BAB II

LANDASAN TEORI

A. Metode Perancangan Perangkat Lunak *Waterfall* (Air Terjun)

Waterfall atau Air Terjun adalah model yang dikembangkan untuk pengembangan perangkat lunak, membuat perangkat lunak. model berkembang secara sistematis dari satu tahap ke tahap lain dalam mode seperti air terjun. Model ini mengusulkan sebuah pendekatan kepada pengembangan *software* yang sistematik dan sekuensial yang mulai dari tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model ini melingkupi aktivitas-aktivitas sebgai berikut : rekayasa dan pemodelan sistem informasi, analisis kebutuhan, desain, koding, pengujian dan pemeliharaan. Model pengembangan ini bersifat linear dari tahap awal pengembangan system yaitu tahap perencanaan sampai tahap akhir pengembangan sistem yaitu tahap pemeliharaan. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan dan tidak bisa kembali atau mengulang ke tahap sebelumnya.

Menurut Buku Rosa Metode pengembangan sistem merupakan proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan metode-metode atau model-model yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya dengan memiliki alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (suport).

B. Kerangka Algoritma Metode Topsis

Langkah-langkah algoritma dalam metode topsis adalah sebagai berikut:

1. Membuat matriks MADM dari permasalahan
2. Membuat matriks normal dari matriks MADM
3. Membuat vektor normal dari vektor bobot kriteria
4. Membuat matriks MADM normal terbobot normal
5. Membuat alternatif ideal positif
6. Membuat alternatif ideal negatif
7. Menghitung preferensi setiap alternatif
8. Merangking alternatif berdasarkan nilai preferensi.

Secara rinci, jika permasalahan dimodelkan dalam bentuk matriks MADM dan menerapkan metode topsis diatasnya maka langkah-langkah topsis menjadi sebagai berikut:

1. Membuat matriks MADM

	Kandidat 1	Kandidat 2	.....	Kandidat n
Kriteria 1	C11	C12	.....	C1n
Kriteria 2	C21	C22	.....	C2n
.....	.....	.....	.....	.....
Kriteria m	Cm1	Cm2	.....	Cmn

Gambar 2.1 Matriks MADM kandidat ketua BEM

2. Membuat matriks normal dari matriks MADM

Selanjutnya bentuk normal dari matriks MADM pada gambar 2.2 adalah didasarkan pada rumus 2.1 berikut:

$$r_{ij} = \frac{c_{ij}}{\sqrt{\sum_{j=1}^{n} c_{ij}^2}} \quad \text{..... (2.1)}$$

Sehingga diperoleh matriks ternormalisasi sebagaimana gambar 2.3.

	Kandidat 1	Kandidat 2	.....	Kandidat n
Kriteria 1	*r*11	*r*12	.....	*r*1n
Kriteria 2	*r*21	*r*22	.....	*r*2n
.....	.....	.....	.....	.....
Kriteria m	*r*m1	*r*m2	.....	*r*mn

Gambar 2.2 Matriks MADM normal

3. Membuat vektor normal dari vektor bobot kriteria

Masing-masing kriteria diberi bobot, baik secara subjective atau berdasar aturan tertentu. akan tetapi jika bobot-bobot kriteria telah ditentukan, misal bobot b_i untuk kriteria C_i maka diperoleh vektor bobot b yaitu (b1,b2b3,..,bm) untuk m buah kriteria. Normalisasi bobot kriteria ini dihitung dengan rumus 2.2.

$$w_i = \frac{b_i}{\sum_{i=1}^{m} b_i} \quad \text{..... (2.2)}$$

4. Membuat matriks MADM normal terbobot normal

Selanjutnya sebuah matriks normal terbobot normal dapat dibuat dengan menggunakan rumus 2.3.

$$y_{ij} = w_i . r_{ij} \quad \text{..... (2.3)}$$

	Kandidat 1	Kandidat 2	.....	Kandidat n
Kriteria 1	*y*11	*y*12	.....	*y*1n
Kriteria 2	*y*21	*y*22	.....	*y*2n
.....	.....	.....	.....	.....
Kriteria m	*y*m1	*y*m2	.....	*y*mn

Gambar 2.3 Matriks MADM normal terbobot normal dari kandidat

5. Membuat alternatif ideal positif

Selanjutnya adalah proses penghitungan kandidat ideal positif dengan cara mengambil seluruh nilai kriteria terbaik pada matriks normal terbobot normal sebagaimana gambar 2.3 lalu mengumpulkannya menjadi satu kandidat khusus yang bernama ideal positif.

6. Membuat alternatif ideal negatif

Dengan cara yang sama, ideal negatif dibentuk dengan mengumpulkan seluruh nilai kriteria terburuk pada matriks gambar 2.3. lalu mengumpulkannya membangun sebuah kandidat ideal yang bernama kandidat ideal negatif.

7. Menghitung preferensi setiap alternatif.

Untuk menghitung pereferensi, terlebih dahulu jarak setiap kandidat terhadap kandidat ideal positif dihitung, begitupun jarak setiap kandidat terhadap kandidat ideal negatif dhitung. Misalkan D_i adalah jarak kandidat A_i terhadap kandidat ideal positif dan misalkan L_i adalah jarak setiap kandidat A_i terhadap kandidat ideal negatif, maka nilai preferensi A_i, dinyatakan sebagai $Pr(A_i)$ adalah sebagaimana rumus 2.4.

$$\Pr(A_i) = \frac{L_i}{L_i + D_i} \quad \text{..... (2.4)}$$

8. Merangking alternatif berdasarkan nilai preferensi.

Langkah selanjutnya adalah merangking seluruh kandidat berdasarkan nilai preferensinya.

C. Pengertian DSS (*Decision Support System*)

Little (1970) mendefinisikan DSS sebagai " sekumpulan prosedur model untuk data pemrosesan dan penilaian guna membantu para manajer mengambil keputusan". Dia menyatakan bahwa untuk sukses, sistem tersebut haruslah sederhana, cepat, mudah dikontrol, adaptif, lengkap dengan isu- isu penting dan mudah berkomunikasi. Alter (1980) mendefinisikan DSS dengan membandingkannya dengan EDP (*electronic data processing)* tradisional pada lima dimensi.

Moore (1980) berpendapat bahwa konsep struktur, seperti yang banyak disinggung pada definisi awal DSS (bahwa DSS dapat menangani situasi semiterstruktur dan tidak tersetruktur), secara umum tidaklah penting, tersetruktur hanya dengan memperhatikan si pengambil keputusan atau suatu situasi spesifik.

Alter,S.L. (1980) mendefinsikan DSS dengan membandingkannya dengan sistem EDP (Electronic Data Processing) tradisional pada lima dimensi yaitu pada tabel DSS versus EDP.

Tabel 1. DSS versus EDP

Dimensi	DSS	EDP
Penggunaan	Aktif	Pasif
Pengguna	Lini manajemen staf	Klerikal
Tujuan	Keefektifan	Efisiensi mekanis
Horison waktu	Masa sekarang dan akan datang	Masa lalu
Tujuan	Fleksibilitas	Konsistensi

Bonczek, R.H, (1980) mendefinsikan "DSS sebagai sistem berbasis computer yang terdiri dari tiga komponen yang saling berinteraksi : sistem bahasa (mekanisme untuk memberikan komunikasi antara pengguna dan komponen DSS lain), sistem pengetahuan (repository pengetahuan domain masalah yang ada pada DSS entah sebagai data atau sebagai prosedur), dan sistem pemrosesan masalah (hubungan antara dua komponen lainnya, terdiri dari satu atau lebih kapabilitas manipulasi masalah umum yang diperlukan untuk pengambilan keputusan).

Keen, P.G.W. (1980) mendefinisikan "DSS sebagai suatu produk dari proses pengembangan dimana pengguna DSS, pembangun DSS, dan DSS itu sendiri mampu mempengaruhi satu dengan yang lainnya, dan menghasilkan evolusi sistem dan pola-pola penggunaan". Sistem interaktif berbasis computer, yang membantu pembuat keputusan dengan menggunakan data dan model untuk

menyelesaikan masalah yang tidak terstruktur. Terlihat bahwa definisi-definisi tersebut diperbandingkan dan dikontraskan dengan memeriksa berbagai konsep yang digunakan untuk mendefinisikan DSS (tabel konsep yang mendasari Definisi DSS).

Tabel 2.1. Konsep yang mendasari definisi DSS

Sumber	DSS yang Didefinisikan
Gorry dan Scott-Morton (1977)	Tipe masalah, fungsi sistem (dukungan)
Little (1970)	Fungsi sistem, karakteristik antarmuka
Alter (1980)	Pola penggunaan, tujuan sistem
Moore dan chang (1980)	Pola penggunaan, kapabilitas sistem
Bonczek, dkk.,(1989)	Komponen-komponen sistem
Keen (1980)	Proses pengembangan

DSS biasanya dibangun untuk mendukung solusi terhadap suatu masalah atau untuk mengevaluasi suatu peluang. DSS yang seperti itu disebut DSS aplikasi.

Definisi DSS yang lain:

a. Sistem berbasis komputer yang interaktif, yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah tak terstruktur.
b. DSS mendayagunakan resources individu-individu secara intelek dengan kemampuan komputer untuk meningkatkan kualitas keputusan. Jadi merupakan sistem pendukung berbasis komputer untuk manajemen pengambilan keputusan yang berhubungan dengan masalah-masalah yang semi terstruktur.
c. Sistem tambahan yang mampu untuk mendukung analisis data secara ad hoc dan pemodelan keputusan, berorientasi pada perencanaan masa depan, dan digunakan pada interval yang tak teratur atau tak terencana.
d. Sistem berbasis komputer yang terdiri atas 3 komponen interaktif: (1) sistem bahasa – mekanisme yang menyediakan komunikasi diantara user dan pelbagai komponen dalam DSS, (2) knowledge systems – penyimpanan knowledge domain permasalahan yang ditanamkan dalam DSS, baik sebagai data ataupun prosedur, dan (3) Sistem pemrosesan permasalahan – link diantara dua komponen, mengandung satu atau lebih kemampuan memanipulasi masalah yang dibutuhkan untuk pengambilan keputusan.

Karakteristik dan Kapabilitas DSS.

Belum ada konsensus mengenai apa sebenarnya DSS, maka jelas belum ada kesepakatan mengenai karakteristik dan kapabilitas standar DSS, namun ada beberapa definisi dapat disimpulkan bahwa karekteristik dan kapabilitas kunci DSS sbb :

1. Dukungan untuk pengambil keputusan, terutama pada situasi semiterstruktur dan tak terstruktur, dengan menyertakan penilaian manusia dan informasi terkomputerisasi. Masalah-masalah tersebut tidak dapat dipecahkan (atau tidak dapat dipecahkan dengan konvenien) oleh sistem computer lain atau oleh metode atau alat kuantitatif standar.
2. Dukungan untuk semua level manajerial, dari eksekutif puncak sampai manajer lini
3. Dukungan untuk individu dan kelompok. Masalah yang kurang terstruktur sering memerlukan keterlibatan individu dari departemen dan tingkat organisasional yang berbeda atau bahkan dari organisasi lain. DSS mendukung tim virtual melalui alat-alat Web kolaboratif.
4. Dukungan untuk keputusan independen dan atau sekuensial. Keputusan dapat dibuat satu kali, beberapa kali, atau berulang (dalam interval yang sama)
5. Dukungan di semua fase proses pengambilan keputusan: intelegensi, desain, pilihan dan implementasi.
6. Dukungan untuk di berbagai proses dan gaya pengambilan keputusan.

7. Adoptivitas sepanjang waktu. Pengambilan keputusan seharusnya reaktif, dapat menghadapi perubahan kondisi secara cepat, dan dapat mengadaptasikan DSS untuk memenuhi perubahan tersebut. DSS bersifat fleksibel dan karena itu pengguna dapat menambahkan, menghapus, menggabungkan, mengubah, atau menyusun kembali elemen-elemen dasar. DSS juga fleksibel dalam hal dapat dimodifikasi untuk memecahkan masalah lain yang sejenis.
8. Pengguna merasa seperti di rumah. Rumah-pengguna, kapabilitas grafis yang sangat kuat, dan antarmuka menusia-mesin interaktif dengan satu bahasa alami dapat sangat meningkatkan keefektifan DSS. Kebanyakan aplikasi DSS yang baru menggunakan antarmuka berbasis-Web.
9. Peningkatan terhadap keefektifan pengambilan keputusan (akurasi, timeliness, kualitas) ketimbang pada efisiensinya (biaya pengambilan keputusan). Ketika DSS disebarkan, pengambilan keputusan sering membutuhkan waktu lebih lama, namun keputusannya lebih baik.
10. Kontrol penuh oleh pengambil keputusan terhadap semua langkah proses pengambilan keputusan dalam memecahkan suatu masalah. DSS secara khusus menekankan untuk mendukung pengambilan keputusan, bukannya menggantikan.
11. Pengguna akhir dapat mengembangkan dan memodifikasi sendiri sistem sederhana. Sistem yang lebih besar dapat dibangun dengan bantuan ahli sistem informasi.
12. Biasanya model-model digunakan untuk menganalisis situasi pengambilan keputusan. Kapabilitas pemodelan memungkinkan

eksperimen dengan berbagai strategi yang berbeda di bawah konfigurasi yang berbeda. Sebenarnya, *model-model membuat DSS berbeda dari kebanyakan MIS*.

13. Akses disediakan untuk berbagai sumber data, format, dan tipe, mulai dari sistem informasi geografis (GIS) sampai sistem berorientasi-objek.
14. Dapat dilakukan sebagai alat standalone yang digunakan oleh seorang pengambil keputusan pada satu lokasi atau didistribusikan di satu organisasi keseluruhan dan di berbagai organisasi sepanjang rantai persediaan. Dapat diintegrasikan dengan DSS lain dan atau aplikasi lain.

Keuntungan DSS :

a. Mampu mendukung pencarian solusi dari masalah yang kompleks.
b. Respon cepat pada situasi yang tak diharapkan dalam kondisi yang berubah-ubah.
c. Mampu untuk menerapkan pelbagai strategi yang berbeda pada konfigurasi yang berbeda secara cepat dan tepat.
d. Pandangan dan pembelajaran baru.
e. Memfasilitasi komunikasi.
f. Meningkatkan kontrol manajemen dan kinerja.
g. Menghemat biaya.
h. Keputusannya lebih tepat.
i. Meningkatkan efektivitas manajerial, menjadikan manajer dapat bekerja lebih singkat dan dengan sedikit usaha.
j. Meningkatkan produktivitas analisis

D. Pengertian MySQL

MySQL adalah *database server* yang cukup popular, cepat dan tangguh, sangat cocok jika digabungkan dengan PHP, dengan *database* kita bisa menyimpan, mencari dan mengklasifikasikan data dengan lebih akurat dan professional. *MySQL* menggunakan SQL *language* (*Structure Query Language*) artinya *MySQL* menggunakan *query* atau bahasa pemprogaman yang sudah standar di dalam dunia *database*.

Implementasi program *server database* ini adalah program daemon '*MySQL*d' dan beberapa program lain serta beberapa pustaka. *MySQL* dibuat oleh TcX dan telah dipercaya mengelola sistem dengan 40 buah *database* berisi 10,000 tabel dan 500 di antaranya memiliki 7 juta baris (kira-kira 100 gigabyte data). *Database* ini dibuat untuk keperluan sistem *database* yang cepat, handal dan mudah digunakan.

Walaupun memiliki kemampuan yang cukup baik, *MySQL* untuk sistem operasi Unix bersifat *freeware*, dan terdapat versi shareware untuk sistem operasi *windows*. Menurut pembuatnya, *MySQL* disebut seperti "*myessqueell*" dan bukan *mysequel*. Sebagaimana *database* sistem yang lain, dalam SQL juga dikenal hierarki *server* dengan *database-database*. Tiap-tiap *database* memiliki table-tabel, tiap-tiap tabel memiliki *field-field*. Umumnya informasi tersimpan dalam table-tabel yang secara logic merupakan struktur dua dimensi terdiri atas baris dan kolom. *Field-field* tersebut dapat berupa data *seperti int , realm char, date, time* dan lainnya.

SQL tidak memiliki fasilitas pemrograman yang lengkap, tidak ada looping ataupun percabangan ,misalnya. Sehingga untuk menutupi kelemahan ini perlu digabung dengan bahasa pemrograman semisal C.

Fitur M*ySQL* :

a. Didukung sepenuhnya oleh bahasa pemprograman C, C++, *eifel*, Java, *Perl*, PHP, *Phyton* dan *Tcl* untuk mengakses *databaseMySQL*.
b. Dapat bekerja pada banyak *platform* yang berbeda, termasuk juga di dalamnya *windows*.
c. Banyaknya tipe kolom : *signed unsigned integer* 1 dan 8 bytes, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR,SET dan tipe ENUM.
d. Mendukung sepenuhnya parameter SQL GROUP BY dan ORDER BY. Fungsi yang dapat dipakai dalam group *query* : (COUNT (), COUNT(DISTENCT, AVG (), STD(), SUM(), MAX(), and MIN()).
e. Sistem *privilege* dan *password* dapat terjaga kerahasiaanya dan dapat diverifikasi berdasarkan nama *host*nya. *Password* terjaga kerahasiaanya karena semua *password* disimpan dalam keadaan terenkripsi.

Sebagai *database* yang memilikikonsep *database* modern, MySQL memiliki banyak sekali keistimewaan. Berikut ini beberapa keistimewaan yang dimiliki oleh MySQL :

a) Portability
b) Open Source
c) Multiuser

d) *Performance Tuning*

e) *Column Types*

f) *Command dan Function*

g) *Security*

h) *Stability dan Limits*

i) *Connectivity*

j) *Localisation*

k) *Interface*

l) *Client dan Tools*

m) Struktur Tabel

E. Tentang PHP

PHP diciptakan oleh Rasmus Lerdorf pada tahun 1994 dan bersifat *open source*. Sampai bulan januari 2007, PHP sudah digunakan oleh kurang lebih 20 juta *domain* dan terus berkembang sampai saat ini. PHP merupakan singkatan dari *Hypertext Preprocessor*, adalah sebuah bahasa scripting yang terpasang pada HTML. Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama bahasa ini adalah untuk memungkinkan perancang *web* menulis halaman *web* dinamis dengan cepat.

PHP merupakan bahasa pemograman *web* yang bersifat *server-side HTML=embedded scripting*, di mana script-nya menyatu dengan HTML dan berada si *server*. Artinya adalah sintaks dan perintah-perintah yang kita berikan akan sepenuhnya dijalankan di *server* tetapi disertakan HTML biasa.

PHP dikenal sebgai bahasa scripting yang menyatu dengan *tag* HTML, dieksekusi di *server* dan digunakan untuk membuat halaman *web* yang dinamis seperti ASP (*Active Server Pages*) dan JSP (*Java Server Pages*).

Seluruh aplikasi berbasis *web* dapat dibuat dengan PHP. Namun kekuatan yang paling utama PHP adalah pada konektivitasnya dengan system *database* di dalam *web*. Sistem *database* yang dapat didukung oleh PHP adalah :

a. *Oracle*
b. *MySQL*
c. *Sybase*
d. *PostgreSQL*

PHP dapat berjalan di berbagai system operasi seperti *windows* 98/NT, UNIX/LINUX, solaris maupun *macintosh*.*Software* ini juga dapat berjalan pada *web server* seperti PWS (*Personal Web Server*), *Apache*, IIS, *AOLServer*, fhttpd, phttpd dan sebagainya. PHP juga merupakan bahasa pemograman yang dapat kita kembangkan sendiri seperti menambah fungsi-fungsi baru. Keunggulan lainnya dari PHP adalah PHP juga mendukung komunikasi dengan layanan seperti protocol IMAP, SNMP, NNTP, POP3 bahkan HTTP. PHP dapat diinstal sebagai bagian atau modul dari apache web *server* atau sebagai CGI *script* yang mandiri.

Banyak keuntungan yang dapat diperoleh jika menggunakan PHP sebagai modul dari apache di antaranya adalah :

a) Tingkat keamanan yang cukup tinggi

b) waktu eksekusi yang lebih cepat dibandingkan dengan bahasa pemograman web lainnya yang berorientasi pada *server-side scripting*.

c) Akses ke sistem *database* yang lebih fleksibel seperti MySQL.

Script yaitu kumpulan instruksi program yang tidak memerlukan kompilasi dan hasilnya ditampilkan pada browser. Yang termasuk kedalam kategori script, yaitu : JavaScript, VBScript, PHP, ASP atau JSP.

TAG merupakan tanda yang mengapit sebuah elemen pada dokumen HTML atau Script. Contoh: <B>, yang merupakan tag <> sedangkan B adalah elemen HTML. Kesimpulannya tag berperan dalam mengatur elemen-elemen dalam dokumen HTML atau script. Script berperan dalam instruksi program antara HTML dan Script PHP dapat saling berdiri sendiri atau juga dapat saling berkaitan.

Contoh HTML atau script PHP yang saling berdiri sendiri :

- HTML
<HTML>
<BODY>
<P>Haloo</P>
</BODY>
</HTML>

- PHP
<?
echo"Selamat Belajar PHP";
?>
Contoh: HMTL atau Script PHP yang saling berkaitan: (Script PHP dalam HTML)
<HTML>
<TITLE>TES</TITLE>
<BODY>
<B><? echo"Selamat Belajar PHP"; ?></B>

</BODY>
</HTML>
Contoh: HTML dalam Script
<?
echo"<HTML>";
echo"<TITLE>TES</TITLE>";
echo"<BODY>";
echo"<P align=center>Mengenal PHP</P>";
echo"<B><Font Size=20>Selamat Belajar PHP</B></Font>";
echo"<BODY>";
echo"<HTML>";
?>

PHP memiliki struktur penulisan seperti berikut :

<?

Isi Script

?>

Setiap script php harus diawali dengan tag <? dan diakhiri dengan tag ?>

Elemen HTML yang berada di antara tag <? dan tag ?> harus diawali dengan script echo.Contoh : <? echo"<B>Belajar PHP</B>"; ?>

Pendeklarasian variable dan pencetakan statemen harus diakhiri dengan ; (semicolon), Contoh : <?$Nama="Sukarno";?> atau <? echo"My Solution";?>

Untuk penulisan keterangan atau komentar, terbagi menjadi 2 cara :

1. Per komentar (per baris), penulisannya diawali dengan tanda //

   <?

   // Ini tanda komentar per komentar

   echo"Komentar per komentar";

   ?>

2. Antar komentar, penulisannya diawali dengan tanda /* dan diakhiri dengan tanda */.

```
<?
/*Ini termasuk komentar juga tetapi komentar jenis antar komentar
Atau
Sistem multiline.
*/
echo”Komentar antar komentar”;
?>
```

Setiap pemrograman web atau aplikasi dibutuhkan apa yang namanya variabel. Variabel adalah sebuah bentuk pendeklarasian nama yang memiliki nilai dengan tipe data tertentu.

Setiap pendeklarasian variabel pada PHP dimulai dengan menggunakan tanda $ (dollar). Bentuk penulisan pendeklarasian variabel seperti berikut :

```
$Nama variabel = nilai variabel;
$A = 1;
$Nama = “Angkasa”;
$Point=””;
Contoh progran phpnya:
<?
$Negara=”Indonesia”;
echo”Negaraku adalah $Negara”;
?>
```

Adapun variabel yang dideklarasikan pada form melalui dokumen HTML dapat dilihat seperti pada file **input.html** seperti berikut:

```
<form method=”POST” action=”cek.php”>
Nama : <input type=”text” name=”Nama”>
Email : <input type=”text” name=”Email”>
Telephone : <input type=”text” name=”Phone”>
```

```
<input type=”submit” value=”Submit” name=”Submit”>
</Form>
```

Dari form di atas terdapat 3 buah variabel, yaitu : Nama, Email, dan Phone. Nilai dari masing-masing variabel tersebut tergantung dari karakter atau angka yang dimasukkan pada textbox tersebut.

Pada form input.html tersebut telihat Methode=Post dan Action = cek.php, itu maksudnya hasil submitnya akan dikirim ke cek.php.

Script cek.php dapat dilihat seperti berikut:

```
<?
echo”Nama : $Nama<br>”;
echo”Email : $Email<br>”;
echo”Telephone : $Phone<br>”;
?>
```

PHP mengenal 5 tipe data, yaitu : String, Integer, Float, Array, dan Objects. Setiap nilai pada suatu variabel emiliki tipe data yang disesuaikan dengan nilai itu sendiri.

Operator adalah sebuah tanda yang memberika arti atau nilai pada suatu variabel. Macam-macam operator yaitu Operator aritmatika, *bitwise*, *assignment*, perbandingan, logika, dan *increament* atau *decreament*.

F. *Flowchart*

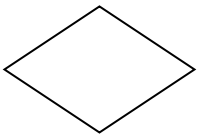
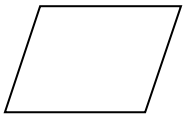
Karena komputer membutuhkan hal-hal yang rinci, maka bahasa pemrograman bukanlah alat baik untuk merancang sebuah *algoritma* awal. Alat yang banyak dipakai untuk membuat *algoritma* adalah diagram alur (*flowchart*).

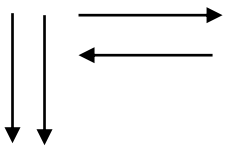
Diagram alur dapat menunjukkan secara jelas arus pengendalian suatu *algoritma*, yakni melaksanakan suatu rangkaian kegiatan secara *logis* dan *sistematis*. Suatu diagram alur dapat memberi gambaran dua *dimensi* berupa simbol-simbol grafis. Masing-masing simbol telah ditetapkan lebih dahulu fungsi dan artinya. Simbol-simbol tersebut dipakai untuk menunjukkan berbagai kegiatan operasi dan jalur pengendalian. Arti khusus dari sebuah *flowchart* adalah simbol-simbol yang digunakan untuk menggambarkan urutan proses yang terjadi di dalam suatu program komputer secara *sistematis* dan *logis*. (Sutabri; 2004; 21).

a. Simbol-simbol *flowchart*

Sudah dikemukakan di atas bahwa diagram alur atau *flowchart* memiliki beberapa simbol yang biasa digunakan untuk menggambarkan rangkaian proses yang harus dilaksanakan. Simbol-simbol tersebut dijelaskan di bawah ini: (Sutabri; 2004; 21-22)

Tabel 2.2 Simbol *Flowchart*

Simbol *Flowchart*	Fungsi
	*TERMINAL* Simbol ini digunakan untuk mengawali atau mengakhiri suatu proses/kegiatan.
	*PREPARATION* Simbol ini digunakan untuk mempersiapkan harga awal/nilai awal suatu variabel yang akan diproses.
	*DECISION* Simbol ini digunakan untuk pengujian suatu kondisi yang sedang diproses.
	PROSES Simbol ini digunakan untuk menggambarkan suatu proses yang sedang dieksekusi.
	*INPUT/OUTPUT* Simbol ini digunakan untuk menggambarkan proses input (*read*) maupun proses output (print).
	*SUBROUTINE* Simbol ini digunakan untuk menggambarkan proses pemanggilan subprogram dari main program.



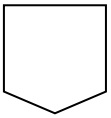
FLOW LINE

Simbol ini digunakan untuk menggambarakan arus proses dari suatu kegiatan ke kegiatan lain.

CONECTOR

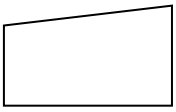
Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya yang ada di dalam suatu lembar halaman.

PAGE CONECTOR



Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya, tetapi berpindah halaman.

MANUAL *OPERATION*



Simbol ini digunakan untuk menggambarkan suatu kegiatan atau proses yang bersifat manualisasi.

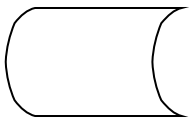
PRINTER

Digunakan untuk menggambarkan suatu kegiatan mencetak suatu informasi dengan mesin printer.

CONSOLE

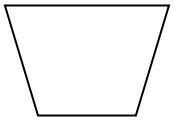
Simbol ini digunakan untuk menggambarkan suatu kegiatan menampilkan data atau informasi melalui monitor atau CRT (*Cathode Ray Tube*).

DISK



Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic disk*.

MANUAL *INPUT*



Simbol ini digunakan untuk menggambarkan proses pemasukan data melalui media *keyboard*.

TAPE

Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic tape*.

---

b. Jenis *flowchart*

Bentuk diagram alur (*flowchart*) yang sering digunakan dalam proses pembuatan suatu program komputer adalah sebagai berikut:

1) Program *flowchart*

Simbol-simbol yang menggambarkan proses secara rinci dan detil antara intruksi yang satu dengan intruksi yang lainnya dalam suatu program komputer yang bersifat *logik*.

2) Sistem *flowchart*

Simbol-simbol yang menggambarkan urutan prosedur secara detil dalam suatu sistem komputerisasi. Bersifat fisik.

c. Teknik pembuatan *flowchart*

Sebelum kita membuat sebuah program komputer, yang harus kita lakukan sebelumnya adalah membuat *flowchart*. Jenis *flowchart* yang sering digunakan adalah program *flowchart*.

Teknik pembuatan program *flowchart* ini dibagi menjadi dua bagian, yaitu:

1) *General way*

Teknik pembuatan *flowchart* dengan cara ini lazim digunakan untuk menyusun logika suatu program. Teknik ini menggunakan pengulangan proses secara tidak langsung (*Non-Direct-Loop*).

2) Iteration way

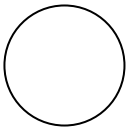
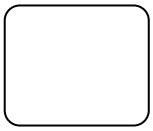
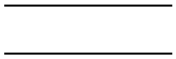
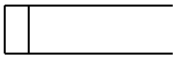
Teknik pembuatan flowchart dengan cara ini biasanya dipakai untuk logika program yang cepat dan bentuk permasalahannya kompleks. Pengulangan proses yang terjadi bersifat langsung (Direct-Loop). (Sutabri; 2004; 24).

G. Tentang DFD (*Data Flow Diagram*)

Data Flow Diagram *(DFD)* adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. Kita dapat menggunakan *DFD* untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem informasi yang ada, atau untuk menyusun dokumentasi untuk sistem informasi yang baru.

Empat simbol yang digunakan :

Tabel 2.3 Simbol *DFD*

Notasi *Yourdon DeMarco*	**Notasi *Gane Sarson***	**Fungsi**
		Simbol Entitas eksternal atau t*erminator* menggambarkan asal atay tujuan data di luar sistem
		Simbol lingkaran menggambarkan entitas atau proses dimana aliran data masuk ditransformasikan ke aliran data keluar
		Simbol aliran data menggambarkan aliran data
		Simbol *file* menggambarkan tempat data disimpan

Ada 3 (tiga) jenis *DFD*, yaitu :

1. *Diagram contex*

   Jenis pertama *Context Diagram*, adalah data *flow* diagram tingkat atas (*DFD Top Level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar *entitas-entitas eksternal*. (CD

menggambarkan sistem dalam satu lingkaran dan hubungan dengan *entitas* luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem).

Beberapa hal yang harus diperhatikan dalam menggambar CD :

a. Terminologi sistem
    1) Batas Sistem adalah batas antara “daerah kepentingan sistem”.
    2) Lingkungan Sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut.
    3) *Interface* adalah aliran yang menghubungkan sebuah sistem dengan linkungan sistem tersebut.
b. Menggunakan satu simbol proses.

    Yang masuk didalam lingkaran konteks (simbol proses) adalah kegiatan pemrosesan informasi (Batas Sistem). Kegiatan informasi adalah mengambil data dari *file*, *mentransformasikan* data, atau melakukan *filing* data, misalnya mempersiapkan dokumen, memasukkan, memeriksa, mengklasifikasi, mengatur, menyortir, menghitung, meringkas data, dan melakukan *filing* data (baik yang melakukan secara manual maupun yang dilakukan secara *terotomasi*).
c. Nama/keterangan di simbol proses tersebut sesuai dengan fungsi sistem tersebut.

d. Antara *Entitas Eksternal/Terminator* tidak diperbolehkan komunikasi langsung.
e. Jika terdapat *termintor* yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda *asterik* (*) atau garis silang (#).
f. Jika Terminator mewakili individu (personil) sebaiknya diwakili oleh peran yang dipermainkan *personil* tersebut.
g. Aliran data ke proses dan keluar sebagai output keterangan aliran data berbeda.

2. *DFD fisik*

DFD *fisik* adalah *representasi* grafik dari sebuah sistem yang menunjukan *entitas-entitas internal* dan *eksternal* dari sistem tersebut, dan aliran-aliran data ke dalam dan keluar dari *entitas-entitas* tersebut. *Entitas-entitas internal* adalah *personal*, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang *mentransformasikan* data. Maka *DFD* fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan.

Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol aliran data) dalam *DFD* fisik menggunakan label/keterangan dari kata benda untuk menunjukan bagaimana sistem *mentransmisikan* data antara lingkaran-lingkaran tersebut.

3. *DFD logis*

DFD *Logis* dalah *representasi* grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Kita menggunakan *DFD logis* untuk membuat dokumentasi sebuah sistem informasi karena *DFD logis* dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana, dan oleh siapa proses-proses dalam sistem tersebut dilakukan. Keuntungan dari *DFD logis* dibandingkan dengan *DFD* fisik adalah dapat memusatkan perhatian pada fungsi - fungsi yang dilakukan sistem (Abdul Kadir, 2008).