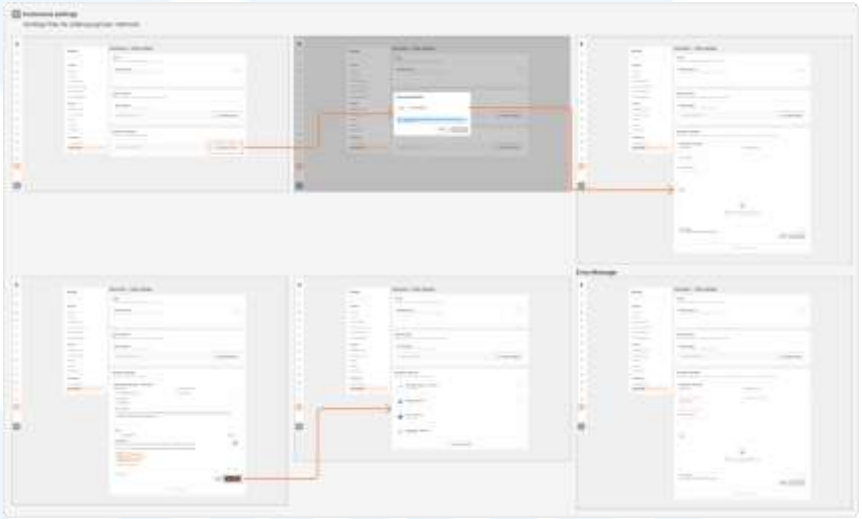
Gambar 3. 96 bitChat *send payment details draft 1*

Penulis juga mengembangkan fitur tambahan untuk mendukung *CS agent* dalam mengirim *payment inquiries* untuk pelanggan yang memiliki *ongoing orders*. Fitur ini memanfaatkan *dynamic variable* untuk *WhatsApp* sehingga informasi pembayaran dapat terisi otomatis sesuai data yang tersedia. Alurnya dimulai dari ikon pembayaran pada *input field* di bitChat yang memicu munculnya *modal* pemilihan metode pembayaran. Setelah agen memilih metode dan mengisi nominal pembayaran, sistem menampilkan pesan siap kirim di *input field* berdasarkan *chat template* yang sudah ditentukan di *settings*. Ide ini dirancang agar proses pengiriman informasi pembayaran menjadi lebih ringkas, akurat, dan minim kesalahan manual.

Gambar 3. 97 *Custom payment method feedback 1*

Setelah ide awal dipresentasikan kepada *Supervisor* dan tim *Product* dalam sesi *UI/UX Sync*, penulis menerima beberapa

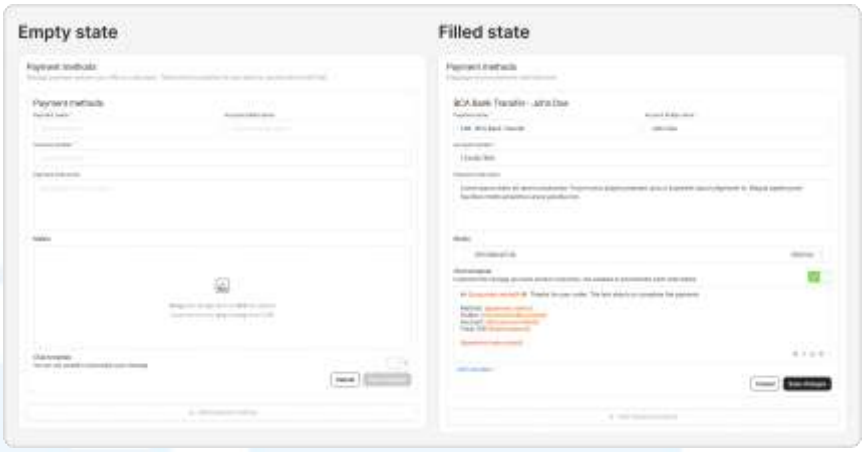
masukan. Salah satu poin utama adalah terkait pengiriman *payment inquiries*. Pada bagian ikon baru atau *modal* baru di *input field*, tim menyarankan untuk memanfaatkan *Order History* yang sudah tersedia di *customer detail panel* pada sisi kanan bitChat. Ini dianggap jauh lebih efisien karena agent tidak perlu berpindah konteks dan alurnya menyatu dengan struktur yang sudah ada. Selain itu, *Product Designer* menegaskan bahwa saat order dibuat, *customer* seharusnya sudah memilih metode pembayaran. Fitur ini tidak perlu lagi menanyakan *payment method* dari awal. Nantinya tombol yang dipakai cukup memunculkan pesan yang sudah *prefilled* di *input field*, lengkap dengan media seperti *QR Code.*



Gambar 3. 98 *Order settings draft 2*

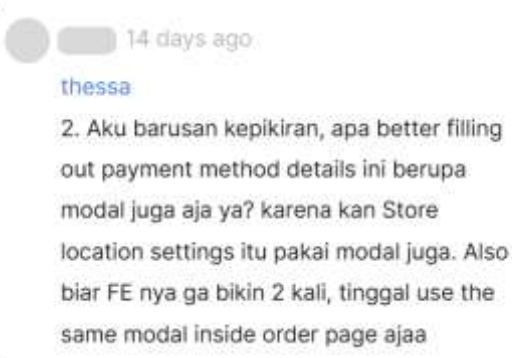
Berdasarkan *feedback* tersebut, penulis mulai membuat *draft* kedua dan mengembangkan alur desain yang lebih rinci. Untuk menambah atau mengubah metode pembayaran, membuka *modal* "*Add payment method*". Setelah *modal* dibuka, *user* akan menemukan opsi "*Custom payment method*" serta *banner* yang menjelaskan integrasi dengan penyedia pembayaran. Ketika *user* memilih opsi *custom payment method*, *modal* akan menutup dan *section* metode pembayaran akan terbuka dengan isian detail seperti

Payment name, *Account holder name*, *Account number*, *Payment instructions*, *Media*, dan *Chat template*.



Gambar 3. 99 *Payment method modal states*

Di sini penulis menyesuaikan *User Interface* untuk unggahan media agar sesuai dengan *design system* terbaru. Bila *user* memilih untuk menambahkan *template* pesan, tersedia opsi untuk "*add variable*". Tombol "*Save changes*" hanya aktif jika seluruh isian wajib sudah lengkap. Penulis juga menyiapkan *error state* seperti "*Payment name/Account holder name/Account number is required*" serta kondisi *disabled* apabila syarat belum terpenuhi.



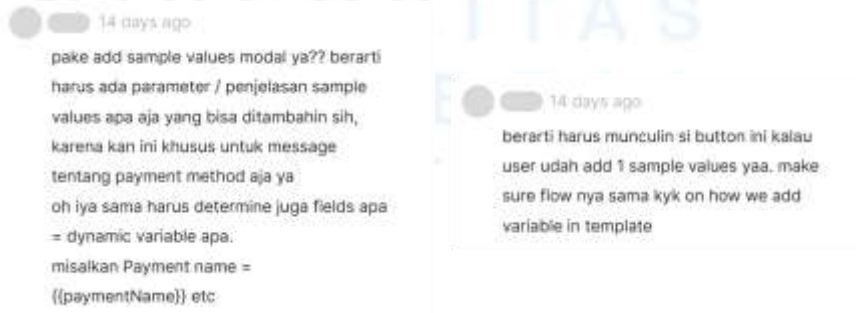
Gambar 3. 100 *Custom payment method feedback 2*

Product Designer pada tim memberikan masukan untuk menyederhanakan implementasi di sisi *front-end*. Tim menyarankan agar isian *payment method* disatukan dalam bentuk *modal* yang sama, baik di halaman *settings* maupun di *order page*, agar tidak perlu membangun dua komponen berbeda di bagian *front-end*.

Banner integrasi juga direvisi: opsi Xendit muncul secara *default*, *banner* tidak lagi digunakan. Jika *user* belum terhubung dengan Xendit dan mengklik opsi tersebut, mereka akan langsung diarahkan ke halaman integrasi melalui *tab* baru dan muncul *modal* integrasi Xendit, serupa alur integrasi WhatsApp yang sudah ada.

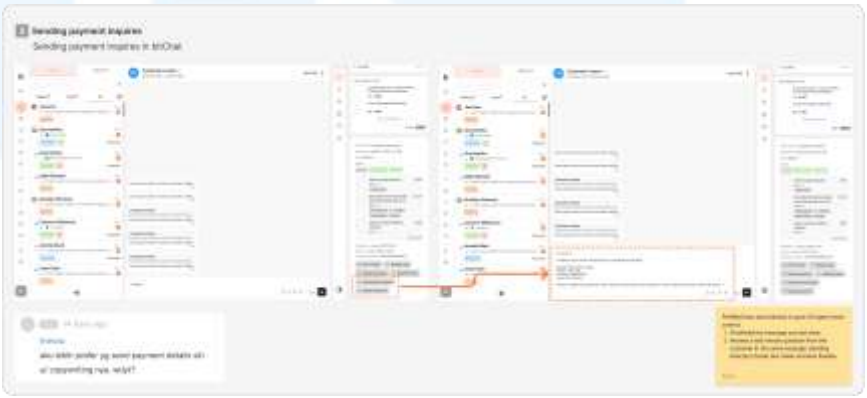
Gambar 3. 101 *Commerce create order draft 2*

Pada bagian *Create Order page*, penulis juga mengembangkan alur pemilihan metode pembayaran. Ketika *user* memilih "*Add payment method*", *modal* akan muncul berisi seluruh metode pembayaran yang telah disimpan. Jika *user* memilih satu metode, *modal* akan tertutup dan informasi tersebut otomatis tampil di kartu order. Bila *user* ingin membuat metode baru, *modal* "*Add payment method*" akan terbuka dengan isian yang sama seperti pada halaman *settings*.



Gambar 3. 102 *Custom payment method feedback 3*

Pada tahap ini, penulis mendapatkan masukan lanjutan terkait pengaturan variabel dalam *chat template*. *Product Designer* mempertanyakan apakah variabel akan dibatasi, bagaimana nilai variabel ditambahkan, dan apakah perlu *modal* terpisah seperti "*add sample values*". *Product Designer* juga menekankan pentingnya menentukan *dynamic fields* seperti "{{*paymentName*}}" dan memastikan tombol untuk menambah variabel hanya muncul setelah *user* menambahkan *sample values*, mengikuti pola yang sudah digunakan dalam fitur *template*.



Gambar 3. 103 bitChat *send payment details draft 2*

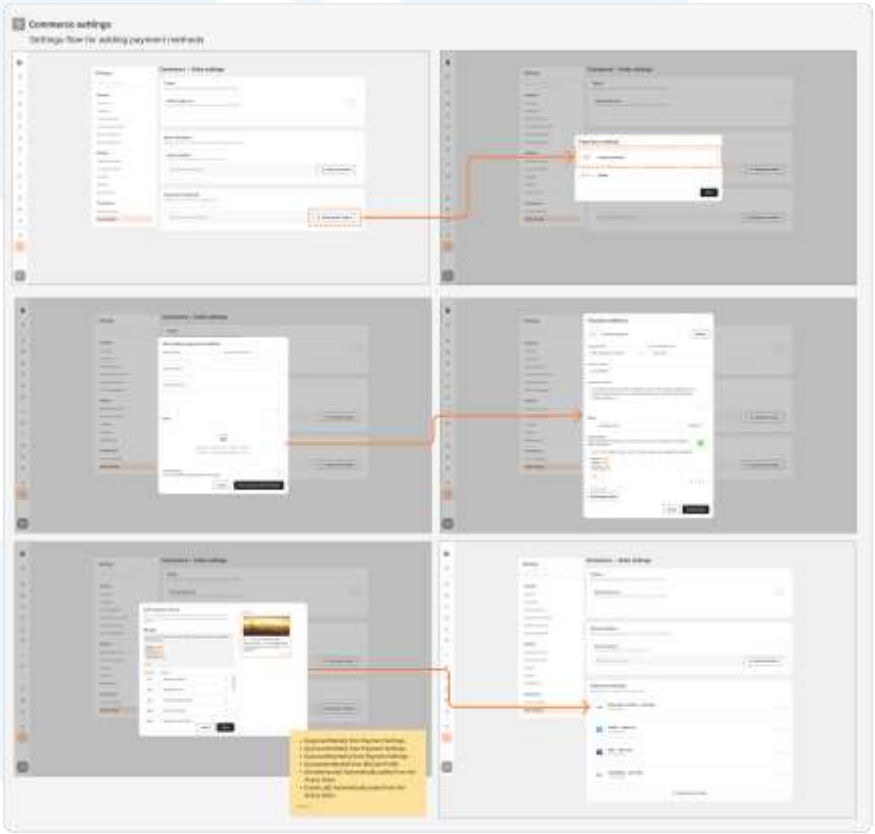
Untuk bagian pengiriman *payment details* di bitChat, penulis memanfaatkan *order history* sesuai arahan *Product Designer*. Penulis menambahkan tombol "*Send payment details"* dibagian itu. tersebut dipilih, *template* pesan yang sudah ditentukan sebelumnya akan otomatis muncul di *input field*. Penulis sengaja tidak membuat pesan terkirim otomatis agar *CS agent* dapat membaca ulang atau menambahkan konteks terakhir sebelum mengirim.

Gambar 3. 104 bitChat *send payment details draft 3*

Media yang ditautkan dalam metode pembayaran di *settings* juga otomatis terlampir. Bila *CS agent* tidak ingin mengirim gambar tersebut, mereka bisa menghapusnya secara manual. Namun penulis mencatat kendala yang ada di bitChat: jika gambar dihapus dari *input field*, teks di dalamnya ikut terhapus. Penulis menandai hal ini agar dapat diteruskan sebagai permintaan penyesuaian kepada tim *Software Developer*.

Gambar 3. 105 *Custom payment method feedback 4*

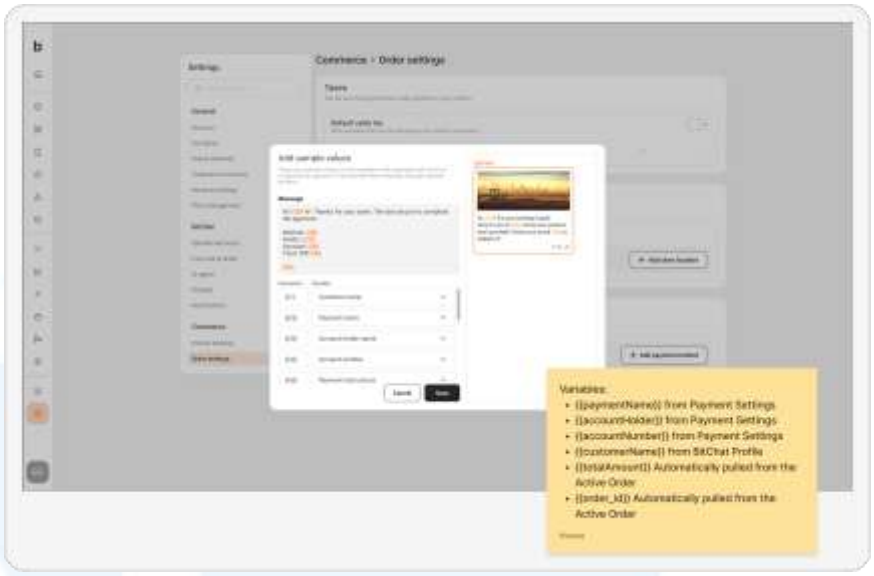
Pada bagian ini mendpatkan *feedback* dari *Product Designer* bertanya apakah alur ini sudah diuji memakai fitur *quick reply* di bitChat, karena mekanismenya dianggap serupa, termasuk dukungan gambar. Setelah penulis menguji fitur tersebut, hasilnya sesuai ekspektasi: gambar dapat dikirim, namun menghapus gambar juga menghapus teks. *Product Designer* mengonfirmasi bahwa alurnya sudah tepat dan penyesuaian kecil di sisi penghapusan media dapat dibahas bersama tim terkait.



Gambar 3. 106 *Order settings draft 3*

Setelah menerima *feedback* dari *Product Designer*, penulis melanjutkan pekerjaan ke *draft* ketiga. Pada bagian *settings*, ketika *user* memilih "*Add payment method*", seluruh proses kini berlangsung dalam *modal*, bukan lagi melalui section yang melakukan *expand*. *Modal* pertama juga sudah direvisi: *banner* integrasi dihilangkan dan diganti dengan opsi Xendit sebagai bagian

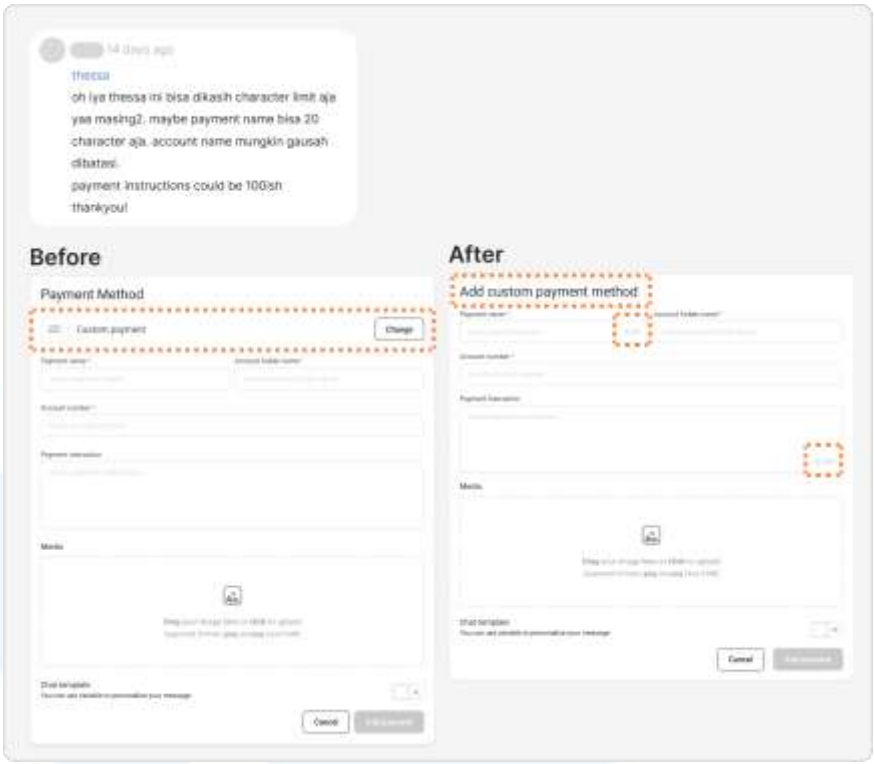
dari daftar pilihan. Setelah itu barulah *user* diarahkan ke pertanyaan yang sama seperti sebelumnya.



Gambar 3. 107 *Add sample values modal*

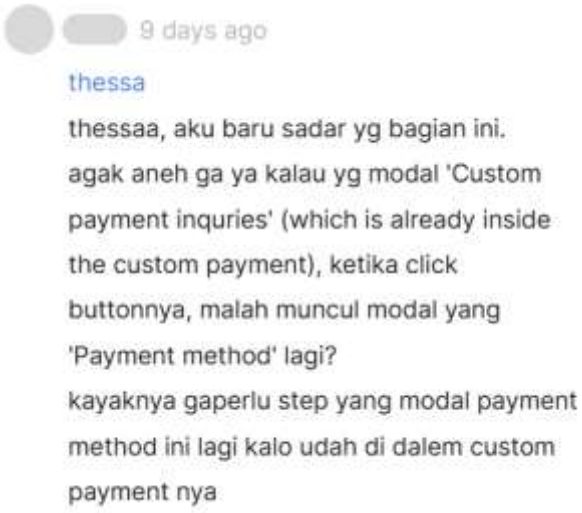
Penulis juga menambahkan *modal* baru untuk "*Add sample values*", lengkap dengan parameter dan *dynamic fields* yang telah disetujui oleh *Product Designer*, yaitu: *{{paymentName}}, {{accountHolder}}, {{accountNumber}}, {{customerName}}, {{totalAmount}},* dan *{{order_id}}*, masing-masing ditarik dari konfigurasi *Payment Settings*, bitChat *Profile*, atau *Active Order*.

Revisi berikutnya dilakukan pada *Create Orders page*. *Flow* pada halaman ini dirapikan agar selaras dengan pola *modal* yang dipakai di *settings*, sehingga kedua area tersebut konsisten dalam interaksi maupun komponen. Kemudian pada fitur pengiriman *Payment details* di bitChat, penulis mulai mendefinisikan parameter dan kondisi agar tombol "*Send payment details*" hanya muncul apabila order tersebut berstatus *Unpaid* dan *Unfulfilled*. Saat tombol dipilih, *template* pesan akan langsung muncul dalam keadaan *pre-filled* di *input field*, dengan *fallback text* apabila metode pembayaran tersebut tidak memiliki *template*.



Gambar 3. 108 *Payment method modal before, after* dan *feedback*

Dalam *draft* ketiga ini, *Prodict Designer* memberikan beberapa masukan tambahan. Salah satunya adalah menerapkan batas karakter pada isian *modal "Add payment method*": *Payment name* dibatasi sekitar 20 karakter, *Account holder name* tanpa batas, dan *Payment instruction* sekitar 100 karakter. Selain itu menyamakan komponen UI yang sebelumnya masih tidak konsisten, termasuk memperbarui *header* yang sebelumnya masih memakai gaya lama.

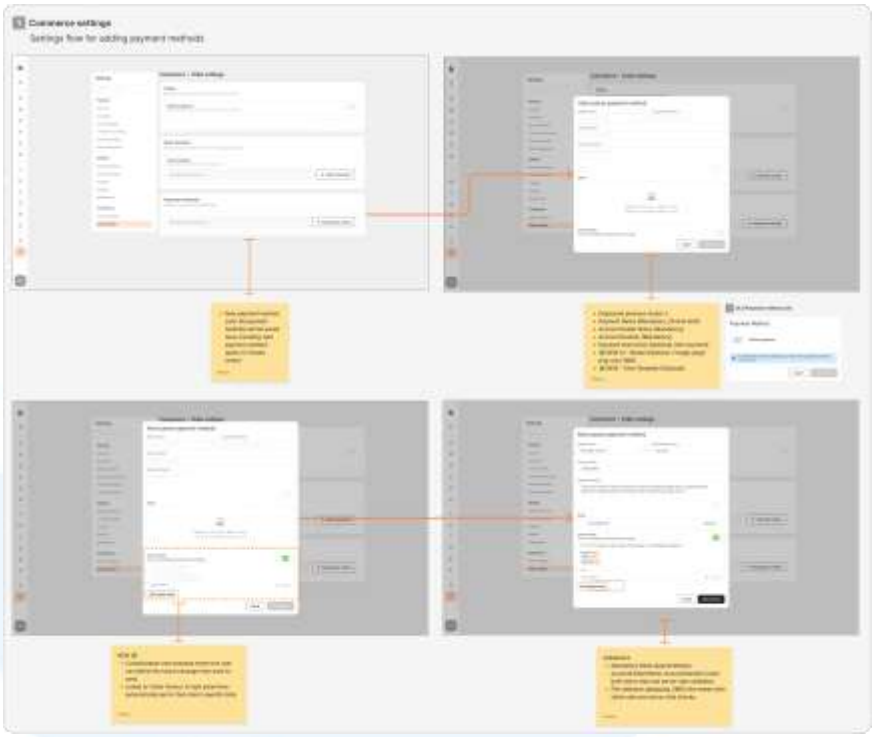


Gambar 3. 109 *Custom payment method feedback 5*

Selain itu, terdapat evaluasi terhadap *flow* terkait *modal*. *Product Designer* menilai bahwa membuka *modal* "*payment method*" perlu diringkas agar tidak berputar-putar. Penambahan *error state* ketika *user* belum mengisi *sample values* dan *disable* tombol *continue. State* ini diperlukan sehinga langkah berikutnya tidak dapat dilanjutkan sebelum seluruh syarat terpenuhi.

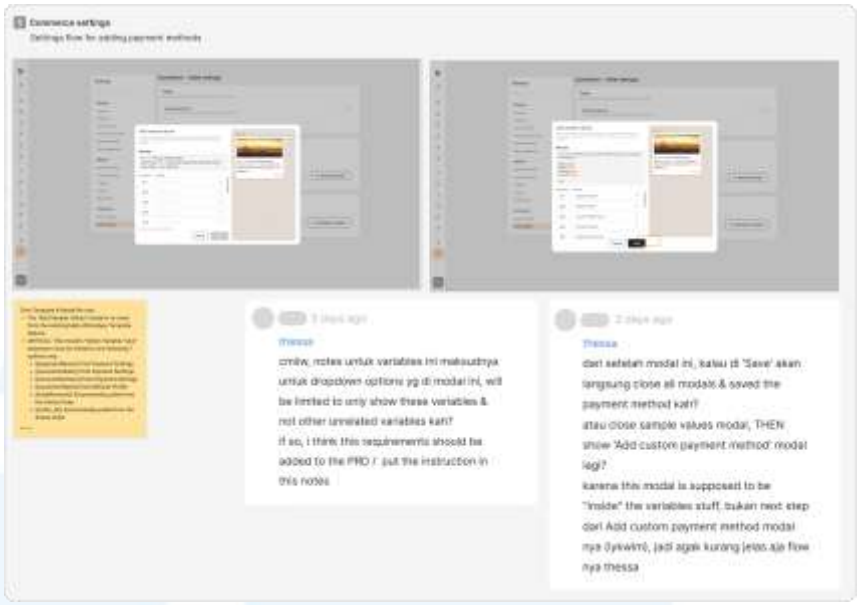
D. Prototype

Setelah seluruh masukan tersebut terkumpul, penulis mulai menyusun *handover file* sesuai struktur baru. Termasuk pembetulan *flow* untuk menghapus step awal untuk memilih *Custom payment method* atau Xendit. Menambahkan batas karakter pada *input field* di *modal* serta menambahkan *modal* pendukung pada "*Add sample values*" untuk melengkapi *flow*.



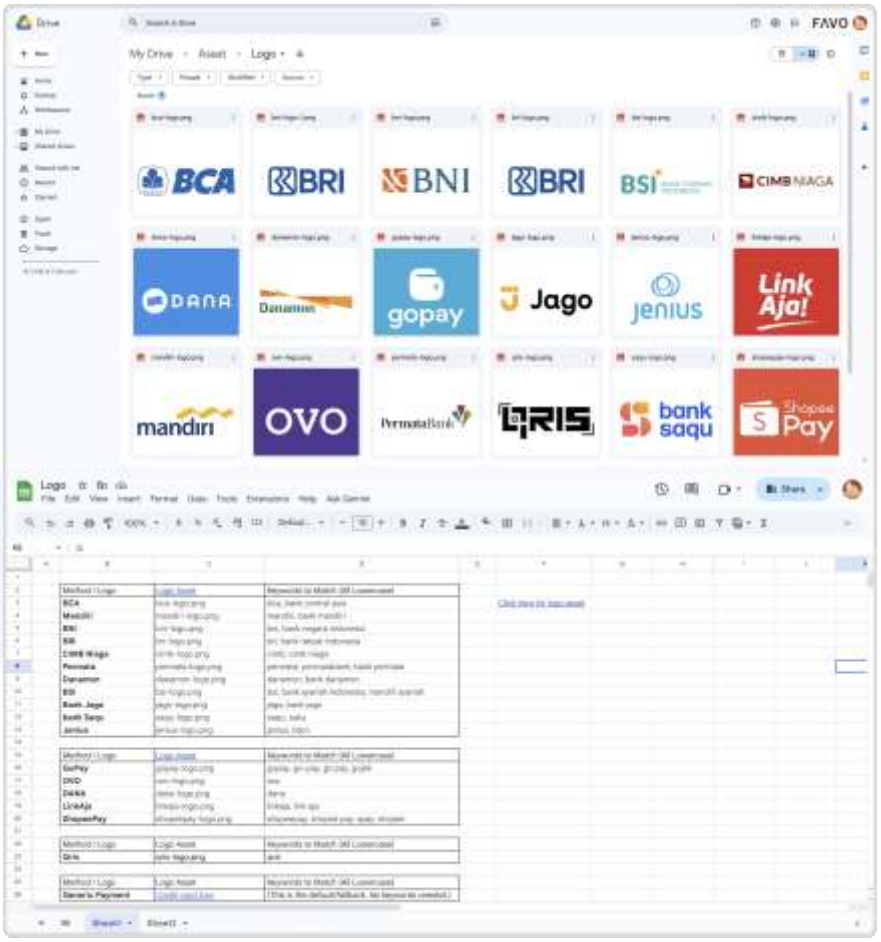
Gambar 3. 110 *Order settings handover file*

Untuk bagian *settings*, penulis membuat rangkaian *flow* lengkap untuk proses penambahan metode pembayaran, termasuk detail komponen pada setiap langkah. Pada *empty state*, ditentukan bahwa seluruh metode pembayaran akan ditampilkan dalam bentuk *new payment method card*. Ketika *user* memilih "*Add new payment method*", *modal* baru berisi daftar isian terbaru: *Payment name*, *Account holder name*, *Account number*, *Payment instruction*, unggahan media, serta area *Chat Template* sebagai fitur baru. *Chat template* ini dapat dikembangkan oleh *user* dan terhubung dengan data order, yang kemudian ditarik secara otomatis saat mengirim detail pembayaran melalui *Order History* di panel kanan.



Gambar 3. 111 *Add sample values feedback*

Dalam *modal* "*Add Sample Values*", penulis mengikuti arahan untuk melakukan *reuse* dari *modal* yang sudah ada pada fitur Meta WhatsApp *Template*. *Dropdown "Select Variable Type*" dibatasi hanya pada tujuh variabel yang relevan. Pada bagian ini, *Product Designer* mengingatkan bahwa *dropdown* harus membatasi pilihan hanya pada variabel terkait pembayaran, dan catatan tersebut perlu dicantumkan di PRD maupun *handover* agar implementasinya tidak keliru. *Product Designer* juga meminta kejelasan mengenai *flow* setelah *modal* "*Add Sample Values*" disimpan: apakah *modal* langsung menutup semua lapisan atau kembali ke *modal* sebelumnya. Catatan tersebut diminta diperjelas agar alur lebih mudah diimplementasikan oleh tim *Software Developer*.

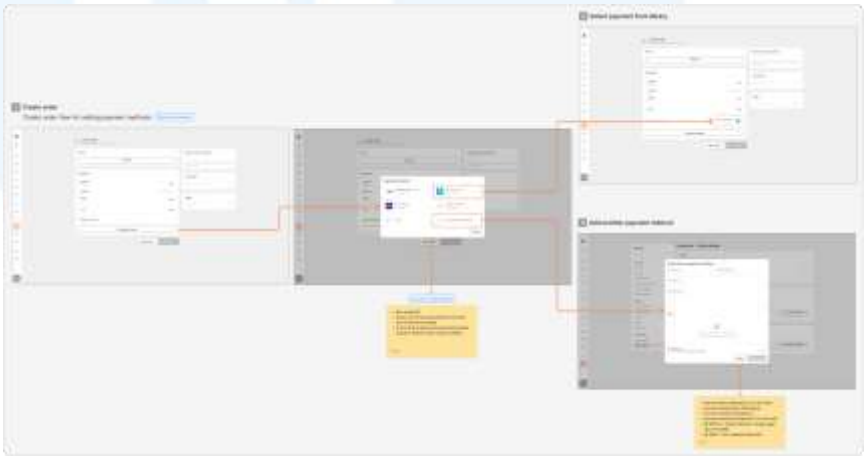


Gambar 3. 112 *Custom payment method assets*

Penulis juga menyiapkan aset logo yang akan digunakan sebagai ikon metode pembayaran ketika *user* menyimpan sebuah *custom payment method*. Sesuai arahan *Supervisor*, fokus awal diarahkan pada metode pembayaran di Indonesia. Penulis membuat daftar ikon serta kata kunci yang terasosiasi dengan setiap logo agar *backend* dapat mencocokkan *input user*, dengan logo yang benar setelah sistem melakukan *case conversion*. Kategori yang disiapkan mencakup bank konvensional, bank digital atau dompet elektronik, opsi QRIS, serta *fallback* ikon berupa kartu kredit generik apabila tidak ditemukan kecocokan.

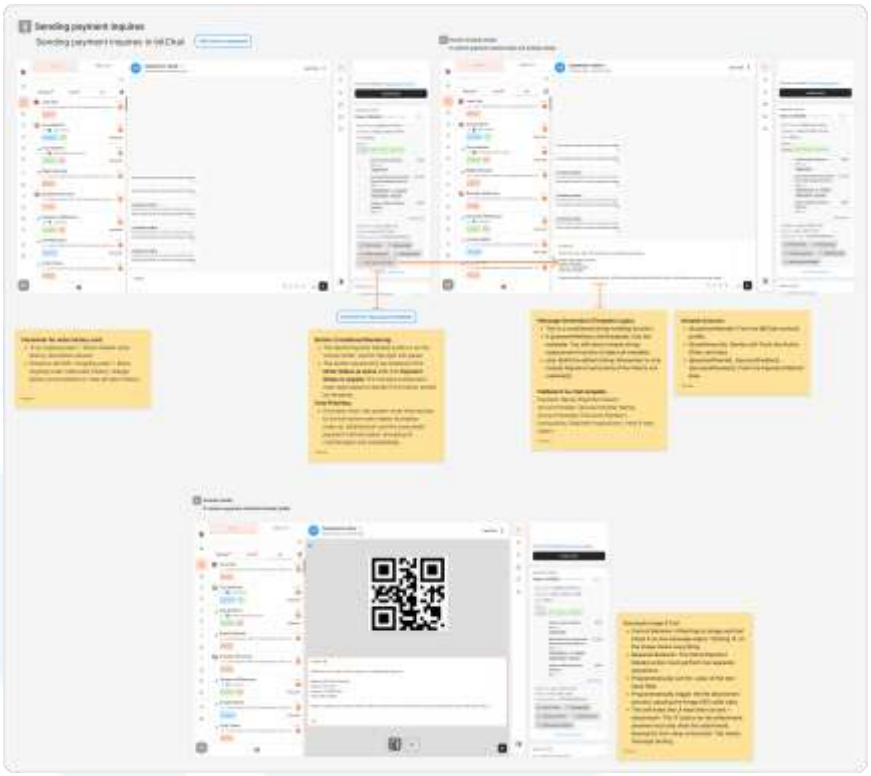
Setelah menerima *feedback* tambahan dari *Product Designer*, penulis melanjutkan ke *draft* ketiga dengan menyesuaikan alur agar lebih selaras dengan *design system* dan kebutuhan teknis.

Pada bagian *settings*, proses penambahan metode pembayaran sepenuhnya dipindahkan ke dalam *modal*, sesuai arahan sebelumnya. *Modal* awal tidak lagi menampilkan *banner*, dan opsi Xendit ditambahkan sebagai alternatif integrasi. Seluruh komponen di dalam *modal* diseragamkan mulai dari struktur *header*, *copywriting*, hingga *input field*. Selain itu, penulis menambahkan *modal* "*Add Sample Values*" serta mengisi daftar *dynamic variable* sesuai arahan *Product Designer.* Variabel ini akan muncul sebagai pilihan khusus di dalam *modal* tersebut.



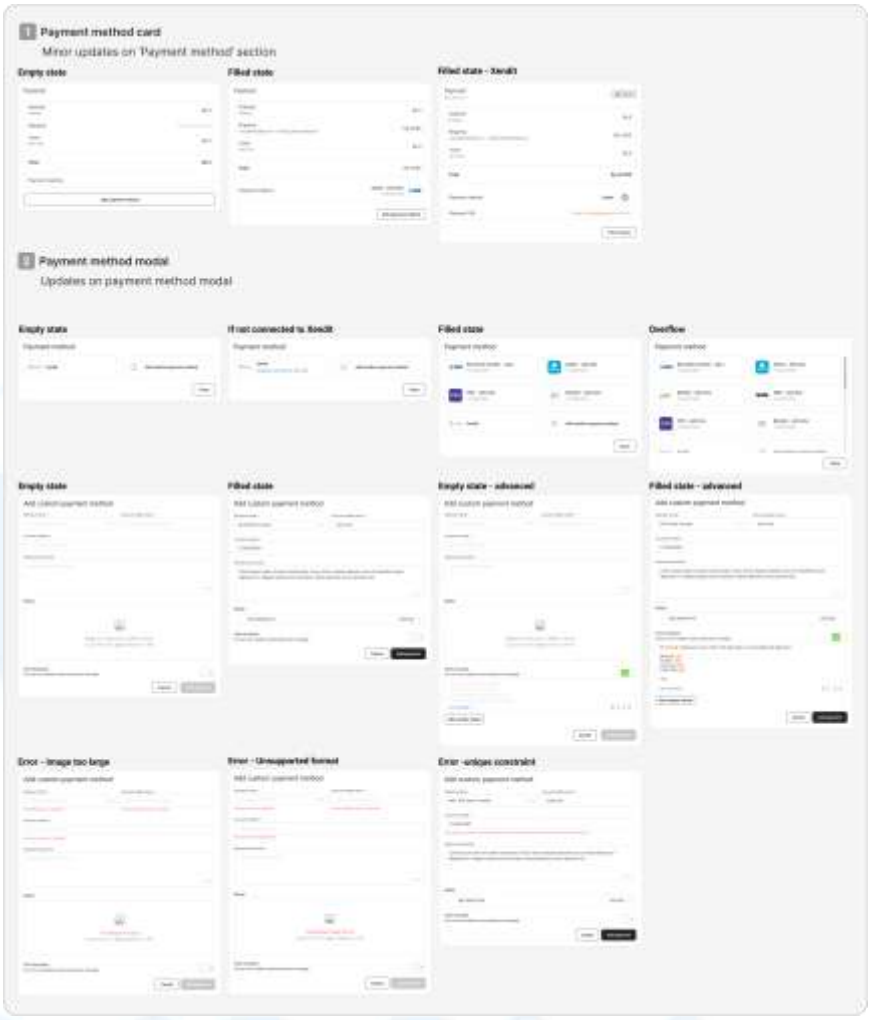
Gambar 3. 113 *Create order handover file*

Pada bagian *Create Order*, penulis menyesuaikan kembali alur agar menggunakan *modal* yang sama seperti di *settings*. Saat pengguna menekan tombol "*Add payment method"*, *modal* akan menampilkan seluruh metode pembayaran yang sudah disimpan berikut opsi integrasi Xendit. Jika jumlah pilihan melebihi enam, *modal* akan bersifat *scrollable* untuk menjaga keterbacaan. Ketika pengguna memilih "*Add another payment method*", *modal* tidak lagi menampilkan langkah tambahan seperti pemilihan "*Custom payment"* atau Xendit. Alurnya kemudian mengikuti struktur yang sama seperti di *settings*, termasuk penambahan *error state* pada *modal* "*Add Sample Values*" sesuai arahan *Product Designer*.



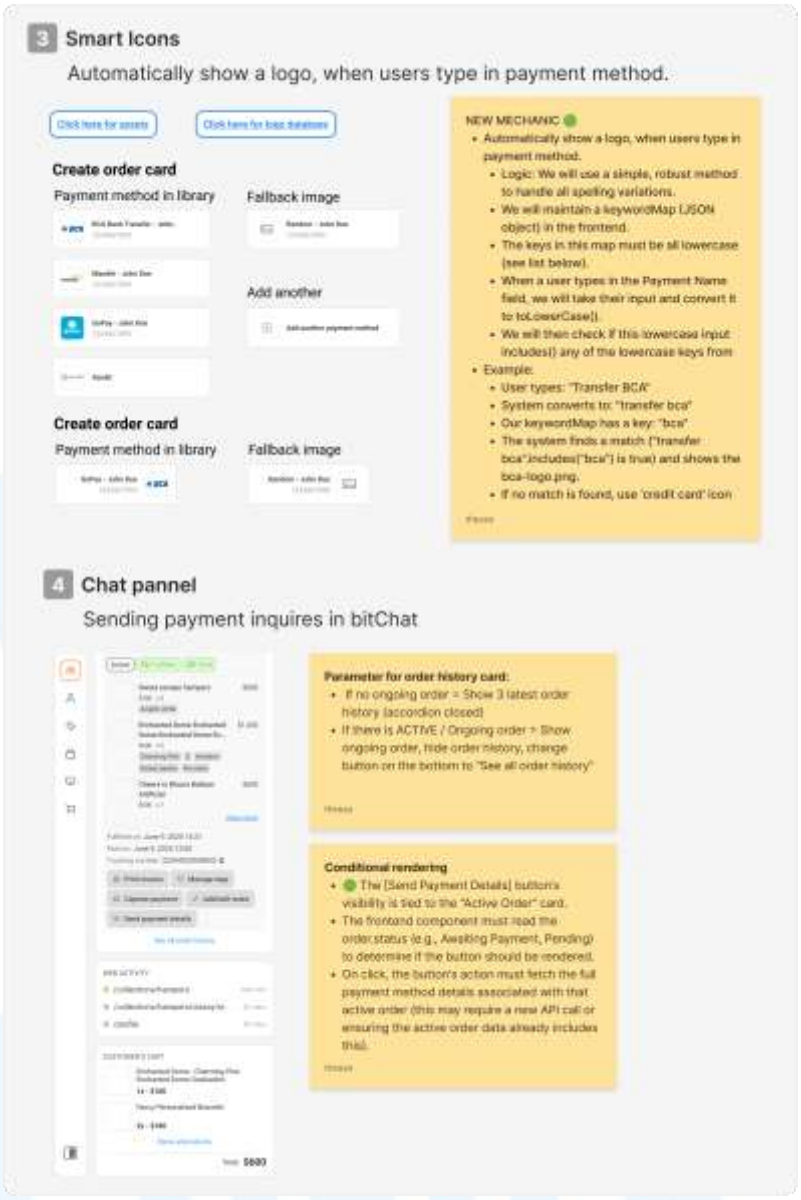
Gambar 3. 114 bitChat *send payment details handover file*

Untuk fitur pengiriman *payment inquiries* melalui bitChat, alurnya tidak mengalami perubahan berarti karena rancangan sebelumnya sudah sesuai. Tombol "*Send Payment Details*" tetap bergantung pada status pesanan yang tampil pada *order history* di panel kanan. Jika pesanan berstatus *unpaid* dan *unfulfilled*, tombol akan muncul dan ketika ditekan akan memunculkan pesan *prefilled* sesuai *chat template* yang sudah ditentukan di *settings*, berikut media yang terlampir jika tersedia. Terakhir, menyiapkan *fallback text* jika metode pembayaran tersebut tidak memiliki *chat template*.



Gambar 3. 115 *Component breakdown handover file 1*

Sesuai permintaan *Product Designer* agar setiap komponen dijabarkan lebih rinci, penulis kemudian memecah *breakdown* komponen ke dalam empat bagian terpisah pada berkas *handover*. Bagian pertama berfokus pada *Payment Method Card* dengan variasi *empty state*, *filled state*, dan kondisi khusus apabila Xendit terhubung. Bagian kedua menjelaskan *Add Custom Payment Method*, mencakup seluruh *state* seperti *empty*, *filled*, *advanced*, serta berbagai skenario *error* termasuk ukuran gambar dan format yang tidak didukung.



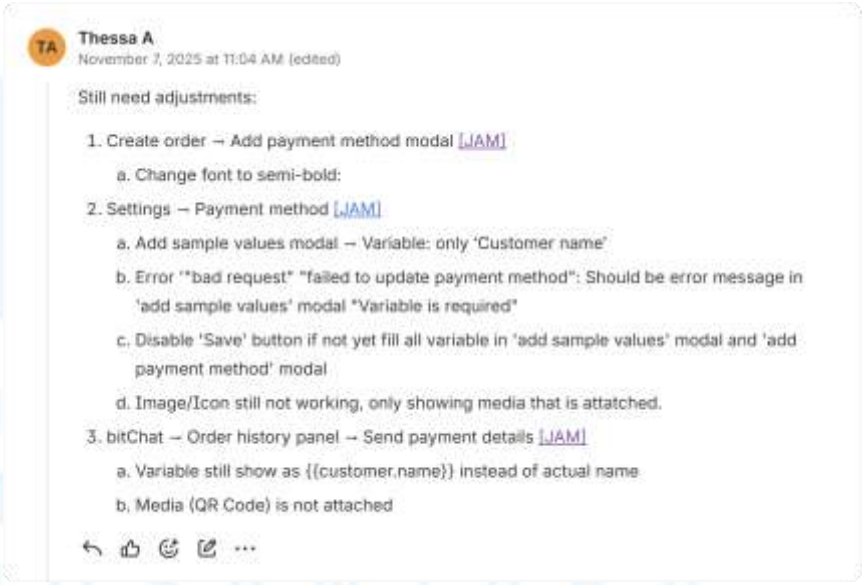
Gambar 3. 116 *Component breakdown handover file 2*

Bagian ketiga membahas *Smart Icons*, yaitu mekanisme untuk menampilkan logo secara otomatis berdasarkan kata kunci pada kolom *Payment Name*. Penulis menyiapkan logika berbasis *keywordMap* yang bekerja dengan mencocokkan huruf kecil dari *input* pengguna dengan daftar kata kunci yang sudah ditetapkan. Jika tidak ditemukan kecocokan, sistem menampilkan ikon *credit card* sebagai *fallback*. Penulis menggunakan bantuan *AI tools* untuk memastikan penjelasan teknis pada bagian ini mudah dipahami oleh

tim *Software Developer*. Bagian keempat menguraikan komponen *Chat Panel* untuk fitur pengiriman *payment inquiries* di bitChat, termasuk parameter pada *order history card* serta kondisi munculnya tombol "*Send Payment Details*".

E. Test

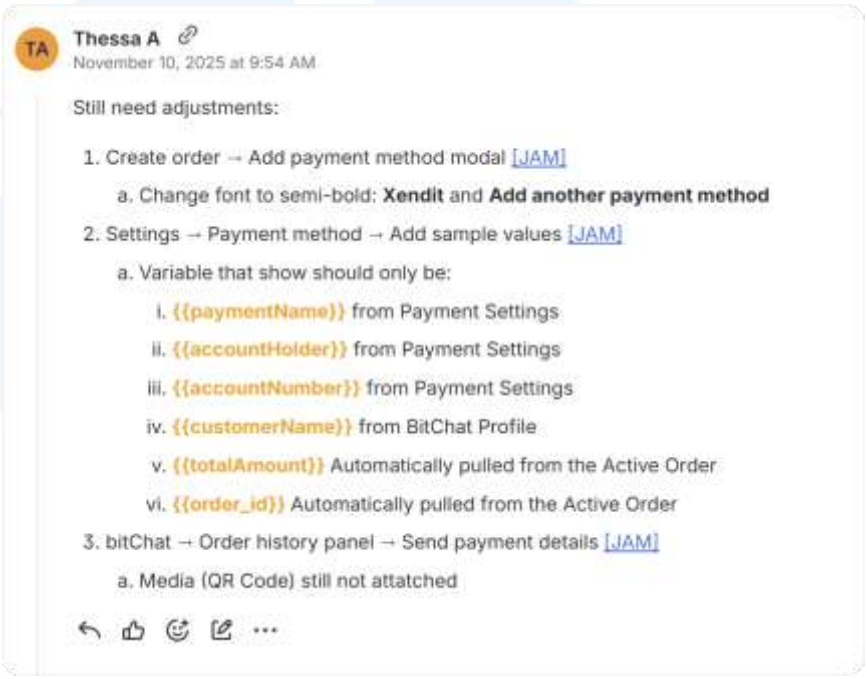
Setelah proses finalisasi *handover file* dan PRD, penulis melaksanakan *handover sync* kepada tim *Software Developer*. Begitu implementasi awal pada *task* Jira selesai, tugas tersebut kembali di-*assign* kepada penulis untuk memasuki tahap *testing*. Pada *testing* pertama, penulis menemukan tiga isu pada alur *Create Order*, terutama pada *modal Add Payment Method*. Beberapa komponen masih memerlukan penyesuaian *font weight*, validasi tombol *Save* belum berjalan dengan benar, serta *error handling* pada *modal Add Sample Values* belum berfungsi sesuai kebutuhan. Elemen media juga belum muncul secara konsisten, sementara di *bitChat*, variabel dinamis pada *Send Payment Details* tidak terisi dan *QR code* belum ikut terkirim.



Gambar 3. 117 *Custom payment method test 1*

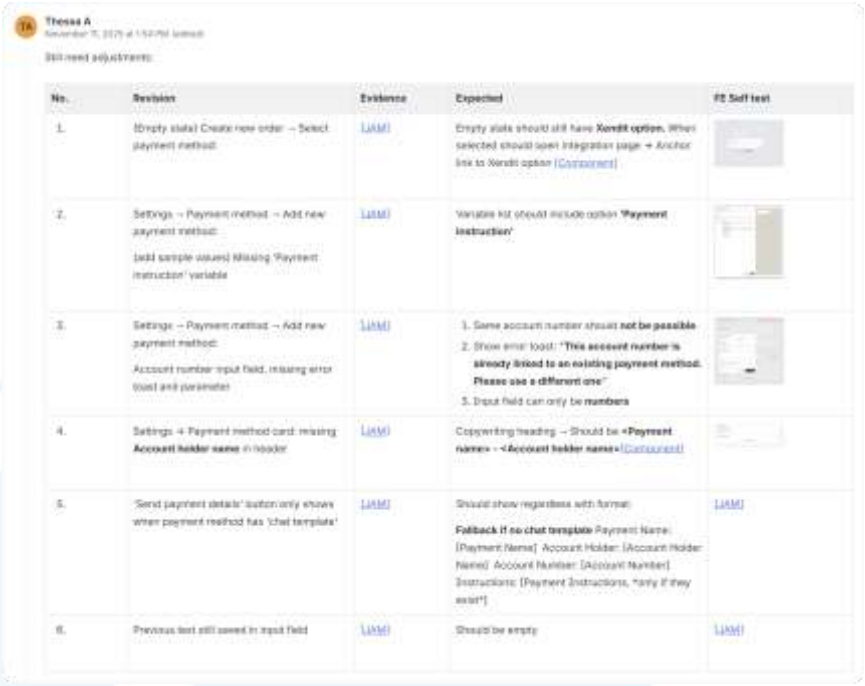
Berdasarkan perbaikan yang dilakukan, *testing* kedua masih menunjukkan ketidaksesuaian. *Modal Add Payment Method* masih

membutuhkan pembaruan *font weight*, dan daftar variabel pada *Add Sample Values* belum sesuai dengan daftar final yang menyertakan data dari *Payment Settings*, profil pengguna bitChat, serta pesanan aktif. Masalah lampiran media pada *Send Payment Details* juga masih terjadi.



Gambar 3. 118 *Custom payment method test 2*

Testing ketiga muncul beberapa isu tambahan yang memerlukan dukungan tim *Backend*, seperti perilaku tombol "X" pada pratinjau lampiran di bitChat yang menghapus seluruh isi pesan, bukan hanya lampirannya. Selain itu, *empty state* pemilihan metode pembayaran masih perlu menampilkan Xendit beserta tautan integrasinya. Variabel *Payment Instruction* juga belum muncul pada *Add Sample Values*, dan formulir *Add New Payment Method* masih kekurangan validasi, termasuk deteksi nomor rekening duplikat. Beberapa komponen lain, seperti informasi *account holder* pada kartu metode pembayaran dan *fallback* tombol "*Send Payment Details*," juga masih perlu disesuaikan. Penulis menemukan *input field* di bitChat belum otomatis kosong setelah *modal* ditutup.



Gambar 3.119 *Custom payment method test 3*

Setelah revisi diselesaikan secara bertahap oleh tim *Software Developer front-end* maupun *back-end*, fitur dinyatakan stabil. Tahapan akhir mencakup *testing* di lingkungan *development*, dilanjutkan dengan *canary deployment*, hingga akhirnya fitur dinyatakan siap dan resmi dirilis ke *production*.

3.4 Kendala dan Solusi Pelaksanaan Kerja

Dalam menjalankan magang di bitbybit, penulis menemukan beberapa kendala yang cukup menantang. Kendala tersebut meliputi aspek koordinasi antar divisi, keterbatasan pemahaman teknis di awal magang, serta penyesuaian dengan alur kerja yang berbeda dari lingkungan akademik. Pengalaman menghadapi kendala ini justru menjadi bagian penting dari proses pembelajaran dan membantu memahami realitas kerja di industri pengembangan produk digital.

3.4.1 Kendala Pelaksanaan Kerja

Selama menjalani magang di perusahaan, penulis menghadapi beberapa kendala yang berkaitan dengan alur kolaborasi lintas divisi, pemahaman proses internal, serta beban *testing* fitur yang mengalami

peningkatan signifikan. Kendala-kendala tersebut muncul pada sejumlah proyek dan berdampak pada akurasi implementasi, kelancaran *handover*, serta efisiensi proses validasi fitur.

A. Koordinasi proses *development*

Dalam beberapa proyek, ditemukan bahwa alur kerja antara tim *Product*, tim *Software Developer*, dan *Quality Assurance* belum sepenuhnya selaras. Kendala koordinasi lintas tim dialami terutama saat mengerjakan proyek *Increase AI Agent Activation Rate* dan beberapa fitur AI lain seperti *AI Studio Polishing*. Setelah *handover* lengkap beserta PRD dan *prototype*, implementasi awal yang dijalankan tim *Software Developer* tidak sesuai spesifikasi.

Salah satu contohnya terjadi ketika fungsi auto-generated questions pada *AI Playground* tidak aktif, sehingga sistem menampilkan *fallback questions*, meskipun komponen tersebut seharusnya hanya muncul jika proses *scraping* atau AI mengalami kegagalan Selain itu, fitur tersebut sempat di-*deploy* tanpa pemberitahuan karena QA *intern* menilai fitur siap padahal QA belum memverifikasi semua requirement dari PRD. Kejadian ini memperpanjang siklus perbaikan karena baru diketahui setelah *deployment*, bukan melalui proses *re-assign* yang ideal dari tim *Software Developer* untuk *testing* akhir.

B. Keterbatasan Penjelasan Teknis dan Alur Internal

Kendala berikutnya muncul dari kurangnya pemahaman awal terhadap alur teknis internal. Pada tahap awal magang, dihadapi tantangan untuk menyesuaikan diri dengan standar prosedur internal yang belum dijelaskan secara menyeluruh. Misalnya, proses *testing* fitur ternyata terdiri dari dua *testing* di lingkungan *development* dan *canary deployment*. Pengetahuan mengenai hal ini baru diperoleh setelah *Supervisor* menanyakan hasil *testing* di *canary deployment*, yang sebelumnya belum pernah dijelaskan. Situasi serupa terjadi pada proses dokumentasi

deployment, di mana diketahui bahwa catatan *deployment* dicantumkan melalui thread Google Chat setelah beberapa proyek *Increase AI Agent Activation* terlanjur berjalan tanpa pemantauan ulang.

Ditemukan juga kendala terkait ekspektasi *timeline* pengerjaan. Pada proyek bitChat *React Native Improvement*, terjadi keterlambatan penyelesaian tugas yang disebabkan oleh kondisi kesehatan. Meskipun pada *sprint-sprint* sebelumnya tugas selalu diselesaikan lebih awal dari *deadline*, keterlambatan kali ini tetap menjadi catatan dalam evaluasi bulanan. *Supervisor* memberikan masukan bahwa apabila beban kerja dirasa terlalu berat, tugas dapat diminta untuk dibagi menjadi dua *sprint*. Namun, informasi mengenai fleksibilitas pembagian tugas ini tidak disampaikan secara eksplisit di awal masa magang, sehingga penulis belum mengetahui bahwa opsi tersebut tersedia.

Selain itu, proses *review* desain terkadang tidak dilakukan secara konsisten tanpa mengirimkan tautan *file* secara proaktif, sehingga progres menunggu lebih lama. Situasi ini menunjukkan bahwa alur komunikasi terkait *review* dan pendampingan masih dapat diperjelas, khususnya bagi peserta magang yang baru mengenal alur kerja perusahaan.

C. Testing end-to-end

Kendala ini muncul karena beban *testing* yang tinggi merupakan dampak dari minimnya personel yang menjalankan fungsi *Quality Assurance*. Namun penyusunan metode dan dokumentasi *testing* justru diarahkan sebagai *intern*. Setelah masa magang *Quality Assurance intern* berakhir, tidak ada lagi personel yang secara khusus menangani proses *testing*, sehingga seluruh *testing*, termasuk fungsionalitas yang berada di luar ranah desain. *Testing* yang dilakukan tidak hanya sebatas validasi desain, tetapi mencakup pemeriksaan fungsionalitas secara menyeluruh, sehingga beban kerja meningkat.

Situasi ini terlihat pada project *Custom Payment Improvements*, di mana diminta menyiapkan tabel *bug/fixes*, *expected result, evidence,* serta meminta *developer* melakukan *self-testing* dengan bukti pendukung setelah perbaikan dilakukan. Sementara itu, implementasi dari sisi *development* masih sering mengalami kekurangan, yang menyebabkan proses *testing* menjadi lebih panjang dan menambah tekanan pada tim *Product*. Berdasarkan pengalaman penulis, tantangan utama bukan hanya pada bagaimana membuat *testing* lebih efisien, tetapi pada peningkatan ketelitian saat implementasi agar kesalahan tidak berulang.

Di tengah kondisi tersebut, sempat muncul usulan agar tim *Product* memanfaatkan AI untuk menghasilkan seluruh *test case* untuk *task* terkait. Namun, penyusunan *test case* dan verifikasi fungsionalitas merupakan tanggung jawab *Quality Assurance*, sehingga ketika peran tersebut bergeser kepada desainer, fokus pekerjaan menjadi melebar dari *jobdesk* awal.

3.4.2 Solusi Pelaksanaan Kerja

A. Solusi untuk Kendala Koordinasi Lintas Divisi

Pertama, solusi yang dapat diterapkan adalah menetapkan sesi *handover* yang lebih terstruktur dan memastikan seluruh anggota tim yang terlibat membaca PRD secara menyeluruh sebelum memulai implementasi. Selain itu, proses *re-assign* setelah pengembangan selesai perlu dijalankan secara konsisten, sehingga *testing* tidak terlewat dan potensi kesalahan dapat diminimalisir sebelum fitur masuk ke tahap *deployment*.

B. Solusi untuk Keterbatasan Pemahaman Teknis di Awal Magang

Terkait keterbatasan pemahaman teknis di tahap awal magang, solusi yang dapat diterapkan adalah menyediakan *onboarding* yang lebih komprehensif untuk peserta magang. *Onboarding* tersebut dapat mencakup penjelasan mengenai lingkungan *testing*, mekanisme pencatatan *deployment*, alur *sprint*, serta alur komunikasi internal yang biasanya dijalankan oleh tim. Dengan adanya penjelasan yang jelas sejak awal,

peserta magang dapat menyesuaikan diri lebih cepat dan mengurangi risiko kesalahpahaman terkait proses kerja.

Solusi bagi ketidaksinkronan ekspektasi *timeline* adalah memperjelas fleksibilitas pembagian tugas dan tingkat kompleksitas pekerjaan sejak awal *sprint*. *Supervisor* dapat memberikan arahan bahwa *intern* diperbolehkan meminta penyesuaian beban kerja apabila suatu task dirasa terlalu besar atau membutuhkan pengerjaan bertahap. Dengan komunikasi yang lebih terbuka mengenai kapasitas kerja dan *timeline*, *intern* dapat mengelola waktu dengan lebih efektif dan menghindari kesenjangan persepsi mengenai performa kerja.

C. Solusi untuk Beban *Testing* yang Meningkat

Untuk kendala terkait proses *testing end-to-end*, solusi utama adalah mengembalikan tanggung jawab *testing* fungsionalitas kepada peran yang sesuai, yaitu *Quality Assurance*. Perusahaan dapat mempertimbangkan kembali penempatan personel *Quality Assurance* atau mencari alternatif jangka panjang agar proses verifikasi tidak dibebankan kepada tim *Product*. Dengan demikian, tim *Software Developer* dapat fokus meningkatkan ketelitian implementasi, sementara *Quality Assurance* memastikan validasi fitur dilakukan secara komprehensif. Selain itu, dokumentasi *testing* seperti tabel *bugs* dan *expected result* sebaiknya tetap dikelola *Quality Assurance* agar alur kerja tidak melebar dari *jobdesk* desainer. Upaya memanfaatkan AI untuk menghasilkan *test case* dapat menjadi pendukung tambahan, bukan menggantikan fungsi *Quality Assurance*.