BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Sistem

Secara sederhana suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari suatu unsur, komponen, atau variabel yang terorganisir, saling tergantung satu sama lain dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kennet Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang mendapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi.

Teori sistem melahirkan konsep-konsep furistik, antara lain yang terkenal adalah konsep sibernitika (*cybernetics*). Konsep atau dibidang kajian ilmiah ini berkaitan dengan upaya menerapkan berbagai ilmu yaitu ilmu perilaku, fisika, biologi dan tehnik. Oleh karena sibernitika biasanya berkaitan dengan usaha-usaha otomasi tugas-tugas yang dilakukan manusia, sehingga melahirkan studi-studi robotika, kecerdasan buatan (*artificial intelengence*). Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan (*input*), pengolahan (*processing*), dan keluaran (*output).* Selain itu, suatu si stem tidak bisa lepas dari lingkungan maka umpan balik (*feed back)* dapat berasal dari lingkungan sistem yang dimaksud. Organisasi dipandang sebagai suatu sistem yang tentunya akan memiliki semua unsur ini (Tata Sutabri, 2012).

Pengertian sistem dilihat dari elemen-elemennya dalam kumpulan elemen-elemen yang saling berkaitan dan bekerjasama dalam melakukan kegiatan untuk mencapai suatu tujuan, sedangkan pengertian sistem jika dilihat dari masukan dan keluarannya adalah suatu rangkaian yang berfungsi menerima input (masukan), mengolah input, dan menghasilkan output (keluaran). (V. Wiratna Sujarweni ; 2015. 1).

Dari kesimpulan diatas, Sistem adalah entitas atau satuan yang terdiri dari dua atau lebih komponen atau subsistem (sistem yang lebih kecil) yang saling terhubung dan terkait untuk mencapai suatu tujuan.

II.1.2. Informasi

Pada Informasi adalah data yang telah diklasifikasikan diolah atau diinterprestasi untuk digunakan dalam proses mengambil keputusan. Sistem pengolahan informasi mengolah data menjadi informasi atau tepatnya mengolah dari bentuk tak berguna menjadi berguna bagi penerimanya.

Informasi adalah data yang telah diklasifikasikan atau diinterprestasi untuk digunakan dalam mengambil keputusan (Tata Sutabri, 2012).

II.1.3. Sistem Informasi

Sistem informasi adalah berupa suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian yang mendukung operasi yang bersifat managerial dengan kegiatan strategi suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Tata Sutabri, 2012).

II.2. Izin

Izin merupakan salah satu perwujudan kewenangan pemerintah dalam menata kehidupan masyarakat. Izin dalam pandangannya merupakan ujung tombak dari instrumen hukum yang mengaplikasikan peraturan dalam hal konkret sektor kehidupan yang beraneka ragam. Sebagai instrumen izin berfungsi selaku ujung tombak instrumen hukum sebagai pengarah, perekayasa dan perancang masyarakat adil dan makmur itu dijelmakan.

Perizinan adalah proses untuk mendapatkan izin untuk terselenggaranya ketertiban. Ketertiban adalah tujuan dari segala hukum. Kebutuhan terhadap ketertiban ini syarat pokok(fundamental) bagi adanya suatu masyarakat manusia yang teratur. (*Yurisdian ; 2013 : 326*)

II.3. Warnet (Warung *Internet*) dan *Game Online*

II.3.1. Warnet (Warung Internet)

Warnet adalah singkatan dari "Warung *Internet*". Warung Internet adalah terjemahan dalam bahasa Indonesia dari istilah asing *Internet Cafe*. Warnet memberi pelayanan akses internet bagi pelanggan. Sebagai gantinya pelanggan membayar biaya tertentu sesuai tarif yang di tetapkan. Perhitungan tarif berdasarkan durasi akses internet atau besarnya data.

Di negara dunia ketiga, warnet adalah tempat kebanyakan orang mengakses internet. Di negara-negara atau daerah-daerah maju dimana akses *internet* sudah ada pada hampir setiap rumah, warnet jarang didapatkan dan mahal tarifnya. (*Dwi Agus Diartono ; 2007 : 77*)

II.3.2. *Game Online*

Online game adalah permainan yang dimainkan secara *online* via internet (Young, 2009). Game dengan fasilitas *online* via internet menawarkan fasilitas lebih dibandingkan dengan game biasa(seperti *video game*) kerena pemain itu bisa berkomunikasi dengan pemain lain diseluruh penjuru dunia melalui chating.

Pada saat ini, bagi remaja yang tidak memiliki fasilitas online game dirumahnya,tersedia warung-warung internet (warnet) yang menyediakan fasilitas online games, kondisi ini membuat remaja menjadi lebih mudah untuk bermain games dimana saja dan kapan saja tanpa mengenal waktu, yang akhirnya menyebabkan remaja menjadi ketergantungan, dituduh menjadikan orang yang berperilaku kompulsif, tak acuh pada kegiatan lain. Dan memunculkan gejala aneh, seperti rasa tak tenang pada saat keinginan bermain tidak dipenuhi. (*Ridwan Syahran ; 2015 : 85).*

II.4. *Database*

Database merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan diperangkat lunak untuk memanipulasinya (Jogiyanto HM : 1999:711). *Database* merupakan salah satu komponen yang sangat penting dalam sistem informasi, karena merupakan basis sistem dalam menyediakan informasi bagi para pemakai.(*Indra Warman, M.Kom ; 2012 ; 45*)

II.4.1. Jenis-Jenis *Database*

Database ada dua jenis menurut *Indra Warman 2012* yaitu :

1. Database hirarki

Yaitu suatu data yang tersusun dengan bentuk hirarki pohon. Susunan yang seperti ini terdiri dari beberapa unsure komponen yang saling mempengaruhi dan tidak dapat dipisahkan, jenis database ini merupakan hubungan satu komponen dengan banyak komponen. (*Indra Warman ; 2012 : 45*)

2. Database Relasi

Adalah suatu data yang disusun dalam bentuk tabel yang terdiri dari dua definisi dan tersusun secara terstruktur. Bentuk susunan dua dimensi ini terdiri dari beberapa kolom dan record yang tersusun berbentuk baris dari kiri kekanan. Data-data yang susunannya berbentuk barus adalah susunan yang menurun kebawah. Dimana pada setiap baris berisikan data- data yang saling berkaitan satu sama lainnya. Artinya setiap pemasukan data yang tersimpan pada field merupakan kesatuan dalam bentuk satu baris.(*Indra Warman ; 2012 : 45*)

II.4.2. Komponen Komponen *Database*

Komponen utama dari sistem *database* terdiri atas beberapa bagian menurut (*Indra Warman ; 2012 : 45*) yaitu sebagai berikut :

1. Data : diutamakan data yang bersifat *integrity* (kesatuan) dan *Share* (pemakaian bersama).
2. Hardware : semua yang menyangkut media penyimpanan eksternal, piranti input dan output.
3. Software : berupa database manajemen sistem seperti SQL, yang merupakan penghubung antara alumni dengan data yang tersimpan didalam media penyimpanan secara fisik.
4. Database : merupakan kumpulan dari file yang saling berhubungan satu dengan yang lainnya. Database secara fisik terdapat dalam media penyimpanan seperti sistem komputer.

II.5. *SQL Server*

SQL (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan 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* (DDL) dan Data *Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap sistem manajemen basis data (SMBD), namun secara umum implementasi setiap bahasa ini memiliki bentuk standar yang ditetapkan oleh ANSI. (*Adelia ; 2011 : 115*)

1. Data *Definition 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 menurut (*Adelia ; 2011 : 115*) adalah :

   a. *CREATE* untuk membuat objek baru.
   b. *USE* untuk menggunakan objek.
   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 menurut (*Adelia ; 2011 : 115*) 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. *Visual Basic*

Visual Basic pada dasarnya adalah sebuah bahasa pemrograman komputer, bahasa pemgroman adalah peritah-perintah atau instruksi-instruksi yang dimegerti oleh komputer untuk melakukan tugas-tugas tertentu. Visual Basic (yang sering disingkat dengan VB) selain disebut sebagai sebuah bahasa pemograman, juga

sering disebut sebagai sarana (*tool*) untuk menghasilkan program-program aplikasi berbasiskan *Windows*. Beberapa kemampuan atau dari *Visual Basic* diantaraya adalah:

1. Untuk membuat program aplikasi berbasiskan *Windows*.
2. Utuk membuat objek-objek pembantu seperti misalnya *Control ActiveX*, *file help*, aplikasi *internet* dan sebagainya.
3. Menguji program (*Debugging*) dan menghasilkan program akhir berakhiran *exe* yang bersifat *executable* atau dapat lansung dijalankan. (*Sophan Sophian ; 2014 : 37*)

Ada beberapa keuntungan dari Visual Basic diantaranya:

1. MS Visual Basic memungkinkan aplikasi pembuatan Graphical User Interface (GUI) atau pemograman yang menggunakan tampilan grafis sebagai alat komunikasi dengan pemakai.
2. Mempunyai fleksibilitas yang sangat baik berhubungan dengan aplikasi yang lain. Kemampuan ini didukung dengan digunakannya Object Linking and Embedding (OLE) yang memungkinkan pembuatan hubungan antara bagian fungsi atau dengan seluruh aplikasi lain.
3. MS Visual Basic sangat kompatibel dengan Visual Basic versi terdahulu.
4. MS Visual Basic juga mendukung penggunaan long file name atau nama variabel sampai sepanjang 255 character. (*Sophan Sophian ; 2014 : 37*)

II.7. UML *(Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan

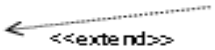
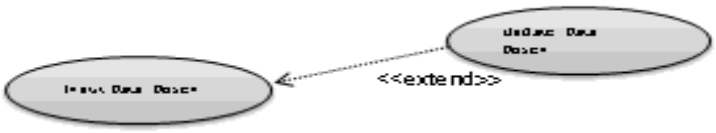
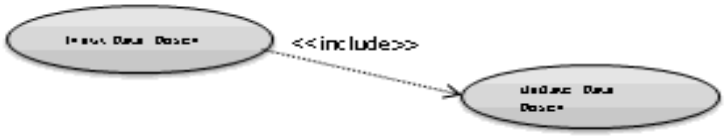
masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

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

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor

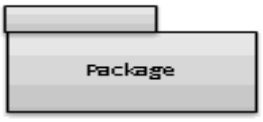
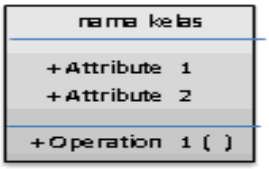
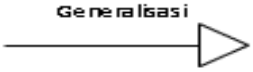
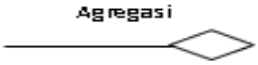
adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

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

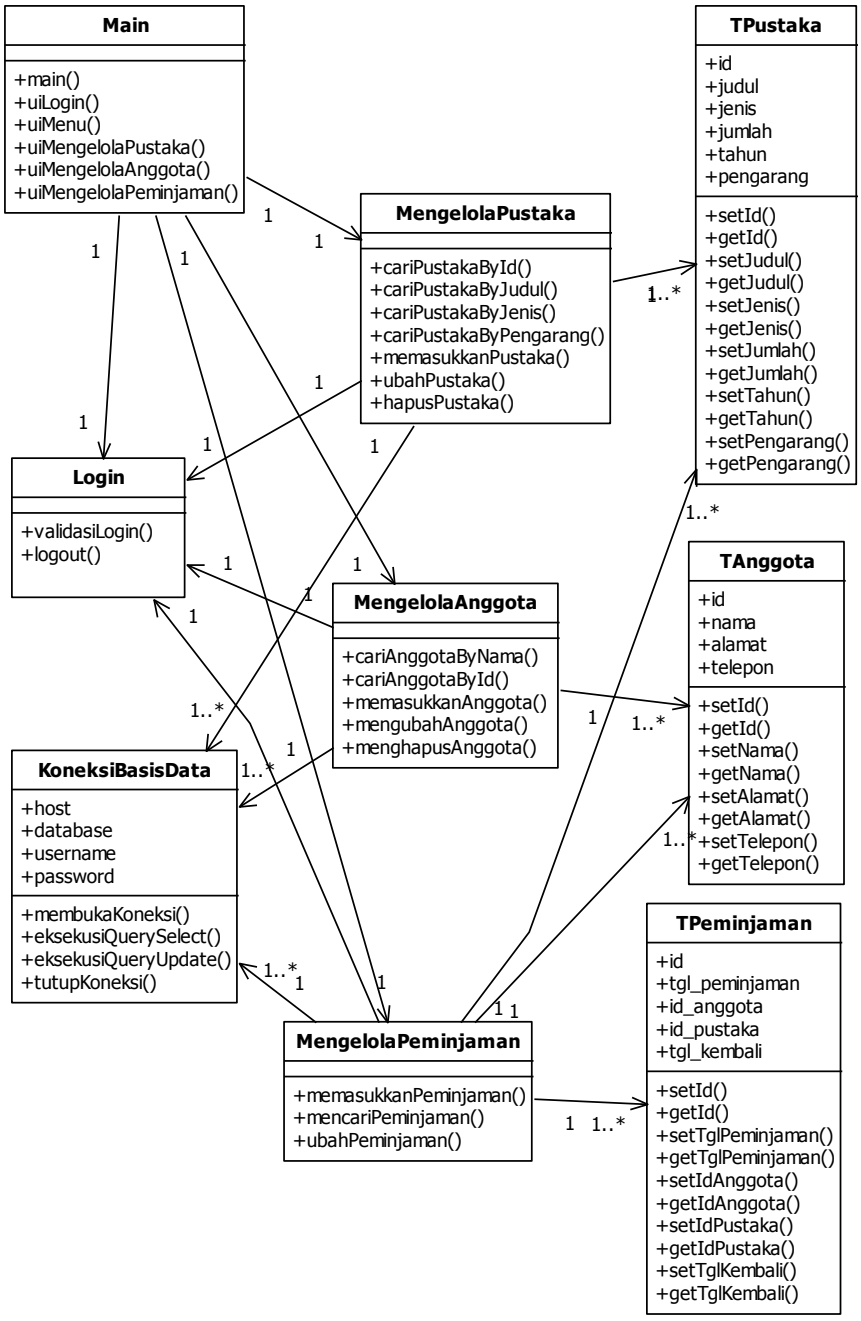
Gambar II.1. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

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

Gambar II.2. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



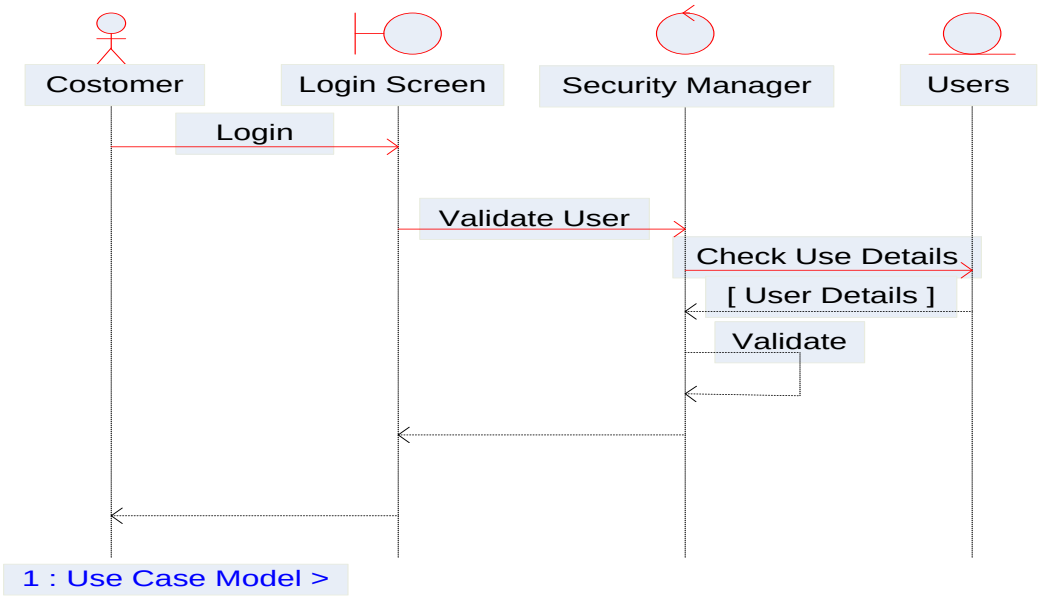
Gambar II.3. Contoh Class Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

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



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

4. *Activity Diagram*

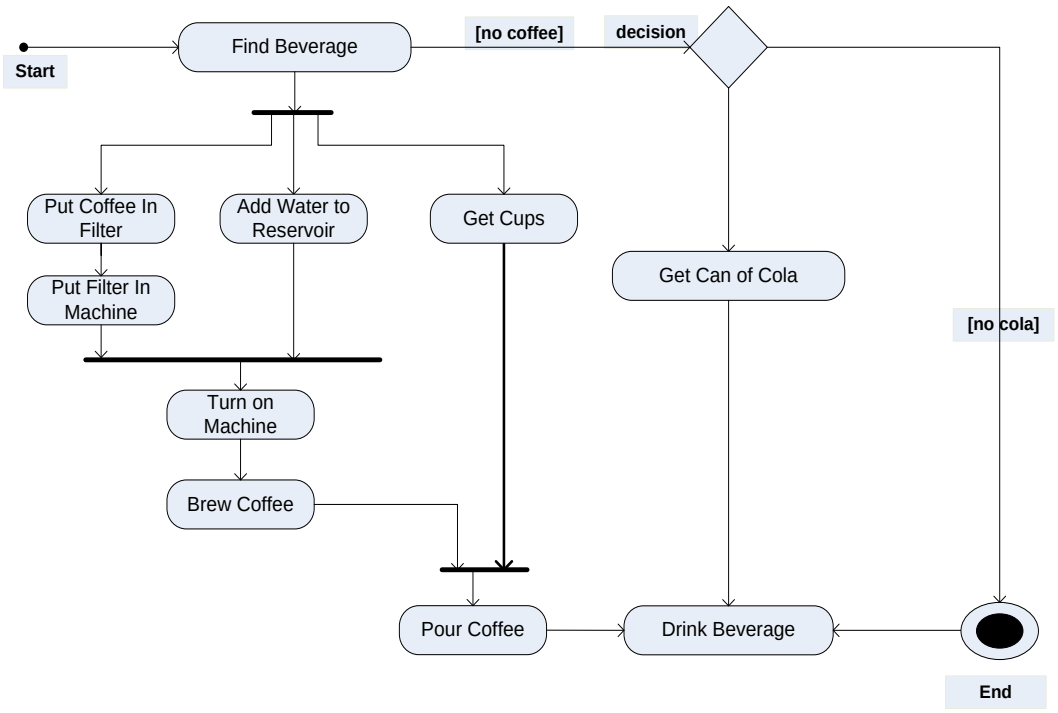
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

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

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



Gambar II.5. Activity Diagram

Sumber : (Yuni Sugiarti ; 2013 : 76)