ANALISIS DAN PERANCANGAN SISTEM PADA *LAUNDRY* HICLEAN

Disusun oleh :

Prasasti Indah Rahayu	**00000028838**
Victoria De Greatha	**00000028843**
Angelina Sanjaya Sugrinting	**00000028887**
Nathalia Hermanto	**00000029105**
Kiky Melani	**00000030141**

Fakultas Teknik dan Informatika

Program Studi Sistem Informasi

Universitas Multimedia Nusantara

2019

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas rahmat yang telah Ia berikan sehingga kami dapat menyelesaikan laporan mengenai "Analisis dan Perancangan Sistem pada *Laundry* HiClean sebagai proyek kelompok Ujian Akhir Semester mata kuliah Analisis dan Perancangan Sistem Informasi dengan bimbingan dari Bapak Johan Setiawan, S.Kom., M.B.A. sebagai dosen mata kuliah Analisis dan Perancangan Sistem Informasi.

Meski terdapat batasan dalam ilmu yang kami miliki, namun dengan bimbingan Bapak Johan Setiawan, S.Kom., M.B.A., kami dapat menyelesaikan laporan ini. Untuk itu, kami ucapkan terima kasih kepada Tuhan Yang Maha Esa, Bapak Johan Setiawan, S.Kom., M.B.A. sebagai dosen pembimbing mata kuliah Analisis dan Perancangan Sistem Informasi, dan orang tua kami.

Bila terdapat kesalahan dalam laporan ini, kami mohon maaf atas kesalahan-kesalahan tersebut. Masih banyak sumber-sumber literatur yang perlu diambil, maka hendaknya dalam penyelesaian laporan ini, kami berharap semoga laporan ini dapat menjadi sumber referensi pembelajaran bagi penulis-penulis berikutnya dengan topik yang sama.

Demikian, semoga laporan ini bermanfaat untuk kami sendiri sebagai penulis maupun pembaca laporan ini.

Tangerang, 17 November 2019

Penulis

DAFTAR ISI

BAB I 1
1.1 The Situation 1
1.2 System Definition 2
BAB II 4
2. 1 Class 4
2.1.1 Find Candidate 4
2.1.2 Select Candidate 5
2.1.3 Event Table (Generic) 9
2. 2 Structure 10
2.2.1 Generalization 10
2.2.2 Aggregation 10
2.2.3 Association 11
2.2.4 Class Diagram 12
2.3 Behaviour 13
2.3.1 Statechart Diagram 13
2.3.2 Behavioral Pattern 16
BAB III 17
3.1. Usage 17
3.1.1 Use Case 17
3.1.2 Actor 19
3.1.3 Activity Diagram 21
3.2. Function 22
3.3 Interface 23
BAB IV 29
4.1 Criteria 29
4.2 Component Architecture 31
4.3 Process Architecture 35
BAB V 38
5.1 Model Component 38
5.2 Function Component 41
5.3 Connecting Component 45
BAB VI 55
DAFTAR PUSTAKA 56

BAB I

PENDAHULUAN

1.1 The Situation

HiClean merupakan sebuah *laundry* yang sedang berkembang di Tangerang. Saat ini, HiClean memiliki dua cabang, yakni di Gading Serpong sebagai pusat dan Bintaro sebagai cabang. HiClean memiliki beberapa pilihan servis antara lain cuci kiloan, *dry clean*, cuci sepatu, cuci karpet, dan *home service*.

HiClean memiliki layanan dimana *client* dapat membawa barang yang akan dicuci langsung ke HiClean atau dapat menghubungi melalui *WhatsApp,* sehingga karyawan HiClean dapat mengambil barang yang akan dicuci, seperti pakaian kotor, sepatu, karpet, dan lain-lain ke rumah *client*. Ketika *client* membawa barang yang ingin dicuci langsung ke HiClean maka barang tersebut akan dicuci ditimbang, dihitung dan transaksi pembayaran dilakukan saat itu juga. Namun jika *client* meminta HiClean untuk mengambil pakaian kotor ke rumah *client* maka pihak HiClean akan menghitung, menimbang serta transaksi pembayaran dilakukan secara langsung di rumah *client.*

Setelah itu karyawan mencatat nama, nomor telepon pelanggan, serta memberikan bukti pembayaran yang berisi nomor pelanggan dan bukti pembayaran tersebut akan dikembalikan kepada karyawan kembali sebagai bukti pengambilan baju yang sudah selesai di *laundry*.

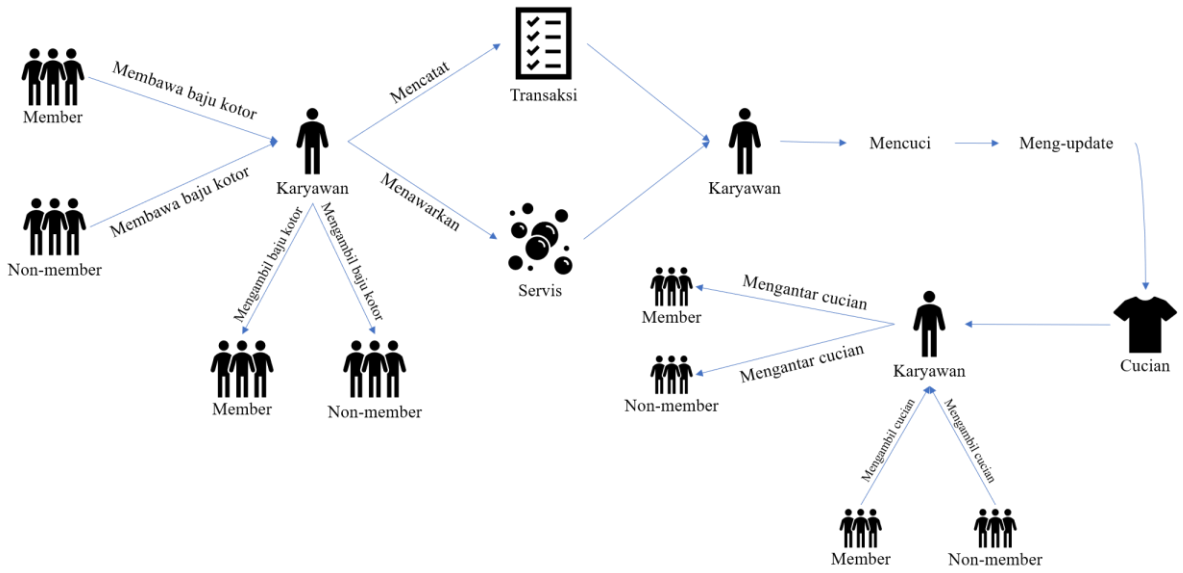
Nomor pelanggan beserta nama, nomor telepon beserta jumlah transaksi akan dicatat ke dalam buku besar yang berfungsi untuk menyimpan riwayat pelanggan. Saat ini sistem yang digunakan oleh HiClean adalah sistem untuk peng-*input*-an data pelanggan dan transaksi, tetapi data tersebut hanya tersimpan dan dapat diakses dalam jangka waktu satu tahun.

Karyawan yang dipekerjakan oleh HiClean tidak terikat dengan kontrak. HiClean menyediakan *membership* dengan persyaratan untuk mengikuti *membership* tersebut adalah memberikan data berupa nama, alamat, dan bukti transaksi minimal 3 kg tanpa batas waktu di HiClean.

Saat ini HiClean belum memiliki sistem *tracking* untuk *client* melacak proses cucian. Beberapa *client* datang ke HiClean karena beranggapan cuciannya sudah selesai, namun ternyata cuciannya belum selesai dengan waktu estimasi yang sudah ditentukan. Hal ini terjadi karena bisa saja karyawan tersebut tidak selalu mengabari melalui *WhatsApp, client* tidak membaca pesan *WhatsApp,* atau karyawan yang tidak memberitahu dari awal bahwa akan diberitahu lewat *WhatsApp* sehingga pelanggannya tidak mengetahui proses cuciannya.

Maka dari itu, kami mengusulkan bahwa permasalahan ini membutuhkan sistem *tracking*, dimana *client* yang menggunakan jasa HiClean dapat melacak proses cuciannya melalui sebuah sistem. Sistem ini menyediakan fitur *inbox,* dimana pada fitur ini akan memberikan *tracking number* sehingga *client* dapat mengetahui proses cucian dari awal hingga selesai. Pihak HiClean juga tidak perlu mengabari *client* satu per satu melalui *WhatsApp.*

Rich Picture



Gambar 1.1.1. Rich picture proses bisnis Laundry HiClean

Rich Picture adalah gambar informal yang merepresentasikan sebuah ilustrasi dari sebuah situasi. *Rich picture* ini menjelaskan proses bisnis yang terjadi di HiClean. *Client member* dan *non-member* membawa baju kotor ke karyawan. Karyawan juga bisa mengambil pakaian/barang kotor ke rumah *client.* Kemudian karyawan mencatat transaksi dari *client*, dan juga menawarkan servis *laundry* yang akan dipilih oleh *client*. Kemudian karyawan akan mencuci barang yang akan dicuci, lalu meng-*update status* pencucian *client* di sistem. Ketika cucian sudah selesai, maka ada dua pilihan yang dapat dipilih oleh baik *client member* maupun *client non-member*. Pilihan tersebut antara lain *client* bisa mengambil barang cuciannya langsung di HiClean-nya, atau *client* bisa meminta karyawan untuk mengantarkan barang cuciannya ke rumah *client*.

1.2 System Definition

a. Functionality
   Membantu *client* dalam melacak proses cucian mereka dan membantu HiClean dalam menyimpan data *client* dan data transaksi.
b. Application Domain
   Mengelola *client member* dan *client* non-*member*, pencatatan transaksi yang telah dilakukan dan meng-*update* proses cucian *client*.
c. Condition
   Sistem beroperasi dan dikembangkan dalam lingkungan kerjasama yang erat antara karyawan dengan *client* member dan *client* non-member.
d. Technology
   PC yang terhubung ke akses internet yang memadai untuk meng-*update* proses cucian *client* serta alat untuk mencetak *receipt* pembayaran.

e. Object
   Karyawan, *client*, transaksi, cucian, member.
f. Responsibility.
   Peralatan administrasi dan pencatatan data transaksi dan *client*.

BAB II

PROBLEM DOMAIN

Berdasarkan Mathiassen, Madsen, Nielsen dan Stage (2000, p.18) terdapat empat prinsip yang digunakan untuk menganalisis dan merancang suatu sistem yaitu:

1. Pemodelan Konteks (*Model the Context)*
   Model dalam konteks suatu sistem dilihat dari 2 perspektif bagian dalam konteks yaitu *problem domain* dan *application domain*. Problem domain adalah bagian yang termasuk ke dalam suatu konteks yang dikendalikan, dikontrol dan diatur oleh sebuah sistem sementara *application domain* adalah suatu organisasi yang mengatur, mengelola, dan mengendalikan suatu problem domain dengan memiliki tujuan untuk mengidentifikasi kebutuhan-kebutuhan yang diperlukan dalam model dalam sistem dan apa saja yang dibutuhkan pengguna sistem dalam sistem itu sendiri.
2. Penekanan pada Arsitektur (*Emphasize the Architecture*)
   Analisis dan perancangan sistem yang berorientasi objek menekankan arsitektur sistem harus memberikan kemudahan kepada pengguna sistem agar dapat mudah dipahami karena arsitektur sistem merupakan bagian dasar dari suatu sistem yang membantu pengguna sistem untuk pengambilan keputusan dan menjadikan arsitektur sistem sebagai alat komunikasi dan kerja dalam pengembangan suatu sistem berikutnya.
3. Penggunaan Kembali Pola-Pola (*Reuse Patterns*)
   Dalam analisis dan perancangan berorientasi objek ini memberikan cara untuk memastikan efisiensi dan kualitas dalam analisis dan perancangan yang telah dibuat dengan 2 cara yaitu dengan menggunakan objek dan komponen yang membangun sistem dan bisa dengan menggunakan pola dari analisis dan perancangan itu sendiri.
4. Penyesuaian Metode (*Tailor The Method*) OOAD
   Penyesuaian metode ini dianggap sebagai pegangan atau pedoman dari analisis dan perancangan suatu sistem sehingga perlunya penyesuaian terhadap organisasi. Namun dalam penyesuaiannya tersebut, dibutuhkan perancangan suatu sistem yang mudah dalam adaptasi, pergantian, dan perbaikan. OOAD (*Object Oriented Analysis & Design*) sendiri menggambarkan 4 aspek dalam sistem dan konteksnya yaitu bagaimana sistem digunakan, isi informasi yang terdapat dalam sistem dan sistem yang dijadikan sebagai keseluruhan komponen-komponen dalam sistem itu sendiri.

Aspek-aspek tersebut terkoneksi dengan aktivitas-aktivitas utama dalam analisis dan perancangan sistem berorientasi objek yaitu *problem domain, application domain, architectural design,* dan *component design*.

2. 1 Class

2.1.1 Find Candidate

Tabel 2.1.1. Daftar *candidate class* sistem *laundry* HiClean

Class	***Event***
Client member	Menjemput
Client non-*member*	Mengantar
Karyawan	Mencuci
Owner	Memasukkan
Servis *laundry*	Menghitung
Status cucian	Membayar
Tempat cucian	Mendaftar
Sabun	Menelepon
Mesin cuci	Memeriksa
Pewangi	Membatalkan
Setrika	Memilih
Motor	Mencetak struk
Timbangan	Mempekerjakan
Transaksi	Memberhentikan
Telepon	Meng-*update* status
WhatsApp	Membuat
Cash, Debit, Credit	Mengambil
Struk	

2.1.2 Select Candidate

Kandidat-kandidat *class* dan *event* yang sudah terdaftar diseleksi kembali. *Class* dan *event* yang dipilih adalah *class* dan *event* yang berperan di dalam sistem. Sementara *class* dan *event* yang tidak berhubungan dengan sistem dieliminasi.

Tabel 2.1.2. Daftar *class* yang digunakan sistem *laundry* HiClean

Class	***Event***
Client Member	Mendaftar
Client Non-*Member*	Membayar
Karyawan	Memeriksa
Transaksi	Membuat
Status cucian	Meng-*update* status
Servis *laundry*	Memasukkan
Owner	Memilih
	Membatalkan
	Mengantar
	Mengambil

Class

(yang terpilih)

a. *Client Member*

Client member adalah *client* yang sudah mendaftarkan diri sebagai *member*. *Client* ini dapat melakukan *laundry* dan dapat melacak proses cuciannya melalui sistem karena *client* ini memiliki akun. Selain itu, *client* yang sudah terdaftar sebagai *member* akan mendapatkan potongan harga.

b. *Client Non-Member*

Client non-member adalah *client* yang belum mendaftarkan diri sebagai *member*. Sama seperti *client member* dimana *client* ini bisa melakukan *laundry* dan melacak proses cuciannya melalui sistem, tetapi yang membedakan antara *member* dan *non-member* adalah *client* ini tidak mendapatkan potongan harga dan hanya dapat membayar dengan harga normal.

c. Karyawan

Karyawan bertugas untuk memberikan pelayanan kepada *client.* Karyawan bertanggung jawab dalam meng-*update* proses cucian ke dalam sistem. Karyawan juga mengirim pakaian/ barang yang sudah dicuci apabila *client* memilih pilihan untuk dikirim oleh karyawan.

d. *Owner*

Owner tidak memonitor pekerjaan karyawan tetapi *owner* menerima pendapatan dan memeriksa data transaksi selama setahun sekali.

e. Transaksi

Client member dan *client non-member* membayar servis yang telah dipilih. Kemudian transaksi ini akan diproses oleh karyawan.

f. Status Cucian

Status cucian adalah status yang menjelaskan proses cucian *client* dari awal sampai selesai. Pada *class* ini, *client* dapat melacak proses cucian mereka sudah sampai mana.

g. Servis *laundry*

Servis *laundry* merupakan pilihan servis yang dapat dipilih oleh *client*. Pilihan servis yang tersedia antara lain cuci kiloan, *dry clean,* cuci sepatu, cuci karpet, dan *home service*.

(yang tidak terpilih)

a. Tempat Cucian

Class tempat cucian merupakan tempat untuk melakukan proses cucian. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

b. Struk

Berisikan mengenai pilihan servis apa yang *client* pilih berikut dengan harga dari servis yang dipilih, tanggal berapa transaksi dilakukan dan juga jenis pembayaran apa yang digunakan *client* untuk membayar karena dalam sistem sudah memberikan fitur pemberitahuan sehingga tidak berhubungan dengan sistem terkait.

c. *Cash, debit, credit*

Merupakan cara pembayaran yang tersedia di HiClean untuk memudahkan *client* bertransaksi. *Class* ini tidak terpilih karena proses dari *class* ini tidak terkomputerisasi dalam sistem dan *class* ini sudah termasuk ke dalam bagian dari *class* transaksi.

d. Telepon

Telepon digunakan *client* untuk memeriksa, meminta dan mengantar pakaian/ barang yang sudah dicuci. Telepon tidak terpilih karena sudah terdapat sistem *tracking* untuk memeriksa proses cucian.

e. *WhatsApp*

WhatsApp digunakan *client* untuk memeriksa atau meminta mengantarkan *laundry*. *WhatsApp* tidak digunakan karena sudah terdapat sistem *tracking* untuk memeriksa proses cucian.

f. Motor

Motor merupakan kendaraan yang digunakan oleh karyawan untuk mengantarkan pakaian/barang yang sudah dicuci. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

g. Setrika

Setrika merupakan alat yang digunakan untuk membantu karyawan melipat pakaian. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

h. Mesin Cuci

Mesin cuci merupakan alat yang digunakan karyawan untuk melakukan proses pencucian barang/pakaian dari *client*. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

i. Timbangan

Timbangan merupakan alat yang digunakan untuk membantu karyawan menghitung jumlah pakaian *client*, serta menentukan harga cucian. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

j. Pewangi

Pewangi merupakan salah satu produk yang digunakan untuk mengharumkan cucian. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

k. Sabun

Sabun merupakan salah satu produk yang digunakan untuk membersihkan cucian. *Class* ini tidak terpilih karena *class* ini tidak berhubungan dengan sistem.

Event

(yang terpilih)

a. Mendaftar

Client dapat mendaftarkan akun lewat sistem. Setelah itu, *client* yang mendaftarkan diri sebagai *member* dapat dilakukan lewat sistem.

b. Membayar

Client melakukan proses pembayaran setelah memilih servis untuk barang/ pakaian kotor yang diberikan kepada pihak HiClean. Jenis pembayaran yang dapat dilakukan yaitu dapat melalui *cash, debit, credit.*

c. Memeriksa

Client dapat memeriksa status cuciannya pada *event* in dari tahap awal pencucian hingga tahap akhir pencucian barang/ pakaian kotor. Dan *owner* juga dapat memeriksa transaksi yang terjadi.

d. Mengantar

Karyawan yang sudah selesai mencuci pakaian *client* akan mengantarkan barang atau pakaian yang sudah bersih ke alamat *client* pada *event* ini dan *client* dapat mengantarkan barang atau pakaian kotor ke HiClean.

e. Meng-*update* status

Karyawan yang sedang melakukan proses mencuci hingga selesai akan meng-*update* status setelah menyelesaikan setiap proses.

f. Memilih

Client dapat memilih servis sesuai dengan kebutuhan. *Event* ini terpilih karena dalam sistem dapat menentukan servis *laundry* yang dibutuhkan.

g. Membatalkan

Client dapat membatalkan bila servis tidak sesuai dengan kebutuhan. *Event* ini terpilih karena dalam sistem dapat menentukan servis *laundry* yang dibutuhkan.

h. Membuat

Client dapat mendaftarkan diri menjadi *member* sesuai dengan syarat dan ketentuan yang sudah ditetapkan.

i. Memasukkan

Event ini menjelaskan karyawan melakukan kegiatan memasukkan data transaksi dari *client* ke sistem.

j. Mengambil

Event ini menjelaskan karyawan mengambil barang atau pakaian kotor dari *client* dan juga menjelaskan bahwa *client* dapat mengambil barang atau pakaian yang sudah bersih dari HiClean.

(yang tidak terpilih)

a. Menjemput

Karyawan dapat mengambil pakaian kotor di rumah *client*. *Event* ini tidak terpilih karena *event* ini tidak berhubungan dengan sistem.

b. Mencuci

Proses membersihkan barang/ pakaian kotor yang diberikan *client* kepada pihak HiClean. *Event* ini tidak terpilih karena tidak adanya hubungan dengan sistem terkait.

c. Menghitung

Proses ini untuk mengetahui jumlah dari barang/ pakaian kotor yang diberikan *client* kepada pihak HiClean. *Event* ini tidak terpilih karena tidak berhubungan dengan sistem.

d. Mencetak Struk

Dalam proses ini ketika sudah transaksi terjadi maka karyawan akan mencetak struk sebagai bukti transaksi. *Event* ini tidak terpilih karena tidak berhubungan dengan sistem.

e. Mempekerjakan

Dalam proses ini *owner* mempekerjakan karyawan. *Event* ini tidak terpilih karena tidak berhubungan dengan sistem.

f. Memberhentikan

Dalam proses ini *owner* dapat memberhentikan karyawan yang kinerjanya buruk. *Event* ini tidak terpilih karena *event* ini tidak berhubungan dengan sistem.

2.1.3 Event Table (Generic)

Dari *class* dan *event* yang terpilih akan menghasilkan *event table*. Bagian vertikal dalam *event table* diisi dengan *event* yang terpilih. Sedangkan bagian horizontal dalam *event table* diisi dengan *class* yang terpilih. Kemudian tanda silang (×) dalam *event table* menunjukkan adanya hubungan antara *class* dan *event*. Berikut merupakan *event table* mengenai hubungan antara *class* dan *event* dibawah ini.

Tabel 2.1.3. Tabel *event generic* sistem *laundry* HiClean

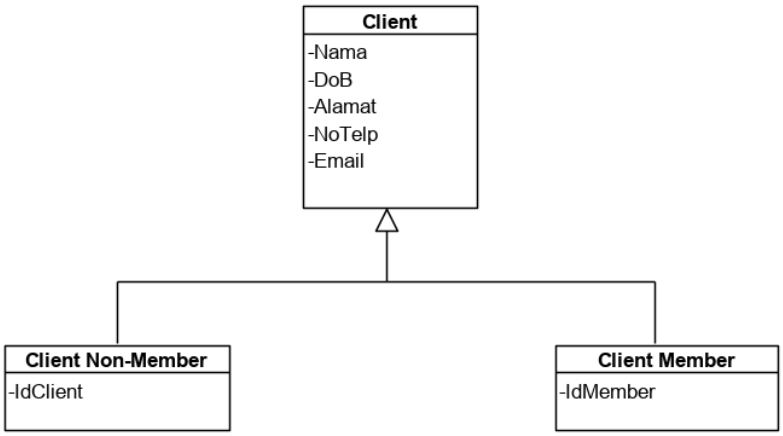
	Non-Member	*Member*	Karyawan	Transaksi	Status Cucian	Servis *Laundry*	*Owner*
Mendaftarkan	×						

Membuat		×					
Memeriksa	×	×			×		×
Meng-*update* Status			×				
Memilih	×	×		×		×	
Membatalkan	×	×		×		×	
Membayar	×	×		×			
Mengirim			×				

2. 2 Structure

2.2.1 Generalization

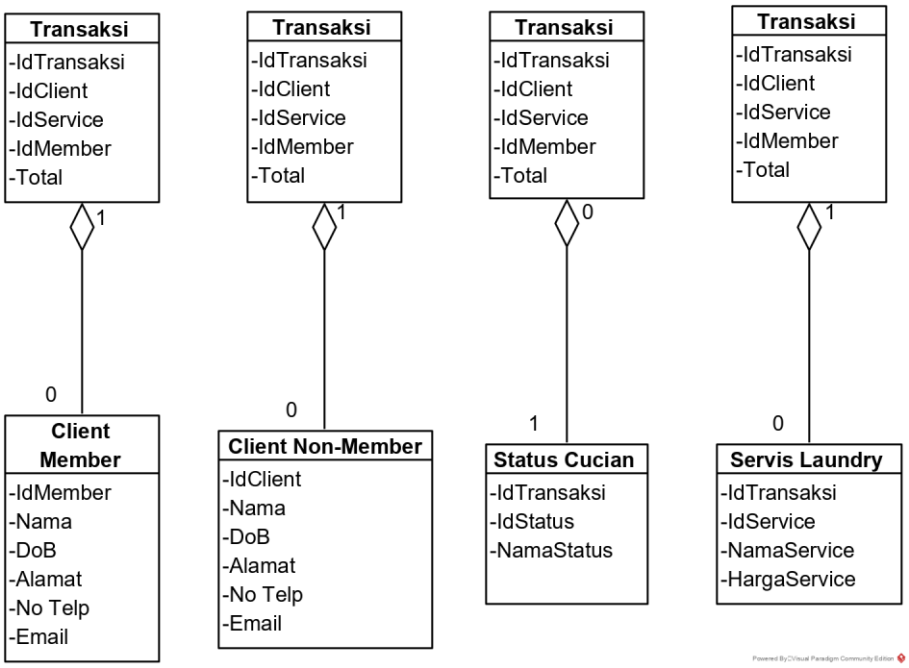
Generalization adalah relasi antara dua atau lebih *specialization class* atau lebih *general class*. *General class* (*super class*) mendeskripsikan properti umum ke kelompok dari *specialization class*. Berikut merupakan generalisasi yang terdapat pada *class diagram Laundry* HiClean:



Gambar 2.2.1. *Generalization structure* sistem *laundry* HiClean

2.2.2 Aggregation

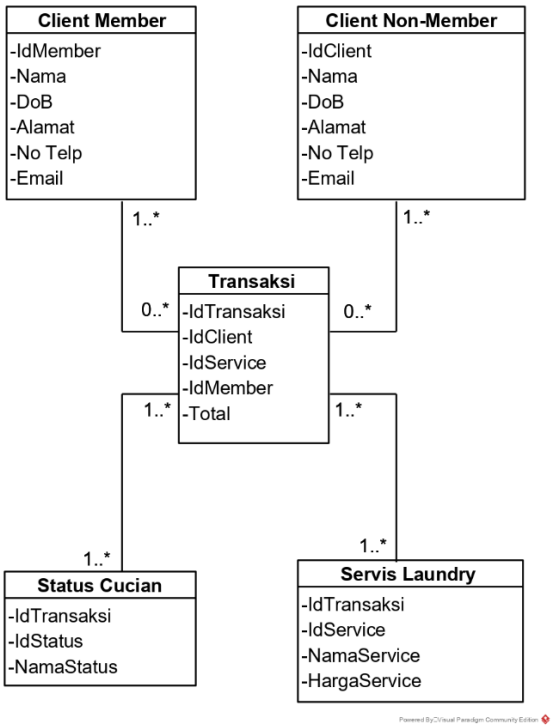
Aggregation adalah relasi antara dua atau lebih objek. Keseluruhan *superior object* terdiri dari sejumlah bagian *inferior object.* Berikut merupakan agregasi yang terdapat pada *class diagram* Laundry HiClean:

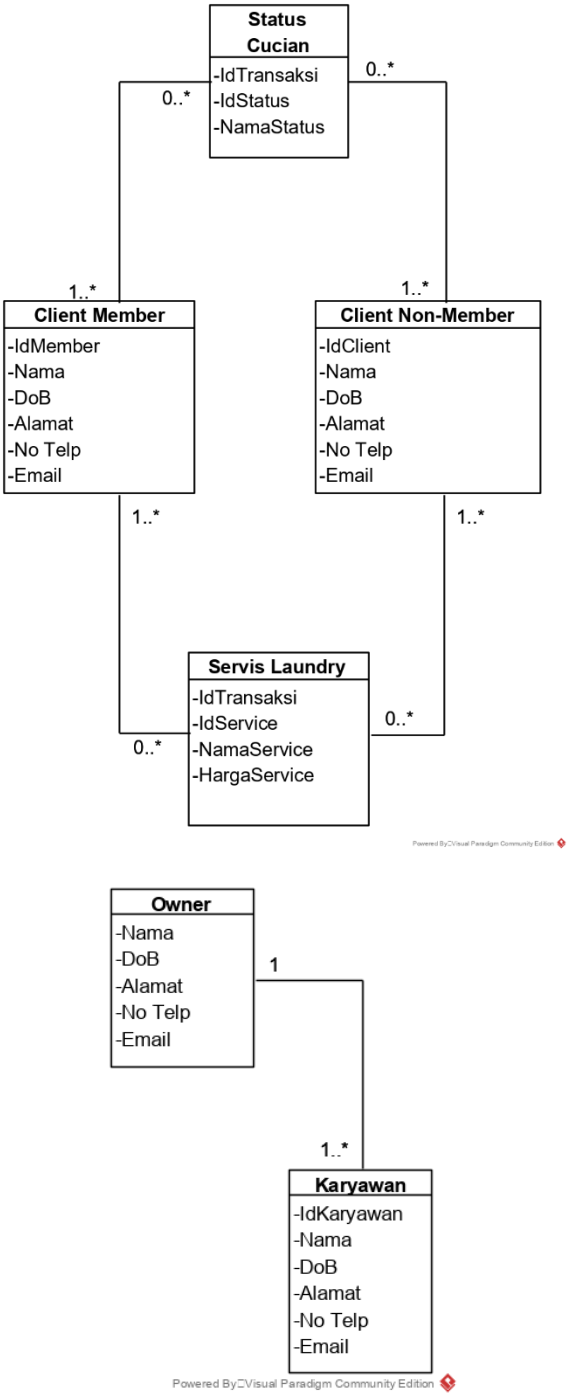


Gambar 2.2.2. *Aggregation structure* sistem *laundry* HiClean

2.2.3 Association

Association adalah relasi antara dua atau lebih objek, tetapi yang membedakan dari *aggregation* adalah objek *association* tidak mendefinisikan properti dari objek. *Association* merupakan relasi dari objek selain *generalization* dan *aggregation*. Berikut merupakan asosiasi yang terdapat pada *class diagram Laundry* HiClean:

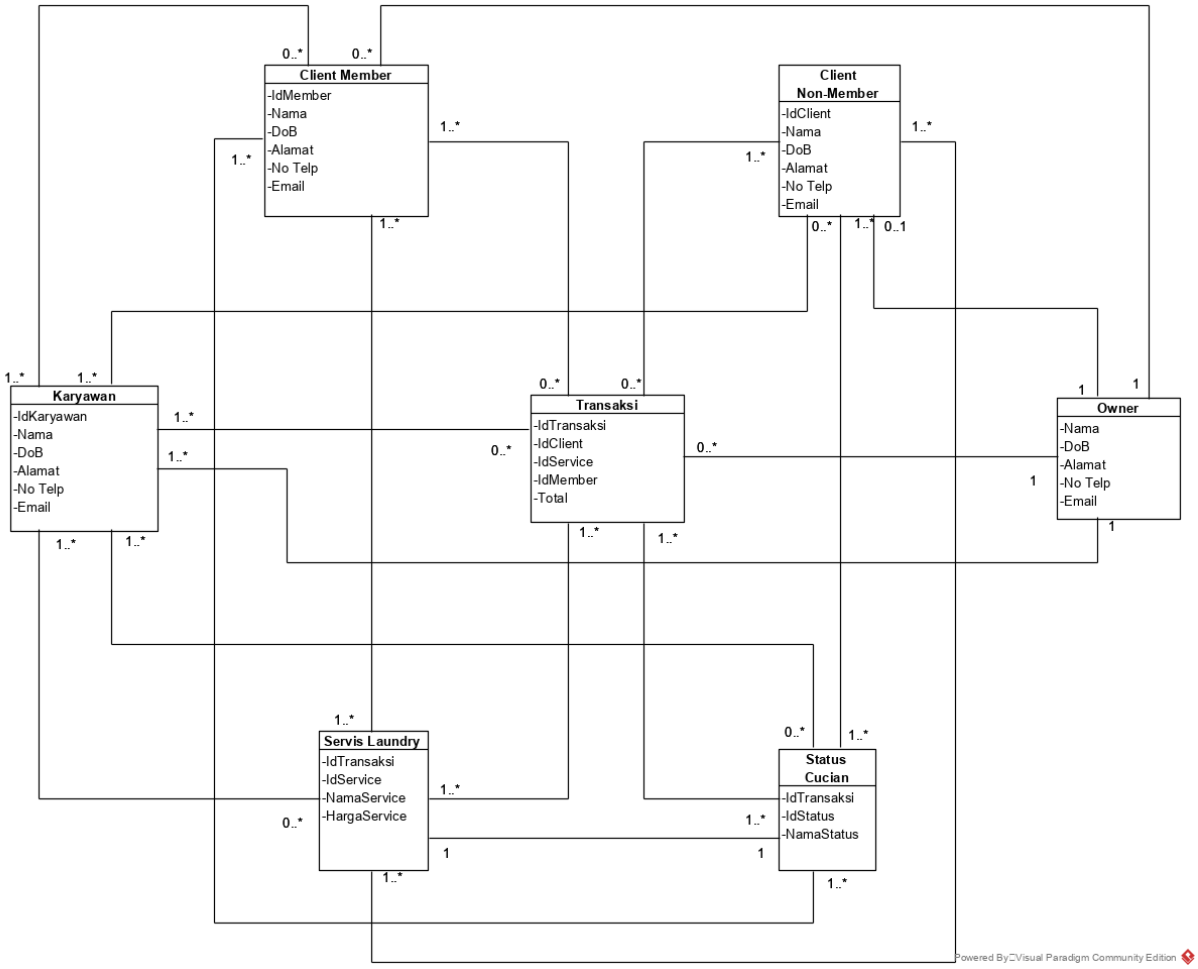




Gambar 2.2.3. *Association structure* sistem *laundry* HiClean

2.2.4 Class Diagram

Berikut class diagram Laundry HiClean yang merupakan rangkaian dari class yang telah dibentuk sebelumnya:



Gambar 2.2.4. *Class diagram* sistem *laundry* HiClean

2.3 Behaviour

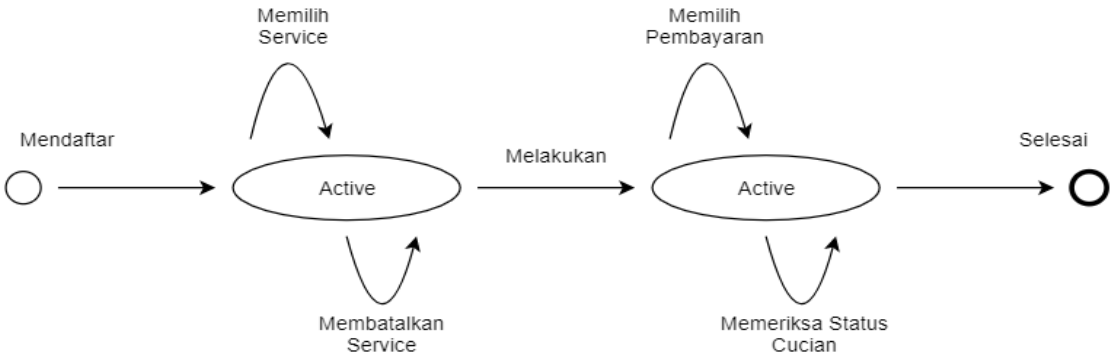
Setelah melihat hubungan antara *class* dengan *event* dalam *event table* secara *generic*, pembuatan *event table* secara *behaviour* diperlukannya mengamati perilaku (behaviour) dari *class* dan *event* yang telah dipilih. Dalam pengamatan tersebut diperlukan mengetahui *behaviour* suatu objek mengenai *event* apa yang dijalankan melalui *statechart*.

2.3.1 Statechart Diagram

Statechart diagram menunjukkan kondisi yang dapat terjadi dari suatu objek dari awal hingga objek tersebut di-*destroy* sehingga setiap objek memiliki diagram status dimana objek-objek dalam *class* memiliki *event-event* yang dapat merubah status. *Statechart diagram* ini direpresentasikan dengan simbol-simbol seperti *initial state*, *final state*, transisi , dan lain-lain.

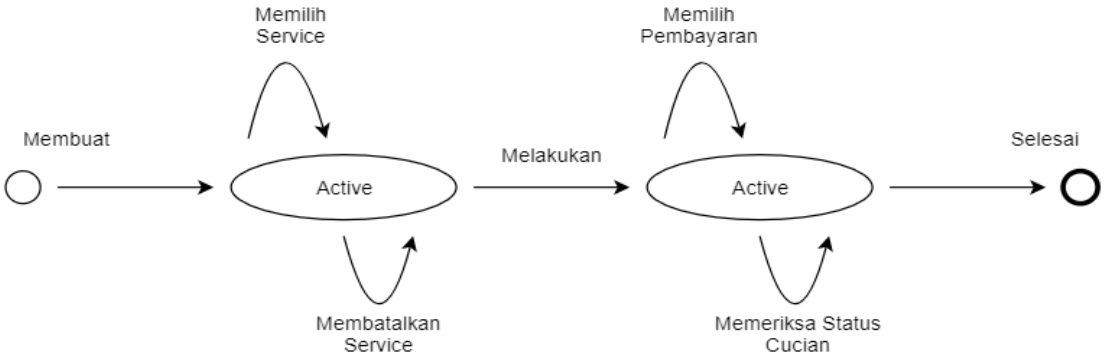
Berikut merupakan *statechart diagram* yang menggambarkan *behaviour pattern* objek-objek yang ada pada sistem *Laundry* HiClean:

1. *Client Non-Member*



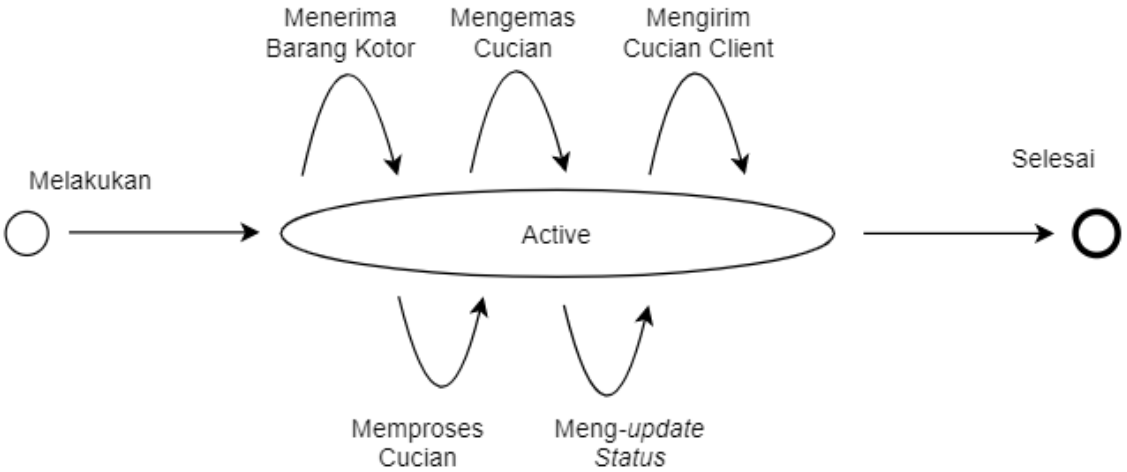
Gambar 2.3.1. *Statechart diagram class client non-member* sistem *laundry* HiClean

2. *Client Member*



Gambar 2.3.2. *Statechart diagram class client member* sistem *laundry* HiClean

3. Karyawan

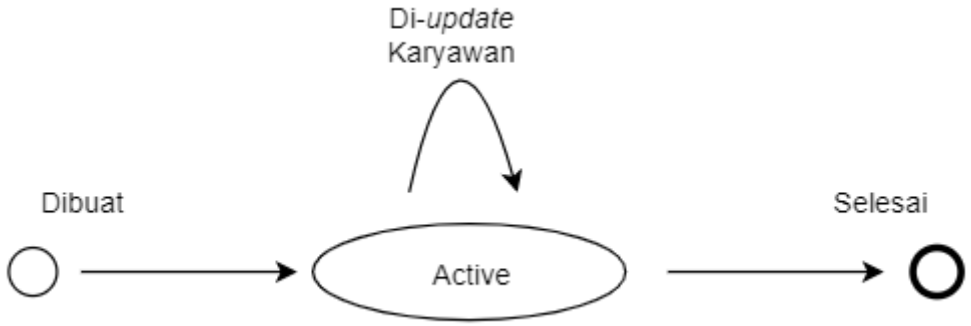


Gambar 2.3.3. *Statechart diagram class* karyawan sistem *laundry* HiClean

4. Transaksi

Gambar 2.3.3. *Statechart diagram class* transaksi sistem *laundry* HiClean

5. Status Cucian

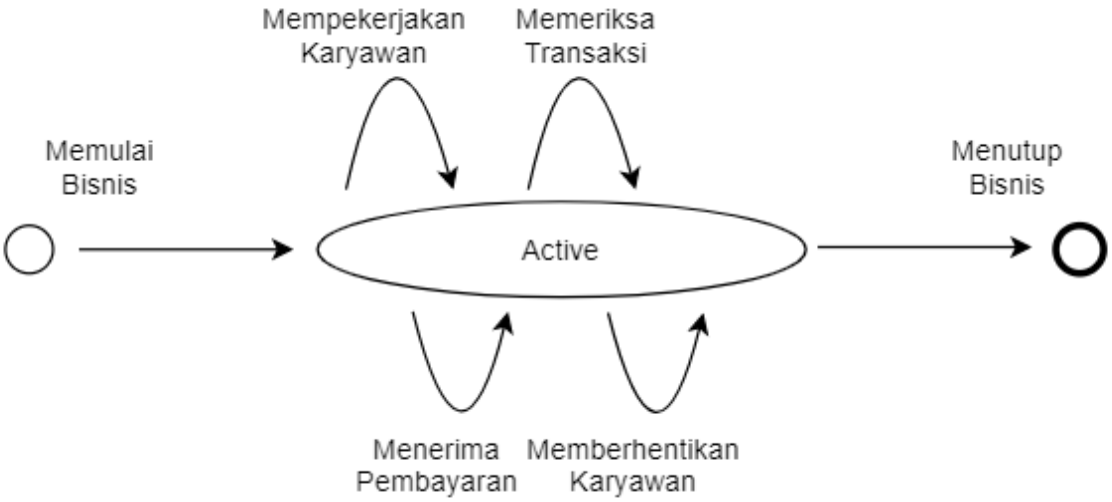


Gambar 2.3.5. *Statechart diagram class* status cucian sistem *laundry* HiClean

6. Servis *Laundry*

Gambar 2.3.6. *Statechart diagram class* servis *laundry* sistem *laundry* HiClean

7. Owner



Gambar 2.3.7. *Statechart diagram class owner* sistem *laundry* HiClean

2.3.2 Behavioral Pattern

Behavioral pattern adalah *pattern* yang mendeskripsikan susunan *event-event* dari suatu objek tertentu. Struktur pengendalian dalam *behavioral pattern* terdiri dari 3 macam :

1. *Sequence* : *event* yang dilakukan oleh suatu objek yang terjadi satu per satu secara berurutan (hanya sekali) dilambangkan dengan simbol "+"
2. *Iteration* : *event* yang dilakukan oleh suatu objek yang terjadi secara berulang kali atau lebih dari satu kali yang dilambangkan dengan simbol " * ".

Tabel 2.3.1. *Behavioral pattern* sistem *laundry* HiClean

	Non-Member	*Member*	Karyawan	Transaksi	Status Cucian	Servis *Laundry*	*Owner*
Mendaftarkan	+						
Membuat		+					
Memeriksa	*	*			*		*
Meng-*update* Status			*				
Memilih	+	+		+		+	
Membatalkan	+	+		+		+	
Membayar	+	+		+			
Mengirim			+				

BAB III

APPLICATION DOMAIN

3.1. Usage

Usage dalam sebuah *application domain* harus mencerminkan bagaimana sistem itu berinteraksi dengan *actor*. *Actor* sendiri merupakan sebuah abstraksi dari pengguna atau sistem yang berinteraksi dengan target suatu sistem.

3.1.1 Use Case

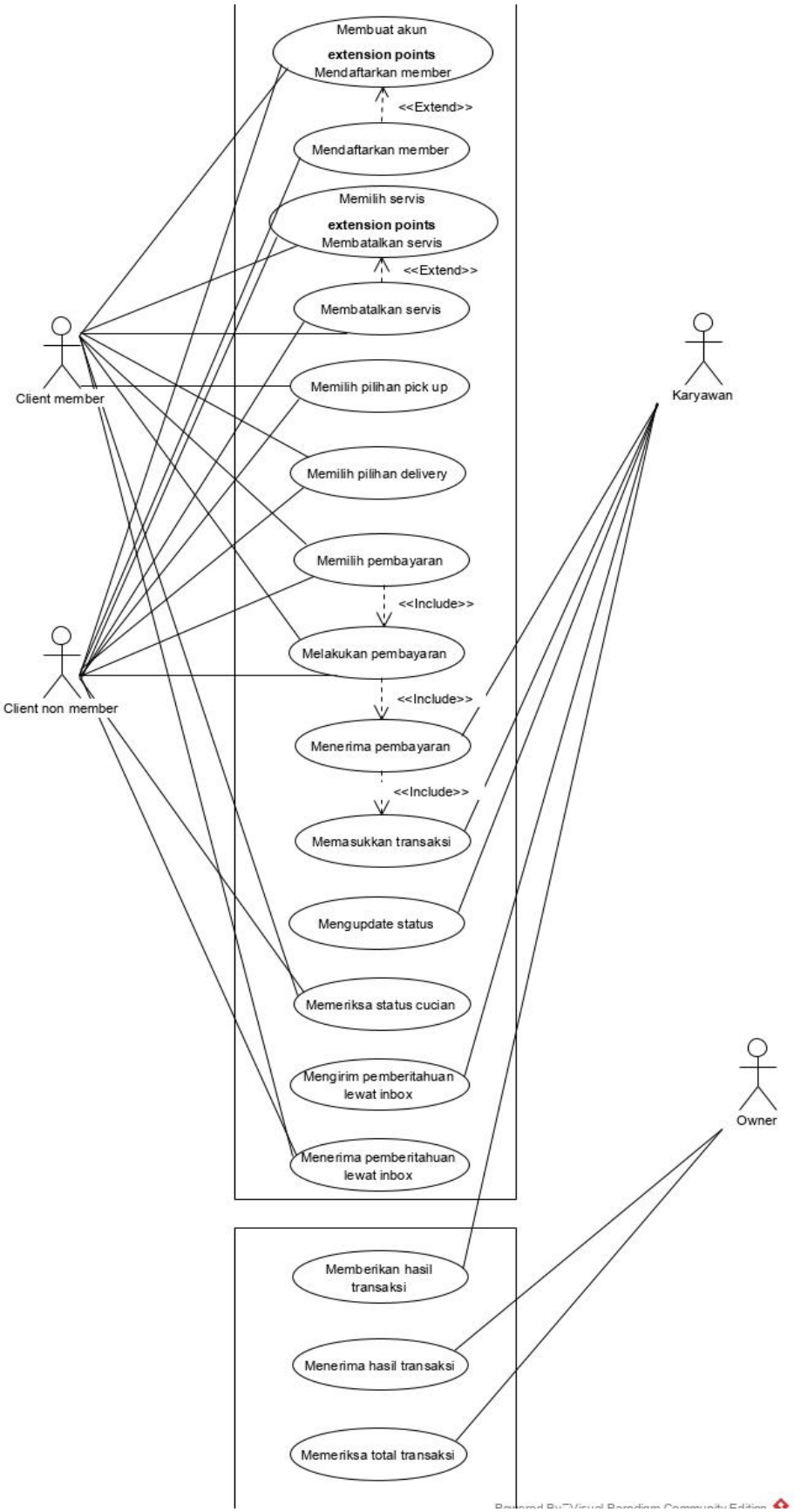
Use Case Diagram merupakan bagian tertinggi dari fungsionalitas yang dimiliki sistem yang akan menggambarkan bagaimana seseorang atau aktor akan menggunakan dan memanfaatkan sistem dalam *application domain*.

Deskripsi *use case* sistem *laundry* HiClean :

Non-Member
Pattern: *Client non-member* memiliki pilihan untuk mendaftarkan sebagai *member* atau tidak, bila tidak maka *client non-member* akan mendapatkan *id client* yang dimana nilainya akan selalu berubah setiap melakukan transaksi. *Client non-member* dapat mendaftarkan *member* dengan *login* ke *mobile*. *Client non-member* dapat memilih servis yang dibutuhkan serta dapat membatalkan bila servis tidak sesuai dengan keinginan. Setelah memilih servis, *client non-member* dapat membayar dengan memilih jenis transaksi dan *client non-member* dapat memeriksa status cucian. *Client non-member* juga dapat memilih apakah pakaian kotor akan di antarkan sendiri ke HiClean atau pakaian kotor akan dijemput oleh karyawan, setelah *laundry* sudah selesai *client non-member* juga dapat memilih apakah ingin mengambil pakaiannya sendiri ke HiClean atau memilih untuk diantar oleh karyawan HiClean.
Objects: IdClient, nama, DoB, Alamat, Telepon, email
Functions : Membuat, mendaftar, memeriksa status, memilih, membatalkan, membayar, mengambil dan mengantar

Member
Pattern: *Client* yang sudah terdaftar menjadi *member* akan mendapatkan *id member* yang dimana nilainya akan sama dalam setiap transaksi, *client member* yang sudah memiliki akun dapat langsung *login* ke aplikasi tanpa harus mendaftar dahulu. *Client member* dapat memilih servis yang dibutuhkan serta dapat membatalkan bila servis tidak sesuai dengan keinginan. Setelah memilih servis, *client member* dapat membayar dengan memilih jenis transaksi dan *client member* dapat memeriksa status cucian. *Client member* juga dapat memilih apakah pakaian kotor akan di antarkan sendiri ke HiClean atau pakaian kotor akan di jemput oleh karyawan, setelah *laundry* sudah selesai *client member* juga dapat memilih apakah ingin mengambil pakaiannya sendiri ke HiClean atau memilih untuk diantar oleh karyawan HiClean.
objects: IdMember, nama, DoB, Alamat, Telepon, Email

Functions: Mendaftar, memeriksa status, memilih, membatalkan, membayar, mengambil dan mengantar



Gambar 3.1.1. *Use case diagram* sistem *laundry* HiClean

3.1.2 Actor

Tabel 3.1.1. *Statechart diagram class* servis *laundry* sistem *laundry* HiClean

USE CASE	***ACTOR***			
	Client member	*Client non-member*	Karyawan	*Owner*
Membuat akun	√	√		
Mendaftarkan *member*		√		
Memilih servis	√	√		
Membatalkan servis	√	√		
Memilih pilihan *pick-up*	√	√		
Memilih pilihan *delivery*	√	√		
Memilih pembayaran	√	√		
Melakukan pembayaran	√	√		
Menerima pembayaran			√	
Memasukkan transaksi			√	
Meng-*update* status			√	
Memeriksa status cucian	√	√		
Mengirim pemberitahuan lewat *inbox*			√	
Menerima pemberitahuan lewat *inbox*	√	√		
Memberikan hasil transaksi			√	
Menerima hasil transaksi				√
Memeriksa hasil transaksi				√

Deskripsi *actor*:

a. ***Client member***

Tujuan: *Client member* adalah seseorang yang melakukan transaksi ke HiClean. Kebutuhan utama *client member* adalah untuk memberikan barang atau pakaian kotor dan melakukan cucian ke *laundry* HiClean.

Karakteristik: Penggunaan sistem antara lain membuat akun, memilih servis, membatalkan servis, memilih pilihan *pick up*, memilih pilihan *delivery*, memilih pembayaran, melakukan pembayaran, memeriksa status cucian, dan menerima pemberitahuan lewat *inbox*.

Contoh: *Client* A mengumpulkan barang atau pakaian kotor yang akan dicuci, *client* A memilih servis sesuai kebutuhannya dan bisa memilih barang atau pakaian kotornya diambil oleh karyawan HiClean atau membawa dan memberikan barang kotornya sendiri ke *laundry* HiClean. Setelah barang kotor *client* A di timbang dan sudah berada di pihak *laundry* HiClean, *client* A akan memilih jenis pembayaran dan melakukan pembayaran pada saat itu juga. Kemudian selama proses pencucian, *client* A dapat memeriksa status cucian melalui *mobile* dan tinggal menunggu pemberitahuan lewat *inbox* dalam *mobile* tersebut.

b. ***Client* non-*member***

Tujuan: *Client* non-*member* adalah seseorang yang akan melakukan transaksi ke HiClean, namun *client* non-*member* dapat mendaftarkan diri menjadi *member* untuk mendapat potongan harga. Kebutuhan utama *client non-member* adalah memberikan barang atau pakaian kotor dan melakukan cucian ke HiClean serta mendaftarkan diri menjadi *member*.

Karakteristik: Penggunaan sistem antara lain membuat akun, mendaftarkan *member*, memilih servis, membatalkan servis, memilih pilihan *pick up*, memilih pilihan *delivery*, memilih pembayaran, melakukan pembayaran, memeriksa status cucian, dan menerima pemberitahuan lewat *inbox*.

Contoh: *Client* A sebagai *non-member* mengumpulkan barang atau pakaian kotor yang akan dicuci, *client* A memilih servis sesuai kebutuhannya dan bisa memilih barang atau pakaian kotornya diambil oleh karyawan HiClean atau membawa dan memberikan barang kotornya sendiri ke *laundry* HiClean. Jika mau mendapatkan potongan harga, *client* A harus mendaftarkan dirinya sebagai *member* melalui *mobile* HiClean dengan syarat dan ketentuan yang sudah dibuat *laundry* HiClean. Setelah barang kotor *client* A ditimbang dan sudah berada dipihak *laundry* HiClean, *client* A akan memilih jenis pembayaran dan melakukan pembayaran pada saat itu juga. Kemudian selama proses pencucian, *client* A dapat memeriksa status cucian melalui *mobile* dan tinggal menunggu pemberitahuan lewat *inbox* dalam *mobile* tersebut.

c. **Karyawan**

Tujuan: Karyawan adalah seseorang yang akan mencuci dan menerima transaksi dari *client*. Karyawan juga berperan dalam memberikan hasil transaksi kepada *owner*. Kebutuhan utama karyawan adalah mencucikan barang kotor yang diberikan *client*, menerima pembayaran, memasukkan transaksi, serta memberikan hasil transaksi kepada *owner*.

Karakteristik: Penggunaan sistem antara lain menerima transaksi, meng-*update* status cucian *client*, memberitahu pemberitahuan lewat *inbox*, serta memberikan hasil transaksi.

Contoh: *Client* A memberikan barang atau pakaian kotor kepada *laundry* HiClean pada tanggal 15 November 2019 kemudian karyawan HiClean menerima transaksi dan mencucikan barang kotor *client* A sampai selesai sesuai waktu yang ditentukan *laundry* HiClean lalu karyawan HiClean akan mencatat dengan memasukkan data transaksi kedalam sistem dan hasil transaksi yang dilakukan *client* A pada 15 November 2019 yang telah dimasukkan dalam sistem akan diberikan kepada *owner*.

d. ***Owner***

Tujuan : *Owner* adalah pemilik dari HiClean. Kebutuhan utama *owner* adalah memeriksa total transaksi yang terjadi pada HiClean.

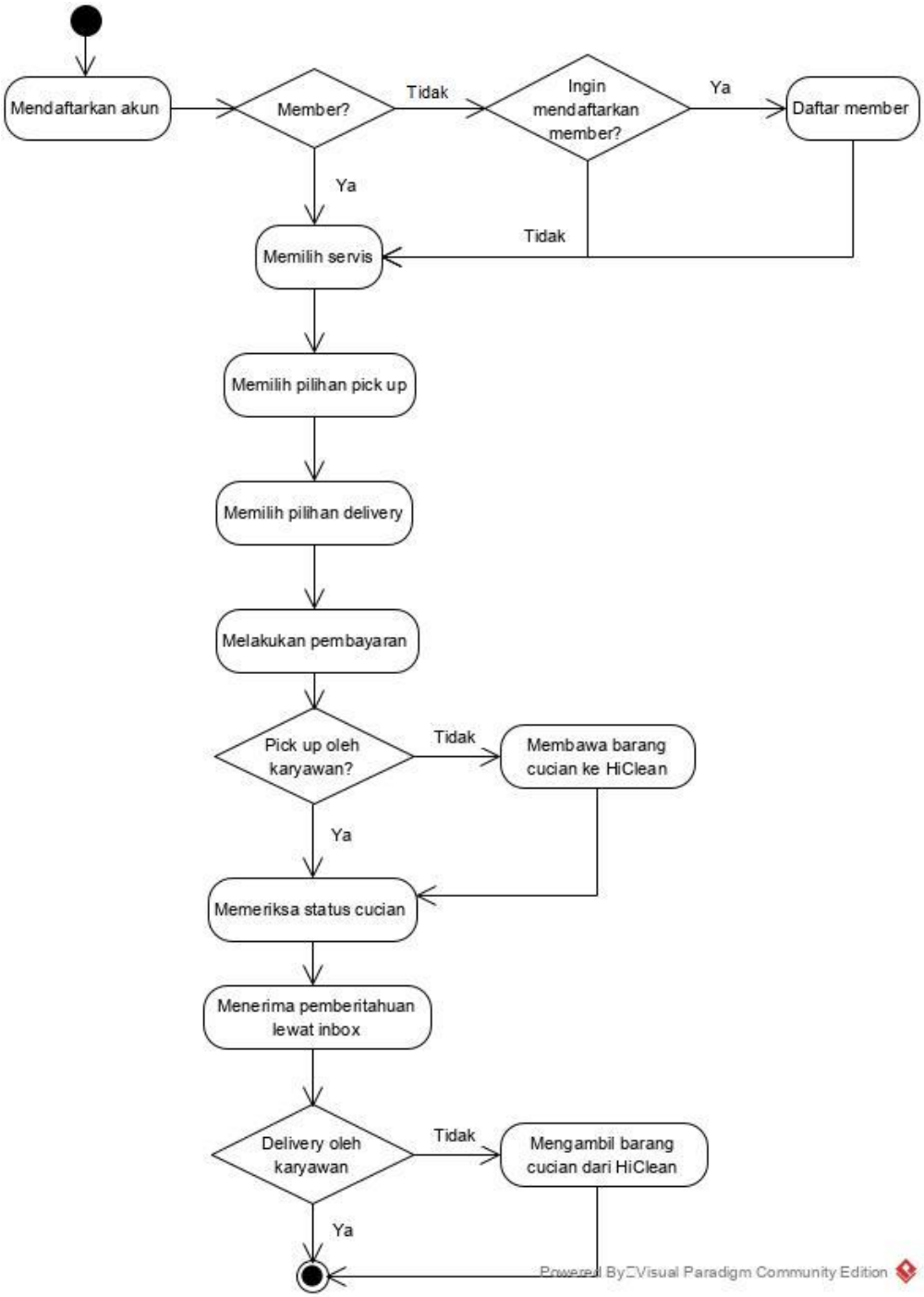
Karakteristik : Penggunaan sistem antara menerima hasil transaksi dan memeriksa total transaksi

Contoh : Pada saat periode tutup buku, *owner laundry* HiClean mendapatkan daftar transaksi yang diberikan oleh karyawan *laundry* HiClean.

3.1.3 Activity Diagram

Activity Diagram adalah diagram yang menggambarkan aliran kontrol dalam sistem yang mengarah ke langkah-langkah pelaksanaan *use case*. Diagram ini menggunakan model aktivitas berurutan dan bersamaan dengan menggunakan *activity diagram*. Maka, alur kerja secara visual digambarkan dengan *activity diagram*. Kemudian diagram ini juga berfokus pada kondisi *flow* dan *sequence* yang terjadi yang mendeskripsikan penyebab *event* tertentu terjadi menggunakan *activity diagram*.

Berikut adalah *activity diagram* dari sistem *Laundry* HiClean:



Gambar 3.1.2. *Activity diagram* sistem *laundry* HiClean

3.2. Function

Function berfokus pada bagaimana sebuah sistem bekerja untuk membantu *actor* dalam kerja mereka. *Function* dianggap sebagai sebuah komputasi, dimana *input data* ditransformasi menjadi *output data*. Intinya, *function* adalah fasilitas untuk membuat sebuah model yang berguna bagi *actor*.

Berikut merupakan *function list* dari *Laundry* HiClean:

Tabel 3.2.1. *Function list* sistem *laundry* HiClean

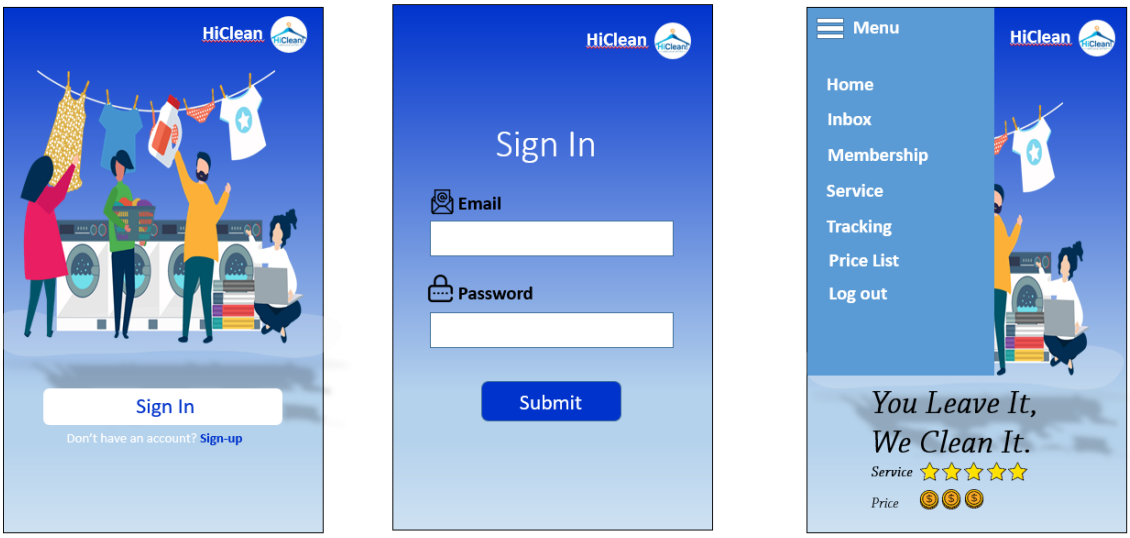
Function	***Complexity***	***Type***
Membuat akun	Simple	Update
Mendaftarkan *member*	Simple	Update
Memilih servis	Complex	Update
Membatalkan servis	Simple	Update
Memilih pilihan *pick-up*	Simple	Update
Memilih pilihan *delivery*	Simple	Update
Memilih pembayaran	Simple	Update
Melakukan pembayaran	Medium	Compute
Menerima pembayaran	Simple	Read
Memasukkan transaksi	Complex	Update
Meng-*update* status	Complex	Update
Memeriksa status cucian	Simple	Read
Mengirim pemberitahuan lewat *inbox*	Complex	Update
Menerima pemberitahuan lewat *inbox*	Simple	Read
Memberikan hasil transaksi	Medium	Update
Menerima hasil transaksi	Simple	Read
Memeriksa hasil transaksi	Simple	Read

3.3 Interface

Interface merupakan mekanisme komunikasi antara pengguna (*user*) dengan sistem. *Interface* dapat menerima informasi dari pengguna (*user*) dan memberikan informasi kepada pengguna (*user*) untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan suatu solusi. Fungsi *interface* untuk meng-*input* pengetahuan baru ke dalam basis pengetahuan sistem pakar (*expert system*), menampilkan penjelasan sistem dan memberikan panduan pemakaian sistem secara *step-by-step* sehingga pengguna mengerti apa yang akan dilakukan terhadap suatu

sistem. Yang paling penting adalah kemudahan dalam menjalankan sistem, interaktif, komunikatif. Tujuan sebuah *interface* adalah mengkomunikasikan fitur-fitur sistem yang tersedia agar *user* mengerti dan dapat menggunakan sistem tersebut. Berikut merupakan *interface* untuk *client* yang digunakan oleh HiClean dalam menjalankan model bisnis sebagai berikut.

Tampilan awal, *sign-in* dan menu untuk *client* sebagai berikut:

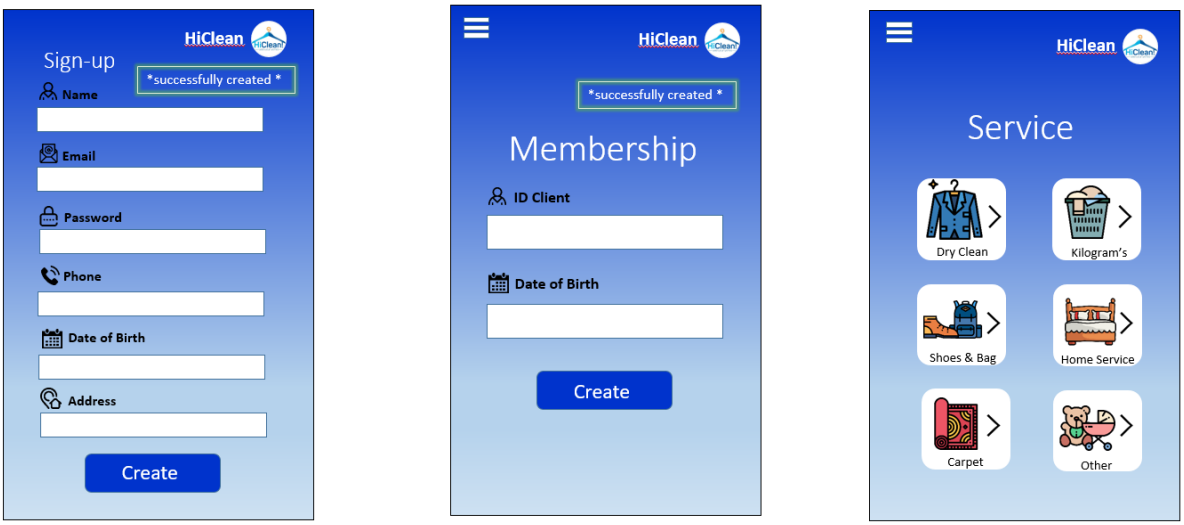


Gambar 3.3.1. *User interface sign-in* dan menu sistem *laundry* HiClean

Pada halaman ini, *client* akan melihat tampilan awal berupa tampilan untuk *sign-in*. Ketika *client* sudah memiliki akun maka *client* bisa langsung memasukkan *e-mail* dan *password* yang sudah didaftarkan. Sedangkan bagi *client* yang belum memiliki akun diharuskan untuk membuat akun terlebih dahulu di menu *sign-up*. Ketika *client* sudah masuk ke dalam aplikasi, *client* akan melihat tampilan beranda dari aplikasi HiClean. Dimana pada tampilan beranda terdapat beberapa menu yaitu, *home*, *inbox*, *membership*, *service*, *tracking*, *price list*, dan *log out*.

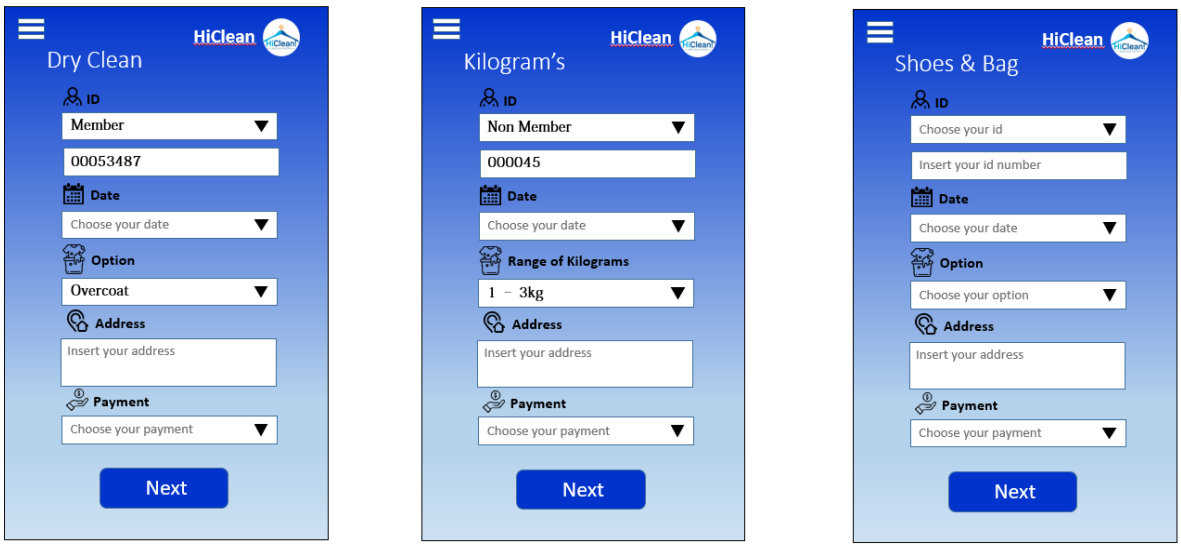
Pada menu *home* berfungsi untuk menampilkan tampilan beranda dari aplikasi HiClean. Menu *inbox* bertujuan untuk menampilkan pemberitahuan mengenai informasi dimulainya prosesnya *laundry* dan ketika *laundry* sudah selesai dengan menggunakan *tracking number*. Pada menu *membership* bertujuan untuk mendaftarkan akun *client* sebagai *member* HiClean. Menu *tracking* bertujuan untuk melacak proses *laundry*. Pada menu *price list* bertujuan untuk menampilkan informasi harga dari setiap pilihan servis yang akan dipilih. Terakhir adalah menu *log out*. Menu *log out* bertujuan untuk keluar dari akun yang telah dibuat.

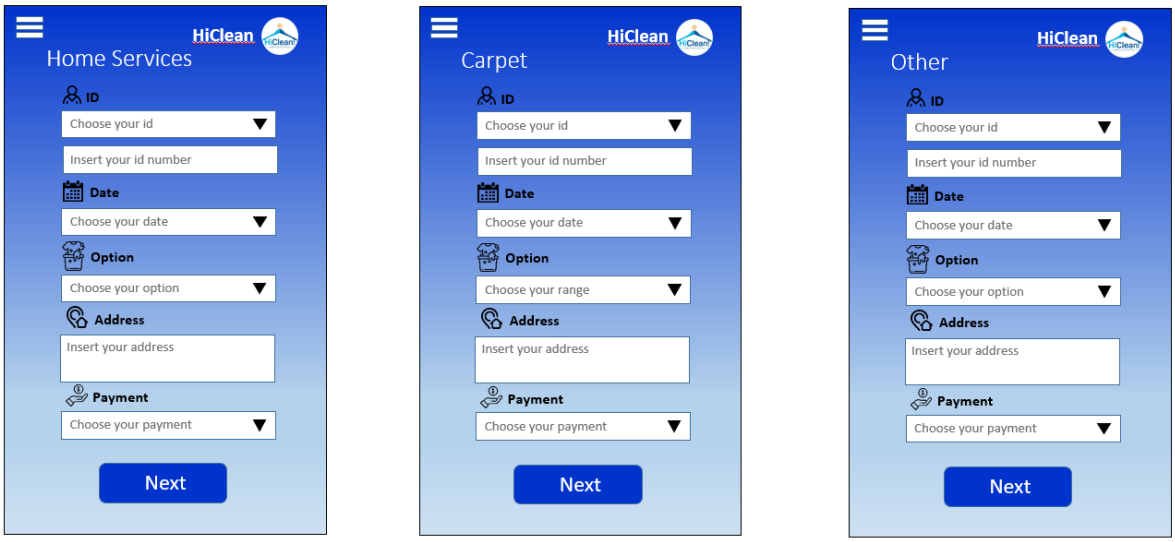
Tampilan menu *sign-up, membership, dan service* untuk *client* adalah sebagai berikut:



Gambar 3.3.2. *User interface sign-up*, *membership*, dan *service* sistem *laundry* HiClean

Pada halaman ini, bagi *client* yang belum memiliki akun dapat langsung menekan *button* menu *sign-up* dan dapat memasukkan data diri. Lalu, *client* dapat mengklik *button create*. Bila muncul *message box* yang bertuliskan kata *"successfully created"*, maka artinya akun sudah berhasil dibuat. Bagi *client* yang ingin mendaftarkan akunnya sebagai *member* dapat memilih menu *membership* pada menu yang telah disediakan. *Client* dapat memasukkan ID *client* yang telah didapatkan saat membuat akun yang dikirimkan melalui *e-mail* dan dapat memasukan tanggal lahir. Setelah itu, *client* dapat menekan *button create*. Bila muncul *message box* yang bertuliskan kata *"successfully created"*, maka artinya *member* telah berhasil didaftarkan. Kemudian, *client* dapat memilih *service* pada menu dan memilih kategori servis yang akan dilakukan. HiClean sendiri memiliki 6 pilihan kategori servis, dimana setiap kategori memiliki pilihan servisnya masing-masing.

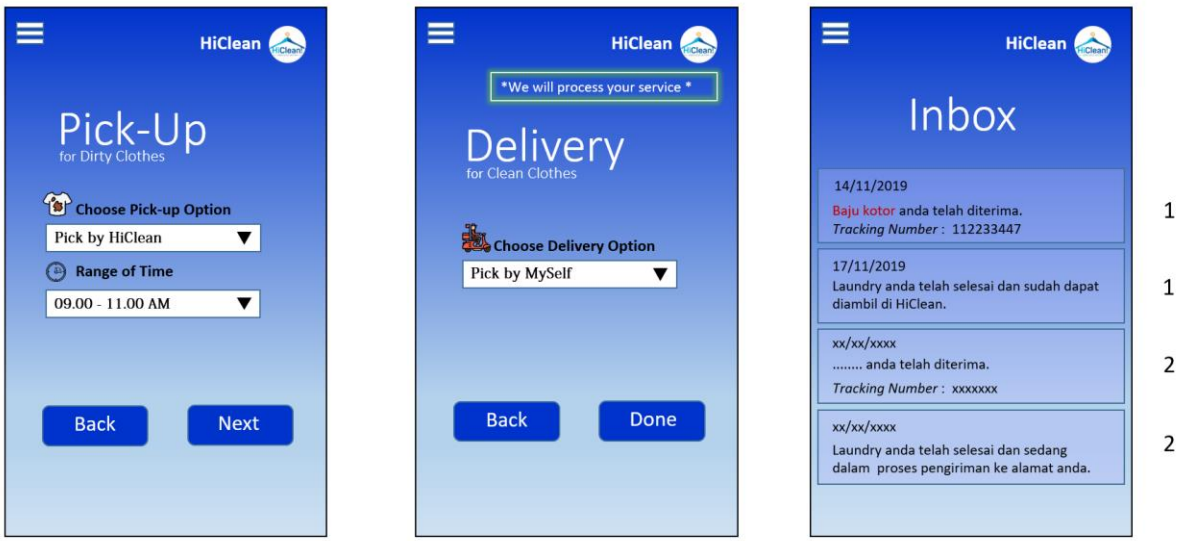




Gambar 3.3.3. *User interface* detail *service* sistem *laundry* HiClean

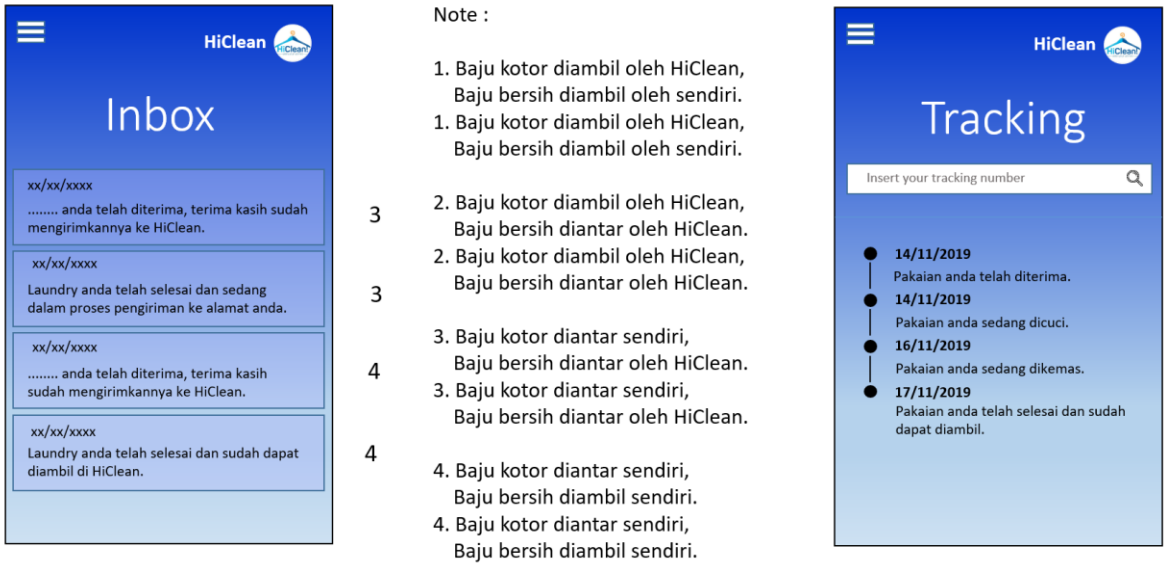
Setelah memilih servis yang akan dilakukan, *client* dapat memilih memasukan ID *member* atau ID *client*. Apabila *client* memilih memasukan *ID member* akan mendapatkan potongan harga. Selanjutnya *client* dapat memasukkan tanggal kapan servis tersebut dilakukan. Lalu, *client* memilih pilihan servis sesuai yang dibutuhkan. Setelah itu, *client* memasukkan alamat dan memilih pilihan pembayaran dimana *invoice*-nya akan dikirimkan melalui *e-mail*. Hal ini berlaku untuk semua kategori servis, namun untuk kategori servis *kilograms*, *client* dapat memasukan perkiraan berat cucian pada *range of kilograms*, dan dapat klik *button next* untuk dialihkan ke halaman selanjutnya.

Tampilan menu *service*, inbox dan tracking untuk *client* adalah sebagai berikut:



Gambar 3.3.4. *User interface service*, *inbox*, dan *tracking* sistem *laundry* HiClean

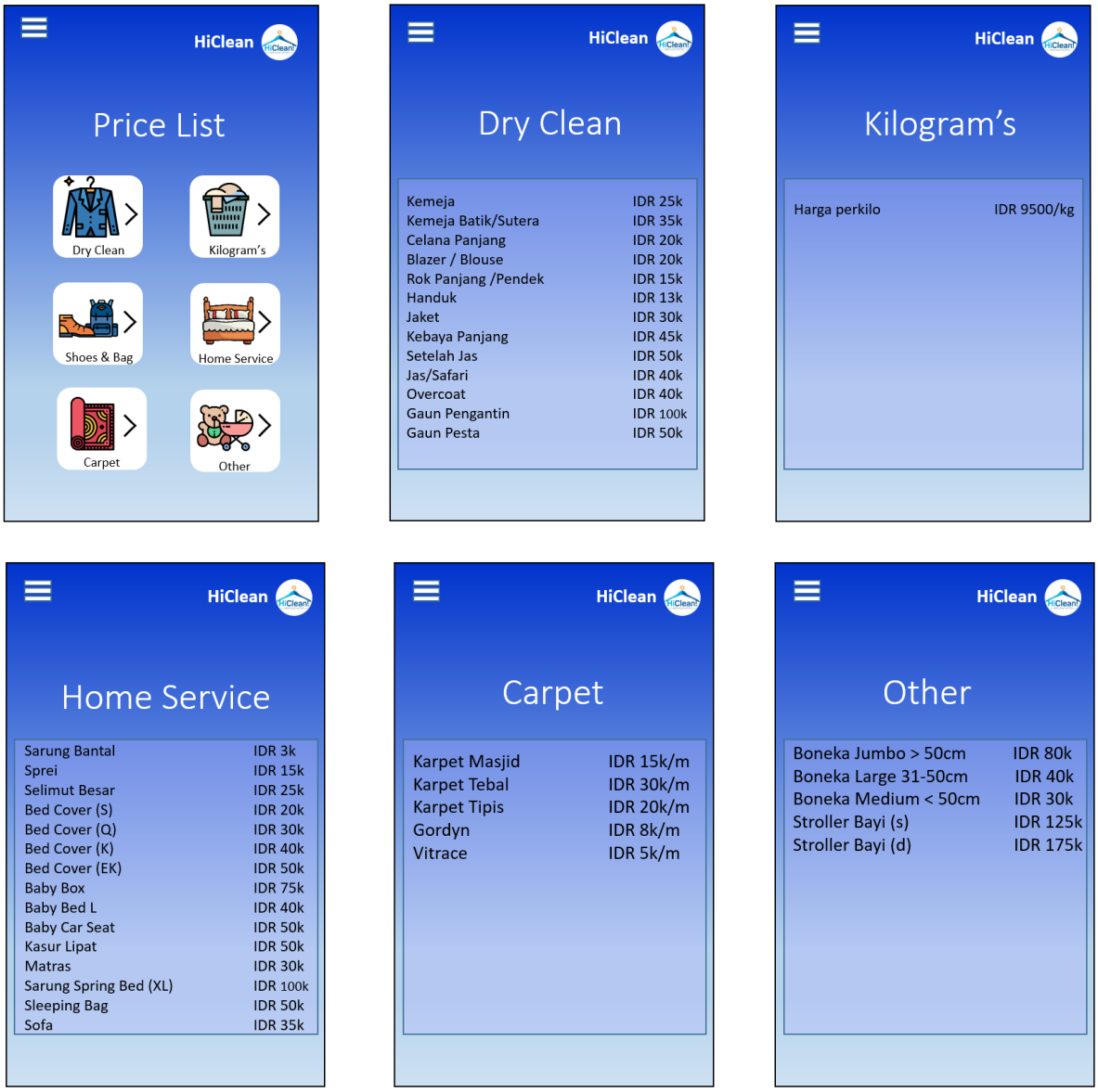
Setelah menekan *button* next, *client* akan dialihkan ke halaman *pick-up*. Menu *pick-up* digunakan untuk memilih *laundry* kotor akan diambil oleh karyawan HiClean atau *client* dapat mengantarkan *laundry* kotornya ke HiClean. Apabila *client* memilih pilihan *Pick by HiClean*, maka *client* harus memilih *range* jam penjemputan *laundry* kotornya lalu klik *button next*. Apabila *client* ingin mengubah servis yang sudah dipilih dapat memilih *button back.* Setelah memilih *button back*, *client* akan dialihkan ke halaman selanjutnya yaitu, *delivery*. *Delivery* digunakan untuk mengantarkan *laundry* yang sudah selesai. Pada menu *delivery* juga terdapat pilihan apakah *laundry* yang sudah selesai ingin diambil sendiri oleh *client* atau *laundry* tersebut dapat diantar oleh karyawan HiClean. Apabila sudah selesai, *client* dapat memilih *button done.* Bila muncul *message box "We will process your service"*, maka servis yang dibuat oleh *client* sudah berhasil dibuat. Setelah servis berhasil dibuat dan *laundry* kotor telah diterima maka akan masuk notifikasi di *inbox* bahwa *laundry* kotor telah diterima dan *client* akan menerima *tracking number* untuk melacak proses *laundry*.



Gambar 3.3.5. *User interface* detail *inbox* dan *tracking* sistem *laundry* HiClean

Setelah *client* mendapatkan *tracking number*, *client* dapat melacak proses cuciannya dengan menekan *button tracking* pada menu. Setelah itu, *client* dapat memasukan *tracking number* pada *search bar* dan menekan *button enter*, lalu akan muncul informasi proses cucian. Karyawan akan meng-*input* informasi proses cucian setiap selesai melakukan satu proses hingga selesai.

Tampilan menu *pricelist* untuk *client* adalah sebagai berikut:



Gambar 3.3.6. *User interface* detail *price list* sistem *laundry* HiClean

Pada halaman *pricelist* berisi harga-harga pilihan servis yang akan dipilih, *client* dapat menekan kategori servis sesuai yang dibutuhkan. Jika *client* ingin mengetahui harganya maka, *client* dapat memilih servis yang disediakan. Selanjutnya, *client* akan dialihkan ke halaman selanjutnya yang berisi harga masing-masing dari servis sesuai kategori servis yang dipilih.

BAB IV
ARCHITECTURAL DESIGN

4.1 Criteria

Kriteria sebuah sistem menjelaskan mengenai sifat dan karakter yang dimiliki sebuah sistem untuk menjalankan fungsi dan tujuan pembuatan sistem tersebut. Kriteria berarti merupakan kumpulan properti yang dipilih untuk sebuah sistem atau desain arsitektur. Penentuan kriteria sistem dengan pendekatan *Object-Oriented Design* dapat membantu untuk menentukan prioritas desain sistem.

Dalam pembuatan sistem untuk *Laundry* HiClean ini, sesuai dengan kebutuhan dan tujuan dari sistem ini yaitu:

Tabel 4.1.1. Kriteria sistem *laundry* HiClean

Criterion	Very Important	Important	Less Important	Irrelevant	Easily Fulfilled
Usable	√				
Secure		√			
Efficient		√			
Correct	√				
Reliable	√				
Maintainable	√				
Testable		√			
Flexible	√				
Comprehensible	√				
Reusable			√		
Portable					√

Interoperable				√	

1. **Very Important**
    a. **Usable**
    Kemampuan sistem untuk menyesuaikan fungsi dengan tujuan dibuatnya sistem ini dengan konteks organisasional dan pekerjaan dalam kegiatan operasional organisasi. *Laundry* HiClean menggunakan fungsi *usable* ini untuk menyesuaikan kegunaan sistem dengan operasional dalam penggunaan oleh *client* untuk melakukan servis *laundry* dan administrasi dalam pencatatan transaksi oleh karyawan dan pengecekan status transaksi oleh *client* dan *owner*. *Usage* sangat penting dalam properti untuk sistem *Laundry* HiClean ini, sesuai dengan penggunaan dan kebutuhan *client*, karyawan, dan *owner* HiClean.
    b. **Correct**
    Kriteria *correct* berarti sistem yang dibuat harus dapat menjalankan fungsi dan kebutuhan yang telah ditentukan oleh pembuat sistem dengan benar. Proses yang perlu dieksekusi secara benar yaitu dalam pemberian status transaksi *laundry* yang dilakukan oleh *client* sesuai dengan keadaan yang nyata dari pencatatan detail transaksi yang dilakukan oleh karyawan.
    c. **Reliable**
    Sistem *laundry* HiClean perlu melakukan fungsi yang diperlukan dalam operasional dengan tepat. Dengan ini, maka pengguna sistem mendapatkan rasa kepercayaan pada sistem yang digunakan untuk melakukan transaksi dengan HiClean. Kriteria ini sangat penting dalam penggunaan sistem *laundry* HiClean.
    d. **Maintainable**
    Biaya yang dibutuhkan dalam penempatan dan perbaikan sistem yang bermasalah atau rusak sangat penting dalam menunjang sistem ini. Apabila adanya kendala yang terjadi dalam sistem *laundry* HiClean, dibutuhkan perbaikan dalam waktu dekat agar tidak menimbulkan masalah berlanjut. Sistem ini digunakan setiap transaksi dalam *laundry* HiClean sehingga membutuhkan kriteria ini.
    e. **Flexible**
    Biaya yang diperlukan untuk memodifikasi sistem yang dibuat merupakan kriteria yang sangat penting karena dalam modifikasi sistem dapat dilakukan dengan mudah dan cepat jika diperlukan perbaikan dan modifikasi sistem.
    f. **Comprehensible**
    Pengguna sistem HiClean harus dapat memahami sistem yang ada dan kegunaan dari setiap fungsi yang terdapat dalam sistem. Sangat penting bagi seluruh pengguna untuk memahami sistem agar dapat melakukan transaksi dengan mudah tanpa perlu banyak mencari tahu kegunaan dari fungsi yang disediakan oleh sistem.

2. **Important**
    a. **Secure**
        Keamanan dari sistem *laundry* HiClean penting untuk menyimpan daftar transaksi beserta *detail*-nya seluruh transaksi yang terjadi di HiClean. *Owner* HiClean dapat mengecek seluruh transaksi yang terjadi pada bisnisnya dan memonitor pendapatan HiClean.
    b. **Efficient**
        Dalam perancangan sistem HiClean, harus dibuat sebuah sistem yang dapat menjalankan fungsinya. Tidak membutuhkan *hardware* dengan spesifikasi secara khusus untuk menjalankan sistem ini. Dalam mengoperasikan sistem *laundry* HiClean ini biaya teknisnya tidak mahal.
    c. **Testable**
        Biaya yang memberikan kepastian bahwa sistem yang digunakan pada setiap cabang dapat melaksanakan fungsi-fungsinya dalam sistem sesuai dengan yang dibutuhkan.
3. **Less Important**
    a. **Reusable**
        Kemampuan untuk sistem atau bagian-bagian dalam sistem dapat digunakan oleh sistem lain kurang penting dalam sistem *laundry* HiClean ini. Sistem yang dibuat ini adalah khusus digunakan oleh *Laundry* HiClean.
4. **Irrelevant**
    a. **Interoperable**
        Kemampuan dalam menyatukan sistem internal dengan sistem lain untuk menjalankannya. *Laundry* HiClean menjalankan sistemnya tanpa membutuhkan sistem lain dari sistem utama, tidak ada penyatuan sistem dan tidak adanya sistem lain sebagai pendukung. Sehingga hanya menggunakan sistem utama yang dibuat secara keseluruhan.
5. **Easily Fulfilled**
    a. **Portable**
        Sistem ini dibuat dengan syarat teknis yang umum dimana untuk menjalankan sistem ini, sehingga untuk memindahkan sistem ini tidak memerlukan biaya yang besar. Sistem dapat digunakan melalui akses *internet* dengan membuka *mobile* dari sistem *laundry* HiClean yang sudah dibuat ini.

4.2 Component Architecture

Arsitektur komponen merupakan sebuah struktur sistem yang terdiri dari komponen-komponen yang saling terhubung satu sama lain dalam sebuah sistem. Komponen dalam sistem ini sendiri merupakan sebuah koleksi yang terdiri dari bagian program yang berisi tugas atau peran yang sudah ditentukan oleh pembuat sistem.

Tujuan dari pembuatan arsitektur komponen ini yaitu untuk menciptakan sebuah sistem yang terstruktur serta menjadi sistem memiliki kriteria *comprehensible* dan *flexible*. Kriteria ini sudah ditentukan oleh pembuat sistem sesuai dengan prioritas kriteria sistem yang ingin dibuat.

Prinsip yang digunakan dalam menentukan arsitektur komponen ini yaitu:

a. Mengurangi kerumitan atau kompleksitas dari sistem dengan memisahkan fungsi-fungsi yang digunakan oleh sistem. Memisahkan fungsi-fungsi ke dalam komponen yang berbeda dapat meningkatkan kelengkapan komponen dalam sistem yang dibuat. Kelengkapan komponen ini dapat mengeksekusi fungsi yang diperlukan atau masalah yang perlu diperbaiki atau diatasi dengan baik. Saat analisa sistem, pembuatan *problem domain* dan *application domain* digunakan untuk merepresentasikan karakteristik struktur dari sistem yang stabil yang merupakan penggambaran dari prinsip ini.
b. Komponen arsitektur harus dapat merefleksikan struktur keadaan sistem yang stabil. Prinsip ini berkaitan dengan prinsip pemisahan komponen di atas. Arsitektur komponen yang dibuat harus memberikan kemampuan untuk sistem agar fleksibel meskipun tidak dapat melakukan semua perubahan atau modifikasi secara keseluruhan. Dalam banyak kasus, *application domain* lebih banyak berubah dari *problem domain*. Salah satu komponen dari *application domain* yaitu *user interface*, dimana *user interface* ini menentukan komponen yang digunakan oleh arsitektur sistem. Menggunakan *architectural patterns* merupakan pelaksanaan prinsip ini.
c. Menggunakan komponen yang sudah ada. Saat membuat pembagian-pembagian sistem dalam *system requirement*, yaitu: model, fungsi, *user interface*, dan *system interface*. Pemisahan komponen sistem ini digunakan sebagai arsitektur komponen yang umum dan sederhana. Kemudian dari arsitektur sistem dasar ini, dapat dirancang komponen yang lebih banyak. Konsep ini merupakan konsep *reuse*.

Pola yang dapat digunakan untuk membuat rancangan arsitektur komponen berupa tiga pola arsitektur. Dekomposisi sistem menjadi model, fungsi, *user interface*, dan *system interface* dapat dilakukan dengan menggunakan pola arsitektur yang digunakan untuk mendukung proses dekomposisi tersebut. Pola arsitektur tersebut yaitu:

a. Layered Architecture Pattern

   Pola *layered architecture* digunakan pada banyak *software*. Contoh dari penggunaan pola ini yaitu pada *Open Systems Interconnection* (OSI) yang merupakan sebuah standar yang dibuat oleh *International Organization of Standardization* (ISO). Standar OSI ini terdiri dari tujuh *layer* jaringan dari level terendah (berupa sinyal elektronik saling bertemu secara fisik) hingga level tertinggi (*user* memanfaatkan *software* untuk menggunakan fasilitas jaringan). Dalam standar OSI, setiap *layer* mempunyai tujuan yang sudah ditentukan dan memiliki *interface* yang jelas dengan *layer* di atas dan di bawahnya.

   Rancangan setiap komponen menjelaskan tugas dan *interface* ke atas dan ke bawah dari komponen tersebut (*upward* dan *downward interface*). *Downward interface* menjelaskan operasi yang dapat diakses komponen pada layer di bawahnya. Sementara *upward interface* menjelaskan operasi yang tersedia pada suatu komponen ke *layer* di atasnya. Secara detail, desain internal

dari setiap komponen harus memenuhi syarat *interface* tersebut, sementara desain lainnya disesuaikan oleh kebutuhan atas tugas yang ingin diberikan ke komponen.

Layered architecture ini sangat bermanfaat untuk melakukan dekomposisi atau penguraian sebuah sistem ke dalam komponen-komponen. Pada tiap tingkat, dekomposisi dapat dilakukan secara vertikal untuk membuat *layer* baru dalam bentuk sub-komponen. Dekomposisi dapat juga dilakukan secara horizontal dengan menciptakan beberapa bagian dari *layer* dalam bentuk sub-komponen baru. Dekomposisi ini dapat digunakan untuk menjabarkan *layer* ke dalam *layer-layer* yang lebih kecil, dan beberapa bagian ke dalam sebuah *layer* atau bagian baru kembali.

b. Generic Architecture Pattern

Dalam *generic architecture*, komponen dalam sistem dibagi ke dalam beberapa kategori komponen, yaitu komponen *interface*, fungsi, dan model. Komponen model menjadi *layer* terbawah, kemudian di atasnya terdapat komponen fungsi, dan yang diletakkan paling atas adalah komponen *interface*.

Banyak dilakukan penggabungan untuk beberapa komponen yang mempunyai *interface* terdefinisi ke dalam beberapa bagian platform teknis. Dengan arsitektur dasar, dapat digunakan untuk membuat sebuah *interface* untuk *library* yang mengimplementasikan *user interface*. *Library* tersebut dapat dienkapsulasi dalam sebuah *user-interface system component* (UIS) yang terpisah. Pemisahan ini juga dapat dilakukan pada *database system* (DBS) yang menyimpan isi dari model-model, serta dengan *network software* (NS) yang mengimplementasikan *systems interface*.

c. Client-Server Architecture Pattern

Client-server architecture pada awalnya diciptakan untuk melakukan distribusi sebuah sistem dalam beberapa prosesor yang tersebar secara geografis. Pola ini banyak digunakan dalam industri *software*. Alasan dari penggunaan pola ini yaitu sebagai solusi untuk menghubungkan banyak *personal computer* (PC) dengan komputer *mainframe* utama.

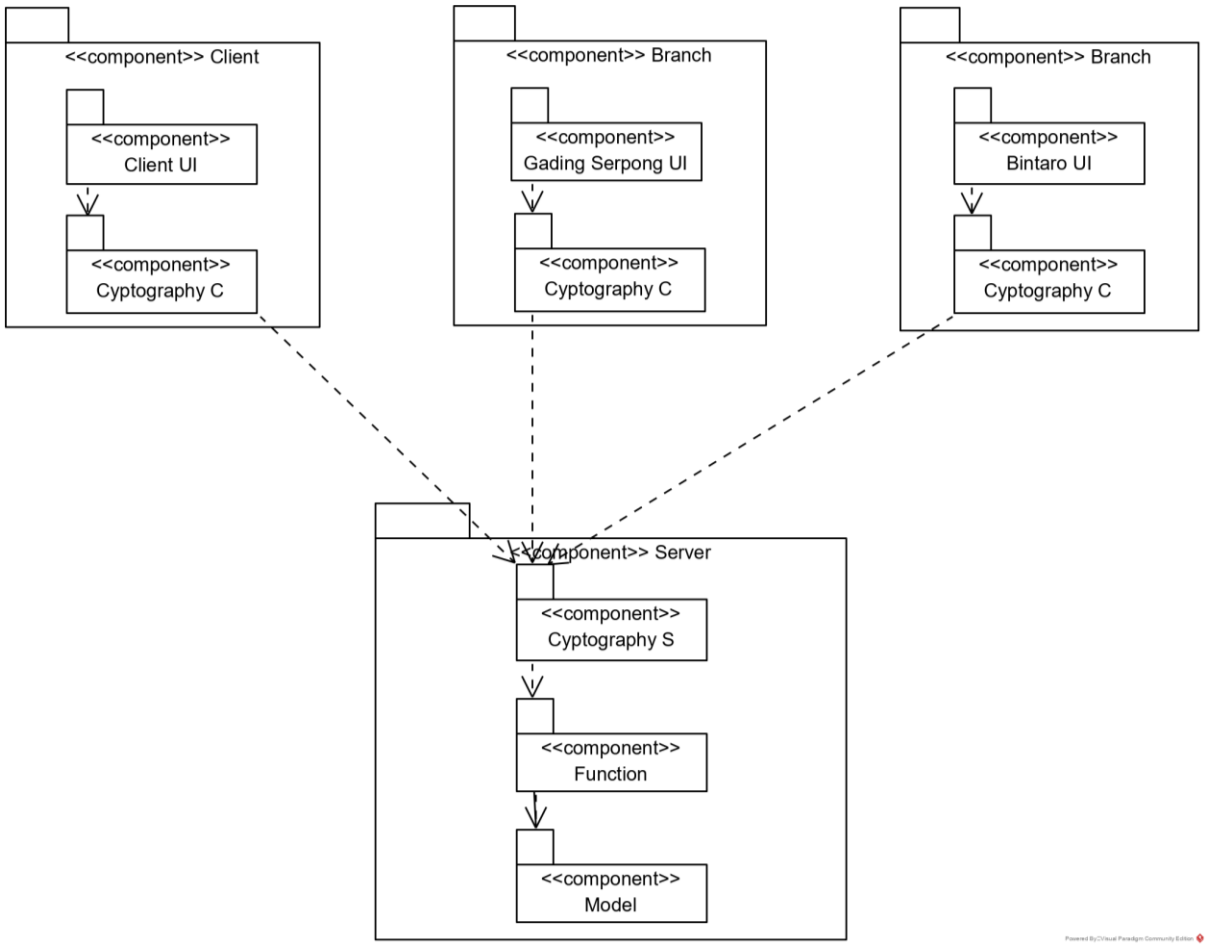
Komponen dari arsitektur *client-server* terdiri dari sebuah *server* dan beberapa *client*. *Server* bertugas mengerjakan sekumpulan operasi yang bisa dilakukan oleh *client*. Rancangan arsitektur ini berbentuk asimetris dimana *server* tidak mengambil informasi dari *client*, namun *client* harus memahami *server* dan operasi-operasi publiknya. Fokus utama *client-server architecture* yaitu *client* menggunakan *server* secara mandiri satu sama lain, dan tidak harus menjadi bagian dari jaringan secara bersamaan.

Operasi publik dalam server diberikan kepada *client* melalui *database* atau sumber lain yang digunakan bersama. *Server* biasanya menyediakan operasi kepada *client* melalui sebuah jaringan. Masing-masing *client* perlu memiliki *interface* lokal yang digunakan untuk mengakses *server* yang dibutuhkan. Pola ini menyediakan ekspresi untuk *network thinking*. Selain itu, juga *client-server* dapat dilihat sebagai struktur 2 *layer*, dimana *client* sebagai *top layer* dan *server* sebagai *bottom layer*.

Dalam sistem *Laundry* HiClean, komponen sistem yang digunakan yaitu *Client-Server Architecture*. Sistem *laundry* ini dipilih menggunakan *Client-Server Architecture* karena beberapa alasan, yaitu:

1. Sistem *laundry* HiClean adalah sistem berbasis *mobile*, dimana *client* dan karyawan membuka *mobile* servis *laundry* HiClean menggunakan perangkat pribadinya.
2. Sistem laundry HiClean membutuhkan sebuah *server* yang menyimpan dan menyediakan data dari seluruh transaksi dan data dari *client* serta karyawan *laundry* HiClean.

   Berikut adalah diagram *component architecture* dari sistem *Laundry* HiClean:



Gambar 4.2.1. *Component architecture* sistem *laundry* HiClean

Dalam menggunakan *client-server architecture* dapat dijabarkan menjadi bentuk-bentuk yaitu :

Tabel 4.2.1. Bentuk distribusi *client-server architecture*

Client	*Server*	*Architecture*
U	U+F+M	*Distributed Presentation*
U	F+M	*Local Presentation*
U + F	F+M	*Distributed Fuctionality*
U + F	M	*Centralized Data*
U + F + M	M	*Distributed Data*

4.3 Process Architecture

Process architecture berfokus pada distribusi dan eksekusi, serta bekerja dengan proses dan objek sebagai lawan pada komponen dan *class*. *Process architecture* juga berurusan dengan perangkat fisik dimana sistemnya akan mengeksekusikan dan mempertimbangkan apakah kita perlu untuk mengkoordinasikan sumber daya bersama atau tidak. Aktivitas proses terstruktur melalui dua level abstraksi. Pertama adalah *overall level*, dimana kita mendeskripsikan distribusi dari komponen program pada prosesor sistem yang tersedia. Sedangkan yang kedua berkaitan dengan proses dengan kolaborasi struktur antara kehadiran objek selama eksekusi.

Pada *process architecture* terdapat tiga *architecture*, yakni *centralized pattern, distributed pattern,* dan *decentralized pattern.*

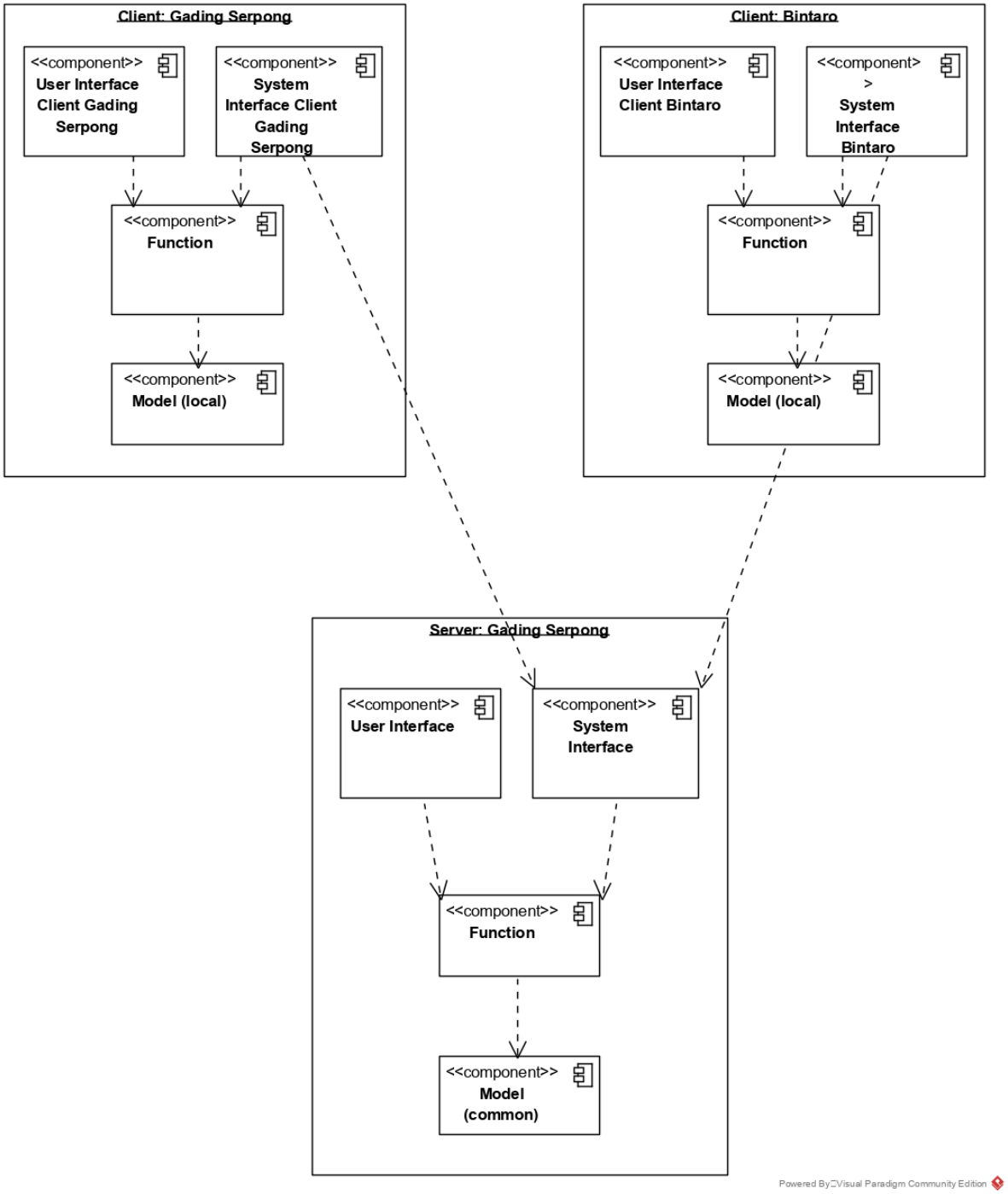
a. *Centralized Pattern*
   Centralized pattern menyimpan seluruh data ke dalam *server* sentral dan *client* hanya bisa melihat *user interface* saja. Keuntungan dari *architecture* ini adalah dapat diimplementasikan kepada *client* secara murah. Sedangkan kerugian dari *architecture* ini adalah ketahanan tingkat rendah, dimana ketika *server* atau jaringannya sedang *down*, maka *client* tidak bisa melakukan apa-apa. Selain itu, kerugian lainnya adalah waktu akses akan relatif tinggi karena mengaktifkan fungsi *client* apapun melibatkan perubahan pada *server*.
b. *Distributed Pattern*
   Distributed pattern merepresentasikan desain ideal yang berlawanan dibandingkan dengan *centralized pattern*. Disini semuanya sudah didistribusikan kepada *client*, dan *server* hanya menyebarkan pembaharuan model antara *client*. Keuntungan dari *architecture* ini adalah waktu akses yang

rendah, karena *function* dan modelnya berada pada *local client* sehingga tidak terjadi *network traffic*. Namun kerugian dari *architecture* ini adalah banyaknya redudansi data, sehingga potensi untuk ketidakkonsistenan antara data terancam. *Network traffic* juga tinggi karena pembaharuan pada setiap *client* akan disebarkan kepada *client* lain. Kebutuhan teknis pada *client* juga meningkat karena mereka harus menjalankan model, *function*, dan *interface*.

c. *Decentralized Pattern*

Decentralized pattern berada diantara dua *pattern* sebelumnya. *Client* memiliki datanya sendiri, sehingga hanya data umum pada *client* yang berada dalam *server*. Sehingga, *structural design* pada *client* dan *server* itu sama. Keuntungan dari *decentralized pattern* adalah konsistensi data karena tidak ada duplikasi antara *client* atau antara *client* dan *server*. Pemuatan jaringan juga rendah karena jaringannya hanya digunakan ketika data umum dalam *server* digunakan. Sedangkan kerugian pada *decentralized pattern* adalah setiap prosesor harus mampu untuk mengeksekusikan *function* yang kompleks dan memelihara model dalam jumlah yang besar.

Pada HiClean digunakan *process architecture* dengan pola *Decentralized Pattern* karena HiClean memiliki satu *server* dan banyak *client* dimana HiClean sendiri sudah memiliki 2 cabang, yakni di Gading Serpong dan Bintaro. Setiap cabang memiliki *database* masing-masing yang terdiri dari data *client*, karyawan, transaksi, dan lain-lain yang terhubung ke dalam satu *server* pusat. Berikut adalah *deployment diagram* dengan *decentralized pattern* pada sistem *Laundry* HiClean:



Gambar 4.3.1. *Process architecture* sistem *laundry* HiClean

BAB V

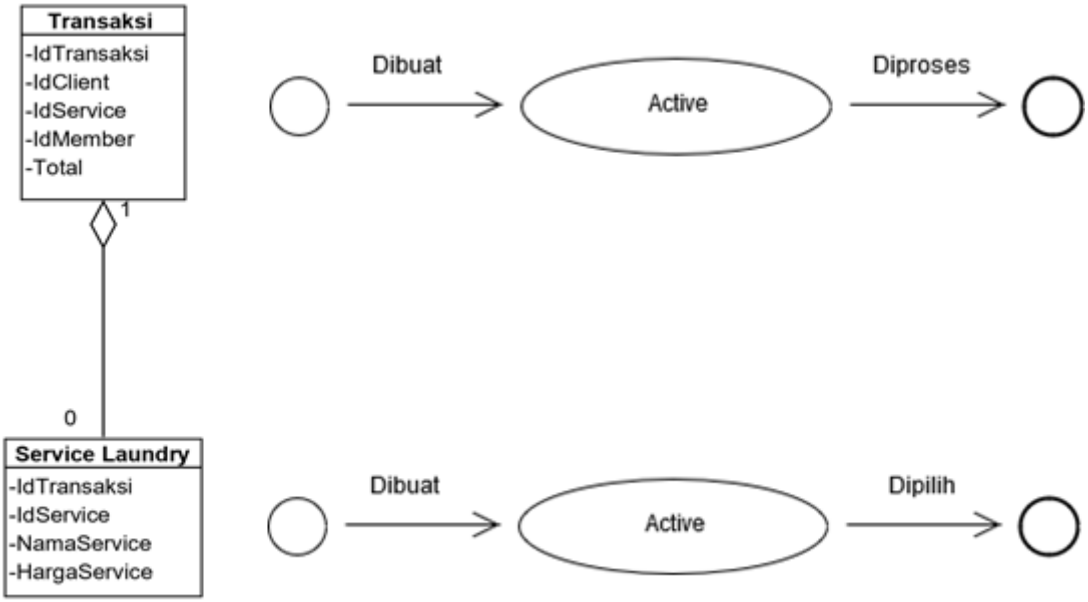
COMPONENT DESIGN

Component architecture adalah sistem struktur yang tersusun atas komponen yang saling berhubungan. *Component* adalah sekumpulan dari bagian-bagian dari keseluruhan *program* yang terdiri dari tanggung jawab yang terdefinisi dengan baik. *Component design* memiliki tujuan untuk memberikan penjabaran terhadap apa yang dibutuhkan dalam *architectural framework*. Mathiassen, Madsen, Nielsen dan Stage (2000:232-271) memberikan pembagian aktivitas dalam *component design* menjadi tiga komponen, yaitu :

1. Model Component
   Bagian dari sistem yang memberikan penjabaran mengenai model *problem domain* yang didefinisikan bagaimana suatu model dapat digambarkan menjadi sebuah *class* dalam sistem.
2. Function Component
   Bagian dari sistem ini menjelaskan bagaimana berbagai macam fungsi yang ada dalam sistem diaplikasikan dengan tujuan untuk memberi akses pada *user interface* dan komponen dalam sistem lain ke dalam model.
3. Connecting Component
   Bagian dari sistem yang bertujuan untuk bagaimana semua komponen dalam sistem saling terhubung satu sama lain. Terdapat dua konsep dalam *connecting component,* yaitu :
   1. *Coupling* : Suatu pengukuran yang menjelaskan seberapa dekat hubungan antara kedua *class* atau komponen dalam sistem.
   2. *Cohesion* : Suatu pengukuran yang menjelaskan dan menggambarkan seberapa baik hubungan suatu *class* atau komponen dalam sistem.

5.1 Model Component

Model component bertujuan untuk menyampaikan data pada masa lalu dan saat ini kepada *function, interface*, dan *user* dan sistem lainnya. *Model component* adalah bagian dari sistem yang mengimplementasikan model *problem domain*.



Gambar 5.1.1. *Aggregation structure* dan *statechart diagram class* transaksi dan servis *laundry* sistem *laundry* HiClean

Berikut adalah *event table* antar-*class* yang akan dipisahkan pada sistem *laundry* HiClean :

Tabel 5.1.1. *User interface* detail *service* sistem *laundry* HiClean

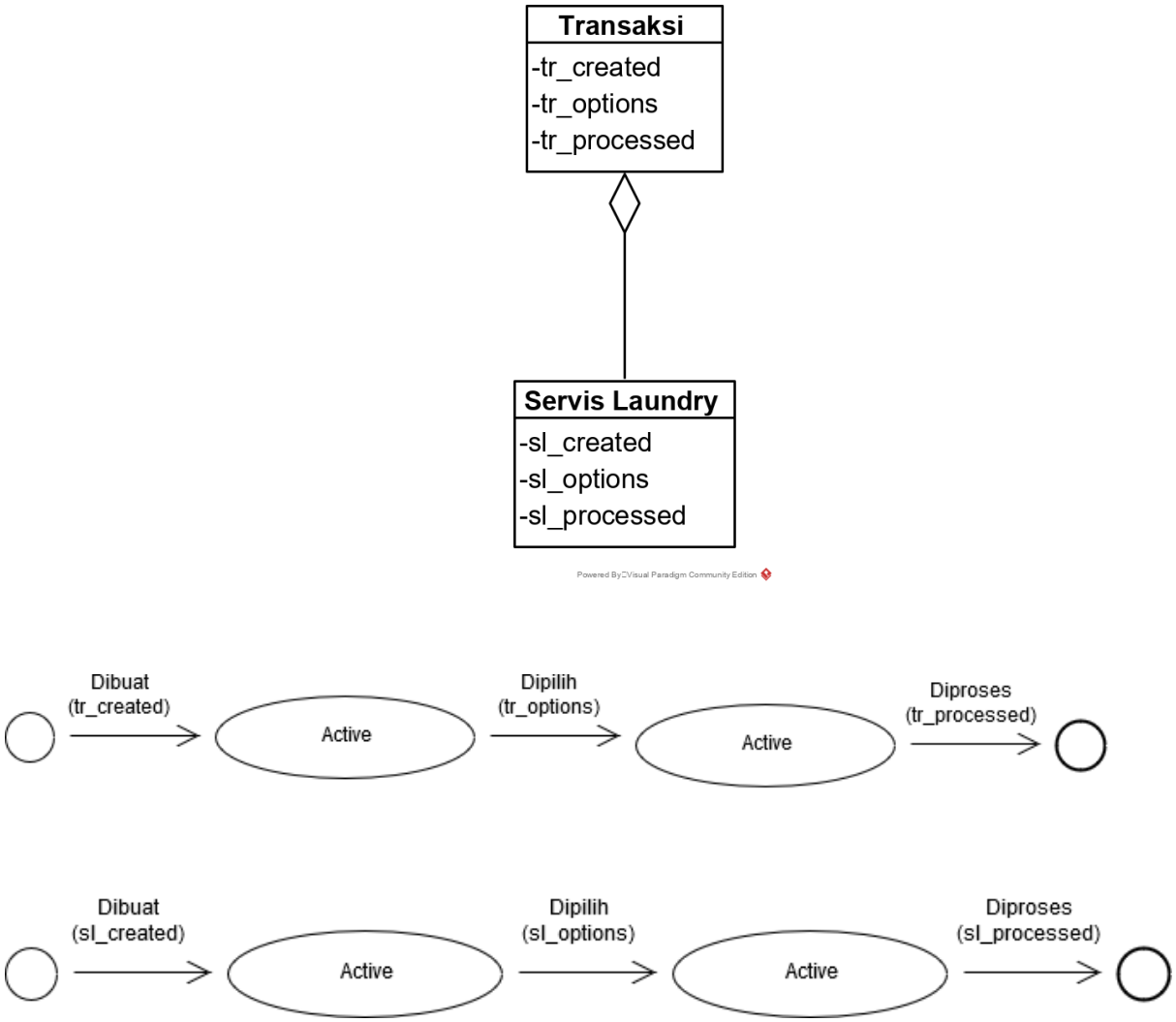
Event	Class	
	Transaksi	Service Laundry
Dibuat	+	+
Diproses	+	+
Dipilih	+	+

Dalam *event* Dibuat, class transaksi dan *class service laundry* terjadi sekali saja, berdasarkan teori representasi *private event*, maka dibentuk atribut baru karena bersifat *same way* dikarenakan kedua class dalam *event* ini hanya dilakukan sekali. Penambahan atribut untuk class transaksi dinamakan "tr_created" yang menunjukkan pencatatan waktu awal transaksi *client* kesistem yang dilakukan oleh karyawan. Sedangkan dalam *class service laundry* juga dilakukan penambahan atribut dengan nama "sl-created" dimana menunjukkan pencatatan waktu awal pencucian barang *client* yang dilakukan oleh karyawan.

Dalam *event* Diproses, *class* transaksi dan *class* servis *laundry* merupakan peristiwa yang terjadi sekali saja, berdasarkan teori representasi *private event*, maka dibentuk atribut baru karena kedua *class* ini melakukan *event* ini sekali saja. Pada *class* transaksi penambahan atribut dinamakan "tr_processed" yang menunjukkan pencatatan selesainya transaksi *client* yang dilakukan oleh karyawan. Sedangkan pada *class* servis *laundry* penambahan atribut dinamakan "sl_processed" yang menunjukkan pencacatan waktu selesainya pencucian barang *client* yang dilakukan oleh karyawan.

Dalam *event* Dipilih, *class* transaksi dan *class* servis *laundry* juga terjadi sekali saja, sehingga dibuat atribut baru untuk masing-masing *class*. Pada *class* transaksi, atribut baru ditambahkan dengan nama "tr_options" yang menunjukkan jenis pembayaran yang dipilih oleh *client*. Kemudian pada *class* servis *laundry*, ditambahkan atribut baru bernama "sl_options" yang menunjukkan pemilihan jenis cucian yang diinginkan oleh *client*.

Berikut *class diagram* dan *statechart diagram* yang dilakukan penataan kembali :



Gambar 5.1.2. *Revised class* dan *statechart diagram* transaksi dan servis *laundry* sistem *laundry* HiClean

Class diagram di atas merupakan *class diagram* yang terdiri dari *class* baru yang sudah memiliki atribut-atribut baru dalam tiap kelasnya untuk memisahkan tiap *class* tersebut dengan *event* yang sama. Namun setiap *event* tiap *class* memiliki nilai yang berbeda. *Event-event* dalam *revised class* tersebut terjadi secara *sequence* atau *selection* yang terjadi sekali setiap *event*-nya.

5.2 Function Component

Function component bertujuan untuk memberikan *interface* dan sistem lainnya kepada *user* untuk mengakses modelnya. *Function component* adalah bagian dari sistem yang mengimplementasikan kebutuhan-kebutuhan fungsional.

Empat tipe *function* yang digunakan pada analisis *application domain* berperan penting dalam mendesain *function* sebagai operasi dalam model *class* dan *function component*. Keempat tipe desain tersebut antara lain *update, read, compute,* dan *signal*.

a. *Update*
   Fungsi *update* berhubungan langsung dengan *event problem domain*. Jika tidak ada apapun yang terjadi di *problem domain*, maka tidak ada yang perlu diganti dari *system model*. Akan tetapi, jika suatu *event problem domain* terjadi, maka harus menuliskan fungsi *update* pada model. Fungsi *update* menerima *input data* yang mendeskripsikan suatu *event*, lalu meng-*output data* berupa *model update*.
b. *Read*
   Fungsi *read* adalah untuk merefleksikan kebutuhan dari *user* atau sistem lain untuk mendapatkan informasi dari model. Sistem ini dilihat sebagai suatu *database*, dimana terdapat informasi yang dapat didefinisikan sebagai atribut. Setiap kebutuhan *read* harus diimplementasikan sebagai fungsi *read*, atau alternatifnya sebagai fungsi *compute.*
c. *Compute*
   Fungsi *compute* adalah untuk menandakan bahwa *user* atau sistem lain membutuhkan pemrosesan data, dimana akan melibatkan pembacaan model.
d. *Signal*
   Fungsi *signal* adalah untuk menggambarkan kebutuhan yang dibutuhkan untuk mengawasi atau mengontrol sistem. Pengawasan atau pengontrolan sistem merupakan fungsi yang krusial dalam sistem, dimana model sistem dipelihara untuk melihat jika *problem domain* mencapai kondisi yang memerlukan reaksi.

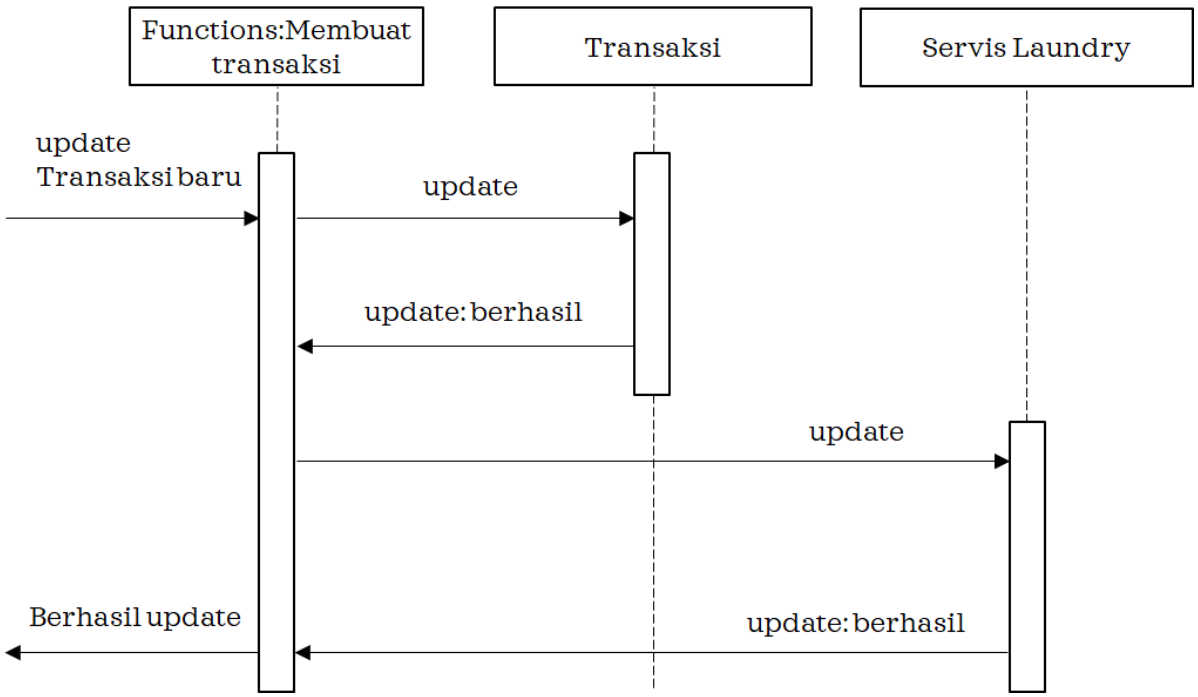
Dalam sistem *Laundry* HiClean, dengan menggunakan *revised class diagram*, fungsi yang terdapat pada class Transaksi dan Servis Laundry yaitu *update*. Berikut adalah tabel *function list* yang berisi fungsi dalam *class* tersebut dan kompleksitasnya:

Tabel 5.2.1. *Function list complexity* sistem *laundry* HiClean

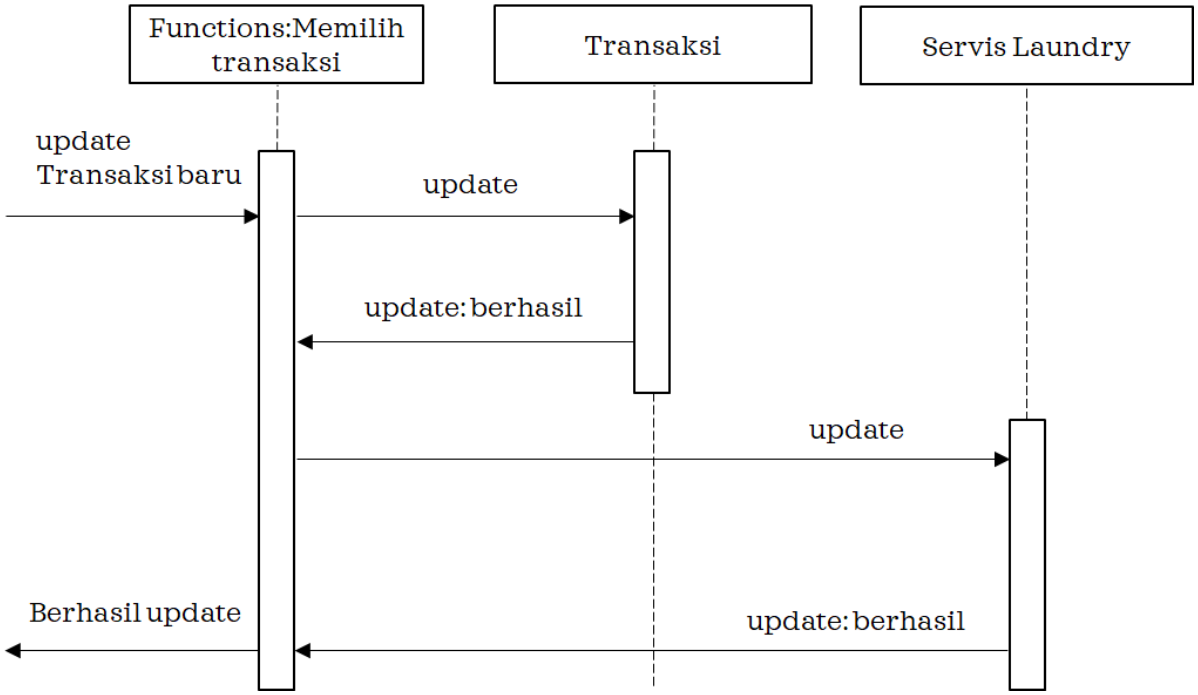
Function	Complexity	Type

Transaksi		
Membuat transaksi	Complex	Update
Memilih transaksi	Medium	Update
Memproses transaksi	Simple	Update
Servis Laundry		
Membuat servis laundry	Medium	Update
Memilih servis laundry	Complex	Update
Memproses servis laundry	Simple	Update

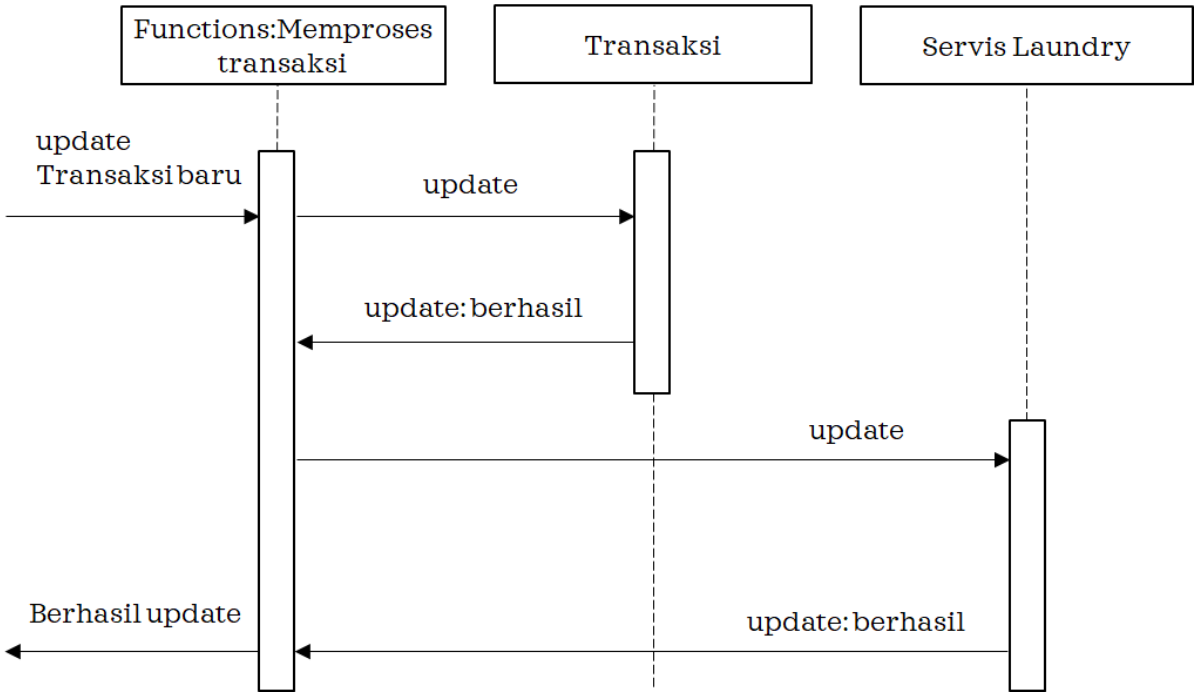
Setiap fungsi tersebut kemudian dimasukkan ke dalam *sequence diagram* sebagai sebuah operasi.



Gambar 5.2.1. *Sequence diagram function* membuat transaksi



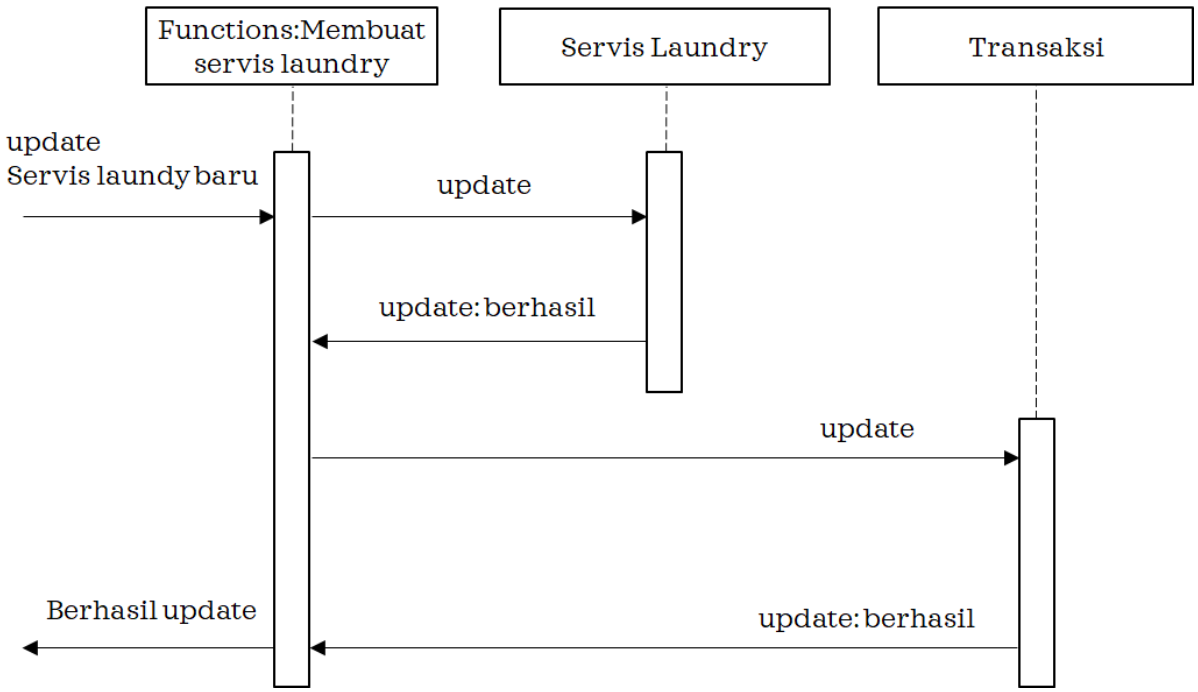
Gambar 5.2.2. *Sequence diagram function* memilih transaksi



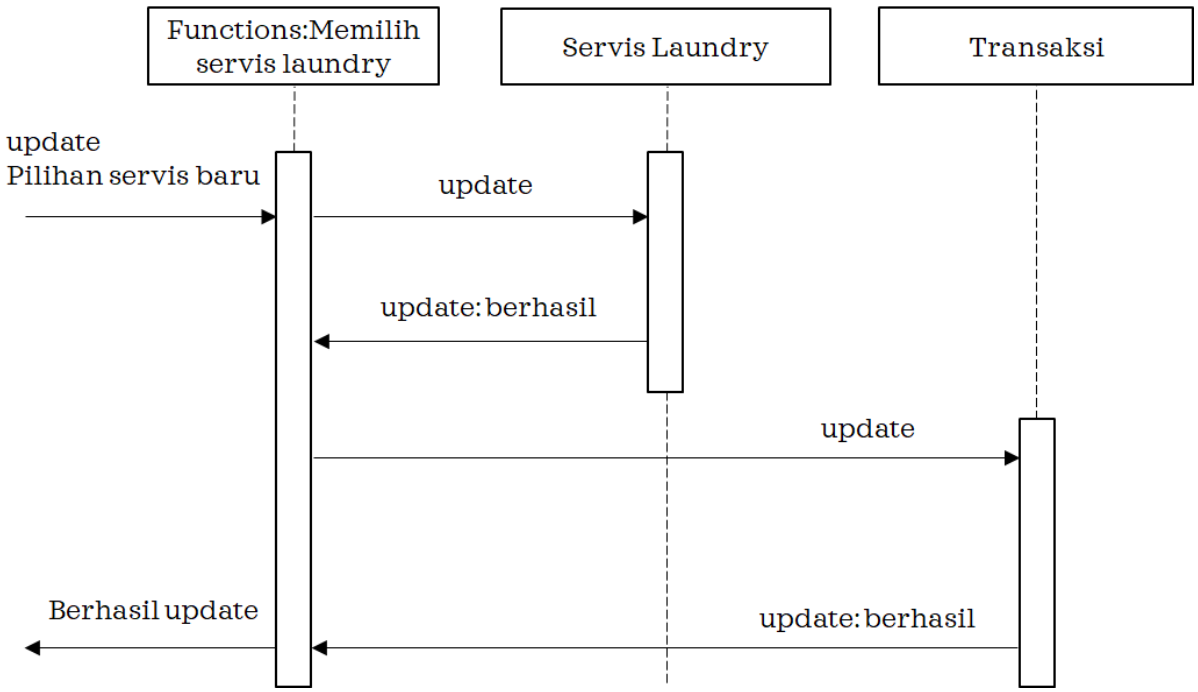
Gambar 5.2.3. *Sequence diagram function* memproses transaksi

Fungsi dalam *class* Transaksi terdiri dari fungsi membuat transaksi, memilih transaksi, dan memproses transaksi. Ketiga *function* tersebut menggunakan fungsi *update* dengan *complex complexity*, *medium complexity*, dan *simple complexity*. Fungsi *update* ini dilakukan pada model *class* Transaksi. *Update* pada model *class* Transaksi ini akan memperbaharui data

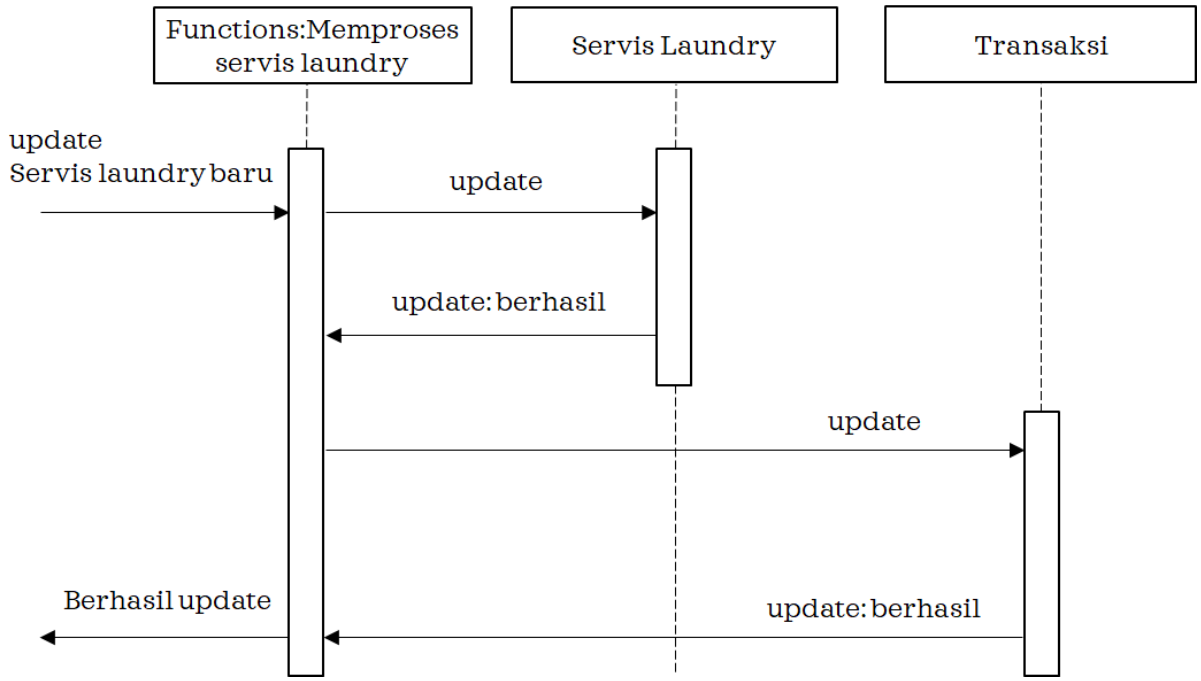
dalam *database* dari HiClean dan dikarenakan model ini terhubung dalam *class* Servis *Laundry* oleh karena itu, ketika *update* dilakukan pada *class* Transaksi, maka *class* Servis *Laundry* akan melakukan pembaharuan data kembali untuk servis *laundry* HiClean.



Gambar 5.2.4. *Sequence diagram function* membuat servis *laundry*



Gambar 5.2.5. *Sequence diagram function* memilih servis *laundry*



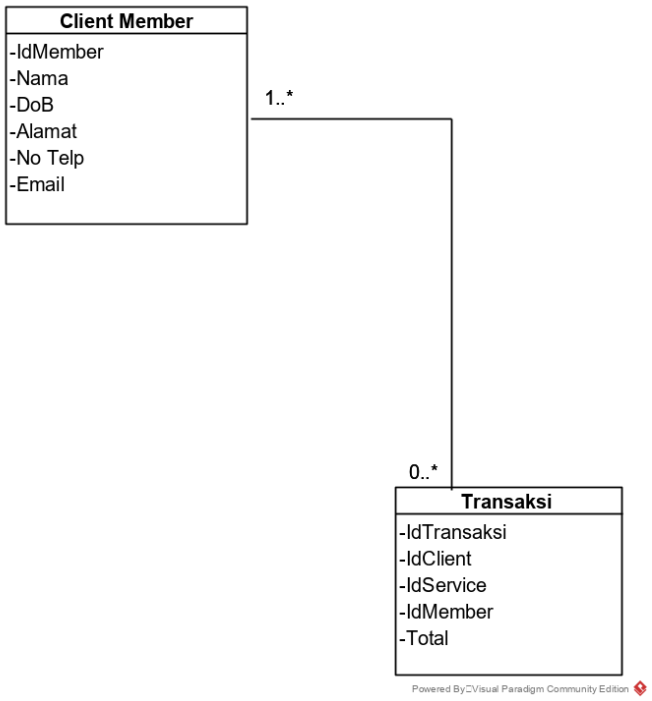
Gambar 5.2.6. *Sequence diagram function* memproses servis *laundry*

Fungsi dalam *class* Service *Laundry* terdiri dari fungsi membuat servis *laundry*, memilih servis *laundry*, dan memproses servis *laundry*. Ketiga function tersebut menggunakan fungsi *update* dengan *complex complexity*, *medium complexity*, dan *simple complexity*. Fungsi *update* ini dilakukan pada model *class* Transaksi. *Update* pada model *class* Service *Laundry* ini akan memperbaharui data dalam *database* dari HiClean dan dikarenakan model ini terhubung dalam *class* Transaksi oleh karena itu, ketika *update* dilakukan pada *class* Servis *Laundry*, maka *class* Transaksi akan melakukan pembaharuan data kembali untuk transaksi HiClean.

5.3 Connecting Component

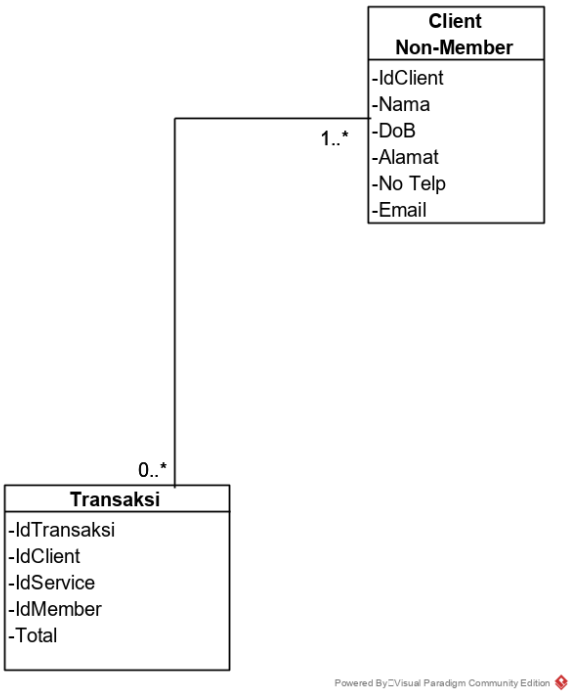
Coupling dalam *connecting component* adalah properti yang bersifat negatif dan harus diminimalkan dalam pembuatan desain. Terdapat 4 macam *coupling* dalam *connecting component* :

1. *Outside Coupling* : Sebuah *class* atau komponen diarahkan langsung ke *public properties* dari *class* atau komponen lain.
2. *Inside Coupling* : Sebuah operasi diarahkan langsung ke *private properties* lainnya dalam *class* yang sama.
3. *Coupling from below* : Kelas khusus diarahkan langsung ke *private properties* dalam *super class.*
4. *Sideways Coupling* : Sebuah *class* yang diarahkan langsung ke *private properties* dalam *class* lain.



Gambar 5.3.1. *Association class client member* dan transaksi sistem *laundry* HiClean

Pada *class Client Member* dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri *client member* dan detail dari transaksi yang dimiliki oleh *client member*. *Client member* dapat mengakses transaksi yang sudah dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



Gambar 5.3.2. *Association class client non-member* dan transaksi sistem *laundry* HiClean

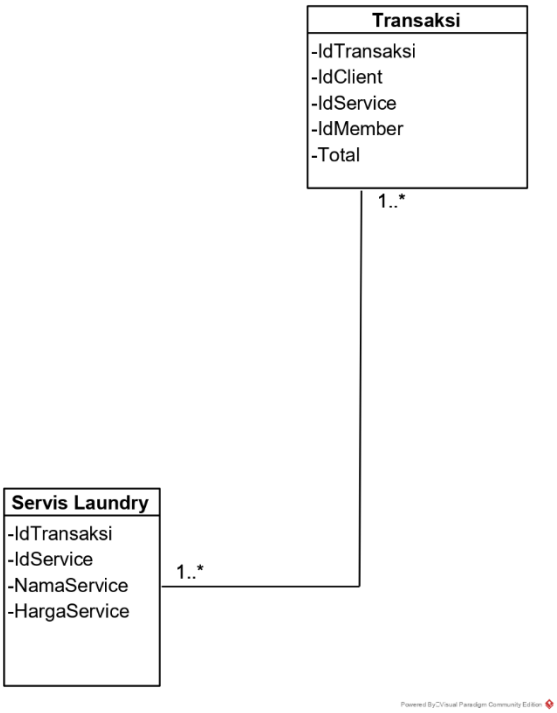
Class Client Non-Member dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri *client non-member* dan detail dari transaksi yang dimiliki oleh *client non-member*. *Client non-member* dapat mengakses transaksi yang sudah dilakukan dengan mengakses *public attributes* dari *class* Transaksi.

Gambar 5.3.3. *Association class* karyawan dan transaksi sistem *laundry* HiClean

Pada *class* Karyawan dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri karyawan dan detail dari transaksi yang dimiliki oleh *client member* maupun *non-member*. Karyawan dapat mengakses transaksi yang sudah dilakukan dan di-*update* olehnya dengan mengakses *public attributes* dari *class* Transaksi.

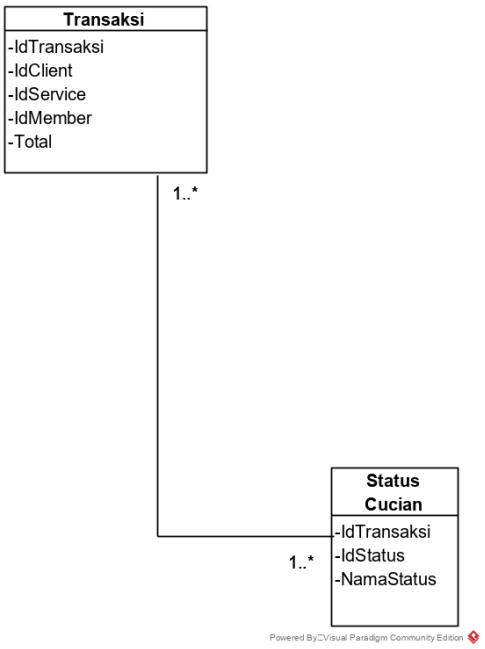
Gambar 5.3.4. *Association class* transaksi dan *owner* sistem *laundry* HiClean

Pada *class Owner* dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri *owner* dan detail dari transaksi yang dimiliki oleh *client member* maupun *client non-member*. *Owner* dapat mengakses transaksi yang sudah dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



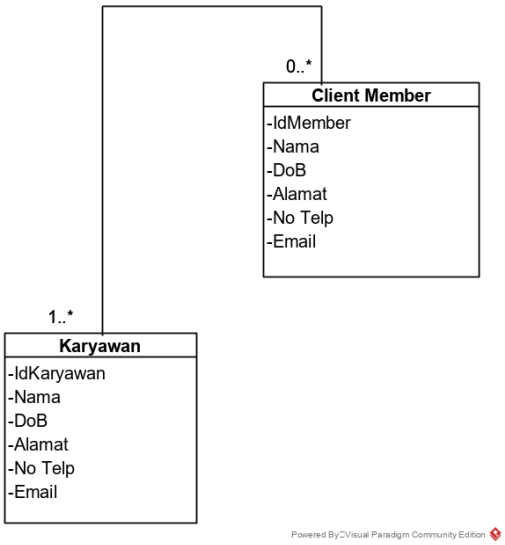
Gambar 5.3.5. *Association class* servis *laundry* dan transaksi sistem *laundry* HiClean

Pada *class* Servis *Laundry* dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail dari servis *laundry* dan detail dari transaksi yang dimiliki oleh *client member* maupun *client non-member*. *Class* Transaksi dapat mengakses detail servis *laundry* yang sudah dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



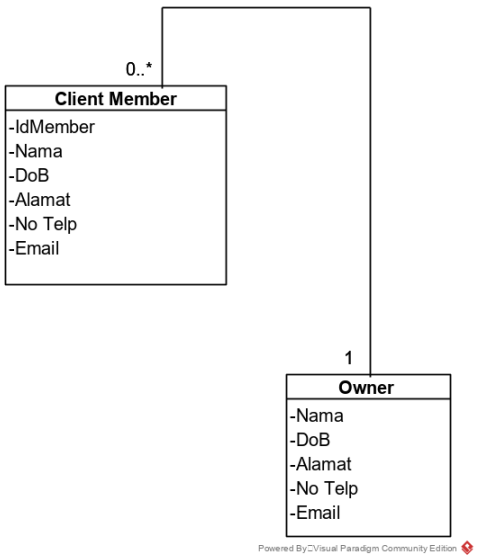
Gambar 5.3.6. *Association class* transaksi dan status cucian sistem *laundry* HiClean

Pada *class* Status Cucian dan Transaksi mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail dari status cucian dan detail dari transaksi yang dimiliki oleh *client member* maupun *client non-member*. *Class* Transaksi dapat mengakses detail status cucian yang sudah dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



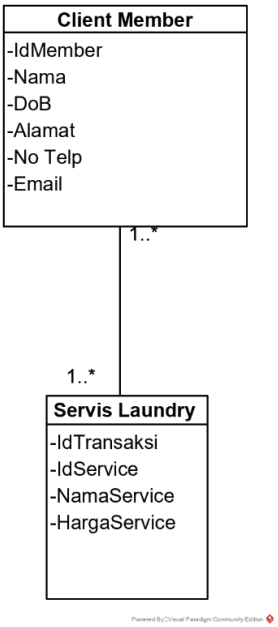
Gambar 5.3.7. *Association class client member* dan karyawan sistem *laundry* HiClean

Pada *class Client Member* dan Karyawan memiliki hubungan *outside coupling.* kedua *class* ini berisikan data-data mengenai informasi detail dari karyawan dan detail dari *client member. Class* karyawan dapat mengakses data *client member* melalui *public attributes* dari *class Client Member.*



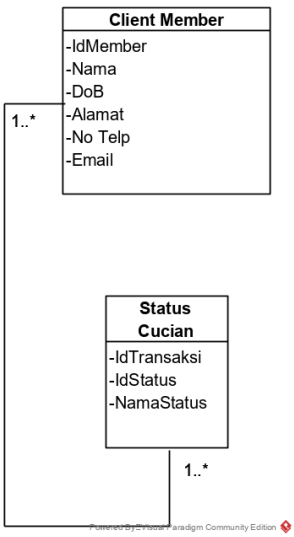
Gambar 5.3.8. *Association class client member* dan *owner* sistem *laundry* HiClean

Pada *class Owner* dan *Client Member* mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri *owner* dan data diri *client member*. *Class Owner* dapat mengakses detail informasi data diri *client member* dengan mengakses *public attributes* dari *class* Transaksi.



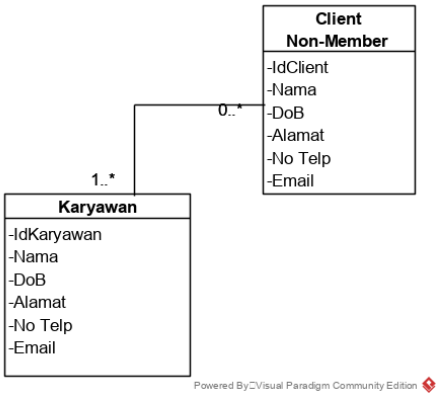
Gambar 5.3.9. *Association class client member* dan servis *laundry* sistem *laundry* HiClean

Pada *class* Servis *Laundry* dan *Client Member* mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail informasi servis *laundry* dan data diri *client member*. *Class Client Member* dapat mengakses detail informasi servis *laundry* yang dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



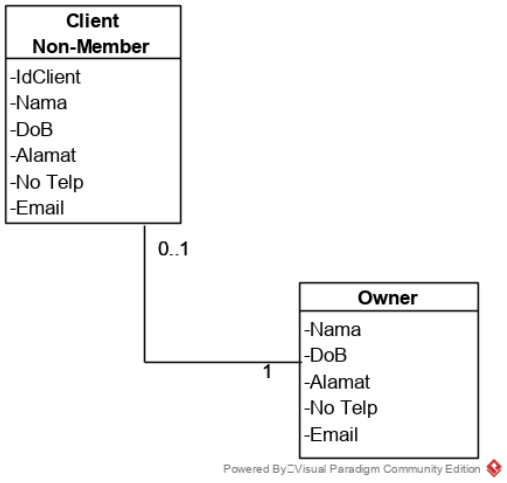
Gambar 5.3.10. *Association class client member* dan status cucian sistem *laundry* HiClean

Pada *class* Status Cucian dan *Client Member* mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail status cucian dan data diri *client member*. *Class Client Member* dapat mengakses detail informasi status cucian yang dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



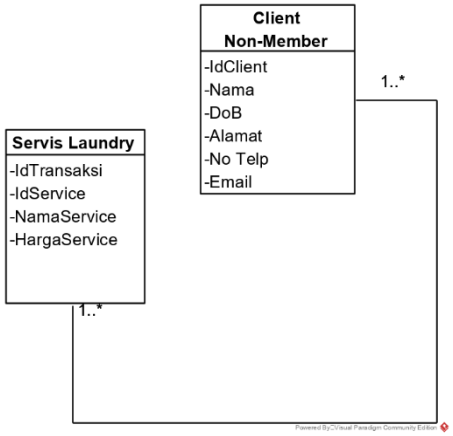
Gambar 5.3.11. *Association class client non-member* dan karyawan sistem *laundry* HiClean

Pada *class Client Non-Member* dan Karyawan memiliki hubungan *outside coupling.* kedua *class* ini berisikan data-data mengenai informasi detail dari karyawan dan detail dari *Client Non-Member. Class* karyawan dapat mengakses data *client member* melalui *public attributes* dari *class Client Non-Member.*



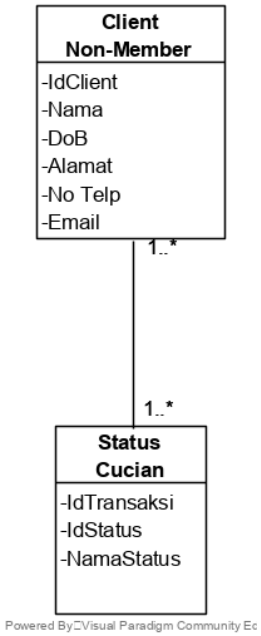
Gambar 5.3.12. *Association class client non-member* dan *owner* sistem *laundry* HiClean

Pada *class Owner* dan *Client Non-Member* mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai data diri *owner* dan data diri *client Non-Member*. *Class Owner* dapat mengakses detail informasi data diri *client Non-Member* dengan mengakses *public attributes* dari *class* Transaksi.



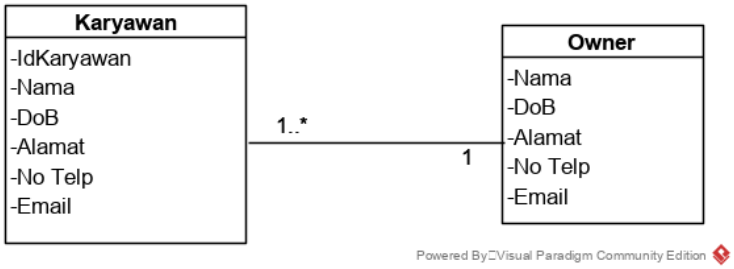
Gambar 5.3.13. *Association class client non-member* dan servis *laundry* sistem *laundry* HiClean

Pada *class Client Non-Member* dan Servis *Laundry* memiliki hubungan *outside coupling.* Kedua *class* ini berisikan data-data mengenai informasi data diri *client non-member* dan detail informasi servis *laundry. Class* Servis *Laundry* dapat mengakses informasi servis *laundry* yang dilakukan melalui *public attributes* dari *class Client Non-Member.*



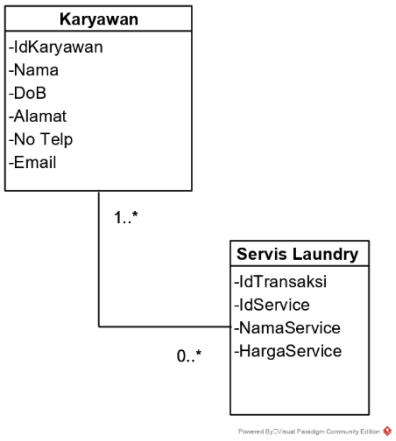
Gambar 5.3.14. *Association class client non-member* dan status cucian sistem *laundry* HiClean

Pada *class* Status Cucian dan *Client Non-Member* mempunyai hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail status cucian dan data diri *client Non-Member*. *Class Non-Member* dapat mengakses detail informasi status cucian yang dilakukan dengan mengakses *public attributes* dari *class* Transaksi.



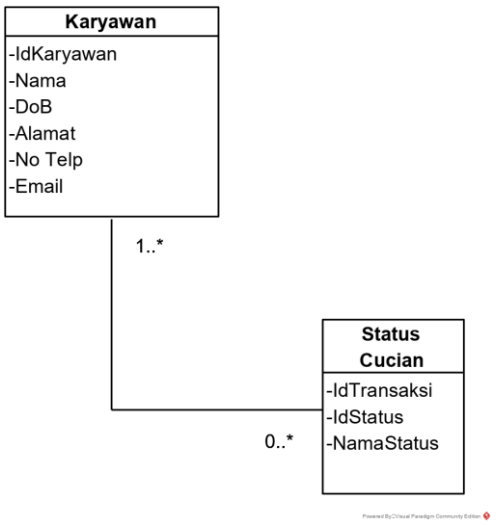
Gambar 5.3.15. *Association class* karyawan dan *owner* sistem *laundry* HiClean

Pada *class* Karyawan dan *Owner* mempunyai hubungan *sideways coupling*. Kedua *class* ini berisikan data mengenai informasi detail data diri dari karyawan dan detail data diri *Owner*. *Class Owner* dapat mengakses detail informasi mengenai karyawan dengan dilakukannya melalui *private attributes* dari *class* Karyawan.



Gambar 5.3.16. *Association class* karyawan dan servis *laundry* sistem *laundry* HiClean

Pada *class* Service *Laundry* dan Karyawan mempunyai hubungan *outside coupling.* Kedua *class* tersebut berisi informasi mengenai detail servis *laundry* dan data diri karyawan. *Class* Karyawan dapat mengakses detail informasi servis *laundry* yang dilakukan oleh *client member* maupun *client non-member* dengan mengakses *public attributes* dari *class* Transaksi.



Gambar 5.3.17. *Association class* karyawan dan status cucian sistem *laundry* HiClean

Pada *class* Karyawan dan Status Cucian memiliki hubungan *outside coupling*. Kedua *class* tersebut berisi informasi mengenai detail status cucian dan data diri karyawan. *Class* Karyawan dapat mengakses informasi detail dari status cucian milik *client member* maupun *client non-member* menggunakan *public attributes* yang ada dalam *class* Status Cucian.

BAB VI

KESIMPULAN

Pada era modern saat ini, teknologi terus berkembang pesat. Seiring bertambah majunya teknologi membuat semua hal menjadi sangat mudah. Perkembangan bisnis pun semakin maju sejalan dengan perkembangan teknologi yang digunakan oleh bisnis. Dalam perancangan sistem *Laundry* HiClean ini, kami ingin meningkatkan kemudahan operasional dan administrasi dalam kegiatan bisnis *Laundry* HiClean.

Saat ini di Indonesia, sudah mulai banyak bisnis yang mulai menggunakan teknologi dan sistem untuk menjalankan bisnisnya. Hal ini didukung dengan Revolusi Industri 4.0. Penggunaan sistem yang dilakukan oleh bisnis-bisnis baik bisnis kecil, menengah, maupun besar sudah mulai menggunakan sedikit demi sedikit sistem.

Sistem yang kami rancang ini menggunakan sistem *tracking* untuk dapat memudahkan *client* dalam melakukan transaksi *laundry*. Dengan menggunakan sistem *tracking*, *client* dapat mengecek status *laundry*-nya sesuai status yang benar dalam proses di *Laundry* HiClean.

Dengan sistem yang sudah didesain, juga memberikan kemudahan seperti pada fitur notifikasi proses *laundry* menggunakan sistem *inbox* yang sudah dirancang. Dalam fitur *inbox* ini, *client* dapat menemukan seluruh notifikasi yang dibuat oleh sistem dan disimpan dalam *inbox* untuk ditinjau kembali oleh *client*.

Laundry HiClean diharapkan dengan menggunakan sistem ini dapat mempercepat dan menghemat waktu *client* dalam melakukan proses transaksi *laundry* di *Laundry* HiClean. Sistem ini juga mempermudah karyawan dan *owner* untuk melakukan *update* dan monitor seluruh transaksi *client* yang terjadi.

DAFTAR PUSTAKA

Mathiassen, L. (2000). *Object-oriented Analysis & Design*. Aalborg, Denmark: Marko Publishing.

Unified Modeling Language (UML): Activity Diagrams. (2018, February 13). Retrieved from https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/.