BAB II

LANDASAN TEORI

II.1 Pengertian Perancangan

Perancangan sistem adalah suatu proses kegiatan untuk menyusun atau mengembangkan sistem informasi yang baru. Dalam tahap ini harus dapat dipastikan bahwa semua persyaratan untuk menghasilkan sistem informasi dapat dipenuhi. Hal pertama menentukan secara tepat dan terperinci operasional manajemen yang berkaitan dengan kegiatan pengolahan data. Kedua mengatur semua kebutuhan dan membaginya secara sistematis pada bagian yang nantinya akan dioperasionalkan. Dan selanjutnya untuk menentukan cara pelaksanaan dari masing-masing jenis tugas tersebut (Solamo, 2003).

Tahap-tahap perancangan sistem informasi adalah sebagai berikut :

1. Menyiapkan rancangan sistem yang terinci
2. Mengidentifikasi berbagai alternatif konfigurasi sistem
3. Mengevaluasi berbagai alternatif konfigurasi sistem
4. Memilih konfigurasi terbaik
5. Menyiapkan usulan penerapan
6. Menyetujui atau menolak penerapan sistem

Model perancangan sesungguhnya adalah model objek yang mendekripsikan realisasi fisik *use case* dengan cara berfokus pada bagaimana

spesifikasi-spesifikasi kebutuhan fungsional dan *non*-fungsional, bersama dengan batasan-batasan lain yang berhubungan dengan lingkungan implementasi.

II.2 Pengertian Aplikasi

Secara istilah aplikasi adalah suatu program yang dibuat untuk siap digunakan dalam melaksanankan suatu fungsi bagi pengguna jasa aplikasi tersebut serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus computer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan bagi penggunanya. Menurut Kamus Besar Bahasa Indonesia, “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. (Juansyah, 2015 : 2).

II.3 Keamanan

Keamanan adalah keadaan bebas dari bahaya. Istilah ini dapat digunakan dengan hubungan kepada kejahatan, dan segala bentuk kecelakaan. Keamanan merupakan topik yang luas termasuk keamanan nasional terhadap serangan teroris, keamanan komputer terhadap hacker, keamanan rumah terhadap maling dan penyusup lainnya, keamanan financial terhadap kehancuran ekonomi dan banyak situasi yang berhubungan lainnya. Host komputer yang terhubung ke network, mempunyai ancaman keamanan lebih besar dari pada host yang tidak terhubung ke jaringan lain. Dengan mengendalikan network security resiko tersebut dapat dikurangi (Kurniadi, 2016 : 2).

Masalah keamanan merupakan salah satu aspek penting dari sebuah sistem informasi. Sayang sekali masalah keamanan ini sering kali kurang mendapat perhatian dari para pemilik dan pengelola sistem informasi. Seringkali masalah keamanan berada di urutan kedua, atau bahkan di urutan terakhir dalam daftar hal hal yang dianggap penting. Apabila menggangu performa dari sistem, seringkali keamanan dikurangi atau ditiadakan.

II.4 Kriptografi (*Criptography)*

Kriptografi berasal dari bahasa Yunani, “kryptos” yang berarti tersembunyi dan “graphein” yang berarti tulisan. Kriptografi dapat diartikan menjadi “tulisan tersembunyi”. Sehingga dapat disimpulkan, Kriptografi adalah ilmu matematika yang berhubungan dengan transformasi data agar arti dari data tersebut menjadi sulit untuk dipahami, mencegahnya dari perubahan tanpa izin, atau mencegahnya dari penggunaan yang tidak sah. Jika transformasinya dapat dikembalikan, kriptografi juga dapat diartikan sebagai proses mengubah kembali data yang terenkripsi menjadi bentuk yang mudah dipahami. (Kurniadi, 2016 : 2).

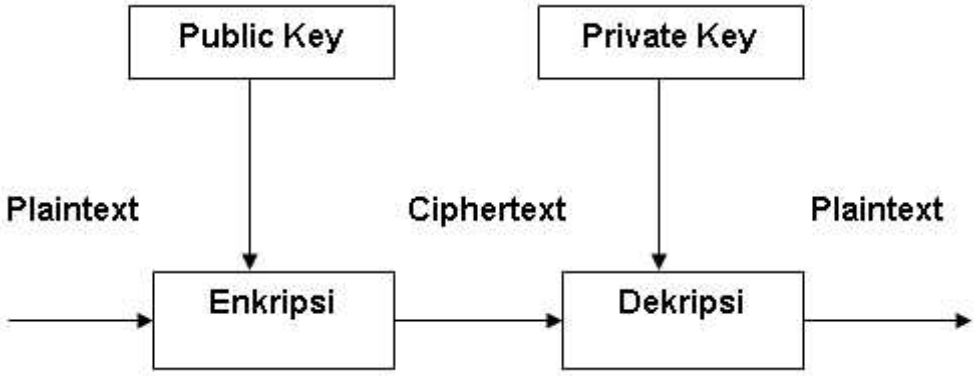
Criptography dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan dikirim dari suatu tempat ke tempat lain, isi pesan tersebut mungkin dapat disadap oleh pihak lain yang tidak berhak untuk mengetahui isi pesan tersebut. Untuk menjaga pesan, maka pesan tersebut dapat diubah menjadi suatu kode yang tidak dapat dimengerti oleh pihak lain (Zain, 2012 : 2).

Algoritma kriptografi berkembang terus dan terbagi atas dua bagian yaitu algoritma kriptografi klasik dan modern. Pada kriptografi klasik menggunakan algoritma sederhana, yang memungkinkan cipherteks dapat dipecahkan dengan mudah (melalui penggunaan statistik, terkaan, intuisi, dan sebagainya). Algoritma kriptografi modern dibuat sedemikian kompleks sehingga kriptanalis sangat sulit untuk memecahkan cipherteks tanpa mengetahui kunci. Pengelompokan algoritma ini juga dilakukan berdasarkan kunci enkripsi – dekripsi yang digunakan, yaitu symmetric cryptosystem atau simetris (menggunakan kunci yang sama untuk proses enkripsi – dekripsi) dan Assymmetric cryptosystem atau asimetris (menggunakan kunci yang berbeda untuk proses enkripsi – dekripsi). (Basri, 2016 : 2).

II.4.1 Enkripsi dan Dekripsi

Proses penyandian pesan dari *plaintext* ke *ciphertext* dinamakan enkripsi/*enchipering*. Sedangkan proses mengembalikan pesan dari *chipertext* ke *plaintext* dinamakan deskripsi/*dechipering*. Proses enkripsi dan deskripsi ini dapat diterapkan pada pesan yang dikirim ataupun pesan yang disimpan. Algoritma Kriptografi dari setiap kriptografi klasik selalu terdiri dari dua bagian yaitu enkripsi dan dekripsi (Basri, 2016 : 2).

Secara sederhana proses kriptografi dapat digambarkan sebagai berikut :



Gambar II.1 Kriptografi Secara Umum

(Sumber : Basri ; 2016)

Ada dua cara yang paling dasar pada kriptografi klasik. yaitu adalah Transposisi dan Subsitusi :

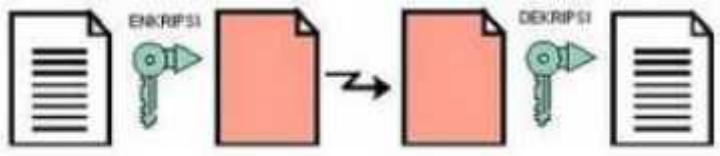
1. Transposisi adalah mengubah susunan huruf pada plaintext sehingga urutannya berubah. Contoh yang paling sederhana adalah mengubah suatu kalimat dengan menuliskan setiap kata secara terbalik.
2. Substitusi yaitu setiap huruf pada plaintext akan digantikan dengan huruf lain berdasarkan suatu cara atau rumus tertentu.

II.4.2 *Symmetric Cryptosystem*

Symmetric cryptosystem atau kriptografi simetris atau disebut juga algoritma kriptografi konvensional adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi. Algoritma kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu *bit* atau satu *byte* data. Sedangkan pada

algoritma blok, proses penyandiannya berorientasi pada sekumpulan *bit* atau *byte* data (per blok) (Basri, 2016 : 3).

Kriptografi simetris adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk membuat pesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut, termasuk pihak-pihak yang tidak diinginkan dapat membuat dan membongkar rahasia *ciphertext*. Problem yang paling jelas disini terkadang bukanlah masalah pengiriman *ciphertext*-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Contoh algoritma kunci simetris yang terkenal adalah DES (*Data EncryptionStandard*) dan RC-4, sebagaimana ditunjukkan pada gambar 3 berikut (Basri, 2016 : 3).:



Gambar II.2 Kunci Simetris

(Sumber : Basri ; 2016)

Ada beberapa kelebihan menggunakan kunci simetris yang sudah diketahui yaitu (Basri, 2016 : 3) :

1. Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetrik walupun hal ini berbanding lurus dengan penambahan ukuran file,
2. Kecepatan proses enkripsi/dekripsi bergantung pada besarnya ukuran file, semakin besar ukuran file semakin banyak waktu yang dibutuhkan untuk enkripsi/dekripsi,

3. Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem real-time.

Namun terdapat pula kelemahannyya, yaitu untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut, dan permasalahan dalam pengiriman kunci itu sendiri yang disebut " key distribution problem".

II.5 Algoritma RC4

RC4 merupakan salah satu jenis stream cipher, yaitu memproses unit atau input data yang pada umumnya sebuah byte atau bahkan kadang kadang bit (byte dalam hal RC4). Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah input data, pesan atau informasi tertentu sebelum diproses, atau menambahkan byte tambahan untuk mengenkrip (Zain, 2012 : 3).

Algoritma RC4 memiliki dua fase, setup kunci dan pengenkripsian. Setup untuk kunci adalah fase pertama dan yang paling sulit dalam algoritma ini setup Sbit kunci (S merupakan panjang dari kunci), kunci enkripsi tersebut digunakan untuk menghasilkan variabel enkripsi yang menggunakan dua buah array, state dan kunci dan sejumlah-S hasil dari operasi penggabungan. Operasi penggabungan ini terdiri dari pemindahan (swapping) byte, operasi modulo dan rumus lain. Operasi modulo merupakan proses yang menghasilkan nilai sisa dari satu pembagian. Sebagai contoh, 11 dibagi 4 adalah 2 dengan sisa pembagian 3, begitu juga jika tujuh modulo empat maka akan dihasilkan nilai tiga. Variabel enkripsi dihasikan dari setup kunci dimana kunci akan di XOR-kan dengan

plaintext untuk menghasilkan teks yang sudah terenkripsi. XOR merupakan operasi logik yang membandingkan dua bit biner. Jika bernilai beda maka akan dihasilkan nilai 1. Jika kedua bit sama maka hasilnya adalah 0. Kemudian penerima pesan akan mendekripnya dengan meng XOR-kan kembali dengan kunci yang sama agar dihasilkan pesan dari plaintext (Dermawan, 2014 : 2).

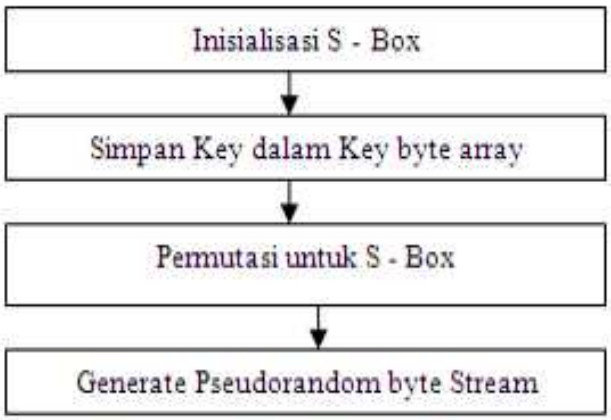
Proses Inisialisasi SBox (Array S), Secara pseudocode langkah ini dapat dituliskan sebagai berikut :

```
for i from 0 to 255
 S[i] := i
endfor
j := 0
for i from 0 to 255
 j := (j + S[i] + key[i mod keylength]) mod 256
 swap values of S[i] and S[j]
endfor
```

Masukan i dari dengan nilai dari 0 sampai 255 setelah itu nilai i menjadi nilai di dalam S[i], kemudian inisialisasikan j dengan 0, kemudian masukan kedalam rumus j menjadi 0 di tambah S[i] yang di beri nilai tadi antara 0 sampai 255 kemudian panjangan dari sebuah text yang akan di masukan di mod kan dengan 256, kemudian setelah di dapat S[i] di tukar kan dengan nilai S[j] yang di dapat di dalam perhitungan di atas.

```
i := 0
j := 0
while GeneratingOutput:
i := (i + 1) mod 256
j := (j + S[i]) mod 256
swap values of S[i] and S[j]
K := S[(S[i] + S[j]) mod 256]
output K
endwhile
```

Step selanjutnya inisialisasikan kembali i dan j dengan angka 0 kemudian di lakukan perhitungan untuk keluaran yang akan di hasilkan dengan memasukan variabel i dan j ke dalam masing-masing rumus yang telah di sediakan, kemudian di lakukan pertukaran kembali antara S[i] dan S[j] sampai teks yang akan di enkrip selesai, untuk mendapatkan nilai keystream nya masukan kembali nilai s dan j nya kemudian di mod kan dan akan mendapatkan hasil dari nilai keystream tersebut. Kemudian langkah yang terakhir yang di lakukan adalah operasi XOR k dengan plaintext nilai k yang sudah didapatkan dari langkah di atas kemudian dimasukkan dalam operasi XOR dengan plaintext yang ada, dengan sebelumnya pesan dipotong-potong terlebih dahulu menjadi byte-byte. Setelah operasi ini dilakukan, langkah satu kembali dilakukan untuk mendapatkan indeks baru dari setiap elemen S. Hal ini dtunjukan oleh Gambar II.3 berikut.



Gambar II.3 Bagan Alir Algoritma RC4

(Sumber : Halim Dermawan ; 2014)

II.6. Text Editor

Text Editor adalah sebuah software aplikasi atau program komputer yang memungkinkan penggunanya membuat, mengubah, atau mengedit file teks. Text Editor dapat digunakan untuk membuat program komputer, mengubah source code bahasa pemograman, serta membuat halaman web atau template web design (Firman, 2016:1)

II.7. *Android*

Android merupakan sistem operasi berbasis *Linux* yang bersifat terbuka (*open source*) dan dirancang untuk perangkat selular layer sentuh seperti smartphone dan komputer tablet. *Android* dikembangkan oleh Android, inc. dengan dukungan financial dari google yang kemudian dibeli pada tahun 2005. *Android* dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Headset Alliance (Sherief Saibino, 2014 : 7).

II.7.1 Sejarah *Android*

Pada bulan Oktober 2003 Android, Inc. didirikan di Palo Alto, California, oleh Andy Rubin (pendiri Danger), Rich Miner (pendiri wildfire Communications, Inc.), Nick Sears (mantan VP T-Mobile), dan Chris White (kepala desain dan pengembangan antar muka WebTV) untuk mengembangkan perangkat smartphone yang lebih sadar akan lokasi dan preferensi penggunanya". Awal tujuan pengembangan Android yaitu untuk mengembangkan sebuah sistem operasi canggih yang ditujukan untuk kamera digital, namun pasar untuk

perangkat kamera digital tidak cukup besar, dan pengembangan Android lalu dialihkan bagi pasar smartphone untuk menyaingi Symbian dan windows Mobile (iPhone Apple belum dirilis saat itu). *Android* Inc. dioperasikan secara diam-diam, hanya diungkapkan para pengembangnya sedang menciptakan sebuah perangkat lunak untuk smartphone. Pada tahun yang sama, Rubin kehabisan uang. Steve Perlman, seorang teman dekat Rubin, meminjaminya $ 10.000 tunai dan menolak tawaran saham di perusahaan. Google mengakuisisi Android Inc. pada tanggal 17 Agustus 2005, menjadikannya sebagai anak perusahaan yang sepenuhnya dimiliki oleh Google. Pendiri Android Inc. seperti Rubin, Miner dan white tetap bekerja di perusahaan setelah diakuisisi oleh Google. Di Google, tim yang dipimpin oleh Rubin mulai mengembangkan platform smartphone menggunakan kemnel Linux. Google memasarkan platform tersebut kepada produsen perangkat seluler dan operator nirkabel, dengan janji bahwa mereka menyediakan sistem yang fleksibel dan bisa diperbarui. Google telah menyeleksi beberapa mitra perusahaan perangkat lunak dan perangkat keras, serta mengisyaratkan kepada operator seluler bahwa kerjasama ini terbuka bagi siapapun yang ingin berpartisipasi. Pada tanggal 5 November 2007, Open Handset Alliance (OHA) didirikan. OHA bertujuan untuk mengembangkan standar terbuka bagi perangkat seluler. Saat itu, *Android* diresmikan sebagai produk pertamanya: sebuah platform perangkat seluler yang menggunakan kemnel Linux versi 2.6. Telepon seluler komersial pertama yang menggunakan sistem operasi android adalah HTC Dream, yang diluncurkan 22 oktober 2008. Sejak tahun 2008, *Android* terus melakukan sejumlah pembaruan untuk meningkatkan kinerja sistem operasi. Setiap versi

utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut atau cemilan bergula: misalnya, versi 1.5 bernama Cupcake, yang kemudian diikuti oleh versi 1.6 Donut. Versi terbaru adalah 5.0 Lollipop, yang dirilis 15 oktober 2014. (Sherief Saibino, 2014 : 8).

II.7.2 Jenis *Android*

Awal sistem android yang dirilis yaitu Android beta pada bulan November 2007. Sedangkan versi komersial pertama, Android 1.0, dirilis pada September 2008. Sejak April 2009, versi Android yang dikembangkan diberi kode nama yang berdasarkan makanan pencuci mulut dan makanan manis. Tiap versi dirilis sesuai urutan alphabet , yakni :

Tabel. II.1 *Versi Android*

Versi	Nama	Rillis	Catatan
1.0	*Android 1.0*	23 September 2008	Android pertama hanya untuk *smartphone*
1.1	*Android 1.1*	9 Februari 2008	
1.5	*Cupcake*	30 April 2009	Mulai pakai kode nama
1.6	*Donut*	15 September 2009	
2.0-2.1	*Éclair*	26 Oktober 2009 (2.0) 12 Januari 2010 (2.1)	
2.2	*Froyo*	20 Mei 2010	
2.3	*Gingerbread*	6 Desember 2010	Digunakan pada *smartphone* jenis lama
3.0-3.2	*Honeycomb*	22 Februari 2011 (3.0) 10 Mei 2011 (3.1) 15 Juli 2011 (3.2)	Hanya untuk tablet
4.0	*ICS (Ice Cream Sandwich)*	19 Oktober 2011	*Smartphone* dan tablet
4.1-4.3	*Jelly Bean*	9 Juli 2012 (4.1) 13 November 2012 (4.2) 24 Juli 2013 (4.3)	*Update* untuk memperbaiki dan menembah *fitur* pada *ICS*
4.4	*Kit kat*	3 September 2013	

5.0	*Lollipop*	12 november 2014 (5.0) 9 Maret 2015 (5.1)	
6.0	*Marshmallow*	5 oktober 015	Terdapat *daze mode*, *Do Not Disturb mode*, mendukung *USB tipe C*, mndukung pembacaan *fingerprint*.

(Sumber : Sherief Saibino ; 2014)

II.7.3 ADT (*Android Development Tools*)

ADT (Android Development Tools) pada *Eclipse* berfungsi untuk memudahkan kita dalam membuat aplikasi *project android*, membuat *GUI* aplikasi dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi dengan menggunakan *Android SDK* melalui *Eclipse. "ADT (Android Development Tools)* adalah *plugin* yang didesain untuk *IDE Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi *android* dengan menggunakan *IDE Eclipse"* (Safaat, 2014 : 6).

ADT (Android Development Tools) juga dapat berfungsi untuk melakukan pembuatan *package android (.apk)* yang dapat digunakan untuk distribusi aplikasi *android* yang telah dirancang. Pengembangan aplikasi *android* dengan menggunakan *ADT* di *Eclipse* sangat mudah dan dianjurkan. Semakin tinggi *platform android* yang digunakan, dianjurkan menggunakan *ADT* yang lebih terbaru, dikarenakan munculnya *platform* baru diikuti dengan versi *ADT* terbaru.

II.7.3 *Android SDK* (*Software Development Kit)*

Android SDK adalah tools *API* (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman *Java*. *Android SDK* merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang di *release* oleh *Google*. Saat ini disediakan *Android SDK* (*Software Development Kit*) sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman *Java*. Sebagai platform aplikasi-netral, *android* memberi anda kesempatan untuk membuat aplikasi yang dibutuhkan yang bukan merupakan aplikasi bawaan *Handphone* atau *Smartphone* (Safaat, 2014 : 1).

II.8. *Java*

Java merupakan bahasa pemrograman tingkat tinggi yang memiliki karakteristik *simple*, *object oriented*, *distributes*, *interpreted* dan memiliki performa yang tinggi. (Komputer, 2015 : 2).

Bahasa pemrograman *java* merupakan *compiler* sekaligus *interpreter*, dimana sebagai *compiler* program yang telah dibuat akan diubah menjadi *Java Bytecodes*. Sedangkan sebagai *interpreter*, *Java Bytecodes* tersebut dijalankan pada komputer. Pada bahasa pemrograman *Java* terdapat berbagai macam fitur yang disediakan oleh platform teknologi Java, yaitu *Java Virtual Machine* (*JVM*), *Garbage Collection* dan *Code Security*.

Gambar II.4 Tampilan Java

(Sumber : Komputer ; 2015)

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

UML merupakan kependekan dari *Unified Modeling Language* yaitu diagram dan metode standar untuk memodelkan dan merepresentasikan *object oriented software* dan sistem bisnis. (Mulyani, 2016 : 243).

Beberapa fungsi dan kegunaan dari UML yaitu (Mulyani, 2016 : 244) :

1. *Visualizing*

   Visualizing yaitu sebagai alat komunikasi konseptual model antara tim pengembang sistem (sistem analis dengan programmer)

2. *Specifying*

   Specifying yaitu sebagai *tools* yang digunakan untuk memodelkan sistem secara tepat dan jelas.

3. *Constructing*

   Constructing yaitu UML sebagai bahasa gratis mampu melakukan *mapping* dan konseptual model kedalam bahasa pemrograman.

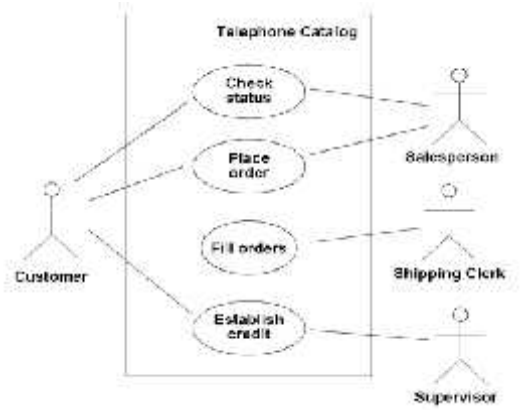
4. *Documenting*

Documenting yaitu UML digunakan sebagai *tools* untuk melakkukan dokumentasi teknis sebuah sistem.

Diagram-diagram yang terdapat dalam UML sangat banyak, berikut ini beberapa diagram yang sering digunakan dalam pengembangan sistem yaitu :

1) *Use case diagram*

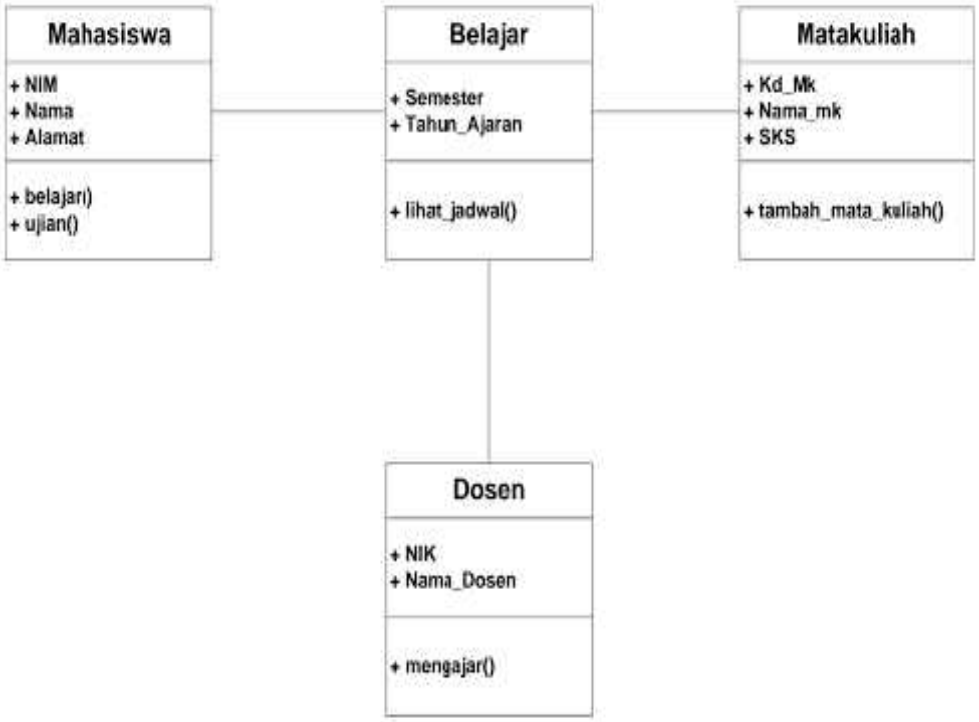
Use case diagram adalah diagram yang menggambarkan dan merepresentasikan aktor, *use cases* dan *dependencies* suatu proyek. *Use case diagram* dapat dilihat pada Gambar II.4 berikut :



Gambar II.5 *Use Case Diagram*
(Sumber : Mulyani ; 2016)

2. Class Diagram

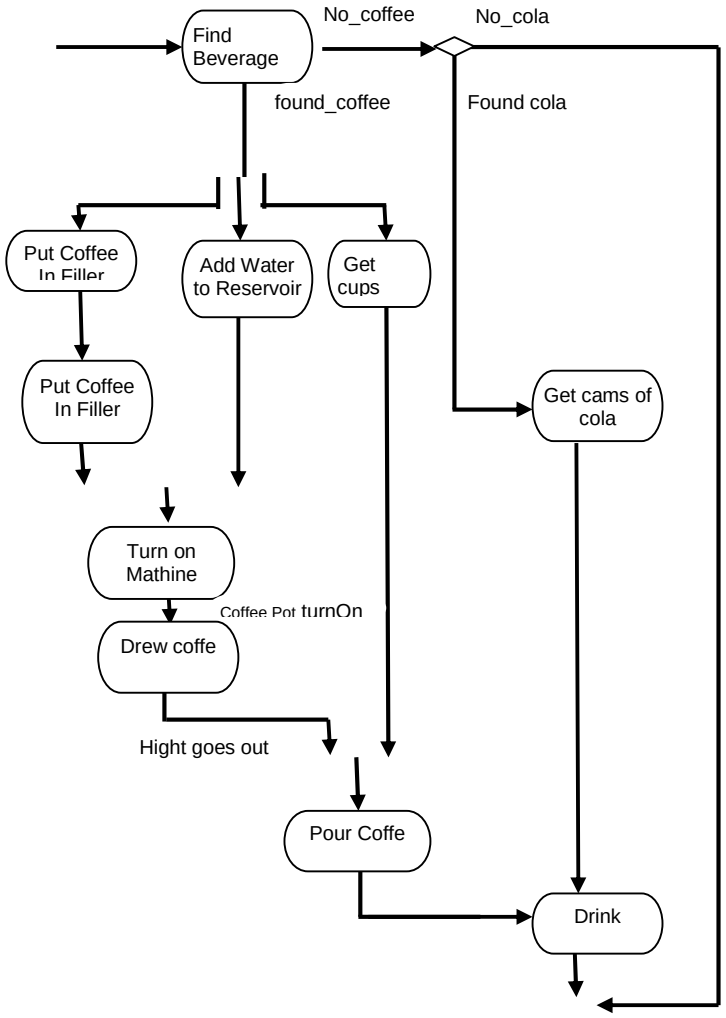
Class diagram adalah diagram yang digunakan untuk merepresentasikan kelas, komponen-komponen kelas dan hubungan antara masing-masing kelas. *Class diagram* dapat dilihat pada Gambar II.5 berikut :



Gambar II.6 ***Class Diagram***
(Sumber : Mulyani ; 2016)

3. Activity Diagram

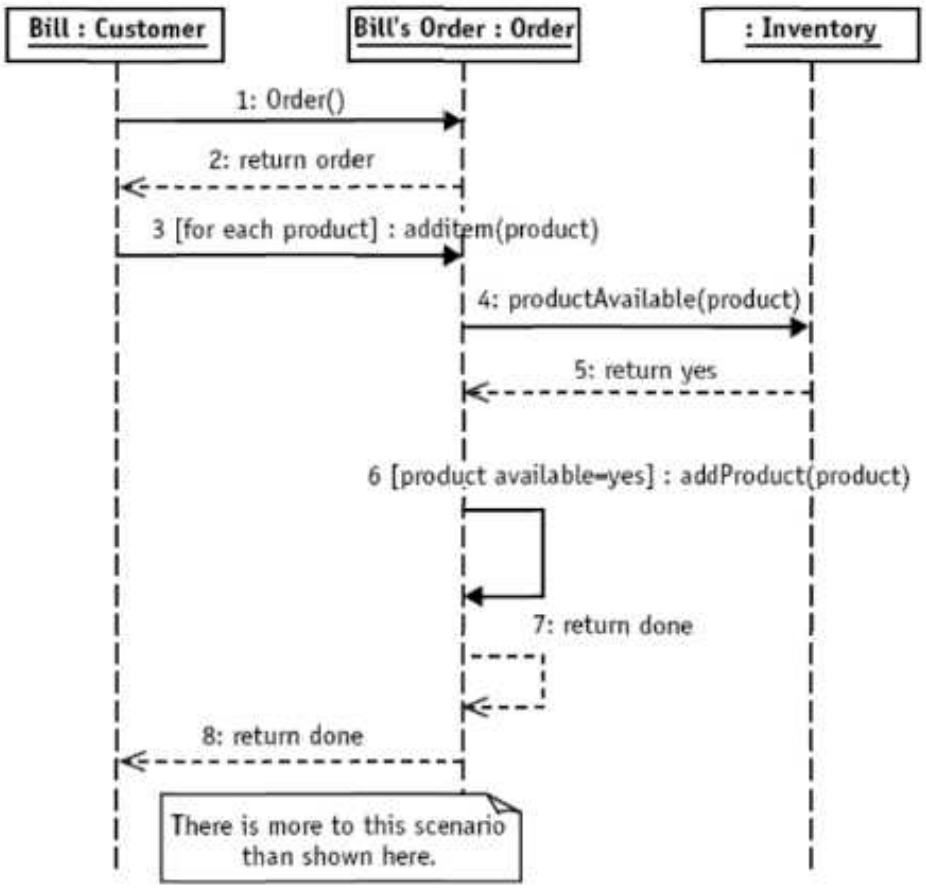
Activity diagram adalah diagram UML yang digunakan untuk menggambarkan alur aktifitas dari satu proses. *Activity diagram* memungkinkan siapapun yang melakukan proses untuk memilih urutan dalam melakukannya, dengan kata lain diagram hanya menyebutkan aturan-aturan rangkaian dasar yang harus kita ikuti. *Acitvity diagram* dapat dilihat pada Gambar II.6 berikut :



Gambar II.7 ***Activity Diagram***
(Sumber : Mulyani ; 2016)

4. Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan interaksi antar objek. *Sequence diagram* secara khusus menjabarkan *behavior* sebuah skenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek ini dalam sebuah *use case*. *Sequence diagram* dapat dilihat pada Gambar II.7 berikut :



Gambar II.8 ***Sequence Diagram***
(Sumber : Mulyani ; 2016)

II.10 *Normalisasi* (*Normalization*)

Normalisasi adalah salah satu cara untuk meminimalisir pengulangan data (*data redundancy*), normalisasi akan diperlukan jika ada indikasi bahwa tabel yang kita buat tidak baik (terjadi pengulangan informasi, potensi inkonsistensi data pada operasi pengubahan, tersembunyinya informasi tertentu dan lain sebagainya) dan diperlukan supaya jika tabel-tabel yang didekomposisi kita gabung kembali dapat menghasilkan tabel awal sebelum didekomposisi, sehingga

diperoleh tabel yang baik. Hasil dari normalisasi adalah himpunan-himpunan data (tabel-tabel) datam bentuk normal (normal form). (Mulyani, 2016 : 132)

Beberapa kegunaan normalisasi adalah (Mulyani, 2016 : 132):

1. Meminimalisir pengulangan data (*data redundancy*)
2. Memudahkan identifikasi *entity* objek.

Beberapa bentuk normal yaitu (Mulyani, 2016 : 132):

1. Bentuk Normal I (*First Normal Form / 1-NF*)

   Suatu relasi memenhi *1-NF* jika dan hanya jika setiap *attribute* dan relasi tersebut hanya memiliki nilai tunggal dalam 1 (satu) baris *record* (memisahkan group berulang).

2. Bentuk Normal II (*Second Normal Form/2-NF*)

   Suatu relasi memenuhi *2-NF* jika dan hanya jika memenuhi *1-NF* dan Setiap *attribute* yang bukan kunci utama tergantung secara fungsional terhadap semua attribute kunci dan bukan hanya sebagian attribute (menghilangkan ketergantungan fungsional pada sebagian/salah satu key).

3. Bentuk Normal Ill (*Third Normal Form/3-NF*)

   Suatu relasi memenhi *3-NF* jika dan hanya jika memenuhi *2-NF* dan setiap *attribute* bukan kunci tidak tergantung secara fungsional kepada *attribute* bukan kunci yang lain dalam relasi tersebut.

4. Boyce-Codd Normal Form (BCNF)

   Suatu relasi memenhi BCNF jika untuk setiap functional dependency terhadap setiap atribut atau gabungan atribut dalam bentuk : X -> Y maka X adalah super key. Tabel tersebut harus di-dekomposisi berdasarkan functional

dependency yang ada, sehingga X menjadi super key dari tabel-tabel hasil dekomposisi. Setiap tabel dalam BCNF merupakan 3NF. Akan tetapi setiap 3NF belum tentu termasuk BCNF. Perbedaannya, untuk functional dependency X -> A, BCNF tidak membolehkan A sebagai bagian dari primary key.

5. Bentuk Normal IV (*Fourth Normal Form/4-NF*)

   Suatu relasi memenhi *4-NF* jika dan hanya jika memenuhi *BCNF* dan tabel tersebut tidak boleh memiliki lebih dari sebuah *multivalued attribute*. Untuk setiap *multivalued attribute* (*MVD*) juga harus merupakan *functional dependencies*.

Beberapa key dalam Normalisasi (Mulyani, 2016 : 132):

1. *Superkey* adalah sejumlah *attribute entity* yang dapat digunakan untuk mengidentifikasi objek secara unik.
2. *Candidate key* adalah *superkey* dengan jumlah attribute minimal dan dapat berdiri sendiri.
3. *Primary key* adalah *superkey* yang dipilih oleh database administrator.
4. *Foreign key* adalah *attribute* disuatu relasi (*tabel*) yang menjadi *primary key* di relasi (*label*) lain.

II.11 *Basis Data* (*Database*)

Database adalah kumpulan dar semua data yang diperlukan oleh sistem. Dengan menggunakan database, beberapa apalikasi berbeda bisa saling

terintegrasi, misalnya aplikasi keuangan, aplikasi kepegawaian dengan penggajian atau aplikasi persediaan. (Mulyani, 2016 : 148).

Suatu bangunan basis data memiliki jenjang sebagai berikut:

1. Karakter, merupakan bagian data terkecil yang berupa angka, huruf, atau karakter khusus yang membentuk sebuah *item* data atau *field*. Contoh A,B,X,Y,2,1,2,9,0,=,<,> dan sebagainya.
2. *Field/item*, merupakan representasi suatu atribut dan *record* (rekaman/tupel) yang sejenis yang menunjukkan suatu *item* dari data. Contoh *field* nama (berisi data nama-nama pegawai) dan lain sebagainya.
3. *Record*/rekaman/tupel: Kumpulan dari field membentuk suatu record atau rekaman. Record menggambarkan suatu unit data individu yang tertentu. Contoh: file pegawai, dimana tiap-tiap *record* berisi kumpulan data nama, alamat, departemen, yang dapat mewakili tiap-tiap data.
4. *File*, merupakan kumpulan dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Contoh *file* pegawai berisi data tentang semua yang berhubungan dengan pegawai seperti nama pegawai, alamat pegawai, departemen, yang dapat mewakili tiap-tiap data.
5. *Database*, merupakan kumupan dari *file* atau *tabel* yang membentuk suatu *database*. Contoh *database* pegawai berisi file pegawai dan sebagainya.

II.11.1 DBMS (*Database Management System)*

Database Management System adalah sistem *software* yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan kontrol akses ke database. (Mulyani, 2016 : 170).

DBMS adalah software yang berinteraksi dengan program aplikasi dan pengguna database Biasanya *DBMS* menyediakan fasilitas sebagai berikut :

1. *DDL* (*Data Definition Language*)

   DDL memungkinkan pengguna untuk menentukan tipe data dan struktur dan kendala pada data yang akan disimpan dalam *database*.

2. *DML*(*Data Manipulation Language*)

   Ini memungkinkan pengguna memasukkan, update, menghapus dan mengambil data dari *database* atau memanipulasi data bahasa (*DML*).

3. Memberikan akses kontrol ke *database* :

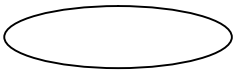
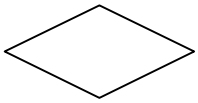
   1) Keamanan sistem: yang mencegah pengguna yang tidak berhak mengakses *database*.
   2) Integritas system: yang menjaga konsistensi data yang tersimpan dalam *database*.
   3) *Concurrency control system*: yang memungkinkan berbagi akses *database*.
   4) Pemulihan sistem *control*: yang mengembalikan *database* ke keadaan yang konsisten sebelumnya setelah *hardware* atau kegagalan *software*.
   5) *User*-diakses katalog, yang berisi deskripsi dari data yang tersimpan dalam *database*.

II.11.2 *Entity Relationship Diagram* (*ERD*)

Entity Relationship Diagram (*ERD*) digunakan untuk menggambarkan hubungan antara penyimpangan atau data storange yang dapat pada *DFD*, *ERD* menggunakan sejumlah notasi untuk menggambarkan struktur dan hubungan antar data. (Mulyani, 2016 : 100).

Notasi atau simbol-simbol yang digunakan pada *ERD* dapat dilihat pada Tabel II.2 seperti berikut.

Tabel II.2 Simbol *ERD*,

Simbol	Nama	Keterangan
	Entitas	Jenis entitas dapat berupa suatu elemen lingkungan, sumber daya atau transaksi yang field-fieldnya dipergunakan dalam aplikasi program
	Atribut	Atribut adalah karakteristik dari sebuah antitas
	Relasi	Menunjukan nama relasi antar satu entitas dengan entitas lainnya.
	Garis Relasi	Menunjukan hubungan (keterkaitan) antar entitas satu dengan lainnya.

(Sumber : Mulyani ; 2016)