BAB II

LANDASAN TEORI

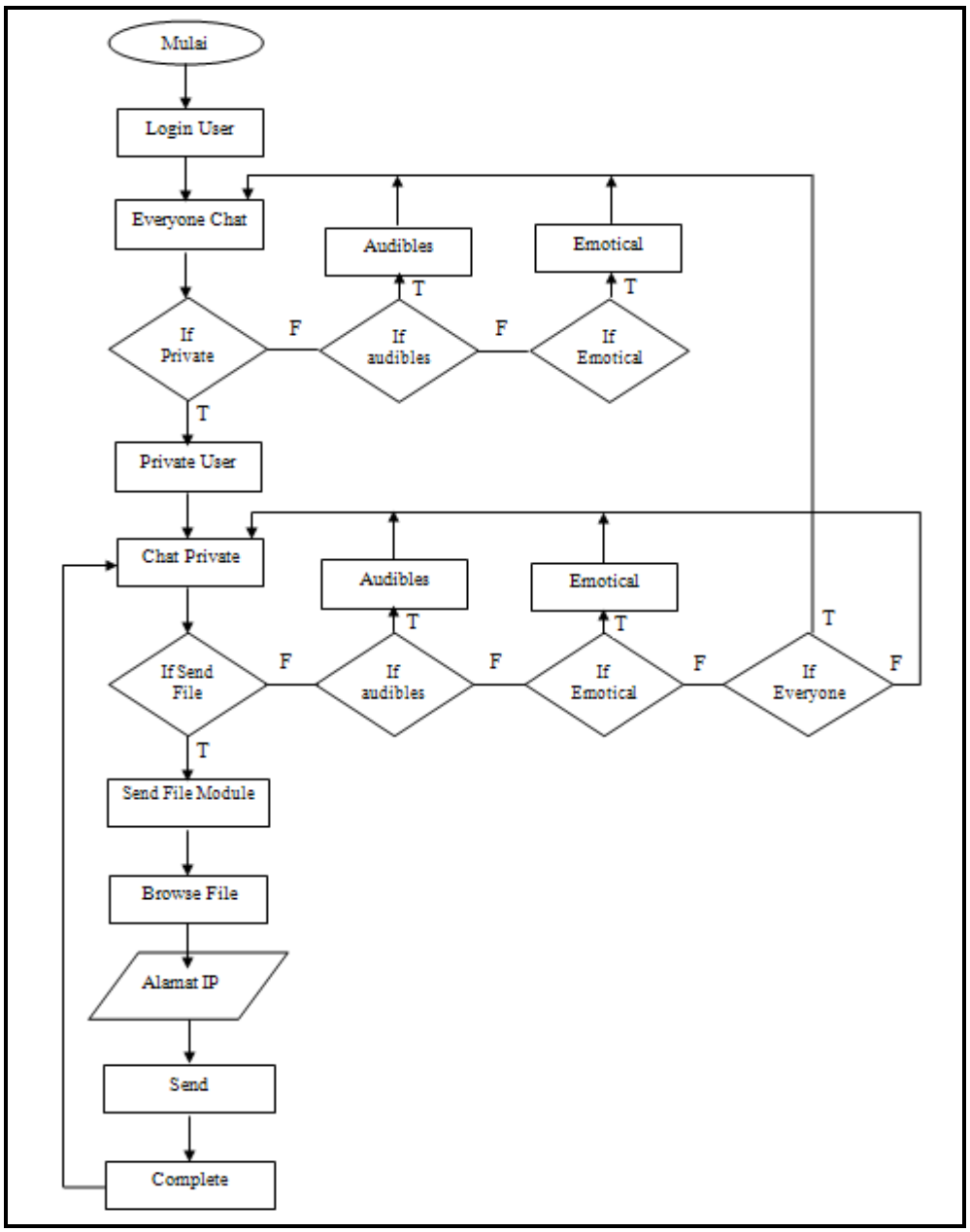
II.1. Aplikasi

Aplikasi adalah suatu *sub* kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yagn diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau pake disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki atarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah. (*Dahlan Abdullah ; 2013 : 152*)

Jenis-jenis *Software* Aplikasi :

1. *Software* aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, games dan sebagainya untuk hiburan.
2. *Software* aplikasi pendidikan yaitu *software* digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.

3. *Software* aplikasi bisnis yaitu *software* yang digunakan untuk aplikasi bisnis
4. *Software* aplikasi khusus
5. *Software* aplikasi untuk produtivitas kerja.



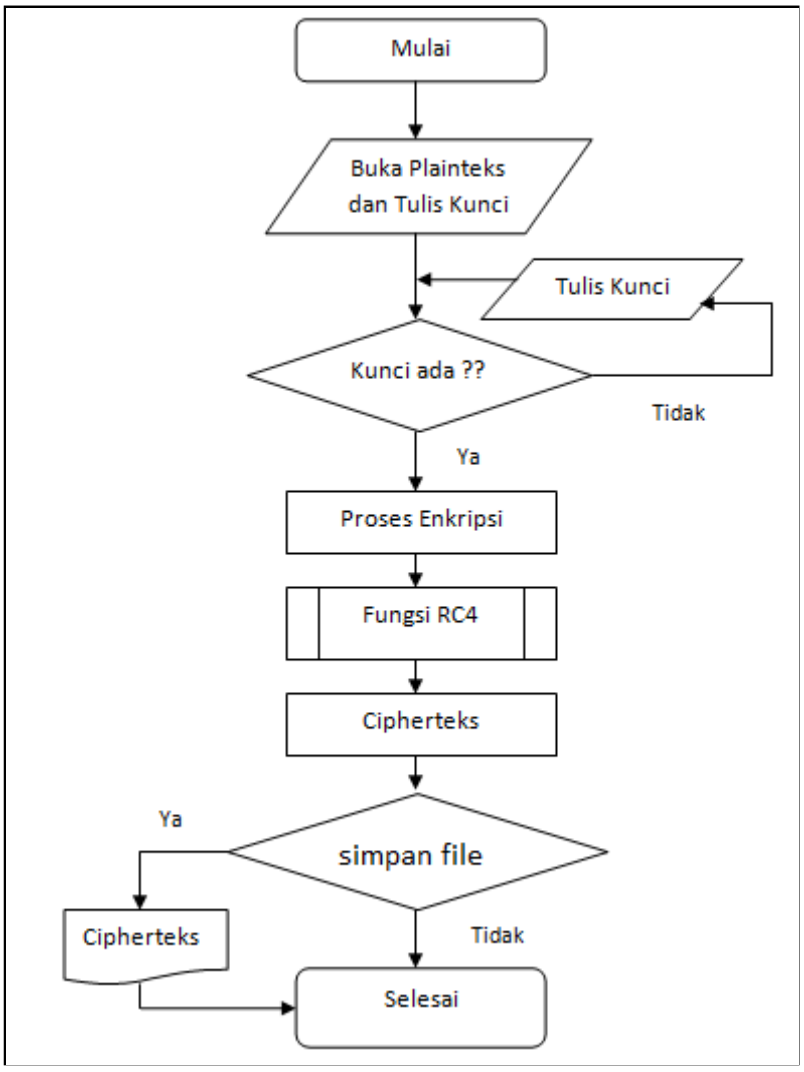
Gambar II.1. ***Disain Struktur Menu Aplikasi***
Sumber : Roni Setiawan ; 2010 : 4

II.2. Keamanan

Keamanan dan kerahasiaan data merupakan salah satu aspek terpenting dalam bidang komunikasi, khususnya komunikasi yang menggunakan media komputer. Salah satu bidang ilmu pengetahuan yang digunakan untuk mengamankan data adalah kriptografi. Kriptografi merupakan ilmu pengetahuan yang menggunakan persamaan matematis untuk melakukan proses enkripsi dan dekripsi data. Enkripsi merupakan proses untuk mengubah plainteks (data yang dapat dibaca) menjadi cipherteks (data yang tidak bisa dibaca) dan dekripsi merupakan kebalikan dari enkripsi, yaitu merubah cipherteks kembali menjadi plainteks. (*Busran ; 2012 : 32*)

II.3. Enkripsi

Enkripsi ialah proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan dan atau alat khusus. Cara yang digunakan untuk melakukan enkripsi ialah dengan cara melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti. Sebuah sistem pengkodean menggunakan suatu *table* atau kamus yang telah didefinisikan untuk mengganti kata dari informasi atau bagian informasi yang dikirim. Enkripsi dapat diartikan sebagai sebuah kode atau *cipher*. Sebuah *cipher* menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) *bit* dari sebuah pesan menjadi *cryptogram* yang tidak dapat dimengerti. (*Busran ; 2012 : 34*)



Gambar II.2. *Flowchart* Proses Enksipri
Sumber : Busran ; 2012 : 35

Terdapat tiga kategori enkripsi, yaitu sebagai berikut.

1. Kunci enkripsi rahasia.

   Dalam hal ini, terdapat sebuah kunci yang digunakan untuk mengenkripsi dan juga sekaligus mendekripsikan informasi.

2. Kunci enkripsi publik.

   Dalam hal ini, terdapat dua kunci yang digunakan, satu kunci untuk proses enkripsi dan kunci yang lain untuk proses dekripsi.

3. Fungsi *one-way* adalah suatu fungsi dimana informasi dienkripsi untuk menciptakan "*signature*" dari informasi asli yang bisa digunakan untuk keperluan autentikasi.

II.4. Kriptografi

II.4.1. Kriptografi *Secret Key*

Kriptografi *secret key* adalah kriptografi yang hanya melibatkan satu kunci dalam proses enkripsi dan dekripsi. Proses dekripsi dalam kriptografi *secret key* ini adalah kebalikan dari proses enkripsi. Kriptografi simetris dapat dibagi menjadi dua, yaitu penyandian blok dan penyandian alir. Penyandian blok bekerja pada suatu data yang terkelompok menjadi blok-blok data atau kelompok data dengan panjang data yang telah ditentukan. Pada penyandian blok, data yang masuk akan dipecah-pecah menjadi blok data yang telah ditentukan ukurannya. Penyandian alir bekerja pada suatu data bit tunggal atau terkadang dalam satu *byte*. Jadi format data yang mengalami proses enkripsi dan dekripsi adalah berupa aliran bit-bit data. (*Busran ; 2012 : 35-36*)

Contoh-contoh algoritma yang menggunakan kunci simetri yaitu :

1. *Data Encryption Standarad* (DES)
2. *Advanced Encryption Standard* (AES)
3. *International Data Encryption Algorithm* (IDEA)
4. *One Time Pad* (OTP)
5. A5
6. RC2

7. RC4
8. RC5
9. RC6
10. *Tiny Encryption Algorithm* (TEA)
11. *Twofish*

II.4.2. Kriptografi *Public Key*

Kriptografi *public key* sering disebut dengan kriptografi asimetris. Berbeda dengan kriptografi *secret key*, kunci yang digunakan pada proses enkripsi dan proses dekripsi pada kriptografi *public key* ini berbeda satu sama lain. Jadi dalam kriptografi *public key*, suatu *key generator* akan menghasilkan dua kunci berbeda dimana satu kunci digunakan untuk melakukan proses enkripsi dan kunci yang lain digunakan untuk melakukan proses dekripsi. Kunci yang digunakan untuk melakukan enkripsi akan dipublikasikan kepada umum untuk dipergunakan secara bebas. Oleh sebab itu, kunci yang digunakan untuk melakukan enkripsi disebut juga sebagai *public key*. Sedangkan kunci yang digunakan untuk melakukan dekripsi akan disimpan oleh pembuat kunci dan tidak akan dipublikasikan kepada umum. Kunci untuk melakukan dekripsi ini disebut *private key.* Dengan cara demikian, semua orang yang akan mengirimkan pesan kepada pembuat kunci dapat melakukan proses enkripsi terhadap pesan tersebut, sedangkan proses dekripsi hanya dapat dilakukan oleh pembuat atau pemilik kunci dekripsi.

Yang termasuk algoritma asimetri adalah :

1. *Digital Signature Algorithm* (DSA)
2. ***R****on Rivest, Adi* ***S****hamir dan Leonard* ***A****dleman* (RSA)

3. *Diffie Helman* (DH)
4. *Elliptic Curve Cryptography* (ECC)
5. Quantum

II.5. Algoritma *Triple Data Standart Encryption* (DES)

DES termasuk ke dalam system kriptografi simetri dan tergolong jenis *cipher* blok. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal (*internal key*) atau up-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) yang panjangnya 64 bit. Skema global dari algoritma DES adalah sebagai berikut:

1. Blok *plainteks* dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP).
2. Hasil permutasi awal kemudian di -*enciphering*- sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP-1) menjadi blok cipherteks.

Di dalam proses *enciphering*, blok plainteks terbagi menjadi dua bagian, kiri (*L*) dan kanan (*R*), yang masingmasing panjangnya 32 bit. Kedua bagian ini masuk ke dalam 16 putaran DES. Pada setiap putaran *i*, blok *R* merupakan masukan untuk fungsi transformasi yang disebut *f*. Pada fungsi *f*, blok *R* dikombinasikan dengan kunci internal *Ki*. Keluaran dai fungsi *f* di-XOR-kan dengan blok *L* untuk mendapatkan blok *R* yang baru. Sedangkan blok *L* yang baru

langsung diambil dari blok *R* sebelumnya. Ini adalah satu putaran DES.(*Rohmat Nur Ibrahim ; 2012 : 87-88*)

a. Enciphering

Proses *enciphering* terhadap blok plainteks dilakukan setelah permutasi awal (lihat Gambar 2.2). Setiap blok plainteks mengalami 16 kali putaran *enciphering*. Setiap putaran *enciphering* merupakan jaringan Feistel yang secara matematis dinyatakan sebagai:

$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

E adalah fungsi ekspansi yang memperluas blok *Ri* – 1 yang panjangnya 32-bit menjadi blok 48 bit. Menurut *Rohmat Nur Ibrahim* 2012 Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi pada tabel II.1 sebagai berikut:

Tabel II.1. Matriks Permutasi Ekspansi

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Sumber : (Rohmat Nur Ibrahim 2012*: 88)*

Selanjutnya, hasil ekpansi, yaitu *E*(*Ri* – 1), yang panjangnya 48 bit di-XOR-kan dengan *Ki* yang panjangnya 48 bit menghasilkan vektor *A* yang panjangnya 48-bit:

$E(Ri - 1) \oplus Ki = A$

Vektor *A* dikelompokkan menjadi 8 kelompok, masing-masing 6 bit, dan menjadi asukan bagi proses substitusi. Proses substitusi dilakukan dengan menggunakan delapan buah kotak-S (*S-box*), *S*1 sampai *S*8. Setiap kotak-S menerima masukan 6 bit dan menghasilkan keluaran 4 bit. Kelompok 6-bit pertama menggunakan S1, kelompok 6-bit kedua menggunakan S2, dan seterusnya.

b. Dekripsi

Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah *K*1, *K*2, …, *K*16, maka pada proses dekripsi urutan kunci yang digunakan adalah *K*16, *K*15, …, *K*1. Untuk tiap putaran 16, 15, …, 1, keluaran pada setiap putaran *deciphering* adalah :

$Li = Ri - 1$
$Ri = Li - 1 \oplus f(Ri - 1, Ki)$

Dalam hal ini, (*R*16, *L*16) adalah blok masukan awal untuk deciphering. Blok (*R*16, *L*16) diperoleh dengan mempermutasikan cipherteks dengan matriks permutasi IP-1. Pra-keluaran dari *deciphering* adalah adalah (*L*0, *R*0). Dengan permutasi awal IP akan didapatkan kembali blok plainteks semula. Tinjau kembali proses pembangkitan kunci internal. Selama *deciphering*, *K*16 dihasilkan dari (*C*16, *D*16) dengan permutasi PC-2. Tentu saja (*C*16, *D*16) tidak dapat diperoleh langsung pada permulaan *deciphering*. Tetapi karena (*C*16, *D*16) = (*C*0, *D*0), maka *K*16 dapat dihasilkan dari (*C*0, *D*0) tanpa perlu

lagi melakukan pergeseran bit. Catatlah bahwa (*C*0, *D*0) yang merupakan bit-bit dari kunci eksternal *K* yang diberikan pengguna pada waktu dekripsi. Selanjutnya, *K*15 dihasilkan dari (*C*15, *D*15) yang mana (*C*15, *D*15) diperoleh dengan menggeser *C*16 (yang sama dengan *C*0) dan *D*16 (yang sama dengan *C*0) satu bit ke kanan. Sisanya, *K*14 sampai *K*1 dihasilkan dari (*C*14, *D*14) sampai (*C*1, *D*1). Catatlah bahwa (*Ci* – 1, *Di* – 1) diperoleh dengan menggeser *Ci* dan *Di* dengan cara yang sama, tetapi pergeseran kiri (*left shift*) diganti menjadi pergeseran kanan (*right shift*).(*Rohmat Nur Ibrahim ; 2012 : 88*)

II.6. *Java*

Java memiliki cara kerja yang unik dibandingkan dengan bahasa perograman lainya yaitu bahasa perograman *java* bekerja mengunakan *interptreter* dan juga *compiler* dalam proses pembuatan program, *Interpreter java* dikenal sebagai perograman *bytecode* yaitu dengan cara kerja mengubah paket *class* pada *java* dengan *extensi*. *Java* menjadi *.class*, hal ini dikenal sebagai *class bytecode,* yaitunya *class* yang dihasilkan agar program dapat dijalankan pada semua jenis perangkat dan juga *platform*, sehingga program *java* cukup ditulis sekali namun mampu bekerja pada jenis lingkungan yang berbeda. (*Defni, Indri Rahmayun* ; 2014 : 64)

II.7. *NetBeans*

NetBeans merupakan salah satu *IDE* yang dikembangkan dengan bahasa pemrograman *java*. *NetBeans* mempunyai lingkup pemrograman java terintergrasi dalam suatu perangkat lunak yang di dalamnya menyediankan pembangunan

pemrograman *GUI, text editor, complier*, dan *interpreter.NetBeans* adalah sebuah perangkat lunak *open source* sehingga dapat digunakan secara gratis untuk keperluan komersial maupun nonkomersial yang didukung oleh *Sun Microsystem*. (*Atik Rusmayanti ; 2013 : 2-3*)

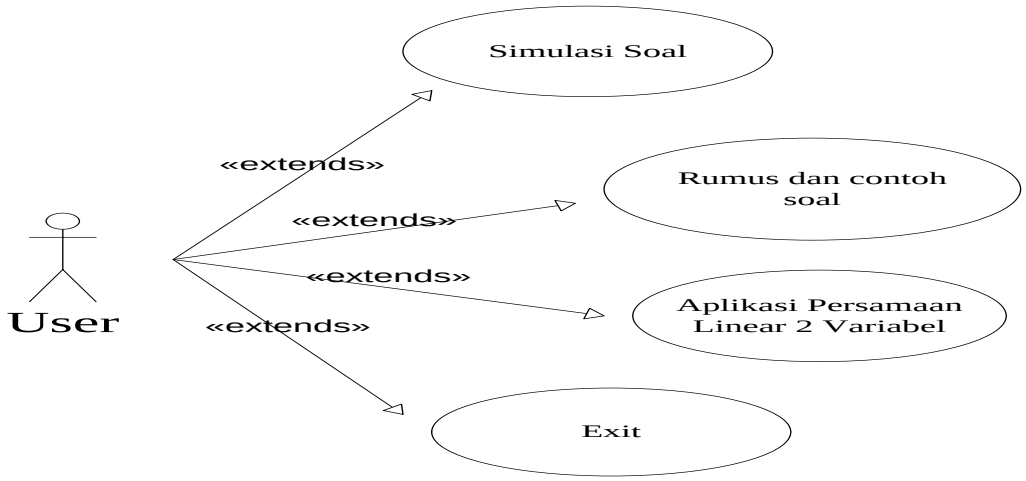
II.8. Unifed Modelling Language (UML)

UML (*Unified Modeling Language)* pertama kali diperkenalkan pada tahun 1990-an ketika Ivar Jacobson (sebelumnya terkenal dengan konsep OOSE-*Object Oriented Software Engineering),* Grady Booch (yang terkenal dengan konsep (OMT-*Object Modeling Technique),* James Rumbaugh (yang terkenal dengan notasi *Booch* yang populer digunakan sebagai salah satu metodologi analisis dan perancangan berorientasi objek) yang mulai mengadopsi ide-ide serta kemapuan-kemampuan tambahan dari masing-masing metodenya dan berusaha membuat metodologi terpadu yang kemudian dinamakan UML (*Unified Modeling Language).* UML merupakan metode pengembangan perangkat lunak (sistem informasi) dengan metode grafis yang relatif mudah.Usaha pengembangan UML dimulai pada Oktober 1994, ketika Rumbaugh bergabung dengan Booch di *Rational Software Corporation.* (*Agus Tiranda Tarigan ; 2014 : 10*)

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

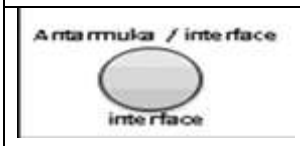
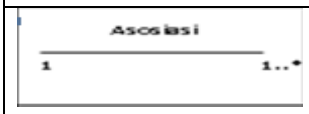
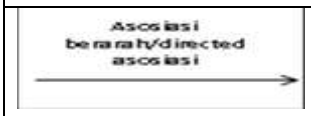
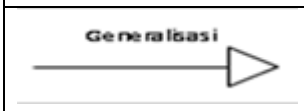
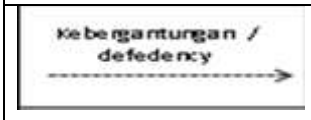
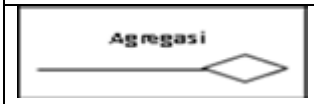
Identifikasi *use case* dilakukan untuk menentukan apa saja yang dilakukan oleh aktor dalam suatu sistem. Entitas eksternal yang menggunakan aplikasi ini hanya *user*. Aplikasi ini melayani satu actor, yaitu *user*. *User* dapat melakukan simulasi soal, rumus dan contoh soal, aplikasi persamaan linear 2 variabel, dan *exit*. Berikut dijelaskan pada gambar dibawah ini. Identifikasi *use case* dilakukan untuk menentukan apa saja yang dilakukan oleh aktor dalam suatu sistem. Entitas eksternal yang menggunakan aplikasi ini hanya *user*. Aplikasi ini melayani satu actor, yaitu *user*. *User* dapat melakukan simulasi soal, rumus dan contoh soal, aplikasi persamaan linear 2 variabel, dan *exit*. Berikut dijelaskan pada gambar dibawah ini. (*Agus Tiranda Tarigan ; 2014 : 10*)



Gambar II.3. Contoh *Use Case* Diagram
Sumber : Agus Tiranda Tarigan ; 2014 : 10

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 :

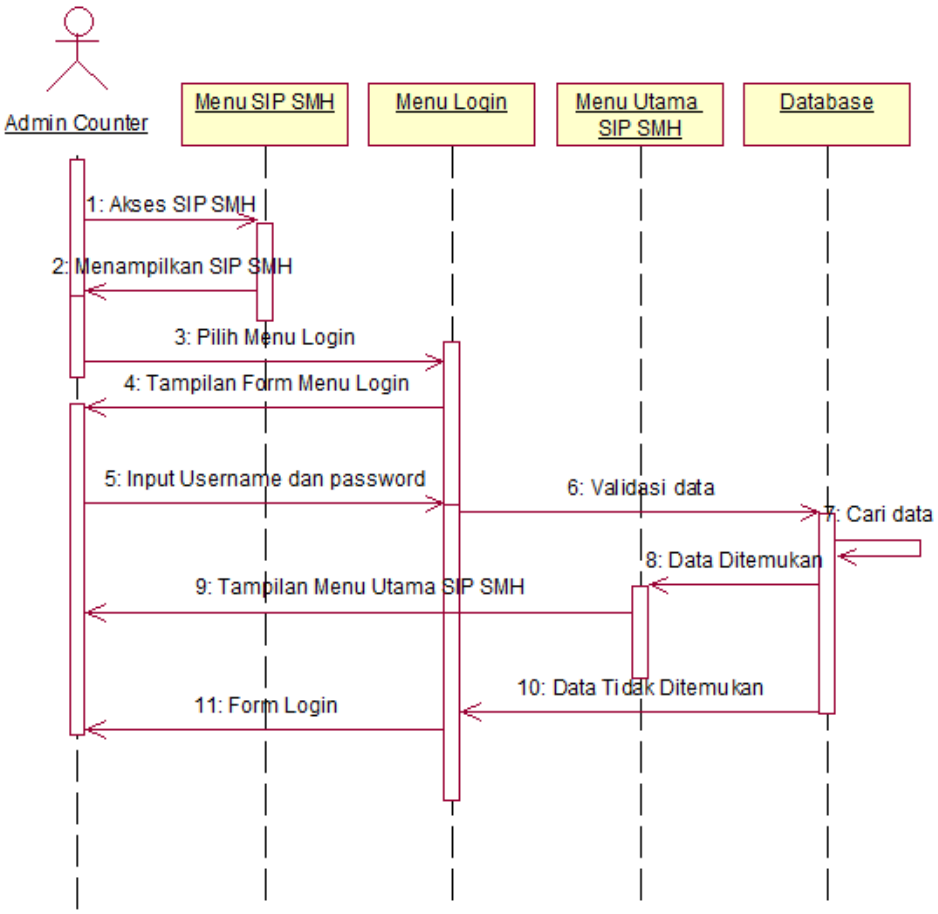
Simbol	Deskripsi
	Package merupakan suatu bungkusan dari satu atau lebih kelas
	Kelas pada struktur kelas
	Sama dengan konsep interface dalam pemrograman berorientasi objek
	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi juga disertai dengan multiplicity
	Relasi antar kelas dengan generalisasi-spesialisasi (umum-khusus)
	Relasi antar kelas dengan makna kebergantungan antar kelas
	Relasi antar kelas dengan makna semua-bagian (*whole- part*)

Gambar II.4. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)

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.6. Contoh *Sequence* Diagram
Sumber : Muhammad Rizal Firdaus 2013 : 5

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.

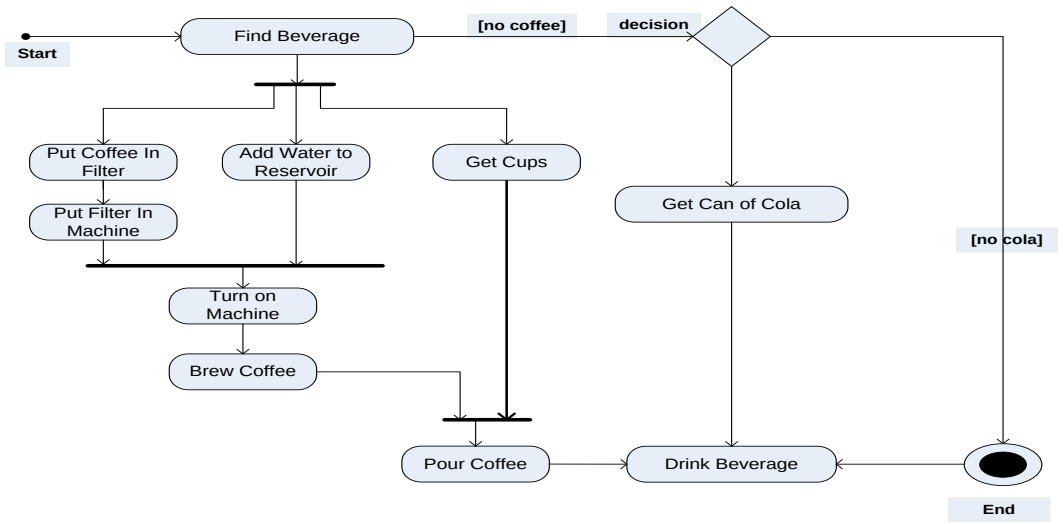
Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak

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.7. *Activify* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)