BAB 2

LANDASAN TEORI

Teknologi informasi sudah menjadi bagian dalam kehidupan manusia untuk melakukan kegiatan sehari-hari. Teknologi informasi berfungsi untuk mencari informasi yang dibutuhkan melalui media-media yang tersedia. Banyak media yang mendukung kegiatan Teknologi informasi tersebut, seperti komputer, *mobile phone*, *laptop*, dan televisi.

Dengan berkembangnya teknologi informasi banyak tersedia aplikasi yang mendukung kegiatan manusia dalam berbisnis, hiburan, dan bersosialisasi. Dengan keterbatasan media yang bisa digunakan, maka manusia lebih memilih media informasi yang bisa digunakan dimana saja. Dan yang mendukung hal tersebut adalah *mobile phone*. Saat ini banyak *mobile phone* dengan berbagai macam platform, seperti android, blackberry, iOS, symbian, dan windows *mobile*. Dengan dukungan dari media *mobile phone* dan aplikasi yang tersedia, dapat memudahkan manusia untuk mencari informasi yang diinginkan. Salah satunya adalah informasi untuk mencari tempat wisata kuliner.

2.1 Teknologi Informasi

Definisi teknologi informasi secara umum adalah teknologi apapun yang dapat membantu manusia dalam membuat, mengubah, menyimpan mengomunikasikan dan menyebarkan informasi. Teknologi informasi menyatukan komputasi dan komunikasi berkecepatan tinggi untuk data, suara, dan video. Teknologi informasi bukan hanya komputer, tetapi semua teknologi

yang dapat memberi informasi seperti ponsel, tv, telpon dan seterusnya (William & Sawyer, 2007).

2.2 Database

2.2.1 Definisi Database

Database adalah Memberikan sekumpulan data yang logis, dan deskripsi dari data tersebut, yang dirancang untuk memberikan informasi yang dibutuhkan pada suatu organisasi. Database adalah tunggal, yang berarti tempat penyimpanan data yang besar dan dapat digunakan secara bersamaan oleh departemen dan *user*. Dengan file terputus dengan data yang berlebihan, semua data terintegrasi dengan jumlah duplikasi yang minimum.

Pendekatan yang diambil sistem database, dimana definisi suatu data terpisah dari sebuah program aplikasi, sama seperti pendekatan yang diambil pembuatan *software* modern, dimana definisi internal suatu objek dan definisi eksternal yang terpisah tersedia. Objek dari *user* hanya dapat melihat definisi eksternal dan tidak menyadari bagaimana objek didefinisikan dan bagaimana fungsinya. Salah satu kelebihan dari pendekatan ini disebut ***data abstraction***.

Istilah terakhir dari definisi database yang dimaksud *logically related* adalah ketika menganalisis sebuah informasi yang diperlukan organisasi, mencoba mendefinisikan entitas, atribut, dan hubungan. Entitas adalah objek yang berbeda (orang, tempat, benda, konsep, atau peristiwa) dalam organisasi yang akan diwakili dalam database. Atribut adalah properti yang menggambarkan beberapa aspek dari objek yang ingin disimpan, dan hubungan asosiasi antara entitas. (Connolly & Begg, 2005, p. 15).

2.2.2 Database Management System

Database management system atau DBMS adalah sebuah sistem *software* yang memungkinkan *user* untuk mendefinisikan, membuat, memelihara, dan kontrol akses ke database. DBMS merupakan *software* yang dapat berinteraksi dengan database dari program aplikasi *user.* Secara khusus, DBMS menyediakan beberapa fasilitas sebagai berikut:

- Hal ini memungkinkan *user* untuk mendefinisikan database, biasanya dengan ***Data Definition Language*** (DDL). DDL memungkinkan *user* untuk menentukan tipe data, struktur, dan batasan pada data yang akan disimpan dakan database.

Hal ini memungkinkan *user* untuk memasukkan, mengubah, menghapus, dan mengambil data dari database, biasanya melalui ***Data Manipulation Language*** (DML).Memiliki pusat penyimpanan untuk semua data dan deskripsi data memungkinkan DML untuk menyediakan fasilitas permintaan data, yang disebut *query language*. Penyediaan *query language* mengurangi permasalahan dengan sistem berbasis file dimana *user* harus bekerja dengan set *query* yang tetap atau ada penyebaran program, memberikan masalah utama pada *software* manajemen. *Query language* yang umum adalah ***Structured Query Language*** (SQL) yang sekarang menjadi bahasa standart untuk DBMS relasional. (Connolly & Begg, 2005, p. 16).

- Hal ini menyediakan akses kontrol untuk database. Sebagai contoh:

- Sistem keamanan, dimana mencegah *user* yang tidak sah mengakses database.
- Sistem integrasi, yang mempertahankan konsistensi data yang disimpan.
- Sistem kontrol *concurrency*, yang memungkinkan berbagi akses database.
- Sistem kontrol *recovery*, yang mengembalikan database ke keadaan yang konsisten sebelumnya yang diakibatkan kerusakan *hardware* atau *software*.
- *User-accessible catalog*, yang berisi deskripsi dari data dalam database.

2.2.3 Entity Relationship Modeling

A. *Entity Type*

Menurut Thomas Connolly dan Begg (2005, p.372), *entity type* adalah sekumpulan obyek dengan properti yang sama yang diidentifikasikan dengan keberadaan yang independen di perusahaan. Sebuah *entity* mempunyai kumpulan properti dan nilai dari properti tersebut mengidentifikasikan *entity* secara unik.

B. *Relationship Type*

Menurut Thomas Connolly dan Begg (2005, p.374) *relationship type* adalah sebuah relasi yang bermakna antara beberapa jenis *entity*. Setiap *relationship type* diberikan nama yang menjelaskan fungsinya. Ada *relationship occurrence*, yaitu sebuah relasi unik yang diidentifikasi mencakup satu kejadian dari setiap *entity* yang berpartisipasi.

a) *Degree of Relationship Type*

Menurut Thomas Connolly dan Begg (2005, p.376) *degree of relationship type* adalah jumlah tipe *entity* yang berpartisipasi di dalam sebuah relasi.

Terdapat beberapa macam relasi berdasarkan derajatnya, yaitu :

a. *Binary* : sebuah relasi dengan derajat dua
b. *Ternary* : sebuah relasi dengan derajat tiga
c. *Quarternary* : sebuah relasi dengan derajat empat

b) *Recursive Relationship*

Menurut Thomas Connolly dan Begg (2005, p.378) *recursive relationship* adalah sebuah tipe relasi dimana ada tipe *entity* yang sama berpartisipasi lebih dari satu kali dengan peran yang berbeda.

C. *Attributes*

Menurut Thomas Connolly dan Begg (2005, p.379) *attributes* adalah sebuah *property* khusus pada sebuah *entity* atau pada tipe relasi.

Setiap *attribute* dihubungkan dengan sebuah set nilai yang disebut *domain*. Jadi, *Attribute domain* adalah sebuah set nilai untuk satu atau lebih *attribute*.

a) *Simple and Composite Attributes*

Menurut Thomas Connolly dan Begg (2005, p.379) *simple attribute* adalah sebuah *attribute* yang terdiri dari sebuah komponen dengan keberadaan yang independen.

Menurut Thomas Connolly dan Begg (2010, p.380) *composite attribute* adalah sebuah *attribute* yang terdiri dari beberapa komponen, dimana setiap komponen tersebut memiliki keberadaan yang independen.

b) *Single-valued and Multi-valued Attributes*

Menurut Thomas Connolly dan Begg (2005, p.380), *single-valued attributes* adalah sebuah *attribute* yang memiliki sebuah nilai untuk setiap kejadian pada tipe *entity*.

Menurut Thomas Connolly dan Begg (2010, p.380), *multi-valued attributes* adalah sebuah *attribute* yang memiliki banyak nilai untuk setiap kejadian pada tipe *entity*.

c) *Derived Attributes*

Menurut Thomas Connolly dan Begg (2005, p.380), *derived attributes* adalah sebuah *attribute* yang memiliki nilai yang berasal dari *attribute* lain yang berhubungan, dan tidak harus berasal dari satu tipe *entity* yang sama.

D. *Key*

Menurut Thomas Connolly dan Begg (2005, p.381) *keys* dapat dibedakan menjadi 3, yaitu :

a) *Candidate Key*

Merupakan sebuah kumpulan minimal *attribute* yang secara unik mengidentifikasi setiap kejadian pada sebuah tipe *entity*.

b) *Primary Key*

Merupakan *candidate key* yang dipilih untuk secara unik mengidentifikasi setiap kejadian pada sebuah tipe *entity.*

c) *Composite Key*

Merupakan sebuah *candidate key* yang memiliki dua atau lebih *attribute*.

Ada dua *key* lain yaitu *alternate* dan *foreign key* :

a) *Alternate Key*

Merupakan suatu *candidate key* yang tidak terpilih yang dapat dijadikan sebagai *primary key*

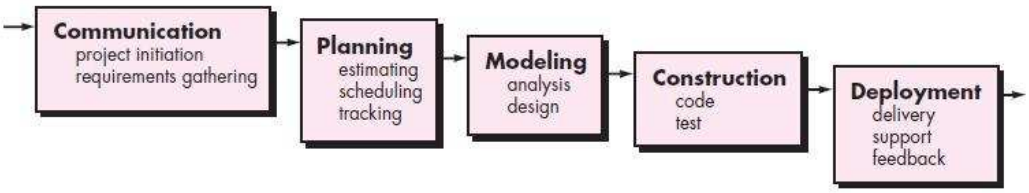
b) *Foreign Key*

Merupakan sebuah kumpulan *field* dalam satu relasi yang digunakan untuk menunjuk ke suatu baris pada relasi lain (merupakan *primary key*).

2.3 Waterfall Model

Menurut Roger S. Pressman (2010, p39-44). Model ini sering disebut dengan "*classic life cycle*" atau model waterfall. Model ini adalah model pengembangan software yang paling tua. Model ini melakukan pendekatan secara sistematis dan urut mulai dari level *Communication,* lalu ke tahap selanjutnya yaitu *Planning, Modelling, Construction,* hingga ke tahap akhir yaitu *Deployment.* Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu

selesainya tahap sebelumnya dan berjalan berurutan. Secara umum tahapan pada model waterfall dapat dilihat pada gambar berikut :



Gambar 2.2 *Waterfall* Model

Berikut penjelasan lengkap dari lima tahapan model waterfall menurut Pressman (2010, p15) :

1. *Communication*

Pada tahap ini pengembang terlebih dahulu berkomunikasi dengan pelanggan, sebelum pekerjaan teknis untuk pembuatan sistem dimulai. Hal ini penting untuk mengetahui sistem apa yang akan dibuat.

2. *Planning*

Tahap ini berguna untuk dapat mengatur kinerja para *software engineer*, mengetahui soal resiko yang dihadapi nanti, serta mengetahui apa saja sumber daya yang dibutuhkan, menghasilkan apa, serta yang paling penting adalah mengatur jadwal pembuatan perangkat lunak yang efektif.

3. *Modeling*

Tahap ini berguna untuk sebuah model yang akan dibuat agar lebih memahami perangkat lunak atau sistem yang akan dihasilkan. Serta dengan tahap pemodelan, *software engineer* juga dapat lebih mengetahui desain perangkat lunak atau sistem agar dapat menyelesaikan masalah yang sedang berlangsung.

4. *Construction*

Tahap ini berguna untuk menggabungkan antara melakukan *coding* dan *testing* yang dibutuhkan dalam pembuatan suatu sistem untuk mengetahui kesalahan apa saja yang terdapat pada sistem atau perangkat lunak tersebut.

5. *Deployment*

Pada tahap ini perangkat lunak yang telah selesai akan dipasarkan ke pelanggan. Selain itu juga mengumpulkan kritik dan saran dari pelanggan yang telah mencoba sistem atau perangkat lunak yang telah dibuat.

2.4 Software

Perangkat lunak adalah program yang terdiri dari instruksi-instruksi elektronik yang meminta komputer untuk mengerjakan tugas-tugas tertentu setahap demi setahap. Tanpa perangkat lunak perangkat keras tidak bisa beroperasi. (Williams & Sawyer, 2007)

2.5 Object Oriented Programming

Menurut G. M. Seed (2001, p9) Object-oriented Programming (OOP) adalah bahasa pemrograman yang terdiri dari obyek. Dan setiap obyek mempunyai karakteristik atau data sendiri. Dan data tersebut secara terus menerus mengirimkan pesan dan menerima pesan dari objek lain.. Bisa dibilang OOP sendiri adalah bahasa pemrograman yang lebih menggunakan objek dan data dari pada action dan logika.

Di dalam OOP, seorang programmer harus mengenal objek yang ingin di manipulasi, dan harus juga tau bagaimana relasi setiap obyek yang berhubungan dengan obyek lain. OOP memiliki beberapa karakteristik, yaitu:

- Di dalam OOP semuanya adalah objek.
- Setiap objek memiliki pesan yang merupakan permintaan atas sekumpulan aksi dengan semua argument yang diperlukan untuk menyelesaikan suatu tugas tertentu.
- Setiap objek adalah wakil atau representasi suatu kelas. Sebuah kelas dapat mewakili sekelompok objek yang sama.
- Kelas adalah kumpulan tingkah laku yang berkaitan dengan suatu objek.
- Setiap objek umumnya memiliki 3 sifat. Yaitu keadaan, operasi, dan identitas objek.
- Operasi adalah tindakan yang dilakukan oleh sebuah objek.
- Keadaan adalah koleksi dari seluruh informasi yang dimiliki oleh objek.
- Informasi yang terkandung pada objek pada akhirnya akan memberikan identitas khusus yang membedakan suatu objek dengan objek lainnya.

OOP memiliki beberapa konsep dasar yang harus dimengerti oleh seorang programmer. Konsep dasar Object-oriented Programming adalah sebagai berikut:

- Abstraction

  Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari “pelaku” abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan

berkomunikasi dengan objek lainnya dalam sistem tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

- Encapsulation

  Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak. Hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut

- Inheritance

  Inheritance adalah mendefinisikan kelas baru dengan mewariskan sifat dari kelas yang sudah ada. Penurunan sifat bisa dilakukan secara bertingkat, jadi semakin kelas tersebut diturunkan, semakin spesifik kelas yang berada di bawah. Dengan konsep penurunan sifat, programmer dapat menggunakan kode di kelas parent berulang kali di kelas-kelas turunannya tanpa menulis ulang kode yang telah ditulis.

- Polymorphism

  Polymorphism adalah kemampuan objek-objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama. Polymorphism juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan pada objek tersebut, tergantung letak objek tersebut pada jenjan pewarisan. Method tersebut adalah Overriding dan Overloading.

2.6 MySql

MySql adalah Relational *Database* Management System yang didistribusikan secara gratis dibawah lisensi GPL(General Public System). Dimana setiap orang bebas untuk menggunakan MySql. MySql sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian *database* terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Keandalan suatu *database* dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh user maupun program-program aplikasinya. Sebagai *database* server.

MySql memiliki beberapa keistimewaan, antara lain adalah :

- **Portabilitas**. MySql dapat berjalan stabil pada berbagai sistem operasi
- **Open Source**. MySql didistribusikan secara open source, sehingga dapat digunakan secara cuma-cuma.
- **Keamanan**. MySql memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, da izin akses pengguna dengan sistem perizinan uang mendetail, serta sandi terenkripsi

Untuk mengakses *database* di MySql, bahasa yang digunakan dibagi menjadi 3 bagian, yaitu DDL, DML, dan DCL.

- **Data Definition Language (DDL)**

  DDL (*Data Definition Language*) adalah kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut *database*, tabel, atribut (kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar

tabel. Perintah yang termasuk dalam kelompok DDL adalah CREATE, ALTER, DROP.

- **Data Manipulation Language (DML)**

  DML (*Data Manipulation Language*) adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam *database*. Misalnya untuk pengambilan, penyisipan, perubahan, dan penghapusan data. Perintah yang masuk kategori DML adalah SELECT, INSERT, UPDATE, DELETE.

2.7 HTML

Hypertext Markup Language (HTML) adalah sekumpulan perintah khusus yang dipakai untuk menentukan struktur, bentuk, dan link pada dokumen ke dokumen multimedia lain di web. HTML merupakan tipe bahasa pemformatan yang menambahkan perintah dalam dokumen teks standar ASCII untuk memberikan tampilan teks dan grafis dua dimensi yang terintegrasi. Hypertext digunakan untuk me-link display.

2.8 PHP

PHP adalah bahasa pemrograman *server-side* dan *cross-platform technology*. PHP dikatakan *server-side* karena mengacu pada fakta bahwa semua yang dikerjakan PHP diproses di dalam server. PHP juga *cross-platform* karena bisa dijalankan di semua sistem operasi, termasuk windows, linux, dan macintosh. (Larry Edward Ullman, 2003)

PHP banyak digunakan untuk membuat dynamic websites, karena PHP lebih baik, lebih cepat, dan lebih mudah di pelajari daripada bahasa pemrograman yang lain.

Diantara banyaknya bahasa pemrograman yang lain, seperti ASP.NET dan JSP, PHP memiliki beberapa kelebihan. Yaitu:

1. PHP adalah sebuah bahasa pemrograman yang tidak melakukan kompilasi dalam penggunaanya.
2. Banyak Web Server yang mendukung PHP.
3. Dalam sisi pengembangan lebih mudah.
4. PHP lebih mudah dipahami daripada bahasa lain.
5. PHP adalah bahasa yang open source. Bisa digunakan di berbagai sistem operasi.

2.9 Interaksi Manusia dan Komputer

2.9.1 Pengertian Interaksi Manusia dan Komputer

Interaksi manusia dan komputer atau *human computer interact* adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia serta studi fenomena-fenomena besar yang berhubungan dengannya (ACM SIGCHI). Interaksi manusia dan komputer berfokus pada perancangan dan evaluasi antarmuka pemakai (*user interface).*

2.9.2 Delapan Aturan Emas Perancangan Antar Muka

Menurut Ben Shneiderman dan Plaisant (2010, p.88), dalam perancangan user interface, digunakan 8 aturan emas perancangan atau yang sering dikenal dengan eight golden rules, yaitu :

1. Berusaha untuk konsisten.
2. Memenuhi kebutuhan bersama
3. Memberikan umpan balik yang informatif.
4. Merancang dialog yang memberikan penutupan.
5. Memberikan pencegah kesalahan dan penanganan kesalahan yang sederhana.
6. Memungkinkan pembalikkan aksi yang mudah.
7. Mendukung pusat kendali internal (internal locus of control).
8. Mengurangi beban ingatan jangka pendek.

2.10 UML

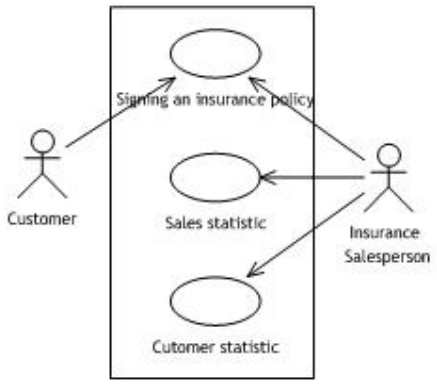
Menurut Whitten dan Bentley (2007, p371) “UML adalah satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek.” UML terdiri dari beberapa diagram, antara lain:

2.10.1 Use Case Diagram

Use Case diagram menggambarkan interaksi antara sistem dan sistem eksternal, serta user. (Whitten & Bentley, 2007, p382). Dengan kata lain, Use Case diagram menggambarkan siapa yang akan menggunakan sistem dan dengan jalan apa yang diinginkan user untuk berinteraksi dengan sistem. Selain itu, Use Case digunakan untuk secara tekstual menggambarkan urutan langkah setiap interaksi tersebut.

Use Case biasanya di gambarkan dengan minimal satu aktor dan operasinya. Aktor biasanya diwakilkan oleh gambar orang. Dan

operasi yang dilakukan digambarkan di dalam sebuah elips. Aktor dan operasi yang dilakukannya dihubungkan oleh sebuah garis lurus. Berikut ini adalah contoh gambar dari sebuah use case:



Gambar 2.3 Contoh Use Case Diagram

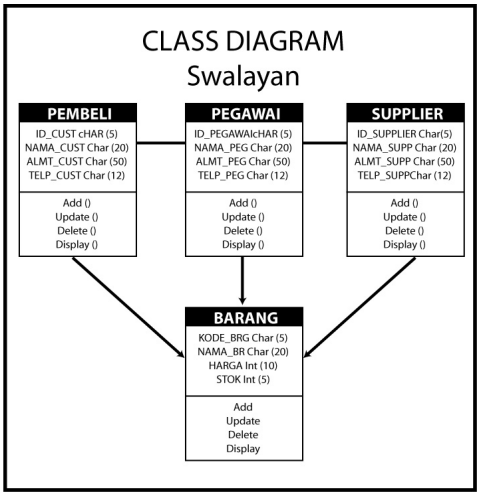
2.10.2 Class Diagram

Whitten dan Bentley (2007, p382) menjelaskan bahwa Class diagram menggambarkan struktur dari sistem. Serta menampilkan Class Object yang berada di dalam sistem serta hubungan antara objek tersebut dan objek lainnya.. Class Diagram terdiri dari beberapa banyak class di dalam sebuah sistem yang memiliki hubungan satu sama lain. Class sendiri adalah sebuah paket yang berisi 3 bagian, yaitu:

a. Nama class, yang isinya adalah nama dari class tersebut

b. Atribut, merupakan property atau isi yang dimiliki oleh sebuah class

c. Operasi, merupakan perintah yang bisa dilakukan oleh sebuah class atau yang dapat dilakukan oleh class lain terhadap sebuah class



Gambar 2.4 Contoh Class Diagram

2.10.3 Statechart Diagram

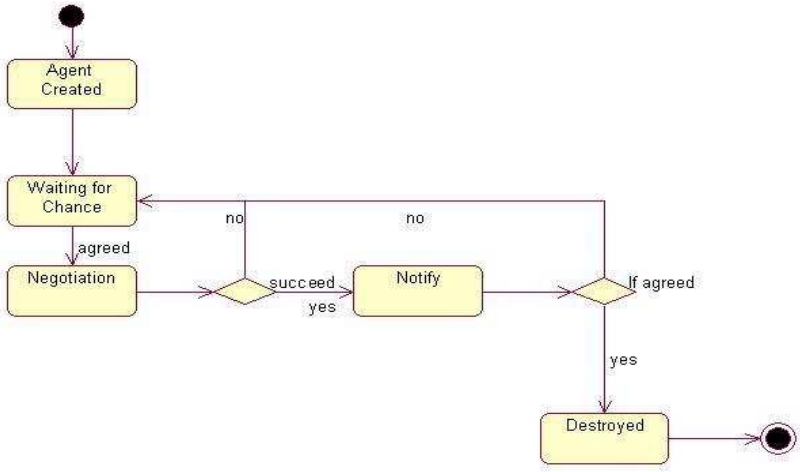
Menurut Whitten dan Bentley (2008, p559) Statechart Diagram adalah sebuah UML diagram yang menggambarkan kombinasi state dari sebuah objek dapat berjalan, peristiwa yang memicu transisi antara state, dan aturan-aturan yang mengatur transisi objek.

Statechart Diagram digambarkan melalui berbagai macam simbol. Berikut ini adalah gambar dari simbol-simbol tersebut:

Simbol	Keterangan
	Start (keadaan awal)
	End (keadaan akhir)

	Aktivitas
	Transisi
	Decision Point

Gambar 2.5 Simbol-Simbol Statechart Diagram.



Gambar 2.6 Contoh Statechart Diagram.

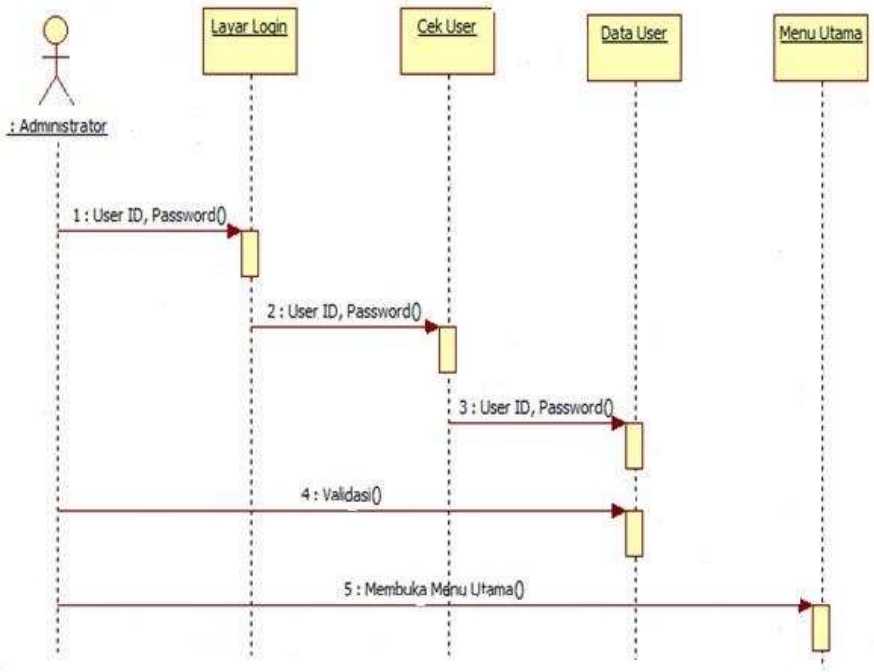
2.10.4. Sequence Diagram

Menurut Whitten dan Bentley (2007, p382) Sequence diagram adalah diagram yang menggambarkan bagaimana objek berinteraksi dengan objek lainnya melalui pesan dalam suatu eksekusi dari sebuah Use Case atau sebuah operasi. Sequence diagram juga

mengilustrasikan bagaimana pesan dikirim dan diterima oleh objek dan terjadi di dalam suatu sequence.

Sequence Diagram dibuat dengan cara menggambarkan obyek yang berpartisipasi di dalam interaksi dengan sistem. Obyek-obyek yang ada di tuliskan menyamping dari kiri ke kanan. Aktor yang menginisiasi interaksi biasanya di taruh di paling kiri. Sedangkan dimensi vertikal di diagram ini berarti waktu dari obyek dan aktor yang ada. Bagian paling atas diagram menjadi titik awal dan waktu berjalan kebawah sampai dengan bagian dasar diagram. Garis itu disebut lifeline. Lalu lifeline tersebut digambarkan menjadi kotak jika melakukan operasi. Kotak itu disebut activation box.

Pesan yang dikirim antar obyek digambarkan sebagai anak panah antara activation box pengirim dan penerima. Dan diatas anak panah tersebut ditulis pesan yang di kirim dan di terima. Berikut ini adalah contoh gambar dari sequence diagram:



Gambar 2.7 Contoh Sequence Diagram

2.11 Aplikasi *Mobile*

Aplikasi *mobile* adalah perangkat lunak yang dirancang khusus untuk bisa digunakan pada perangkat *mobile* seperti *smartphone* dan *tablet.* Aplikasi *mobile* biasanya didapat dengan cara mendownload dan diinstal oleh pengguna *smartphone* tersebut. Aplikasi *mobile* mempunyai beberapa tipe seperti : Aplikasi *mobile* web, aplikasi *native,* aplikasi HTML 5, dan aplikasi hybrid.

2.11.1 Aplikasi *Mobile* Web

Aplikasi *mobile* web merupakan aplikasi yang dapat digunakan melalui *web browser* yang disediakan oleh *smartphone*. Jika anda mempunyai website versi *mobile*, website anda dapat diakses melalui internet browser yang ada di *mobile phone* atau

smartphone seperti internet explorer, safari, opera, dan mozzila firefox.

2.11.2 Aplikasi Native

Aplikasi native adalah aplikasi *mobile phone* atau *smartphone* yang dapat digunakan tanpa harus menggunakan koneksi internet. Aplikasi native bergantung pada pengguna. Jika banyak pengguna menggunakan aplikasi ini developer harus mengembangkan dan menjaga aplikasi tersebut.

2.11.3 Aplikasi HTML 5

Aplikasi *mobile* yang dibuat dengan HTML5 dapat digunakan disemua jenis *mobile platform* yang ada saat ini dengan sekali membuatnya. Kelebihan dari aplikasi ini yaitu fleksibel. Tidak seperti aplikasi native, aplikasi HTML 5 tidak harus membuat versi terbarunya disetiap *platform.* Aplikasi ini juga dapat ditambahkan *icon* di *desktop* seperti aplikasi native.

2.11.4 Aplikasi Hybrid

Aplikasi *hybrid* merupakan gabungan dari aplikasi native dengan aplikasi HTML 5. Aplikasi *hybrid* dibuat dengan teknologi web, dan dikemas sesuai *platform* yang ada. Dengan menggunkan rangka aplikasi native membuat aplikasi hybrid ini terlihat seperti aplikasi native.