BAB 2

LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artificial Intelligence*)

Bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia, bahkan bisa lebih baik dari pada yang dilakukan manusia, untuk mengetahui dan memodelkan proses – proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia [9].

2.1.1 BEDA KECERDASAN BUATAN & KECERDASAN ALAMI

Kelebihan kecerdasan buatan :

1. Lebih bersifat permanen. Kecerdasan alami bisa berubah karena sifat manusia pelupa. Kecerdasan buatan tidak berubah selama sistem komputer & program tidak mengubahnya.
2. Lebih mudah diduplikasi & disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama & keahlian tidak akan pernah dapat diduplikasi dengan lengkap.Jadi jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut & dapat dipindahkan dengan mudah ke komputer yang lain.
3. Lebih murah. Menyediakan layanan komputer akan lebih mudah & murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Bersifat konsisten dan teliti karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Keputusan yang dibuat komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Dapat mengerjakan beberapa task lebih cepat dan lebih baik dibanding manusia.

2.1.2 Kelebihan kecerdasan alami :

1. Kreatif manusia memiliki kemampuan untuk menambah pengetahuan, sedangkan pada kecerdasan buatan untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun
2. Memungkinkan orang untuk menggunakan pengalaman atau pembelajaran secara langsung. Sedangkan pada kecerdasan buatan harus mendapat masukan berupa input-input simbolik.
3. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan sangat terbatas[9].

2.2 Algoritma

Algoritma merupakan salah satu cabang ilmu komputer yang membahas prosedur penyelesaian suatu permasalahan. Algoritma adalah urutan langkah-langkah dalam menentukan suatu masalah. Algoritma juga dapat didefinisikan dengan deretan langkah komputasi yang mentransformasikan masukan (*input*) menjadi keluaran (*output*). Jadi algoritma adalah serangkaian urutan langkah-langkah atau prosedur untuk menyelesaikan suatu masalah dengan memproses nilai masukan menjadi nilai keluaran.

Algoritma dapat ditulis dalam bentuk *pseudocode* ataupun *flowchart*. *Pseudocode* berasal dari kata pseudo yang artinya mirip atau menyerupai dan code berarti kode sehingga *pseudocode* adalah kode yang menyerupai kode program sebenarnya. Sedangkan *flowchart* merupakan bentuk gambar atau diagram yang mempunyai aliran satu atau dua arah secara sekuensial.

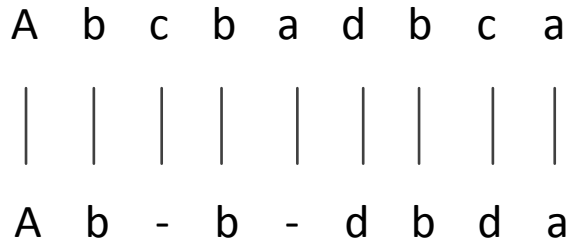
Dalam mempelajari algoritma, diperlukan untuk dapat menganalis algoritma dalam menentukan performansinya. Suatu Algoritma tidak saja harus benar, tetapi juga harus efisien dengan menghitung nilai dari performansinya. Setiap algortima memiliki nilai performansi yang berbeda-beda [10].

2.2.1 Smith Waterman

Algoritma Smith-Waterman diperkenalkan oleh Temple F. Smith dan Michael S. Waterman pada tahun 1981 di sebuah Jurnal Molekular Biologi dalam papernya yang berjudul *Identification of Common Molecular Subsequences* [1]. Algoritma ini digunakan untuk mencari kesamaan yang paling signifikan *(local alignment*) dari dua buah rangkaian sekuens gen, dimana setiap sekuens gen dalam rangkaian yang satu dibandingkan dengan sekuens gen rangkaian yang lain berdasarkan kesamaan strukturnya. Signifikan dalam hal ini memenuhi suatu nilai ambang batas (*threshold*) tertentu.

Cara kerja algoritma ini yaitu dengan cara membandingkan dua buah string dengan mengidentifikasikan apakah terdapat bagian-bagian yang sama di antara kedua string tersebut.

Algoritma *smith waterman* merupakan algoritma klasik yang telah dikenal luas dalam bidang bioinformatika sebagai metode yang dapat mengidentifikasi *local similarities* (penjajaran sekuens) yaitu proses penyusunan dua *local sequences* (rangkaian atau susunan) *neucleotide* atau *protein sequences* sehingga kemiripan antara dua *sequences* tersebut akan terlihat. Berikut ini gambar optimal *aligment* dari dua substring dapat dilihat pada gambar 2.1.



Gambar 2.1 Optimal *Aligment* Dari Dua Substring

Algoritma standar Smith- Waterman yang dipakai untuk perhitungan *local alignment* berdasarkan dokumen dari situs Baylor College of Medicine HGSC adalah [11]:

1. Menambahkan sebuah nilai pada setiap nilai perbandingan

a. Menggunakan nilai positif, apabila memiliki kemiripan.
b. Menggunakan nilai negatif, apabila memiliki perbedaan.
2. Inisialisasi awal matriks dengan nilai 0 (nol).
3. Semua nilai yang terdapat dalam matriks apabila lebih kecil dari 0 (nol), maka nilai dianggap 0 (nol).
4. Memulai *traceback* dari nilai yang tertinggi yang ditemukan dimanapun pada matriks.
5. Perhitungan dilanjutkan hingga skor bernilai 0 (nol).

Algoritma ini juga dapat digunakan dalam permasalahan *string matching* dengan menerapkan *dynamic programming*, yaitu metode penyelesaian masalah dari suatu permasalahan dengan menguraikan solusi menjadi sekumpulan langkah- langkah sedemikian sehingga solusi dari sebuah persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan.

Rumusan skema pemrograman dinamis algoritma *Smith-Waterman* menjadi dua bagian, yaitu [5] :

1. Didefinisikan Sij menjadi nilai maksimum yang didapat dari proses perbandingan sebuah string A pada posisi ke-i dengan sebuah substring B pada posisi ke-j. Hubungan rekurens standar untuk Sij yaitu :

- Jika A(i) = B(j) maka Sij = Si-1j-1+h, atau
- Jika A(1) ≠ B(j) maka Sij = max(0,Si-1,j-d, Si,j-1-d, Si-1,j-1-r)
- Dimana kondisi awal adalah Si,0 = S0,j = 0 untuk semua i,j.

2. Digunakan ide *traceback path* untuk mengkonstruksikan sebuah *local aligment* yang optimal pada posisi ke-i substring A dan posisi substring ke-j substring B agar lebih jelas terlihat. Dengan diberikan sel (i,j) dapat didefinisikan sebuah sel *parent* sebagai berikut :

- Jika Sij = 0 maka sel (i,j) tidak mempunyai *parent*.
- Jika A(i) = B(j), maka sel (i,j) mempunyai *parent* sel (i-1,j-1).
- Sebagai tambahan, sel (i,j) mempunyai *parent* yaitu untuk sel (p,q){(i-1,j),(i,j-1)} sehingga Sij = Spq-d, dan atau sel (i-1,j-1) jika Sij = Si-1,j-1-r.

Jadi, untuk setiap sel yang mengandung nilai tidak nol akan mempunyai sedikitnya satu *parent*, dan mungkin dapat memiliki tiga buah *parent*.

2.2.1.1 Menentukan Perhitungan Bobot

Dalam mendapatkan *local similarities* dan bobot kesamaan kata dari jawaban dan kunci jawaban, maka diperlukan beberapa tahap. Adapun tahap-tahapnya adalah sebagai berikut [1] :

1. Input Jawaban

Pada tahap ini akan dilakukan pengecekan awal kalimat inputan. Inti dari proses ini adalah mencari atau menentukan jumlah kata yang terdapat dalam kalimat jawaban yang diinputkan.

2. Konversi Huruf

Untuk membuat sistem mampu menyamakan tiap karakter dari jawaban dengan kunci jawaban lainnya, maka semua huruf dalam kalimat atau kata dari tiap jawaban perlu diubah menjadi huruf kecil.

Tabel 2.1 Konversi huruf menjadi kecil

Sebelum di konversi menjadi huruf kecil	Sesudah di konversi menjadi huruf kecil
Pernyataan yang dimiliki dengan mengemukakan hal yang umum kemudian disusun dengan uraian yang khusus.	pernyataan yang dimiliki dengan mengemukakan hal yang umum kemudian disusun dengan uraian yang khusus.

3. Tokenisasi

Proses Tokenisasi yaitu proses untuk membagi kalimat menjadi token-token atau bagian-bagian terkecil. Karakter spasi dan karakter lain berupa tanda baca pun dipisahkan. Adapun beberapa tanda baca yang dipisahkan adalah koma (,), titik (.), titik dua (:), garis miring (/), tanda tanya (?), dan tanda garis/pisah (-). Tanda baca dipisahkan guna memisahkan kata dari karakter tanda baca. Hasil dari proses tokenisasi ini disebut sebagai token.

Tabel 2.2 Proses Tokenisasi

pernyataan yang dimiliki dengan mengemukakan hal yang umum kemudian disusun dengan uraian yang khusus.	Pernyataan yang dimiliki dengan mengemukakan hal yang umum kemudian disusun dengan uraian yang khusus

4. Proses Smith Waterman

Algoritma *smith waterman* memerlukan suatu kombinasi skor yang digunakan utnuk mengisi setiap nilai pada matrik kesamaan. Kombinasi skor tersebut terdiri dari tiga jenis yaitu :

1. *Match* adalah skor untuk pasangan yang sama.
2. *Mismatch* adalah skor untuk pasangan karakter yang tidak sama.
3. *Gap* adalah skor untuk pasangan karakter yang berasal dari baris dan kolom yang berlainan.

a. Pengisian Matriks

Pada tahap ini diperlukan sebuah matriks dua dimensi berukuran dimana $Mm+1, n+1$ dimana m adalah panjang dari sekuens A dan n adalah panjang dari sekuens B, aturan inisialisasi pada suatu matrik $Mm+1, n+1$ adalah sebagai berikut :

$M_{i,0} = 0, 0 \leq i \leq m$

$M_{0,j} = 0, 0 \geq j \geq n$

Setelah inisialisasi selesai kemudian akan dilakukan proses pengisian nilai pada matriks kesamaan. Aturan pengisian nilai dari setiap elemen sebuah sekuens pada matriks berdasarkan pada aturan berikut :

$$H_{i,j} = maks \begin{Bmatrix} 0 \\ H(i-1,j-1) + nilai\ match/mismatch \\ H(i-1,j) + nilai\ gap \\ H(i,j-1) + nilai\ gap \end{Bmatrix}$$

Aturan diatas berati bahwa isi dari setiap nilai dalam matriks $H_{i,j}$ yang dibentuk dari sekuens A dan B merupakan nilai maksimum.

b. Langkah Balik (*Traceback*)

Langkah balik (*traceback*) yaitu langkah penelusuran jalan solusi yang dimulai dari $maks_{i,j}$ yaitu elemen matriks yang memiliki skor terbesar. Setelah $maks_{i,j}$ diperoleh, akan dimulai penelusuran dengan cara mencari nilai mana dalam matriks yang memberikan nilai $maks_{i,j}$ pada proses pembentukan matriks kesamaan, apakah berasal dari $H(i-1,j-1)$ (pergerakan diagonal), $H(i-1,j)$ (pergerakan atas), atau $H(i,j-1)$ (pergerakan kiri). Proses tersebut dilakukan berulang- ulang dan berhenti sampai ditemukan nilai nol. Selama proses penelusuran langkah balik, tidak semua pasangan akan menghasilkan *match* dan *mismatch* tetapi juga *gap* yang ditandai dengan penyisipan celah kosong pada salah satu atau kedua sekuens tersebut.

5. Perhitungan Bobot

Dari proses perbandingan tersebut, akan didapatkan informasi tentang kalimat yang dianggap memiliki kesamaan. Token atau kata yang sama pada tiap-tiap kalimat tersebut akan diakumulasi dan digunakan untuk menentukan seberapa besar kesamaan antara jawaban yang dibandingkan.

$$X = \frac{jumlah\ kata\ sama\ (match)}{jumlah\ kata\ bersih\ (clean)}\ x\ 100\%$$

Dimana :

x, merupakan prosentase kesamaan jawaban yang dibandingkan dan nilainya tidak kurang dari 10%, jika kurang dari itu dianggap sebagai jawaban yang saling berbeda. Jumlah kata sama (*match*), merupakan akumulasi kesamaan kata *match* dalam perbandingan antar kata dalam jawaban. Jumlah kata bersih (*clean*), merupakan jumlah akumulasi total kata dalam jawaban.

2.3 Pengertian Basis Data (*Database*)

Database dapat diartikan sebagai kumpulan data yang terdiri atas satu atau lebih tabel yang terintegrasi satu sama lain, dimana setiap user diberi wewenang (otoritas) untuk dapat mengakses (mengubah, menghapus, menganalisis, menambah, memperbaiki) data dalam tabel-tabel tersebut. Tabel-tabel tersebut berfungsi untuk menyimpan data dan merupakan suatu kumpulan data yang berhubungan dengan topik tertentu. Beberapa istilah yang harus diketahui pada saat bekerja dengan sebuah tabel database adalah :

a. *Field*: merupakan tempat dimana data atau informasi dalam kelompok yang sama atau sejenis dimasukkan. Field itu pada umumnya tersimpan dalam bentuk kolom vertikal pada tabel.
b. *Record*: merupakan data lengkap dalam jumlah tunggal yang biasanya tersimpan dalam bentuk baris secara horizontal pada tabel.

2.3.1 *Database Management System* (DBMS)

DBMS (*Database Management System*) atau dalam bahasa indonesia sering disebut sebagai sebagai Sistem Manajemen Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal dapat menyediakan fasilitas untuk mengelola akses data, mampu menangani integrasi data, dan mampu menangani backup data.

DBMS sudah mulai berkembang sejak tahun 1960-an. Kemudian sekitar tahun 1970-an mulai berkembang teknologi Relational DBMS yaitu DMBS berbasis relasional model. Relasional model pertama kali dikembangkan oleh Edgar J. Codd pada tahun 1970. Secara sederhana relasional model dapat dipahami sebagai suatu model yang memandang data sebagai sekumpulan tabel yang saling terkait.

Pada tahun 1980-an mulai berkembang *Object Oriented DBMS* (OODBMS). OODBMS berkembang seiring dengan perkembangan teknologi

pemrograman berorientasi objek. Secara umum dapat diartikan bahwa OODBMS merupakan DBMS yang memandang data sebagai suatu objek.

Terdapat beberapa macam DBMS versi komersial yang paling sering digunakan di dunia saat ini, diantaranya Oracle, Microsoft SQL Server, IBM DB2, dan Microsoft Access. Sedangkan DBMS versi *open source* yang cukup berkembang dan sering digunakan saat ini adalah diantaranya MySQL, PostgreSQL, Firebird, dan SQLite. Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS.

2.3.2 *Structured Query Language* (SQL)

SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada DBMS. Meskipun SQL diadopsi dan diacu sebagai bahasa standar oleh hampir sebagian besar RDBMS yang beredar saat ini, tetapi tidak semua standar yang tercantum dalam SQL diimplementasikan oleh seluruh DBMS tersebut. Sehingga terkadang terdapat perbedaan prilaku (hasil yang ditampilkan) oleh DBMS yang berbeda padahal query yang dimasukan sama.

2.4 ERD (*Entity Relationship Diagram*)

ERD (*Entity Relationship Diagram*) merupakan notasi grafis dalam pemodelan data konseptual, yang mendeskripsikan hubungan antara penyimpanan. ERD digunakan untuk memodelkan struktur data dan hubungan antar data, karena hal ini relatif kompleks. Dengan ERD, model dapat diuji dengan mengabaikan proses yang harus dilakukan. ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data, pada dasarnya ada 3 macam usur yang digunakan:

1. *Entity*: Adalah suara objek yang dapat diidentifikasikan dalam lingkungan pemakai, suatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat.
2. Attribut: *Entity* mempuyai elemen yang disebut atribut, dan berfungsi mendeskripsikan karakter *entity*.

3. Relasi: *Relationship* sebagaimana *entity* maka dalam hubunganpun harus dibedakan dalam hubungan atau bentuk hubungan antar *entity* dengan isi dari hubungan itu sendiri.
4. Garis Relasi: Garis yang merelasikan dari satu entitas ke entitas lain.

Relasi yang terjadi diantara dua himpunan entitas (misalnya A dan B) dalam satu basis data yaitu:

1. Satu ke satu (*One to one*): Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.
2. Satu ke banyak (*One to many*): Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.
3. Banyak ke banyak (*Many to many*): Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

2.4.1 Varian Entitas dan Relasi

Idealnya himpunan entitas yang terlibat dalam sebuah ERD adalah himpunan entitas kuat/bebas. Himpunan entitas demikian tidak memiliki ketergantungan dengan himpunan entitas lainnya. Namun demikian, dalam pembentukan ERD kita tidak selalu dapat melibatkan himpunan entitas seperti itu. Ada kalanya kita juga melibatkan himpunan entitas yang lemah atau merupakan bagian dari himpunan entitas lainnya.

2.4.1.1 Himpunan Entitas Lemah (*Weak Entity Sets*)

Berisi entitas-entitas yang kemunculannya tergantung pada eksistensinya dalam sebuah relasi terhadap entitas lain. Himpunan entitas demikian biasanya tidak memiliki atribut yang dapat berfungsi sebagai key, yang dapat menjamin keunikan entitas di dalamnya.

2.4.1.2 Sub Entitas (*Subtype Entities*)

Merupakan himpunan entitas yang beranggotakan entitas-entitas yang merupakan bagian dari himpunan entitas yang lebih superior (utama). Sub Entitas

ini merupakan hasil dekomposisi (spesialisasi) himpunan entitas berdasarkan pengelompokan tertentu.

2.5 Pemograman Terstruktur

Pemrograman Terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program [12]. Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis, dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami.

Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik/langkah tertentu, maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya/kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (*Loop*).

2.5.1 DFD (Data Flow Diagram)

DFD adalah merupakan model dari system untuk menggambarkan pembagian system ke modul yang lebih kecil [13]. Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai atau user yang kurang menguasai bidang komputer untuk mengerti system yang dikerjakan.

Dibawah ini terdapat istilah-istilah untuk DFD yaitu:

1. *Entity*: *Terminator* atau *Source* atau *Destination* atau dikenal juga dengan *External Entity*, berupa orang, organisasi atau sistem lain yang berada diluar batas sistem yang berinteraksi dengan sistem yang sedang dikembangkan.
2. *Proses* : Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, prosedur atau alat yang digunakan untuk mentrasformasikan data.
3. *Data Flow* : (Arus Data), data yang mengalir dengan arah tertentu dari asal ke tujuan. Data yang mengalir dapat berupa dokumen, surat atau bentuk lainnya.
4. *Data Store*: (Penyimpanan Data), digunakan untuk menyimpan dan mengambil data oleh proses. Data yang disimpan dapat berupa data yang terkomputerisasi maupun tidak terkomputerisasi.

2.5.2 Komponen Terminator/Entitas Luar

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya terminator dikenal dengan nama entitas luar (*external entity*).

Sesuatu yang berada diluar system, tapi memberikan data kedalam sistem atau memberikan data dari sistem. Disimbolkan dengan notasi kotak [13].

2.5.3 Komponen Proses

Komponen proses merupakan apa yang dikerjakan oleh *system*, proses dapat mengolah data atau aliran data masuk menjadi aliran data keluar. Proses berfungsi mentransformasikan satu atau beberapa data masukan menjadi satu atau beberapa data keluaran sesuai dengan spesifikasi data yang dinginkan [13].

2.5.4 Komponen Data Store

Komponen ini digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak, misalnya Mahasiswa. *Data store* ini biasanya berkaitan dengan penyimpanan-penyimpanan, seperti file atau database yang berkaitan dengan penyimpanan secara komputerisasi, misalnya file disket, file harddisk, file pita magnetik. *Data store* juga berkaitan dengan penyimpanan secara manual seperti buku alamat, file, folder, dan agenda.

Suatu *Data Store* dihubungkan dengan alur data hanya pada komponen proses, tidak dengan komponen DFD lainnya. Alur data yang menghubungkan *Data Store* dengan suatu proses mempunyai pengertian sebagai berikut :

a. Alur data dari Data Store yang berarti sebagai pembacaan atau pengaksesan satu paket tunggal data, lebih dari satu paket data, sebagian dari satu paket tunggal data, atau sebagian dari lebih dari satu paket data untuk suatu proses.
b. Alur data dari Data Store yang berarti sebagai peng-update-an data, seperti menambah satu paket data baru atau lebih, menghapus satu paket atau lebih, atau mengubah/memodifikasi satu paket data atau lebih.

2.5.5 Komponen *Data Flow*/ Alur Data

Arus data merupakan tempat mengalirnya informasi dan digambarkan dengan garis yang menghubungkan komponen dari sistem [13]. Arus data ditunjukan dengan arus panah dan garis diberi nama atas arus data yang mengalir. Arus data ini mengalir diantara proses, data store dan menunjukan arus data dari data yang berupa masukan untuk sistem atau hasil proses sistem.

2.6 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean [8].

2.6.1 Pengujian *Black box*

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* bukan merupakan alternatif dari teknik *white box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white box* [8].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

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

2.7 Bahasa Pemograman yang Digunakan

Bahasa pemrograman (*programming language*) adalah sebuah instruksi standar untuk memerintah komputer agar mempunyai fungsi tertentu. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang *programmer* dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi.

2.7.1 HTML (*Hyper Text Markup Languange*)

HTML adalah bahasa pemrograman standar yang digunakan untuk membuat sebuah halaman web, yang kemudian dapat diakses untuk menampilkan berbagai informasi di dalam sebuah penjelajah *web internet* (*Browser*).

Menurut Betha, Ir. Husni I. Pohan dan Ir. M. Eng dalam bukunya yang berjudul pemograman web dengan HTML, mengatakan bahwa:

HTML *kependekan dari Hyper Text Markup Language,Dokumen HTML adalah file teks murni yang dapat dibuat dengan editor teks sembarang* [14].

HTML dapat juga digunakan sebagai link link antara file-file dalam situs atau dalam komputer dengan menggunakan localhost, atau *link* yang menghubungkan antar situs dalam dunia internet. Supaya dapat menghasilkan tampilan wujud yang terintegerasi pemformatan *hyperteks* sederhana ditulis dalam berkas format ASCII sehingga menjadi halaman web dengan perintah-perintah HTML. HTML merupakan sebuah bahasa yang bermula bahasa yang sebelumnya banyak dipakai di dunia percetakan dan penerbirtan yang disebut *Standard Generalized Markup Language* (SGML).

Sekarang ini HTML merupakan standar Internet yang dikendalikan dan didefinisikan pemakaiannya oleh *World Wide Web Consortium* (W3C). Pada tahun 1989, HTML dibuat oleh kolaborasi Berners-lee Robert dengan Caillau TIM pada saat mereka bekerja di CERN (CERN merupakan lembaga penelitian fisika energi tinggi di Jenewa). Dari awal perkembangan sampai sekarang ini ada

beberapa macam versi HTML, ada HTML level 1.0, HTML 2.0, HTML 3.0, HTML 4.0 dan yang terbaru adalah HTML 5.0. Biasanya untuk mempercantik tampilan suatu *web* tidak hanya menggunakan HTML sajah, tapi mengkombinasikan juga dengan CSS dan JavaScript.

2.7.2 PHP (*Hypertext Processor*)

Menurut Sutarman dalam bukunya yang berjudul Membangun Aplikasi Web dengan PHP & MySQL mengatakan bahwa:

PHP adalah salah satu bahasa server side yang di desain khusus untuk aplikasi web [15].

Intinya PHP adalah bahasa pemograman *server side* yang didesain untuk aplikasi *web*. PHP ini programnya *open source* dibawah lisensi GNU atau GPU (*General Public License*) yang dapat didownload di situs http://www.php.net. dengan seiringnya perkembangan internet terdapat beberapa vesi PHP, seperti PHP/FI, PHP2, PHP3, PHP4, PHP5 dan juga PHP banyak mendukung basis data, seperti MySQL, Oracle, ODBC dan lainya.

PHP mendukung sebagai *database*, sehingga *database* yang kita buat dengan MySql ataupun jenis *database* lainya dapat diakses oleh PHP dan bisa untuk menampilkan isinya ataupun bisa memanipulasi datanya melalui halaman *web*.

2.7.3 CSS (*Cascading Style Sheets*)

Cascading Style Sheets atau CSS adalah suatu bahasa lembar gaya yang digunakan untuk mengatur tampilan suatu dokumen yang ditulis dalam bahasa markah. Penggunaan yang paling umum dari CSS adalah untuk memformat halaman web yang ditulis dengan HTML dan XHTML (*eXtensible Hyper Text Markup Language*). Walaupun demikian, bahasanya sendiri dapat dipergunakan untuk semua jenis dokumen XML (*eXtensible Markup Language*) termasuk SVG (*Scalable Vector Graphics*) dan XUL (*XML User Interface Language*). Spesifikasi CSS diatur oleh *World Wide Web Consortium* (W3C).

CSS biasanya digunakan oleh penulis maupun pembaca halaman web untuk menentukan warna, jenis huruf, tata letak, dan berbagai aspek tampilan dokumen. CSS digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan HTML atau bahasa markah lainnya) dengan presentasi dokumen (yang ditulis dengan CSS). Pemisahan ini dapat meningkatkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas serta pengulangan pada stuktur isi.

CSS memungkinkan halaman yang sama untuk ditampilkan dengan cara yang berbeda untuk metode presentasi yang berbeda, seperti melalui layar, cetak, suara (sewaktu dibacakan oleh penjelajah web basis-suara atau pembaca layar), dan juga alat pembaca *braille*. Halaman HTML atau XML yang sama juga dapat ditampilkan secara berbeda, baik dari segi gaya tampilan atau skema warna dengan menggunakan CSS.

2.7.4 JavaScript

Menurut Wahana dalam bukunya yang berjudul Menguasai Pemograman web dengan JavaScript 2009, mengatakan bahwa:

JavaScript merupakan bahasa yang berbentuk kumpulan skrip yang berfungsi untuk memberikan tampilan yang tampak lebih interaktif pada dokumen web [16].

JavaScript adalah bahasa *scripting* yang berfungsi untuk memberikan tampilan lebih menarik dan interaktif pada dokumen web. JavaScript ini adalah bahasa pemograman yang tidak perlu sebuah kompilator untuk mengeksekusinya dikarenakan JavaScript ini dapat memberikan kemampuan tambahan kepada bahasa pemograman HTML (*Hypertext Markup Language*), dengan mengijinkan pengeksekusian dengan *script–script* yang ada untuk mempercantik dokumen *web* di HTML. Pada dasarnya bahasa pemograman JavaScript ini pemasanganya terselip disebuah dokumen HTML.

2.8 Perangkat Lunak yang Digunakan

Perangkat lunak yang digunakan dalam pembangunan aplikasi essai ini adalah sebagai berikut:

2.8.1 *Adobe Dreamweaver* CS5

Adobe Dreamweaver adalah sebuah HTML *editor professional* untuk mendesain secara visual dan mengelola suatu web. Keunggulan *Dreamweaver* CS5 dibandingkan editor lainnya adalah memungkinkan pengguna berkreasi secara bebas dan cepat pada suatu lingkungan visual, tanpa menulis sebaris pun kode atau tag HTMLnya, dan setelah itu kita dapat menguji tampilan halaman web kita langsung di *browser* yang kita inginkan.

2.8.2 XAMPP

XAMPP merupakan paket PHP yang berbasis *open source* yang dikembangkan oleh sebuah komunitas *Open Source*. Dengan menggunakan XAMPP tidak dibingungkan dengan penginstalan program-program lain, karena semua kebutuhan terlah tersedia oleh XAMPP. Yang terdapat pada XAMPP di antaranya: Apache, MySQL, PHP, FileZilla FTP Server, phpMyAdmin dan lain-lain.

Fungsi XAMPP adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.