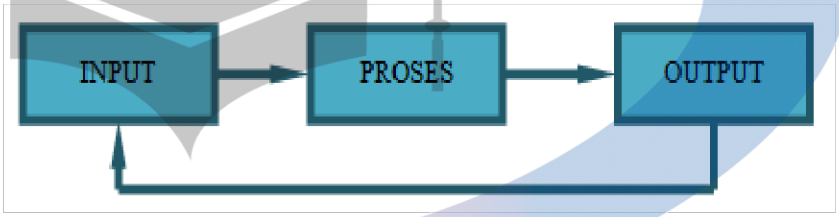
BAB II

TINJAUAN PUSTAKA

2.1 Konsep Sistem Informasi

2.1.1 Informasi

Informasi merupakan salah hasil dari pengolahan data dengan cara tertentu sehingga dapat bermanfaat bagi penerimanya. Lalu informasi akan di proses agar si penerima dapat memahami informasi yang telah diberikan tersebut. Sumber informasi merupakan data yang diolah melalui siklus pengolahan data (*Data Processing Life Cycle*). Informasi dapat dikatakan sangat berharga jika informasi tersebut dapat mengambil keputusan dengan cara yang baik [7].



Gambar 2.1 Siklus Pengolahan Data

1. Jenis-Jenis Informasi

   Beberapa jenis-jenis informasi yaitu sebagai berikut [7]:

a. *Absolute Information*

   Absolute information merupakan jenis informasi yang utama dari jenis informasi yang telah disampaikan dengan tanggungan dan tidak memerlukan penjelasan yang selanjutnya.

b. *Substitusional Information*

   Substitusional information merupakan jenis informasi yang memiliki konsep yang akan digunakan kepada beberapa informasi. Substitusional informasi sering juga disebut sebagai komunikasi.

c. *Philosophic Information*

   Philosophic information merupakan suatu jenis informasi yang berkaitan dengan konsep-konsep informasi yang menghubungkan informasi kepada pengetahuan dan kebijakan.

d. *Subjective Information*

Subjective information merupakan suatu jenis informasi ini berkaitan dengan reaksi dan informasi manusia. Informasi ini sangat bergantung pada orang yang telah menyampaikan informasi.

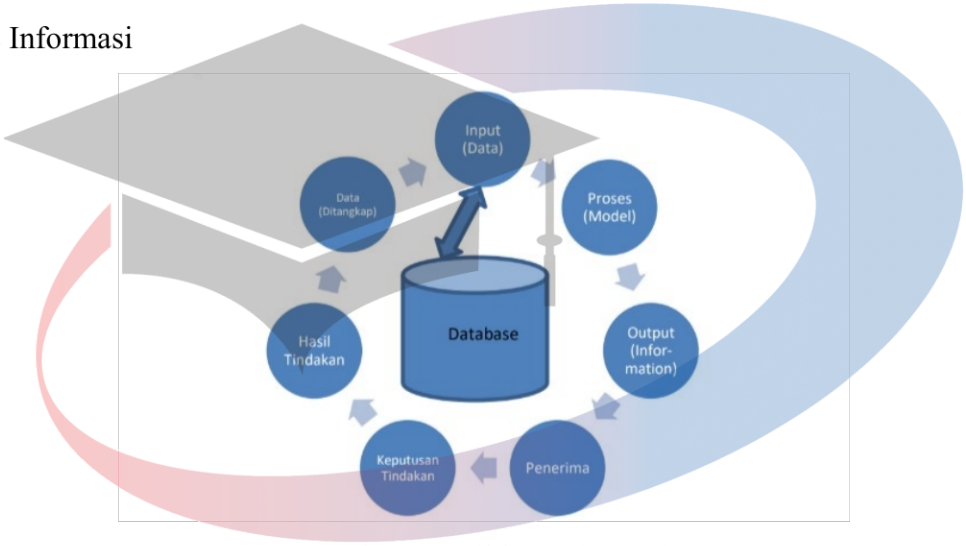
e. *Objective Information*

Objective information merupakan suatu Jenis informasi yang mengarahkan kepada informasi tertentu.

f. *Kultural Information*

Kultural Information merupakan suatu jenis informasi yang memberikan tekanan pada dimensi kultural.

2. Siklus Informasi



Gambar 2.2 Siklus Informasi

Data di *input* digunakan untuk menghasilkan suatu informasi data dan diolah sehingga akan menerima hasil. Dalam melakukan pengolahan data diperlukan model eksklusif dan akan menghasilkan suatu informasi yang sangat bermanfaat bagi si penerima dalam mengambil keputusan maupun melakukan aktivitas serta evaluasi. Data yang belum diolah akan disimpan yang dalam bentuk basis data. Data yang telah disimpan dapat diambil kembali ketika diolah dan akan menjadi sebagai informasi. Data tersebut akan menjadi *input*, lalu diproses dengan menggunakan model, sehingga menghasilkan *output* yang akan didapatkan oleh si penerima. Dalam membuat suatu keputusan akan melakukan tindakan dan akan membentuk sebuah siklus yang disebut sebagai siklus informasi (*information cycle*) [7].

3. Nilai Informasi (*Cost-Effectiveness*)

Suatu informasi yang telah ditentukan dengan dua hal diantaranya adalah manfaat dan biaya dalam mendapat suatu informasi. Sebuah informasi lebih bernilai jika bermanfaat dibandingkan dengan biaya untuk mendapatkannya Informasi tersebut. Nilai lain dari

Accuracy, Relevance, Timeliness, Cost-effectiveness, juga memiliki atribut yang lainnya yaitu sebagai berikut [7]:

a. *Completeness*

   Suatu informasi yang dapat menguraikan suatu hal yang harus diketahui dalam memahami situasi. Bertujuan untuk mengumpulkan selengkap mungkin mengenai informasi tersebut.

b. *Auditability*

   Suatu keahlian dalam melakukan pemeriksaan yang lengkap dan secara akurat dalam suatu informasi. Dalam melakukan penentuan keakuratan informasi yang akan membawa pernyataan dari kegunaan informasi serta kemampuan audit sangat diperlukan.

c. *Reliability*

   Suatu Informasi yang tidak akurat. Dengan nilai dari keenam atribut tersebut (*accuracy, relevance, timeliness, cost- effectiveness, anditability, reliability*) atribut reliabilitas dapat diambil nilainya.

4. Kualitas Informasi.

   Terdapat tiga aspek dalam kualitas informasi yaitu diantaranya akurat (*accurate*), tepat waktu (*timeliness*), dan relevan (*relevance*). Kualitas dari informasi (*quality of information*) dapat digambarkan dengan bentuk sebuah pilar yang dapat dilihat melalui gambar dibawah ini [7].

Gambar 2.3 Pilar Kualitas Informasi

Berikut penjelasan mengenai gambar yang berada pada bagian diatas yaitu sebagai berikut [7]:

1. Akurat (*Accuracy*)

Sebuah informasi yang didapatkan harus sempurna serta terhindar dari kesalahan-kesalahan. Serta sebuah informasi harus sinkron, tidak ada informasi yang palsu dan tidak ambigu ketika sampai kepada si penerima informasi.

2. Tepat waktu (*Timeliness*)

   Sebuah informasi yang akan di sampai kepada penerima dengan menggunakan waktu yang tepat, serta suatu informasi yang tidak bernilai merupakan informasi yang telah lama. Penyampaian informasi sangat simpel dan penerimaan juga cepat dalam memperoleh informasi membutuhkan teknologi serta informasi tersebut merupakan informasi yang terbaru.

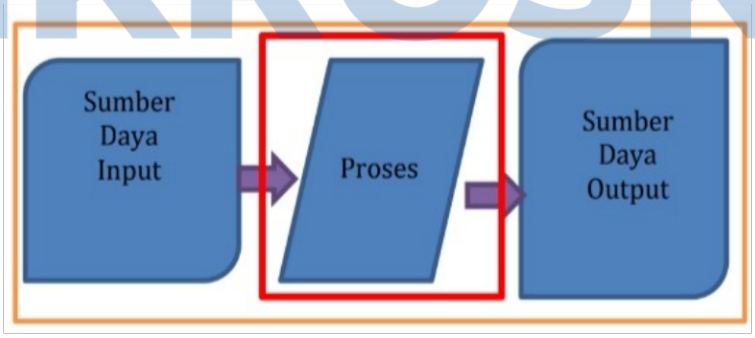
3. Relevan (*Relevance*)

   Sebuah informasi yang baik dan sangat berguna bagi penerima. Sebuah relevansi informasi terjadi saat perbedaan yang didapat oleh si penerima yang satu dengan penerima yang lainnya.

2.1.2 Sistem

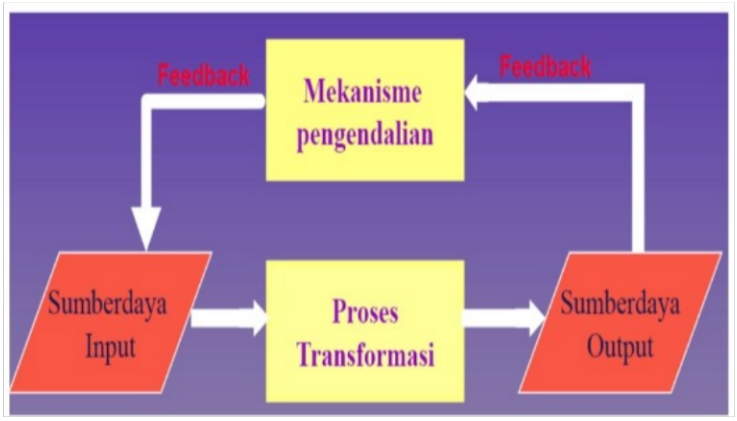
Sistem adalah suatu perangkat elemen yang mengelola berbagai berbentuk aktivitas-aktivitas maupun prosedur dengan tujuan yang sama dalam menjalankan data dengan target yang telah ditentukan sehingga dapat menghasilkan suatu informasi. Sistem mempunyai pendekatan pada sebuah prosedur kerja yang saling terhubung, dan bekerja untuk mendapatkan pencapaian tujuan yang diinginkan. Jenis sistem terbagi menjadi dua bagian dimana yang pertama dinamakan sebagai sistem terbuka dan yang kedua disebut sebagai sistem tertutup [7].

Sistem terbuka merupakan suatu hubungan dari proses sistem dengan daerah dan melalui sumber daya [7].



Gambar 2.4 Sistem Terbuka

Sistem tertutup merupakan suatu jenis sistem yang tidak dipengaruhi dengan pihak luar dengan mekanisme pengendalian [7].



Gambar 2.5 Sistem Tertutup

Beberapa karakteristik yang dimiliki oleh sistem yaitu sebagai berikut [7]:

1. Komponen (*Component*)

   Komponen merupakan suatu sistem yang terdiri dari beberapa komponen yang saling berkerja sama, diantaranya adalah melakukan interaksi dengan membangun dan saling bekerja sama.

2. Lingkungan Luar Sistem (*Environment*).

   Lingkungan merupakan pengaruh dari operasi sistem dari lingkungan luar sistem. Lingkungan luar yang bersifat menguntungkan harus dijaga dan yang bersifat merugikan harus dikendalikan.

3. Batasan Sistem (*Boundar*).

   Batasan sistem merupakan daerah sistem yang dibatasi oleh suatu sistem dengan sistem yang lainnya. Sistem menggunakan batas sistem lain yang sesuai dengan lingkaran daerahnya.

4. Penghubung Sistem (*Interface*).

   Penghubung sistem merupakan suatu alat bantu yang akan menghubungkan antara satu subsistem ke subsistem lainnya. Melalui penghubung sumber data dan memungkinkan mengalir berasal subsistem ke subsistem lain. Keluaran (*output*) asal subsistem ini akan menjadi masukkan (*input*) untuk bagian dari subsistem menggunakan alat bantu penghubung ini.

5. Masukkan Sistem (*Input*).

   Masukkan Sistem merupakan energi yang akan dimasukkan ke dalam sistem, dapat berupa perawatan (*maintenance input*), dan masukkan sinyal (*signal input*). *Maintenance input* merupakan suatu energi yang berkaitan dengan perawatan sistem. Kemudian perawatan

akan dimasukkan agar sistem bisa beroperasi. *Signal input* merupakan energi yang berpengaruh besar terhadap suatu proses yang akan dijalankan. Informasi dari sistem tersebut akan diteruskan untuk mendapatkan keluaran.

6. Keluaran Sistem (*Output*)

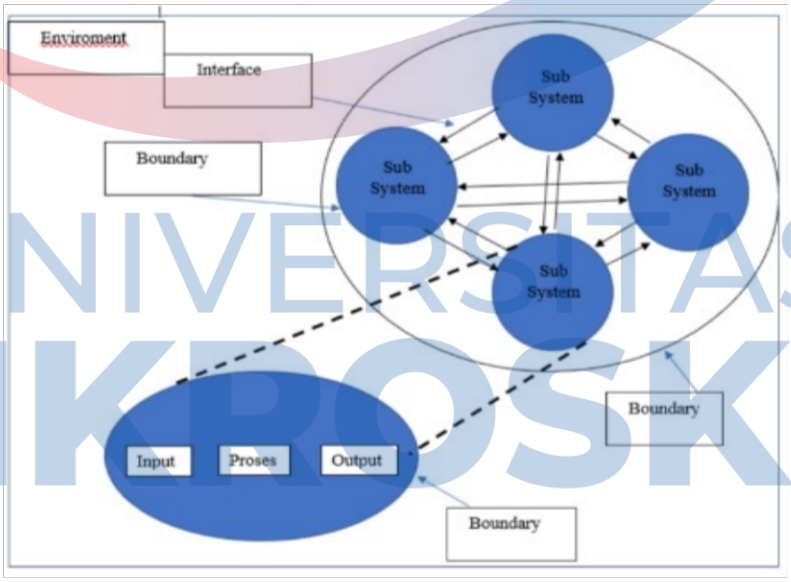
   Keluaran sistem merupakan energi yang akan menghasilkan masukan keluaran yang dibuang maupun dibutuhkan. Contoh seperti komputer mengeluarkan suhu panas yang merupakan energi dari pembuangan dan informasi sebagai keluaran energi yang akan dibutuhkan.

7. Pengolah Sistem

   Pengolahan sistem merupakan bagian dari proses yang akan mengubah *input* menjadi *output*. Seperti contohnya adalah sistem akuntasi dengan pengolahan data-data terkait dengan pengeluaran perusahaan dan berubah menjadi laporan-laporan keuangan.

8. Sasaran Sistem

   Sasaran sistem merupakan tujuan (*goal*) atau sasaran (*objective*). Sistem dari sasaran sangat berpengaruh dalam menentukan *input* yang akan dibutuhkan pada sistem, dan keluaran yang akan dihasilkan oleh sistem tersebut.



Gambar 2.6 Karakteristik Sistem Informasi

Beberapa klasifikasi dari sistem yaitu sebagai berikut [7]:

1. Sistem Abstrak (*abstract system*)

   Sistem Abstrak merupakan suatu sistem yang berasal dari inspirasi yang secara fisikal tidak terlihat. Model sistem berupa gagasan atau pendapat berupa hubungan antara manusia serta yang kuasa.

2. Sistem Fisik (*physical system*)

   Sistem fisik merupakan suatu sistem yang dapat ditinjau dan mempunyai bentuk fisiknya yang sesuai dengan kebutuhan.

3. Sistem Tertentu (*deterministic system*)

   Sistem eksklusif merupakan suatu sistem yang dapat berjalan dengan menggunakan langsung dapat diprediksi dengan absolut sehingga *output* nya juga absolut.

2.1 Rapid Application Development

Rapid Application Development (RAD) merupakan suatu proses strategi dari siklus hidup pengembangan dengan waktu pengerjaan yang lebih cepat dan dapat memberikan hasil yang sangat berkualitas dibandingkan dengan hasil yang akan di capai melalui siklus pengembangan yang lainnya. RAD merupakan salah satu gabungan dari beberapa teknik terstruktur dan juga dengan teknik *prototype* yang terkait dengan teknik dari sistem perancangan *application* dengan menggunakan metode RAD dapat mempersingkat waktu dari pengembangan aplikasi yang akan dirancang [6].

Gambar 2.7 Tahapan Pengembangan RAD

Berikut ini merupakan tahapan-tahapan pengembangan dari *Rapid Application Development* (RAD) antara lain yaitu sebagai berikut [6]:

1. *Requirements Planning* (Perencanaan Syarat-Syarat): pada tahapan ini pengguna dan penganalisis bertemu untuk mengidentifikasi tujuan-tujuan dari sistem aplikasi serta mengidentifikasi syarat-syarat mengenai informasi yang akan ditimbulkan dari tujuan-tujuan sistem aplikasi tersebut. Tujuan dari tahapan *requirements planning* adalah untuk menganalisis dan menyelesaikan masalah-masalah yang dihadapi oleh perusahaan.
2. *Workshop Design* (Proses Desain): pada tahapan ini adalah membuat rancangan dan memperbaiki gambaran dari *design workshop*. Penganalisis dan pemrogram yang dapat

bekerja dan membangun untuk menunjukkan representasi visual desain *prototype* dan pola kerja kepada pengguna. *Workshop design* ini dapat dilakukan selama beberapa hari tergantung dari ukuran aplikasi yang akan ditingkatkan. Selama *workshop* desain berlangsung pengguna dapat merespon desain dari sistem *prototype* yang sudah dirancang kemudian penganalisis dapat memperbaiki *prototype* yang dirancang berdasarkan dari respon pengguna.

3. *Implementation* (Implementasi): pada tahapan ini penganalisis dapat berkerja sama dengan para pengguna secara sungguh-sungguh selama *workshop design* dan merancang aspek-aspek bisnis dan nonteknis perusahaan. Setelah aspek-aspek disetujui dan sistem yang akan di bangun dan disaring, lalu sistem yang baru akan melakukan pengujian dan akan diperkenalkan kepada organisasi.

2.2 Teknik Pengembangan Sistem

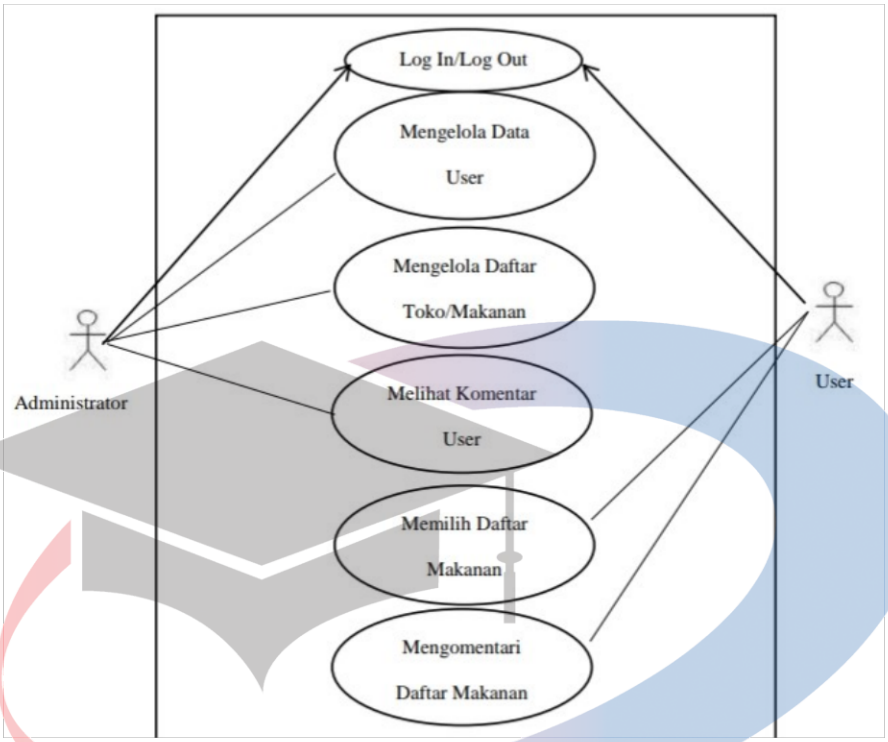
1.2.1 Use Case Diagram

Use Case Diagram adalah salah satu jenis model diagram *Unified Modelling Language* (UML) yang menggambarkan hubungan dari proses interaksi antara sistem dan aktor, *use case diagram* juga dapat mendeskripsikan tipe interaksi antara pengguna sistem dengan sistemnya. *Use case diagram* adalah jenis model dari gambaran grafis yang berasal dari beberapa aktor yang saling berhubungan antar sistem yang satu dengan sistem yang lainnya. Oleh karena itu terdapat sebuah hubungan antara aktor dengan sistem. *Use case diagram* bertujuan untuk menggambarkan secara singkat mengenai siapa saja yang akan menggunakan sistem dan apa saja yang akan dilakukan di dalam sistem tersebut. Serta mengetahui fungsi-fungsi apa saja yang terdapat di sistem aplikasi tersebut. Dan dapat merepresentasikan hubungan antara pengguna terhadap sistem, serta dapat mengetahui kebutuhan yang ada di luar sistem [8].

Manfaat *use case diagram* yaitu sebagai berikut [8].

1. Dapat Memudahkan hubungan antara penggunaan *user* yang satu ke bagian *user* yang lainnya.
2. Dapat memperlihatkan proses dari aktivitas-aktivitas secara beruntun dalam sistem aplikasi.
3. Mempermudah dalam melihat proses dari aktivitas sistem aplikasi yang akan dijalankan.
4. Memberikan pemahaman terhadap kebutuhan (*requirement*) terhadap sebuah sistem aplikasi.

5. Digunakan untuk mengidentifikasi siapa saja yang sedang melakukan interaksi dengan menggunakan sistem, dan dapat mengetahui apa saja yang akan dilakukan terhadap sistem tersebut.

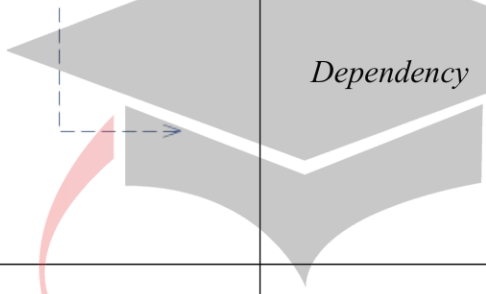


Gambar 2.8 Contoh Tampilan *Diagram Use Case*

Diagram use case diatas menjelaskan mengenai sistem aplikasi E Lelang sistem tersebut memiliki 2 *actor* yaitu: *actor* pertama adalah *administrator* serta *actor* kedua adalah *user*. Tugas-tugas yang akan dilakukan oleh *actor administrator* adalah: melihat list lelang barang, melakukan pencarian barang, meng-*input* harga penawaran. *Actor administrator* dapat melakukan *log in* maupun *log out,* mengelola data *user*, mengelola daftar toko makanan, dan dapat melihat komentar user. Sedangkan tugas yang akan dilakukan oleh *actor user* adalah: dapat melakukan *log in* maupun *log out,* Memilih daftar makanan, dan dapat memberikan sebuah komentar terhadap daftar makanan [9].

Tabel 2.1 Simbol-Simbol Dasar Use Case Diagram

NO	SIMBOL	NAMA	KETERANGAN

1.		*Actor*	Menggambarkan seorang tokoh yang mendeskripsikan hubungan antara kedudukan setiap peran dari sistem. Fungsi dari *actor* adalah sebagai simbol untuk memberikan informasi pada setiap sistem yang akan digunakan.
2.		*Dependency*	Korelasi dimana terdapat perubahan yang terjadi pada suatu elemen mandiri (*independent*) yang akan mempengaruhi elemen yang bergantung pada elemen yang tidak *independent*.
3.		*Generalization*	Korelasi dimana objek anak (*descendent*) membuatkan sikap dan struktur data yang berasal dari objek yang terdapat di atas objek induk (*ancestor*).
4.		*Include*	Menunjukkan bahwa *use case* asal dengan *use case* yang berada di seluruh sistem yang lainnya merupakan fungsionalitas dari *use case* yang lainnya.
5.		*Extend*	Menunjukkan bahwa *use case* sasaran memperluas sikap asal *use case* sumber pada suatu titik yang diberikan.
6.		*Association*	Simbol yang bertujuan untuk menghubungkan antara objek yang satu ke objek yang lainnya.

7.		*System*	Mendeskripsikan paket-paket yang akan menampilkan sistem secara terbatas.
8.		*Use Case*	Deskripsi yang berasal dari barisan aksi-aksi yang akan menampilkan sistem yang membentuk sesuatu yang akan terjadi pada *actor.*

Use Case Descriptive merupakan ilustrasi secara *general* mengenai fungsionalitas dari suatu proses usaha berupa *actor* yang melibatkan berjalannya suatu sistem. *Use case descriptive* berfungsi melengkapi diagram *use case* agar kerangka kerja lebih mudah dipahami. *Descriptive* menjelaskan mengenai tentang bagaimana pengguna akan melakukan tugas dari sistem yang akan diciptakan. *Use case descriptive* akan menguraikan, dari sudut pandang pengguna, perilaku sistem saat merespons permintaan. Setiap *use case* direpresentasikan sebagai urutan dari langkah-langkah kerja sistem, dimulai dengan tujuan pengguna dan akan berakhir ketika tujuan tersebut terpenuhi. *Use case descriptive* disajikan dalam bentuk tabel, terdapat 12 komponen yang terdapat pada *use case descriptive* untuk mengungkapkan masing-masing *use case* secara lengkap. Berikut ini merupakan penjelasan dari asal komponen-komponen yang terdapat pada *use case descriptive* [10].

Tabel 2.2 Komponen Use Case Descriptive

No	Komponen	Deskripsi
1.	*Use Case ID*	Digunakan untuk memberikan kode ID untuk membedakan setiap *use case*.
2.	*Use Case Name*	Digunakan untuk memberikan nama dari *use case* tersebut.
3.	*Description*	Digunakan untuk menjelaskan secara singkat mengenai fungsi dari *use case*.
4.	*Actor*	Digunakan untuk memberikan informasi aktor yang

		akan terlibat dalam proses gambaran *use case* tersebut.
5.	*Pre Condition*	Digunakan sebagai pemberitahuan kondisi yang akan terjadi sebelum *use case* dijalankan. Atau persyaratan apa saja yang harus terpenuhi sebelum *use case* dijalankan
6.	*Post Condition*	Suatu proses yang akan dihasilkan dari kegiatan-kegiatan *use case*.
7.	*Basic Flow*	Menjelaskan mengenai langkah-langkah normal yang akan berakhir dengan sukses. Ditandai dengan awal, *body*, dan akhir.
8.	*Exceptions Flow*	Digunakan untuk memberikan informasi mengenai kendala yang menjadi penyebab *actor* awal yang tidak terpenuhi
9.	*Variations*	Berisi mengenai tindakan dari *alternative* apabila terdapat pengecualian dari awal *actor*. Merupakan representasi notasi <*extend*> pada diagram *use case*
10.	*Extensions*	Berisi mengenai *actor* tambahan yang merujuk pada notasi <*include*> biasanya akan diidentifikasi apabila terdapat langkah-langkah yang akan terlibat, akan tetapi tidak terkait dengan arus konteks *actor* awal
11.	*Business Rules*	Berisi mengenai aturan-aturan dari penjelasan yang terdapat pada gambar *use case*.

Use Case ID		
Use Case Name		
Description		
Actor		
Pre-Condition		
Post-Condition		
Basic flow	*Actor Action*	*System Response*
Variations		
Exceptions		
Extensions		
Business Rules		

Gambar 2.9 *Use Case Descriptive*

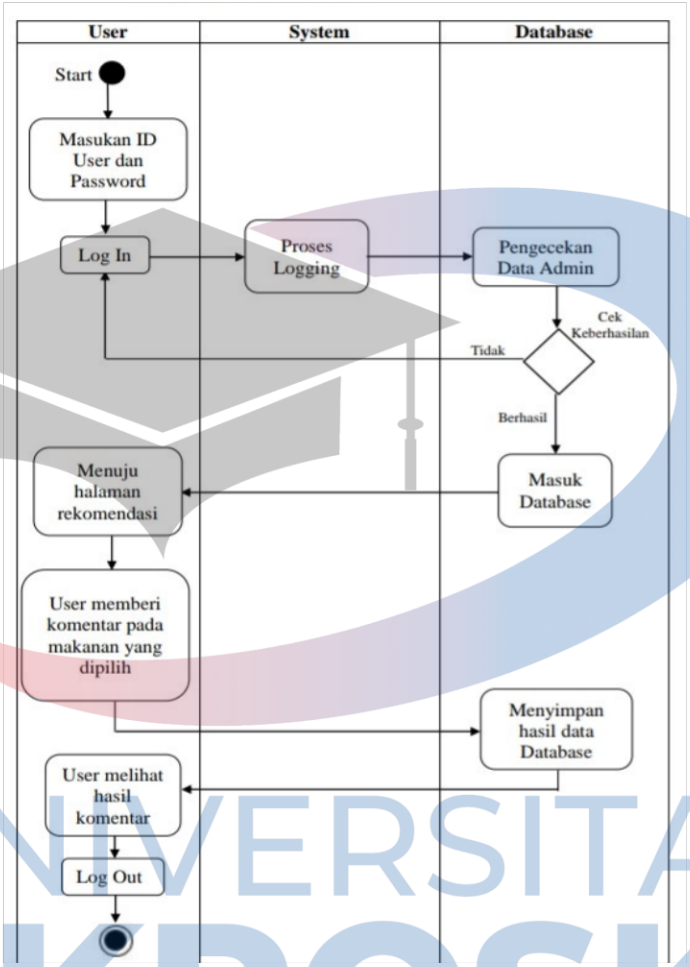
1.2.2 Activity Diagram

Activity Diagram merupakan salah satu jenis model diagram *Unified Modelling Language* (UML) yang menggambarkan hubungan dari proses suatu alur kerja yang berisi mengenai aktivitas dan juga tindakan terhadap sistem aplikasi yang akan di rancang. *Activity diagram* diciptakan untuk menjelaskan mengenai alur kerja dalam sistem aplikasi. Selain itu *activity diagram* juga dapat menggambarkan mengenai alur kontrol secara garis besar. *Activity diagram* sering dianggap sama seperti *flow chat*, meskipun terlihat sama akan tetapi sebenarnya berbeda. *Activity diagram* juga memiliki komponen yang akan dihubungkan melalui tanda panah, dan tanda panah tersebut akan mengarahkan pada barisan aktivitas dari sistem aplikasi yang akan dirancang. Tujuan dari *activity diagram* adalah untuk menggambarkan dan juga menunjukkan aliran alur dari proses aktivitas sistem agar proses lebih mudah dipahami [8].

Manfaat *activity diagram* yaitu sebagai berikut [8].

1. Dapat memperlihatkan barisan-barisan di dalam aktivitas proses dari sistem cara kerja dari aplikasi.
2. Dapat membantu dan juga memahami alur proses dari cara kerja aplikasi secara menyeluruh.

3. Menggambarkan alur proses kerja aplikasi serta barisan aktivitas di dalam proses sistem aplikasi.
4. Dapat menggambarkan alur proses cara kerja dari sistem aplikasi.
5. *Activity diagram* juga digunakan untuk mempermudah dalam mengembangkan perangkat lunak.



Gambar 2.10 Contoh *Activity Diagram*

Pada contoh gambar *activity diagram* diatas dimulai dengan *user* memasukkan ID *user* dan *password* setelah *user* melakukan *log in* maka sistem akan melakukan proses *logging* dan pada bagian *database* akan melakukan pengecekan data admin. Jika pengecekan data admin tersebut "tidak berhasil" maka akan melakukan *log in* kembali. Jika pengecekan data admin tersebut "berhasil" maka data tersebut akan masuk ke dalam *database*. Jika data tersebut sudah masuk ke dalam *DataBase* maka *user* akan menuju ke halaman rekomendasi, selanjutnya *user* dapat memberikan komentar pada makanan yang akan dipilih. Dan hasil

komentar tersebut akan disimpan melalui *database*, selanjutnya *user* dapat melihat hasil komentar yang diberikan, lalu *user* bisa melakukan *log out* [9].

Tabel 2.3 Simbol-Simbol Dasar Activity Diagram

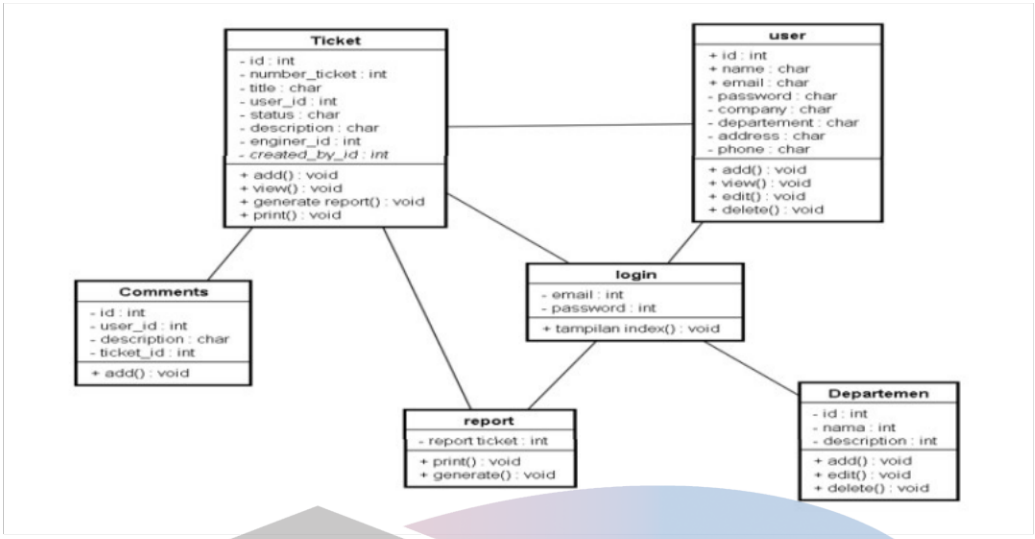
NO	SIMBOL	NAMA	KETERANGAN
1.		Status Awal	Menggambarkan simbol status awal dari keseluruhan aktivitas sistem aplikasi yang akan dijalankan.
2.		Aktivitas / *activity*	Menggambarkan dari aktivitas yang akan dilakukan oleh sistem aplikasi, simbol ini biasanya diawali dari kata kerja dari sistem yang akan dijalankan.
3.		Percabangan / *decision*	Menggambarkan percabangan, dimana jika ada melakukan pemilihan dari aktivitas sistem yang akan dijalankan.
4.		Penggabungan / *join*	Menggambarkan simbol dari penggabungan lebih dari satu aktivitas sistem aplikasi dan akan digabungkan menjadi satu.
5.		Status akhir / *final*	Menggambarkan simbol akhir dari aktivitas-aktivitas sistem yang akan dijalankan
6.		*Swimlane*	Menggambarkan Pemisahan dari sistem aplikasi yang akan bertanggung jawab dalam menjalankan aktivitas yang akan terjadi.

1.2.3 Class Diagram

Class Diagram adalah salah satu jenis model diagram *Unified Modelling Language* (UML) yang menggambarkan hubungan antar kelas dan penjelasan detail mengenai hubungan antar kelas dan kelas lainnya tiap-tiap kelas di dalam suatu model *desain* suatu sistem, dan juga dapat menunjukkan aturan-aturan serta tanggung jawab dari entitas yang akan menentukan perilaku sistem. Selain itu *class diagram* dapat menunjukkan operasi-operasi dan atribut-atribut dari sebuah kelas dengan objek yang saling terhubung. *Class diagram* mempunyai spesifikasi diantaranya seperti: kelas (*class*), *generalitation*, *aggregation*, attribut (*attributes*), *relasi assosiations*, *visibility*, dan juga operasi (*operation/method*). Hubungan antar kelas mempunyai keterangan yang disebut sebagai "*Multiplicity* atau *Cardinality*" [11].

Manfaat *class diagram* yaitu sebagai berikut [12].

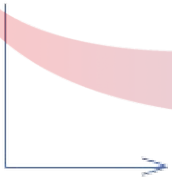
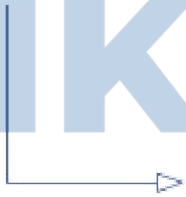
1. *Class diagram* dapat menjelaskan mengenai suatu model data pada suatu program informasi, dan juga tidak akan memperhatikan apakah model dari data tersebut sederhana atau kompleks.
2. Dengan melakukan rancangan *class diagram* maka dapat meningkatkan pemahaman mengenai gambaran skema dari program yang akan dirancang.
3. Dapat memberitahukan tampilan mengenai *class diagram* untuk kebutuhan spesifik dari suatu informasi serta dapat berbagi informasi.
4. *Class diagram* dapat membuat suatu bagan secara detail dan jelas, dengan cara memperhatikan kode spesifik yang dibutuhkan oleh program. Hal ini mampu mengimplementasikan kepada bagian struktur yang akan dijelaskan.
5. *Class diagram* dapat memberikan penjelasan mengenai implementasi-independen dari suatu program yang digunakan, lalu akan dilewatkan diantara berbagai komponen-komponennya.

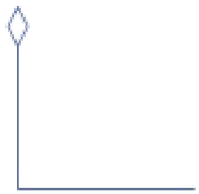


Gambar 2.11 Contoh *Class Diagram*

Pada contoh gambar *class diagram* diatas terdiri dari 6 jenis *class* yang saling terhubung. *Class* pertama dengan nama "*Ticket*" dengan jumlah atribut yang akan digunakan sebesar 8 atribut, dengan metode (operasi) sebanyak 4 metode. *Class* kedua dengan nama "*User*" dengan jumlah atribut yang akan digunakan sebesar 8 atribut, dengan metode (operasi) sebanyak 4 metode. *Class* ketiga dengan nama "*Comments*" dengan jumlah atribut yang akan digunakan sebesar 4 atribut, dengan metode (operasi) sebanyak 1 metode. *Class* keempat dengan nama "*Report*" dengan jumlah atribut yang akan digunakan sebesar 1 atribut, dengan metode (operasi) sebanyak 2 metode. *Class* kelima dengan nama "*Login*" dengan jumlah atribut yang akan digunakan sebesar 2 atribut, dengan metode (operasi) sebanyak 1 metode. Dan *class* keenam dengan nama "*Departemen*" dengan jumlah atribut yang akan digunakan sebesar 3 atribut, dengan metode (operasi) sebanyak 3 metode. Pada setiap nama dari atribut dan juga dari metode digunakan simbol (+) dan (-). Pada simbol (+) dapat diartikan sebagai "*Public*" sedangkan apa simbol (-) dapat diartikan sebagai "*Private*" [13].

Tabel 2.4 Simbol-Simbol Dasar Class Diagram

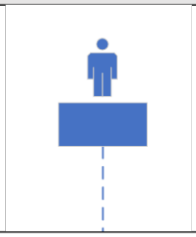
NO	SIMBOL	NAMA	KETERANGAN
1.		*Class*	Berfungsi sebagai simbol dari kelas struktur sistem yang akan digunakan
2.		*Interface*	Berfungsi sebagai konsep interface dalam suatu pemrograman berorientasi objek
3.		*Association*	Berfungsi sebagai relasi antar *class* dengan arti umum, asosiasi biasanya disertai dengan *multiplicity*
4.		*Directed Association*	Berfungsi sebagai relasi antar kelas dengan makna dari kelas yang akan digunakan oleh kelas-kelas yang lainnya, asosiasi biasanya disertai dengan *multiplicity*
5.		*Generalisasi*	Berfungsi sebagai relasi antar kelas dengan mempunyai makna dari *generalisasi-spesialisasi* (umum khusus)
6.		*Dependency*	Berfungsi sebagai relasi antar kelas dengan mempunyai makna dari kebergantungan antar kelas

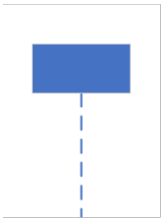
7.		*Aggregation*	Berfungsi sebagai relasi antar kelas dengan mempunyai makna semua-bagian (*whole-part*)

2.3.4 *Sequence* Diagram

Sequence Diagram adalah salah satu jenis model diagram *Unified Modelling Language* (UML) yang menggambarkan mengenai kedudukan dari *behavior* pada sebuah *actor* tunggal. Oleh karena itu *Sequence diagram* tersebut dapat digambarkan ketika sudah mengetahui objek-objek yang akan terlibat dalam sebuah *use case*. *Sequence diagram* dapat menjelaskan bagaimana cara objek berinteraksi dengan cara mengirim dan menerima pesan. *Sequence diagram* menggambarkan suatu perilaku mengenai objek pada suatu proses dan dapat mendeskripsikan waktu hidup mengenai objek dan pesan yang diterima dan dikirimkan oleh antar objek. Oleh karena ketika ingin menggambarkan *Sequence diagram* maka harus mengetahui terlebih dahulu objek-objek yang terlibat di dalam sebuah proses beserta metode-metode yang dimiliki oleh kelas yang diinstansiasi dan akan menjadi objek [14]. Berikut ini merupakan simbol-simbol dasar yang digunakan pada *Sequence diagram* [15].

Tabel 2.5 Simbol-Simbol Dasar *Sequence* Diagram

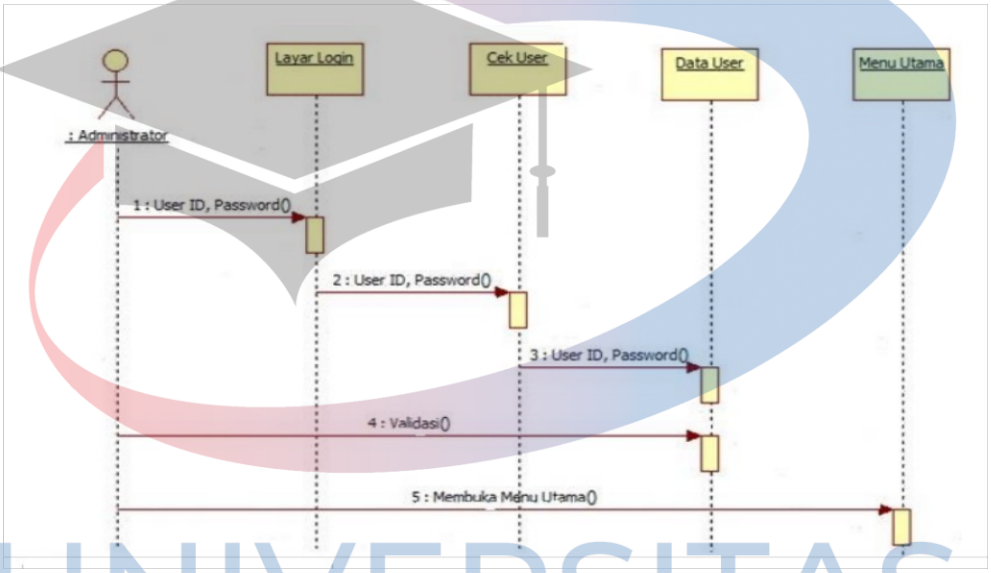
NO	SIMBOL	NAMA	KETERANGAN
1.		*Actor*	Berfungsi sebagai perwakilan dari peran yang akan digunakan oleh sistem

2.		*Object*	Berfungsi sebagai *instance* dari sebuah class dan akan dituliskan secara tersusun dalam bentuk *horizontal*. Yang digambarkan sebagai sebuah *class* (kontak) dengan nama objek yang di dalamnya diawali dengan sebuah tanda titik koma.
3.		*Message*	Menggambarkan dengan tanda panah yang berbentuk *horizontal* di antara *activation* message mengindikasikan komunikasi antar objek-objek tersebut.
4.		*Activation*	*Activation* digambarkan dengan bentuk persegi empat yang digambarkan pada suatu *lifeline*. Yang berfungsi sebagai mengindikasikan sebuah objek yang akan melakukan dari suatu aksi.
5.		*Lifeline*	*Lifeline* mengindikasikan keberadaan dari sebuah objek dalam basis waktu. Bentuk gambaran mengenai *lifeline* adalah garis putus-putus dengan berbentuk *vertikal* yang ditarik dari sebuah *objek*

Komponen - komponen yang ada pada *Sequence diagram* yaitu sebagai berikut [12]:

1. *Object*: merupakan komponen berbentuk kotak yang akan digunakan untuk mewakili sebuah *class* atau object. *Object* tersebut nantinya akan mendemonstrasikan seperti apa *object* berperilaku dalam sistem tersebut.

2. *Activation*: merupakan komponen yang berbentuk persegi panjang yang digunakan untuk menggambarkan suatu waktu yang dibutuhkan dalam sebuah *object* untuk menjalankan tugas.-tugas tersebut. Semakin lama waktu yang dibutuhkan, maka *activation* akan semakin panjang.
3. *Actor*: merupakan komponen yang memiliki bentuk seperti *stick figure*. Yang digunakan untuk merepresentasikan *user* yang berinteraksi dengan sistem tersebut.
4. *Lifeline*: merupakan komponen yang memiliki bentuk garis putus – putus yang akan memuat suatu informasi mengenai nama dari *object*. *Lifeline* sebagai berfungsi menggambarkan aktifitas dari *object*.



Gambar 2.12 Contoh *Sequence Diagram*

Pada contoh gambar *Sequence diagram* terdapat sebuah aktor dengan nama "*Administrator*" dan terdiri dari empat objek yaitu layar *login*, cek *user*, data *user*, dan menu utama. Pertama-tama *administrator* akan memasukkan *user* ID dan juga *password*. Lalu *user* ID beserta *Password* yang telah dimasukkan akan dilakukan pengecekan oleh *user*, Lalu Data *user* akan melakukan validasi kepada *administrator*. *Administrator* akan membuka menu utama. Pada gambar diatas *activation* biasanya memiliki garis yang memberitahukan aktifitas yang akan terjadi ketika *actor* atau objek berinteraksi ke objek yang lainnya, [12].

2.1. Basis Data

Basis Data merupakan suatu kumpulan data yang terhubung dan disimpan secara bersama-sama dengan menggunakan metode tertentu dengan menggunakan komputer,

sehingga dapat menyediakan informasi yang optimal yang akan di perlukan dalam suatu perusahaan. Basis data yang dikelola berdasarkan sebuah skema atau struktur tertentu, dan dengan menggunakan perangkat lunak untuk melakukan penggunaannya. Dan dengan membutuhkan *software* untuk melakukan proses informasi dari data tersebut. Basis data bisa diartikan sebagai data yang akan disusun dalam bentuk beberapa tabel yang saling memiliki hubungan maupun berdiri sendiri [16].

Operasi dasar dari basis-basis yaitu sebagai berikut [16]:

1. *Create DataBase*: merupakan suatu perintah yang akan digunakan untuk membuat basis data dengan nama yang akan diberikan.
2. *Drop DataBase*: merupakan suatu perintah yang akan digunakan untuk menghapus basis data dengan nama yang akan diberikan.
3. *Create Tabel*: merupakan suatu perintah yang akan digunakan untuk membuat suatu tabel dalam basis data.
4. *Drop Tabel*: merupakan suatu perintah yang akan digunakan untuk menghapus suatu tabel dalam basis data.
5. *Insert*: merupakan suatu perintah yang akan digunakan untuk memasukkan data (record) ke dalam tabel.
6. *Update*: merupakan suatu perintah yang digunakan untuk memperbaharui data (*record*) pada suatu tabel.
7. *Delete*: merupakan suatu perintah yang akan digunakan untuk menghapus data (*record*) pada suatu tabel.

Tujuan dalam merancang basis data yaitu sebagai berikut [17]:

1. Kecepatan dan kemudahan (*Speed*)

   Dalam pemanfaatan basis data memungkinkan untuk melakukan menyimpan data atau dapat melakukan perubahan terhadap data atau dapat menampilkan kembali data tersebut dengan cepat dan simpel.
2. Efisiensi ruang penyimpanan (*Space*)

   Penggunaan ruang penyimpanan di dalam basis data dilakukan dengan tujuan untuk mengurangi jumlah pengulangan data. Baik dilakukan dengan cara penerapan sejumlah pengkodean atau dalam bentuk file antara kelompok-kelompok data yang saling berhubungan.
3. Keakuratan (*Accuracy*)

Pemanfaatan pengkodean atau pembentukan relasi antara data bersama dengan penerapan aturan atau batasan tipe data, domain data, keunikan data, dan lain sebagainya. Yang akan diterapkan dalam basis data tersebut, sangat bermanfaat untuk menentukan ketidakakuratan dalam pemasukan atau dalam penyimpanan data tersebut.

4. Ketersediaan (*Availability*)

   Pertumbuhan dari data sejalan dengan waktu dan akan semakin membutuhkan ruang penyimpanan yang besar. Data yang tidak pernah digunakan lagi dapat diatur untuk dilepaskan dari sistem basis data, dengan cara melakukan penghapusan atau dengan cara memindahkan data tersebut ke media penyimpanan yang lain.

5. Kelengkapan (*Completeness*)

   Agar data yang telah di kelola bersifat relatif baik, baik terhadap kebutuhan pemakaian maupun terhadap kebutuhan waktu. Dalam sebuah basis data, struktur dari basis data harus disimpan. Untuk kebutuhan dan kelengkapan dari data yang semakin berkembang, tidak cukup hanya menambah *record- record* data, akan tetapi juga harus menambah struktur dalam basis data.

6. Keamanan (*Security*)

   Suatu sistem untuk melindungi basis data dari berbagai ancaman, digunakan untuk menentukan siapa saja yang akan menggunakan basis data tersebut dan menentukan jenis-jenis operasi apa saja yang akan dilakukan. Seseorang yang mempunyai hak untuk mengontrol dan mengatur *DataBase* biasanya disebut sebagai *administrator*.

7. Kebersamaan pemakai (*Shareability*)

   Sistem dalam pemakai basis data sering kali tidak terbatas. Basis data yang telah dikelola oleh sistem (aplikasi) yang mendukung lingkungan *multiuser*, akan dapat memenuhi kebutuhan, akan tetapi dengan menjaga atau menghindari terhadap munculnya persoalan baru seperti dalam berubah-ubah suatu data.

2.4.1 Bahasa basis data

Bahasa Basis Data merupakan suatu bahan yang akan digunakan oleh user ketika sedang melakukan komunikasi yang telah ditetapkan oleh pembuat *database Manajemen Sistem* (DBMS). Bahasa basis data dapat dibedakan menjadi dua , yaitu sebagai berikut [16]:

1. *Data Definition Language* (DDL)

DDL merupakan suatu kumpulan dari perintah-perintah SQL yang digunakan untuk membuat (*create*), mengubah (*alter*) dan menghapus (*drop*) struktur dan definisi tipe data dari objek- objek *database* tersebut. Dengan bahasa ini kita dapat membuat tabel baru,

membuat *indeks*, mengubah tabel, menentukan struktur tabel, dan lain sebagainya. Dan hasil dari kompilasi perintah DDL akan menjadi kamus data, yaitu data yang menjelaskan tentang data sesungguhnya.

2. *Data Manipulation Language* (DML)

Data Manipulation Language merupakan suatu kumpulan perintah *query* yang akan digunakan untuk memanipulasi data pada *database* tersebut. DML bermanfaat untuk melakukan manipulasi dan melakukan pengambilan data pada suatu basis data. Yang meliputi *insert*, *update*, *delete*, dan lain sebagainya. Ada 2 jenis yaitu: pertama *prosedural* yang dapat di artikan sebagai: akan menentukan data yang akan diinginkan dan cara mendapatkannya. Dan yang kedua adalah *non-prosedural* dapat diartikan sebagai: tanpa melakukan penyebutan cara mendapatkannya.

1. *Data Control Language (DCL)*

Data Control Language merupakan suatu perintah yang berada di dalam SQL untuk digunakan agar memanipulasikan suatu data di *database* tersebut. DCL bermanfaat untuk melakukan mengontrol data di *database* seperti memanipulasi data user dan hak akses user. Yang termasuk perintah yang ada di dalam DCL ada dua yaitu *Grant, Revoke*. Perintah *Grant* digunakan seorang administrator basis data agar memberikan hak akses kepada pengguna tertentu, sedangkan perintah *Revoke* digunakan seorang administrator basis data agar membatalkan / menghentikan semua hak akses yang sudah di berikan sebelumnya kepada pengguna [18].

2.1 Aplikasi Mobile

2.5.1 Aplikasi Mobile

Aplikasi *mobile* merupakan suatu alat untuk berkomunikasi yang sudah tersebar di seluruh penjuru dunia, untuk itu diperlukan aplikasi-aplikasi agar dapat mempermudah pekerjaan seseorang dimanapun dan kapanpun agar mendapatkan sebuah informasi. Aplikasi *mobile* adalah sebutan untuk aplikasi yang berjalan di *mobile device*. Dengan menggunakan aplikasi *mobile*, dapat mempermudah dalam melakukan berbagai macam aktifitas dimulai dari hiburan, berjualan, belajar, *browsing* dan lain sebagainya [19].

Aplikasi *mobile* salah satu jenis program yang siap digunakan untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh para pengguna aplikasi yang tertuju. Pemakaian aplikasi *mobile* untuk hiburan paling banyak disukai oleh para pengguna *smartphone*, oleh karena itu dengan adanya pemakaian program aplikasi

mobile, maka masyarakat semakin mudah dalam menikmati hiburan kapan saja dan dimana saja. Perangkat *mobile* memiliki banyak jenis mengenai ukuran, *desain layout*, akan tetapi Perangkat aplikasi *mobile* memiliki *memory* yang kecil [19].

2.5.2 Asisten Rumah Tangga

Pada zaman sekarang, manusia untuk memenuhi kehidupannya untuk menyambung hidupnya di masa era modern seperti ini tidak hal yang sangat mudah. Terlebih dalam aktivitas-aktivitas yang dipadatkan dengan berbagai macam kegiatan seperti kegiatan mengurus rumah, kegiatan pekerjaan dan lain sebagainya. Bagi para wanita karir yang disibukkan dengan urusan pekerjaannya, tidak menutup kemungkinan keperluan pekerjaan rumah tangga sering sekali terbengkalai dan tidak terurus. Yang membantu pekerjaan di dalam rumah tangga yaitu sering juga disebut sebagai asisten rumah tangga (ART). Asisten rumah tangga merupakan bagian terpenting dalam kehidupan sehari-hari, dan bahkan terkadang menjadi orang kepercayaan dari pemilik rumah untuk mengurusi segala keperluan yang ada di lingkungan rumah tersebut. Dengan alasan yang seringkali digunakan seseorang ketika akan memutuskan untuk mempekerjakan asisten rumah tangga adalah pasangan suami istri yang sedang sibuk berkerja, kurangnya keterampilan rumah tangga, contohnya seperti: memasak, dan rasa malas untuk melakukan pekerjaan rumah, dan lain sebagainya. Peran pekerja rumah tangga dalam kehidupan sehari-hari amat sangat penting. Asisten rumah tangga bukan hanya mengurusi pekerjaan yang berhubungan dengan rumah tangga saja, akan tetapi bisa mencakup perihal penanganan atas perangkat berteknologi yang serba canggih. Misalnya saja dalam menangani dan bertanggung jawab atas alat-alat elektronika [20].

Hak-Hak asisten rumah tangga menurut peraturan menteri nomor 2 tahun 2015 pasal 7 hak-hak asisten rumah tangga antaranya yaitu sebagai berikut:

1. Memperoleh informasi mengenai pengguna
2. Mendapat perlakuan yang baik dari pengguna dan anggota keluarganya.
3. Mendapatkan upah sesuai perjanjian kerja.
4. Mendapatkan makanan dan minuman yang sehat.
5. Mendapatkan waktu istirahat yang cukup.
6. Mendapatkan hak cuti sesuai kesepakatan.
7. Mendapat kesempatan melakukan ibadah sesuai dengan agama dan kepercayaan yang di anut.
8. Mendapat tunjangan hari raya dan Berkomunikasi dengan keluarganya.

Kewajiban pembantu rumah tangga telah di atur dalam pasal 8 Peraturan Menteri Ketenagakerjaan Nomor 2 Tahun 2015. Yaitu sebagai berikut [21]:

1. Melaksanakan tugas dan tanggung jawab sesuai dengan perjanjian kerja yang telah disepakati.
2. Menyelesaikan pekerjaan dengan baik dan benar.
3. Menjaga etika dan sopan santun di dalam rumah majikan.
4. Memberitahukan kepada majikan bahwa dalam waktu yang cukup apabila asisten rumah tangga akan berhenti bekerja.