BAB III

LANDASAN TEORI

3.1 Sistem Pendukung Keputusan

Sistem adalah kumpulan dari obyek-obyek seperti orang, *resources*, konsep, dan prosedur yang ditujukan untuk melakukan fungsi tertentu atau memenuhi suatu tujuan.

Pengertian sistem pendukung keputusan menurut McLeod (1998) dalam Pratiwi (2016) adalah sistem penghasil informasi yang ditujukan pada suatu masalah yang harus diselesaikan oleh manajer, sistem pendukung keputusan merupakan suatu sistem informasi yang ditujukan untuk membantu manajemen dalam memecahkan masalah yang dihadapinya.

Sistem Pendukung Keputusan (*Decision Support System)* adalah sistem berbasis komputer yang interaktif dalam membantu pengambil keputusan dengan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tak terstruktur. *Decision Support Sistem* dengan didukung oleh sebuah sistem informasi berbasis komputer dapat membantu seseorang meningkatkan kinerjanya dalam pengambilan keputusan. (Pratiwi, 2016:4)

3.1.1 Jenis-Jenis Keputusan

Jenis-jenis keputusan menurut Pratiwi (2016:5) :

a. Keputusan terstruktur yaitu keputusan-keputusan yang berkaitan dengan persoalan yang telah diketahui sebelumnya. Proses pengambilan keputusan seperti ini biasanya didasarkan atas teknik-teknik tertentu dan sudah dibuat standarnya. Kategori keputusan ini juga dapat dikatakan sebagai suatu proses jawaban secara otomatis pada kebijakan yang sudah ditentukan sebelumnya.
b. Keputusan tidak terstruktur yaitu keputusan-keputusan yang berkaitan dengan berbagai persoalan baru. Keputusan tidak

terstruktur biasanya juga berkaitan dengan persoalan yang cukup pelik, karena banyak parameter yang tidak diketahui atau belum diketahui. Oleh karena itu, untuk mengambil keputusan biasanya intuisi serta pengalaman seseorang pelaku akan sangat membantu.

c. Keputusan semi terstruktur ditandai dengan peraturan-peraturan yang tidak lengkap untuk mengambil keputusan, dan adanya kebutuhan untuk membuat penilaian serta pertimbangan subjektif sebagai pelengkap analisis data yang formal.

3.1.2 Tujuan Sistem Pendukung Keputusan

Tujuan sistem pendukung keputusan menurut Pratiwi (2016:7) :

1. Membantu membuat keputusan untuk memecahkan masalah semi terstruktur.
2. Mendukung penilaian bukan mencoba menggantikannya
3. Meningkatkan efektifitas pengambilan keputusan daripada efisiensinya.

3.1.3 Komponen Sistem Pendukung Keputusan

Menurut Pratiwi (2016:14), Untuk dapat menerapkan sistem pendukung keputusan ada empat subsistem yang harus disediakan yaitu subsistem manajemen data, subsistem manajemen model, subsistem manajemen pengetahuan dan subsistem antar muka pengguna.

a. Subsistem manajemen data

   Merupakan subsistem yang menyediakan data bagi sistem. Sumber data berasal dari data internal dan data eksternal. Subsistem ini termasuk basis data, berisi data yang relevan untuk situasi dan diatur oleh perangkat lunak yang disebut *Database Management System* (DBMS). Subsistem ini dapat diinterkoneksikan dengan data warehouse.

b. Susbsistem manajemen model

   Merupakan subsistem yang berfungsi sebagai pengelola berbagai model. Model harus bersifat flcksibel artinya mampu membantu

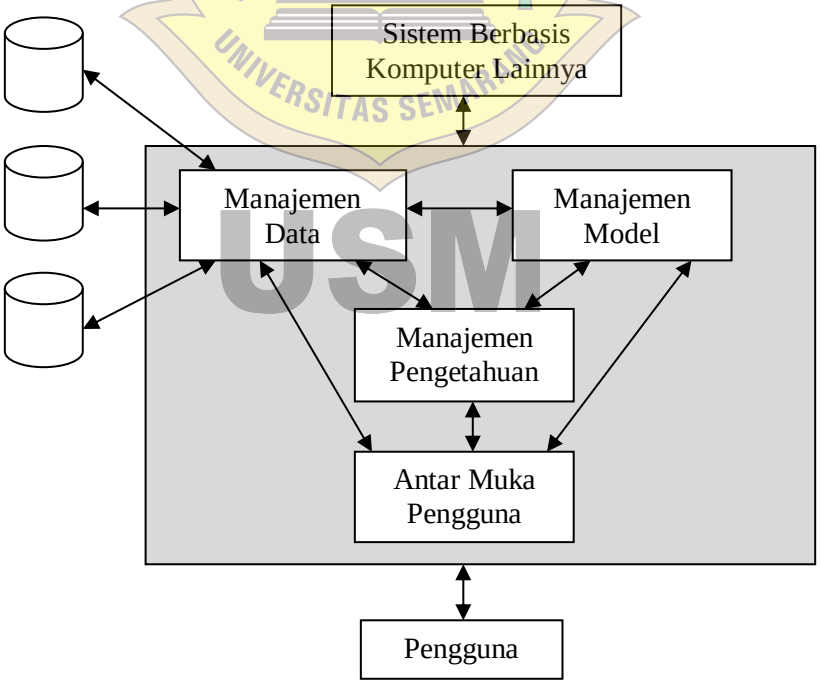
pengguna untuk memodifikasi atau menyempurnakan model, seiring dengan perkembangan pengetahuan. Perangkat lunak ini disebut *Model Base Management System* (MBMS).

c. Subsistem manajemen pengetahuan

Sebagai pendukung sembarang subsistem yang lain atau sebagai suatu komponen yang bebas. Subsistem ini berisi data item yang diproses untuk menghasilkan pemahaman, pengalaman, kumpulan pelajaran dan keahlian.

d. Subsistem antar muka pengguna

Merupakan fasilitas yang mampu mengintegrasikan sistem terpasang dengan pengguna secara interaktif. Melalui sistem dialog ini sistem diartikulasikan sehingga dapat berkomunikasi dengan sistem yang dirancang atau pengguna dapat berkomunikasi dengan sistem pendukung keputusan dan memerintah sistem pendukung keputusan melalui sistem ini



Gambar 3.1 Skema Sistem Pendukung Keputusan

Sumber : Buku Ajar Sistem Pendukung Keputusan. Pratiwi, 2016:15.

3.1.4 Tipe Sistem Pendukung Keputusan

Menurut Pratiwi (2016:23), Tipe sistem pendukung keputusan dibedakan menjadi dua macam yaitu: sistem berorientasi pada data dan sistem yang berorientasi pada model. Sistem yang berorientasi pada data adalah suatu sistem yang memberi beberapa fungsi untuk pemanggilan data, analisa dan presentasi data, sedangkan sistem pendukung keputusan yang berorientasi pada model adalah sistem pendukung keputusan yang memberi beberapa fungsi seperti model akuntansi, model simulasi, dan model optimasi yang dapat membantu manajemen dalam membuat keputusan.

3.2 Metode TOPSIS

TOPSIS adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang (1981). TOPSIS menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak Euclidean untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut.

TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan. Hal ini disebabkan konsepnya sederhana, mudah dipahami, komputasinya efisien, dan memiliki kemampuan mengukur kinerja relatif dari alternatif-alternatif keputusan. (Yoon & Hwang, 1981)

3.2.1 Langkah-langkah metode TOPSIS

Langkah-langkah yang ada di metode TOPSIS adalah :

1. Membuat matriks keputusan ternormalisasi

   Metode TOPSIS memerlukan rating kinerja tiap alternatif pada setiap criteria yang ternormalisasi. Persamaan matriks ternormalisasi dapat dilihat pada persamaan 1.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad \text{.....(1)}$$

   Dengan i = 1, 2, …, m; dan j = 1, 2, …, n.

   r_{ij} = matriks keputusan ternormalisasi.

   x_{ij} = bobot kriteria ke j pada alternatif ke i.

   i = alternatif ke i.

   j = subkriteria ke j.

2. Membuat matriks keputusan ternormalisasi terbobot

   Nilai matriks ternormalisasi terbobot dilambangkan dengan yij, dapat dihitung dengan persamaan 2.

$$y_{ij} = w_j r_{ij} \quad \text{..... (2)}$$

   Dengan i = 1, 2, …, m; dan j = 1, 2, …, n. Di mana w_j adalah bobot dari kriteria ke-j. Pemberian bobot dengan memakai hasil dari perhitungan AHP sebelumnya.

3. Menentukan matriks solusi ideal positif dan solusi ideal negative

   Berdasarkan rating bobot ternormalisasi maka dapat menentukan solusi ideal positif (A^+) dan solusi ideal negatif (A^-) Untuk dapat menentukan solusi ideal sebelumnya harus ditentukan apakah atribut bersifat keuntungan (*benefit*) atau bersifat biaya (*cost*).

$$A^+ = (y_1^+, y_2^+, \dots, y_n^+) \quad \text{..... (3)}$$

$$A^- = (y_1^-, y_2^-, \dots, y_n^-) \quad \text{..... (4)}$$

Di mana,

$$y_j^+ = \begin{cases} \max_i y_{ij} & \text{jika j adalah atribut keuntungan} \\ \min_i y_{ij} & \text{jika j adalah atribut biaya} \end{cases}$$

$$y_j^- = \begin{cases} \min_i y_{ij} & \text{jika j adalah atribut keuntungan} \\ \max_i y_{ij} & \text{jika j adalah atribut biaya} \end{cases}$$

Atribut keuntungan adalah atribut yang diberikan nilai tinggi untuk mendapatkan jarak terdekat dengan solusi ideal positif dan terjauh dengan solusi ideal negatif. Sebaliknya, atribut biaya adalah atribut yang diberikan nilai kecil untuk mendapatkan jarak terjauh dari solusi ideal positif dan terdekat dari solusi ideal negatif.

y_j^+ adalah nilai terbesar dari matriks y pada tiap kriteria ke j.

y_j^- adalah nilai terkecil dari matriks y pada tiap kriteria ke j.

4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan solusi ideal negatif. Jarak antara nilai alternatif ke i dengan solusi ideal positif dapat dirumuskan dengan persamaan 5, dan jarak antara nilai alternatif ke i dengan solusi ideal negatif dapat dirumuskan dengan persamaan 6.

$$D_i^+ = \sqrt{\sum_{j=1}^{n} (y_j^+ - y_{ij})^2} \quad \text{..... (5)}$$

$$D_i^- = \sqrt{\sum_{j=1}^{n} (y_{ij} - y_j^-)^2} \quad \text{..... (6)}$$

D_i^+ adalah jarak antara nilai alternatif ke i dengan solusi ideal positif.

D_i^- adalah jarak antara nilai alternatif ke i dengan solusi ideal negatif.

5. Menentukan nilai preferensi untuk setiap alternatif

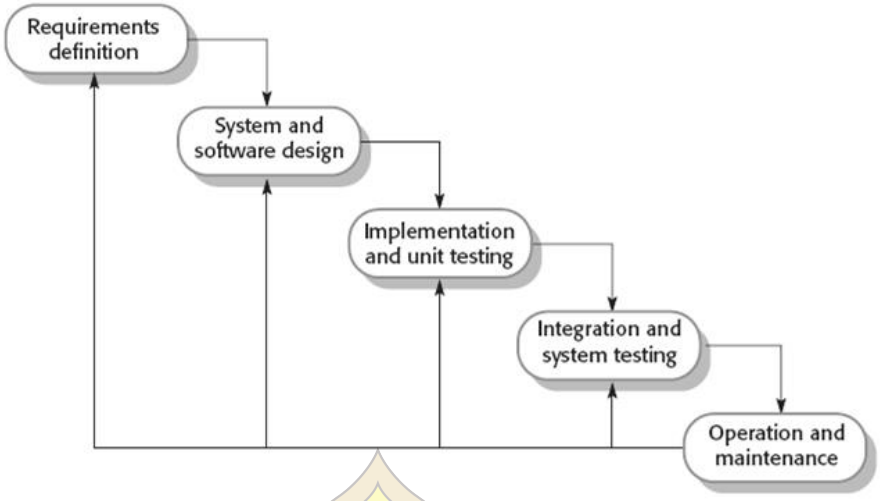
   Nilai preferensi (V_i) terbesar menunjukkan alternatif ke i lebih layak untuk dipilih sebagai solusi terbaik. Nilai V_i dapat dihitung dengan persamaan 7.

$$V_i = \frac{D_i^-}{D_i^- + D_i^+} \quad \text{..... (7)}$$

   V_i adalah nilai preferensi yang menunjukkan nilai dari alternatif ke i. Setelah didapat nilai V_i, maka alternatif akan dirangking berdasarkan urutan nilai V_i. Nilai terbesar dari V_i menunjukkan bahwa alternatif ke i adalah solusi yang paling disarankan. (Trianto, 2013)

3.3 Model Proses Air Terjun (*Waterfall*)

Waterfall menyajikan proses pengembangan perangkat lunak sebagai sejumlah tahap. Karena bentuknya tumpang tindih dari satu fase ke fase lain, model ini dikenal sebagai model air terjun atau siklus hidup perangkat lunak (*System Delevlopment Life Circle* - SDLC). Model air terjun adalah contoh dari sebuah proses yang digerakkan oleh rencana. Pada prinsipnya, paling tidak, analis sistem merencanakan dan menjadwalkan semua proses kegiatan sebelum memulai pengembangan perangkat lunak.



Gambar 3.2 Diagram proses *waterfall*

Sumber : Sommerville, 2016:47

Tahapan model air terjun mencerminkan dasar pengembangan perangkat lunak. Tahapan tersebut adalah :

1. Definisi kebutuhan

   Layanan, batasan, dan tujuan ditetapkan melalui konsultasi dengan pengguna sistem. Kebutuhan sistem kemudian didefinisikan secara detail dan berfungsi sebagai spesifikasi sistem.

2. Perancangan sistem dan perancangan perangkat lunak

   Proses perancangan sistem mengalokasikan persyaratan ke sistem perangkat keras dan sistem perangkat lunak. Perancangan sistem menetapkan keseluruhan arsitektur sistem. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran fundamental abstraksi sistem perangkat lunak dan hubungan antar mereka.

3. Implementasi dan pengujian unit

   Selama tahap ini, perancangan perangkat lunak diwujudkan menjadi sebuah program atau unit program. Pengujian unit melibatkan verifikasi setiap unit apakah memenuhi spesifikasinya.

4. Pengujian sistem dan pengujian integrasi

   Unit program terkecil program atau gabungan kode program diintegrasikan dan diuji begitu sistem telah lengkap untuk memastikan kebutuhan software (software requirement) telah terpenuhi. Setelah pengujian, software diberikan kepada konsumen atau pemilik proyek.

5. Penerapan dan perawatan

   Umumnya, penerapan dan perawatan adalah tahapan siklus yang terlama. Sistem telah terpasang dan digunakan dalam pekerjaan. Perawatan terkait perbaikan error yang tidak ditemukan ditahap awal pengembangan sistem, memperbaiki unit sistem, dan meningkatkan layanan sistem karena munculnya kebutuhan baru.

Pada prinsipnya, hasil dari setiap fase dalam model air terjun adalah satu atau lebih dokumen yang disetujui ("ditandatangani"). Fase berikutnya tidak boleh dimulai sampai fase sebelumnya telah selesai. Untuk pengembangan perangkat keras, di mana produksi biaya tinggi terlibat, penggunaan model ini masuk akal. Namun, untuk pengembangan perangkat lunak, tahapan proses metode ini tumpang tindih dan memberi informasi satu sama lain. Selama desain, masalah dengan persyaratan diidentifikasi; selama coding masalah desain ditemukan, dan seterusnya. Proses perangkat lunak, dalam praktiknya, tidak pernah merupakan model linier sederhana tetapi melibatkan umpan balik dari satu fase ke fase lainnya.

Karakteristik perangkat lunak yang baik menurut Sommerville (2016:22) :

1. Akseptabilitas

   Perangkat Lunak harus dapat diterima oleh penggunanya. Ini berarti bahwa perangkat lunak harus dapat dimengerti, digunakan, dan kompatibel dengan sistem lain yang mereka gunakan.

2. Ketergantungan dan keamanan

   Ketergantungan perangkat lunak mencakup berbagai karakteristik termasuk keandalan, keamanan, dan keamanan. Perangkat lunak yang

dapat diandalkan tidak seharusnya menyebabkan kerusakan fisik atau ekonomi jika terjadi kegagalan sistem. Perangkat lunak harus aman sehingga pengguna jahat tidak dapat mengakses atau merusak sistem.

3. Efisiensi

   Perangkat lunak tidak boleh menggunakan sumber daya sistem secara boros seperti memori dan siklus prosesor. Oleh karena itu efisiensi termasuk tanggap, waktu pemrosesan, pemanfaatan sumber daya, dll.

4. Maintainability

   Perangkat Lunak harus ditulis sedemikian rupa sehingga dapat berevolusi memenuhi kebutuhan pelanggan yang berubah. Ini adalah atribut penting karena perubahan perangkat lunak merupakan persyaratan yang tak terelakkan untuk mengubah lingkungan bisnis.

3.4 Bahasa Pemodelan Unified *Modeling Language* (UML)

Definisi UML

Menurut Adi Nugroho (2010:6), ”*UML (Unified Modeling Language)* adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek).” Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Jadi UML adalah sebuah bahasa yang berbentuk grafik atau gambar untuk menvisualisasikan obyek, menspesifikasikan, membangun dan mendokumentasikan sebuah sistem dalam proses pengembangan perangkat lunak.

3.4.1 *Use case Diagram*

Menurut Munawar (2018:89) *Use case* adalah deskripsi fungsi dari sebuah *system* dari prespektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah *system* dengan sistemnya sendiri melalui sebuah cerita bagaimana *system* dipakai. Urutan langkah- langkah yang menerangkan antara

pengguna dengan sebuah *system* yang biasa disebut dengan *scenario*. Setiap *scenario* mendeskrepsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, system yang lain, perangkat keras atau urutan waktu. Dengan demikian *use case* adalah serangkaian skenario yang digabungkan bersama-sama oleh tujuan umum pengguna.

Tujuan dari *use case diagram* menurut Munawar (2018:90) adalah sebagai berikut :

1. Mengumpulkan kebutuhan dari sebuah sistem.
2. Mendapat pandangan dari luar sistem.
3. Mengidentifikasi faktor yang mempengaruhi sistem baik internal maupun eksternal.
4. Menunjukan interaksi dari para *actor* dari sistem.

Adapun simbol-simbol yang digunakan dalam *use case diagram* menurut Munawar (2018) adalah sebagai berikut:

1. *Actor*

   Aktor adalah *abstraction* dari orang dan *system* yang lain mengaktifkan fungsi dari target *system*. Orang atau *system* bisa muncul dalam beberapa peran.

2. *Use case*

   Use case adalah abstraksi dari interaksi antara *system* dengan *actor*. *Use case* dibuat berdasarkan keperluan *actor*. *Usecase* harus merupakan ‘apa’ yang dikerjakan software aplikasi, bukan ‘bagaimana’ software aplikasi mengerjakannya. Nama *usecase* boleh terdiri dari beberapa kata dan tidak boleh ada dua *usecase* yang memiliki nama yang sama. *Usecase* digambarkan dengan ellips (dengan nama didalamnya).

3. *Association*

   Mengidentifikasikan interaksi antara actor tertentu dengan *use case* tertentu. Digambarkan dengan garis antara actor terhadap *use case*. Asosiasi bisa berarah (dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian.

4. *Stereotype*

   Memungkinkan perluasan *UML* tanpa memodifikasinya. Menyediakan informasi lebih banyak mengenai peranan dari elemen tanpa menyebutkan implementasinya.

5. *Dependency*

   <<include>> mengidentifikasi hubungan antar dua use-case dimana salah satu memanggil yang lain.

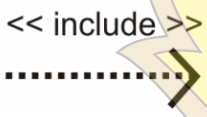
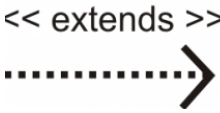
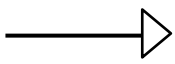
   <<extend>> jika pemanggilan memerlukan adanya kondisi tertentu.

6. *Generalization*

   Mengidentifikasi selasi antara dua actor atau dua *use case*. Salah satunya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya.

Tabel 3.1 Simbol *Use case diagram*

(Sumber : Analisis Perancangan Sistem Berorientasi Objek Dengan UML. Munawar, 2018.)

SIMBOL	NAMA	Keterangan
	*Actor*	Seseorang atau apa saja yang berhubungan dengan sistem yang sedang dibangun.
	*Use case*	Menggambarkan bagaimana seseorang menggunakan system.
	Relasi asosiasi	Relasi yang dipakai untuk menunjukkan hubungan antara *actor* dan *use case*.
	*Relasi include*	Memungkinkan satu use case menggunakan fungsionalitas yang disediakan oleh usecase lainnya.
	*Relasi extend*	Memungkinkan suatu use case secara optional menggunakan fungsionalitas yang disediakan oleh usecase lainnya.
	*Generalisasi*	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)

3.4.2 Class Diagram

Class Diagram adalah diagram statis Munawar(2018:101). *Class diagram* tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan berbagai aspek sistem tetapi juga untuk membangun kode eksekusi dari aplikasi perangkat lunak.

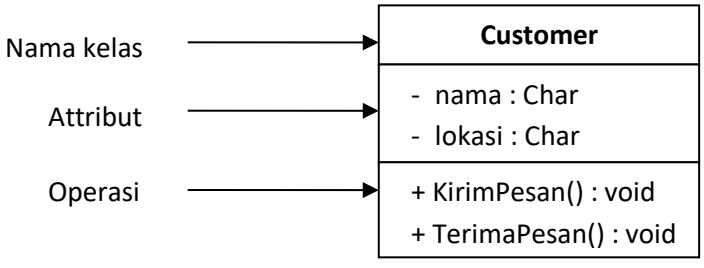
Class diagram menggambarkan atribut, operation dan juga constrain yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem objek oriented karena mereka adalah satu-satunya diagram *UML*, yang dapat dipetakan langdung dengan bahasa berorientasi objek.

Class diagram menunjukan koleksi *Class*, antar muka, asosiasi, kolaborasi, dan constraint. *Class* diagram juga dikenal sebagai diagram struktural.

Tujuan dari class diagram menurut Munawar (2018:102) adalah sebagai berikut :

1. Analisis dan desain pandangan statis aplikasi.
2. Menjelaskan tanggung jawab suatu sistem.
3. Basis untuk diagram komponen dan penyebaran (*deployment*).
4. *Forward and reverse enginering*.

Adapun contoh dari class diagram adalah sebagai berikut :



Gambar 3.3 Contoh *Class Diagram*

(Sumber : Analisis dan Perancangan Sistem Berorientasi Objek dengan *UML*. Munawar, 2018.)

Nilai kardinalitas

Nilai kardinalitas atau *multiplicity* menunjukkan jumlah suatu objek yang dapat berhubungan dengan objek lainnya. Macam jenis *multiplicity* ditunjukkan dalam Tabel berikut ini :

Tabel 3.2 Jenis-jenis *multiplicity*
(Sumber : Analisis dan Perancangan Sistem Berorientasi Objek dengan UML. Munawar, 2018.)

Indikator	Arti
0..1	Nol atau satu
1	Hanya satu
0..*	Nola atau lebih
1..*	Satu atau lebih
N	Hanya n (dengan n>1)
0..n	Nol sampai n (dengan n>1)
1..n	Satu sampai n (dengan n>1)

Object diagram

Object diagram adalah gambaran obyek-obyek secara ringkas di sebuah sistem pada suatu waktu. *Object diagram* sering disebut instance diagram karena menunjukan *instance-instance* dari *class*. *Object diagram* bisa digunakan untuk menunjukan contoh konfigurasi dari obyek-obyek. Hal ini sangat berguna untuk menunjukan hubungan yang mungkin ada di antara obyek-obyek yang sangat komplek.

Object diagram tidak hanya digunakan untuk menvisualisasi, spesifikasi dan dokumentasi model struktual, akan tetapi juga penting untuk pembangunan aspek statis dari suatu sistem melalui *forward enginering*.

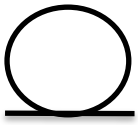
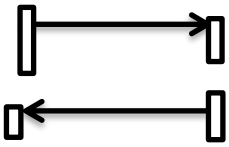
3.4.3 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan message (pesan) yang diletakkan diantara obyek-obyek ini didalam *usecase*. Munawar (2018:137)

Dalam squence diagram penggunaan class sangatlah dibutuhkan. Hanya saja stereotype class yang dipakai meliputi boundary, control dan entity.Stereotype ini terkait dengan perubahan sebuah model meski hanya pada area kerja tertentu. Secara khusus stereotype ini berguna dalam identifikasi class pada saat analisis dan desain awal. Lebih detailnya akan diuraikan sebagai berikut :

Tabel 3.3 Simbol *Squence Diagram*
(Sumber : Analisis dan Perancangan Sistem Berorientasi Objek dengan UML. Munawar, 2018.)

SIMBOL	NAMA	KETERANGAN
	*Actor*	Orang ataupun pihak yang akan mengelola system.
	*Boundary*	Pemodelan bagian dari sistem yang bergantung pada pihak lain disekitarnya dan merupakan pembatas sistem dengan dunia luar.
	*Control*	Permodelan“perilaku mengatur” khusus untuk satu atau beberapa use case saja.

	*Entity*	Permodelan informasi yang harus disimpan oleh sistem yang memperlihatkan struktur data dari suatu sistem.
	*Message*	Mengindifikasikan urutan komunikasi yang terjadi antar objek.
	*Selfstimulus*	Menyatakan suatu objek mengirimkan pesan untuk menjalankan oprasi yang ada pada objek lain.

Untuk menunjukan looping dengan satu *fragment* dan letakkan iterasi dasar di guard. Untuk logika bersyarat gunakan operator Alt dan diletakkan di setiap syarat pada setiap *fragment*. Hanya fragment yang memiliki guard *True* yang akan dijalankan. Pada kasus di atas, region hanya ada satu yaitu operator Opt. *Interaction frame* ini adalah hal baru yang ada di *UML* 2. Pada *UML* 1, digunakan *iteration marker* adalah sebuah yang ditambahkan pada nama message. Beberapa teks bisa ditambahkan dalam kurang kotak untuk menunjukan basis iterasi. Guard adalah ekspresi bersyarat yang ditempatkan dalam kurung kotak untuk menunjukan bahwa message tersebut hanya akan dikirim jika kondisinya *True*. Adapun operator operator yang umum digunakan dalam *interaction frame* adalah sebagai berikut :

Tabel 3.4 Simbol Operator *Squence Diagram*
(Sumber : Analisis dan Perancangan Sistem Berorientasi Objek dengan UML. Munawar, 2018.)

Operator	Keterangan
Alt	Alternatif dari banyak fragmen. Hanya kondisinya True yang akan dijalankan
Opt	*Optional fragmen;* akan dijalankan jika kondisi yang mendukungnya *True*
Par	*Paralel*; setiap *fragmen* dijalankan secara *paralel*
Loop	*Looping*; fragmen mungkin dijalankan berulang kali dan guard menunjukan basis iterasi
Region	Critical region; fragmen hanya dapat mempunyai satu thread untuk menjalankannya
Neg	Negative; fragmen menunjukan interaction yang salah
Ref	Reference; menunjukan ke sebuah iteraction yang didefinisikan pada diagram yang lain
Sd	Squence diagram

3.4.4 Activity diagram

Menurut mulyani (2016:55) *Activity diagram* yaitu diagram yang digunakan untuk menggambarkan alur kerja (aktifitas) pada *use case* (proses), logika, proses bisnis dan hubungan antara faktor dengan alur-alur kerja *use case*.

Adapun simbol-simbol dari activity diagram menurut Mulyani (2016:55) adalah sebagai berikut:

1. Start point

   Sebuah simbol untuk memulai suatu activity diagram.
2. End point

   Sebuah simbol yang digunakan untuk mengakhiri activity diagram
3. Activitiies

   Sebuah proses komputasi yang bisa berupa kata kerja atau ekspresi dan bersifat tidak dapat didekomposisi
3. Decision

   Digunakan sebagai pilihan aktifitas untuk mengambil keputusan
4. Iteraction

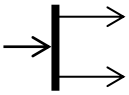
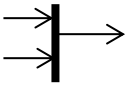
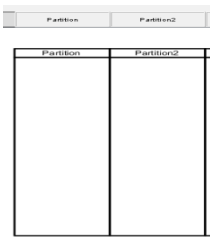
   Menunjukan sebuah alur aktifitas dari sebuah sistem
5. Fork

   Menunjukan sebuah percabangan paralel dari sebuah aktifitas sistem
6. Join

   Berfungsi untuk menggabungkan beberapa aktifitas
7. Swimlines yaitu elemen yang digunakan untuk memisahkan antara aktor dan sistem

   Berfungsi untuk menggabungkan beberapa aktifitas

Tabel 3.5 Simbol-simbol dalam Activity Diagram
(Sumber : Analisis Perancangan Sistem Berorientasi Objek Dengan UML. Mulyani, 2016:55)

SIMBOL	NAMA	KETERANGAN
	*Start point*	Titik awal atau permulaan
	*End point*	Titik akhir atau akhir dari aktivitas
	*Activity*	*Activity* atau aktivitas yang dilakukan oleh *actor*
	*Decision*	Pilihan untuk mengambil keputusan
	Interaction	Alur
	*Fork*	Menunjukan adanya percabangan secara paralel dari aktivitas
	*Join*	Menunjukan adanya penggabungan aktivitas
	*Swimlines*	Untuk mimisahkan antara aktor dengan sistem

3.5 PHP (Hypertext Preprocessor)

PHP sering dipakai para programmer untuk membuat situs web yang bersifat dinamis karena gratis dan berguna dalam merancang aplikasi web. Supono dan Virdiandry P. (2016:3) mengemukakan bahwa ”*PHP* (*PHP: Hypertext Preprocessor*) adalah suatu bahasa pemrograman yang digunakan untuk menerjemahkan baris kode program menjadi kode mesin yang dapat dimengerti oleh komputer yang berbasis server-side yang dapat ditambahkan ke dalam *HTML*”. Sedangkan, menurut Solichin (2016:11) mengemukakan bahwa “PHP merupakan salah satu bahasa pemrograman berbasis web yang ditulis oleh dan untuk pengembang web”.

3.6 MYSQL

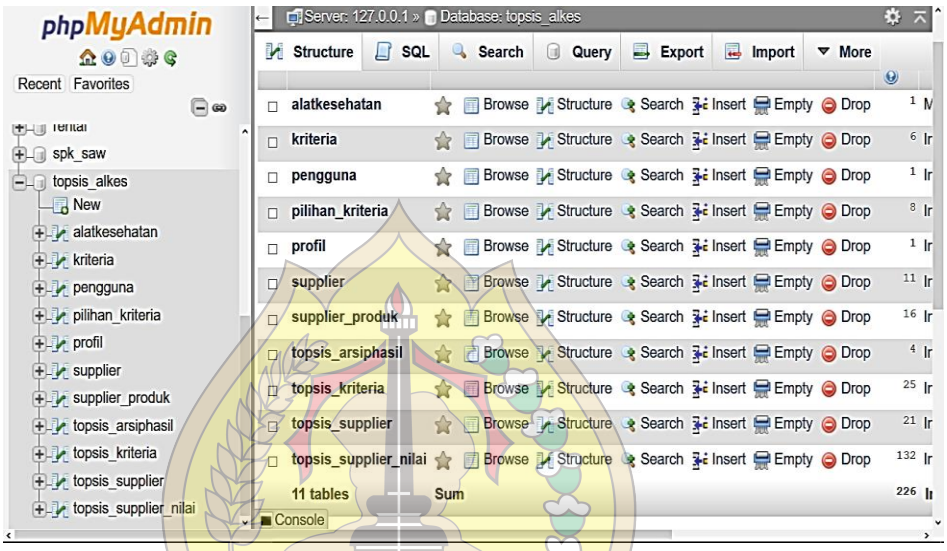
Menurut Arief (2011:152) “*MySQL* adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengolahan datanya”. *MySQL* dikembangkan oleh perusahaan swedia bernama *MySQL AB* yang pada saat ini bernama *Tcx DataKonsult AB* sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak tahun 1979. Awalnya Tcx merupakan perusahaan pengembang software dan konsultan database, dan saat ini *MySQL* sudah diambil alih oleh *Oracle Corp*.

Kepopuleran MySQL antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja query cepat, dan mencukupi untuk kebutuhan database perusahaan-perusahaan yang berskala kecil sampai menengah, *MySQL* juga bersifat open source (tidak berbayar).

MySQL merupakan database yang pertama kali didukung oleh bahasa pemrograman script untuk internet (*PHP dan Perl*). *MySQL* dan *PHP* dianggap sebagai pasangan software pembangun aplikasi web yang ideal. *MySQL* lebih sering digunakan untuk membangun aplikasi berbasis

web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman script *PHP*.(Arief, M.Rudianto. 2011. Pemrograman Web Dinamis Menggunakan Php dan Mysql. Yogyakarta: ANDI.)

Adapun tampilan dari MySQL adalah sebagai berikut



3.7 Sublime Text 3

Sublime Text adalah aplikasi editor untuk kode dan teks yang dapat berjalan diberbagai platform operating system dengan menggunakan teknologi *Phyton API*. Terciptanya aplikasi ini terinspirasi dari aplikasi Vim, Aplikasi ini sangatlah fleksibel dan powerfull. Fungsionalitas dari aplikasi ini dapat dikembangkan dengan menggunakan sublime-packages. *Sublime Text* bukanlah aplikasi opensource dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, akan tetapi beberapa fitur pengembangan fungsionalitas (packages) dari aplikasi ini merupakan hasil dari temuan dan mendapat dukungan penuh dari komunitas serta memiliki linsensi aplikasi gratis.

Sublime Text mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur syntax highlight hampir di semua bahasa pemrogramman yang didukung ataupun dikembangkan oleh komunitas seperti*; C, C++, C#, CSS, D, Dylan, Erlang, HTML,*

Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, *Textile and XML*. Biasanya bagi bahasa pemrograman yang didukung ataupun belum terdukung secara default dapat lebih dimaksimalkan atau didukung dengan menggunakan add-ons yang bisa didownload sesuai kebutuhan user.

Berikut beberapa fitur yang diunggulkan dari aplikasi Sublime Text:

a. *Goto Anything*

Fitur yang sangat membantu dalam membuka file ataupun menjelajahi isi dari file hanya dengan beberapa keystrokes.

b. *Multiple Selections*

Fitur ini memungkinkan user untuk mengubah secara interaktif banyak baris sekaligus, mengubah nama variabel dengan mudah, dan memanipulasi file lebih cepat dari sebelumnya.

c. *Command Pallete*

Dengan hanya beberapa keystorkes, user dapat dengan cepat mencari fungsi yang diinginkan, tanpa harus menavigasi melalu menu.

d. *Distraction Free Mode*

Bila user memerlukan fokus penuh pada aplikasi ini, fitur ini dapat membantu user dengan memberikan tampilan layar penuh.

e. *Split Editing*

Dapatkan hasil yang maksimal dari monitor layar lebar dengan dukungan editing perpecahan. Mengedit sisi file dengan sisi, atau mengedit dua lokasi di satu file. Anda dapat mengedit dengan banyak baris dan kolom yang user inginkan.

f. *Instant Project Switch*

Menangkap semua file yang dimasukkan kedalam project pada aplikasi ini. Terintegrasi dengan fitur *Goto Anything* untuk

menjelajahi semua file yang ada ataupun untuk beralih ke file dalam project lainnya dengan cepat.

g. *Plugin API*

Dilengkapi dengan *plugin API* berbasis *Phyton* sehingga membuat aplikasi ini sangat tangguh.

h. *Customize Anything*

Aplikasi ini memberikan user fleksibilitas dalam hal pengaturan fungsional dalam aplkasi ini.

i. *Cross Platform*

Aplikasi ini dapat berjalan hampir disemua operating system modern seperti *Windows, OS X,* dan *Linux based operating system*.

3.8 Pengujian *Black box*

Menurut Shalahuddin dan Rosa (2011) *blackbox testing* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian yang dimaksud untuk mengetahui apakah sebuah fungsi, masukan, dan keluaran sesuai dengan spesifikasi yang dibutuhkan. Pengujian ini bersifat mencoba semua fungsi dengan memakai perangkat lunak yang telah dibangun apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box testing* harus dibuat dengan kasus benar maupun kasus salah.

Pengujian Black Box berusaha menemukan kesalahan dalam kategori :

1. Fungsi - fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Sehingga dalam uji coba Blackbox harus melewati beberapa proses sebagai berikut:

1. Menganalisis kebutuhan dan spesifikasi dari perangkat lunak.
2. Pemilihan jenis input yang memungkinkan menghasilkan output benar serta jenis input yang memungkinkan output salah pada perangkat lunak yang sedang diuji.
3. Menentukan output untuk suatu jenis input.
4. Pengujian dilakukan dengan input - input yang telah benar - benar diseleksi.
5. Melakukan pengujian.
6. Pembandingan output yang dihasilkan dengan output yang diharapkan

3.9 Pengujian *White box*

Menurut Pressman (2010), pengujian *White-box* atau *Glass-box* adalah metode *test-case* desain yang menggunakan struktur kontrol desain *procedural* untuk memperoleh *test-case*. Dengan menggunakan metode pengujian *white-box*, perekayasa *system* dapat memperoleh *test-case* yang:

1. Memberikankan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
2. Menggunakan semua keputusan logis dari sisi *true* dan *false.*
3. Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
4. Menggunakan struktur *internal* untuk menjamin validitasnya.

3.9.1 Uji Coba Basis *Path*

Uji coba basis *path* adalah teknik uji coba *white box* yang diusulkan Tom Mc Cabe. Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunkan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case*

yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba. Terdapat beberapa proses yang harus di lakukan dalam uji coba basis *path* yaitu diantaranya:

1. Notasi Diagram Alir

   Sebelum metode basis path diperkenalkan, terlebih dahulu akan dijelaskan mengenai notasi sederhana dalam bentuk diagram alir (grafik alir). Diagram alir menggambarkan aliran kontrol logika yang menggunakan notasi.

2. Kompleksitas Siklomatis

   Kompleksitas siklomatis adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian basis path, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu pemrograman memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang memlalui progarm yang mengintroduksi sedikitnya satu rangkaian statemen proses baru atau suatu kondisi baru. Bila dinyatakan dengan terminologi grafik alir, jalur independen harus bergerak sepanjang paling tidak satu edge yang tidak dilewatkan sebelum jalur tersebut ditentukan.

3. Melakukan Test Case

   Metode uji coba basis *path* juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah - langkah uji coba basis *path*.

4. Matriks Grafis

   Prosedur untuk mendaptakan grafik alir dan menentukan serangkaian basis *path*, cocok dengan mekanisasi. Untuk mengembangkan peranti perangkat lunak yang membantu pengujian

basis *path*, struktur data yang disebut matriks grafis dapat sangat berguna. Matriks garfis adalah matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir. Masing - masing baris dan kolom sesuai dengan simpul yang diidentifikasikan dan *entry matriks* sesuai dengan *edge* diantara simpul.