BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Perancangan adalah suatu proses pemilihan dan pemikiran yang menghubungkan fakta-fakta berdasarkan asumsi-asumsi yang berkaitan dengan masa datang dengan menggambarkan dan merumuskan kegiatan-kegiatan tertentu yang diyakini diperlukan untuk mencapai tujuan-tujuan tertentu dan menguraikan bagaimana pencapaiannya. (*Anggraheni Rukmana ; 2011 : 2*)

II.2. Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju. Perangkat lunak aplikasi adalah suatu sub kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untukmelakukan tugas yang diinginkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja,dan pemutar media. (*Fricles Ariwisanto Sianturi ; 2013 : 43*)

II.3. Transfer Data

Menurut Wiiliam J. Seller (1981 : 9) Transfer Data adalah jumlah data dalam *bit* yang melewati suatu medium dalam satu detik dimana symbol verbal dan nonverbal dikirimkan, diterima dan diberi arti. Umumnya dituliskan dalam *bit*

per detik (*bit per second*) dan disimbolkan *bit/s* atau *bps* bukan *bits/s*. Adapun tipe transfer data komunikasi logika pada lapisan *transport* dapat berbentuk :

a. *Reliable* atau *unreliable*

   Reliable adalah data berarti data ditransfer ke tujuannya dalam suatu urutan seperti ketika dikirim. *Unreliable* : Pengiriman data *Unreliable* sangat menggantungkan diri pada lapisan jaringan di bawahnya, sehingga tidak dapat menyakinkan apakah *segment* data dapat dikirimkan sampai ditujuannya atau tidak.

b. *Stateful* atau *Stateless*

   Stateful adalah informasi yang dimasukkan pada satu *request,* yang dikirimkan dari pengirim ke penerima, dapat dimodifikasi untuk *request* berikutnya. *Stateless* adalah Informasi dalam satu request tidak dapat dikaitkan dengan *request* lainnya, sehingga tidak dapat digunakan untuk *request* lain ya. (*Sony Bahagia Sinaga* ; 2012 : 45-46)

II.4. *Wifi*

Awalnya *wifi* ditujukan untuk pengunaan perangkat *nirkabel* dan Jaringan Area Lokal (LAN), namun saat ini lebih banyak digunakan untuk mengakses *internet*. Hal ini memungkinan seseorang dengan komputer dengan kartu *nirkabel* (*wireless card*) atau *personal digital assistant* (PDA) untuk terhubung dengan *internet* dengan menggunakan titik akses (*hotspot*) terdekat. *Wifi* dirancang berdasarkan spesifikasi IEEE 802.11. Sekarang ini ada empat variasi dari 802.11, yaitu: 802.11a, 802.11b, 802.11g, and 802.11n. Spesifikasi b merupakan produk

pertama *wifi*. Variasi g dan n merupakan salah satu produk yang memiliki penjualan terbanyak pada 2005. spesifikasi *wifi* yaitu : Tabel 1 : Spesifikasi *Wifi*

Tabel II.1. Spesifikasi *Wifi*

Spesifikasi Wifi			
Spesifikasi	Kecepatan	Frekuensi Band	Cocok Dengan
802.11b	11 Mb/s	2.4 GHz	b
802.11a	54 Mb/s	5 GHz	A
802.11g	54 Mb/s	2.4 GHz	b,g
802.11n	100 Mb/s	2.4 GHz	B,g,n

Sumber: (Sony Bahagia Sinaga ; 2012 : 47)

Versi *wifi* yang paling luas dalam pasaran AS sekarang ini (berdasarkan dalam IEEE 802.11b/g) beroperasi pada 2.400 MHz sampai 2.483,50 MHz. Dengan begitu mengijinkan operasi dalam 11 *channel* (masing-masing 5 MHz), berpusat di frekuensi berikut :

1. Channel 1 - 2,412 MHz
2. Channel 2 - 2,417 MHz
3. Channel 3 - 2,422 MHz
4. Channel 4 - 2,427 MHz
5. Channel 5 - 2,432 MHz
6. Channel 6 - 2,437 MHz
7. Channel 7 - 2,442 MHz
8. Channel 8 - 2,447 MHz
9. Channel 9 - 2,452 MHz
10. Channel 10 - 2,457 MHz

11. Channel 11 - 2,462 MHz

Secara teknis operasional, *wifi* merupakan salah satu varian teknologi komunikasi dan informasi yang bekerja pada jaringan dan perangkat WLAN (*wireless local area network*). Dengan kata lain, *wifi* adalah sertifikasi merek dagang yang diberikan pabrikan kepada perangkat telekomunikasi (*internet*) yang bekerja di jaringan WLAN dan sudah memenuhi kualitas kapasitas interoperasi yang dipersyaratkan. Teknologi *internet* berbasis *wifi* dibuat dan dikembangkan sekelompok insinyur Amerika Serikat yang bekerja pada *Institute of Electrical and Electronis Engineers* (IEEE) berdasarkan standar teknis perangkat bernomor 802.11b, 802.11a dan 802.16. Perangkat *wifi* sebenarnya tidak hanya mampu bekerja di jaringan WLAN, tetapi juga di jaringan *Wireless Metropolitan Area Network* (WMAN). Karena perangkat dengan standar teknis 802.11b diperuntukkan bagi perangkat WLAN yang digunakan di frekuensi 2,4 GHz atau yang lazim disebut frekuensi ISM (*Industrial, Scientific* dan *Medical*). Sedang untuk perangkat yang berstandar teknis 802.11a dan 802.16 diperuntukkan bagi perangkat WMAN atau juga disebut Wi-Max, yang bekerja di sekitar pita frekuensi 5 GHz. (*Sony Bahagia Sinaga ; 2012 : 47-48*)

II.5. *VB.NET*

Visual Basic adalah salah satu bahasa pemrograman berbasis *desktop* yang dikeluarkan oleh perusahaan perangkat lunak komputer terbesar yaitu *Microsoft*. Pada dasarnya *Visual Studio .NET* didesain untuk menampung berbagai macam bahasa pemrograman dan terlingkup dalam *Visual Studio .NET*. Dengan demikian, dapat membangun aplikasi-aplikasi Windows di dalam *Visual Studio .NET*. *Visual*

Basic .NET merupakan salah satu bahasa pemrograman yang dapat digunakan untuk membuat program aplikasi. (Rian Aldy Hidayat ; 2013 ; 253)

II.6. Algoritma *River's Chiper*

RC4 merupakan jenis stream cipher yang mempunyai sebuah S-Box, S0,S1,...,S255, yang berisi permutasi dari bilangan 0 sampai 255, dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Algoritma RC4 memiliki dua fase, yaitu setup kunci dan pengenkripsian. Dengan kunci yang sama maka pada proses dekripsi data kembali ke bentuk semula. Sampai saat ini RC4 masih banyak digunakan orang dalam mengenkripsi informasi berupa data teks karena algoritmanya yang sederhana namun memiliki kecepatan yang hampir sama dibandingkan algoritma yang lebih rumit. Data tersebut berupa .txt yang dienkrip per karakter dengan sebuah kunci yang mengubah plainteks menjadi cipherteks.(Putri Mandarani ; 2012 ; 33)

II.7. Jaringan Komputer

Jaringan komputer adalah komunikasi data antar komputer, yaitu minimal 2 komputer. Jaringan komputer dapat dilakukan melalui media kabel ataupun nirkabel (*wireless*). Pada sistem antrian rumah sakit ini, jaringan komputer dilakukan melalui media kabel antara 2 komputer. *Interface* yang digunakan adalah DB-25 atau *port* paralel. Data yang dikirimkan antar komputer adalah berupa kode biner 1 dan 0, yang dirancang pada masing-masing program yaitu pada program *server* dan program *client*. Ada tiga tipe jaringan yang umum yang digunakan antara lain : Jaringan *WorkGroup*, Jaringan LAN dan Jaringan WAN

II.7.1. Jaringan Komputer Berdasarkan Skala

1. Jaringan LAN

LAN (*Local Area Network*) adalah suatu kumpulan komputer, dimana terdapat beberapa unit komputer (*client*) dan 1 unit komputer untuk bank data (*server*). Antara masing-masing *client* maupun antara *client* dan *server* dapat saling bertukar *file* maupun saling menggunakan printer yang terhubung pada unit-unit komputer yang terhubung pada jaringan LAN.

2. Jaringan MAN

Metropolitan Area Network atau MAN, merupakan Jenis Jaringan Komputer yang lebih luas dan lebih canggih dari Jenis Jaringan Komputer LAN. Disebut *Metropolitan Area Network* karena Jenis Jaringan Komputer MAN ini biasa digunakan untuk menghubungkan jaringan komputer dari suatu kota ke kota lainnya. Untuk dapat membuat suatu jaringan MAN, biasanya diperlukan adanya operator telekomunikasi untuk menghubungkan antar jaringan komputer.

3. Jaringan WAN

WAN (*Wide Area Network*) adalah kumpulan dari LAN dan/atau *Workgroup* yang dihubungkan dengan menggunakan alat komunikasi modem dan jaringan Internet, dari/ke kantor pusat dan kantor cabang, maupun antar kantor cabang. Dengan sistem jaringan ini, pertukaran data antar kantor dapat dilakukan dengan cepat serta dengan biaya yang *relative* murah. Sistem jaringan ini dapat menggunakan jaringan *Internet* yang sudah ada, untuk menghubungkan antara kantor pusat dan kantor cabang atau dengan PC Stand *Alone/Notebook* yang berada di lain kota ataupun negara. (*Mardison, al husni ; 2012 : 59*)

II.7.2. Jaringan Komputer Bertopologi / Pemasangan

1. Bus

Topologi bus merupakan topologi yang banyak dipergunakan pada masa penggunaan kabel sepaksi menjamur. Dengan menggunakan T-*Connector* (dengan terminator 50ohm pada ujung *network*), maka komputer atau perangkat jaringan lainnya bisa dengan mudah dihubungkan satu sama lain. Instalasi jaringan Bus sangat sederhana, murah dan maksimal terdiri atas 5-7 komputer. Kesulitan yang sering dihadapi adalah kemungkinan terjadinya tabrakan data karena mekanisme jaringan relative sederhana dan jika salah satu node putus maka akan mengganggu kinerja dan trafik seluruh jaringan.

2. *Ring*

Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk jalur melingkar membentuk cincin. Pada topologi cincin, komunikasi data dapat terganggu jika satu titik mengalami gangguan.

3. *Star*

Topologi bintang merupakan bentuk topologi jaringan yang berupa *konvergensi* dari *node* tengah ke setiap *node* atau pengguna. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah. (*Mardison, Al Husni ; 2012 : 59*)

4. *Mesh*

Topologi jala atau Topologi mesh adalah suatu bentuk hubungan antar perangkat dimana setiap perangkat terhubung secara langsung ke perangkat

lainnya yang ada di dalam jaringan. Akibatnya, dalam topologi *mesh* setiap perangkat dapat berkomunikasi langsung dengan perangkat yang dituju (*dedicated links*).

5. Pohon

Topologi Pohon adalah kombinasi karakteristik antara topologi bintang dan topologi bus. Topologi ini terdiri atas kumpulan topologi bintang yang dihubungkan dalam satu topologi bus sebagai jalur tulang punggung atau *backbone*. Komputer-komputer dihubungkan ke *hub*, sedangkan *hub* lain di hubungkan sebagai jalur tulang punggung. Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai hirarki semakin tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer. (*Mardison, Al Husni ; 2012 : 59*)

6. *Linier*

Jaringan komputer dengan topologi runtut (*linear topology*) biasa disebut dengan topologi bus beruntut, tata letak ini termasuk tata letak umum. Satu kabel utama menghubungkan tiap titik sambungan (komputer) yang dihubungkan dengan penyambung yang disebut dengan Penyambung-T dan pada ujungnya harus diakhiri dengan sebuah penamat (*terminator*). Penyambung yang digunakan berjenis BNC (*British Naval Connector*: Penyambung *Bahari Britania*), sebenarnya BNC adalah nama penyambung bukan nama kabelnya, kabel yang digunakan adalah RG 58 (Kabel *Sepaksi Thinnet*). Pemasangan dari topologi bus

beruntut ini sangat sederhana dan murah tetapi sebanyaknya hanya dapat terdiri dari 5-7 komputer.

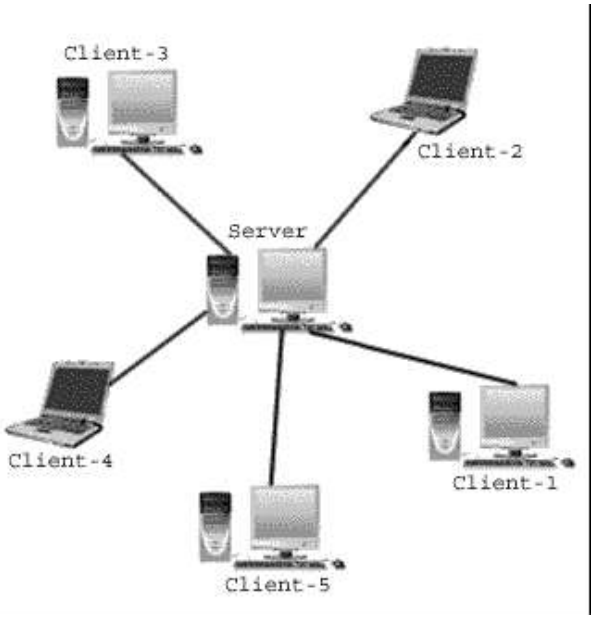
II.8. *Client Server*

Client – Server adalah bentuk *distributed computing* dimana sebuah program (*client*) berkomunikasi dengan program lain (*server*) dengan tujuan untuk bertukar informasi, pada umumnya sebuah *client* memiliki tugas sebagai berikut :

1. Menterjemahkan permintaan pengguna ke dalam bentuk *protocol* yang sesuai.
2. Mengirimkan permintaan pengguna ke *server*.
3. Menunggu respon dari *server*.

Kata *client* juga sering disebut dengan kata *host* yang menandakan bahwa *device* tersebut tersambung dalam sebuah jaringan. Sedangkan sebuah *server* memiliki tanggung jawab sebagai berikut :

1. Mendengarkan permintaan dari *client*.
2. Memproses permintaan tersebut.
3. Mengembalikan hasil proses tersebut ke client. (*Painem* ; 2013 : 16-17)



Gambar II.1. Model *Client Server*
Sumber : (Painem ; 2013 : 17)

II.9. UML *(Unified Modelling Language)*

Unified Modelling Language (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 menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan

bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock,* dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson,* yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma

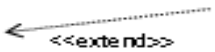
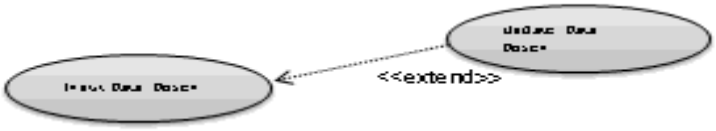
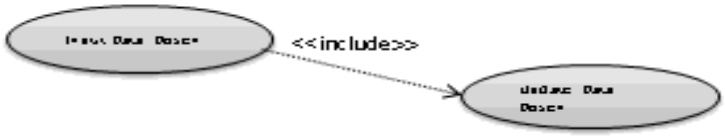
menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

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* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri.

Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

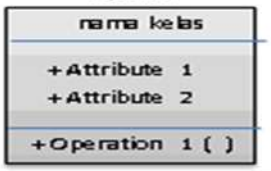
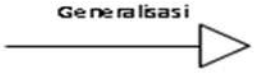
Simbol	Deskripsi
Use Case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Extend  <<extend>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjukan pada use case yang dituju contoh : 
Include  <<include>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Gambar II.2. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

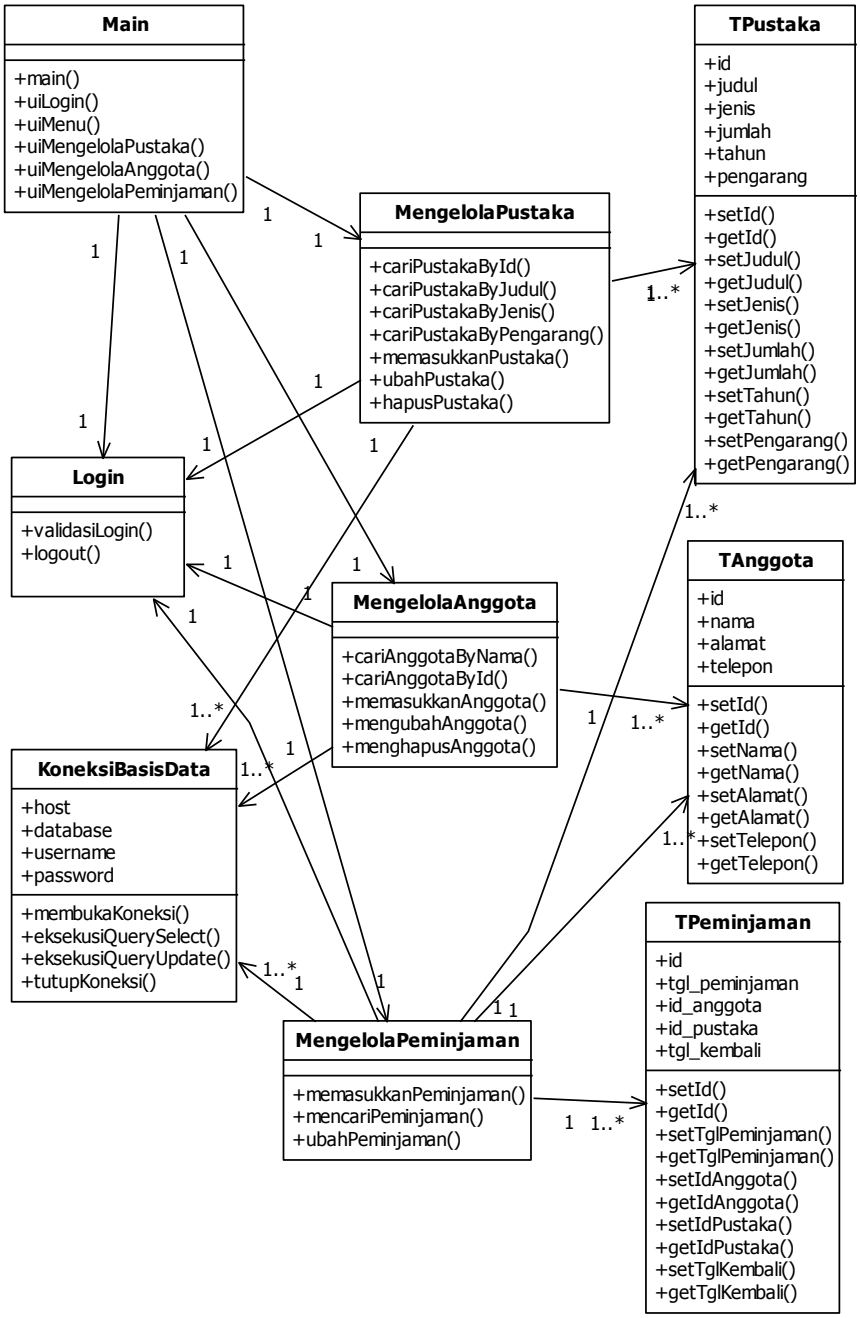
2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas

memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface  interface	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.3. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



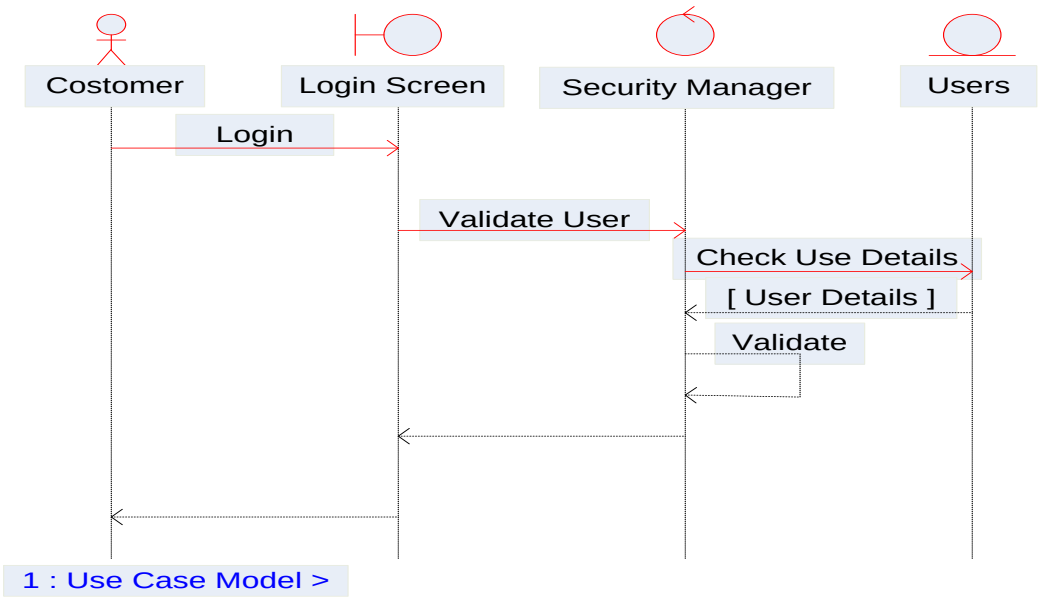
Gambar II.4. Contoh Class Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan

diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.5. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

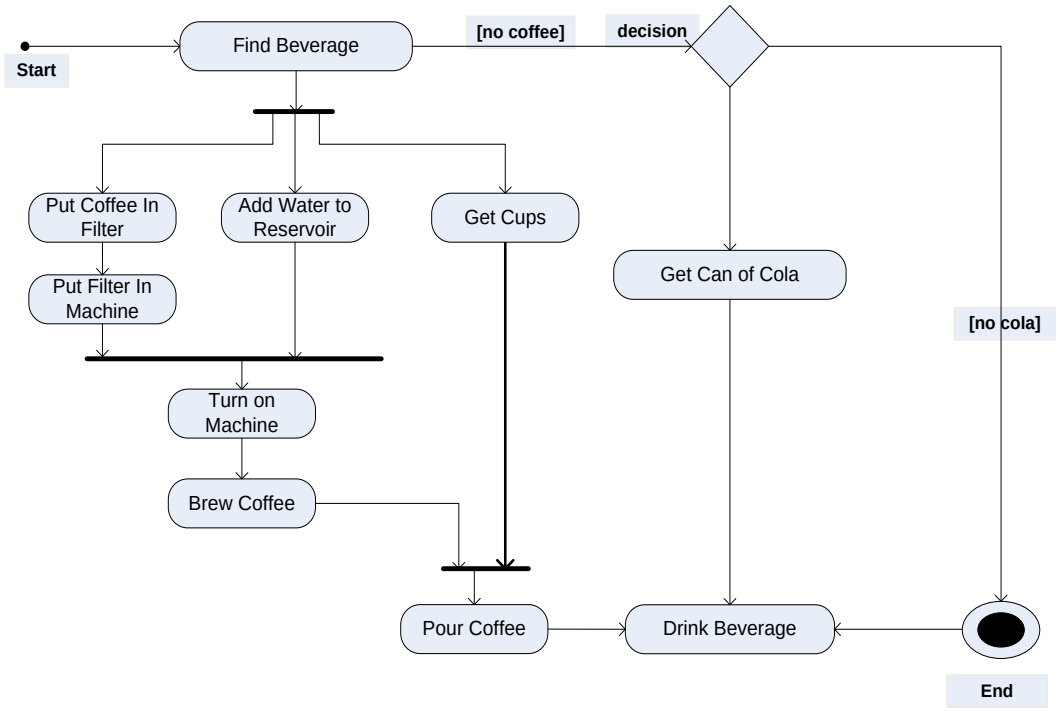
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.6. Activity Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)