



# **BAB II**

# **DASAR TEORI**



## BAB II DASAR TEORI

### 2.1 Sistem Pendukung Keputusan

Sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah dengan kondisi terstruktur disebut dengan Sistem Pendukung Keputusan (SPK)[5]. Sistem ini berfungsi untuk membantu pengambilan keputusan dalam situasi semi terstruktur dan tidak terstruktur. Dan berikut ini pengertian sistem pendukung keputusan menurut para ahli antara lain:

Sistem pendukung keputusan menurut Nofriansyah dan Defit adalah suatu sistem informasi spesifik yang ditunjukkan untuk membantu manajemen dalam mengambil keputusan yang berkaitan dengan persoalan yang bersifat semi terstruktur. Sedangkan menurut Little dalam Buku Febrian Sari Sistem Pendukung Keputusan merupakan informasi berbasis komputer yang menghasilkan macam pilihan keputusan untuk membantu manajemen dalam menangani kendala yang terstruktur maupun tidak terstruktur dengan menggunakan data dan sistem.

SPK digunakan karena bersifat Objektivitas yaitu dapat memberikan penilaian yang objektif, selain itu juga dalam penggunaan SPK sendiri bersifat konsisten dan terstandarisasi selain itu efisiensi waktu dan panitia yang menyeleksi. Dapat disimpulkan bahwa sistem pendukung keputusan merupakan sistem informasi yang dirancang untuk memberikan solusi manajemen level menengah dalam mengambil keputusan dengan bentuk setengah terstruktur atau semi terstruktur menggunakan pemodelan sistem. Sistem pendukung keputusan memiliki keterkaitan dengan metode perhitungan yang membantu pengambilan keputusan yang tepat dan efektif salah satunya adalah metode *Technique for Order Performance of Similarity to Ideal Solution (TOPSIS)* merupakan salah satu sistem pendukung keputusan multikriteria. *TOPSIS* mempunyai prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi *ideal* positif dan mempunyai jarak terjauh dari solusi *ideal* negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* (jarak antara dua titik) untuk menentukan kedekatan relative dari suatu alternatif[6].

Penyelesaian metode *TOPSIS* sebagai berikut:

- a. Membuat matriks keputusan yang ternormalisasi.

$$rij = \frac{xij}{\sqrt{\sum_{i=1}^m x_{ij}^2}}$$

Dengan  $i = 1, 2, \dots, m$ ; dan  $j = 1, 2, \dots, n$ .

$rij$  = matriks keputusan ternormalisasi

$xij$  = bobot kriteria ke  $j$  pada alternatif ke  $i$

$i$  = alternatif ke  $i$

$j$  = subkriteria ke  $j$

- b. Membuat matriks keputusan yang ternormalisasi terbobot

$$Y_{ij} = W_{ij}$$

Dengan  $i = 1, 2, \dots, m$ ; dan  $j = 1, 2, \dots, n$ .

Dimana  $w_j$  adalah bobot dari kriteria ke- $j$ .

Menentukan matriks solusi *ideal* positif dan matriks solusi *ideal* negatif.

$$A^+ = (y_1^+, y_2^+, \dots, y_n^+)$$

$$A^- = (y_1^-, y_2^-, \dots, y_n^-)$$

- c. Menghitung jarak antar nilai alternatif ke solusi *ideal* positif dan solusi *ideal negative* dengan rumus:

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

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

Dimana:

$D_i^+$  = adalah jarak alternatif ke-I dengan solusi *ideal* positif.

$y_i^+$  = adalah elemen solusi *ideal* positif [i].

$y_{ij}$  = adalah elemen matriks ternormalisasi terbobot [i][j].

$D_i^-$  = adalah jarak alternatif ke-I dengan solusi *ideal negative*.

$y_j^-$  = adalah elemen solusi *ideal negative* [i].

- d. Menghitung nilai preferensi tiap alternatif.

$$V_i = \frac{D_i^-}{D_i^- - D_i^+}$$

Dimana:

$V_i$  = adalah kedekatan tiap alternatif terhadap solusi *ideal*.

$D_i^+$  = adalah jarak alternatif ke-I dengan solusi *ideal* positif.

$D_i^-$  = adalah jarak alternatif ke-I dengan solusi *ideal negative*[7].

## 2.2 Rekayasa Perangkat Lunak

Perangkat lunak (*software*) merupakan program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user manual*). Sedangkan rekayasa perangkat lunak (*software engineering*) merupakan pembangunan dengan menggunakan prinsip rekayasa yang bertujuan untuk

menghasilkan perangkat lunak sehingga memiliki nilai ekonomi yang dipercaya dan dapat bekerja secara efisien menggunakan mesin[8].

Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria sebagai berikut:

- a. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring perkembangan teknologi dan lingkungan.
- b. Dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi.
- c. Efisien dari segi sumber daya dan penggunaan.
- d. Kemampuan untuk dipakai sesuai dengan kebutuhan.

Untuk mengembangkan rekayasa perangkat lunak maka dibutuhkan metode pengembangan sistem, metode pengujian sistem, dan *tools* (alat bantu).

Metode pengembangan sistem yang digunakan pada pengembangan sistem seleksi anggota osis yaitu Model *waterfall* adalah salah satu model *SDLC* yang sering digunakan dalam pengembangan sistem informasi atau perangkat lunak. *Model waterfall* menggunakan pendekatan sistematis dan berurutan[4].

### 2.3 Pemrograman Berbasis Objek (PBO)

Pemrograman Berorientasi Objek merupakan sebuah istilah yang diberikan kepada bahasa pemrograman yang menggunakan 2 teknik berorientasi atau berbasis pada sebuah objek dalam pembangunan program aplikasi tersebut. Maksudnya bahwa orientasi pembuatan program tidak lagi menggunakan orientasi linear melainkan berorientasi pada objek yang terpisah. Suatu perintah dalam bahasa ini diwakili oleh sebuah objek yang didalamnya berisi beberapa perintah-perintah standar sederhana[9].

*Unified Modeling Language* (UML) adalah bahasa visual untuk komunikasi dan pemodelan mengenai sebuah sistem menggunakan diagram dan teks-teks pendukung. Penggunaan UML tidak terbatas pada metodologi tertentu karena UML hanya berfungsi untuk melakukan pemodelan. Chonoles (2003) mengatakan bahwa UML sebagai bahasa memiliki arti sintaks dan semantik. Ketika membuat model dengan UML, kita juga tidak sembarangan dalam menggambarkan diagram-diagramnya. Karena di dalam diagram tersebut juga menceritakan konteks dari sistem yang kita buat.

Ada beberapa diagram dalam UML yaitu:

#### a) *Use Case Diagram*

Diagram *use case* merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. *Use case* mendeskripsikan interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Penamaan pada *use case* didefinisikan sesederhana mungkin. Terdapat dua hal utama pada *use case* yaitu pendefinisian aktor dan *use case*.








##### 1) Aktor

Aktor adalah orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Jadi, meskipun simbol dari aktor adalah gambar orang, tetapi aktor tersebut belum tentu merupakan orang.

2) *Use Case Diagram*

*Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit yang saling bertukar pesan antar satu unit atau aktor dengan unit atau aktor lain[10]. Simbol-simbol pada diagram *use case* dapat dilihat pada Table 2.1





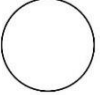
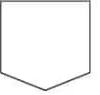

Tabel 2. 1 Simbol *Use case diagram*



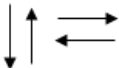



No	Nama	Gambar	Deskripsi
1	<i>Use case</i>		<i>Use case</i> merupakan fungsionalitas yang disediakan sistem sebagai unit yang saling bertukar pesan antar satu unit atau aktor dengan unit atau aktor lain. <i>Use case</i> biasanya dinyatakan menggunakan kata kerja di awal frase nama <i>use case</i> .
2	Aktor / <i>Actor</i>		Aktor adalah orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Jadi, meskipun simbol dari aktor adalah gambar orang, tetapi aktor tersebut belum tentu merupakan orang.
3	<i>Asosiasi</i>		<i>Asosiasi</i> antara aktor dan <i>use case</i> digambarkan dengan garis tanpa panah yang dapat diartikan sebagai siapa atau apa yang meminta interaksi secara langsung.
4	<i>Generalization</i>		Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
5	<i>Include</i>		<i>Include</i> , terdapat di dalam <i>use case</i> lain atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi sistem.
6	<i>Extend</i>		<i>Extend</i> , merupakan perluasan dari <i>use case</i> lain apabila kondisi atau syarat terpenuhi.
7	<i>Boundary</i>		<i>Boundary</i> , merupakan batasan dari sebuah sistem.

**b) Flowchart**

*FlowChart* berfungsi untuk menggambarkan alur kerja sebuah sistem atau apa yang sedang dikerjakan oleh sebuah sistem secara keseluruhan[11]. *Flowchart* digambarkan dengan simbol-simbol yang mewakili proses tertentu, simbol tersebut kemudian dihubungkan menggunakan garis penghubung untuk membentuk suatu proses atau langkah-langkah sistem. *Flowchart* menjelaskan langkah-langkah yang harus dilakukan dalam sebuah proses, beserta kondisi-kondisi yang harus dipenuhi dan keputusan-keputusan yang harus diambil. Simbol-simbol pada *flowchart* dapat dilihat pada Tabel 2.2.

**Tabel 2. 2** Simbol *Flowchart*

No	Nama	Gambar	Deskripsi
1.	Terminal		Terminal menyatakan permulaan atau akhir dari suatu program.
2.	<i>Input / Output</i>		<i>Input / Output</i> menyatakan proses <i>input</i> dan <i>output</i> tidak tergantung dengan jenis peralatannya.
3.	<i>Process</i>		<i>Process</i> menyatakan suatu proses yang dilakukan oleh komputer.
4.	<i>Decision</i>		<i>Decision</i> menyatakan suatu kondisi tertentu yang menghasilkan dua kemungkinan jawabannya: ya / tidak.
5.	<i>Connector</i>		<i>Connector</i> merupakan sambungan dari proses ke proses lain dalam halaman yang sama.
6.	<i>Offline Connector</i>		<i>Offline Connector</i> merupakan sambungan dari proses ke proses lain dalam halaman yang berbeda.
7.	<i>Predefined Process</i>		<i>Predefined Process</i> merupakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan harga awal.

8.	<i>Punched Card</i>		<i>Punched Card</i> dapat dikatakan input yang berasal dari kartu atau <i>output</i> yang ditulis ke kartu.
9.	<i>Document</i>		<i>Document</i> berfungsi untuk mencetak jalannya arus suatu proses.
10.	<i>Flow</i>		<i>Flow</i> dapat menggambarkan jalannya suatu proses.
11.	<i>Manual Input</i>		Memasukan data secara manual dengan menggunakan online keyboard.
12.	<i>Manual operation</i>		Menggambarkan kegiatan yang dilakukan tanpa menggunakan perangkat komputer.
13	<i>Multiple document</i>		Serupa dengan simbol dokumen, tetapi dokumen yang digunakan dapat digunakan oleh lebih dari satu orang.

### 2.3 Basis Data

Basis data (*database*) adalah penampung sekumpulan data yang saling terkait satu sama lain, sehingga memudahkan pengelolaan data menjadi informasi yang bermanfaat. Dalam jurnal lain basis data adalah tempat pengelolaan data yang ada pada sistem[12].

Basis Data atau *Database* adalah sistem terkomputerisasi yang memiliki tujuan utamanya untuk memelihara data yang sudah diolah atau informasi, serta membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Database Management System (DBMS)*.

*Database Management System (DBMS)* atau dalam bahasa Indonesia sering disebut sebagai Sistem Manajemen Basis Data merupakan suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut:

1. Menyediakan fasilitas untuk mengelola akses data.
2. Mampu menangani integritas data.



3. Mampu menangani *backup* data.

DBMS dibagi menjadi dua versi, yaitu versi komersial dan versi *open source*. Untuk versi komersial yang paling banyak digunakan di dunia saat ini adalah:

1. *Oracle*.
2. *Microsoft SQL Server*.
3. *IBM DB2*.
4. *Microsoft Access*.

Sedangkan DBMS versi *opensource* yang paling banyak digunakan saat ini adalah:

1. *MySQL*.
2. *PostgresSQL*.
3. *Firebird*.
4. *SQLite*.

Selain penggunaan bahasa SQL, pada DBMS juga dibutuhkan pemodelan awal untuk mempermudah pembuatan *database*. *Database Management System* (DBMS) atau merupakan sebuah perangkat lunak (*software*) yang memungkinkan pengguna untuk mendefinisikan, membuat, mengambil data dan mengontrol akses kepada *database*. DBMS adalah sebuah perangkat lunak (*software*) yang mengintegrasikan antara *database* dengan aplikasi program pada pengguna. DBMS menyediakan berbagai fasilitas seperti[9]:

#### 1. *Data Definition Language* (DDL)

DDL merupakan perintah-perintah yang digunakan sebagai pengoperasian skema struktur pada *database*, DDL memperbolehkan pengguna untuk menjelaskan *database*, misalnya merincikan tipe dan batasan data yang akan disimpan dalam *database*[13]. Perintah yang termasuk dalam DDL yaitu:

##### a. *Create*

*Create* adalah perintah untuk membuat termasuk didalamnya untuk membuat *database* dan *table*.

##### b. *Alter*

*Alter* adalah perintah yang berfungsi untuk merubah struktur tabel yang sebelumnya telah dibuat. Hal tersebut mencakup mengubah nama tabel, menambah kolom, menghapus kolom, mengubah kolom, dan memberi atribut pada kolom.

##### c. *Drop*

*Drop* adalah perintah untuk menghapus *database* maupun tabel.

#### 2. *Data Manipulation Language* (DML)

DML adalah perintah yang digunakan untuk melakukan pengelolaan pada *database* yang telah dibuat[13]. Perintah penting pada DML yaitu:

##### a. *Insert*

*Insert* adalah perintah yang digunakan untuk memasukkan data baru ke dalam *database*. Perintah ini bisa dijalankan ketika *database* dan *table* sudah dibuat, fungsi *insert* ini untuk menambah data pada *table* tersebut.

##### b. *Update*

*Update* adalah perintah yang berfungsi untuk memperbarui data pada sebuah tabel.

##### c. *Select*

*Select* adalah perintah yang berfungsi untuk menampilkan dan mengambil data yang telah dimasukkan.

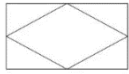
d. *Delete*

*Delete* adalah perintah untuk menghapus data.

Salah satu pemodelan awal yang paling sering digunakan adalah *Entity Relationship Diagram (ERD)*. *Entity Relationship Diagram (ERD)* digunakan untuk pemodelan relasional. Tujuan ERD adalah untuk menunjukkan objek data dan *relationship* yang ada pada objek tersebut. Di samping itu, ERD merupakan salah satu alat untuk perancangan dalam basis data. Simbol-simbol pada ERD dapat dilihat pada Tabel 2.3.




**Tabel 2. 3** Simbol ERD

No	Nama	Gambar	Deskripsi
1.	Entitas ( <i>Entity</i> )		Entitas merupakan data inti yang akan disimpan, bakal tabel pada basis data, benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer. Penamaan entitas biasanya lebih ke arah kata benda dan belum merupakan nama tabel.
2.	Relasi ( <i>Relationship</i> )		Relasi yang menghubungkan antar entitas biasanya diawali dengan kata kerja.
3.	Atribut		Atribut adalah karakteristik dari entitas atau dari relasi yang menyediakan penjelasan detail tentang entitas atau relasi tersebut. Atribut berupa <i>field</i> atau kolom data yang butuh disimpan dalam suatu entitas.
4.	<i>Link</i>		Garis sebagai penghubung antara himpunan, relasi, dan

			himpunan entitas dengan atributnya.
5.	<i>Associative Entity</i>		Entitas yang digunakan pada relasi <i>many-to many</i> .

ERD memiliki derajat relasi atau biasa disebut dengan kardinalitas. Kardinalitas mendeskripsikan batasan jumlah keterhubungan suatu *entity* dengan *entity* lainnya. Macam-macam kardinalitas dapat dilihat pada Tabel 2.4.

**Tabel 2. 4** Macam-Macam Kardinalitas

No	Nama	Gambar	Deskripsi
1.	Relasi Satu ke Satu ( <i>One to One</i> )		Relasi yang menunjukkan bahwa setiap himpunan entitas berhubungan dengan tepat satu himpunan entitas lainnya
2.	Relasi Satu ke Banyak ( <i>One to Many</i> )		Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak, begitu pula sebaliknya
3.	Relasi Banyak ke Banyak ( <i>Many to Many</i> )		Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya

## 2.4 Pendidikan

Definisi Pendidikan dalam arti luas adalah Hidup. Artinya bahwa pendidikan adalah seluruh pengetahuan belajar yang terjadi sepanjang hayat dalam semua tempat serta situasi yang memberikan pengaruh positif pada pertumbuhan makhluk hidup. Pendidikan dalam arti kata sempit adalah sebuah sekolah. Sistem itu berlaku untuk orang dengan berstatus sebagai murid yaitu

disekolah, atau peserta didik pada suatu universitas. Disekolah juga terdapat berbagai organisasi dan ekstrakurikuler yang berfungsi untuk menyalurkan bakat dan minat bagi siswa, seperti OSIS, MPK, IPNU-IPPNU, Pramuka, PMR, Taekwondo, Silat dan lainnya[14].

### **2.5 Laravel**

Laravel merupakan sebuah *Framework* PHP (*Hypertext Preprocessor*) yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (*Model, View, Controller*). Laravel adalah *Framework* PHP yang dibuat Taylor Otwell dan dirilis pertama kali pada tahun 2011. Laravel memiliki banyak Fitur Modern yang membantu dalam proses pengembangan *website* seperti *artisan, blade, template engine, database migration, pagination, dan eloquent ORM (Object Relation Mapping)*[7].