BAB II

LANDASAN TEORI

II.1. Konsep Dasar

II.1.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra, 2012 : 157 - 158).

Berikut ini merupakan pengertian sistem dari beberapa ahli:

1. Jerry FithGerald "Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.
2. Ludwig Von Bartalanfy "Sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi di antara unsur-unsur tersebut dengan lingkungan.
3. Anatol Raporot "Sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain.
4. L. Ackof "Sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung satu sama lainnya.

Dari uraian di atas, sehingga dapat disimpulkan bahwa sistem adalah sekumpulan elemen yang saling terkait atau terpadu untuk mencapai tujuan tertentu.

Sustu sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan. Sebagai misal, sistem komputer dapat terdiri dari subsistem perangkat keras dan subsistem perangkat lunak. Subsistem perangkat keras dapat terdiri dari alat masukan, alat pemproses, alat keluaran dan simpanan luar. Subsistem saling berinteraksi membentuk satu kesatuan sehingga tujuan atau sasaran sistem dapat tercapai. Suatu sistem mempunyai komponen, batas sistem, lingkungan luar sistem, penghubung, masukan, keluaran, pengolah dan sasaran (Dina Andayati : 2014).

II.1.2. Pengertian Informasi

Informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna bagi penerimanya yang menggambarkan suatu kejadian yang nyata yang digunakan untuk mengambil keputusan. Sumber dari informasi adalah data.

Data merupakan bentuk yang masih mentah, belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk menghasilkan informasi. Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model dan seterusnya membentuk siklus (Dina Andayati : 2014).

II.1.3. Sistem Informasi

Sistem informasi dapat didefinisikan sebagai suatu sistem di dalam suatu organisasi yang merupakan kombinasi dari orang, fasilitas, teknologi, media, prosedur dan pengendalian yang ditunjukan untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian internal dan eksternal yang penting dan menyediakan suatu dasar informasi untuk pengambilan keputusan yang cerdik (Dina Andayati : 2014).

II.2. Sistem Pendukung Keputusan

Sistem Pendukung Keputusan (SPK) atau *Computer Based Decision Support System* (DSS) merupakan salah satu bagian dari sistem informasi yang berguna untuk meningkatkan efektifitas pengambilan keputusan. Permasalahan yang umum dijadikan objek pada SPK ada yang bersifat yang bersifat semi terstruktur atau terstruktur (Tri Murni et,al : 2015).

Sistem Pendukung Keputusan terdiri atas tiga komponen utama atau subsistem yaitu (Yohanes et,al : 2009).

a. Subsistem data (data base).
b. Subsistem model (model base).
c. Subsistem dialog (user system interface).

II.2.1. Karakteristik Sistem Pendukung Keputusan

Karakteristik dalam sistem pendukung keputusan, antara lain (Erliza Septia Nagara dan Rini Nurhayati, 2015):

1. Sistem Pendukung Keputusan dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur dengan menambahkan kebijaksanaan manusia dan informasi komputerisasi.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan penggunaan model-model analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi.
3. Sistem Pendukung Keputusan, dirancang sedemikian rupa sehingga dapat digunakan/dioperasikan dengan mudah.
4. Sistem Pendukung Keputusan dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi.

II.2.2. Keuntungan dan Keterbatasan Sistem Pendukung Keputusan

Sistem pendukung keputusan dapat memberikan berbagai manfaat atau keuntungan bagi pemakainya, antara lain (Nungsiati, 2013) :

1. Memperluas kemampuan pengambil keputusan dalam memproses data/informasi bagi pemakainya.

2. Membantu pengambilan keputusan dalam hal penghematan waktu yang dibutuhkan untuk memecahkan masalah terutama berbagai masalah yang sangat kompleks dan tidak terstruktur.
3. Dapat menghasilkan solusi dengan lebih cepat serta hasilnya dapat diandalkan.
4. Walaupun suatu Sistem Pendukung Keputusan, mungkin saja tidak mampu memecahkan masalah yang dihadapi oleh pengambil keputusan, namun dapat menjadi stimulant bagi pengambil keputusan dalam memahami persoalannya, Karena sistem pendukung keputusan mampu menyajikan berbagai alternatif

II.2.3. Komponen Sistem Pendukung Keputusan

Untuk dapat menerapkan SPK, ada 4 komponen subsistem yang harus disediakan yaitu (Syukron Hidayat dan Imam Mukhlash, 2015) :

1. Subsistem manajemen data

   Subsistem ini menyediakan data bagi sistem, termasuk didalamnya basis data. Berisi data yang relevan untuk situasi dan diatur oleh perangkat lunak yang disebut *Database Management System* (DBMS).
2. Subsistem manajemen model

   Subsistem ini berfungsi sebagai pengelola berbagai model, mulai dari model keuangan, statistik, matematik, atau model kuantitatif lainnya yang memiliki kemampuan analisis dan manajemen perangkat lunak yang sesuai. Perangkat lunak ini sering disebut *Model Base Management System* (MBMS).

3. Subsistem manajemen pengetahuan

   Subsistem ini mendukung berbagai subsistem lainnya, atau dapat dikatakan berperan sebagai komponen yang independen. Subsistem ini menyediakan intelegensi untuk menambah pertimbangan pengambil keputusan.

4. Subsistem manajemen antar muka pengguna

   Subsistem ini berupa tampilan yang disediakan yang mampu mengintegrasikan sistem terpasang dengan pengguna secara interaktif. Melalui subsistem ini pengguna dapat berkomunikasi dengan sistem pendukung keputusan serta memerintah sistem pendukung keputusan.

II.3. Metode Perbandingan *Eksponensial* (MPE)

Menurut Marimin (2005), yang dikutip oleh Didie Nanda Pribadi (2011) Metode Perbandingan *Eksponensial* (MPE) merupakan salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak (Andri Januardi, 2013).

Dalam menggunakan Metode Perbandingan *Eksponensial* ada beberapa tahap yang harus dilakukan, yaitu:

1. Menyusun alternatif-alternatif keputusan yang akan dipilih.
2. Menentukan kriteria atau perbandingan keputusan yang penting untuk dievaluasi.
3. Menentukan tingkat kepentingan dari setiap kriteria keputusan.
4. Melakukan penilaian terhadap semua alternatif pada setiap kriteria.
5. Menghitung skor atau nilai total setiap alternative.

6. Menentukan urutan prioritas keputusan didasarkan pada skor atau nilai total masingmasing alternative.

Adapun rumus matematika yang dipakai dalam menggunakan Metode Perbandingan Ekponensial adalah :

$$\text{Total nilai } (TN_i) = \sum_{j=1}^{m} (RK_{ij})^{TKKj} \quad \text{..... (1)}$$

Keterangan :

a. TNi : Total nilai alternatif ke-i
b. RKij : Derajat kepentingan relatif kriteria ke-j pada pilihan keputusan i
c. TKKj : Derajat kepentingan kriteria keputusan ke-j; TKKj > 0;
d. m : Jumlah kriteria keputusan
e. n : Jumlah pilihan keputusan
f. j : 1,2,3,…m; m : Jumlah kriteria
g. i :1,2,3,…,n; n : Jumlah pilihan alternatif

II.4. *Microsoft Visual Basic 2010*

Pada akhir tahun 1999, Teknologi .NET diumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML Web *Services*. XML Web *services* memungkinkan aplikasi tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apapun melalui internet.

Visual Basic.NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat

mengambil data dari server dengan tipe apa pun asalkan terinstal .NET Framework. (Priyanto Hidayatullah, 2012 ; 5).

II.5. *SQL Server 2008*

SQL *(Structured Query Language)* adalah sebuah bahasa yang dipergunakan umtuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang dipergunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

SQL terdiri dari dua bahasa, yaitu *Data Definition Language Manipulation Language (DML)*. Implementasi DDL dan DML sistem manajemen basis data *(SMBD)*, namun secara umum implemen bahasa ini memiliki bentuk standar yang ditetapkan oleh *ANSI*. (Adelia, Jimmy Setiawan : 2011 ; 115).

1. *Data Defenition Language (DDL)*

   DDL digunakan untuk mendefinisikan, mengubah, serta menghapus basis data dan objek-objek yang diperlukan dalam basis data, misalnya tabel, *view*, *user*, dan sebagainya. *DDL* biasanya digunakan oleh administrator basis data dalam pembuatan sebuah aplikasi basis data. Secara umum *DDL* yang digunakan adalah:

   a. *CREATE* untuk membuat objek baru.
   b. *USE* untuk menggunakan objek yang sudah ada.
   c. *ALTER* untuk mengubah objek yang sudah ada.
   d. *DROP* untuk menghapus objek.

2. *Data Manipulation Language (DML)*

   DML digunakan untuk memanipulasi data yang ada dalam suatu tabel. Perintah-perintah yang umum dilakukan adalah:

   a. *SELECT* untuk menampilkan data.
   b. *INSERT* untuk menambahkan data baru.
   c. *UPDATE* untuk mengubah data yang sudah ada.
   d. *DELETE* untuk menghapus data.

II.6. Pengertian Basis Data

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa *mengatap* satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*]), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali; data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya; data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Edy Sutanta, 2011 : 29-30).

II.7. Normalisasi

Menurut Martin (1995), Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah

timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta ; 2011 : 174).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Edy Sutanta ; 2011 : 175) :

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Edy Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

   a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)

b. Jika relasi membuat *set atribut* berulang *(non single values)*

c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*

b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*

c. Jika relasi tidak memuat set atribut berulang

d. Jika semua record mempunyai sejumlah atribut yang sama.

Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

a. Tidak dapat menyisipkan informasi parsial

b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

a. Jika memenuhi kriteria *First Norm Form (*1NF).

b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK).

Permasalahan dalam *Second Normal Form /* 2NF adalah sebagai berikut

a. Kerangkapan data *(data redundancy)*

b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*

c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Second Normal Form* (2NF)
b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasi TDF relasi *Second Normal Form* (2NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)
   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.
   b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

   Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

II.8. *Unified Modeling Language* (UML)

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi diagram (Rosana Junita Sirait, et al., 2015).

Menurut Rama (2008:111), "*Unified Modeling Language* (UML) adalah suatu bahasa permodelan untuk menyebutkan, memvisualisasikan, membuat dan mendokumentasikan sistem informasi." Menurut Henderi (2007:4), "*Unified Modeling Language* (UML) adalah sebuah bahasa pemodelan yang telah menjadi standar dalam industri *software* untuk visualisasi, merancang, dan mendokumentasikan sistem perangkat lunak." Bahasa pemodelan UML lebih cocok untuk pembuatan perangkat lunak dalam bahasa pemrograman berorientasi

objek (C+, Java, VB.NET), namun demikian tetap dapat digunakan pada bahasa pemrograman prosedural.

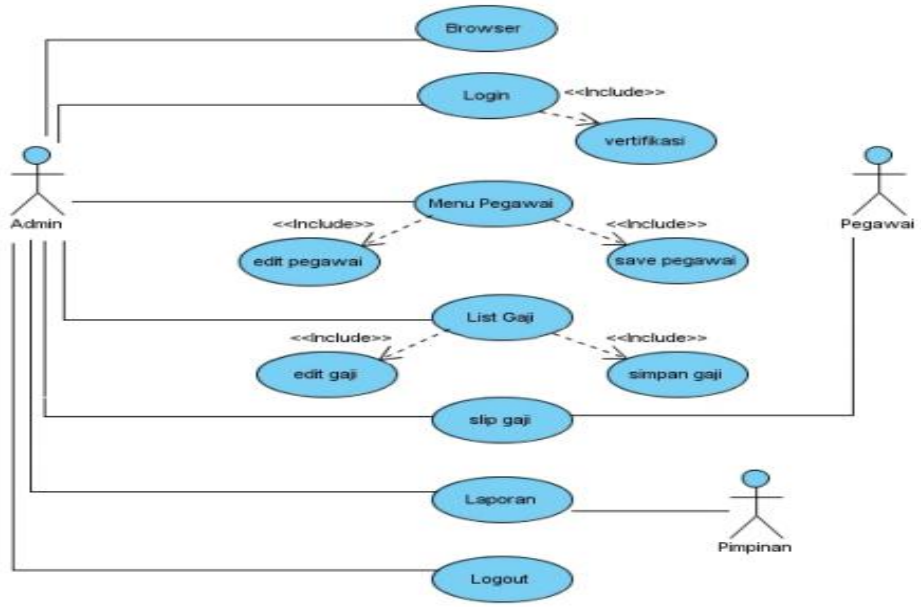
Berdasarkan beberapa pendapat dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*) (Aris, et al., 2015).

II.8.1. *Use Case* Diagram

Use case adalah deskripsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai (Oktafiansyah, 2012).

Suatu *use case* diagram menampilkan sekumpulan *use case* dan aktor (pelaku) dan hubungan diantara *use case* dan aktor tersebut. *Use case* diagram digunakan untuk penggambaran *use case* statik dari suatu sistem. *Use case* diagram penting dalam mengatur dan memodelkan kelakuan dari suatu sistem. *Use case* menjelaskan apa yang dilakukan sistem (atau subsistem) tetapi tidak menspesifikasi cara kerjanya. *Flow of event* digunakan untuk menspesifikasi cara kerjanya kelakuan dari *use case*. *Flow of event* menjelaskan *use case* dalam bentuk tulisan dengan sejelas-jelasnya, dintaranya bagaimana, kapan *use case* dimulai dan berakhir, ketika *use case* berinteraksi dengan aktor, objek apa yang

digunakan, alur dasar dan alur alternatif. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use cases*, aktor dan relasi (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).



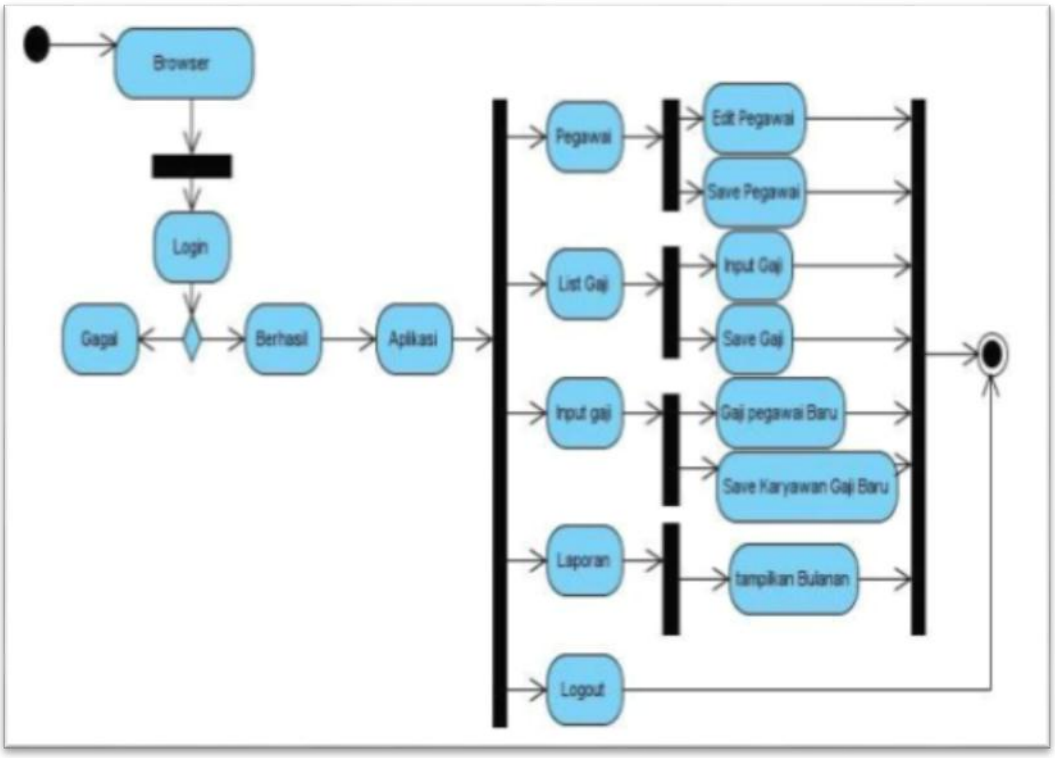
Gambar II.1. *Use Case* Diagram
(Sumber : Aris, et al., 2015)

II.8.2. *Activity* Diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus (Oktafiansyah, 2012).

Activity diagram memperlihatkan alur langkah demi langkah dalam suatu proses. Suatu aktivitas menunjukkan sekumpulan aksi (secara sekuensial atau bercabang dari satu aksi ke aksi lain), dan nilai yang dihasilkan atau digunakan oleh aksi-aksi yang terjadi. Activity diagram ditunjukkan untuk memodelkan fungsi dari suatu sistem dan menekankan pada alur dari kontrol didalam

pelaksanaan dari suatu tindakan (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

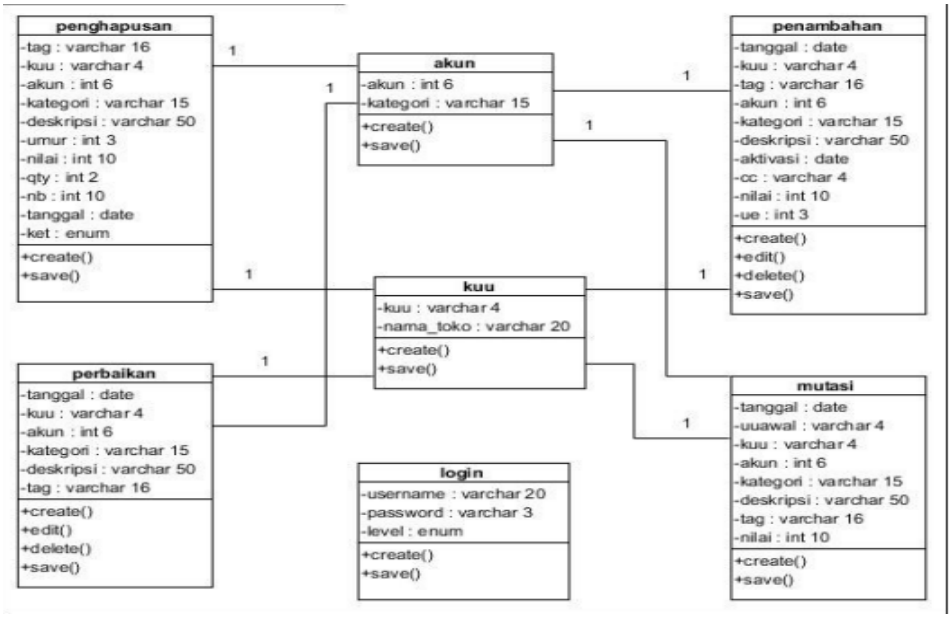


Gambar II.2. *Activity* Diagram
(Sumber : Aris, et al., 2015)

II.8.3. *Class* Diagram

Class diagram menunjukan sekumpulan kelas, antarmuka, dan kerjasama serta hubungannya. *Class* diagram digunakan untuk memodelkan perancangan statik dari gambaran sistem. Biasanya meliputi pemodelan *vocabulary* dari sistem, pemodelan kerjasama, atau pemodelan skema. *Class* diagram dapat digunakan untuk membangun sistem yang dapat dieksekusi melalui teknik *forward and reverse*, selain untuk penggambaran, menspesifikasikan, dan pendokumentasian struktur model (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat (Haviluddin, 2011).

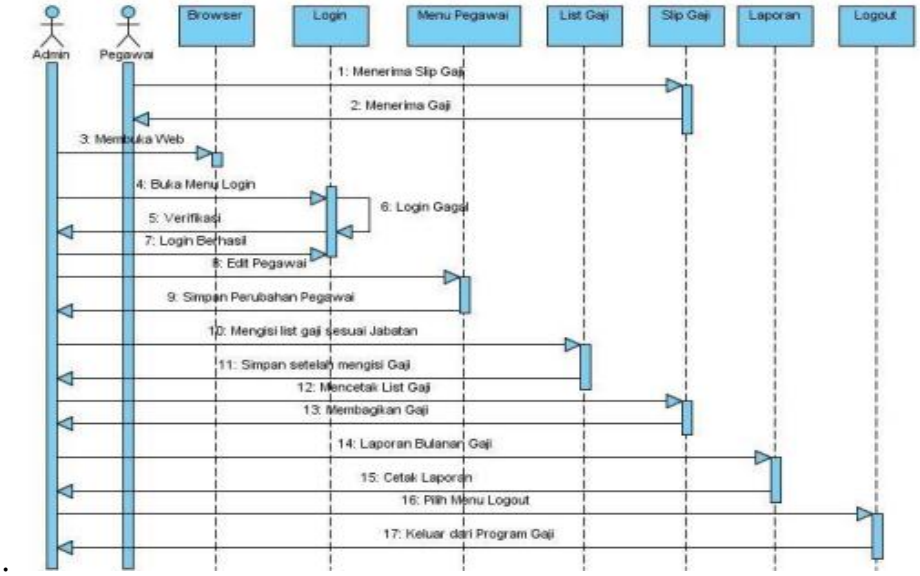


Gambar II.3. *Class* Diagram
(Sumber : Rosana Junita Sirait, et al., 2015)

II.8.4. *Sequence* Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan jumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case* (Oktafiansyah, 2012).

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram* (Haviluddin, 2011).



Gambar II.4. *Sequence* Diagram
(Sumber : Aris, et al., 2015)