BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Informasi adalah suatu kerangka yang sumber daya-sumber daya (manusia dan komputer) dikoordinasikan untuk mengkonversi masukan (data) menjadi keluaran (informasi) dalam rangka pencapaian tujuan organisasi (McLeod, 1998). Sistem informasi memiliki sifat-sifat antara lain mempunyai jaringan komunikasi dan mempunyai tahapan-tahapan dan fungsi-fungsi konversi data. Sistem informasi manajemen adalah sekumpulan sistem yang saling berinteraksi, yang memberikan informasi baik untuk kepentingan operasi atau kegiatan manajerial (Goerge M. Scott, 1986).

Sistem informasi adalah keterpaduan sistem manusia-mesin untuk menyediakan informasi yang mendukung fungsi operasi dan menjamin pengambilan keputusan dalam sebuah organisasi (Davis, 1974). Sistem ini menggunakan perangkat keras dan perangkat lunak komputer, prosedur pedoman, model untuk analisis, perencanaan, pengawasan dan pengambilan keputusan dan suatu database. Yang terpenting dari aspek tersebut adalah keseluruhannya, karena sistem informasi manajemen akan melintasi seluruh sistem penyedia informasi di berbagai lapisan dalam organisasi. Oleh karena itu, perlu ditekankan bahwa sistem informasi manajemen adalah kumpulan sistem informasi dan bukan sistem keseluruhan (total sistem). Secara teoritis, komputer bukan prasyarat mutlak bagi sebuah sitem informasi manajemen, namun dalam praktek agaknya menjadi keniscayaan bahwa sistem informasi manajemen yang baik tidak akan ada tanpa bantuan pemrosesan dari sebuah komputer. Walaupun harus diingat bahwa di dalam sistem informasi manajemen akan selalu ada unsur komputer. Komputer diperlukan sebagai alat bantu dalam pengolahan data sistem informasi manajemen untuk mempercepat pekerjaan, mempermudah pekerjaan, dapat menangani pekerjaan dalam waktu yang bersamaan dengan teliti, cepat dan benar

sehingga didapat informasi yang tepat pada saat dibutuhkan dengan tingkat kesalahan yang kecil.

2.2 Algoritma *Breadth First Search*

Breadth First Search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara *preorder* yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetanggadengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

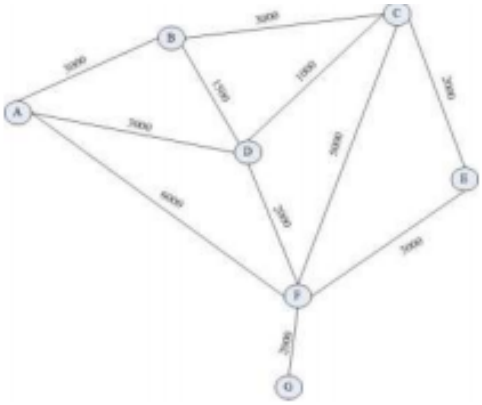
Algoritma ini memerlukan sebuah antrian *q* untuk menyimpan simpul yang telah dikunjungi. Simpul-simpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungi masuk ke dalam antrian hanya satu kali. Algoritma ini juga membutuhkan *tableboolean* untuk menyimpan simpul yang telah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali.

2.2.1 Cara Kerja Algoritma *Breadth First Search*

Dalam algoritma BFS, simpul anak yang telah dikunjung disimpan dalam antrian.

Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengan yang akan dikunjung kemudian sesuai urutan pengantrian untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya. Berikut langkah-langkah BFS :

1. Masukkan simpul ujung (akar) ke dalam antrian
2. Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian. 2.6.2 Contoh algoritma *breadth first search* Algoritma *Breadth First Search*penelusuran setiap titik dilakukan pada simpul yang bertetangga dengan simpul tersebut (simpul anak), seperti terlihat pada gambar 1 berikut.



Gambar 2.1 *Graph* untuk BFS dan *Tabu Search*

Misalnya menentukan jalur terpendek dengan *Breadth First Search* dari A ke E, maka berdasarkan gambar diatas kita dapat mengimplementasikan kedalam BFS untuk memperoleh jalur terpendek dengan langkah-langkah sebagai berikut.

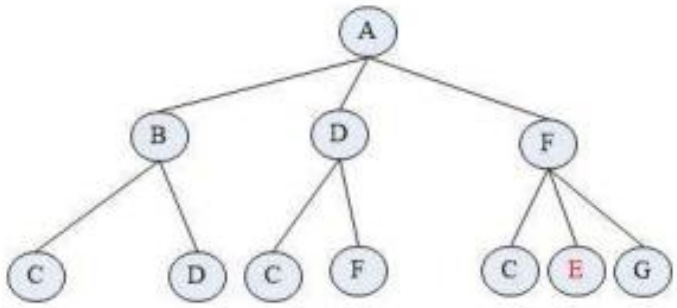
Langkah 1

Kita ambil state awal adalah A, A memiliki pilihan jalur yaitu menuju B dan D dan F sehingga kita dapat repsentasif pohon pencarian.

Langkah ke 2

Kita melakukan pencarian jalur terpendek berdasarkan konsep dari *breadth first search* seperti yang dijelaskan sebelumnya tujuan dari pencarian adalah menemukan titik E. Pencarian diawali dengan menelusuri titik pangkal yaitu titik A, karena titik A merupakan titik dengan level tertinggi maka penelusuran dilanjutkan dengan menjelajahi titik-titik pada level di bawahnya atau simpul yang bertetangga dengan simpul tersebut (simpul anak). Penelusuran selanjutnya adalah pada level kedua atau simpul yang bertetangga dengan simpul tersebut yaitu titik B, D dan F, Penelusuran selanjutnya adalah memasukkan simpul yang bertetangga dengan B yaitu C dan D Penelusuran selanjutnya adalah memasukkan simpul yang bertetangga dengan D yaitu C dan F, tahap selanjutnya memasukkan simpul yang bertetangga dengan F yaitu C dan E dan G. dalam diagram ini titik E

merupakan titik tujuan, maka dengan demikian proses pencarian berhenti, karena sudah menemukan solusi, seperti yang terlihat dalam gambar 2.1



Gambar 2.2 Proses penelesuran dengan metode *Breadth First Search*

berdasarkan penelusuran diatas dengan metode *Breadth First Search* ditemukan jalur terpendek dari A menuju E adalah A,F,E.

2.3 *Location Based Service* (LBS)

LBS adalah layanan berbasis lokasi, yaitu sebuah layanan berbasis internet yang berfungsi untuk mencari dengan teknologi Global Positioning Service (GPS) dan Google's cellbased location. Map dan layanan berbasis lokasi menggunakan lintang dan bujur untuk menentukan lokasi geografis. Android menyediakan geocoder yang mendukung forward dan reverse geocoding. Menggunakan geocoder, anda dapat mengkonversi nilai lintang bujur menjadi alamat dunia nyata atau sebaliknya. Location based service atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan.

1. *Unsur Utama LBS*
   a. *Location Manager* (*API Maps*): Menyediakan perangkat bagi sumber atau *source* untuk LBS, *Application Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan atau memanipulasi peta.
   b. *Location Providers* (*API Location*): Menyediakan teknologi pencarian lokasi yang digunakan oleh perangkat. API Location berhubungan dengan

data GPS (*Global Positioning System*) dan data lokasi *real-time*. *API Location* berada pada paket Android yaitu dalam paket "android.location". Lokasi, perpindahan, serta kedekatan dengan lokasi tertentu dapat ditentukan melalui *Location Manager.*

2. Komponen LBS

1. Mobile Device Piranti Mobile tersebut diantaranya adalah smartphone, PDA, dan lainnya yang berfungsi sebagai alat navigasi seperti alat navigasi kendaraan berbasis GPS.

2. Communication network Berupa jaringan telekomunikasi bergerak yang memindahkan data pengguna dari perangkat ke penyedia layanan.

3. Position Component Berfungsi untuk menenentukan posisi pengguna layanan. Posisi ini didapatkan dengan jaringan telekomunikasi GPS.

4. Service dan Content Provider Penyedia layanan yang yang menyediakan berbagai macam layanan seperti pencarian rute, kalkulasi posisi, dan lainnya. (Alfeno 2017)

2.4 Global Positioning System (GPS)

GPS adalah singkatan dari Global Positioning System,yang merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Cara kerja GPS secara logik ada 5 langkah:

1. Memakai perhitungan "triangulation" dari satelit.

2. Untuk perhitungan "triangulation", GPS mengukur jarak menggunakan travel time sinyal radio.

3. Untuk mengukur travel time, GPS memerlukan memerlukan akurasi waktu yang tinggi.

4. Untuk perhitungan jarak, kita harus tahu dengan pasti posisi satelit dan ketingian pada orbitnya.

5. Terakhir harus menggoreksi delay sinyal waktu perjalanan di atmosfer sampai diterima reciever.

Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima (receiver) di permukaan, dimana GPS receiver ini akan mengumpulkan informasi dari satelit GPS. Sebuah GPS receiver harus mengunci sinyal minimal tiga satelit untuk menghitung posisi 2D (latitude dan longitude) dan track pergerakan. Jika GPS reciever dapat menerima empat atau lebih satelit, maka dapat menghitung posisi 3D (latitude,longitude dan altitude). Jika sudah dapat menentukan posisi user, selanjutnya GPS dapat menghitung informasi lain,seperti kecepatan, arah yang dituju, jalur, tujuan perjalanan,jarak tujuan, matahari terbit dan matahari terbenam dan lainlain. Sinyal yang dikirimkan oleh satelit ke GPS akan digunakan untuk menghitung waktu perjalanan (travel time). Waktu perjalanan ini sering juga disebut sebagai Time of Arrival (TOA). Sesuai dengan prinsip fisika, bahwa untuk mengukur jarak dapat diperoleh dari waktu dikalikan dengan cepat rambat sinyal. Dari beberapa pemakaian GPS di atas dikategorikan menjadi:

a. Waktu. GPS receiver menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
b. Lokasi. GPS memberikan informasi lokasi:

1) Latitude

2) Longitude

3) Altitude

c. Kecepatan. Ketika berpindah tempat, GPS dapat menunjukkan informasi kecepatan berpindah tersebut.

d. Arah perjalanan. GPS dapat menunjukkan arah tujuan. Simpan lokasi. Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS receiver.

e. Komulasi data. GPS receiver dapat menyimpan informasi track, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan, dan sebagainya.

f. Tracking. Membantu untuk memonitoring pergerakan obyek. Membantu memetakan posisi tertentu, dan perhitungan jaringan terdekat**.** (Alfeno 2017)

2.5 Latitude& Longitude

Latitude adalah garis yang melintang di antar kutub utara dan kutub selatan, yang menghubungkan antara sisi timur dan barat bagian bumi. Garis ini memiliki posisi membentangi bumi, sama halnya seperti garis *equator* (khatulistiwa), tetapi dengan kondisi nilai tertentu. Garis lintang inilah yang dijadikan ukuran dalam mengukur sisi utara-selatan koordinat suatu titik di belahan bumi.

Latitude dibedakan menjadi 2 wilayah, yaitu utara atau yang biasa kita sebut lintang utara dan selatan atau yang biasa kita sebut lintang selatan, dimana nilai koordinat di bagian utara selalu positif dan nilai koordinat di bagian selatan adalah negatif.

Berikut nilai-nilai yang dijadikan patokan ukuran garis lintang ini.

1. Garis paling atas (kutub utara) = 90 derajat
2. Garis paling tengah (equator) = 0 derajat
3. Garis paling bawah (kutub selatan) = -90 derajat.

Dengan “mem-persamakan” derajat ke dalam bentuk satuan kilometer (km) maka ukurannya seperti ini :

1 derajat latitude = 111 km

1 menit latitude = 1.85 km

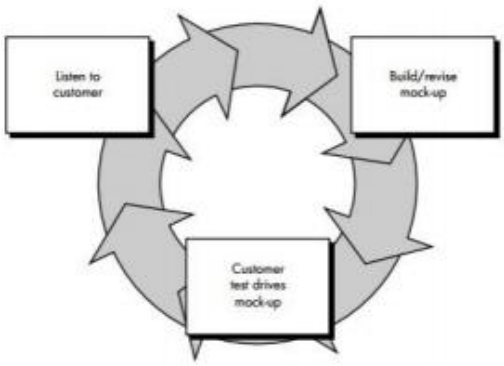
Sedangkan *longtitude* adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi (kutub). Garis bujur ini digunakna untuk mengukur sisi barat-timur koordinat suatu titik di belahan bumi. Sama seperit *equator* pada *latitude* yang berada ditengah dan memiliki nilai 0 (nol) derajat, pada *longitude*, garis tengah yang bernilai 0 (nol) derajat disebut garis *prime meridian* (garis bujur). Sedangkan garis yang berada paling kiri memiliki nilai -90 derajat, dan yang paling kanan memiliki nilai 90 derajat.

Longitude juga dibedakan menjadi 2 wilayah, yaitu bujur timur dan bujur barat, dimana koordinat yang berada di timur selalu bernilai negatif, dan sebaliknya yang berada di barat selalu positif. Nilai satuan ukuran derajat menjadi kilometer pada *longitude* juga sama seperti pada *latitude*.

Jadi, dalam metode pengukuran koordinat, suatu titik terlebih dulu diukur derajatnya berdasarkan *latitude* dan *longitude*-nya, setelah itu barulah di translasikan kedalam bentuk satuan kilometer, baik itu dalam format *degree* (DDD) maupun *degree-minutes-second* (DMS).(Mulyana, A. and Sofyan, S., 2015)

2.6 Metode Pengembangan Sistem *Prototype*

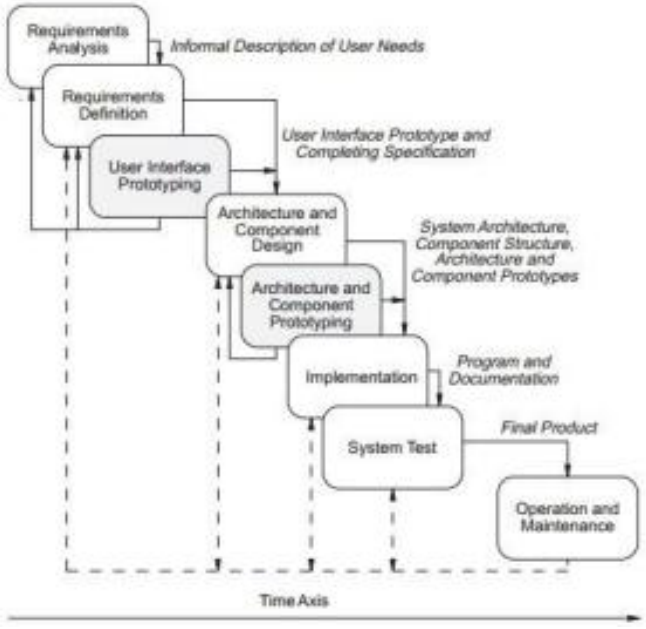
Sebuah *prototype* adalah versi awal dari system perangkat lunak yang digunakan untuk mendemonstrasikan konsep-konsep, percobaan rancangan, dan menemukan lebih banyak masalah dan solusi yang memungkinkan (Sommerville, 2011). Sistem *prototype* memperbolehkan pengguna untuk mengetahui bagaimana sistem berjalan dengan baik. Penggunaan metode *prototyping* di dalam penelitian ini bertujuan agar peneliti mendapatkan gambaran aplikasi yang akan dibangun melalui tahap pembangunan aplikasi *prototype* terlebih dahulu yang akan dievaluasi oleh user. Aplikasi *prototype* yang telah dievaluasi oleh user selanjutnya akan dijadikan acuan untuk membuat aplikasi yang dijadikan produk akhir sebagai *output* dari penelitian ini.



Gambar 2.3 *Prototyping Model*

Gambar 2 menjelaskan bahwa metode *prototyping* dimulai dengan mendengarkan kebutuhan dan masukan dari pengguna. Pengembang dan pengguna bertemu dan bersama-sama menentukan tujuan keseluruhan untuk perangkat lunak dan mengidentifikasi apapun persyaratan yang diperlukan. Lalu pengembang membuat sebuah gambaran tentang aplikasi yang selanjutnya dapat dipresentasikan kepada pelanggan. Gambaran tersebut berfokus pada representasi aspek-aspek aplikasi yang akan terlihat oleh pelanggan/pengguna. Beberapa keunggulan dalam menggunakan metode *prototyping* :

1. Pengembang sistem dan pengguna saling berkomunikasi khususnya dalam hal penyamaan persepsi terhadap pemodelan sistem yang akan menjadi dasar pengembangan system operasionalnya,
2. Pelanggan/pengguna ikut terlibat secara aktif dan berpartisipasi dalam menentukan model sistem dan sistem operasionalnya sehingga pelanggan/pengguna akan puas karena sistem yang dibuat sesuai dengan keinginan dan harapannya.
3. Sistem yang dibangun memiliki kualitas yang diinginkan karena sesuai dengan kebutuhan yang ada.



Gambar 2.4 Alur proses metode *Prototyping*

Gambar 3 menjelaskan mengenai alur pengembangan sistem dengan menggunakan metode *PrototypingOriented Software*. Pada tahap pertama, dilakukan analisis kebutuhan dan pendefinisian kebutuhan.Kebutuhan yang dimaksudkan disini adalah kebutuhan pelanggan/pengguna. Selanjutnya pada tahap kedua dilakukan pembuatan *prototype* dari aplikasi yang akan dibangun, mulai dari user interface *prototyping* dan dilanjutkan hingga penyusunan arsitektur dan komponen-komponen yang berkaitan dengan aplikasi yang akan dibangun. Selanjutnya dilakukan pengembangan sistem, dimana aplikasi akan dibangun sesuai dengan *prototype* yang telah dibuat sebelumnya, dan setelah aplikasi berhasil dibuat sesuai dengan kebutuhan maka dilakukan proses pengujian aplikasi sebelum aplikasi tersebut diimplementasikan. (Betri 2017)

2.7 *Black-Box Testing* (Pengujian Kotak Hitam)

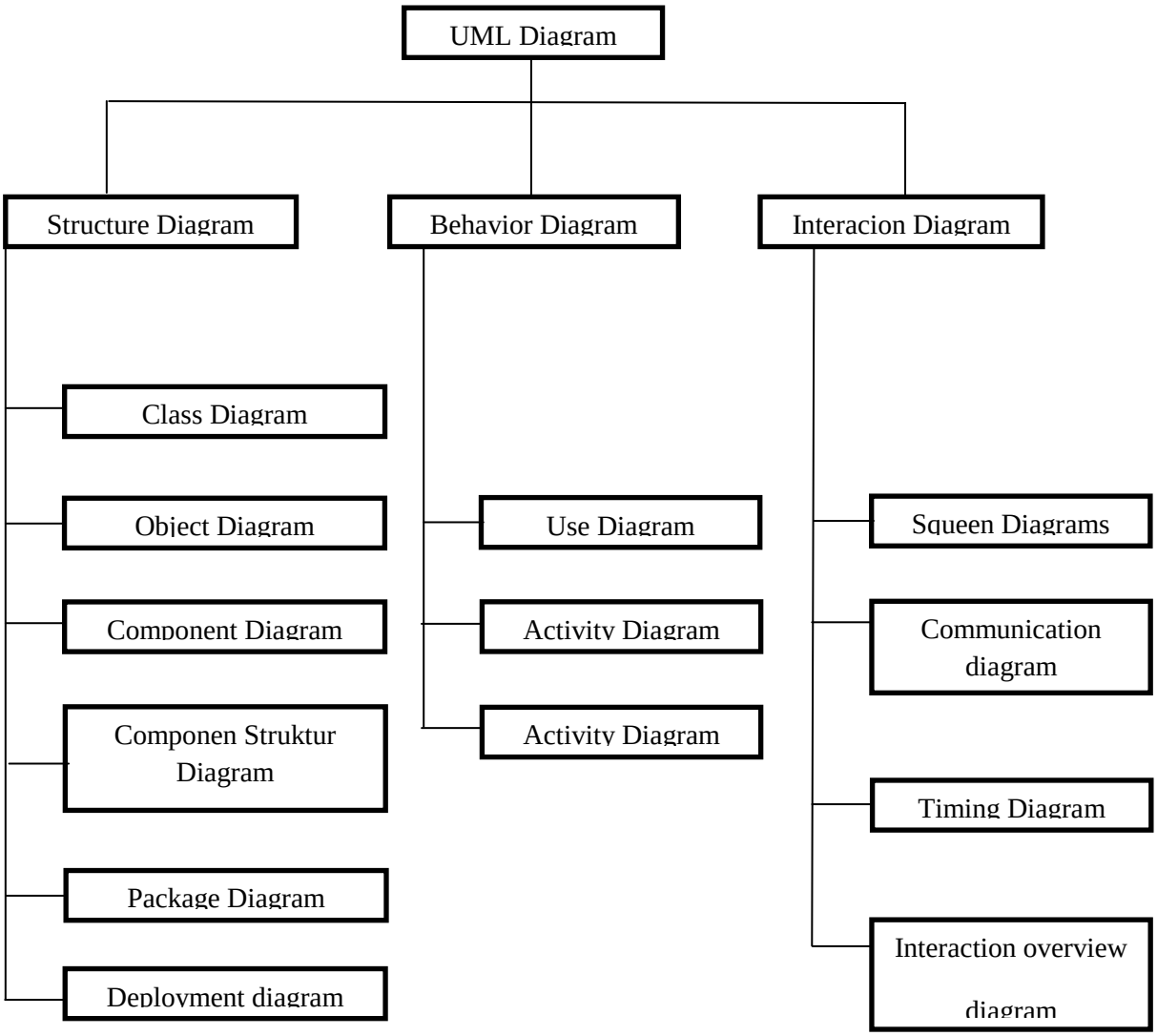
Black-box testing yaitu perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, missal untuk proses login maka kasus uji yang dibuat adalah :

1. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

2.8 Diagram UML

Diagram berbentuk grafik yang menunjukkan symbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :



Gambar 2.3 Diagram UML

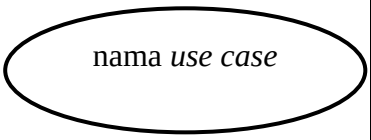
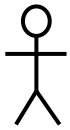
Berikut penjelasan singkat dari pembagian kategori tersebut :

1. *Structure* diagram yaitu sekumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior* diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction* diagram yaitu diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem ini maupun interaksi antar subsistem pada suatu sistem.

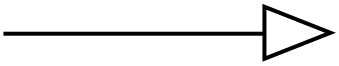
2.9 *Use Case Diagram*

"*Use Case* Diagram merupakan pemodelan untuk mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara besar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. (Triandini, E. and Suardika, I.G., 2012)

Tabel 2.6 Simbol-Simbol Use Case Diagram

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau faktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*.
Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem infromasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

Tabel 2.7 Simbol-Simbol *Use Case* Diagram

Simbol	Deskripsi
Asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi/*extend*  <<*extend*>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau 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.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang lebih umum dari lainnya.
Menggunakan/*Include*/*uses*  <<*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.

2.10 *Class Diagram*

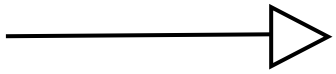
Mengatakan bahwa*Class* diagram merupakan gambaran dari struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Sususan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

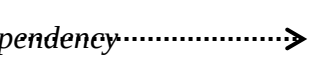
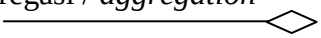
1. Kelas main, yaitu kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, yaitu kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*, yaitu kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, yaitu kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. (putranto, 2017)

Tabel 2.8. Simbol-Simbol *Class* Diagram

Simbol	Deskripsi
Kelas nama_kelas +attribut +operasi()	Kelas pada struktur sistem.
Antarmuka/*interface*  nama-interface	Sama dengan konsep interface dalam pemrograman berorientasi objek.
Asosiasi / *association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

Generalisasi 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

Tabel 2.8. Simbol-Simbol *Class Diagram* (Lanjutan)

Simbol	Deskripsi
Keberuntungan / *dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*).

2.11 *Activity* Diagram

Activity Diagram adalah diagram aktivitas yang menggambarkan sistem bukan apa yang dilakukan actor, jadi aktivitas yang dapat dilakukan oleh sistem.

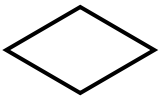
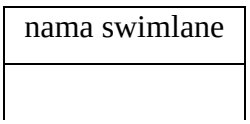
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urusan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya. (Ibnu Said, A., 2013)

Tabel 2.9. Simbol-Simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.

Tabel 2.9. Simbol-Simbol *Activity Diagram*(Lanjutan)

Simbol	Deskripsi
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.12 Bahasa Pemrograman Yang Digunakan

Komputer membutuhkan *software* untuk beroperasi dan membutuhkan sistem operasi atau program-program untuk membuat komponen-komponen komputer bekerja secara baik. Merupakan perangkat yang dapat dilihat oleh mata, tetapi tidak dapat diraba. *Software* juga sering digunakan untuk menunjukkan semua program yang dapat dipakai dalam sistem komputer. Dalam pengertian

yang sempit, istilah ini menunjuk pada sebuah program yang dapat mempermudah pemakai dari berbagai jenis komputer untuk mendayagunakan *hardware* dengan baik. Untuk merancang dan membangun aplikasi ini pembuat membutuh kan software-software penunjang untuk memaksimalkanya antara lain :

2.13 PHP (Hypertext Preprocessor)

Hypertext Preprocessor atau sering disebut PHP merupakan bahasa permrograman bebasis server-side yang dapat melakukan parsing script php menjadi menjadi script web sehingga dari sisi client menghasilkan suatu tampilan yang menarik. PHP merupkan pengembangan dari FI atau Form Interface yang dibuat oleh Rasmus Lerdoff pada tahun 1995.

Berbeda dengan HTML, kode PHP tidak diberikan ssecara langsung oleh server ketika ada permintaan atau request dari sisi client namun dengan cara pemrosesan dari sisi server. Kode PHP disisipkan pada kode HTML. Perbedaan dari kode (script) HTML dan PHP yaitu setiap kode PHP ditulis selalu diberi tag pembuka yaitu <?php dan pada ahir kode PHP diberi tag penutup yaitu ?> . Adapun kelebihan dan kekurangan bahasa PHP antara lain sebagai berikut: (Pratitis, A.N., 2013

Tabel 2.2 Kelebihan dan kekurangan PHP

No.	Kelebihan	Kekurangan
1.	PHP menjadi popular karena kesederhanaannya dan kemampuannya dalam menghasilkan berbagai aplikasi web seperti *counter*, sistem artikel/ CMS, *e-commerce*, *bulletin board*, dll.	Tidak detail untuk pengembangan skala besar.
2	PHP adalah salah satu bahasa *server-side* yang didesain khusus untuk aplikasi web.	Tidak detail untuk pengembangan skala besar.

3	PHP termasuk dalam *Open Source Product*	Tidak memiliki system pemrogaman berorientasi objek yang sesungguhnya.
4	Aplikasi PHP cukup cepat dibandingkan dengan aplikasi CGI dengan *Perl* atau *Phyton* bahkan lebih cepat dibanding dengan ASP maupun Java dalam berbagai aplikasi web.	Tidak bisa memisahkan antara tampilan dengan *logic* dengan baik.

2.14 HTML 5 (*Hypertext Markup Language 5*)

HTML5 adalah sebuah *markup* untuk menstrukturkan dan menampilkan isi dari halaman web. HTML (yang pertama kali diciptakan pada tahun 1990 dan versi keempatnya, HTML4, pada tahun 1997) dan hingga bulan juni 2011 tetap dalam proses pengembangan. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca manusia dan juga mudah dimengerti oleh mesin.

HTLM5 merupakan salah satu karya *World Wide Web Consortium, W3C* untuk mendifinisikan sebuah bahasa *markup* tunggal yang dapat ditulis dengan cara HTML ataupun XHTML. HTML5 merupakan jawaban atas pengembangan HTML 4.01 dan XHTML 1.1 yang selama ini berjalan terpisah, dan diimplementasikan secara berbeda-beda oleh banyak perangkat lunak pembuat *web.* (Sianipar, R.H., 2015)

2.15 MySQL

MySQL dikembangkan oleh sebuah perusahaaan Swedia bernama MySQL AB yang pada saat itu bernama TcX DataKonsult AB sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak 1979. Awalnya TcX membuat MySQL dengan tujuan mengembangkan aplikasi web untuk klien. TcX merupakan perusahaan pengembang *software* dan konsultan *database*. Saat ini

MySQL sudah diakusisi oleh Oracle Crop. MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang mana *database* sebagai sumber dan pengelolaan datanya.

Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses database sehingga mudah untuk digunakan. MySQL juga bersifat *open source* dan *free* pada berbagai *platform* kecuali pada *windows* yang bersifat *shareware*. MySQl didistribusikan dengan lisensi open source GPL (*General Public License*) mulai versi 3.23, pada bulan Juni 2000. (Dumairi, A., 201