BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

Penelitian ini dilakukan untuk merancang suatu sistem informasi Penjualan yang berbasis *Intranet*. Objek penelitian pada penyusunan skripsi ini adalah pada LeCafe yang terletak di Jalan Cisitu Lama No.11 Bandung dan LeMilk Factory yang terletak di Jalan Batik Kumeli No.80A Bandung.

3.1.1 Sejarah Singkat LeCafe dan LeMilk Factory.

Pada awalnya LeCafe adalah suatu tempat wisata kuliner yang bergerak di bidang penjualan makanan dan minuman yang berdiri pada awal tahun 2012 di Kota Bandung. Untuk menu minuman pada LeCafe selain berupa *tea* ada juga berupa susu dan yoghurt yang diproduksi oleh LeCafe sendiri. Susu dan yoghurt ini pun selain dijual pada LeCafe mereka pun berhasil menjual *retail*, grosir, dan *booth* yang omzetnya bisa lebih dari target. Dengan melihat peningkatan penjualan tersebut pemilik mendirikan kembali *cafe* khusus untuk penjualan susu dan yoghurt pada Oktober 2012 yang terletak di Jalan Batik Kumeli No.80A Bandung dengan nama cafe yaitu LeMilk Factory. LeCafe dan LeMilk Factory berada di bawah naungan ICDC (*Innovation for Community development Center*) Indonesia. ICDC adalah perusahaan yang bergerak dibidang pengembangan potensi masyarakat desa, mulai dari bidang pendidikan, kesehatan, lingkungan, dan khusunya ekonomi. Potensi itu dikelola sehingga masyarakat bisa hidup

secara mandiri. Dalam bidang ekonomi, perekonomian masyarakat yang ingin dibangun berasal dari potensi masyarakat itu sendiri. Misal, lembang menghasilkan susu dari sapi perah dan tugas pemilik adalah membantu dan memastikan masyarakat memiliki pasar dengan membuka channel distribusi seluas luasnya. Sedangkan LeCafe dan LeMilk Factory adalah bagian dari proses distribusi produk susu secara langsung ke konsumen. Dengan demikian LeCafe dan LeMilk Factory adalah dua *cafe* yang berada pada satu perusahaan.

3.1.2 Visi dan Misi LeCafe dan LeMilk Factory

3.1.2.1 Visi LeCafe dan LeMilk Factory

" Menjadi kedai susu pertama dan satu-satunya di Bandung yang mengedepankan kualitas dan mampu terbrand hingga skala nasional."

3.1.2.2 Misi LeCafe dan LeMilk Factory

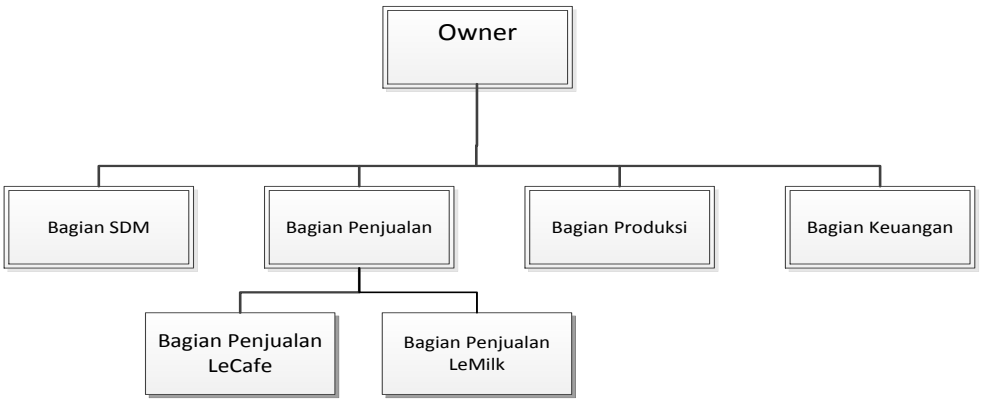
1. Menjaga dan meningkatkan mutu dan standar produk.
2. Menciptakan sistem distribusi, marketing, dan penjualan yang baik dan terintegrasi.
3. Memberikan pelayanan terbaik kepada konsumen.
4. Menjaga hubungan dengan entitas luar/partner.
5. Mempertahankan nilai-nilai sociopreneur.

3.1.3 Struktur Organisasi

Dalam melakukan aktifitas sehari-hari LeCafe dan LeMilk Factory, maka sangat diperlukan suatu struktur organisasi, yang tersusun dan tertata dengan fungsi dan tugasnya masing-masing, karena dengan adanya struktur organisasi dapat membantu perusahaan untuk melakukan penetapan pembagian kerja. Berikut ini adalah struktur organisasi pada LeCafe yang terletak di Jalan Cisitu Lama No.11 Bandung dan LeMilk Factory yang terletak di Jalan Batik Kumeli No.80A Bandung.

Struktur Organisasi

LeCafe dan LeMilk Factory



Gambar 3.1 Struktur Organisasi

(Sumber : Data LeCafe dan LeMilk Factory)

3.1.4 Deskripsi Kerja

Deskripsi tugas atau *job description* adalah suatu rincian yang menunjukan posisi, tanggung jawab, wewenang, fungsi dan tugas tugas yang harus dilakukan. Deskripsi tugas perlu dibuat agar masing-masing bagian mengerti akan kedudukan didalam organisasi. Adapun uraian jabatan pada LeCafe, yaitu :

1. Owner/Pemilik : Memimpin, merencanakan, dan mengendalikan pengaturan Sumber Daya Manusia (SDM), produksi, penjualan dan pemesanan. Serta menerima laporan secara berkala dari bagian penjualan yang berkaitan dengan perkembangan tempat usaha.
2. Bagian SDM : Bertanggung jawab atas segala sesuatu yang berkaitan dengan SDM, dan keluhan pelanggan.
3. Bagian Penjualan LeCafe : Melayani, memberi, menerima dan melakukan pelayanan yang dibutuhkan pelanggan dan melakukan transaksi pembayaran pada pelanggan, serta membuat laporan transaksi penjualan yang ada pada cafe.
4. Bagian Penjualan LeMilk : Melayani, memberi, menerima dan melakukan pelayanan yang dibutuhkan pelanggan dan melakukan transaksi pembayaran pada pelanggan, serta membuat laporan transaksi penjualan yang ada pada cafe.
5. Bagian Produksi : Menangani segala sesuatu yang berkaitan dengan produksi serta kebutuhan pokok produksi.
6. Bagian Keuangan : Mengelola keuangan perusahaan dan membuat laporan keuangan.

3.2 Metode Penelitian

"Metode penelitian pada dasarnya merupakan cara ilmiah untuk mendapatkan data yang valid dengan tujuan dapat ditemukan, dibuktikan, dan dikembangkan suatu pengetahuan sehingga pada gilirannya dapat digunakan untuk memahami, memecahkan, dan mengantisipasi masalah."

3.2.1 Desain Penelitian

Desain penelitian adalah rencana dan struktur penyelidikan yang digunakan untuk mendapatkan bukti-bukti empiris dan dapat dalam menjawab penelitian,dalam melakukan suatu penelitian diperlukan perencanaan penelitian agar penelitian yang dilakukan dapat berjalan dengan baik, sistematis serta efektif. Dalam perancangan penelitian ini yang digunakan oleh penulis yaitu Metode *Deskriptif* dan Metode *Action.*

Metode *Deskriptif* yang mana metode ini merupakan metode penelitian yang berusaha menggambarkan dan menginterpretasikan objek sesuai dengan apa adanya. Metode ini merupakan metode penelitian yang digunakan untuk menangani rumusan masalah yang pertama yaitu mengenai Bagaimana Sistem Pemesanan dan Penjualan yang sedang berjalan pada LeCafe dan LeMilk Factory.

Metode *Action* ini merupakan metode penelitian yang berupa tindakan dan manipulasi data dalam penelitian. Metode ini merupakan metode penelitian yang digunakan untuk menangani rumusan masalah yang kedua dan ketiga yaitu mengenai Bagaimana Perancangan Sistem Pemesanan dan Penjualan pada LeCafe

dan LeMilk Factory dan Bagaimana Pengujian Sistem Pemesanan dan Penjualan pada LeCafe dan LeMilk Factory.

3.2.2 Jenis dan Metode Pengumpulan Data

Sumber data primer data yang diperoleh langsung dari unit pengamatan/responden penelitian, sedangkan sumber data sekunder adalah data yang digunakan untuk mendukung data primer, dapat berupa kajian pustaka dan lain sebagainya.

3.2.2.1 Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelititan ini yaitu dengan mengumpulkan data yang kemudian dianalisis dan memaparkan hasil analisis menjadi sebuah skripsi. Adapun metode pengumpulan data yang digunakan adalah sebagai berikut :

1. Tahap pengumpulan data
    a. Pengumpulan data dengan cara mengumpulkan literature, jurnal, *paper* danbaca-bacaan yang ada kaitannya dengan judul penelitian.
    b. Wawancara *(interview)*, Yaitu Pengumpulan data yang dilakukan dengan tatap muka dan tanya jawab secara langsung antara perancang dengan pihak manajemen yang terkait yaitu, kepada pemilik LeCafe, kepada bagian marketing,dan tidak lupa juga kepada atasan/owner LeCafe tersebut yang memiliki kuasa dan kendali secara penuh dari semua sistem yang sedang berjalan.

c. Observasi *(observation)* Yaitu melakukan penelitian dan pengamatan kegiatan sehingga penulis memperoleh data yang akurat.

3.2.3 Metode Pendekatan dan Pengembangan Sistem

Metode pendekatan dan pengembangan sistem yang diambil dalam skripsi ini adalah metode pendekatan berorientasi objek dengan pengembangan sistem model *Waterfall*, hal ini diambil sesuai masalah yang dipecahkan dan kemampuan dari penulis.

3.2.3.1 Metode Pendekatan Sistem Berorientasi Objek

Metode pendekatan yang digunakan adalah metode pendekatan yang berorientasi objek, yang mencakup analisis dan desain yang disebut *OOAD* (*Object Oriented Analysis and Design*). *Object-oriented analysis* adalah metode analisis yang memeriksa *requirements* (syarat/keperluan yang harus dipenuhi suatu sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup permasalahan. Sedangkan *Object-oriented design* adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

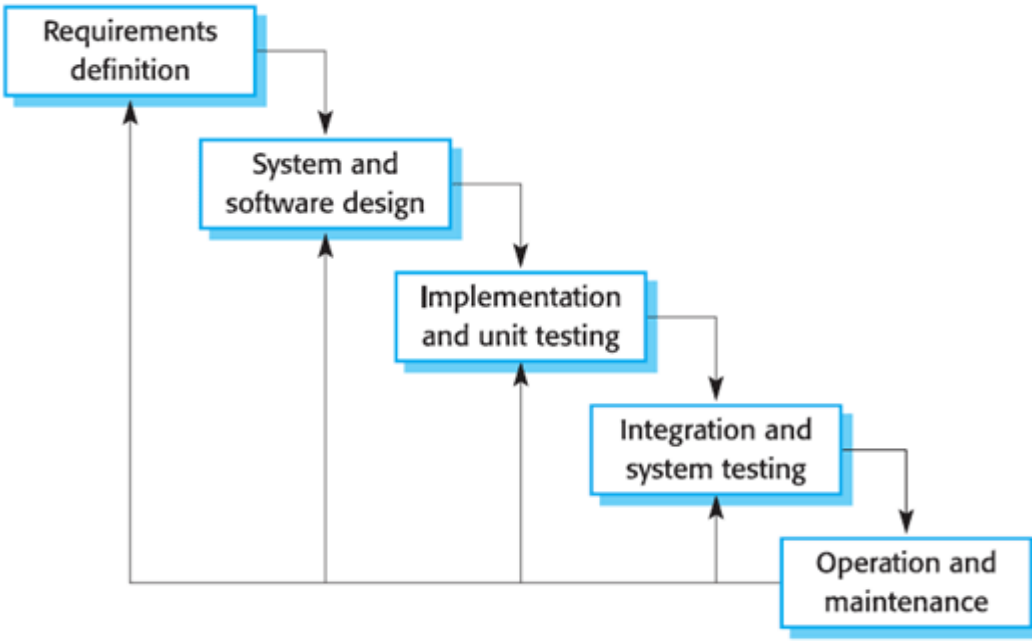
Menurut (Yourdan,1994) karakteristik kunci pendekatan berorientasi objek untuk pengembangan sistem yaitu :

1. **Abstraksi** yaitu mekanisme yang menyederhanakan realitas yang kompleks menjadi model yang lebih sederhana.

2. **Enkapsulasi** yaitu mekanisme yang menyembunyikan implementasi objek sehingga komponen lain tidak akan menyadari bagaimana data itu disimpan pada objek.
3. **Inheritance** yaitu mekanisme menurunkan atribut dan fungsi dari suatu superkelas ke subkelas dalam suatu hirarki.

3.2.3.2 Metode Pengembangan Sistem

Metode Pengembangan yang akan digunakan pada kasus ini adalah model *Waterfall*. Model ini mengusulkan sebuah pendekatan kepada perkembangan aplikasi yang sistematik dan sequensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa konvensional, model sequensial linier melingkupi aktivitas-aktifitas sebagai berikut :



Gambar 3.2 Skema model *Waterfall*

(Sommervile, 2010)

A. *Requirement Analysis and Definition*

Mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan dibangun. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dsb. Tahap ini sering disebut dengan *Project Definition.*

B. System and Software Design

Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang *domain* informasi dari *software*, misalnya fungsi yang dibutuhkan, *user interface*, dsb. Dari dua aktivitas

tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada *user*. Proses *software design* untuk mengubah kebutuhan-kebutuhan di atas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

C. *Implementation and Unit Testing*

Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.

D. *Integration and System Testing*

Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka deain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap desain yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing)*.

E. *Operation and Maintenance*

Sesuatu yang dibuat haruslah diujicobakan demikian juga dengan *software*. Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error* dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya. Pemeliharaan suatu *software*

diperlukan, termasuk di dalamnya adalah pengembangan karena *software* yang dibuat tidak selamaya akan seperti itu. Ketika dijalankan mungkin saja masihada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Penambahan diperlukan ketika adanya perubahan dai eksternal perusahaan seperti ketika adanya penggantian sistem operasi atau perangkat lainnya.

a. Tahapan Metode *Waterfall*

Dalam pengembangannya metode *waterfall* memiliki beberapa tahapan yang runtut: *requirement, design, implementation, verification* dan *maintenance*. Tahap *requirement* atau spesifikasi kebutuhan sistem adalah analisa kebutuhan sistem yang dibuat dalam bentuk yang dapat dimengerti oleh klien dan staf pengembang. Dalam tahap ini klien atau pengguna menjelaskan segala kendala dan tujuan serta mendefinisikan apa yang diinginkan dari sistem. Setelah dokumen spesifikasi disetujui maka dokumen tersebut menjadi kontrak kerja antara klien dan pihak pengembang. Tahap selanjutnya adalah desain, dalam tahap ini pengembang akan menghasilkan sebuah arsitektur sistem secara keseluruhan, dalam tahap ini menentukan alur perangkat lunak hingga pada tahap algoritma yang detil. Selanjutnya tahap implementasi, yaitu tahapan dimana keseluruhan desain diubah menjadi kode-kode program. kode program yang dihasilkan masih berupa modul-modul yang selanjutnya akan di integrasikan menjadi sistem yang lengkap untuk meyakinkan bahwa persyaratan perangkat lunak telah dipenuhi. Tahap

selanjutnya adalah verifikasi oleh klien, klien menguji apakah sistem tersebut telah sesuai dengan kontrak yang telah disetujui. Tahap akhir adalah pemeliharaan yang termasuk diantaranya instalasi dan proses perbaikan sistem sesuai kontrak.

b. **Kelemahan Metode *Waterfall***

Kelemahan pengembangan software dengan metode *waterfall* yang utama adalah lambatnya proses pengembangan perangkat lunak. Dikarenakan prosesnya yang satu persatu dan tidak bisa diloncat-loncat menjadikan model klasik ini sangat memakan waktu dalam pengembangannya. Disisi lain, pihak klien tidak dapat mencoba sistem sebelum sistem benar-benar selesai pembuatannya. Kelemahan yang lain adalah kinerja personil yang tidak optimal dan efisien karena terdapat proses menunggu suatu tahapan selesai terlebih dahulu.

Secara keseluruhan model pendekatan pengembangan *software* dengan metode *waterfall* cocok untuk pengembangan *software* / perangkat lunak dengan tingkat resiko yang kecil, dan memiliki ukuran yang kecil serta waktu pengembangan yang cukup panjang. Model ini tidak disarankan untuk ukuran perangkat lunak yang besar dan tingkat resiko yang besar.

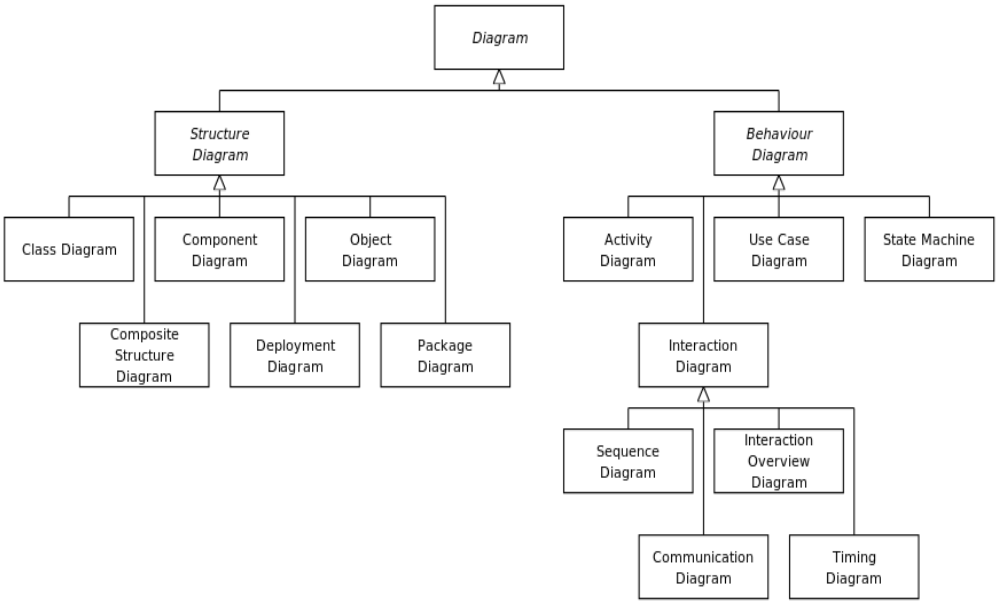
3.2.3.3 Alat Bantu Analisis dan Perancangan

Unified Modelling Languange (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem dengan berfokus pada objek. UML mendefinisikan berbagai diagram, tujuan penggunaan UML antara lain yaitu :

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk Mengembangkan dan saling menukar model dengan mudah dan mengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (blue print) cetak biru karena sangat lengkap dan detail, dengan cetak biru ini maka akan bisa diketahui informasi secara detail tentang coding program atau bajakan membaca program dan menginterprestasikan kembali ke dalam bentuk diagram.

3.2.3.3.1 Diagram UML

UML merupakan sintak umum untuk membuat model logika dari suatu sistem dan digunakan untuk menggambarkan sistem agar dapat dipahami selama fase analisis dan desain. UML biasanya disajikan dalam bentuk diagram/gambar yang meliputi class beserta atribut dan operasinya, serta hubungan antar class yang meliputi inheritance, association dan komposisi.



Gambar 3.3 *Diagram UML*

(Sumber: *Pemodelan Sistem Informasi Berorientasi Objek Dengan UML*, Sholiq)

3.2.3.3.2 Jenis-jenis Diagram UML

Berikut ini adalah jenis-jenis Diagram di dalam UML :

1. ***Use Case Diagram***

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antara *actor* dengan sistem. *Use case* menggambarkan kata kerja seperti *Login* ke sistem, *maintenance user* dan sebagainya.

a. Lambang-lambang Dalam *Use Case Diagram*

1. Aktor merupakan sebuah entitas yang berinterkasi dengan *use case*. Nama actor dituliskan di bawah gambar tersebut.

2. *Use case* menggambarkan sebuah fungsi tertentu yang disediakan oleh sistem, sebuah subsistem atau urutan pertukaran pesan antara anggota sistem dan satu atau lebih *actor* melakukan aksi yang dikerjakan oleh sistem.
3. Hubungan, menggambarkan hubungan *association*. Garis ini digunakan untuk menghubungkan antara *actor* dengan *use case*.

b. Komponen-komponen *Use Case Diagram*

1. *Actor* : *Actor* menggambarkan pengguna *software* aplikasi (*user*). *Actor* membantu memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi.
2. *Use Case* : *Use-case* menggambarkan perilaku *software* aplikasi, termasuk didalamnya interaksi antara *actor* dengan *software* aplikasi tersebut.
3. *Association / Directed Association* : Asosiasi, yaitu hubungan statis antar element. Umumnya menggambarkan *element* yang memiliki atribut berupa element lain, atau *element* yang harus mengetahui eksistensi *element* lain. Panah *navigability* menunjukkan arah query antar *element*.
4. *Generalization* / Pewarisan : Pewarisan, yaitu hubungan hirarkis antar *element*. *Element* dapat diturunkan dari *element* lain dan mewarisi semua atribut dan metoda *element* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari

element yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi

5. *Dependency*/ketergantungan adalah suatu jenis hubungan yang menandakan bahwa satu *element*, atau kelompok *element*, bertindak sebagai klien tergantung pada unsur lain atau kelompok *element* yang berlaku sebagai penyalur. Ini merupakan suatu hubungan lemah yang menandakan bahwa jika penyalur klien diubah maka klien secara otomatis akan terpengaruh oleh perubahan tersebut. Ini merupakan suatu hubungan searah.

2. *Class Diagram*

Class adalah sebuah spesifikasi yang jika di-instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan berorientasi objek. *Class* menggambarkan keadaan (*attribute/property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *packed* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan lainnya.

Class juga memiliki 3 area pokok (utama) yaitu : nama, atribut, dan operasi. Nama berfungsi untuk member identitas pada sebuah kelas, atribut fungsinya adalah untuk member karakteristik pada data yang dimiliki suatu objek di dalam kelas, sedangkan operasi fungsinya adalah memberikan sebuah fungsi ke sebuah objek. Dalam mendefinisikan metode yang ada di dalam kelas harus diperhatikan yang namanya *Cohesion* dan *Coupling*, *Cohesion* adalah

ukuran keterkaitan sebuah instruksi di sebuah metode, *Coupling* adalah ukuran keterkaitan antar metode. Di dalam *class diagram* terdapat hubungan antar kelas secara konseptual, yang disebut Relasi antar *Class*, di UML disediakan macam-macam relasi antar *Class*, diantaranya: Asosiasi (Hubungan statis antar kelas), Agregasi (hubungan dari keseluruhan objek), Generalisasi (relasi beberapa subkelas ke super kelas), *Dependency* (keterhubungan tiap kelas.)

3. *Package Diagram*

Package Diagram atau disebut juga dengan diagram paket, diagram ini memperlihatkan kumpulan kelas-kelas yang merupakan bagian dari diagram komponen. Diagram ini bersifat statis

4. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

5. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak

menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

6. *Sequence Diagram*

Diagram ini menggambarkan interaksi antara objek di dalam sistem dan objek yang di interaksi dengan objek, berupa *message* yang di gambarkan terhadap waktu. *Diagram Sequence* terdiri dari dimensi vertical (waktu 0 dan dimensi horizontal objek yang terkait). *Diagram Sequence* di gunakan untuk menggambarkan *scenario* atau rangkaian langkah-langkah yang di lakukan oleh respon dari suatu *event* untuk menghasilkan *output* tertentu. Di awali dari pemicu aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal.

7. *Collaboration Diagram*

Diagram Collaboration adalah diagram interksi antara objek-objek seperti *diagram Sequence*, tetapi menekankan pada peran dari masing-masing objek. Setiap massege memiliki *Sequence number*, berdasarkan *level* penyampaian *message*. *Collaboration diagram* juga menunjukkan informasi yang sama pesis dengan *diagram squensial*, tetapi dalam bentuk dan tujuan yang berbeda, pada *diagram squensial*, keseluruhan interaksi berdasarkan urutan waktu, tetapi pada diagram kolaborasi, interaksi antar obyek atau aktor ditunjukkan dengan arah

panah tanpa keterangan waktu. Pada dasarnya diagram ini sama dengan *diagram squensial* tetapi orang menggunakan diagram ini untuk alasan yang berbeda diagram ini berbentuk seperti bintang dengan beberapa obyek yang berkomunikasi dengan sebuah obyek.

8. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time*.

Masing-masing kelas dalam model component diagram akan dipetakan ke sebuah komponen kode pustaka, setelah semua komponen dibuat, komponen tersebut akan ditambahkan kedalam komponen yang memberikan relasi antara komponen-komponen, relasi yang terjadi antara komponen biasanya hanya satu tipe relasi yaitu dependensi yang menunjukkan ketergantungan *compile-time* dan *run-time* antara komponen-kompone tersebut.

9. *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah *server, workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

3.2.4 Pengujian Software

Metode pengujian yang akan dilakukan penulis pada pengujian perangkat lunak yang akan dibangun adalah metode pengujian *black-box*, dimana pada pengujian *black-box* ini hanya berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian didapatkan serangkaian kondisi input yang sepenuhnya mengginakan semua persyaratan fungsional untuk suatu program. Metode pengujian ini dilakukan untuk menemukan kesalahan yang mungkin terjadi pada sebuah perangkat lunak seperti :

1. Fungsi-fungsi tidak benar atau hilang.
2. Kesalahan antar muka
3. Kesalahan dalam struktur data atau akses basis data eksternal
4. Kesalahan kinerja.
5. Kesalahan inisialisasi dan kesalahan terminasi.

Pengujian *black box* merupakan pendekatan komplementer dari teknik *white box*, karena pengujian *black box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik *white box*. Pengujian *black box* berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Faktor pengujian adalah hal-hal faktor-faktor yang diperhatikan selama pengujian. Faktor pengujian yang akan digunakan penulis pada tahap pengujian yaitu:

a. *Access Control* (kendali akses)

Menekankan sumberdaya sistem harus dilindungi dari kemungkinan modifikasi, pengrusakan, penyalahgunaan dan prosedur keamanan harus dijalankan secara penuh untuk menjamin integritas data dan program aplikasi. Adapun pengujian yang penulis akan lakukan dengan menguji *login*, sebagai perlindungan dalam program aplikasi ini.

b. *Ease Of Use* (mudah dalam penggunaan)

Menekankan perluasan usaha yang diminta untuk belajar, mengoperasikan dan menyiapkan *input*-an, dan menginterpretasikan *output* dari sistem. Faktor ini tersangkut dengan *usability system* terhadap interaksi antara manusia dan sistem. Dalam penggunaan program aplikasi ini, *user* akan dimudah dalam pengisian kegiatan operasional, yang akan mempercepat penginputan karena disajikan hanya dalam satu halaman saja.