PENGEMBANGAN APLIKASI MOBILE UNTUK DISTRIBUTOR LACOCO MENGGUNAKAN METODE PROTOTYPING

Disusun Oleh:

N a m a : Rahmad Arifan Hr

NIM : 13523077

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN APLIKASI MOBILE UNTUK DISTRIBUTOR LACOCO MENGGUNAKAN METODE PROTOTYPING

TUGAS AKHIR

Disusun Oleh:

N a m a : Rahmad Arifan Hr

NIM : 13523077

Yogyakarta, 07 Oktober 2020

Pembimbing 1	Pembimbing 2
(Hendrik S.T., M.Eng.)	(Hari Setiaji, S.Kom., M.Eng.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN APLIKASI MOBILE UNTUK DISTRIBUTOR LACOCO MENGGUNAKAN METODE PROTOTYPING

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 07 Oktober 2020

Tim Penguji

Hendrik S.T., M.Eng. _____

Anggota 1

Erika Ramadhani, S.T., M.eng. _____

Anggota 2

Irving Vitra Paputungan, S.T., M.Sc. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Rahmad Arifan Hr

NIM : 13523077

Tugas akhir dengan judul:

PENGEMBANGAN APLIKASI MOBILE UNTUK DISTRIBUTOR LACOCO MENGGUNAKAN METODE PROTOTYPING

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 07 Oktober 2020

(Rahmad Arifan Hr)

HALAMAN PERSEMBAHAN

Bismillahirahmanirrahim,

Puji syukur atas izin Allah Subhanahu wa Ta'ala akhirnya saya dapat menyelesaikan tugas akhir ini dengan baik. Sholawat dan salam tak lupa pula saya panjatkan kepada Nabi Muhammad Shallallahu alaihi wasallam beserta keluarganya, dan orang-orang yang selalu mengikuti tuntunan darinya.

Terimakasih yang tak terhingga saya ucapkan kepada Ayah dan Ibu yang selalu mendoakan saya dan selalu memberi dukungan baik dalam motivasi, do'a, moril, materil, dan segala bentuk kebaikan lainnya. Dengan laporan tugas akhir inilah salah bentuk wujud terimakasih saya dan berharap semoga mampu membuat bahagia dan bangga kepada anaknya. Tak lupa pula kepada keluarga saya, kakak dan adik yang selalu membantu dalam do'a dan motivasi selama pengerjaan tugas akhir ini. Serta kepada teman-teman dan sahabat-sahabat saya yang tidak bisa saya sebutkan satu persatu.

Terakhir saya ucapkan terimakasih sebanyak-banyaknya kepada semua pihak yang telah membantu, mendo'akan, dan mendukung saya selama pengerjaan tugas akhir ini. Semoga Allah Subhanahu wa Ta'ala membalas kebaikan semua pihak terkait dan selalu memberikan kesehatan kepada kita semua, Aamiin.

HALAMAN MOTO

“ Janganlah kamu berduka cita, sesungguhnya Allah bersama kita. ”

(Q.S. At – Taubah: 40)

“Intashduqillaha yashduqka”

“ Jika engkau jujur kepada Allah, niscaya Allah akan jujur kepadamu.”

(HR. An-Nasai)

KATA PENGANTAR

Assalaamu'alaikum warahmatullaahi wabarakatuh

Alhamdulillah puji dan syukur selalu penulis panjatkan kepada Allah Subhanahu wa Ta'ala atas segala nikmat iman dan nikmat islam, serta karunia-Nyalah akhinya dapat menyelesaikan laporan tugas akhir dengan baik. Sholawat serta salam selalu penulis haturkan kepada Nabi Besar kita, Nabi Muhammad Shallallahu alaihi wasallam karena berkat bimbinganyalah kita sebagai kaum muslimin bisa menuju ke jalan yang benar, dan semoga kita semua akan dikumpulkan bersamanya di Surga kelak, Aamin.

Tugas akhir ini merupakaan salah satu syarat yang harus diselesaikan untuk memperoleh gelar sarjana di Jurusan Informatika Universitas Islam Indonesia. Selama proses pengerjaan tugas akhir ini yang berjudul **"Pengembangan Aplikasi Mobile Untuk Distributor Lacoco menggunakan Metode Prototyping"** penulis memperoleh banyak ilmu baru, pengetahuan, dan tentunya pengalaman baru ketika menghadapai kesulitan selama pengerjaannya, namun dengan bantuan berbagai pihak akhirnya dapat terselesaikan dengan baik. Oleh karena itu pada kesempatan kali ini, izinkan penulis untuk mengucapkan terimakasih kepada:

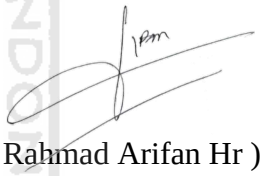
1. Kedua orang tua penulis beserta kakak adik yang selalu mendoakan, memotivasi, dan tidak mungkin penulis sebutkan satu per satu jasa mereka.
2. Bapak Fathul Wahid, S.T., M.Sc., Ph.D selaku Rektor Universitas Islam Indonesia.
3. Bapak Hari Purnomo, Prof., Dr., Ir., M.T. selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Dr. Raden Teguh, S.T., M.Sc., selaku Ketua Program Studi Informatika – Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.
5. Bapak Hendrik S.T., M.Eng. dan Bapak Hari Setiaji, S.Kom., M.Eng. selaku dosen pembimbing yang selalu memberikan masukan, ide, saran, dan bantuannya selama pengerjaan tugas akhir ini.
6. Seluruh staf pengajar di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
7. Staf administrasi FTI, secara khusus kepada staf administrasi Program Studi Informatika – Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia yang selalu memberikan informasi terkait tugas akhir.
8. Hanif Annur Rahman dan Mas Bayu yang telah membantu memberikan solusi ketika penulis mengalami kendala selama pengerjaan tugas akhir.

9. Aris Nurul Huda, Arif Budiman, dan teman – teman PT. Avo Innvovation Technology yang juga membantu memberikan ide dan solusi.
10. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang juga turut membantu penulis selama pengerjaan tugas akhir.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna, masih banyak kekurangan yang mungkin bisa ditingkatkan lagi. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun demi kesempurnaan tugas akhir ini. Akhir kata, penulis berharap tugas akhir ini dapat bermanfaat bagi semua pihak.

Wassalaamu'alaikum warahmatullaahi wabarakatuh

Yogyakarta, 07 Oktober 2020



(Rahmad Arifan Hr)

SARI

PT AVO Innovation Technology (AVO) adalah salah satu perusahaan yang bergerak di bidang industri kosmetik. Perusahaan ini telah mengeluarkan beberapa produk kecantikan, di antaranya Avoskin, Looke, dan Lacoco. Lacoco merupakan produk kosmetik perawatan wajah yang telah menggunakan *electronic commerce* (*e-commerce*) berbasiskan web sebagai solusi untuk mengenalkan produk – produknya kepada konsumen. Namun sistem *e-commerce* Lacoco yang telah dikembangkan belum sepenuhnya sempurna dari sisi distributor karena mungkin saja pihak distributor akan terlambat memproses pesanan – pesanan yang masuk dari *customer*. Hal ini terjadi dikarenakan pihak distributor tidak selama 24 jam bisa aktif menggunakan sistem *e-commerce* untuk melihat pesanan yang masuk. Oleh karena itu, untuk meningkatkan kualitas pelayanan pada Lacoco maka perlu dibuat sebuah aplikasi bergerak (*mobile*) untuk sisi distributor. Aplikasi ini diharapkan dapat membangun hubungan yang lebih produktif antara distributor dengan customer dan distributor dengan pihak AVO sebagai pemilik Lacoco. Fitur-fitur yang akan dikembangkan sesuai dengan kebutuhan dari pihak AVO yaitu notifikasi, manajemen pesanan, analitik terkait riwayat penjualan, dan pembelian produk dari distributor ke AVO.

Metode pengembangan yang digunakan selama pengembangan aplikasi *mobile* untuk distributor Lacoco adalah metode *prototyping.* Metode ini terdapat empat tahapan mendasar yaitu analisis kebutuhan, desain, membangun sistem, dan pengujian. Fokus dari metode ini adalah pada tahap desain yang mana terdapat beberapa iterasi didalam membangun desain, setiap iterasi tersebut terdapat beberapa tahapan yaitu membuat desain sederhana, membuat *prototype,* evaluasi desain ke *customer,* dan merevisi desain jika desain belum sesuai dengan keinginan dan kebutuhan *customer.* Pada penelitian tugas akhir ini terdapat dua kali iterasi.

Untuk mendapatkan hasil yang optimal, maka penelitian ini menggunakan dua jenis pengujian yaitu pengujian *black box* dan pengujian langsung kepada pemilik aplikasi. Dari hasil pengujian yang dilakukan dapat disimpulkan bahwa aplikasi berjalan sesuai harapan penulis dan aplikasi dapat diterima dengan baik oleh *customer.*

Kata kunci: PT. AVO Innovation Technology, distributor, *prototyping*, *black box*.

GLOSARIUM

Reseller	Orang yang menjual kembali produk PT. Avo ke Customer
Distributor	Pada penelitian ini distributor sama halnya dengan *reseller*
Customer	Orang yang membeli produk Lacoco
E-commerce	Aktivitas jual beli melalui media elektronik
Iterasi	Perulangan
Activity Diagram	Alur kerja dari sebuah fitur atau sistem
User Interface	Antarmuka pengguna
Mockup	Visualisasi atau *preview* dari konsep desain

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kosmetik adalah salah satu unsur dari kecantikan dan kecantikan pada saat ini tak hanya menjadi sebuah keinginan melainkan sudah menjadi sebuah kebutuhan. Hal ini berdampak kepada semakin meningkatnya dunia industri kosmetik. Terbukti industri kosmetik di Indonesia pada tahun 2018 mengalami pertumbuhan mencapai 20% atau empat kali lipat lebih tinggi dari pertumbuhan ekonomi nasional pada tahun 2017. Berdasarkan siaran pers Menteri Perindustrian, Airlangga Hartarto mengatakan industri kosmetik dalam negeri yang pada tahun 2017 hanya memiliki 153 perusahaan, hingga pada tahun 2018 jumlahnya mencapai lebih dari 760 perusahaan (Kemenperin, 2018).

PT AVO Innovation Technology (AVO) merupakan salah satu perusahaan yang bergerak di bidang industri kosmetik. PT. AVO berdiri pada 10 Oktober 2014, di Yogyakarta oleh Anugrah Pakerti. Perusahaan ini telah mengeluarkan beberapa produk kecantikan, di antaranya Avoskin, Looke, dan Lacoco. Produk-produk kosmetik yang telah diproduksi berupa *facial wash and soap, hydrating essence, facial cream, lotion,* dan lain-lain.

Lacoco adalah salah satu produk baru dari AVO, yang merupakan kosmetik perawatan wajah berupa *essence, face mask, eye serum*, dan lain-lain. Lacoco telah menggunakan *electronic commerce* (*e-commerce*) berbasiskan web sebagai solusi untuk mengenalkan produk-produknya kepada konsumen. Data penjualan produk AVO pada tahun 2017 tercatat sejumlah 21.437 unit dari *e-commerce*. Saat ini Lacoco telah memiliki ribuan calon mitra bisnis atau distributor yang menjadi tulang punggung penjualan produk Lacoco yang mana distributor tersebut didapatkan dari hasil kerjasama AVO dengan PT. Natural Nusantara (Huda, 2018). Distributor disini sama halnya dengan *reseller* pada proses bisnis lain yaitu orang yang menjual kembali produk AVO kepada *customer*.

Namun sistem *e-commerce* Lacoco yang telah dikembangkan belum sepenuhnya sempurna dari sisi distributor karena mungkin saja pihak distributor akan terlambat memproses pesanan – pesanan yang masuk dari *customer*. Hal ini terjadi dikarenakan pihak distributor tidak selalu aktif menggunakan sistem *e-commerce* untuk melihat pesanan yang masuk.

Oleh karena itu, untuk meningkatkan kualitas pelayanan pada Lacoco maka perlu dibuat sebuah aplikasi bergerak (*mobile)* untuk sisi distributor. Aplikasi ini diharapkan dapat

membangun hubungan yang lebih produktif antara distributor dengan *customer* dan distributor dengan pihak AVO sebagai pemilik Lacoco. Fitur-fitur yang ada di dalam aplikasi tersebut sesuai dengan kebutuhan dari pihak AVO yaitu notifikasi, manajemen pesanan, analitik terkait riwayat penjualan, dan pembelian produk dari distributor ke AVO. Diharapkan aplikasi ini dapat memberikan informasi yang lebih cepat dan *private* kepada distributor baik itu informasi dari *customer* ataupun pihak AVO.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka permasalahan yang dapat diangkat adalah bagaimana mengembangkan aplikasi bergerak (*mobile*) berbasiskan Android untuk distributor dengan menggunakan metode pengembangan *Evolutionary Prototyping*?

1.3 Batasan Masalah

Adapun batasan-batasan dalam membangun sistem ini adalah :

a. Aplikasi *mobile* yang dikembangkan untuk sistem operasi Android 6.0 ke atas.
b. Aplikasi yang dikembangkan merupakan aplikasi komplementer atau pelengkap untuk sisi Distributor dari sistem *e-commerce* yang ada pada Lacoco.
c. Sumber data yang digunakan pada aplikasi ini didapatkan dari platform web sistem *e-commerce* Lacoco yaitu lacoco.co.id.
d. Fitur yang dibangun sesuai dengan kebutuhan dari pihak PT. AVO.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah mengembangkan sebuah aplikasi bergerak (*mobile)* untuk distributor Lacoco yang dapat membantu PT. AVO Innovation Technology dalam melakukan pemasaran.

1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dalam penelitian ini adalah sebagai berikut:

a. Memberikan informasi atau notifikasi yang cepat kepada distributor Lacoco jika ada pesanan yang masuk dari customer.
b. Memberikan informasi kepada distributor terkait riwayat penjualan dalam bentuk analitik.

c. Memberikan kemudahan kepada distributor jika ingin melakukan pembelian produk kepada pihak supplier AVO.

1.6 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini yaitu Metode *Evolutionary Prototyping*. Metode ini terdiri dari beberapa tahap, yaitu:

a. Analisis, merupakan tahapan mengidentifikasi permasalahan, mengkaji data yang diperlukan, dan mengumpulkan kebutuhan-kebutuhan yang diperlukan untuk mencapai tujuan yang akan dicapai nantinya.
b. Desain, merupakan tahapan perancangan dan desain sistem. Pada Metode *Evolutionary Prototyping* tahap ini terdapat iterasi yaitu membuat desain sederhana, membuat *Prototyping*, evaluasi dari *customer*, revisi desain. Iterasi ini terus berjalan sampai mencapai kesepakatan desain yang sesuai dengan keinginan *customer*.
c. Membangun sistem, yaitu tahapan menterjemahkan data atau memecahkan masalah yang telah dirancang sebelumnya menjadi sebuah sistem yang diharapkan dengan menggunakan bahasa pemrograman yang telah ditentukan.
d. Pengujian, merupakan tahapan untuk melakukan uji coba terhadap sistem yang telah dibangun, yang bertujuan untuk memastikan bahwa aplikasi yang telah dibangun dapat beroperasi dengan baik dan sesuai dengan kebutuhan *customer*.

1.7 Sistematika Penulisan

Laporan ini terdiri dari lima bab dengan penjelasan masing – masing bab sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini merupakan pengantar permasalahan yang akan diteliti. Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Dalam bab ini membahas tentang teori-teori dasar yang berkaitan dengan permasalahan yang akan diselesaikan. Teori-teori yang berkaitan dengan penelitian ini, antara lain: Lacoco, *Prototyping, Application Programming Interface* (API), Firebase, Midtrans dan Laravel.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi tentang analisis kebutuhan dan perancangan sistem, yang mana dua hal tersebut adalah dua tahap awal pada metodologi *prototyping* yang digunakan pada penelitian kali ini.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan tentang lanjutan dari dua tahap yang telah dilakukan pada bab sebelumnya yaitu mengimplementasikan rancangan menjadi sebuah sistem yang diharapkan. Selain itu juga memuat pengujian.

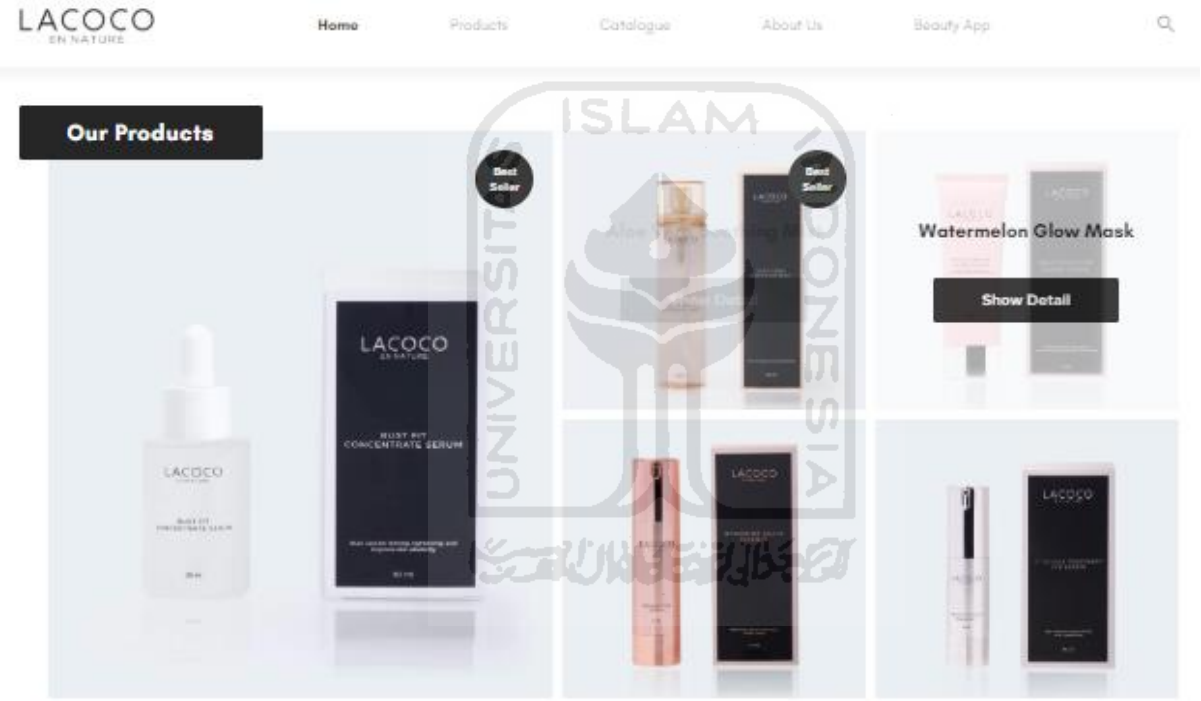
BAB V PENUTUP

Bagian ini berisi tentang refleksi *prototyping*, dan kesimpulan dari keseluruhan proses kerja penelitian ini. Selain itu, bab ini juga berisi saran untuk penelitian – penelitian berikutnya.

BAB II
LANDASAN TEORI

2.1 Lacoco

Lacoco merupakan salah satu produk dari PT. Avo Innovation Technology yang diluncurkan pada tahun 2018. Lacoco adalah produk kosmetik perawatan wajah berupa *essence*, *face mask*, *eye serum*, dan lain-lain. Lacoco selama ini menggunakan *electronic commerce* (*e-commerce*) berbasiskan web sebagai solusi untuk mengenalkan produk – produknya kepada konsumen dengan alamat url lacoco.co.id. Lihat Gambar 2.1 untuk halaman awal pada lacoco.co.id.

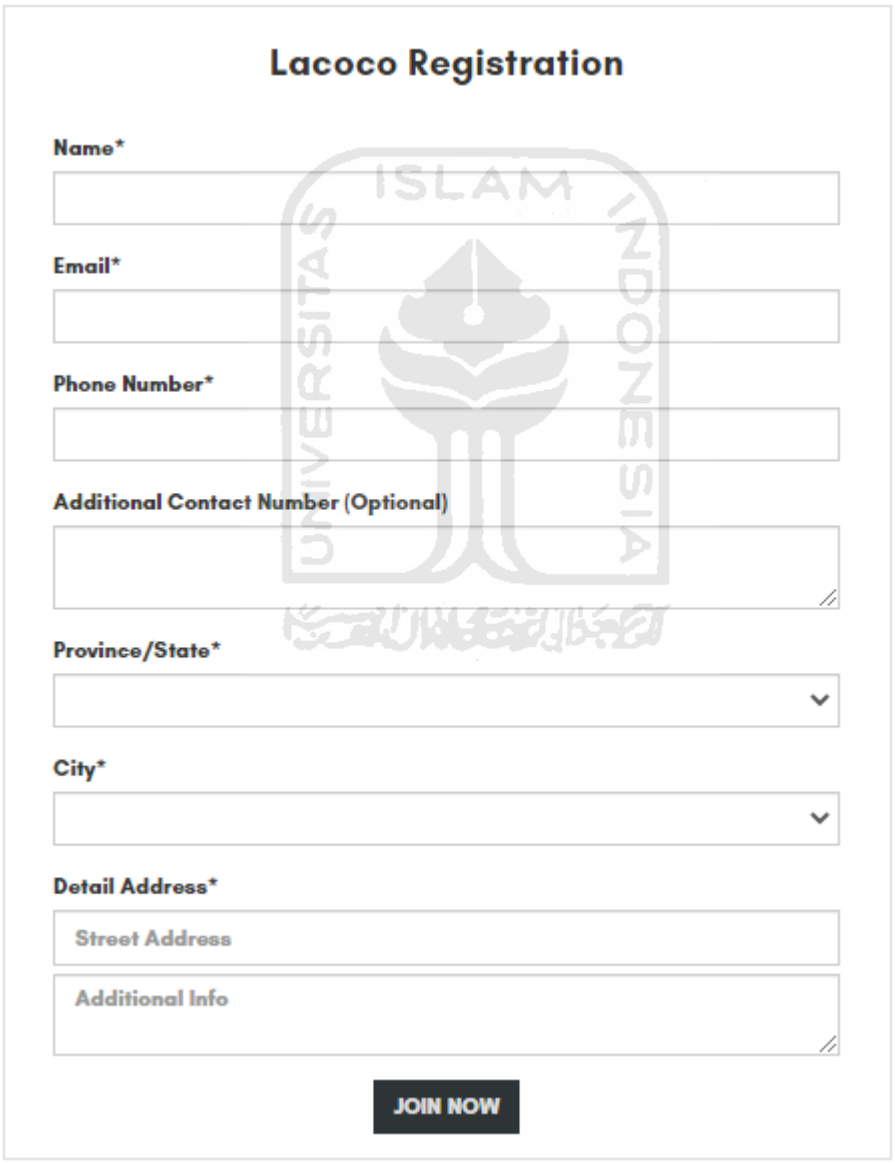


Gambar 2.1 Halaman Awal Lacoco.co.id

Halaman Lacoco.co.id pada saat ini hanya berfungsi untuk memperkenalkan produk – produknya saja, sedangakan untuk penjualannya melalui instagram atau *marketplace* yang tersedia di Indonesia. Data penjualan produk AVO pada tahun 2017 tercatat sejumlah 21.437 unit dari *e-commerce*. Lacoco telah memiliki ribuan calon mitra bisnis atau distributor yang menjadi tulang punggung penjualan produk Lacoco yang mana distributor tersebut didapatkan dari hasil kerjasama AVO dengan PT. Natural Nusantara (Huda, 2018).

Dengan besarnya angka distributor yang dimiliki oleh PT. AVO, Aris Nurul Huda melakukan penelitian pada tahun 2018 dengan mengembangkan sistem informasi dan penjualan Lacoco berbasis website. Model bisnis yang diterapkan pada produk Lacoco berbeda dengan produk PT. AVO lainnya, yang mana Lacoco mengandalkan distributor untuk menjualkan produknya. Jadi ketika seorang *customer* atau pembeli ingin membeli produk Lacoco, maka ia akan diarahkan ke distributor yang tersedia.

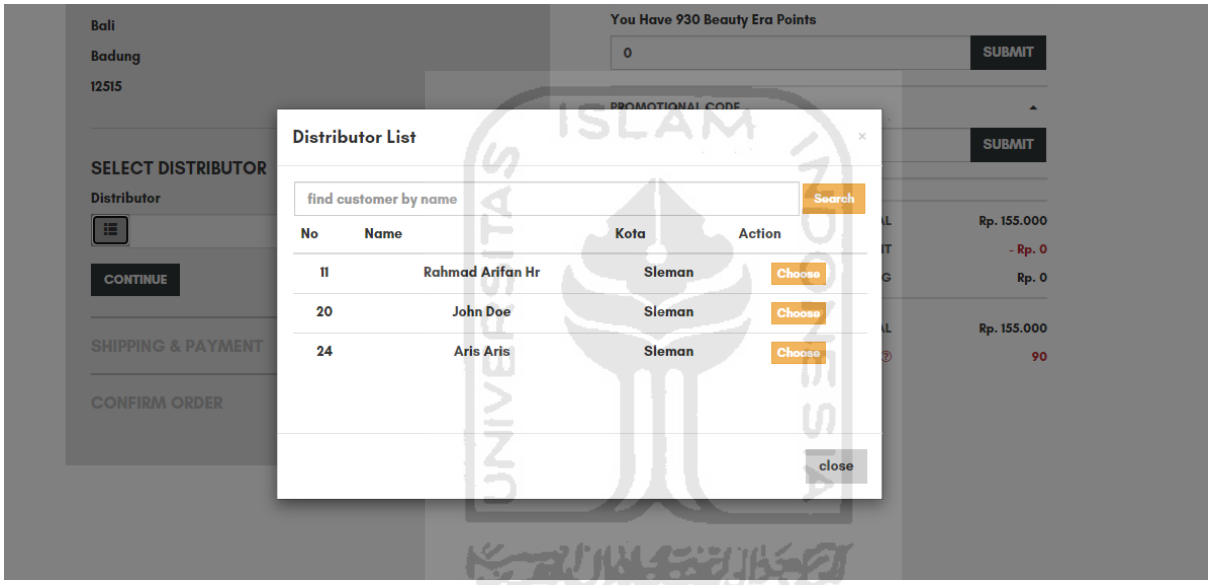
Berdasarkan penelitian yang dilakukan oleh Huda (2018), untuk menjadi seorang distributor maka calon distributor terlebih dahulu mendaftar melalui website yang telah dikembangkannya. Gambar 2.2 menunjukkan *form* pendaftaran untuk menjadi seorang distributor di Lacoco.



Gambar 2.2 Halaman Pendaftaran Distributor

Setelah mendaftar, calon distributor tidak langsung menjadi distributor melainkan harus melalui proses review oleh admin PT. AVO. Jika calon distributor disetujui, maka sistem akan membuatkan secara otomatis akun untuk *login* dan menginformasikannya ke *email* distributor pada saat mendaftar. Setelah menjadi seorang distributor, maka ia bisa membeli produk untuk menyetok barang agar bisa dijual ke *customer*.

Selanjutnya seorang *customer* jika ingin membeli produk Lacoco, bisa dengan mendaftar menjadi seorang *user customer* terlebih dahulu atau langsung *checkout* produk yang akan dibelinya. Pada saat *checkout*, seorang *customer* harus memilih distributor yang tersedia. Gambar 2.3 memperlihatkan halaman pemilihan distributor saat *customer* melakukan *checkout.*



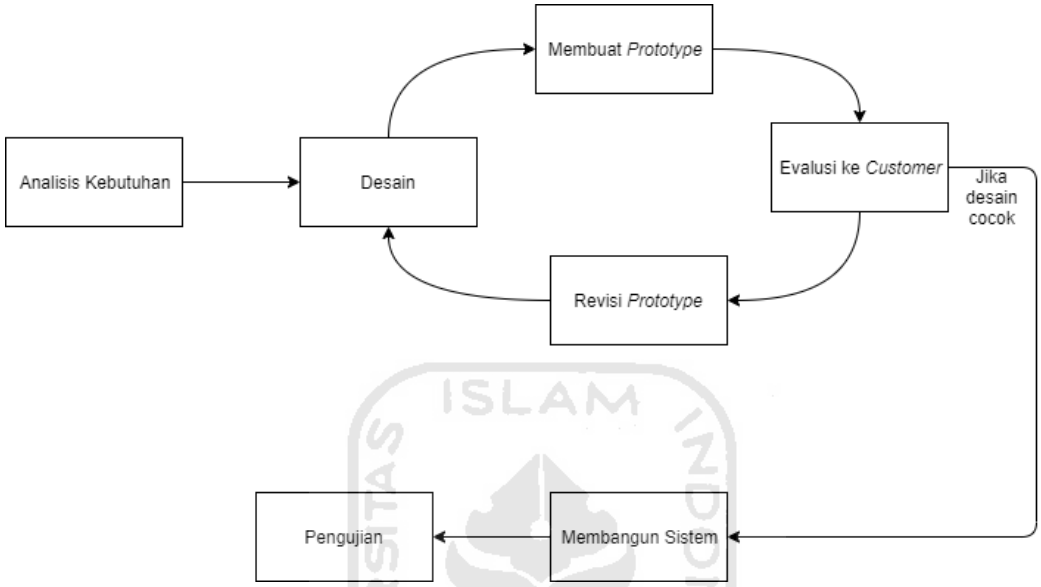
Gambar 2.3 Halaman Pemilihan Distributor Saat *Checkout*

2.2 *Prototyping*

Prototyping merupakan salah satu metodologi pengembangan perangkat lunak, yang mana fokus dari metodologi ini adalah pada proses desain yaitu dengan membuat desain sederhana, membuat *prototype*, melakukan evaluasi kepada *costumer* (pemilik aplikasi), dan merevisi *prototype*. Tahap desain ini berupa iterasi yang terus berjalan sampai mencapai kesepakatan desain yang sesuai dengan keinginan *customer.*

2.2.1 Tahapan Metodologi *Prototyping*

Pada metodologi pengembangan sistem, secara umum tahapan untuk menyelesaikan masalah adalah analisis, desain, membangun sistem, lalu pengujian. Berikut gambaran tahapan pada metodologi *prototyping,* yang diperlihatkan pada Gambar 2.4.



Gambar 2.4 Metodologi *Prototyping*

Pada Gambar 2.4 menunjukan tahapan dari metodologi *prototyping* yaitu :

1. Analisis Kebutuhan

Tahapan ini merupakan tahapan mengidentifikasi permasalahan, mengkaji data yang diperlukan, dan mengumpulkan kebutuhan-kebutuhan yang diperlukan selama pembuatan perangkat lunak.

2. Desain

Tahapan ini merupakan tahapan mendesain sistem yang di dalamnya terjadi proses perulangan atau iterasi. Iterasi tersebut memuat beberapa tahapan yaitu membuat desain sederhana, lalu membuat *prototype*, melakukan pengujian atau evaluasi ke *customer* (pihak pemilik aplikasi), merevisi *prototype* jika desain belum sesuai dengan keinginan *customer*. Iterasi ini bisa berulang satu kali, dua kali, atau bahkan lebih, tergantung dari hasil evaluasi desain kepada pemilik aplikasi. Jika semua fitur sudah selesai dirancang dan desain sudah sesuai dengan keinginan pemilik aplikasi maka iterasi itupun berhenti. Jika iterasi berhenti, maka tahap desain pada metodologi *prototyping* sudah selesai, dan siap untuk melanjutkan tahap selanjutnya.

3. Membangun Sistem

Pada tahapan ini *developer* mulai membangun sistem atau perangkat lunak sesuai dengan desain yang telah disepakai pada tahap sebelumnya, dan dengan bahasa pemrograman tertentu yang telah ditentukan pada tahap analisis kebutuhan.

4. Pengujian

Tahapan ini merupakan tahapan untuk melakukan uji coba terhadap perangkat lunak yang telah dibangun, yang bertujuan untuk memastikan bahwa aplikasi dapat berjalan dengan baik dan sesuai dengan kebutuhan *customer*.

2.2.2 Tipe *Prototyping*

Ada beberapa tipe metodologi *prototyping* antara lain *Rapid Throwaway Prototype, Evolutionary Prototype, Incremental Prototype,* dan *Extreme Prototyping.* (Guru99, 2020)

a. *Rapid Throwaway Prototype*

Rapid Throwaway Prototype merupakan salah satu tipe dari metodologi *prototyping*. Tipe ini pada tahap desain selalu melakukan tahap analisis kembali secara lebih detail kepada *customer*, berdasarkan analisis ini kemudian dibuat desain *prototype,* lalu *prototype* tersebut dievaluasi oleh *customer*. Singkatnya tipe ini memvisualisasikan kebutuhan yang disampaikan oleh *customer,* kemudian dari visualisasi itu dibuat desain akhir yang akan digunakan oleh developer pada tahap implementasi dan hasil visualisasi itu akan dibuang atau tidak digunakan untuk tahap selanjutnya.

b. *Evolutionary Prototype*

Tipe ini merupakan tipe yang selalu melakukan evolusi terkait desain *prototype* yang telah dibuat. *Prototype* ini juga bisa berasal dari hasil evolusi dari sistem yang sebelumnya sudah ada. Pembuatan *prototype* dimulai dari sketsa sampai desain akhir dari sistem yang akan dikembangkan. *Prototype* ini yang akan dievaluasi oleh *customer* dan akan melakukan evolusi untuk iterasi selanjutnya. Tipe ini juga me

c. *Incremental Prototype*

Tipe ini dipecah menjadi beberapa *prototype* kecil yang berbeda dan dikembangkan secara individual. Pada tahap akhir, *prototype – prototype* tersebut digabungkan menjadi satu produk utuh.

d. *Extreme Prototyping*

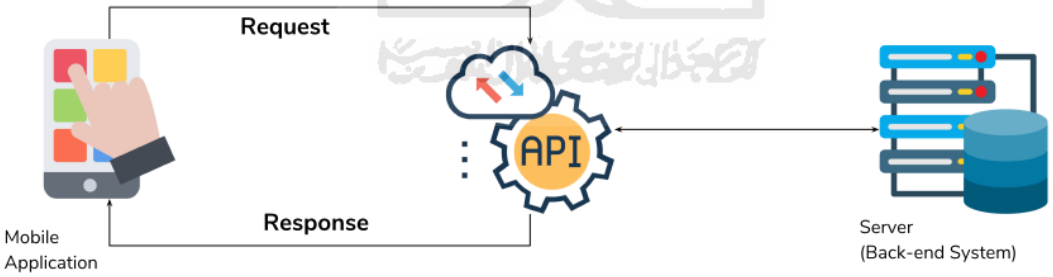
Extreme Prototyping merupakan tipe metodologi *prototyping* yang banyak digunakan untuk pengembangan web. Tipe ini terdiri dari tiga fase yaitu pertama *prototype* ditampilkan

dalam bentuk halaman web dengan format html, kemudian yang kedua adalah menulis kode untuk simulasi proses data, lalu yang ketiga mengimplementasikan dan integrasi proses data tersebut kedalam halaman web html sebelumnya.

Dari empat tipe metodologi *prototyping* penulis menggunakan tipe *Evolutionary Prototype*, karena tipe ini yang menurut penulis paling tepat untuk digunakan. Pada pengembangan aplikasi *mobile* untuk distributor Lacoco ini merupakan hasil evolusi dari sistem Lacoco yang sudah ada sebelumnya yaitu lacoco.co.id. Selain itu tipe ini selalu mendengarkan *feedback* dari *customer*, sehingga desainnya mengalamai evolusi dan desain tersebut lebih sesuai dengan kebutuhan *customer*.

2.3 *Application Programming Interface* (API)

Application Programming Interface atau biasa disingkat API adalah suatu aplikasi berupa antarmuka sebagai protokol komunikasi atau penghubung antara klien dan server untuk pertukaran data. Sehingga jika klien mengirimkan permintaan (dalam format tertentu) ke server, maka server akan membalas respon (dalam format tertentu) atau melakukan suatu aksi (Kothalawala, 2018). Klien dapat berupa aplikasi mobile, desktop, atau web. Server merupakan computer yang menyimpan informasi atau data. Tujuan API adalah mengintegrasikan semua data yang ada di server agar bisa diakses bersamaan oleh pihak klien. Gambar 2.5 menunjukkan bagaimana proses kerja dari sebuah API.



Gambar 2.5 Proses Kerja API

Sumber: Kothalawala (2018)

Pada Gambar 2.5 dicontohkan aplikasi mobile sebagai klien mengirimkan permintaan atau *request* melalui API kepada server, lalu server mengembalikan respon melalui API kepada klien dalam hal ini adalah aplikasi mobile. Format untuk mengirimkan *request* atau mengembalikan respon bisa berupa JSON (*JavaScript Object Notation*), XML (*Extensible* (Guru99, 2020)*Markup Language*), dan lain sebagainya.

2.4 Firebase

Untuk membuat aplikasi yang memiliki fitur notifikasi maka dibutuhkan layanan *Cloud Messaging*. Untuk penelitian ini layanan *Cloud Messaging* yang digunakan adalah Firebase. Firebase didirikan di San Francisco pada tahun 2011 oleh Andrew Lee dan James Tamplin. Perusahaan ini diakuisisi oleh Google pada bulan Oktober 2014 (Tamplin, 2014). Firebase merupakan teknologi yang memungkinkan para pengembang *software* untuk membuat aplikasi tanpa pemrograman di sisi server, dengan kata lain dengan menggunakan Firebase seorang *Software Developer* tidak perlu membuat bagian *backend* dari awal. Dengan demikian pengembangannya menjadi lebih mudah dan lebih cepat. Firebase dapat digunakan di berbagai *platform* di antaranya Android, iOs, web, dan OS X.

Fitur-fitur yang disediakan oleh Firebase adalah *Realtime Database, Crash Reporting, Authentication, Cloud Storage, Cloud Messaging, Analytics,* dan lain-lain. Berikut beberapa penjelasan fitur-fitur dari Firebase :

a. *Realtime Database*

Realtime database adalah *database* berbasis NoSQL yang memungkinkan pengguna untuk menyimpan data dan mensinkronkan data tersebut antar pengguna aplikasi *database* tersebut secara *realtime*. Data yang baru diperbaharui di dalam *database* akan disinkron ke semua perangkat pengguna yang terhubung dalam waktu milidetik. Data tetap akan tersimpan secara lokal pada aplikasi jika sedang *offline* dan secara otomatis akan memperbaharui ke dalam database jika aplikasi terhubung ke internet. Untuk para *software developer* dapat menggunakan fitur ini secara gratis dengan maksimal 100 pengguna pada *database* tersebut dan penyimpanan maksimal 1 GB.

b. *Authentication*

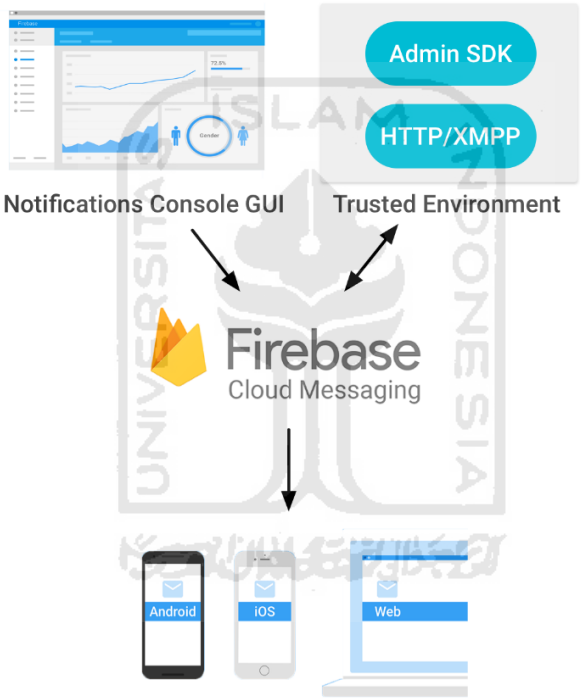
Authentication adalah fitur untuk mengelola akun pengguna aplikasi yang dimiliki oleh aplikasi tersebut dengan cara sederhana dan aman. *Authentication* ini menawarkan beberapa metode untuk otentikasinya yaitu menggunakan email dan sandi, penyedia akun pihak ketiga (*third-party*) seperti akun Google dan Facebook, atau menggunakan sistem akun sendiri.

c. *Cloud Storage*

Cloud Storage dirancang untuk membantu para pengembangan aplikasi dengan mudah dan cepat untuk menyimpan dan menyajikan konten dari pengguna aplikasi tersebut, seperti foto, audio dan video. Fitur ini bisa digunakan secara gratis dengan penyimpanan maksimal 5 GB.

d. *Firebase Cloud Messaging*

Firebase Cloud Messaging atau FCM adalah layanan *cloud* gratis dari Google yang mengizinkan para *software developer* untuk mengirimkan data berupa notifikasi dan pesan ke user dari berbagai platform baik itu Android, iOS, dan aplikasi web. FCM pada awalnya disediakan oleh Firebase Inc., namun pada 21 Oktober 2014 Google mengakuisisi perusahaan tersebut, dan layanan Google Cloud Messaging yang dimiliki sebelumnya oleh Google digantikan menjadi FCM. Implementasi FCM meliputi dua buah komponen yaitu server sebagai pengirim notifikasi atau pesan dan klien sebagai penerima pesan, klien dapat berupa *user* Android, iOS, ataupun web (JavaScript). Lihat Gambar 2.6 untuk gambaran proses kerja FCM.



Gambar 2.6 Proses Kerja FCM

Sumber: Firebase (2018)

Pada Gambar 2.6 dideskripsikan proses kerja dari Firebase Cloud Messaging, yaitu data dapat dikirim melalu Notification Console GUI dari Firebase ataupun melalui *Environment* terpercaya seperti server HTTP/XMPP. Data tersebut kemudian dikirimkan ke Firebase Cloud Messaging lalu diteruskan kepada user atau klien.

2.5 Midtrans *Mobile* SDK

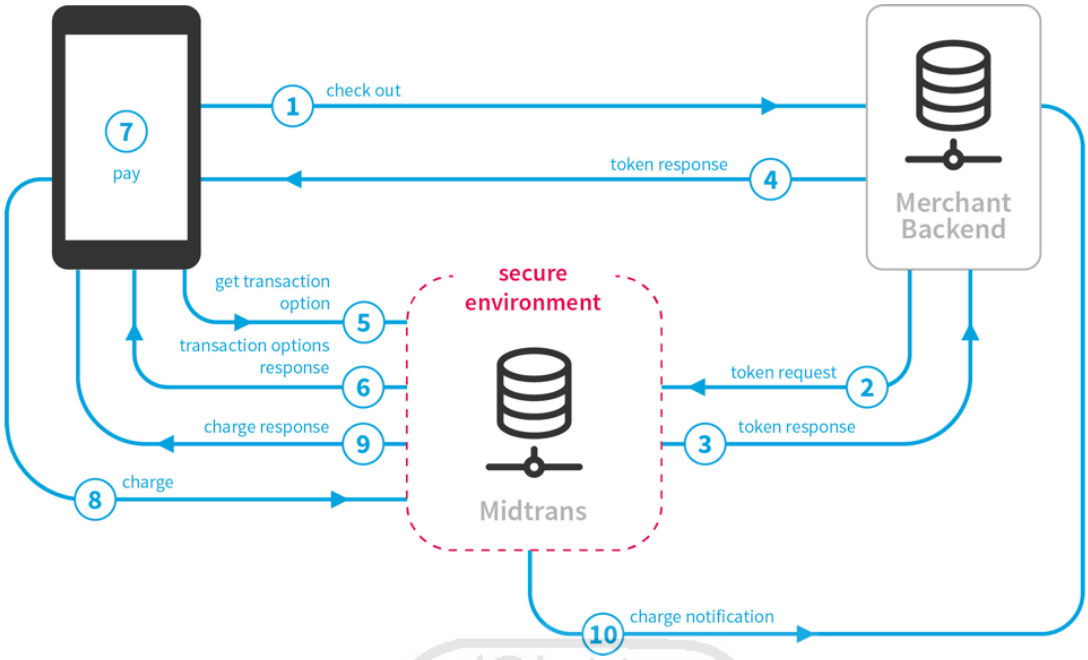
Untuk mengembangkan sebuah aplikasi yang berbasiskan *e-commerce,* biasanya aplikasi tersebut memerlukan layanan *payment gateway*. *Payment gateway* adalah layanan pembayaran yang memberi otorisasi kartu kredit atau metode pembayaran lainnya dalam melakukan proses jual beli. Sebagaimana penelitian yang dilakukan oleh Huda (2018), aplikasi Lacoco yang dikembangkannya telah menggunakan Midtrans sebagai *payment gateway*. Oleh karena itu penulis juga menggunakan Midtrans sebagai layanan pembayaran pada aplikasi Lacoco Mobile.

Midtrans merupakan salah satu *payment gateway* yang dipercaya oleh ribuan bisnis di Indonesia, di antaranya Gojek, Tokopedia, Bukalapak, Traveloka, Garuda Indonesia, dan bisnis – bisnis lainnya. Midtrans memiliki metode pembayaran yang cukup lengkap sesuai dengan kebutuhan warga Indonesia, di antara kartu kredit, kartu kredit virtual, *bank transfer,* kartu debit, *e-wallet,* atau melalui konter – konter yang telah bekerja sama seperti Indomaret dan Alfamart. (Midtrans, 2020).

Midtrans *mobile* SDK merupakan sebuah *library* yang memungkinkan *merchant* untuk mengimplementasikan *payment gateway* melalui aplikasi bergerak *(mobile).* Dengan Midtrans *mobile* SDK ini, maka akan memudahkan *software developer* untuk menerapkan *payment gateway* pada aplikasinya. *Library* ini telah menyediakan *user interface* untuk melakukan transaksi pada semua metode pembayaran yang disediakan oleh Midtrans. Pada penerapannya Midtrans *Mobile* SDK di*install* di dalam aplikasi *mobile* lalu diimplementasikan fitur – fitur yang akan digunakan. Ada 4 pihak yang terlibat dalam proses pembayaran menggunakan Midtrans pada aplikasi *mobile* yaitu :

1. *Customers*¸ dalam hal ini *customers* merupakan orang yang akan melakukan pembayaran
2. *Merchant Backend* atau *Merchant Server*, merupakan halaman *backend* dari pemilik aplikasi
3. Midtrans *mobile* SDK, *library* ini telah di*install* pada aplikasi *mobile*
4. Midtrans *Backend (Payment Processor),* merupakan halaman *backend* dari Midtrans yang menyimpan dan memproses semua pembayaran

Sedangkan alur kerja dari Midtrans *Mobile* SDK akan digambarkan pada Gambar 2.7.



Gambar 2.7 Alur Kerja Midtrans *Mobile* SDK

Sumber: Midtrans (2020)

Gambar 2.7 menjelaskan alur kerja dari Midtrans *Mobile* SDK. Berikut penjelasannya:

1. Seorang *customer* meminta untuk melakukan *checkout*, lalu aplikasi melakukan *request* untuk melakukan proses *checkout* kepada *merchant backend*.
2. Lalu *merchant backend* meminta *request* token kepada Midtrans *backend* dengan informasi *Order* yang telah diisi oleh *customer*.
3. Midtrans *backend* memberikan *response* kepada *merchant backend* yang berisi token yang valid.
4. *Response* token yang didapat dari Midtrans *backend* akan diteruskan ke *Mobile* SDK.
5. *Mobile* SDK melakukan *request* ke Midtrans *backend* untuk mendapatkan informasi metode pembayaran menggunakan token yang didapatkan sebelumnya.
6. Midtrans *backend* memberi *response* ke *Mobile* sdk. Lalu *response* tersebut diolah kemudian menampilkan *user interface* terkait informasi metode pembayaran.
7. *Customer* memilih metode pembayaran lalu mengklik "*Pay*".
8. *Mobile* SDK akan mengirimkan *Charge* yang berisi metode pembayaran beserta informasi detail pembayaran ke Midtrans *backend*, lalu diproses.
9. Midtrans *backend* memberikan *response* ke *Mobile* SDK yang akan memberikan informasi sukses, gagal, atau pending ke aplikasi.

10. Midtrans *backend* akan memberikan notifikasi ke *Merchant Backend* bahwa *customer* telah menyelesaikan transaksinya.

2.6 Laravel

Laravel adalah salah satu *framework* web berbasi PHP yang *open source* dan gratis dengan menggunakan konsep arsitektur Model-View-Controller (MVC), yang dibuat oleh Taylor Otwell. Laravel rilis pertama kali pada Juni 2011 dibawah lisensi MIT, dan saat ini telah mencapai versi 7.0 (Surguy, 2013). Ada banyak fitur yang disediakan oleh Laravel, salah satunya adalah Restful Controllers yaitu memudahkan *developer* untuk membuat suatu API (*Application Programming Interface*).

BAB III
ANALISIS DAN PERANCANGAN SISTEM

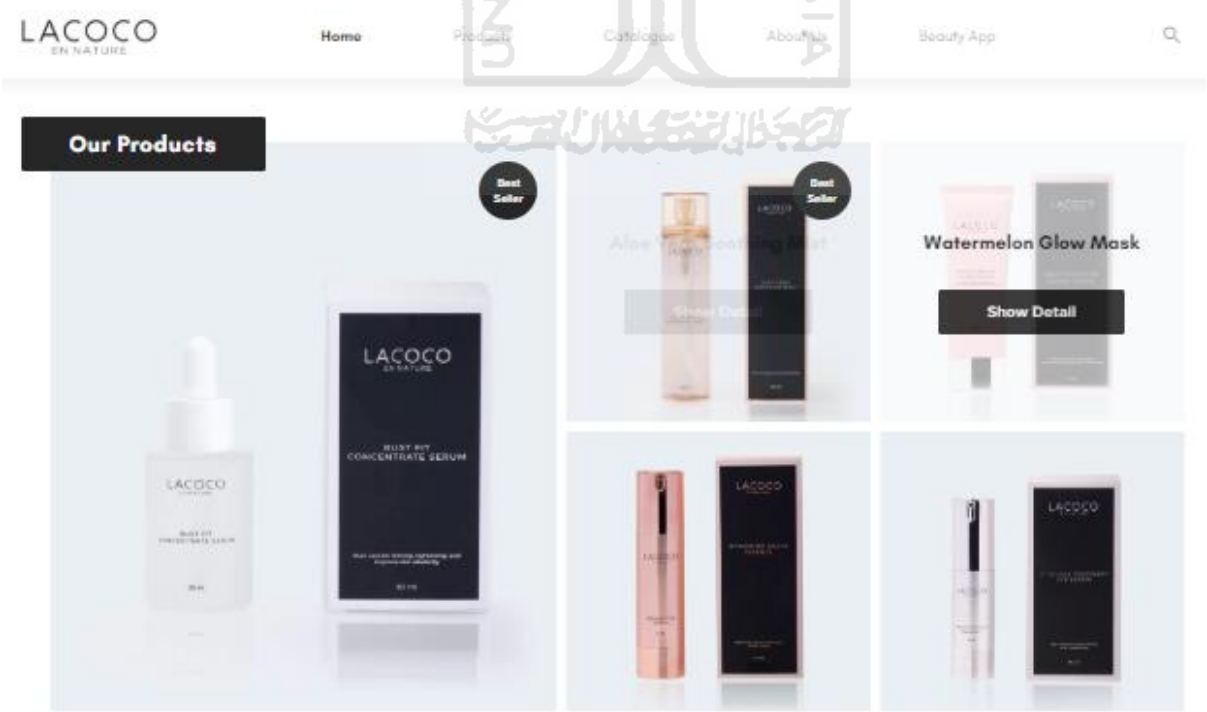
3.1 Analisis Kebutuhan

Untuk mendapatkan hasil yang komprehensif, analisis kebutuhan dilakukan melalui 3 tahapan berikut :

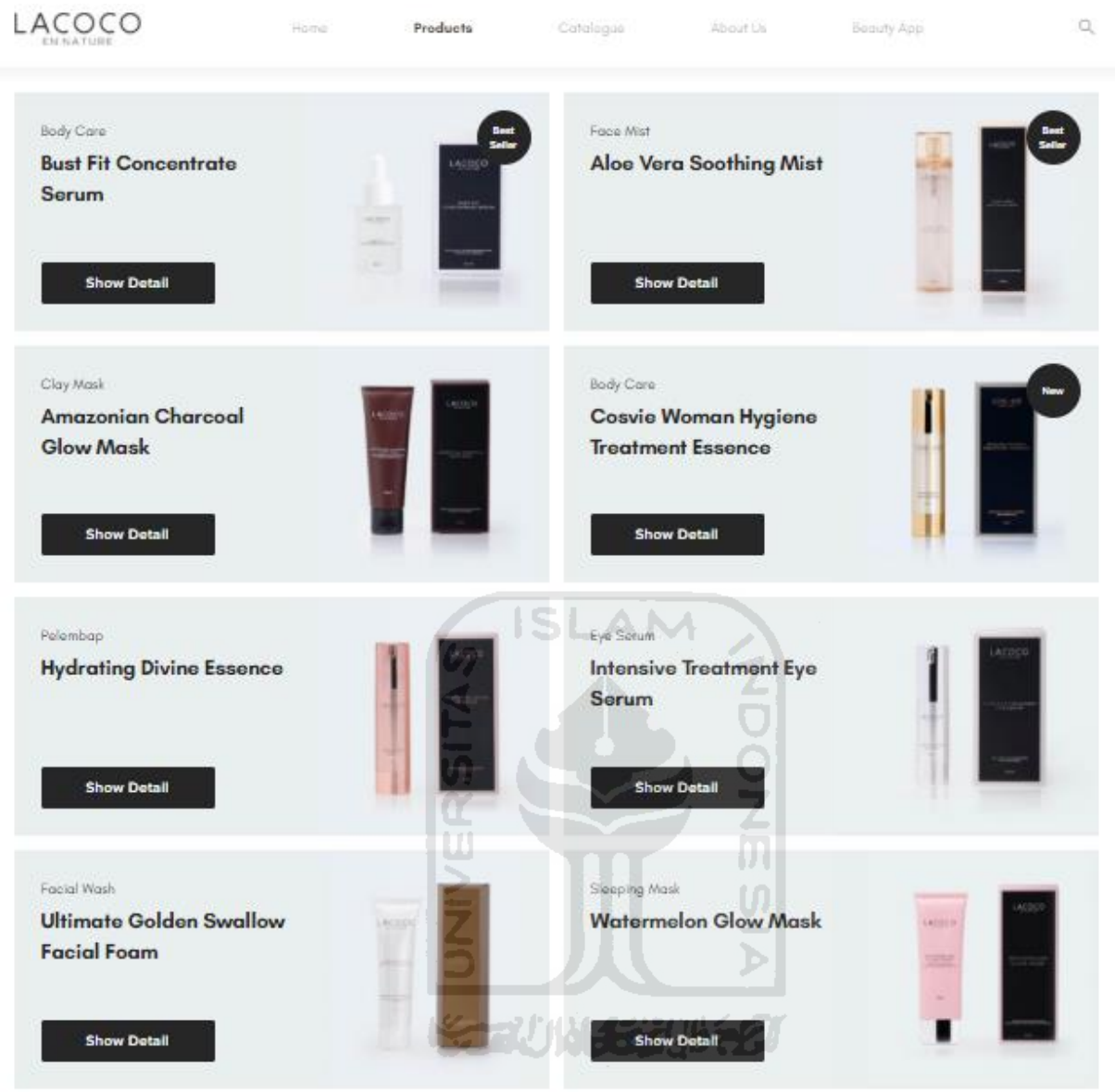
a. Observasi

Tahapan ini dimulai dengan mengamati website PT. AVO (Lacoco.co.id) yang bertujuan untuk mendapatkan gambaran umum tentang produk Lacoco dan *user interface* yang digunakan. Selain itu observasi juga dilakukan dengan cara mempelajari penelitian sebelumnya yang berhubungan, yaitu tugas akhir Aris Nurul Huda (2018) dengan judul Pengembangan Sistem Informasi dan Penjualan Lacoco Berbasis Website. Melalui dua langkah ini maka prediksi awal tentang aplikasi yang dibutuhkan bisa dilakukan.

Berdasarkan pengamatan terhadap website lacoco.co.id diketahui bahwa hingga saat penulisan laporan ini terdapat 8 produk yang telah diproduksi oleh PT. AVO. Lihat Gambar 3.1 untuk halaman awal dari lacoco.co.id dan Gambar 3.2 adalah halaman *products* Lacoco yang dijual oleh PT. AVO.



Gambar 3.1 Halaman Awal Lacoco.co.id



Gambar 3.2 Halaman *Products* Lacoco.co.id

Selain itu diketahui pula bahwa tema yang digunakan pada lacoco.co.id adalah *grid style* dengan tema warna yang mendominasi adalah hitam dan putih. Observasi terhadap skripsi Huda (2018) memberikan informasi tentang sistem penjualan dan sistem informasi berbasis website yang kedepannya akan digunakan oleh PT. AVO. Pada saat observasi website dilakukan, lacoco.co.id masih hanya berfungsi sebagai katalog online. Dari skripsi Huda (2018) diketahui bahwa kedepannya website tersebut akan dirancang agar mampu mengakomodasi pembelian dari *customer*. Di dalam pembeliannya, *customer* tidak membeli langsung kepada PT. AVO, akan tetapi diarahkan ke distributor – distributor

yang tersebar di seluruh Indonesia. Untuk mendukung hal tersebut maka Huda (2018) telah mengembangkan fitur – fitur berikut

a. Halaman *Customer Home,*
b. Halaman Produk,
c. Halaman Detail Produk,
d. Halaman *Catalogue,*
e. Halaman *About Us,*
f. Halaman *Find Store,*
g. Halaman *Login and Registration Customer,*
h. Halaman Registrasi Distributor,
i. Halaman Distributor *Account,*
j. Halaman *Buy Product Distributor,*
k. Halaman *Popup Cart,*
l. Halaman *Checkout,*
m. Halaman *Order History Distributor,*
n. Halaman *Point History,*
o. Halaman *Dashboard* Admin,
p. Halaman *Products* Admin,
q. Halaman *Customers* Admin,
r. Halaman *Resellers* Admin,
s. Halaman *Users* Admin,
t. Halaman *Loyalty Admin,*
u. Halaman *Reports* Admin,
v. Halaman *Referrals* Admin

b. Wawancara

Tahapan berikutnya dari analisis kebutuhan adalah wawancara dengan Aris Nurul Huda. Informan tersebut dipilih dengan dua pertimbangan. Pertama, Aris Nurul Huda merupakan *Chief Information Officer* (CIO) PT. AVO. Kedua, informan tersebut juga merupakan penulis skripsi yang telah disebutkan pada tahapan sebelumnya (tahapan observasi).

Tahapan wawancara ini dilakukan pada 12 September 2018 di kantor PT. AVO. Penulis mengajukan beberapa pertanyaan pada proses wawancara ini, di antaranya :

- Bagaimana gambaran dan proses bisnis aplikasi Lacoco yang telah dikembangkan?

- Permasalahan apa saja yang belum terselesaikan pada skripsi Huda (2018)?
- Permasalahan apa saja yang mungkin muncul ketika *website* Lacoco yang telah dikembangkan oleh Huda (2018) digunakan oleh distributor?
- Apakah PT. AVO memiliki kebutuhan lainnya untuk sistem yang akan dikembangkan?

Jawaban atas empat pertanyaan di atas tertuang pada Tabel 3.1.

Tabel 3.1 Daftar Pertanyaan dan Jawaban Wawancara

Pertanyaan	**Jawaban**
Bagaimana gambaran dan proses bisnis aplikasi Lacoco yang telah dikembangkan?	1. PT AVO memiliki ribuan distributor yang tersebar di seluruh Indonesia 2. Untuk menjadi distributor harus melalui proses pendaftaran terlebih dahulu lalu melalui proses peninjauan yang dilakukan oleh admin. Jika disetujui, distributor akan diberitahukan melalui email oleh sistem secara otomatis 3. Distributor harus membeli produk atau barang ke AVO untuk menyetok barang 4. Adapun untuk *customer* yang ingin membeli produk Lacoco ada dua pilihan yaitu dengan membuat akun terlebih dahulu atau sebagai tamu (*as a guest*) tanpa perlu membuat akun terlebih dahulu 5. *Customer* memilih barang yang akan dibeli, lalu perlu memilih distributor 6. Pesanan dari *customer* masuk ke list pesanan yang ada pada distributor yang dipilih. 7. Pesanan akan diproses ketika *customer* sudah melakukan pembayaran. Jika tidak melakukan pembayaran dalam waktu tertentu maka pesanan secara otomatis dibatalkan oleh sistem.

Pertanyaan	Jawaban
	8. Jika sudah dibayar, distributor harus memperbaharui status pesanan tersebut, apakah delay atau sudah dikirimkan. 9. Jika sudah dikirimkan distributor perlu memasukkan nomor resi (*waybill).*
Permasalahan apa saja yang belum terselesaikan pada skripsi Huda (2018)?	Pada saat *customer* melakukan *checkout* pembelian barang, *customer* masih memilih secara manual distributor mana yang akan dibeli sesuai list distributor yang disediakan (list dipastikan memiliki stok). Oleh karena itu pada saat pendadaran skripsi, penguji meminta agar *customer* pada saat *checkout* tidak perlu memilih distributor, melainkan lebih baik distributor dipilih secara otomatis dari sistem. Di antara faktor pemilihan distributor secara otomatis adalah distributor yang memiliki stok, lokasi terdekat, dan lain – lain.
Permasalahan apa saja yang mungkin muncul ketika *website* Lacoco yang telah dikembangkan oleh Huda (2018) digunakan oleh distributor?	Distributor tidak mendapatkan pemberitahuan ketika ada pesanan baru atau pesanan yang diperbaharui secara otomatis oleh sistem baik karena sudah dilakukan pembayaran atau dibatalkan karena belum membayar dalam waktu tertentu. Sehingga tanpa pemberitahuan ini memungkinkan terjadi keterlambatan proses.
Apakah PT. AVO memiliki kebutuhan lainnya untuk sistem yang akan dikembangkan?	1. Notifikasi ke Distributor jika ada pesanan baru atau pesanan yang diperbaharui oleh sistem. 2. Analisis terkait pesanan yang masuk dari *Customer* ke Distributor, dalam bentuk grafik. 3. Melihat pesanan yang masuk dari *Customer* ke Distributor dan memperbaharui status pesanan.

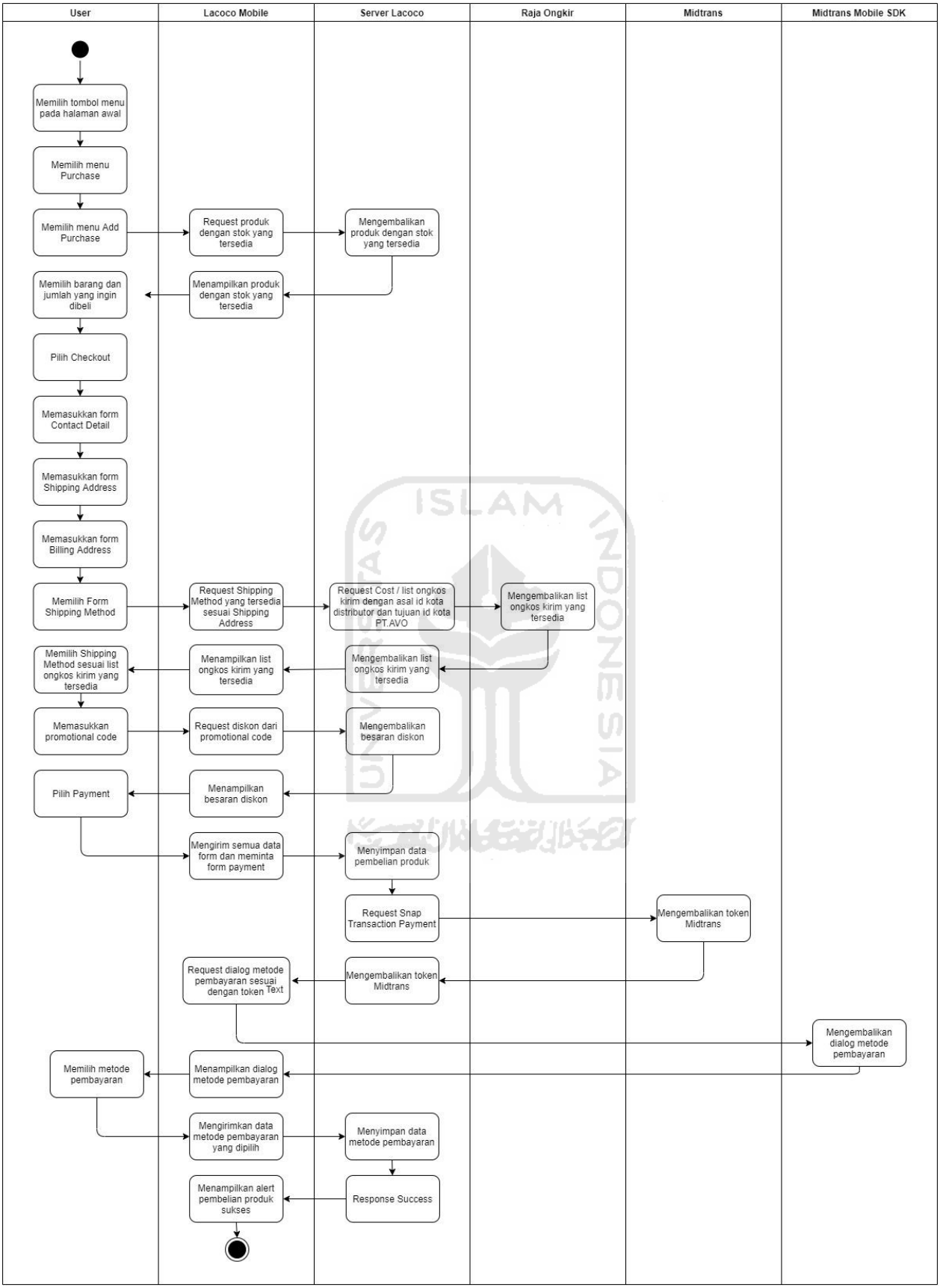
Pertanyaan	Jawaban
	4. Pembelian produk dari Distributor ke PT AVO.

c. Pasca Wawancara (Sintesis)

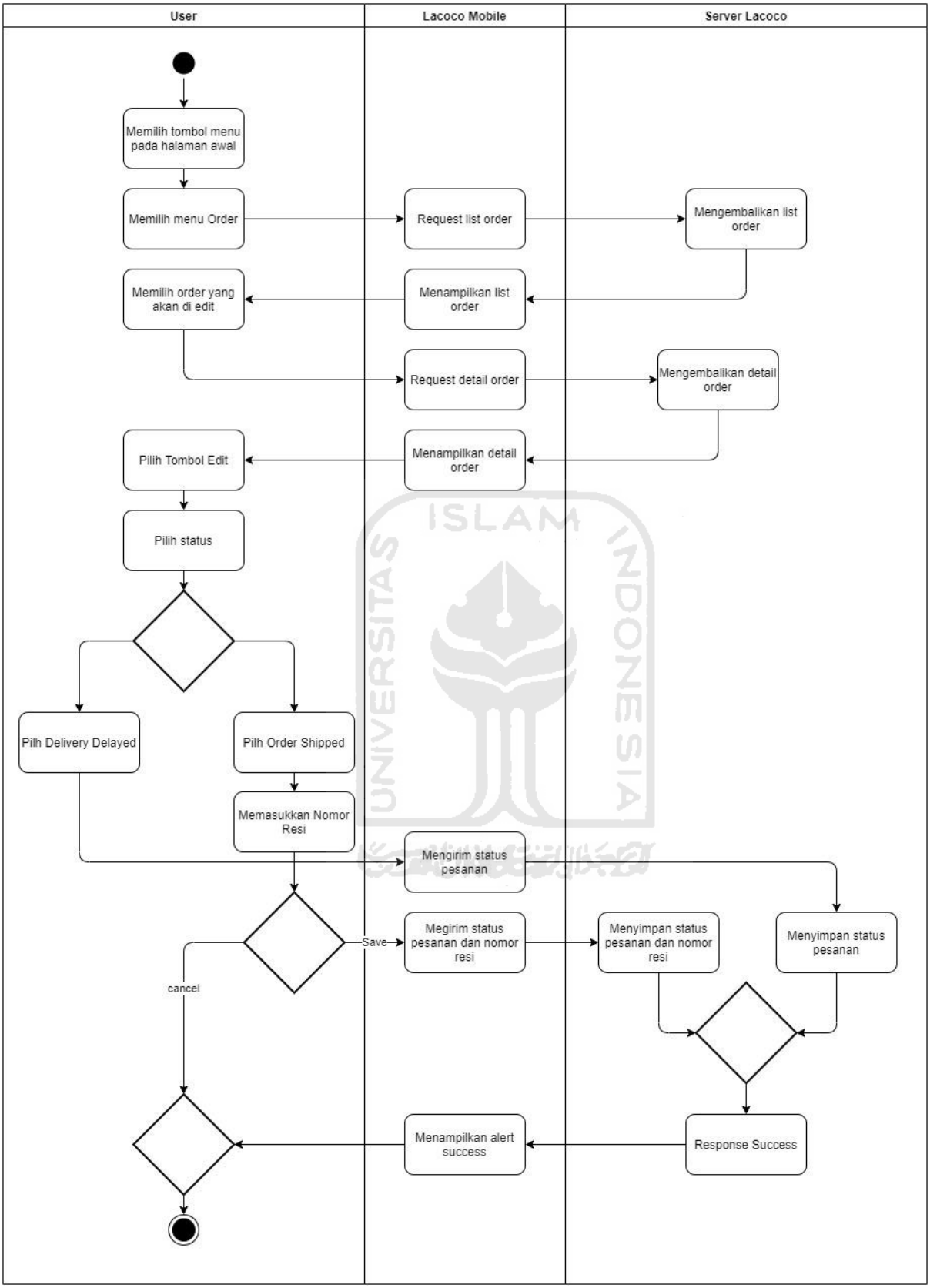
Berdasarkan penjelasan pada tahapan observasi maka didapatkan gambaran awal bahwa aplikasi yang akan dibangun berorientasi pada kebutuhan distributor, aplikasi ini merupakan aplikasi bergerak (*mobile*) berbasiskan Android. Dengan kata lain aplikasi yang akan dibangun akan digunakan oleh para distributor PT. AVO dengan fitur utamanya adalah notifikasi dan analisis terkait riwayat penjualan.

Lebih lanjut, berbagai jawaban wawancara yang terdapat pada Tabel 3.1 mengkonfirmasi konsep awal yang didapat dari tahapan observasi (notifikasi dan analisis terkait riwayat penjualan). Selain itu disimpulkan bahwa perlu ada dua fitur tambahan lainnya pada aplikasi yang akan dibangun yaitu manajemen pesanan dan pembelian produk dari distributor ke PT. AVO. Dari wawancara ini diketahui pula, di antara fitur yang diharapkan oleh PT. AVO di dalam aplikasi yang akan dibangun adalah pemilihan distributor secara otomatis ketika *customer* melakukan *checkout* melalui website, dengan pertimbangan stok, lokasi terdekat, dan lain – lain. Akan tetapi, dengan menimbang keterbatasan waktu dan sumber daya yang tersedia di dalam pengerjaan skripsi ini, fitur tersebut tidak diakomodasi di dalam penelitian ini. Namun dengan demikian fitur tersebut akan disarankan menjadi penelitian berikutnya yang akan disampaikan di subbab saran (subbab 5.3).

Dari berbagai kebutuhan tersebut, maka terdapat dua kebutuhan yang dapat divisualisasikan ke dalam *Activity Diagram. Activity Diagram* adalah bentuk visual dari alur kerja yang di dalamnya memiliki tindakan atau aktivitas. Dua kebutuhan tersebut dapat dilihat pada Gambar 3.3 dan Gambar 3.4.



Gambar 3.3 *Activity Diagram* Pembelian Produk Baru



Gambar 3.4 *Activity Diagram* Ubah Status Order

Lebih detailnya, berbagai kebutuhan tersebut akan diakomodasi ke dalam halaman – halaman pada aplikasi, yang dijelaskan pada Tabel 3.2. Selain itu tabel tersebut juga menjelaskan hubungannya dengan *website* Lacoco.

Tabel 3.2 Daftar Rencana Halaman yang akan Dibangun Di Aplikasi

No.	**Nama Halaman**	**Fungsi**	**Fitur dari *Website* Lacoco yang Digunakan**
1	Halaman *Splashscreen*	Menampilkan identitas perusahaan	-
2	Halaman *Login*	Otentikasi distributor	Halaman *Login*
3	Halaman *Register*	Daftar menjadi distributor	Halaman Registrasi
4	Halaman Menu	Menampilkan menu	-
5	Halaman *Analytics Sub Orders*	Grafik terkait pesanan	-
6	Halaman *Analytics Sub Profits*	Grafik terkait pendapatan	-
7	Halaman *Analytics Sub Items*	Grafik terkait item yang terjual	-
8	Halaman *Filter Analytics Sub Orders*	*Filter* grafik pesanan	-
9	Halaman *Filter Analytics Sub Profits*	*Filter* grafik pendapatan	-
10	Halaman *Filter Analytics Sub Items*	*Filter* grafik item yang terjual	-
11	Halaman *Notifications*	Menampilkan *list* notifikasi	-
12	Halaman *Orders*	Menampilkan *list* pesanan yang masuk dari *customer*	-
13	Halaman *Filter Orders*	*Filter* pesanan yang masuk dari *customer*	-
14	Halaman *Detail Order*	Menampilkan detail pesanan yang masuk dari *customer*	-
15	Halaman Ubah *Order Status*	Mengubah status pesanan	-
16	Halaman *Input Waybill*	Memasukkan nomor resi	-

No.	Nama Halaman	Fungsi	Fitur dari *Website* Lacoco yang Digunakan
17	Halaman *Purchases*	Menampilkan *list* pembelian oleh distributor	Halaman *Order History Distributor*
18	Halaman *Filter Purchases*	*Filter* pembelian oleh distributor	-
19	Halaman *New Purchase*	Menambahkan pembelian produk baru oleh distributor	Halaman *Buy Product Distributor*
20	Halaman *List Form New Purchase*	Menampilkan *list form* yang harus dilengkapi untuk pembelian produk baru oleh distributor	-
21	Halaman *Form Contact Details New Purchase*	*Form* detail kontak pembeli untuk pembelian produk baru	-
22	Halaman *Form Shipping Address New Purchase*	*Form* alamat pengiriman untuk pembelian produk baru	-
23	Halaman *Form Add New Address*	*Form* untuk menambahkan alamat baru	
24	Halaman *Form Billing Address New Purchase*	*Form* alamat pembayaran untuk pembelian produk baru	-
25	Halaman *Form Shipping Method New Purchase*	*Form* pemilihan metode pengiriman untuk pembelian produk baru	-
26	Halaman *Payment New Purchase*	Menampilkan metode pembayaran untuk pembelian produk baru	-

No.	Nama Halaman	Fungsi	Fitur dari *Website* Lacoco yang Digunakan
27	Halaman *Points*	Menampilkan *list* poin yang dimiliki distributor	Halaman *Point History*
28	Halaman *Settings*	Pengaturan	-

3.2 Perancangan Sistem

Setelah melakukan analisis kebutuhan selanjutnya tahapan metodologi *prototyping* adalah tahapan desain atau perancangan sistem. Seperti yang sudah dijelaskan pada subbab 2.2.1 tahapan perancangan sistem terdiri dari empat tahap yaitu membuat desain sederhana, membuat *prototype*, melakukan evaluasi ke *customer* atau pemilik aplikasi, dan merevisi *prototype* jika desain belum sesuai keinginan *customer*. Empat tahap tersebut terus berulang atau melakukan iterasi sampai semua fitur sudah selesai dirancang dan desain sudah sesuai dengan kebutuhan dan keinginan *customer*. Berikut penjelasan empat tahapan yang akan penulis lakukan.

1. Membuat Desain Sederhana

Desain sederhana dibuat berdasarkan hasil analisis kebutuhan. Tujuan desain sederhana ini dibuat adalah untuk memberikan gambaran kasar kepada *customer* atau pemilik aplikasi, dalam hal ini adalah PT. AVO. Desain sederhana dibuat menggunakan aplikasi Figma. Desain ini dibuat berdasarkan konsep *low fidelity*. *Low fidelity* adalah sketsa dari elemen – elemen dasar dari sebuah antar muka aplikasi, seperti tombol, teks, input data, dan lain – lain.

2. Membuat Prototype

Prototype merupakan pengembangan lebih lanjut dari desain sederhana. Sebagaimana desain sederhana dibuat menggunakan aplikasi Figma, *prototype* juga dibuat di aplikasi yang sama. *Prototype* ini dibuat berdasarkan konsep *high fidelity*. *High fidelity* adalah desain yang lebih detail dari *low fidelity*, sehingga lebih mudah dipahami oleh *user*. *High fidelity* mencakup pewarnaan, ikon, dan lain – lain. Dengan kata lain *prototype* merupakan perwujudan aplikasi yang akan dibuat dalam bentuk gambar.

3. Evaluasi *Prototype*

Dengan adanya peluang perbedaan konsep yang dibangun di dalam *prototype* dengan apa yang telah dirumuskan pada tahapan analisis kebutuhan, maka diperlukan adanya evaluasi untuk memastikan konsep yang dibangun masih sesuai dengan konsep yang disepakati di awal.

Tahapan ini dilakukan dengan proses tatap muka, dimana *prototype* yang telah dibangun ditunjukkan secara langsung kepada PT. AVO, yang dalam hal ini diwakili oleh Aris Nurul Huda dan Arif Budiman. Evaluasi *prototype* ini dilakukan pada 9 dan 16 Maret 2020, di kantor PT. AVO. Hasil evaluasi dicatat dengan metode *note-taking*.

4. Revisi *Prototype*

Setelah mendapatkan *feedback* atau evaluasi terhadap *prototype* yang telah disampaikan kepada pemilik aplikasi, maka selanjutnya evaluasi tersebut ditindak lanjuti. *Prototype* akan didesain ulang atau direvisi sesuai dengan evaluasi tersebut.

Perancangan sistem pada penelitian kali ini dilakukan dengan dua tahap dan dua iterasi. Dua tahap ini terjadi karena sistem yang akan dibangun cukup besar dan memiliki fitur dan halaman yang banyak, maka proses perancangan dibagi dua tahap. Tahap pertama perancangan terkait fitur analisis terkait riwayat penjualan dan notifikasi. Tahap kedua adalah manajemen pesanan dan pembelian produk. Setiap tahap tersebut akan disampaikan kepada *customer* pada setiap iterasi. Sehingga fitur yang ada pada tahap pertama akan disampikan pada iterasi 1 dan fitur yang ada pada tahap kedua akan disampaikan pada iterasi 2, dan juga *feedback* yang disampaikan oleh *customer* pada iterasi 1 akan dievaluasi pada iterasi 2.

3.2.1 Iterasi 1

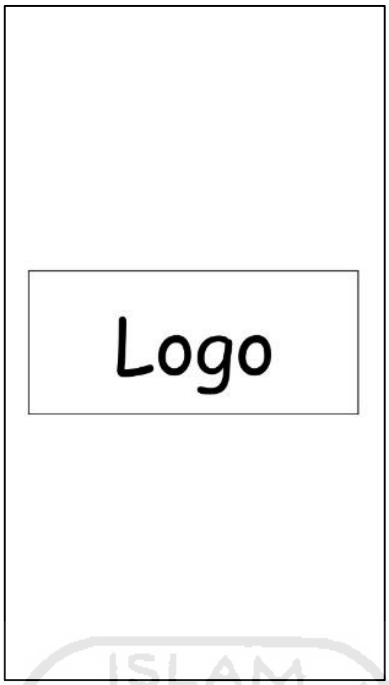
Fitur yang menjadi objek iterasi 1 adalah analisis terkait riwayat penjualan dan notifikasi. Akan tetapi, sebelum fitur tersebut dibuat, terlebih dahulu perlu ada halaman pelengkap seperti *splashscreen*, halaman *login*, dan halaman menu. Hasil empat tahapan pada perancangan sistem adalah sebagai berikut.

1. Membuat Desain Sederhana

Beberapa halaman di dalam aplikasi yang akan dibangun memiliki tampilan yang mirip, maka tidak semua halaman – halaman yang telah disebutkan pada Tabel 3.2 dibuat desain sederhananya. Sebagai contoh halaman *Analytics Sub Orders*, halaman *Analytics Sub Profits*, dan halaman *Analytics Sub Items* ketiganya memiliki kemiripan desain sehingga desain sederhana yang dibuat hanya satu yaitu yang akan disampaikan pada poin D.

a. Halaman *Splashscreen*

Halaman s*plashscreen* adalah halaman statis yang hanya memunculkan logo yang direncanakan akan tampil selama 3 detik. Lihat Gambar 3.5 untuk melihat lebih detail.



Gambar 3.5 Mockup Halaman *Splashscreen*

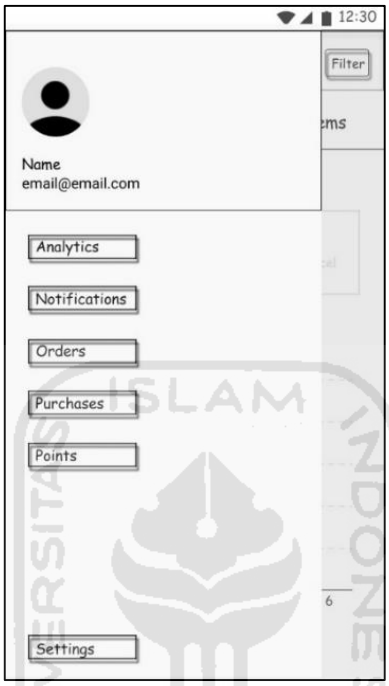
b. Halaman *Login*

Halaman *login* terdiri dari logo, *input text username*, *input text password.* Selain itu pada halaman ini pengguna diberikan dua pilihan *login* dan *register* yang akan dibuat dalam bentuk *button*. Lihat Gambar 3.6 untuk lebih detail.

Gambar 3.6 Mockup Halaman *Login*

c. Halaman Menu

Halaman menu ditampilkan dalam bentuk *navigation drawer*, yang terdiri dari *user profile*, tombol menu *analytics*, tombol menu *notification*, tombol menu *orders*, tombol menu *purchases*, tombol menu *points*, dan tombol menu *settings*. *User profile* dihalamn ini terdiri dari foto *profile*, nama pengguna, dan e-mail pengguna. Lihat Gambar 3.7 untuk lebih detail.



Gambar 3.7 Mockup Halaman Menu

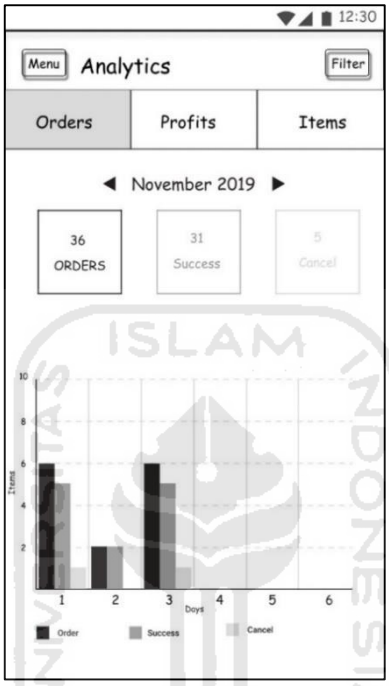
d. Halaman Analis Terkait Riwayat Penjualan

Halaman analisis terkait riwayat penjualan memiliki *action bar* dengan label *Analytics*, memiliki tombol menu pada sisi kiri, dan tombol filter pada sisi kanan. Sedangkan konten utama yang akan ditampilkan dalam bentuk *tab layout* yang terdiri dari 3 *tab* yaitu *Orders, Profits,* dan *Items.* Pada setiap tab pada menu ini akan menampilkan tanggal sesuai *fitler* yang dipilih pengguna, yang bisa di *swap* kanan untuk tanggal berikutnya, dan *swap* kiri untuk tanggal sebelumnya.

Pada *tab Orders* data ditampilkan dalam bentuk *bar chart,* yang terdiri dari jumlah *orders,* jumlah order yang berhasil (*success),* dan jumlah order yang batal (*cancel).* Selain menampilkan *chart*, *tab* ini menampilkan data total dari setiap *orders, success,* dan *cancel.* Lihat Gambar 3.8 untuk lebih detail.

Sebagaimana *tab orders*, pada *tab profits* pun data ditampilkan dalam bentuk *bar chart*, yang terdiri dari *revenue* dan *income.* Selain itu pula menampilkan data total dari setiap *revenue*, dan *income.*

Sedangkan pada *tab items* data ditampilkan dalam bentuk *pie chart*, yang menampilkan jumlah prorduk terjual pada setiap item produknya. Banyaknya data *chart* yang ditampilkan bergantung pada *filter* yang dipilih oleh pengguna.



Gambar 3.8 Mockup Halaman *Analytics*

e. Halaman Notifikasi

Halaman notifikasi memiliki *action bar* dengan label *Notifications* dan tombol menu pada sisi kiri. Sedangkan konten utama ditampilkan dalam bentuk *list*. Setiap *item list* akan menampilkan ikon sesuai tipe notifikasi, teks detail notifikasi, waktu notifikasi, dan memiliki warna *background* tertentu sebagai penanda bahwa notifikasi tersebut belum dilihat. Lihat Gambar 3.9 untuk lebih detail.

Gambar 3.9 Mockup Halaman *Notifications*

2. Membuat *Prototype*

Berdasarkan hasil desain sederhana tersebut, kemudian dibuat *prototype* dimana pada tahapan ini sudah diberi pewarnaan dan penempatan ikon. *Prototype* untuk halaman *splashscreen*, halaman *login*, halaman menu, halaman analitik, dan halaman notifikasi bisa dilihat dari Gambar 3.10 sampai Gambar 3.20.

a. Halaman *Splashscreen*

Halaman *splashscreen* pada Gambar 3.10 akan menampilkan logo *Lacoco En Nature* selama 3 detik.

Gambar 3.10 *Prototype* Halaman *Splashscreen*

b. Halaman *Login*

Halaman *login* pada Gambar 3.11 memiliki *input text* email dengan ikon email pada sisi kiri dan *input text password* dengan ikon gembok pada sisi kiri dan ikon mata pada sisi kanan. *Button login* memiliki warna *background* hitam dan warna *font* putih.

Gambar 3.11 *Prototype* Halaman *Login*

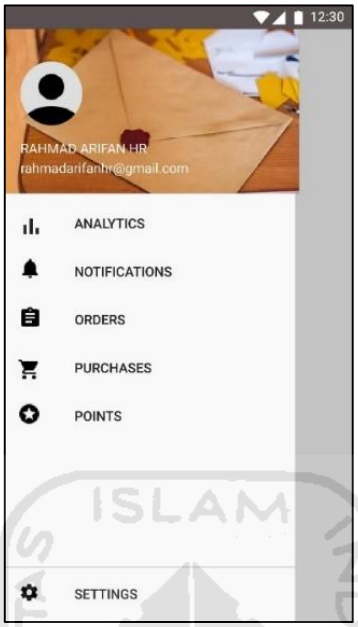
c. Halaman *Register*

Halaman *register* pada Gambar 3.12 terdiri dari logo dan beberapa *input text* antara lain *name, email, phone number, instagram (optional),* dan *addres.* Selain itu pada halaman ini terdapat *spinner province* dan *city.* Lalu terdapat *button Register* untuk mendaftar menjadi distributor baru, dan *button Login* bagi pengguna yang sudah terdaftar. *Button register* memiliki warna *background* hitam dan warna *font* putih.

Gambar 3.12 *Prototype* Halaman *Register*

d. Halaman Menu

Untuk List Menu yang tersedia pada Aplikasi Lacoco Mobile dapat dilihat pada Gambar 3.13.



Gambar 3.13 *Prototype* Halaman List Menu

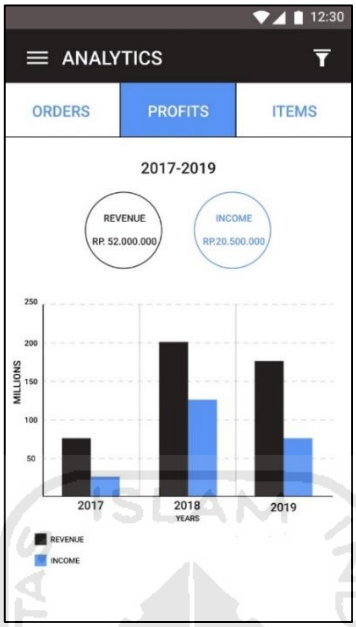
e. Halaman *Analytics Sub Orders*

Halaman *Analytics Sub Orders* pada Gambar 3.14 terdapat *bar chart* yang menampilkan jumlah *orders* dalam warna hitam, jumlah *order success* dalam warna biru, dan jumlah *order cancel* dalam warna merah.

Gambar 3.14 *Prototype* Halaman *Analytics Sub Orders*

f. Halaman *Analytics Sub Profits*

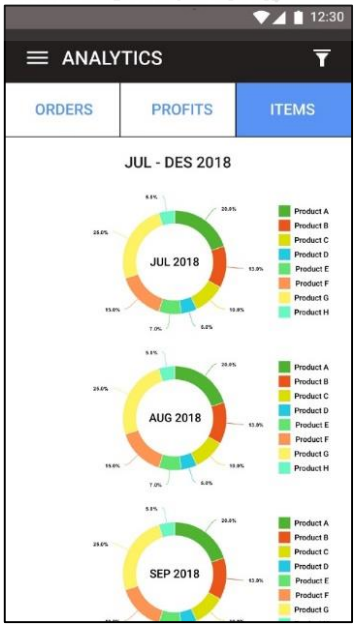
Halaman *Analytics Sub Profits* pada Gambar 3.15 terdapat *bar chart* yang menampilkan jumlah *revenue* dalam warna hitam dan *income* dalam warna biru.



Gambar 3.15 *Prototype* Halaman *Analytics Sub Profits*

g. Halaman *Analytics Sub Items*

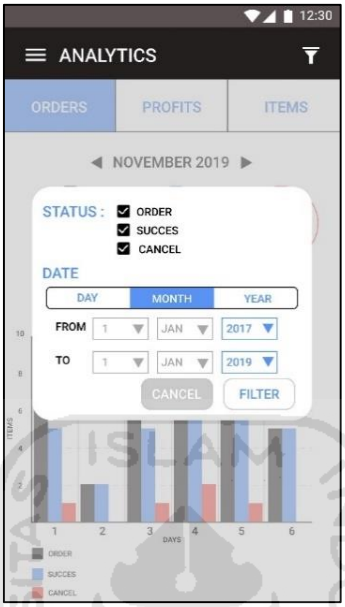
Halaman *Analytics Sub Items* pada Gambar 3.16 terdapat *pie chart* yang menampilkan jumlah item yang terjual untuk setiap produknya, dengan warna yang sesuai dengan penanda untuk setiap produk.



Gambar 3.16 *Prototype* Halaman *Analytics Sub Items*

h. Halaman *Filter Analytics Sub Orders*

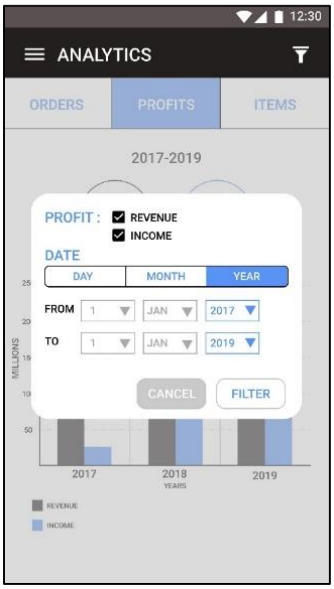
Untuk melakukan *filter* pada halaman *Analytics Sub Orders* terdapat tiga *status* yang bisa dipilih untuk ditampilkan yaitu *order*, *success*, dan *cancel*. Lalu bisa di filter berdasarkan *date range*. Lihat Gambar 3.17 untuk lebih detail.



Gambar 3.17 *Prototype* Halaman *Filter Analytics Sub Orders*

i. Halaman *Filter Analytics Sub Profits*

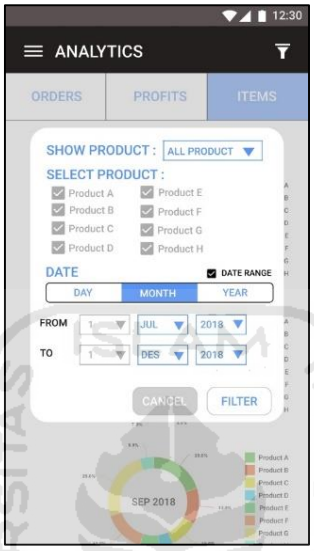
Untuk melakukan *filter* pada halaman *Analytics Sub Profits* terdapat dua *profit* yang bisa dipilih untuk ditampilkan yaitu *revenue* dan *income*. Lalu bisa di filter berdasarkan *date range* Lihat Gambar 3.18 untuk lebih detail.



Gambar 3.18 *Prototype* Halaman *Filter Analytics Sub Profits*

j. Halaman *Filter Analytics Sub Items*

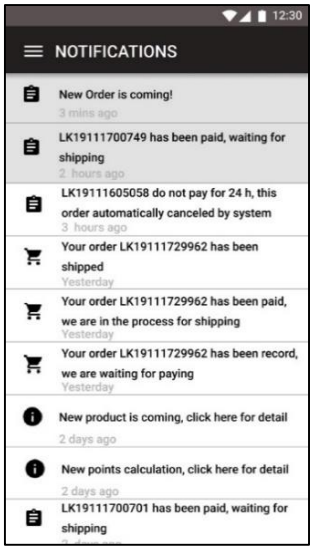
Untuk melakukan *filter* pada halaman *Analytics Sub Items*, terdapat pilihan dalam menampilkan produk yaitu *all product* atau *any product*. Jika yang dipilih adalah *all product* maka semua *product* akan ditampilkan. Jika yang dipilih adalah *any product* maka pengguna harus memilih *product* yang akan ditampilkan. Lalu bisa di filter berdasarkan *date range*. Lihat Gambar 3.19 untuk lebih detail.



Gambar 3.19 *Prototype* Halaman *Filter Analytics Sub Items*

k. Halaman *Notifications*

Pada halaman *notifications* (pada Gambar 3.20) setiap notifikasi yang belum terbaca maka akan diberi warna *background* abu – abu muda. Untuk ikon yang ditampilkan pada setiap *item list* tergantung pada jenis notifikasi, yang dijelaskan pada Tabel 3.3.



Gambar 3.20 *Prototype* Halaman *Notifications*

Tabel 3.3 Jenis Ikon pada Halaman Notifications

Ikon	Penjelasan
	Ikon untuk jenis notifikasi *order* atau pesanan yang masuk atau diperbaharui
	Ikon untuk jenis notifikasi *purchase* atau pembelian yang diperbaharui
	Ikon untuk jenis notifikasi *points* yang diperbaharui
	Ikon untuk jenis notifikasi *news* atau informasi baru

3. Evaluasi dan Revisi *Prototype*

Setelah membuat *prototype,* selanjutnya desain tersebut dievaluasi oleh PT. AVO sebagai pemilik aplikasi dan revisi *prototype* juga sekaligus penulis jabarkan pada poin – poin evaluasi yang akan disampaikan. Evaluasi *prototype* ini dilakukan pada 9 Maret 2020, di kantor PT. AVO. Poin – poin yang disampaikan dalam proses evaluasi tersebut dirangkum sebagaimana paragraf – paragraf di bawah.

a. Halaman Analytics Tab Orders

Pada *prototype* yang disampaikan, data *Analytics Tab Orders* ditampilkan menggunakan *bar chart.* Namun setelah dilakukan evaluasi, PT. AVO meminta agar data yang ditampilkan menggunakan *chart* yang bergaris atau *line chart*, sehingga dalam satu baris memiliki 3 status, dengan warna yang berbeda untuk setiap status. Lihat Gambar 3.21 untuk hasil revisi desain.

Gambar 3.21 Hasil Evaluasi *Prototype* Menu *Analytics Tab Orders*

b. Halaman Analytics Tab Profits

Terdapat beberapa poin yang disampaikan oleh PT. AVO dalam evaluasi terkait *Analytics Tab Profits.* Poin – poin tersebut dirangkum pada Tabel 3.4.

Tabel 3.4 Hasil Evaluasi Menu *Analytics Tab Profits*

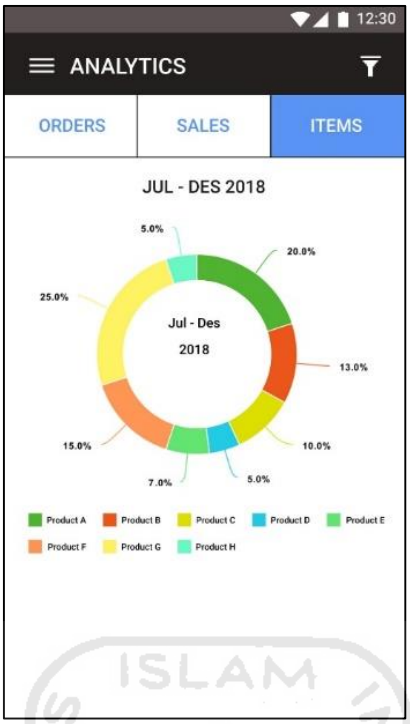
Prototype	**Hasil Evaluasi**
Nama menu tab “Profits”	Nama menu tab “Sales”
Data yang ditampilkan adalah *Revenue* dan *Income*	Data yang ditampilkan hanya *Revenue*, tetapi ditambahkan dengan menampilkan jumlah *Order* yang sukses
Data ditampilkan menggunakan *bar chart*	Data ditampilkan dalam bentuk tabel
Pada *chart* menampilkan jumlah *Revenue* dan *Income*	Tabel berisi tentang semua produk yang tersedia beserta jumlah total rupiah yang laku terjual

Hasil perbaikan atau revisi dari evaluasi *prototype* pada menu *Analytics Tab Profits* dapat dilihat pada Gambar 3.22.

Gambar 3.22 Hasil Evaluasi *Prototype* Menu *Analytics Tab Profits*

c. Halaman Analytics Tab Items

Prototype yang disampaikan pada saat evaluasi dengan PT. AVO, data yang ditampilkan sebanyak jumlah hari, bulan, tahun yang dipilih oleh pengguna pada saat melakukan *filter*, data ditampilkan dalam bentuk *pie chart* sehingga *chart* yang ditampilkan bisa lebih dari 1. Namun setelah evaluasi, data yang ditampilkan hanya hasil akumulasi dari semua jumlah hari, bulan, tahun yang di *filter* oleh pengguna, sehingga *pie chart* yang ditampilkan hanya 1. Hasil revisi desain dari evaluasi *prototype* pada menu *Analytics Tab Items* dapat dilihat pada Gambar 3.23.



Gambar 3.23 Hasil Evaluasi *Prototype* Menu *Analytics Tab Items*

3.2.2 Iterasi 2

Pada iterasi 2, penulis memperlihatkan hasil revisi desain pada iterasi 1 kepada *customer* sekaligus mempresentasikan hasil desain pada tahap dua. Fitur yang menjadi objek iterasi 2 adalah manajemen pesanan dan pembelian produk. Selain itu pada iterasi ini juga dibangun fitur pelengkap berupa halaman *Points* dan halaman *Settings.*

1. Membuat Desain Sederhana

Terdapat tujuh desain sederhana yang dibuat pada iterasi 2, berikut penjelasannya:

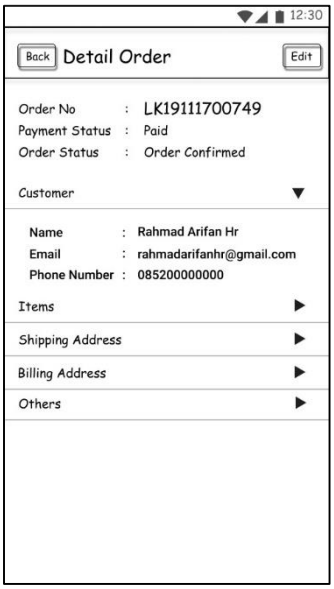
a. Halaman *Orders*

Halaman manajemen pesanan memiliki *action bar* dengan label *Orders*, memiliki tombol menu pada sisi kiri, dan tombol filter pada sisi kanan. Lalu pada konten utamanya, terdapat *input text* dengan tombol *search*, dan terdapat *list – list* pesanan dari *customer* yang masuk ke distributor. Setiap *item* pada *list* memiliki warna *background* tertentu sebagai penanda bahwa pesanan tersebut adalah pesanan baru dan belum dilihat oleh distributor atau pengguna. Setiap *item* menampilkan sekilas informasi terkait pesanan, di antaranya menampilkan tanggal dan waktu pesanan, *order number*, nama pemesan, e-mail pemesan, status pembayaran, dan status pesanan. Lihat Gambar 3.24 untuk lebih detail.

Gambar 3.24 Mockup Halaman *Orders*

b. Halaman *Detail Order*

Halaman detail pesanan memiliki *action bar* dengan label *Detail Order*, memiliki tombol *back* pada sisi kiri. Pada sisi kanan *action bar* terdapat tombol *edit*, namun pada penerapannya nanti tombol ini dibuat bersifat *optional*, dimana tombol ini hanya akan muncul jika status pembayaran telah dibayar dan pesanan belum dikirim. Lalu pada konten utama yang ditampilkan terdapat *order number*, status pembayaran, status pesanan, dan *tab* yang berisi tentang detail *customer, items, shipping address, billing address,* dan *others.* Lihat Gambar 3.25.



Gambar 3.25 Mockup Halaman *Detail Orders*

c. Halaman Ubah Status Pesanan

Halaman ubah status pesanan memiliki tampilan yang sama dengan halaman *Detail Orders,* akan tetapi pada konten utamanya terdapat sebuah *dialog* yang berisi tombol *Order Shipped* dan tombol *Delivery Delayed.* Lihat Gambar 3.26.

Gambar 3.26 Mockup Halaman Ubah *Order Status*

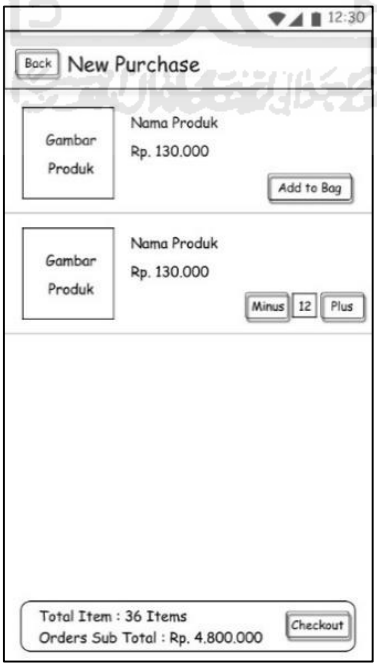
d. Halaman *Purchases*

Halaman pembelian produk memiliki *action bar* dengan label *Purchases*, memiliki tombol menu pada sisi kiri, tombol *add purchase* pada sisi kanan, dan tombol *filter* pada sebelah kanan *add purchase.* Lalu pada konten utamanya, terdapat *input text* dengan tombol *search*, dan terdapat *list – list* pembelian produk yang dilakukan oleh distributor atau pengguna. Setiap *item* pada *list* memiliki warna *background* tertentu sebagai penanda bahwa pembelian produk tersebut adalah pembelian yang diperbaharui dan belum dilihat oleh distributor atau pengguna. Setiap *item* menampilkan sekilas informasi terkait pembelian, di antaranya menampilkan tanggal dan waktu pesanan, *purchase number*, total pembelian, status pembayaran, dan status pesanan. Lihat Gambar 3.27 untuk lebih detail.

Gambar 3.27 Mockup Halaman *Purchases*

e. Halaman *New Purchase*

Halaman pembelian produk baru memiliki *action bar* dengan label *New Purchases*, memiliki tombol *back* pada sisi kiri. Pada konten utamanya menampilkan list produk Lacoco yang dijual oleh PT. AVO. Pada setiap *item list* terdapat gambar produk, nama produk, harga, dan tombol Add to Bag. Tombol Add to Bag bersifat fleksibel, jika tombol tersebut dipencet maka akan menampilkan tombol *minus*, jumlah produk yang akan dibeli, dan *plus.* Pada sisi bagian bawah halaman terdapat *dialog* yang menampilkan *total item*, *order sub total*, dan tombol *checkout.* Lihat Gambar 3.28.



Gambar 3.28 Mockup Halaman *New Purchase*

f. Halaman *List Form New Purchase*

Halaman list form pembelian produk baru *action bar* dengan label *Your Purchases*, memiliki tombol *back* pada sisi kiri. Pada konten utamanya menampilkan *list form* yang harus dilengkapi oleh pengguna. *Form* tersebut antara lain *My Bag, Contact Details, Shipping Address, Billing Address,* dan *Shipping Method.* Pada setiap *form* memiliki gambar pada sisi kiri sebagai penanda bahwa *form* tersebut telah selesai atau belum. Lalu pada bagian bawah *list form* terdapat *input text promotional code* dan tombol edit, *order sub total, discount, shipping, total,* dan *points earned.* Lihat Gambar 3.29.

Gambar 3.29 Mockup Halaman *List Form New Purchase*

g. Halaman *Points*

Halaman *points* memiliki *action bar* dengan label *Orders*, memiliki tombol menu pada sisi kiri. Lalu pada konten utamanya, menampilkan total *points* dari pengguna dan terdapat *list* untuk detail *points* yang didapat oleh pengguna. Setiap *item list* menampilkan tanggal dan waktu mendapatkan *points,* informasi alasan mendapatkan *points,* dan total *points*. Lihat Gambar 3.30 untuk lebih detail.

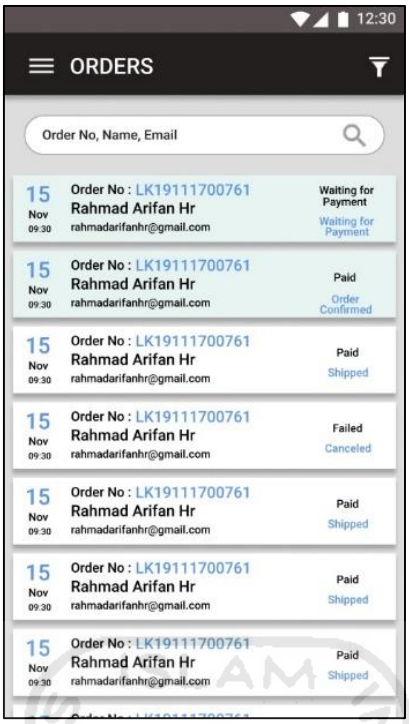
Gambar 3.30 Mockup Halaman *Points*

2. Membuat Prototype

Berdasarkan hasil desain sederhana tersebut, kemudian dibuat *prototype* dimana pada tahapan ini sudah diberi pewarnaan dan penempatan ikon. *Prototype* untuk halaman *Orders*, halaman *Purchases,* halaman *Points* dan halaman *Settings* bisa dilihat dari Gambar 3.31 sampai Gambar 3.47.

a. Halaman *Orders*

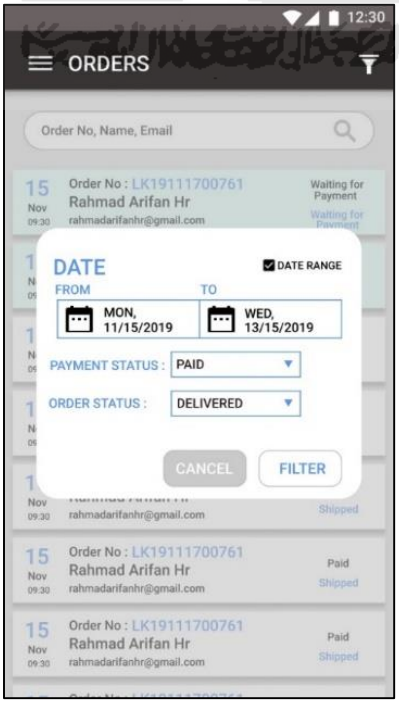
Halaman *Orders* pada Gambar 3.31 setiap *item list* yang merupakan pesanan baru dan belum dilihat oleh pengguna maka diberi warna *background* biru muda. *Text* pada tanggal dan nomor pesanan *font* akan diberi warna biru. Lalu untuk mempermudah pengguna dalam membedakan status pembayaran dan status pesanan maka *font* status pembayaran akan diberi warna hitam dan *font* status pesanan akan diberi warna biru.



Gambar 3.31 *Prototype* Halaman *Orders*

b. Halaman *Filter Orders*

Untuk melakukan *filter* pada halaman *Orders* terdapat *date* yang bisa menampilkan *single date* atau *date range.* Lalu terdapat status pembayaran dan status pesanan yang dapat dipilih dalam bentuk *spinner.* Lihat Gambar 3.32 untuk lebih detail.



Gambar 3.32 *Prototype* Halaman *Filter Orders*

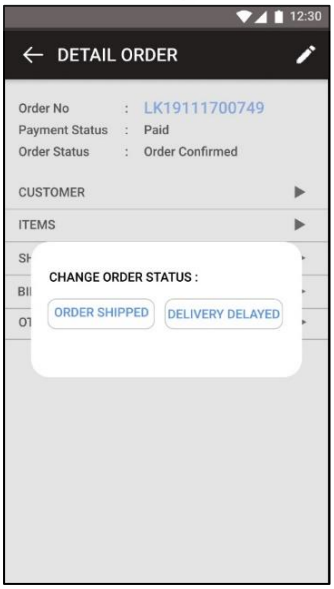
c. Halaman *Detail Order*

Halaman *Detail Order* pada Gambar 3.33 terdapat *order number* dengan font berwarna biru. Pada halaman detail *order* terdapat beberapa *section* yang bisa di *collapse* atau *expand* antara lain informasi *customer* yang membeli, item yang dibeli, alamat pengiriman, alamat penagihan, dan informasi lainnya.

Gambar 3.33 *Prototype* Halaman *Detail Order*

d. Halaman Ubah *Order Status*

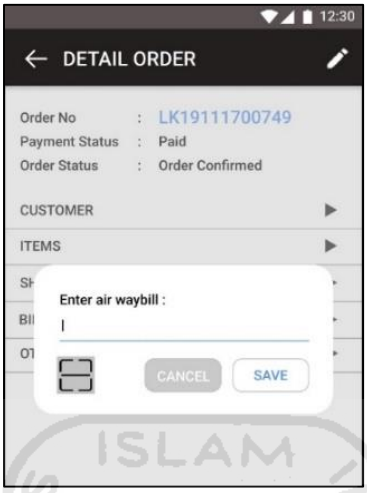
Halaman Ubah *Order Status* pada Gambar 3.34 terdapat dua *Button* yaitu *Order Shipped* dan *Delivery Delayed*. *Button* tersebut memiliki warna *background* transparan dan *font* berwarna biru.



Gambar 3.34 *Prototype* Halaman Ubah *Order Status*

e. Halaman *Input Waybill*

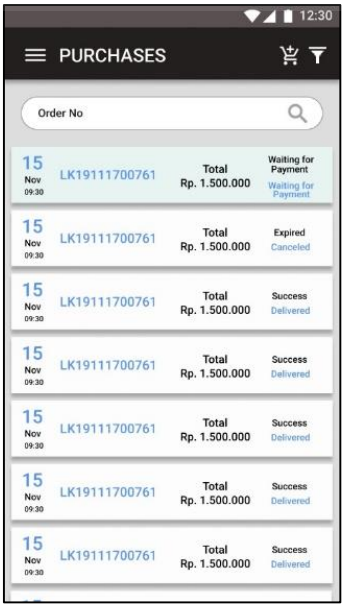
Halaman *Input Waybill* pada Gambar 3.35 terdapat *input text* nomor resi. Terdapat *Button* dengan ikon *scanner* yang berfungsi untuk memindai nomor resi dalam bentuk *barcode.* Lalu terdapat dua *Button* yaitu *Cancel* dan *Save.*



Gambar 3.35 *Prototype* Halaman *Input Waybill*

f. Halaman *Purchases*

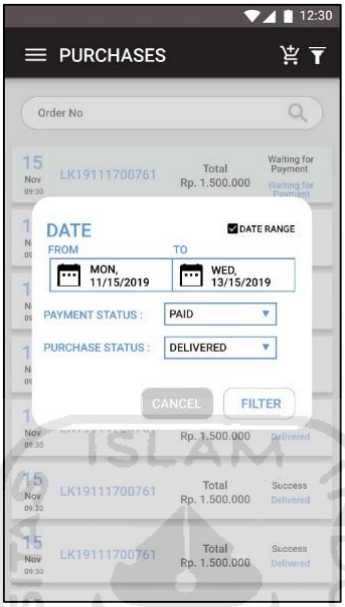
Halaman *Purchases* pada Gambar 3.36 setiap *item list* yang merupakan pembelian produk baru yang diperbaharui dan belum dilihat oleh pengguna maka akan diberi warna *background* biru muda. *Text* pada tanggal dan nomor pesanan *font* akan diberi warna biru. Lalu untuk mempermudah pengguna dalam membedakan status pembayaran dan status pembelian maka *font* status pembayaran akan diberi warna hitam dan *font* status pembelian akan diberi warna biru.



Gambar 3.36 *Prototype* Halaman *Purchases*

g. Halaman *Filter Purchases*

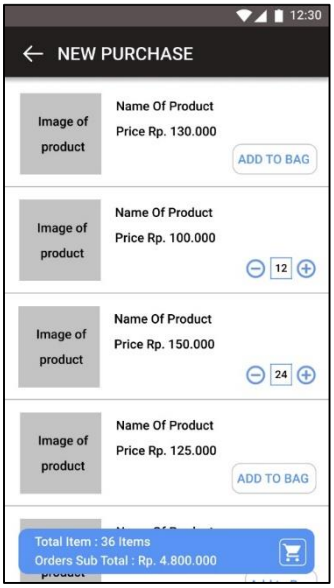
Untuk melakukan *filter* pada halaman *Purchases* terdapat *date* yang bisa menampilkan *single date* atau *date range*. Lalu terdapat status pembayaran dan status pembelian yang dapat dipilih dalam bentuk *spinner*. Lihat Gambar 3.37 untuk lebih detail.



Gambar 3.37 *Prototype* Halaman *Filter Purchases*

h. Halaman *New Purchase*

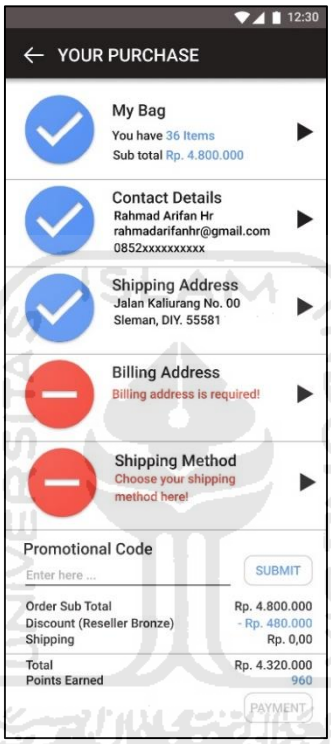
Button "*Add to Bag"* diberi *background* warna transparan dan *font* warna biru. *Button* tambah dan kurang akan menggunakan ikon berwarna biru. Lalu pada dialog pada sisi bawah akan diberi *background* warna biru dan *font* warna putih. Lihat Gambar 3.38.



Gambar 3.38 *Prototype* Halaman *New Purchase*

i. Halaman *List Form New Purchase*

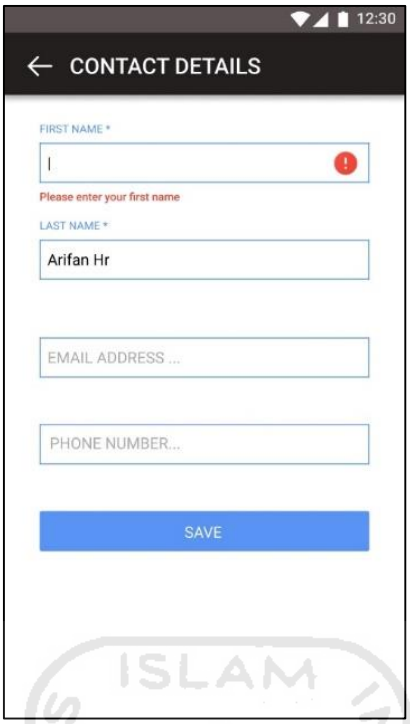
Pada Gambar 3.39 setiap *item form* memiliki sebuah gambar sebagai penanda, apabila *form* sudah selesai maka akan diberi gambar centang dengan *background* warna biru, dan apabila *form* belum selesai maka akan diberi gambar setrip dengan *background* warna merah. Jika semua *form* belum diselesaikan oleh pengguna, maka *Button "Payment"* akan di*disable,* lalu diberi *font* dan *outline* berwarna abu – abu. Jika semua *form* sudah lengkap maka *Button* tersebut akan *enable,* lalu diberi *font* dan *outline* berwarna biru.



Gambar 3.39 *Prototype* Halaman *List Form New Purchase*

j. Halaman *Form Contact Details New Purchase*

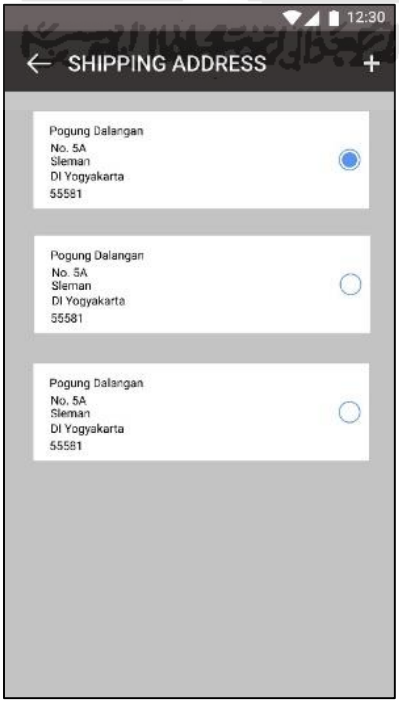
Halaman *Form Contact Details* pada Gambar 3.40 memiliki beberapa *input text* antara lain *first name, last name, email address,* dan *phone number.* Setiap *input text* tersebut memiliki *error handling* jika data yang dimasukkan tidak sesuai, pesan *error* akan ditampilkan di bawah *input text* dan memunculkan ikon *error* pada sisi kanan di dalam *input text.* Lalu terdapat *Button "Save"* pada bagian bawah *form,* dengan *background* warna biru dan *font* warna putih.



Gambar 3.40 *Prototype* Halaman *Form Contact Details New Purchase*

k. Halaman *Form Shipping Address New Purchase*

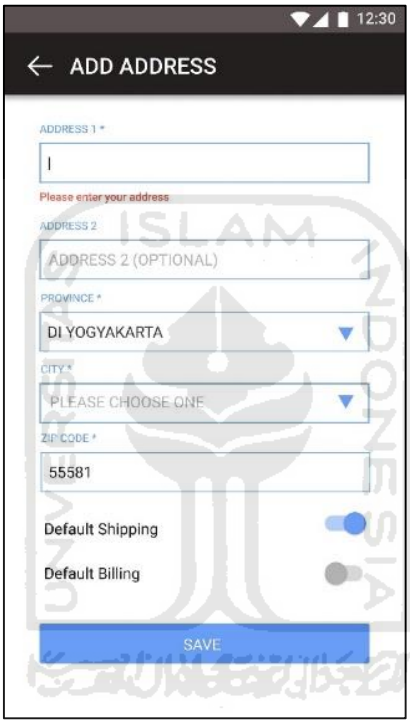
Halaman *Form Shipping Address* pada Gambar 3.41 menunjukkan list alamat yang telah disimpan oleh *user*. Sistem akan memilih secara otomatis jika di-*set* sebagai alamat *default shipping* ketika menambahkan alamat, atau *user* bisa memilih salah satu dari alamat tersebut.



Gambar 3.41 *Prototype* Halaman *Form Shipping Address New Purchase*

l. Halaman *Form Add New Address*

Halaman *Form Add New Address* pada Gambar 3.42 memiliki beberapa *input text* antara lain *address 1, address 2,* dan *zip code.* Setiap *input text* tersebut memiliki *error handling* jika data yang dimasukkan tidak sesuai, pesan *error* akan ditampilkan di bawah *input.* Lalu terdapat pilihan *list province* dan *list city* dalam bentuk *spinner*. Kemudian terdapat *Button "Save"* pada bagian bawah *form*, dengan *background* warna biru dan *font* warna putih. Namun sebelum *Button "Save"*, terdapat sebuah *checkbox "Add to Address Book"* yang berfungsi untuk membuat pilihan apakah data *Shipping Address* tersebut mau disimpan atau tidak.

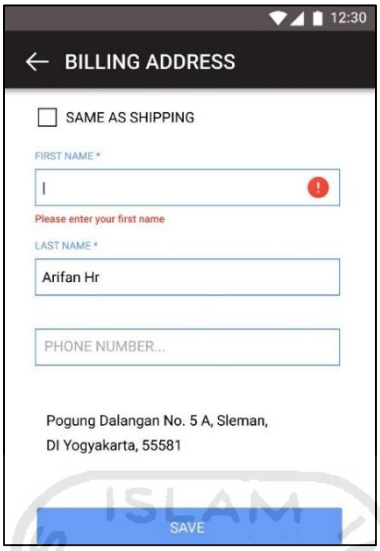


Gambar 3.42 *Prototype* Halaman *Form Add New Address*

m. Halaman *Form Billing Address New Purchase*

Halaman *Form Billing Address* pada Gambar 3.43 terdapat *checkbox "Same as Shipping"* yang berfungsi untuk membuat pilihan apakah data *Billing Address* sama dengan data pembeli pada *Contact Details* dan *Shipping Address.* Lalu *form* ini terdapat beberapa *input text* antara lain *first name, last name, phone number.* Setiap *input text* tersebut memiliki *error handling* jika data yang dimasukkan tidak sesuai, pesan *error* akan ditampilkan di bawah *input text.* Kemudian terdapat *Button "Save"* pada bagian bawah *form*, dengan *background* warna biru dan *font* warna putih. Namun sebelum *Button "Save"*, terdapat *containter* yang menampilkan alamat dari *Billing Address* yang dipilih. Ketika *container* tersebut di-klik maka

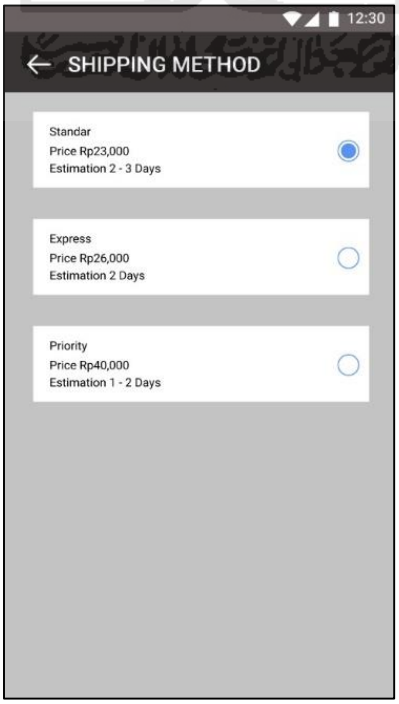
akan memunculkan list alamat yang telah diinputkan oleh pengguna, sama seperti halnya pada *Shipping Address.*



Gambar 3.43 *Prototype* Halaman *Form Billing Address New Purchase*

n. Halaman *Form Shipping Method New Purchase*

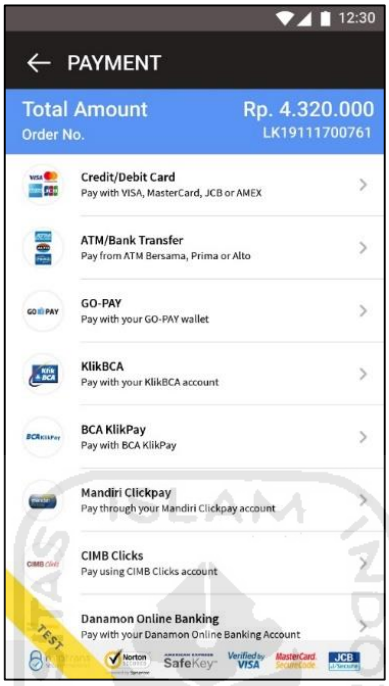
Halaman *Form Shipping Method* pada Gambar 3.44 terdapat *container* yang menampilkan *list* metode pengiriman beserta biaya dan estimasinyanya. Lalu disisi kanannya terdapat *Radio Button* yang menunjukkan apakah metode itu yang dipilih atau bukan.



Gambar 3.44 *Prototype* Halaman *Form Shipping Method New Purchase*

o. Halaman *Payment New Purchase*

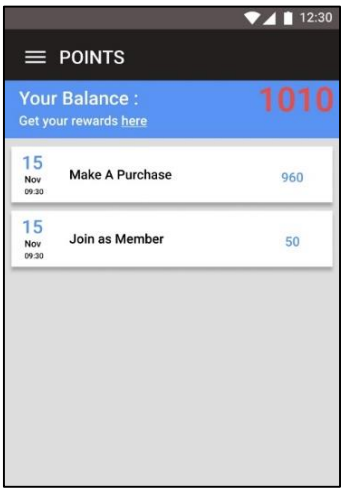
User Interface pada halaman *Payment* merupakan tampilan bawaan dari Midtrans *Mobile* SDK dengan *bar* berwarna biru, lihat Gambar 3.45.



Gambar 3.45 *Prototype* Halaman *Payment New Purchase*

p. Halaman *Points*

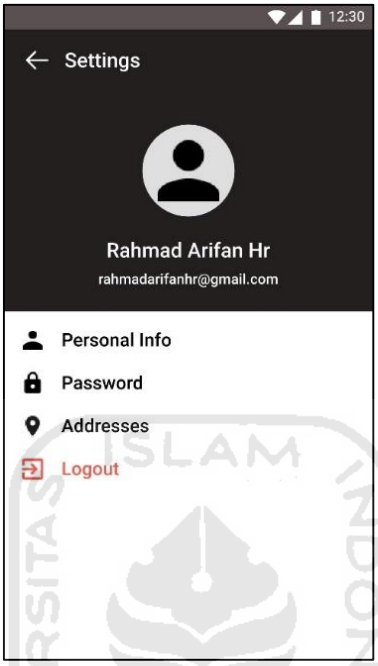
Halaman *Points* pada Gambar 3.46 terdapat *header* dengan *background* warna biru dan *font* poin dengan warna merah. Lalu untuk setiap *item* pada *list points*, *font* tanggal dan poin diberi warna biru.



Gambar 3.46 *Prototype* Halaman *Points*

q. Halaman *Settings*

Halaman *Settings* pada Gambar 3.47 terdapat info profil berisi gambar profil, nama dan email dengan *background* warna hitam dan *font* warna putih. Lalu dibawah info profil, terdapat empat *Button* dengan ikon dan teks yaitu *Personal Info, Password, Addreses,* dan *Logout.*



Gambar 3.47 *Prototype* Halaman *Settings*

3. Evaluasi dan Revisi *Prototype*

Evaluasi *prototype* iterasi 2 dilakukan pada 16 Maret 2020, di kantor PT. AVO. Terdapat beberapa poin yang disampaikan oleh PT. AVO pada evaluasi iterasi 2. Poin pertama yaitu terkait warna *font* pada status pesanan, status pembelian, dan status pembayaran yang diminta untuk diubah warnanya sesuai status pesanan dan status pembayaran.

Tabel 3.5 berikut menjelaskan status pesanan dan status pembelian beserta warna yang dipilih. Adapun Tabel 3.6 menjelaskan status pembayaran dan warna yang dipilih.

Tabel 3.5 Warna Status Pesanan dan Status Pembelian

Status	**Warna**
Waiting for Payment	Hitam
Order Confirmed	Hijau
Shipped / Delivered	Biru
Cancelled	Merah
Order Delayed	Kuning

Tabel 3.6 Warna Status Pembayaran

Status	**Warna**
Waiting for Payment	Hitam
Paid	Biru
Failed	Merah

Poin kedua, semua pemisah angka pada penulisan yang terkait mata uang perlu diubah menjadi koma dari yang awalnya titik. Selain itu antara simbol Rp dengan angka setelahnya tidak perlu diberi spasi. Sebagai contoh Rp. 1.000.000 menjadi Rp1,000,000.

Iterasi hanya terjadi dua kali karena berdasarkan hasil evaluasi pada iterasi 2, *feedback* yang disampaikan oleh *customer* hanya terkait warna dan penulisan, bukan terkait tata letak maupun proses bisnis dari alur aplikasi. Sehingga hasil revisi iterasi 2 tidak perlu dievaluasi kembali oleh pemilik aplikasi. Oleh karena itu iterasi pada tahap desain penelitian ini berhenti, dan siap untuk melakukan tahap selanjutnya yaitu pengembangan sistem.

3.3 Evolusi *Prototyping*

Setelah melakukan tahap desain, dapat disimpulkan bahwa desain mengalami evolusi yang terjadi dalam dua iterasi. Berikut kesimpulan hasi evolusi desain.

1. Evolusi dari Website Lacoco.co.id ke Aplikasi Lacoco *Mobile*

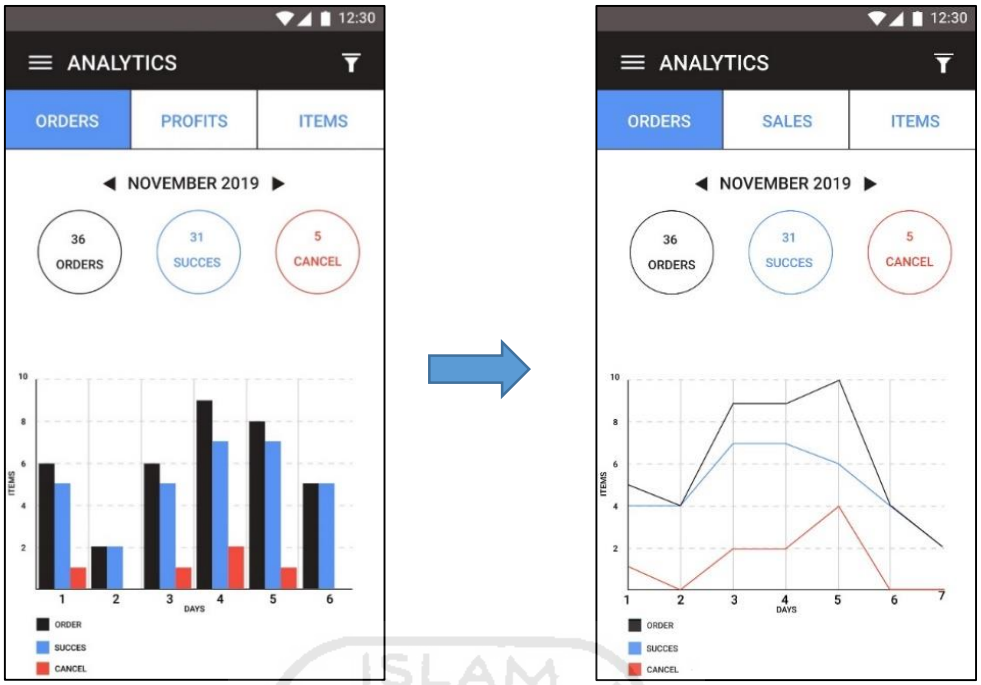
Berdasarkan observasi pada tahap analis penulis mempelajari website lacoco.co.id yang kemudian dari hasil observasi penulis kemudian membuat desain *prototype* untuk aplikasi Lacoco *mobile.* Maka disinilah terjadi evolusi. Evolusi terjadi dari sistem yang sudah ada lacoco.co.id kemudian dibawa ke aplikasi Lacoco *mobile.* Evolusi ini mencakup terkait pewarnaan yaitu didominasi oleh warna hitam dan putih, lalu *font* yang digunakan adalah Glacial Indifference.

2. Evolusi dari Iterasi 1 ke Iterasi 2

Setelah melakukan iterasi 1 terdapat beberapa *feedback* dari *customer*, sehingga desain *prototype* yang telah dibangun pada tahap pertama mengalami evolusi. Evolusi mencakup pada tiga halaman yaitu.

a. Halaman *Analytics Tab Orders*

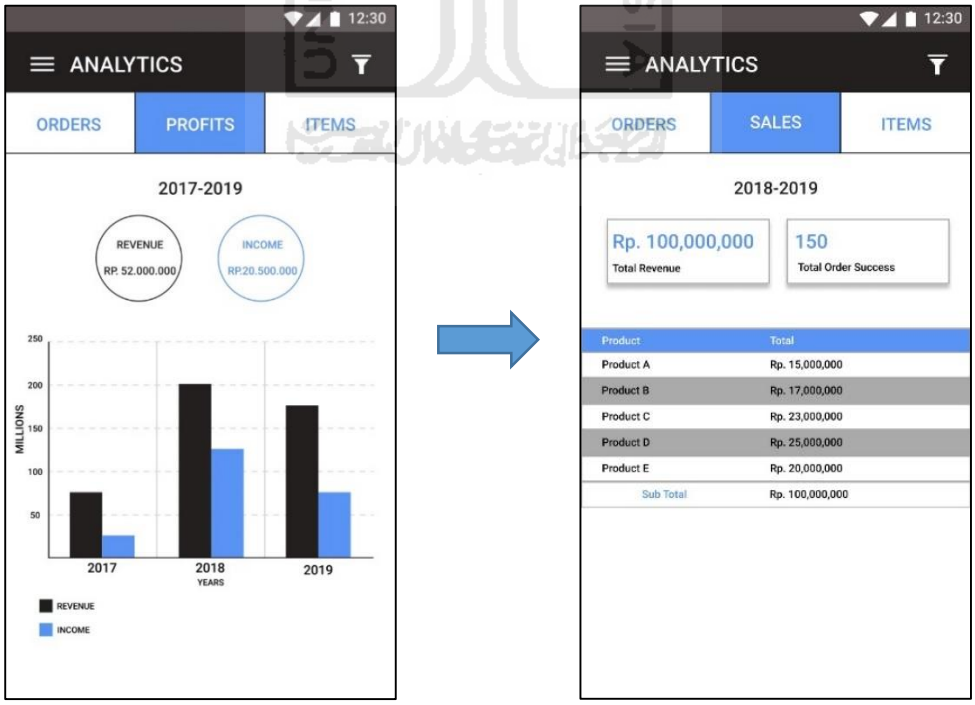
Pada Gambar 3.48 menunjukkan hasil evolusi dari halaman *Analytics Tab Orders*. Untuk keterangan lebih detail terkait evolusi telah dijelaskan pada tahap Evaluasi Prototype iterasi 1.



Gambar 3.48 Evolusi Halaman *Analytics Tab Orders*

b. Halaman *Analytics Tab Profits*

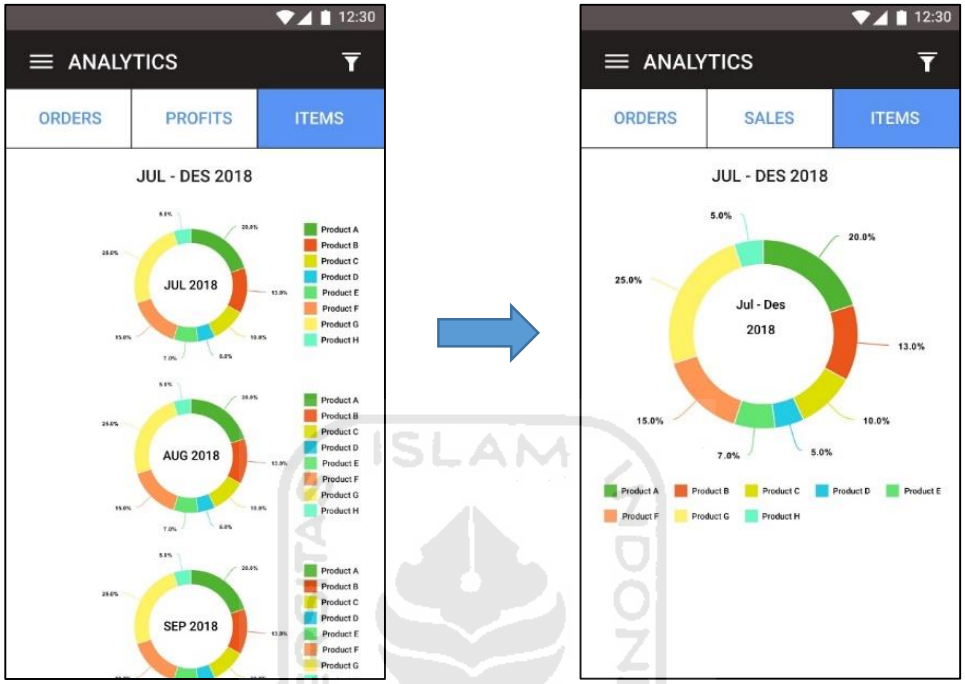
Pada Gambar 3.49 menunjukkan hasil evolusi dari halaman *Analytics Tab Profits*. Untuk keterangan lebih detail terkait evolusi telah dijelaskan pada tahap Evaluasi Prototype iterasi 1.



Gambar 3.49 Evolusi Halaman *Analytics Tab Profits*

c. Halaman *Analytics Tab Items*

Pada Gambar 3.50 menunjukkan hasil evolusi dari halaman *Analytics Tab Items*. Untuk keterangan lebih detail terkait evolusi telah dijelaskan pada tahap Evaluasi Prototype iterasi 1.



Gambar 3.50 Evolusi Halaman *Analytics Tab Items*

3. Evolusi Iterasi 2

Evolusi desain pada iterasi 2 telah dijelaskan pada Evaluasi dan Revisi *Prototype* iterasi 2, yaitu hanya berkaitan dengan pewarnaan *font* dan penulisan mata uang.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Pengembangan Sistem

Setelah melakukan tahap analisis dan desain, tahapan selanjutnya dari metode *prototyping* adalah membangun sistem. Sistem dibangun berdasarkan *prototype* yang telah dievaluasi pada dua iterasi pada tahap desain sebelumnya. Sebagaimana hasil analisis kebutuhan pada subbab 3.1, aplikasi ini merupakan aplikasi bergerak (*mobile)* yang berbasiskan Android maka aplikasi ini dibangun menggunakan aplikasi Android Studio yang menggunakan bahasa Java. Serta beberapa aplikasi yang digunakan untuk membangun API, di antaranya XAMPP sebagai web server, Sublime sebagai teks editor, Postman sebagai pengujian API. Dalam membangun sistem pada penelitian kali ini secara umum dilakukan dengan enam langkah, yaitu:

1. Membuat API
2. Menambahkan route API
3. Melakukan request melalui Postman
4. Membuat model dari response di Android Project
5. Membuat UI di projek Android
6. Membuat kode untuk melakukan *request* API dari aplikasi Android

Keenam langkah diatas secara umum dilakukan pada halaman – halaman yang telah disebutkan pada Tabel 3.2. Namun tidak semua halaman dalam tahapan membangun sistem menggunakan enam langkah tersebut seperti halaman *Splashscreen* dan halaman Menu, ini dikarenakan halaman tersebut tidak memerlukan hubungan atau bertukar data dengan website lacoco.co.id.

Adapun teknis implementasinya didalam laporan tugas akhir ini penulis hanya menjabarkan salah satu contoh bagaimana teknis pengimpelementasian sistem. Contoh yang dituliskan pada laporan ini adalah pengimplementasian halaman *orders*. Langkah – langkah untuk implementasi halaman *order*s akan dijelaskan dibawah.

1. Membuat API *Get List Orders*

Sebelum menampilkan data list *orders* ke aplikasi, perlu adanya pengambilan data dari *database* atau server. Untuk melakukan pengambilan data tersebut maka perlu membuat API. Sebagaimana penelitian yang telah dilakukan sebelumnya oleh (Huda, 2018), website

lacoco.co.id dibangun menggunakan *framework* Laravel maka kode untuk membuat API dibangun menggunakan bahasa PHP. Lihat Gambar 4.1 untuk kode pada fungsi *get list orders.*

```
public function customer(Request $request)
    {
        $user = $request->user();
        $customer = $user->customer;
        $orders = [];
        if($customer) {
            $query  =  Order::hasPayment()->where('customer_id',  $customer->id)-
>with('payment');
            $query->orderBy('created_at', 'desc');
            $orders = $query->paginate(10);
        }
        $return = [
            'status' => true,
            'message' => 'Success',
            'data' => $orders
        ];
        return response()->json($return, 200);
    }
```

Gambar 4.1 Fungsi *Get List Orders*

2. Menambahkan route API

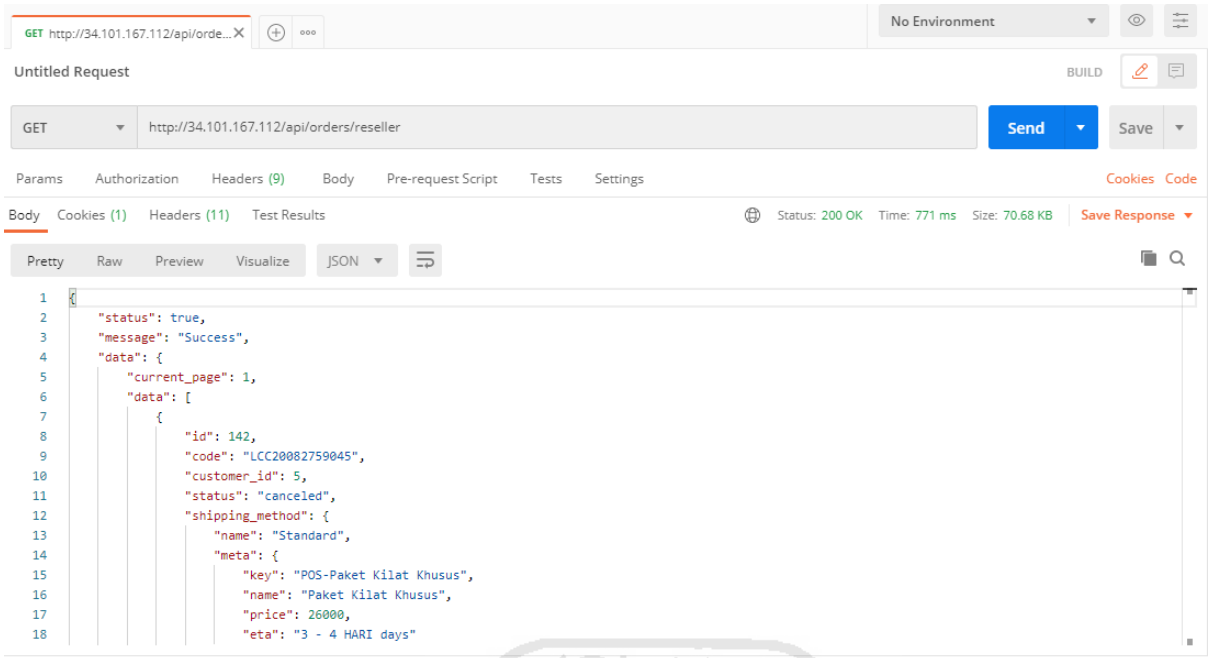
Setelah membuat kode untuk mengambil data, selanjutnya perlu melakukan *route* atau membuat jalur agar bisa diakses menggunakan URL. Lihat Gambar 4.2 untuk *route* API *get list orders.*

```
Route::group(['prefix' => 'orders', 'middleware' => 'auth:api'], function(){
    Route::get('/customer', 'Api\OrderController@customer');
});
```

Gambar 4.2 *Route Get List Orders*

3. Melakukan request melalui Postman

Setelah menambahkan *route*, selanjutnya adalah melakukan uji coba API tersebut melalui aplikasi Postman. Uji coba ini bertujuan untuk memastikan apakah *response* yang diharapkan sesuai dengan data yang ada diserver dan *response* tersebut sesuai dengan yang dibutuhkan oleh aplikasi *mobile* berdasarkan desain *prototype* yang telah dibuat sebelumnya. Lihat Gambar 4.3 untuk *screenshot response* API dari aplikasi Postman dan Gambar 4.4 untuk detail dari isi *response* yang diberikan.



Gambar 4.3 *Response* API *Get List Orders* melalui Postman


```
{
    "status": true,
    "message": "Success",
    "data": {
        "current_page": 1,
        "data": [
            {
                "id": 142,
                "code": "LCC20082759045",
                "customer_id": 5,
                "status": "canceled",
                "shipping_method": {
                    "name": "Standard",
                    "meta": {
                        "key": "POS-Paket Kilat Khusus",
                        "name": "Paket Kilat Khusus",
                        "price": 26000,
                        "eta": "3 - 4 HARI days"
                    }
                },
                "waybill": null,
                "redeemed_points": 0,
                "discount_id": null,
                "is_read": true,
                "created_at": "2020-08-27 09:16:50",
                "updated_at": "2020-08-28 09:18:26",
                "reseller_id": 11,
                "sub_total": 195000,
                "total": 221000,
                "discount_value": 0,
                "points_earned": 90,
                "total_weight": 200,
                "payment": {
                    "id": 78,
                    "status": "denied",
                    "meta": {
                        "meta": {
                            "va_numbers": [
```

```
                                {
                                    "va_number": "98931091957",
                                    "bank": "bca"
                                }
                            ],
                            "transaction_time": "2020-08-27 09:18:21",
                            "transaction_status": "expire",
                            "payment_type": "bank_transfer"
                        }
                    }
                },
                "variants": [
                    {
                        "id": 1,
                        "product_id": 1,
                        "fullname": "Viola - SteelBlue",
                        "validprice": "195000",
                    }
                ],
                "discount": null,
                "customer": {
                    "id": 5,
                    "user_id": 9,
                    "first_name": "Rahmad",
                    "last_name": "Arifan Hr",
                    "email": "rahmadarifanhr3@gmail.com",
                    "phone": "082328329262",
            },
        ],
        "from": 1,
        "last_page": 1,
        "next_page_url": null,
        "path": "http://34.101.167.112/api/orders/reseller",
        "per_page": 10,
        "prev_page_url": null,
        "to": 9,
        "total": 9
    }
}
```


Gambar 4.4 *Response* API *Get List Orders* dari Postman

4. Membuat Model dari Response di Android Project

Selanjutnya membuat model di projek Android. Gambar 4.5 merupakan salah satu contoh dari Model dari *response* API *Get List Orders*

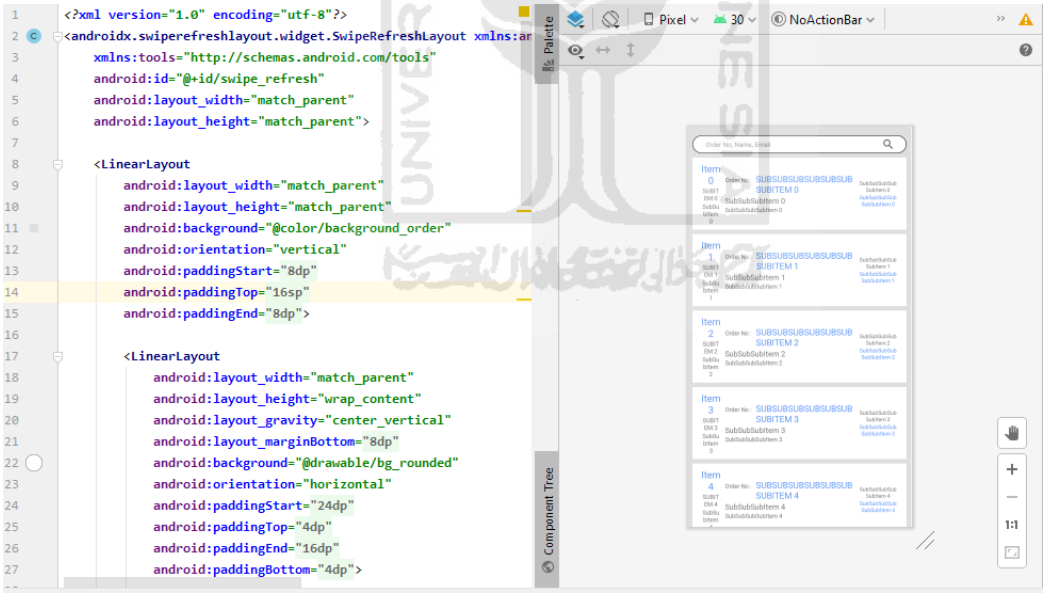
```
public class Main {

    @SerializedName("status")
    @Expose
    private Boolean status;
    @SerializedName("message")
    @Expose
    private String message;
    @SerializedName("data")
    @Expose
    private Data data;

    public Boolean getStatus() {
        return status;
    }

    public void setStatus(Boolean status) {
```

```
        this.status = status;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Data getData() {
        return data;
    }

    public void setData(Data data) {
        this.data = data;
    }
}
```

Gambar 4.5 Salah Satu Model dari API *Get List Orders*

5. Membuat UI di projek Android

Selanjutnya membuat tampilan atau *User Interface* (UI) pada projek Android sesuai dengan desain yang telah dibuat dan dievaluasi pada subbab 3.2.1 dan 3.2.2. Lihat Gambar 4.6 untuk gambaran membuat UI pada Android Studio.



Gambar 4.6 Membuat UI di Android Studio

6. Membuat kode untuk melakukan *request* API dari aplikasi Android

Untuk melakukan *request* API pada Android, perlu membuat dua kelas utama yaitu API *Client* dan API *service*. Kelas API *Client* adalah kelas yang berfungsi untuk membuat *Base* URL dan *header* sekaligus membuat objek retrofit. Retrofit adalah *library* Android yang

digunakan untuk melakukan pertukaran data antara Android dengan server melalui API. Lihat Gambar 4.7 untuk kode pada kelas API *Client.*

```
public class LacocoAPIClient {
    public static Retrofit getClient(String token) {

        HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

        OkHttpClient client = new OkHttpClient.Builder()
                .addInterceptor(interceptor)
                .addInterceptor(chain -> {
                    Request request = chain.request();
                    request = request.newBuilder().addHeader("Authorization",
"Bearer " + token).build();
                    return chain.proceed(request);
                })
                .build();

        return new Retrofit.Builder()
                .baseUrl(BuildConfig.BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .client(client)
                .build();
    }
}
```

Gambar 4.7 Retrofit API *Client*

Sedangkan kelas API *Service* berisi *method* dan *function* yang akan digunakan untuk mengakses data dari server atau mengirim data ke server. Lihat Gambar 4.8 untuk kode pada kelas API *Service.*

```
public interface LacocoAPIService {

    @GET("/api/orders/reseller")
    Call<OrderResponse.Main> getOrderReseller(
            @Query("per_page") int perPage,
            @Query("page") int page,
            @Query("search") String search,
            @Query("status_order") String statusOrder,
            @Query("status_bayar") String statusBayar,
            @Query("date_format") String dateFormat,
            @Query("date_from") String dateFrom,
            @Query("date_to") String dateTo,
            @Query("date") String date);
}
```

Gambar 4.8 Retrofit API *Service*

Selanjutnya melakuan *initiate* objek dari kelas *service* yang telah dibuat didalam kelas Fragment atau *Activity* dari fitur list *Orders*. Lalu memanggil *method* yang dibutuhkan, dalam hal ini memanggil *method* getOrderReseller yaitu untuk mengambil list *orders*. Lihat Gambar 4.9 untuk kode lebih lengkap.

```
public class OrderFragment extends Fragment {

    private LacocoAPIService lacocoAPIService;
    private List<OrderResponse.Datum> orders;
    private int per_page;
    private OrderAdapter orderAdapter;


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        initiate();
        getData();
        return inflater.inflate(R.layout.fragment_order, container, false);

    }


private void initiate() {
    per_page = 10;
    lacocoAPIService = LacocoAPIClient
                .getClient(token)
                .create(LacocoAPIService.class);

    orders = new ArrayList<>();
    orderAdapter = new OrderAdapter(getContext(), orders);
}

private void getData() {
        lacocoAPIService.getOrderReseller(
                per_page,
                page,
                null,
                null,
                null,
                null,
                null,
                null,
                null).enqueue(new Callback<OrderResponse.Main>() {
            @Override
            public void onResponse(@NotNull Call<OrderResponse.Main> call,
@NotNull Response<OrderResponse.Main> response) {
                if (response.isSuccessful()) {
                    OrderResponse.Main main = response.body();
                    if (main != null && main.getMessage().equals("Success") {
                        OrderResponse.Data data = main.getData();
                        if (data != null) {
                            orders.addAll(data.getData());
                            orderAdapter.notifyDataSetChanged();
                        }
                    }
                }
            }

            @Override
            public void onFailure(@NotNull Call<OrderResponse.Main> call,
@NotNull Throwable t) {
                hideProgressBar();
                Toast.makeText(mainActivity, R.string.server_error,
Toast.LENGTH_SHORT).show();
            }
        });
}
```

Gambar 4.9 Kelas Fragment untuk Halaman List *Orders*

Selanjutnya perlu membuat kelas Adapter yang digunakan pada kelas Fragment sebelumnya. Kelas Adapter ini digunakan untuk *binding* data array dari *response* API. Lihat Gambar 4.10 untuk kode pada kelas Adapter.

```
public class OrderAdapter extends RecyclerView.Adapter<OrderAdapter.ViewHolder> {
    private List<OrderResponse.Datum> orders;
    Context context;
    public OrderAdapter(Context context, List<OrderResponse.Datum> orders) {
        this.context = context;
        this.orders = orders;
    }
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView tvDay, tvMonth, tvTime, tvOrderNo, tvName, tvEmail,
tvPaymentStatus, tvOrderStatus;
        LinearLayout llBackground;
        View container;
        ViewHolder(View view) {
            super(view);
            container = view;
            tvDay = view.findViewById(R.id.tv_day);
            tvMonth = view.findViewById(R.id.tv_month);
            tvTime = view.findViewById(R.id.tv_time);
            tvOrderNo = view.findViewById(R.id.tv_order_no);
            tvName = view.findViewById(R.id.tv_name);
            tvEmail = view.findViewById(R.id.tv_email);
            tvPaymentStatus = view.findViewById(R.id.tv_status_payment);
            tvOrderStatus = view.findViewById(R.id.tv_status_order);
            llBackground = view.findViewById(R.id.ll_background);
        }
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view =
LayoutInflater.from(context).inflate(R.layout.item_order_list, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
        OrderResponse.Datum order = orders.get(position);
        if (order.getRead()){

viewHolder.llBackground.setBackgroundColor(context.getColor(R.color.colorWhite));
        } else {

viewHolder.llBackground.setBackgroundColor(context.getColor(R.color.unread_order_
purchase));
        }
        viewHolder.tvDay.setText(order.getDay());
        viewHolder.tvMonth.setText(order.getMonth());
        viewHolder.tvTime.setText(order.getTime());
        viewHolder.tvOrderNo.setText(order.getCode());
        if (order.getCustomer() != null) {
            viewHolder.tvName.setText(order.getCustomer().getFullname());
            viewHolder.tvEmail.setText(order.getCustomer().getEmail());
        }
        if (order.getPayment() != null) {

viewHolder.tvPaymentStatus.setText(order.getPayment().getString(context));

viewHolder.tvPaymentStatus.setTextColor(order.getPayment().getColor(context));
        }
        viewHolder.tvOrderStatus.setText(order.getStatusString(context));
```

```
            viewHolder.tvOrderStatus.setTextColor(order.getStatusColor(context));
        }
    }
```

Gambar 4.10 Kelas Adapter untuk List *Orders*

Setelah selesai menjalankan enam langkah tersebut, seharusnya halaman *orders* sudah bisa digunakan di aplikasi *mobile* namun perlu dilakukan pengujian dahulu, yang akan dijelaskan pada subbab selanjutnya. Adapun API yang dibangun pada penelitian kali dapat dilihat pada Tabel 4.1.

Tabel 4.1 API yang Dibangun

No.	Nama	Fungsi	URL	Method
1	Login	Authorization	http://34.101.167.112/api/login	POST
2	Register	Pendaftaran *reseller*	http://34.101.167.112/api/register	POST
3	Products	List produk dan stok	http://34.101.167.112/api/products	GET
4	Orders	List pesanan yang masuk	http://34.101.167.112/api/orders/reseller	GET
5	Purchases	List pesanan dari *reseller* ke AVO	http://34.101.167.112/api/orders/customer	GET
6	Order detail	Melihat order detail	http://34.101.167.112/api/orders/{id}	GET
7	Update order	Mengubah status order	http://34.101.167.112/api/orders/{id}	PUT
8	Points	List points	http://34.101.167.112/api/user/points	GET
9	Addresses	List alamat pengguna	http://34.101.167.112/api/user/address	GET
10	Add address	Menambahkan alamat pengguna	http://34.101.167.112/api/user/address	POST
11	Update address	Mengubah alamat pengguna	http://34.101.167.112/api/user/address/{id}	PUT
12	Delete address	Menghapus alamat pengguna	http://34.101.167.112/api/user/address/{id}	DELETE

No.	Nama	Fungsi	URL	Method
13	Customer Info	Informasi pengguna sebagai customer	http://34.101.167.112/api/user/customer_info	GET
14	Analytics Orders	Data analytics orders	http://34.101.167.112/api/stat/order	GET
15	Analytics Sales	Data analytics sales	http://34.101.167.112/api/stat/sale	GET
16	Analytics Items	Data analytics items	http://34.101.167.112/api/stat/product	GET
17	Shipping method	List metode pengiriman	http://34.101.167.112/api/checkout/shipping	POST
18	Coupon	Kupon yang tersedia di sever	http://34.101.167.112/api/checkout/coupon/{code}	GET
19	Checkout	Checkout pembelian produk	http://34.101.167.112/api/checkout	POST
20	Notifications	List notifikasi	http://34.101.167.112/api/user/notifications	GET
21	Notification Count	Total notifikasi	http://34.101.167.112/api/user/notifications/count	GET
22	Account	Informasi akun pengguna	http://34.101.167.112/api/user/account	GET
23	Change Password	Mengganti password	http://34.101.167.112/api/user/change_password	POST

Sedangkan hasil dari seluruh halaman yang telah diimplementasikan, akan dijelaskan sebagai berikut.

1. Halaman *Splashscreen*

Halaman *Splashscreen* berfungsi untuk menampilkan logo dari Lacoco selama 3 detik. Halaman ini akan muncul saat aplikasi Lacoco Mobile pertama kali dibuka. Lihat Gambar 4.11 untuk *screenshot* halaman *splashscreen.*

Gambar 4.11 Hasil Halaman *Splashscreen*

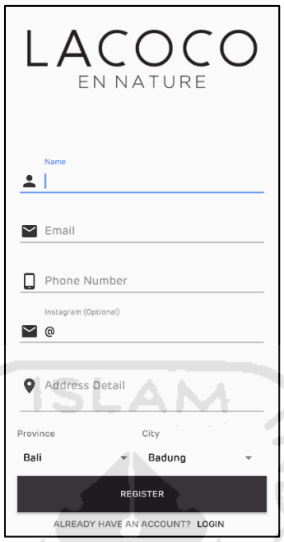
2. Halaman *Login*

Halaman *Login* berfungsi untuk melakukan otentikasi pengguna menggunakan *email* dan *password*. Lihat Gambar 4.12 untuk *screenshot* halaman *login.*

Gambar 4.12 Hasil Halaman *Login*

3. Halaman *Register*

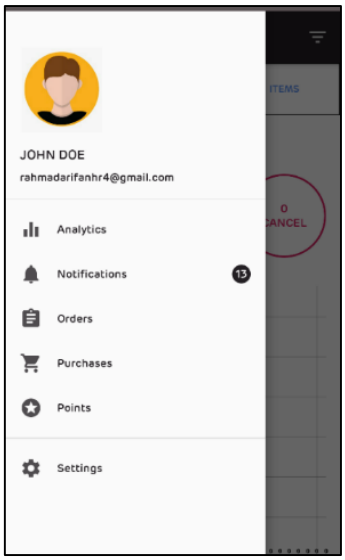
Halaman *Register* berfungsi untuk melakukan pendaftaran menjadi *user* atau menjadi seorang distributor atau *reseller*. Data yang dibutuhkan untuk mendaftar menjadi *user* adalah *name, email, phone number,* instagram, *address detail, province,* dan *city.* Lihat Gambar 4.13 untuk lebih detail.



Gambar 4.13 Hasil Halaman Register

4. Halaman Menu

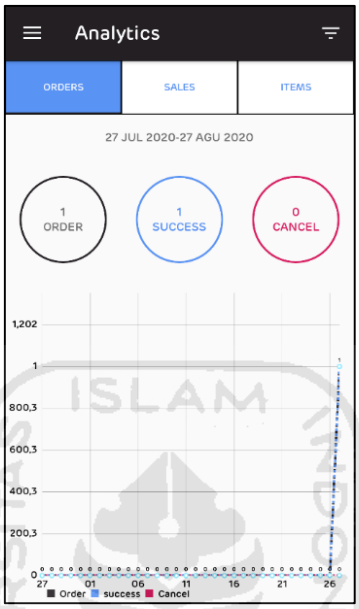
Halaman Menu berfungsi untuk menampilakn list menu yang tersedia di dalam aplikasi Lacoco Mobile. Menu yang tersedia antara lain *analytics, notifications, orders, purchases, points,* dan *settings.* Selain itu di halaman ini juga menampilkan nama dan *email* pengguna yang telah *login.* Lihat Gambar 4.14 untuk lebih detail.



Gambar 4.14 Hasil Halaman Menu

5. Halaman *Analytics Sub Orders*

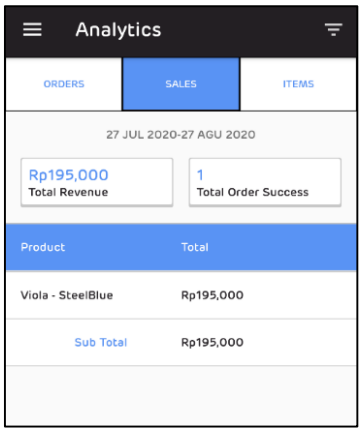
Halaman ini menampilkan analitik terkait pesanan yang masuk ke pengguna yang telah *login*. Diantaranya menampilkan total *orders* yang masuk, *orders* yang sukses, dan *orders* yang dibatalkan. Halaman ini juga menampilkan grafik dengan *chart* bergaris atau *line chart*. Lihat Gambar 4.15 untuk lebih detail.



Gambar 4.15 Hasil Halaman *Analytics Sub Orders*

6. Halaman *Analytics Sub Sales*

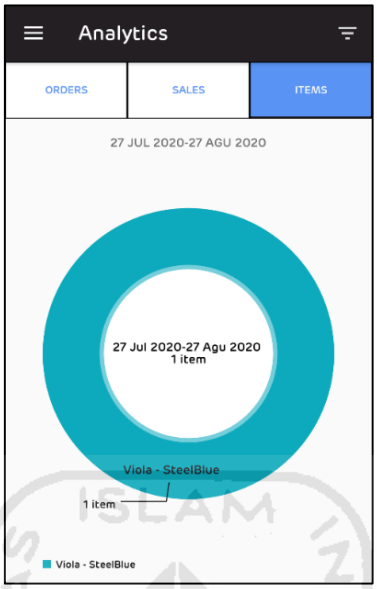
Halaman *Analytics Sub Sales* merupakan hasil evaluasi dari halaman *analytics sub profit*. Halaman ini menampilkan total *revenue* dari penjualan dan total *order success*. Selain itu juga menampilkan semua produk yang terjual dengan total rupiahnya. Lihat Gambar 4.16 untuk lebih detail.



Gambar 4.16 Hasil Halaman *Analytics Sub Sales*

7. Halaman *Analytics Sub Items*

Halaman *Analytics Sub Items* menampilkan analitik terkait item yang terjual. Data ditampilkan dalam bentuk *pie chart.* Lihat Gambar 4.17 untuk lebih detail.



Gambar 4.17 Hasil Halaman *Analytics Sub Items*

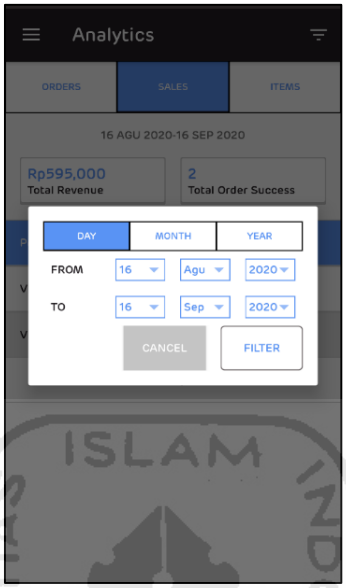
8. Halaman *Filter Analytics Sub Orders*

Halaman *Filter Analytics Sub Orders* merupakan halaman untuk menyaring data yang akan ditampilkan pada halaman *Analytics Sub Orders.* Data yang dapat disaring adalah status pesanan, dan tanggal. Lihat Gambar 4.18 untuk lebih detail.

Gambar 4.18 Hasil Halaman *Filter Analytics Sub Orders*

9. Halaman *Filter Analytics Sub Sales*

Halaman *Filter Analytics Sub Sales* merupakan halaman untuk menyaring data yang akan ditampilkan pada halaman *Analytics Sub Sales*. Data yang dapat disaring adalah tanggal. Lihat Gambar 4.19 untuk lebih detail.



Gambar 4.19 Hasil Halaman *Filter Analytics Sub Sales*

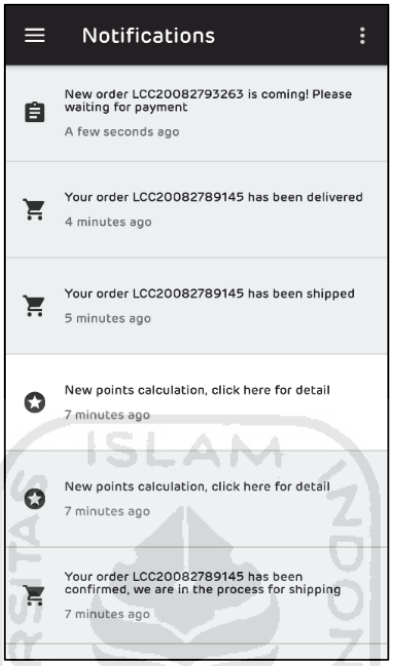
10. Halaman *Filter Analytics Sub Items*

Halaman *Filter Analytics Sub Items* merupakan halaman untuk menyaring data yang akan ditampilkan pada halaman *Analytics Sub Items*. Data yang dapat disaring adalah produk yang akan ditampilkan dan tanggal. Lihat Gambar 4.20 untuk lebih detail.

Gambar 4.20 Hasil Halaman *Filter Analytics Sub Items*

11. Halaman *Notifications*

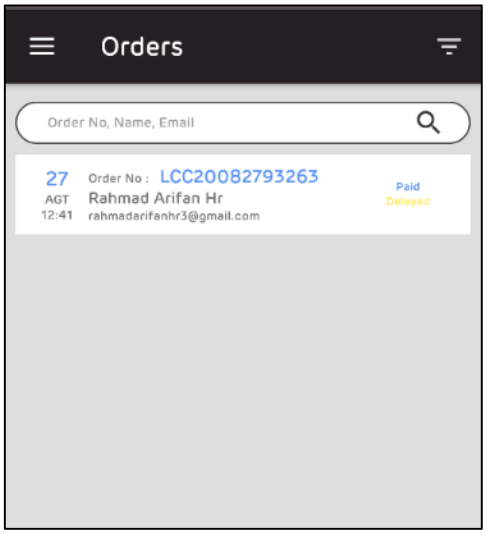
Halaman ini berfungsi untuk menampilkan list notifikasi yang masuk ke pengguna. Untuk jenis notifikasinya telah dijelaskan pada *prototype* halaman *notifications* pada subbab 3.2.1. Lihat Gambar 4.21 untuk lebih detail.



Gambar 4.21 Hasil Halaman *Notifications*

12. Halaman *Orders*

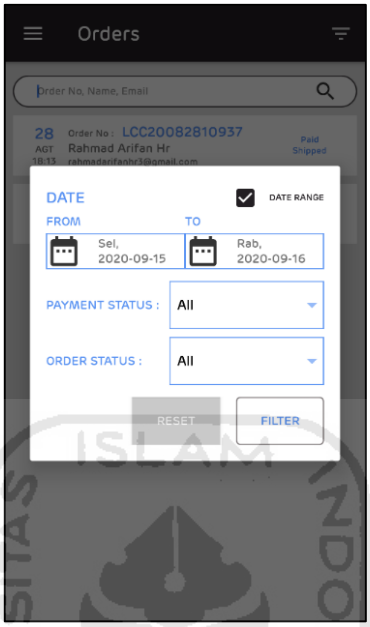
Halaman ini berfungsi untuk menampilkan list pesanan yang masuk ke pengguna. Lihat Gambar 4.22 untuk lebih detail.



Gambar 4.22 Hasil Halaman *Orders*

13. Halaman *Filter Orders*

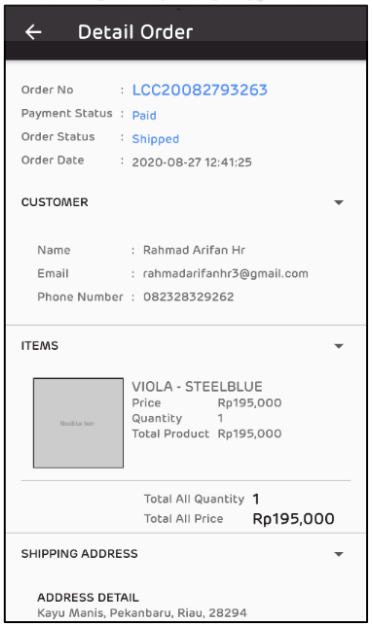
Halaman *Filter Orders* berfungsi untuk menyaring list pesanan yang masuk. Data yang bisa disaring antara lain tanggal, status pembayaran, dan status pesanan. Lihat Gambar 4.23 untuk lebih detail.



Gambar 4.23 Hasil Halaman *Filter Orders*

14. Halaman *Detail Order*

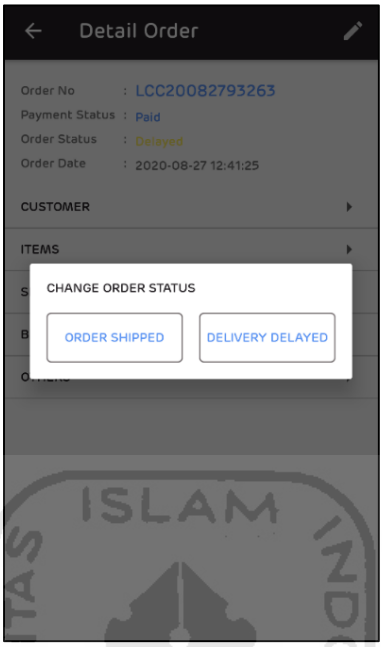
Halaman ini untuk menampilkan detail pesanan yang masuk. Lihat Gambar 4.24 untuk lebih detail.



Gambar 4.24 Hasil Halaman *Detail Orders*

15. Halaman Ubah *Order Status*

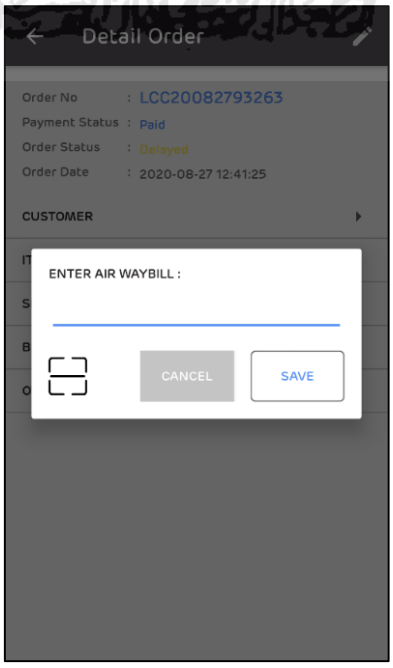
Halaman ini untuk mengubah status pesanan menjadi *shipped* atau *delayed.* Lihat Gambar 4.25 untuk lebih detail.



Gambar 4.25 Hasil Halaman Ubah *Order Status*

16. Halaman *Input Waybill*

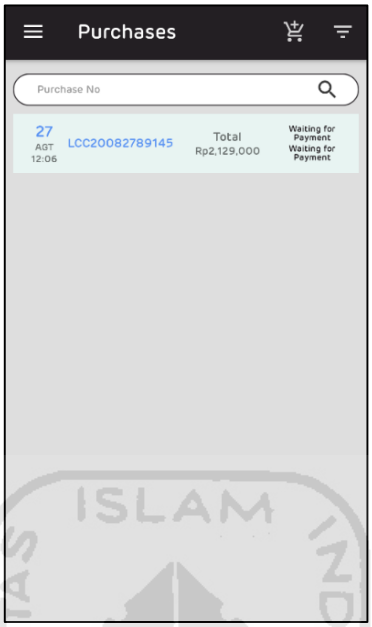
Halaman ini untuk memasukkan nomor resi dari pengiriman. Halaman ini akan muncul jika status pesanan diubah menjadi *shipped.* Lihat gambar Gambar 4.26 untuk detail.



Gambar 4.26 Hasil Halaman *Input Waybill*

17. Halaman *Purchases*

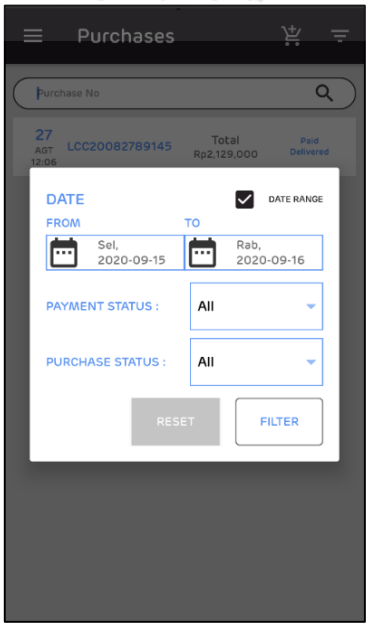
Halaman *Purchases* berfungsi untuk menampilkan list pembelian yang berasal dari pengguna yang telah *login.* Lihat Gambar 4.27 untuk lebih detail.



Gambar 4.27 Hasil Halaman *Purchases*

18. Halaman *Filter Purchases*

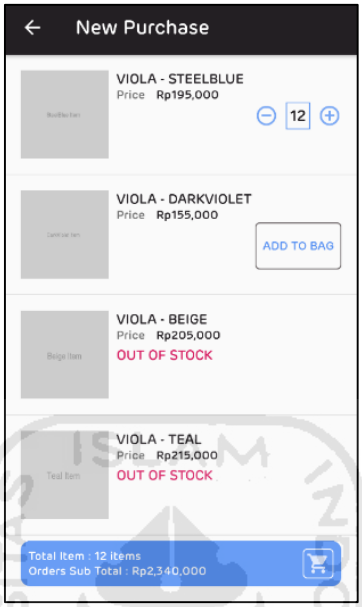
Halaman ini berfungsi untuk menyaring list pembelian yang berasal dari pengguna. Data yang bisa disaring adalah tanggal, status pembayaran, dan status pembelian. Lihat Gambar 4.28 untuk lebih detail.



Gambar 4.28 Hasil Halaman *Filter Purchases*

19. Halaman *New Purchase*

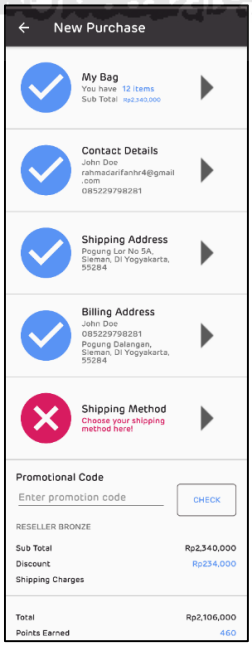
Halaman *New Purchase* merupakan halaman awal dari pembelian produk. Halaman ini menampilkan produk yang dijual oleh PT. Avo dan akan dibeli oleh pengguna nantinya. Lihat Gambar 4.29 untuk lebih detail.



Gambar 4.29 Hasil Halaman *New Purchase*

20. Halaman *List Form New Purchase*

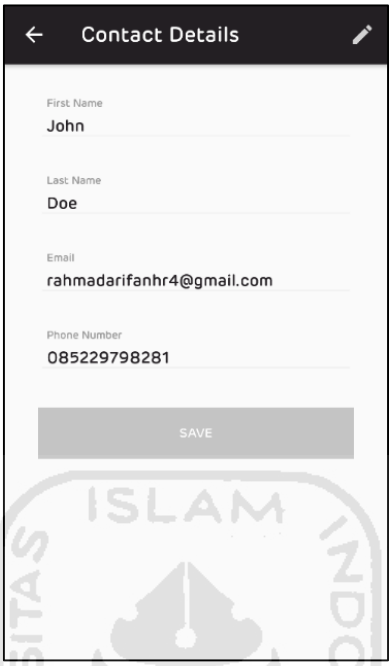
Halaman ini menampilkan *form* – *form* yang harus dilengkapi oleh pengguna. Lihat Gambar 4.30 untuk lebih detail.



Gambar 4.30 Hasil Halaman *List Form New Purchase*

21. Halaman *Form Contact Details New Purchase*

Halaman ini merupakan halaman untuk melengkapi *form* kontak detail dari pembeli. Lihat Gambar 4.31 untuk lebih detail.



Gambar 4.31 Hasil Halaman *Form Contact Details New Purchase*

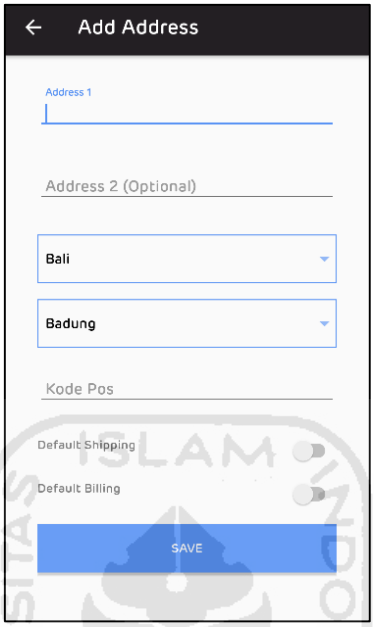
22. Halaman *Form Shipping Address New Purchase*

Halaman ini merupakan halaman untuk memilih alamat untuk pengiriman. Lihat Gambar 4.32 untuk lebih detail.

Gambar 4.32 Hasil Halaman *Form Shipping Address New Purchase*

23. Halaman *Form Add New Address*

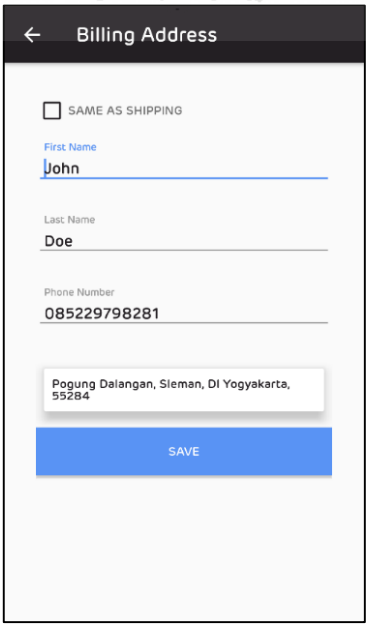
Halaman ini berfungsi untuk menambahkan alamat baru yang akan digunakan untuk *shipping address* atau *billing address.* Lihat Gambar 4.33 untuk lebih detail.



Gambar 4.33 Hasil Halaman *Form Add New Address*

24. Halaman *Form Billing Address New Purchase*

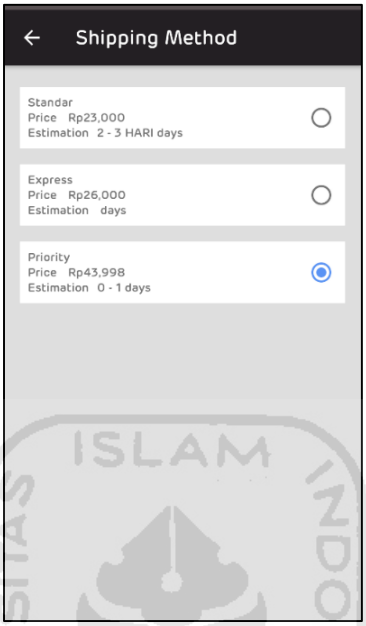
Halaman ini berfungsi untuk menambahkan detail dari alamat penagihan. Lihat Gambar 4.34 untuk lebih detail.



Gambar 4.34 Hasil Halaman *Form Billing Address New Purchase*

25. Halaman *Form Shipping Method New Purchase*

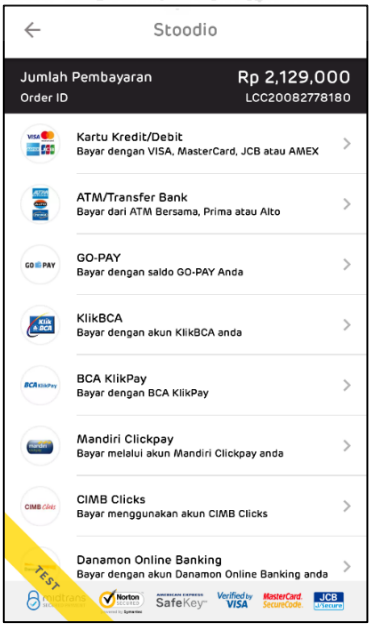
Halaman ini berfungsi untuk memilih metode pengiriman. Metode pengiriman yang ditampilkan di aplikasi sesuai dengan metode pengiriman yang ditampilkan di website lacoco.co.id. Lihat Gambar 4.35 untuk lebih detail.



Gambar 4.35 Hasil Halaman *Form Shipping Method New Purchase*

26. Halaman *Payment New Purchase*

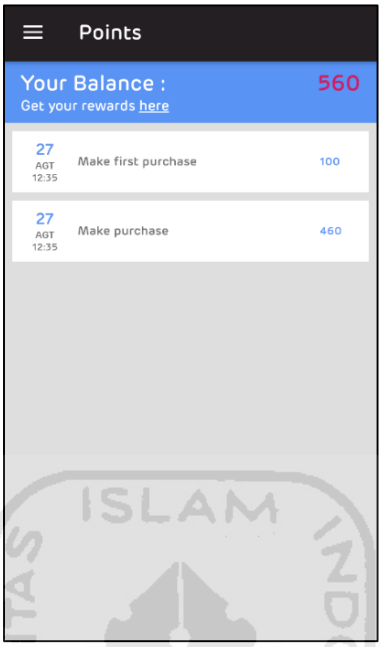
Halaman ini adalah halaman memilih metode pembayaran. Halaman ini ditampilkan dari *library* midtrans *mobile sdk.* Lihat Gambar 4.36 untuk lebih detail.



Gambar 4.36 Hasil Halaman *Payment New Purchase*

27. Halaman *Points*

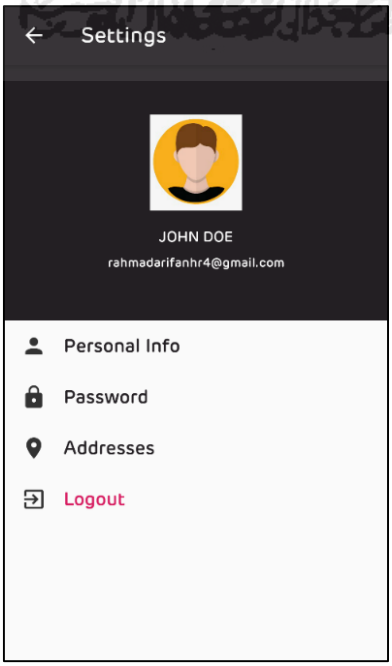
Halaman ini berfungsi untuk menampilkan list poin yang didapatkan oleh pengguna. Lihat Gambar 4.37 untuk lebih detail.



Gambar 4.37 Hasil Halaman *Points*

28. Halaman *Settings*

Halaman ini menampilkan pengaturan yang tersedia di aplikasi Lacoco Mobile. Lihat Gambar 4.38 untuk lebih detail.



Gambar 4.38 Hasil Halaman *Settings*

4.2 Pengujian

Setelah selesai mengimplementasikan halaman – halaman yang diharapkan pada analisis kebutuhan subbab 3.1, selanjutnya adalah tahapan pengujian. Pengujian ini dilakukan agar memastikan semua fitur yang dibangun sesuai dengan apa yang diharapkan oleh penulis dan juga oleh pemilik aplikasi yaitu PT. AVO. Pengujian pada penelitian kali ini dilakukan dengan dua cara, yaitu pengujian *black box* dan pengujian langsung kepada pemilik aplikasi.

4.2.1 Pengujian *Black Box*

Pengujian *black box* merupakan pengujian yang dilakukan dengan mengamati hasil eksekusi berdasarkan data uji dan fungsional dari perangkat lunak yang dibangun. Dengan analogi seperti kita melihat sebuah kotak hitam, yang mana kita hanya bisa melihat penampilan luarnya saja, tanpa tahu ada apa dibalik kotak hitam tersebut. Sama seperti halnya pengujian *black box*, melakukan evaluasi hanya dari sisi tampilan luar (*interface*) dan fungsionalitasnya, tanpa mengetahui seperti apa sesungguhnya yang terjadi dalam proses pembangunannya (Astuti, 2018).

Jadi dapat disimpulkan bahwa *black box* adalah pengujian yang hanya menjalankan semua fungsi dan fitur yang ada, lalu dievaluasi apakah hasil dari menjalankan fungsi dan fitur tersebut sesuai dengan yang diharapkan. Berikut hasil pengujian *black box* yang telah dilakukan penulis yang dirangkum pada Tabel 4.2.

Tabel 4.2 Tabel Hasil Pengujian *Black Box*

No.	Nama Pengujian	Kegiatan	Hasil	Status
1	*Splashscreen*	Membuka aplikasi Lacoco Mobile pertama kali	Menampilkan logo Lacoco selama 3 detik	Sukses
2	*Register* sukses	Mendaftar menjadi *reseller* dengan data terisi lengkap dan benar	Menampilkan dialog "Thankyou for your interest to become out partner"	Sukses
3	*Register* data salah	Mendaftar menjadi *reseller* dengan data yang salah atau tidak lengkap	Menampilkan *alert error* sesuai dengan data yang salah atau tidak lengkap	Sukses

No.	Nama Pengujian	Kegiatan	Hasil	Status
4	*Register* email yang telah terdaftar	Mendaftar menjadi *reseller* dengan email yang telah terdaftar	Menampilkan *alert* "The email has already been taken"	Sukses
5	*Login* sukses	*Login* dengan email dan password yang telah di-*accept* oleh admin menjadi *Reseller*	Menampilkan halaman awal yaitu Menu *Analytics*	Sukses
6	*Login* gagal	*Login* dengan email dan password salah	Menampilkan *alert* "Unauthorised"	Sukses
7	*New Purchase*	Memilih barang yang akan di beli lalu klik ikon keranjang	Menampilkan list *form* yang harus dilengkapi	Sukses
8	*New Purchase – My Bag*	Memilih *form* "My Bag"	Menampilkan list produk yang tersedia	Sukses
9	*New Purchase – Contact Details*	Memilih *form* "Contact Details" dan memasukkan kontak detail dari pembeli	Menampilkan kontak detail yang telah diisi	Sukses
10	*New Purchase – Shipping Address*	Memilih *form* "Shipping Address" dan memilih salah satu alamat	Menampilkan list *Shipping Address*	Sukses
11	*New Purchase – Billing Address*	Memilih *form* "Billing Address" lalu memasukkan data penagihan	Menampilkan data *Billing Address*	Sukses
12	*New Purchase – Shipping Method*	Memilih *form* "Shipping Method" lalu memilih salah satu metode pengiriman	Menampilkan list metode pengiriman yang tersedia	Sukses
13	*New Purchase – Payment*	Mengklik tombol "Payment"	Menampilkan list metode pembayaran yang disediakan oleh Midtrans	Sukses
14	List *Purchases*	Memilih menu *Purchases*	Menampilkan list *purchases*	Sukses
15	Detail *Purchase*	Memilih menu *Purchases,* lalu pilih salah satu pesanan	Menampilkan detail *purchase*	Sukses
16	List *Order*	Memilih menu *Orders*	Menampilkan list *orders*	Sukses
17	Detail *Order*	Memilih menu *Orders,* lalu pilih salah satu pesanan	Menampilkan detail *order*	Sukses

No.	Nama Pengujian	Kegiatan	Hasil	Status
18	Mengubah status *Order*	Memilih menu *Orders*, dan memilih *Order* yang akan diubah statusnya	Menampilkan *alert* sukses sesuai statusnya	Sukses
19	List Notifikasi	Memilih menu *Notifications*	Menampilkan list notifikasi	Sukses
20	Notifikasi dari server	Melakukan pembelian produk, atau *customer* membeli produk	Menampilkan notifikasi di bar notifikasi pada HP	Sukses
21	*Analytics - Orders*	Memilih menu *Analytics* dan tab *Orders*	Menampilkan grafik *line chart* terkait pesanan yang masuk dari *customer*	Sukses
22	*Analytics - Sales*	Memilih menu *Analytics* dan tab *Sales*	Menampilkan total *revenue*, *order success*, dan tabel yang berisi produk yang terjual beserta total rupiah	Sukses
23	*Analytics – Items*	Memilih menu *Analytics* dan tab *Items*	Menampilkan *pie chart* terkait produk yang terjual dalam bentuk total item	Sukses
24	List poin	Memilih menu *Points*	Menampilkan list poin	Sukses

4.2.2 Pengujian kepada Pemilik Aplikasi

Pengujian ini dilakukan dengan cara langsung menguji sistem kepada pemilik aplikasi yaitu PT. AVO, dengan mendengarkan respon penggunanya. Diharapkan melalui pengujian ini mendapatkan impresi pemilik aplikasi berdasarkan pertanyaan yang telah diajukan oleh penulis. Pengujian ini dilakukan pada tanggal 28 Agustus 2020 kepada pemilik aplikasi yaitu PT. Avo Innovation Technology yang diwakilkan oleh Aris Nurul Huda.

Untuk mendapatkan impresi pemilik aplikasi, maka penulis memberikan beberapa pertanyaan yang akan dijawab oleh pengguna. Berikut pertanyaan dan jawaban dari impresi pemilik aplikasi yang juga dilampirkan pada Tabel 4.3.

1. *Device* apa yang digunakan?
   Jawaban : Samsung A20S

2. Versi android di *device* anda?
   Jawaban: Android 10

Tabel 4.3 Pertanyaan dan Jawaban tentang Impresi Pemilik Aplikasi

No	Pertanyaan	1	2	3	4	5
3.	Saya merasa mudah dalam mendaftar menjadi *Reseller*				✓	
4.	Saya merasa mudah dalam melakukan login				✓	
5.	Saya merasa mudah memahami informasi terkait analitik					✓
6.	Saya merasa mudah dalam melakukan *purchase* atau pembelian produk dari *reseller* ke PT. Avo				✓	
7.	Saya merasa dipermudah ketika melihat riwayat *purchase*					✓
8.	Saya merasa dipermudah ketika melihat pesanan yang masuk dari *customer* / *orders*					✓
9.	Saya merasa mudah dalam menggunakan aplikasi Lacoco Mobile				✓	
10.	Aplikasi ini sangat penting bagi *reseller*					✓
11.	Notifikasi sangat penting bagi saya ketika ada *order* baru dari customer					✓
12.	Aplikasi ini sangat efektif digunakan ketika *reseller* ingin membeli produk ke PT. Avo				✓	
13.	Aplikasi ini memiliki tampilan yang menarik				✓	

Keterangan :

1. Sangat tidak setuju
2. Tidak setuju
3. Biasa saja
4. Cukup setuju
5. Sangat setuju

14. Apakah ada *feedback* terkait aplikasi?

- Belum ada fitur *forgot password* di aplikasi.

Fitur *forgot password* cukup penting untuk diimplementasikan. Fitur ini bertujuan untuk mereset *password* pengguna, karena bisa saja pengguna lupa terhadap *password* yang dimilikinya.

- Ongkos kirim yang masih *overprice*, bisa disesuaikan lagi dengan ekspedisi khusus kargo

Ongkos kirim yang ditampilkan di aplikasi Lacoco Mobile hanya menampilkan dari tiga metode pengiriman diantaranya *Standar, Express,* dan *Priority*. Metode pengiriman ini sudah sesuai dengan metode pengiriman yang ditampilkan di website lacoco.co.id. Salah satu solusi untuk *feedback* ini adalah dengan menampilkan semua metode pengiriman yang tersedia.

+ *Credential password* dibuat ketika pengguna diterima sebagai *reseller* oleh admin PT. AVO.

Membuat *password* ketika pengguna diterima sebagai *reseller* cukup penting, karena salah satu keuntungannya adalah pengguna hanya bisa *login* ketika telah diterima sebagai *reseller*.

+ Terdapat jumlah pesanan di menu *Analytics Sub Orders* sehingga *reseller* bisa melihat sekilas jumlah pesanannya.

Melihat jumlah pesanan dalam proses jual beli adalah hal yang sangat penting, untuk melihat seberapa perkembangan dan keuntungan selama proses penjualan oleh *reseller.*

+ Fitur *analytics* cukup membantu *reseller* untuk monitoring penjualan dan stok produk yang terjual

Fitur *analytics* cukup membantu *reseller* selama proses jual beli, diantaranya untuk monitoring penjualan dan memanajemen stok produk. Sehingga *reseller* bisa melihat perkembangan dari proses penjualannya dan memprediksi stok yang dibutuhkan untuk bulan selanjutnya.

Terkait dua poin minus yang disampaikan oleh Aris Nurul Huda, dikarenakan keterbatasan waktu dalam pengerjaan tugas akhir ini dan juga poin minus yang disampaikan berada diluar rumusan masalah utama maka *feedback* tersebut tidak diakomodasi pada penelitan kali ini. Namun tidak menutup kemungkinan akan diakomodasi atau disempurnakan pasca dituliskan laporan ini.

BAB V
PENUTUP

5.1 Refleksi *Prototyping*

Setelah menggunakan metode pengembangan *Evolutionary Prototyping* pada penelitian kali ini, penulis menyimpulkan bahwa terdapat empat tahap dalam membangun sistem antara lain analisis kebutuhan, desain, membangun sistem, dan pengujian. Secara umum tahapan ini sama dengan metode *waterfall*, namun perbedaan menggunakan metode *prototyping* adalah metode ini lebih berfokus kepada tahapan desain. Tahapan desain pada metode *prototyping* terdapat beberapa iterasi atau perulangan bergantung dari jumlah fitur yang dimiliki dan tingkat kepuasan *customer* (pemilik aplikasi) terhadap desain yang telah dibuat. Setiap iterasi tersebut terdapat beberapa tahapan yaitu membuat desain sederhana, membuat *prototype*, evaluasi desain ke *customer,* dan merevisi desain jika desain belum sesuai dengan keinginan dan kebutuhan *customer*.

Selain itu terdapat beberapa kelebihan dan kekurangan yang penulis dapat simpulkan berdasarkan penelitian yang penulis lakukan. Diantara kelebihan yang penulis dapatkan adalah komunikasi yang baik antara pengembang sistem dan *customer* karena *customer* terlibat aktif selama tahapan desain, serta dengan terlibatnya *customer* hasil dari desain sistem lebih reliable karena sesuai dengan kebutuhan dan keinginan *customer*. Lalu kekurangan yang penulis rasakan selama pengerjaan penelitian ini adalah kurang fleksibel atau tidak bisa menerima perubahan desain dan fitur jika telah memasuki tahapan implementasi, dalam kata lain jika desain sudah ditetapkan maka selama implementasi *customer* tidak bisa menerima perubahan desain sistem. Tidak menerima perubahan desain ini terjadi karena setiap tahapan pada metodologi *prototyping* bersifat sekuensial atau berurutan, sehingga jika telah memasuki tahap implemenetasi tidak bisa kembali ke tahap desain. Kekurangan selanjutnya menggunakan metode *prototyping* adalah jumlah iterasi yang tidak jelas, karena iterasi akan berhenti ketika *customer* sudah puas dengan desain yang telah dibuat, oleh karena itu iterasi ini bisa saja membuat waktu pengerjaan jadi lebih lama.

5.2 Kesimpulan

Berdasarkan hasil penelitian pengembangan aplikasi mobile untuk distributor Lacoco menggunakan metode prototyping terdapat beberapa kesimpulan, yaitu:

a. Aplikasi yang telah dibuat sudah dapat digunakan dan berfungsi dengan baik sesuai dengan yang diharapkan penulis dan *customer*.
b. Fitur-fitur yang ada dibutuhkan oleh pihak AVO yaitu notifikasi, manajemen pesanan, analitik terkait riwayat penjualan, dan pembelian produk dari distributor ke AVO sudah diimplementasikan dan berfungsi dengan baik.
c. Terdapat dua kali iterasi selama tahapan desain dan tiga evolusi yang terjadi.
d. Hasil pengujian *black box* pada semua halaman yang telah dibuat memiliki nilai sukses sesuai dengan harapan penulis.
e. Berdasarkan hasil pengujian kepada pemilik aplikasi pada subbab 4.2.2 dapat disimpulkan bahwa aplikasi dapat diterima dengan baik oleh *customer*

5.3 Saran

Setelah melakukan penelitian pada tugas akhir ini, penulis merasa ada beberapa hal yang masih memiliki kekurangan, yang mana kekurangan tersebut masih bisa dikembangkan pada penelitian selanjutnya. Berikut beberapa saran yang bisa penulis berikan :

1. Membuat fitur pemilihan distributor secara otomatis ketika *checkout* melalui website.
2. Melibatkan distributor dalam proses desain dan pengujian, karena aplikasi akan digunakan langsung oleh distributor.

DAFTAR PUSTAKA

Astuti, P. (2018). Penggunaan Metode Black Box Testing (Boundary Value Analysis) Pada Sistem Akademik (SMA/SMK).

Guru99. (2020). *Prototyping Model in Software Engineering: Methodology, Process, Approach*. Retrieved from Guru99: https://www.guru99.com/software-engineering-prototyping-model.html

Huda, A. N. (2018). Pengembangan Sistem Informasi dan Penjualan Lacoco Berbasis Website.

Kemenperin. (2018, Maret 20). Retrieved from https://kemenperin.go.id/artikel/18957/Industri-Kosmetik-Nasional-Tumbuh-2018

Kothalawala, A. (2018, Juny 14). *What is an API? How does it work?* Retrieved from https://medium.com/@ama.thanu/what-is-an-api-how-does-it-work-f4ea552d741f

Midtrans. (2020). Retrieved from https://midtrans.com/

RajaOngkir. (2020). Retrieved from https://rajaongkir.com/

Surguy, M. (2013, July 27). *History of Laravel PHP framework, Eloquence emerging*. Retrieved from https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

Tamplin, J. (2014, October 21). *Firebase is Joining Google!* Retrieved from https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/

LAMPIRAN