BAB II
LANDASAN TEORI

2.1 Akuaponik

Akuaponik adalah kombinasi akuakultur dan hidroponik yang bertujuan untuk memelihara ikan dan tanaman dalam satu sistem yang saling terhubung. Dalam sistem ini, limbah yang dihasilkan oleh ikan digunakan sebagai pupuk untuk tanaman, kemudian air yang dialirkan dengan sistem resirkulasi dari media pemeliharaan ikan dibersihkan oleh tanaman sehingga dapat digunakan kembali oleh ikan (Wahab et al. 2010). Interaksi antara ikan dan tanaman menghasilkan lingkungan yang ideal untuk tumbuh sehingga lebih produktif dari metode tradisional (Rakocy et al. 1997).

Penelitian tentang akuaponik dimulai oleh Universitas Virgin Island (UVI) sejak tahun 1971, penelitian berawal dari sulitnya memelihara ikan air tawar dan sayuran di pulau Semiarid, Australia. Hasil dari penelitian tersebut kemudian digunakan sebagai dasar pada sistem akuaponik untuk tujuan komersil, namun upaya pengembangan sistem ini masih mengalami banyak kendala, barulah pada tahun 1980-an sistem akuaponik mulai berkembang luas (Rakocy et al. 1997). Sampai tahun 1980-an, seluruh usaha dalam menggabungkan akuakultur dan hidroponik tidak semuanya berhasil, namun beragam inovasi yang dilakukan telah mengubah teknologi akuaponik menjadi salah sistem untuk memproduksi bahan makanan (Diver, 2006). Karena akuaponik hemat energi, mencegah keluarnya limbah ke lingkungan, menghasilkan pupuk organik untuk tanaman (lebih baik dari bahan kimia), menggunakan kembali air limbah melalui biofiltrasi dan menjamin produksi bahan makanan melalui multi-kultur, membuat akuaponik pantas dikatakan salah satu model panutan untuk green technology (Wahap et al. 2010).

Pada sistem akuaponik, aliran air kaya nutrisi dari media pemeliharan ikan digunakan untuk menyuburkan tanaman hidroponik. Hal ini baik untuk ikan karena akar tanaman dan rhizobakter mengambil nutrisi dari air. Nutrisi yang berasal dari feses, urin dan sisa pakan ikan adalah kontaminan yang menyebabkan

meningkatnya kandungan racun pada media pemeliharaan, tetapi air limbah ini juga menyediakan pupuk cair untuk menumbuhkan tanaman secara hidroponik. Sebaliknya, media hidroponik berfungsi sebagai biofilter, yang akan menyerap amonia, nitrat, nitrit dan fosfor sehingga air yang sudah bersih dapat di alirkan kembali ke media pemeliharaan (Diver, 2006). Bakteri nitrifikasi yang terdapat pada media hidroponik memiliki peran penting dalam siklus nutrisi, tanpa mikroorganisme ini seluruh sistem tidak akan berjalan. Amonia dan nitrit bersifat racun bagi ikan, tetapi nitrat lebih aman dan merupakan bentuk dari nitrogen yang dianjurkan untuk pertumbuhan tanaman seperti buah-buahan dan sayuran (Rakocy et al. 2006).

Kelebihan akuaponik dari sistem lainnya (ECOLIFE, 2011) :

1. Sistem akuaponik berjalan dengan prinsip zero enviromental impact. Akuaponik dapat menghasilkan ikan berkualitas baik dan tanaman organik tanpa pupuk buatan, pestisida maupun herbisida.
2. Sistem akuaponik memanfaatkan air dengan bijak. Sistem ini menggunakan 90% lebih sedikit air daripada menanam tanaman dengan cara konvensional dan menggunakan air 97% lebih sedikit dari sistem akuakultur biasa.
3. Sistem akuaponik serbaguna dan mudah beradaptasi. Sistem ini dapat dibangun dengan segala ukuran dan cocok untuk berbagai tempat.

Sebagian besar ikan air tawar yang tahan terhadap padat tebar tinggi akan tumbuh dengan baik pada sistem akuaponik (Rackocy et al. 2006). Beberapa jenis ikan yang telah dibudidayakan menggunkan sistem akuaponik adalah lele (Catfish), rainbow trout, mas (Common carp), koi, mas koki dan baramundi (Asian sea bass). Tanaman yang digunakan dalam sistem akuaponik berupa tanaman sayur (bayam, kemangi, kangkung) dan tanaman buah (tomat, mentimun, paprika). Media tanam yang digunakan dalam sistem akuaponik sama dengan cara bertanam hidroponik, yaitu dengan menggunakan batu apung, pasir, sabut kelapa, batu kerikil dan nutrient film (ECOLIFE, 2011).

2.2 Rekayasa Perangkat Lunak

2.2.1 UML *(Unified Modeling Language)*

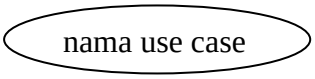
UML merupakan salah satu salah satu pemodelan perangkat lunak yang saat ini paling banyak digunakan. UML adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, seta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rossa, 2013).

2.2.1.1 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan oleh siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

Tabel 2.1 Simbol *Use Case Diagram*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
Aktor / *actor*  nama aktor	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

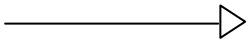
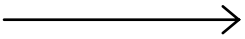
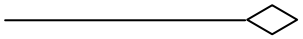
Asosiasi / *association* ———————	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor. Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu. Simbol ini mirip *inheritance* pada pemrograman berorientasi objek.
Ekstensi / *extend* <<*extend*>>	Hubungan *generalisasi* dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan /*include* /*uses* <<*include*>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

2.2.1.2 *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Tabel 2.2 Simbol *Class Diagram*

Simbol	**Deskripsi**
Kelas nama_kelas +atribut +operas()	Kelas pada struktur sistem.
Antarmuka/ *interface* ◯ nama_interface	Sama dengan konsep interface dalam pemrograman berorientas objek.
Asosiasi / *association* ———————	Relasi antarkelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*.
Asosiasi berarah/ *directed association* ——————→	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas lain, asosiasi biasanya juga disertai dengan *multiplicity*.

Generalisasi / *generalization* 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus).
Kebergantungan / *dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agresiasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

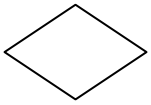
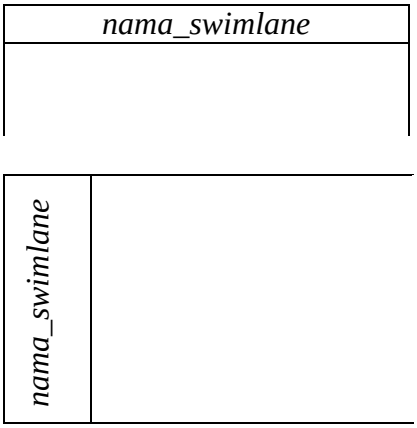
2.2.1.3 *Activity Diagram*

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. urutan pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. rancangan menu yang ditampilkan pada perangkat lunak.

Tabel 2.3 Simbol *Activity Diagram*

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

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

2.2.2 Data Flow Diagram

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *Bubble chart*, *Bubble diagram*, model proses, diagram alur kerja, atau model fungsi (Pressman, 2007).

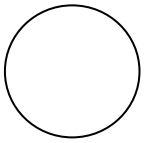
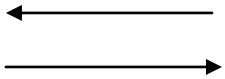
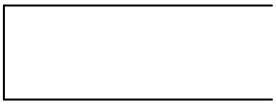
Pada saat informasi mengalir melalui perangkat lunak, dimodifikasi oleh suatu deretan transformasi. Diagram alir data/ DFD (*data flow diagram*) adalah sebuah teknis grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari *input* menjadi *output* (Pressman, 2007).

Berikut ini adalah aturan-aturan pembuatan (DFD) :

1. Didalam *Data Flow Diagram* (DFD) tidak boleh menghubungkan antara *enternal entity* dengan *entity* lainnya secara langsung.
2. Didalam *Data Flow Diagram* (DFD) tidak boleh menghubungkan *data store* yang satu dengan *data store* yang lainnya secara langsung.
3. Didalam *Data Flow Diagram* (DFD) tidak boleh menghubungkan *data store* dengan *enternal entity* secara langsung.
4. Setiap proses harus memiliki data *flow* yang masuk dan juga data *flow* yang keluar.

Berikut adalah simbol-simbol DFD :

Tabel 2.4 Simbol-simbol *Data Flow Diagram* (DFD)

Simbol	**Keterangan**
	*External Entity* Simbol ini digunakan untuk menggambarkan asal atau tujuan data
	Proses Simbol ini menujukan proses pengolahan atau tranformasi data
	*Data Flow* Simbol ini digunakan untuk menggambarkan aliran data yang berjalan
	*Data Store* Simbol ini menggambarkan *data Flow* yang sudah disimpan atau diarsipkan

Jenis-jenis DFD (*Data Flow Diagram*) :

1. *Context Diagram* (CD)

   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
b. Batas Sistem adalah batas antara “daerah kepentingan sistem”
c. Lingkungan Sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut
d. *Interface* adalah aliran yang menghubungkan sebuah sistem dengan linkungan sistem tersebut

2. Diagram Level *n* / *Data Flow Diagram Levelled*

   Dalam diagram n DFD dapat digunakan untuk menggambarkan diagram fisik maupun diagram diagram logis. Dimana Diagram Level *n* merupakan hasil pengembangan dari *Context Diagram* ke dalam komponen yang lebih detail tersebut disebut dengan *top-down partitioning*. Jika kita melakukan pengembangan dengan benar, kita akan mendapatkan DFD-DFD yang seimbang.

   a. 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.

   b. DFD Logis

      Adalah 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.

2.2.3 Bagan Alir (*flowchart*)

Untuk menggambarkan dan menganalisis sistem yang berjalan dan sistem yang diajukan, penulis memerlukan sebuah bagan alir (*flowchart*) agar lebih mudah dalam menunjukan sebuah proses atau prosedur secar logika.

Bagan alir (*flowchart*) adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam proses atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan dokumentasi.

Ada tiga macam bagan alir, yaitu sebagai berikut (Jogiyanto,2005):

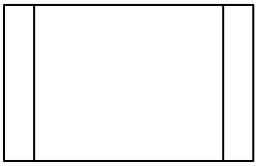
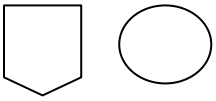
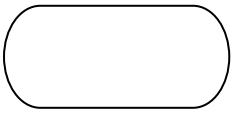
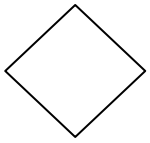
1. Bagan alir sistem *(system flowchart*).

   Bagan alir system merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem yang menjelaskan tentang urutan-urutan dari prosedur-prosedur yang ada di dalam sistem. Bagan alir system menunjukkan apa yang dikerjakan di sistem. (Jogiyanto,2005).
2. Bagan alir dokumen (*document flowchart*).

   Bagan alir dokumen (document flowchart) atau disebut juga bagan alir formulir (*from flowchart*) atau (*paperwork flowchart)* merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya. Bagan alir dokumen ini memakai simbol-simbolnya yang sama dengan yang digunakan di dalam bagan alir sistem.
3. Bagan alir program (*program flowchart).*

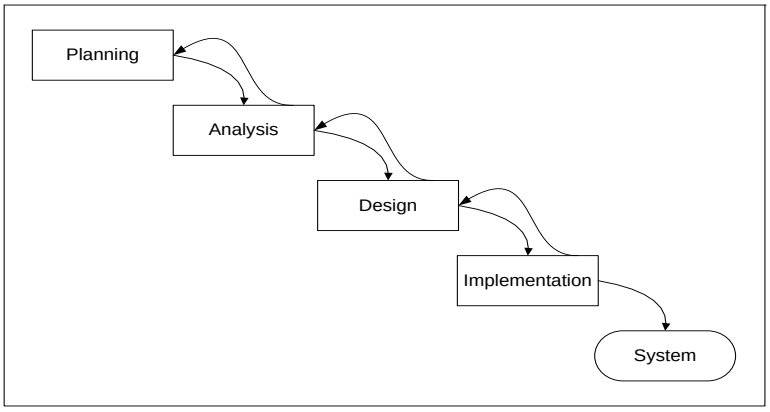
   Bagan alir program (program flowchart) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem. Bagan alir terdiri dari simbol-simbol yang memiliki fungsi yang akan dikerjakan. Bagan alir program merupakan alat bantu yang digunakan untuk merancang program yang diperlukan pada suatu aplikasi, bagan tersebut menampilkan aliran program secara global. Bagan alir program dibuat dengan menggunakan simbol-simbol.

Tabel 2.5 Simbol-Simbol Bagan Alir Program (Jogiyanto, 2005).

Simbol	Keterangan
Proses Program 	Menunjukkan proses program yang dilakukan
Proses Terdefinisi 	Menunjukkan proses terdefinisi yang rinciannya dijelaskan dalam bentuk flowchart di halaman lain.
Penghubung 	Menunjukkan penghubung ke halaman yang masih sama atau ke halaman lain.
Input/Output 	Menunjukkan input/output program.
Terminator 	Menunjukkan awal/akhir dari bagan alir program
Keputusan 	Menunjukkan percabangan/kondisi dari alur program.

2.2.4 Metode Pengembangan Perangkat Lunak Menggunakan Metode *System Development Life Cycle* model *Waterfall*

Pada metode penelitian ini dilakukan rekayasa perangkat lunak yang digunakan adalah model *Waterfall* seperti pada gambar berikut ini:



(Sumber : Alan Dennis, Barbara H Wixom. 2003)

Gambar 2.1 Metode Pengembangan Model *Waterfall*

Keterangan:

1. *Planning (Perencanaan)*

   Tahap perencanaan merupakan proses penting untuk mengetahui mengapa sistem harus dibuat dan menentukan bagaimana cara membangun sistem tersebut. Langkah pertama dari proses tersebut adalah dengan mengidentifikasi peluang apakah dapat memberikan kemungkinan biaya rendah tetapi menghasilkan keuntungan.

2. *Analysis (Analisis)*

   Analisis sistem dilakukan untuk memberikan jawaban pertanyaan siapa yang akan menggunakan sistem. Apa yang akan dilakukan oleh sistem, dimana dan kapan sistem tersebut digunakan. Pada tahap ini pembuat sistem akan melakukan observasi dan pengamatan terhadap sistem yang lama, kemudian mengidentifikasi, memanfaatkan dan mengembangkan peluang, dan membangun konsep untuk sebuah sistem baru.

3. *Design (perancangan)*

   Tahap perancangan dilakukan untuk menetapkan bagaimana sistem akan dioperasikan. Hal ini berkaitan dengan menentukan perangkat keras, perangkat lunak, jaringan, tampilan program, *form* dan laporan yang akan dipakai. Selain itu perlu juga menspesifikasi program, database dan file yang dibutuhkan.

4. *Implementation*

   Merupakan tahap berikutnya untuk menerjemahkan data atau pemecahan masalah yang telah dirancang ke dalam bahasa pemrograman komputer yang telah ditentukan. Semua tahap ini desain perangkat lunak sebagai sebuah program lengkap atau unit program.

5. *System*

   Tahapan ini, merupakan hasil sistem yang telah dibuat dalam bentuk perangkat lunak yang telah dipasang dan digunakan, termasuk didalamnya proses pemeliharaan dan perbaikan kesalahan. Perangkat lunak yang telah selesai dibuat dapat mengalami perubahan-perubahan atau penambahan sesuai dengan permintaan *user* atau perubahan sistem.

2.3 Basis Data

Database adalah kumpulan file-file yang mempunyai kaitan antara satu file dangan file yang lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu. (Kristanto: 2003).

Dengan memahami pengertian di atas, maka istilah basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol (*controlled redudancy*), data disimpan dengan cara-cara tertentu sehingga mudah untuk digunakan atau ditampilkan kembali, data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakanya, data disimpan sedemikian rupa sehingga proses penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Sutanta, 2004).

Istilah-istilah yang digunakan dalam basis data:

1) *File*

   Merupakan kumpulan dari atribut *record-record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda-beda dalam data *value*-nya.

2) *Record*

   Merupakan kumpulan dari elemen-elemen yang saling berhubungan atau berkaitan menginformasikan tentang *entry* secara lengkap.

3) *Field*

   Merupakan sekumpulan tanda-tanda yang berbentuk kesatuan tersendiri, merupakan bagian terkecil dari *record* dan bentuknya unik dijadikan *field* kunci yang dapat mewakili *record*-nya.

4) Entity

   Merupakan tempat kejadian atau konsep yang informasikan direkam.

Sutanta (2004), mengatakan bahwa perancangan basis data dan model data secara umum dapat dibagi menjadi beberapa kelompok, yaitu :

1. Model data berbasis objek
2. Model data berbasis *record*
3. Model data fisik
4. Model data konseptual

2.4 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL adalah Relational Database Management System (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (General Public License). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh

dijadikan produk turunan yang bersifat closed source atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis (Sidik, 2005).

2.5 HTML (Hyper Text Markup Language)

HTML merupakan kepanjangan dari *Hyper Text Markup Language* adalah suatu bahasa yang digunakan untuk membuat halaman-halaman hypertext (hypertext page) pada internet. Dengan konsep hypertext ini, untuk membaca suatu dokumen anda tidak harus melakukannya secara urut, baris demi baris, atau halaman demi halaman. Tetapi anda tidak dapat dengan mudah melompat dari satu topik ke topic lainnya yang anda sukai, seperti halnya jika anda melakukan pada online Help dari suatu aplikasi Windows. HTML dirancang untuk digunakan tanpa tergantung pada suatu platform tertentu (platform independent) (Kadir, 2006).