BAB II

LANDASAN TEORI

2.1. Konsep Dasar Web

Menurut Arief (2006:5) “ Web adalah fasilitas yang paling sering digunakan dan diakses setiap orang di internet ”. Web sudah berkembang sedemikian pesat dewasa ini. Banyak sekali web baru bermunculan di internet. Perkembangan web tersebut menarik minat setiap orang untuk mempelajari bagaimana membuatnya. Membuat web tidaklah susah. Sebuah web sederhana dan informatif dapat dibuat dengan ccepat menggunakan HTML, CSS, dan JapaScript.

A. *Website*

Interconnected Network atau yang lebih popular dengan sebutan *internet* menurut Yuhefizar (2009:2) adalah “Keseluruhan halaman-halaman web yang terdapat dalam sebuah domain yang mengandung informasi. Sebuah website biasanya dibangun atas banyak halaman web yang saling berhubungan. Hubungan antara satu halaman web dengan halaman web lainnya disebut dengan hyperlink, sedangkan teks yang dijadikan media penghubung disebut hypertext”.

Istilah-istilah yang umum di kenal di *internet* antara lain:

1. WWW(*World Wide Web*).

   World Wide Web merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung internet.

2. *Browser.*

Browser adalah *software* yang digunakan untuk menampilkan informasi dari *server web*.

3. *Universal Resource Locator*(*URL*).

   Universal Resource Locator adalah konsep nama file standar yang diperluas dengan jaringannya.

4. *Hypertext Markup Language* (*HTML*).

   HTML merupakan singkatan dari *Hypertext Markup Language.* artinya bahasa ini adalah bahasa markup untuk memformat konten halaman *web*. Atau dengan kata lain, bahasa untuk mengatur bagaimana penampilan dan pemformatan konten di *web*. Menurut arief (2009:2) " HTML digunakan untuk membuat halaman web. Sebuah file dokumen yang ditulis dalam format HTML akan dibaca dan diterjemahkan oleh web browser (misal Internet Explorer) untuk kemudian disajikan dalam bentuk web ".

5. *Domain Name Service*(*DNS*).

   Domain menurut Hidayat (2010:9) adalah " alamat unik yang digunakan untuk mengidentifikasi sebuah *website*, atau dengan kata *domain* adalah alamat yang digunakan untuk mencari dan menentukan sebuah *website* pada dunia *internet*, sedangkan sistem pengalamatan berbasis *domain* di kenal sebagai *Domain Name Service(DNS) ".*

B. Bahasa Pemrograman

1. *Javascipt*

Menurut Brooks (2008:3) *Javascript* adalah " bahasa pemrograman yang diinterpretasikan bukan dikompile, diadopsi dari bahasa *C atau C++* yang dikembangkan menjadi bahasa pemrograman *web client-side* ". *Javascript* didesain untuk bekerja sama dengan *HTML* membuat *web page* yang interaktif. Menurut McFarland (2008:1), *Javascript* adalah " bahasa pemrograman yang digabungkan dengan *HTML* untuk membuat halaman *web* yang beranimasi, interaktif dan memiliki *visual effect* yang dinamis ".

Javascript merupakan bahasa pemrograman yang cukup mudah dikuasai dan memiliki banyak fungsi yang dapat digunakan untuk meningkatkan efek *visual* dari halaman *web*. Kode dari *javascript* harus diapit oleh *tag*, diawali dengan *tag* <*script language="javascript"*> dan diakhiri dengan *tag*</*script*>.

2. CSS (*Cascading Style Sheet*)

Menurut Juju (2008:95) *CSS* adalah " bahasa *stylesheet* yang digunakan untuk mengatur tampilan dokumen. *CSS* memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda ".

Dengan *CSS*, tampilan *website* akan lebih cantik dan konsisten. Ada dua cara untuk menuliskan kode *CSS.* Pertama secara *internal*, yaitu menuliskan langsung diantara *tag HTML/XHTML*. Kedua secara *eksternal*, yaitu kode *CSS* disimpan dalam *file* yang terpisah kemudian dipanggil saat halaman *web* dibuka. *CSS* sendiri

merupakan sebuah teknologi *internet* yang direkomendasikan oleh *World Wide Web Consurtium* (W3C) dan diperkenalkan pada tahun 1996.

3. *Personal Home Page* (PHP)

   Menurut Hidayatullah (2014: 231) " PHP merupakan salah satu bahasa yang harus dikuasai. PHP *Hypertext Preprocessor* atau singkat dengan PHP ini adalah suatu bahasa *scripting* khususnya digunakan untuk *web delevopment*. Karena sifatnya yang *server side scripting*, maka untuk menjalankan PHP harus menggunakan *web server "*. PHP juga dapat diintegrasikan dengan *HTML, JavaScript, JQuery, Ajax*. Namun, pada umumnya PHP lebih banyak digunakan bersamaan dengan *file* bertipe *HTML*.

4. *Hyper Text Markup Language* (HTML)

   Menurut Sidik (2014: 9) " HTML kependekan dari *Hyper Text Markup Language.* Dokumen HTML adalah *file* teks murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai *web page "*. Dokumen HTML merupakan dokumen yang disajikan dalam *browser web surfer*. Dokumen ini umumnya berisi informasi atau *interface* aplikasi di dalam Internet. Ada dua cara untuk membuat sebuah *web page* : dengan HTML editor atau dengan editor teks biasa (misalnya notepad). Untuk latihan atau mencoba materi pada tulisan ini sebaiknya menggunakan notepad, setelah itu pada bagian mendekati akhir dapat menggunakan editor HTML, hal ini dimaksudkan agar anda memahami dan terbiasa secara primitif membuat web.

C. Basis Data *(Database*)

Pada umumnya aplikasi berbasis komputer yang digunakan diberbagai institusi menggunakan *database*. Kadir (2013:136), memberikan batasan bahwa: "*database* merupakan suatu bentuk pengelolaan data yang ditujukan agar pengaksesan terhadap data dapat dilakukan dengan mudah". Sistem ysng ditujukan untuk menangani database biasa disebut DBMS (*database menegement system*), Dengan menggunakan DBMS, pemakai dapat melakukan hal- hal dengan *mudah* seperti menambahkan data, menghapus data, mengubah data, mencari data, menampilakan data dengan kriteria tertentu ataupun pengurutkan data.

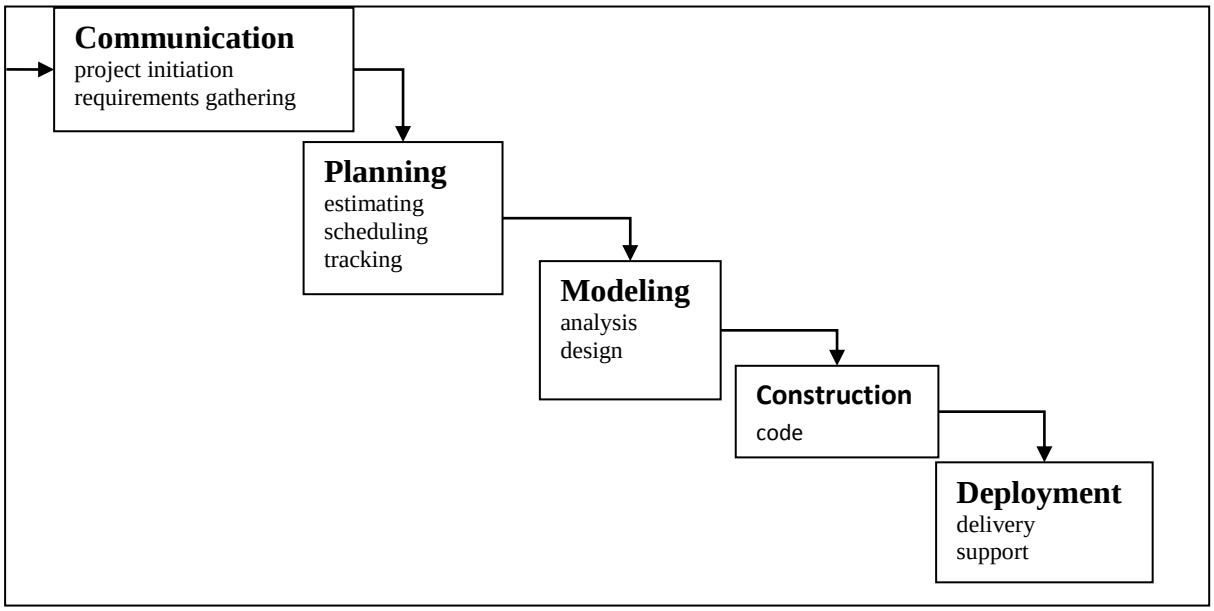
Salah satu *database* yang dipakai dalam pembuatan *website* Sistem Informasi Perpustakaan Berbasis *Web* pada Sma Taman Madya 1 Jakarta adalah database MySql. Menurut Kadir (2013:15), "*MySql* adalah sebuah database server yang berfungsi untuk menangani database". Dengan menggunakan *MySql*, kita bisa menyimpan data dan kemudian data bisa diakses dengan cara mudah dan cepat. Selain itu, *MySql* tergolong sebagai *database* relasional karena pada model ini data dinyatakan dalam bentuk dua dimensi yang secara khusus dinamakan tabel.Tabel tersusun atas baris dan kolom.

D. Model Pegembangan Perangkat Lunak

Pengembangan *software waterfall* model pertama kali diperkenalkan oleh Winston Royce tahun 1970. *Waterfall* model merupakan model klasik yang sederhana dengan aliran sistem yang linier.*Output* dari setiap tahap merupakan input bagi tahap berikutnya. Model ini telah diperoleh dari proses rekayasa lainnya dan menawarkan cara pembuatan rekayasa perangkat lunak secara lebih nyata. Model ini

melibatkan tim SQA (*Software Quantity Assurance*) dengan 5 tahapan, dimana setiap tahapan selalu dilakukan verifikasi atau testing.

Menurut Pressman (2010:39) "model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*". Secara umum tahapan pada model *waterfall* dapat dilihat pada gambar berikut :



Sumber: Pressman (2010:39)

Gambar II.1

Pemodelan Waterfall

Berikut ini adalah fase-fase dalam pemodelan *waterfall*, diantaranya:

A. Analisa Kebutuhan *Software*

Mengumpulkan data secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan dibangun. Hal ini sangat penting mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database* dan lain-lainnya

(Pressman, 2010:39). Beberapa macam *kebutuhan adalah sebagai berikut:*

1. *User Requirement* **(Kebutuhan Pengguna)**

   Pernyataan tentang layanan yang disediakan sistem dan tentang batasan-batasan operasionalnya.Dilengkapi dengan gambar/diagram yang dapat dimengerti dengan mudah.

2. *System Requirement* **(Kebutuhan Sistem)**

   Sekumpulan layanan/kemampuan sistem dan batasan-batasannya yang ditulis secara detil. Ini bisa berlaku sebagai kontrak antara klien dan pembangun.

3. *Software Requirement Specification* **(Spesifikasi Kebutuhan Perangkat Lunak).**

   Gambaran abstrak dari rancangan perangkat lunak yang menjadi dasar bagi perancangan dan implementasi yang lebih detil.

B. Desain

Proses pencarian kebutuhan diintensikan dan difokuskan pada *software* untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang informasi dari *software*, misalnya fungsi yang dibutuhkan *user interface*. Dari dua aktivitas tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada *user*.

Proses *softwaredesign* untuk mengubah kebutuhan-kebutuhan di atas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum coding dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini

juga harus didokumentasikan sebagai konfigurasi dari *software*, Pressman (2010:40).

C. Code Generation

Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan.Program yang dibangun langsung diuji baik secara unit, Pressman (2010:40).

D. Testing

Untuk dapat mengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu kedalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing*), Pressman (2010:41).

Metode pengetesan dilakukan dengan cara sebagai berikut:

1. *Black box testing*

   Black box testing memperlakukan pengujian perangkat lunak sebagai "kotak hitam" tanpa pengetahuan tentang pelaksanaan internal.

2. *White box testing*

   White box testing adalah ketika penguji memiliki akses ke struktur data internal dan algoritma termasuk *source code*.

E. Support

Sesuatu yang dibuat harus diujicobakan. Demikian juga dengan *software* Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari

error, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu.Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut.Pengembangan diperlukan ketika adanya perubahan dari *eksternal* perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya, Pressman (2010:41).

Kelebihan dari model ini adalah selain karena pengaplikasian menggunakan model ini mudah, kelebihan dari model ini adalah ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar di awal proyek, maka *software engineering* (SE) dapat berjalan dengan baik dan tanpa masalah. Meskipun seringkali kebutuhan sistem tidak dapat didefinisikan se-eksplisit yang diinginkan dalam hal uang (lebih murah), usaha, dan waktu yang terbuang lebih sedikit jika dibandingkan problem yang muncul pada tahap-tahap selanjutnya.

Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya.

Masalah dengan *waterfall*:

1. Perubahan sulit dilakukan karena sifatnya kaku.

2. Karena sifatnya kaku, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bias ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bias memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
3. *Waterfall* pada umumnya digunakan untuk rekayasa sistem yang besar yaitu dengan proyek yang dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek.

2.2. Teori Pendukung

A. Struktur Navigasi

Menurut Sutopo (2007:6) "Dalam pengembangan *web*, terdapat beberapa model navigasi dasar, yang harus dikenal dengan baik oleh desainer, Karena setiap model navigasi dapat memberikan solusi untuk kebutuhan yang berbeda".

Struktur navigasi adalah struktur atau alur dari suatu program yang merupakan rancangan hubungan (rantai kerja) dari beberapa area yang berbeda dan dapat membantu mengorganisasikan seluruh elemen pembuatan *Website*. Menentukan struktur navigasi merupakan hal yang sebaiknya dilakukan sebelum membuat suatu *Website*. Ada empat macam bentuk dasar dari struktur navigasi yang biasa digunakan dalam proses pembuatan *Website*, yaitu :

a. Struktur Navigasi linier

Struktur navigasi linier hanya mempunyai satu rangkaian cerita yang berurut, yang menampilkan satu demi satu tampilan layar secara berurut menurut urutannya.

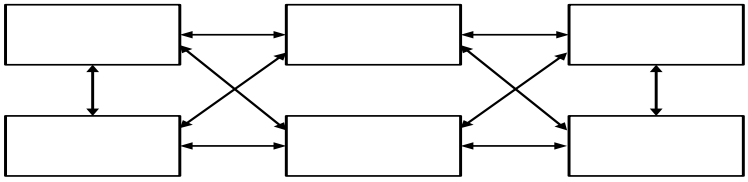
Tampilan yang dapat ditampilkan pada sruktur jenis ini adalah satu halaman sebelumnya atau satu halaman sesudahnya, tidak dapat dua halaman sebelumnya atau dua halaman sesudahnya.

Sumber:Sutopo (2007:6)

Gambar II.2.
Struktur Navigasi linier

b. Struktur Navigasi Non-linier

Struktur navigasi non-linier atau struktur tidak berurut merupakan pengembangandari struktur navigasi linier. Pada struktur ini diperkenankan membuat navigasi bercabang. Percabangan yang dibuat pada struktur nonlinier ini berbeda dengan percabangan pada struktur hirarki, karena pada percabangan nonlinier ini walaupun terdapat percabangan, tetapi tiap-tiap tampilan mempunyai kedudukan yang sama yaitu tidak ada *Master Page* dan *Slave Page*.

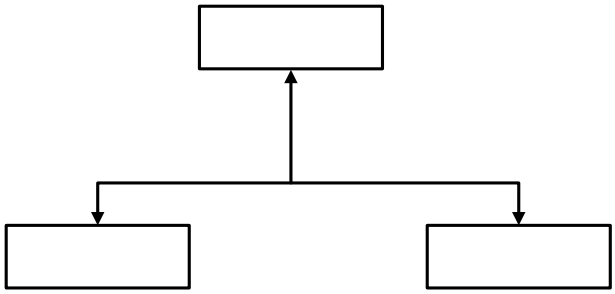


Sumber: Sutopo (2007:6)

Gambar II.3.
Struktur Navigasi Non-linier

c. Struktur Navigasi Hirarki

Struktur navigasi hirarki biasa disebut struktur bercabang, merupakan suatustruktur yang mengandalkan percabangan untuk menampilkan data berdasarkan kriteria tertentu. Tampilan pada menu pertama akan disebut sebagai *Master Page*(halaman utama pertama), halaman utama ini mempunyai halaman percabangan yang disebut *Slave Page* (halaman pendukung). Jika salah satu halaman pendukung dipilih atau diaktifkan, maka tampilan tersebut akan bernama *MasterPage* (halaman utama kedua), dan seterusnya. Pada struktur navigasi ini tidak diperkenankan adanya tampilan secara linier.



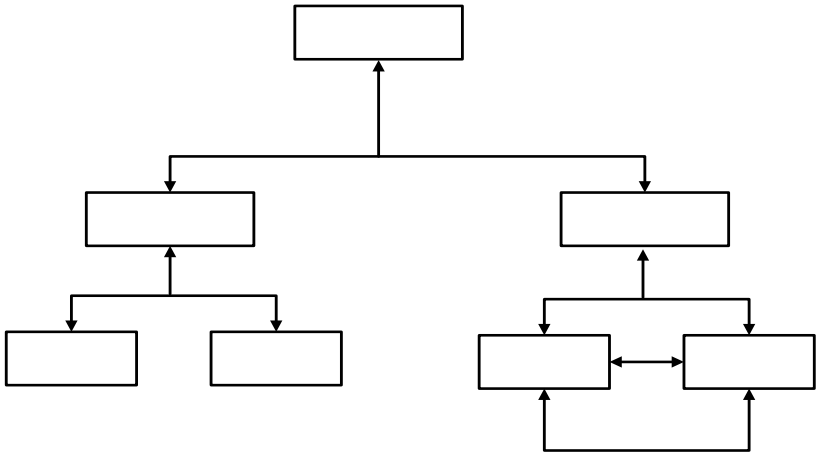
Sumber: Sutopo (2007:6)

Gambar II.4.
Struktur Navigasi Hirarki

d. Struktur Navigasi Campuran

Struktur navigasi campuran merupakan gabungan dari ketiga struktur sebelumnya yaitu linier, non-linier dan hirarki. Struktur navigasi ini juga biasa disebut dengan struktur navigasi bebas. Struktur navigasi ini banyak digunakan dalam

pembuatan *website* karena struktur ini dapat digunakan dalam pembuatan *website* sehingga dapat memberikan ke-interaksian yang lebih tinggi.



Sumber: Sutopo (2007:6)

Gambar II.5.
Struktur Navigasi Campuran/*Composite*

B. *Enterprise Relationship Diagram*

1. ERD (*Entity Relationship Diagram)*

Menurut Sukamto dan M. Shalahuddin (2011:49) "*Entity Relationship Diagram* dikembangkan berdasarkan teori himpunan dalam bidang matematika. *Entity Relationship Diagram* digunakan untuk pemodelan basis data relasional".

A. Elemen-elemen Diagram Hubungan Entitas

1. *Entity*

Entity adalah sesuatu yang ada didalam *system*, nyata maupun abstrak dimana data tersimpan. Entitas diberi nama dengan kata benda atau

dikelompokkan dalam empat jenis nama yaitu:orang, benda, lokasi, kejadian.

2. *Relationship*

   Relationship adalah hubungan alamiah yang terjadi antara entitas. Relationship diberi nama dengan kata kerja dasar sehingga memudahkan untuk melakukan pembacaan relasinya.

3. *Relationship Degree*

   Relationship degree adalah jumlah entitas yang berpartisipasi dalam satu *relationship*. Derajat *relationship* yang sering dipakai adalah :

   a. *Unary relationship* : Model *relationship* yang terjadi diantara *entity* yang berasal dari *entity* set yang sama.

   b. *Binary relationship* : Model *relationship* antara *instance-instance* dari suatu tipe entitas.

   c. *Ternary relationship* : *Relationship* antara instance-instance dari tiga tipe entitas secara serentak.

4. Atribut *Value*

   Atribut *value* adalah suatu *occurrence* tertentu dari sebuah atribut didalam suatu *entity* atau *relationship*.

   Ada dua jenis atribut, yaitu :

   a. *Identifier* (*key*) digunakan untuk menentukan suatu *entity* secara unik (*primary key*).

   b. *Descriptor* (*nonkey attribute*) digunakan untuk menspesifikasikan karakteristik dari suatu *entity* yang tidak unik.

5. Kardinalitas (*Cardinality*)

   Kardinalitas relasi menunjukkan jumlah maksimum tupel yang dapat berelasi dengan entitas pada entitas yang lain. Terdapat tiga macam kardinalitas relasi yaitu :

   a. *One to one* : Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama,, hanya mempunyai satu hubungan dengan entitas yang kedua dan sebaliknya.
   b. *One to many* : Tingkat hubungan satu ke banyak adalah sama dengan banyak ke satu, tergantung dari arah mana hubungan tersebut dilihat.
   c. *Many to many* : Tingkat hubungan kebanyakan terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya.

6. *Participation Constraint*

   Participation Constraint merupakan batasan yang menjelaskan apakah keberadaan suatu entity tergantung pada hubungannya dengan *entity* lain.

   Terdapat 2 macam *participation constraint*, yaitu :

   a. *Total participation* : Keberadaan suatu entity tergantung pada hubungannya dengan entity lain. Didalam diagram ERD digambarkan dengan dua garis penghubung antar entity dan relationship.
   b. *Partial participation* : Keberadaan suatu entity tidak bergantung pada hubungannya dengan entity lain. Didalam diagram ERD digambarkan dengan satu garis penghubung.

2. LRS (*Logical Record Structure)*

Menurut Sukamto dan M. Shalahuddin (2011:69) "LRS (*Logical Record Structure*) dibentuk dengan nomor dari tipe *record"*. Beberapa tipe *record* digambarkan dengan kotak empat persegi panjang dengan nama yang unik. LRS juga terdiri dari hubungan diantara tipe *record*. Salah satu metode pembuatan LRS yaitu dimulai dengan membuat ERD kemudian dikonversi ke dalam LRS.

C. Pengujian Web

Untuk dapat mengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu kedalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing*), Pressman (2010:41). *Black box testing* memperlakukan pengujian perangkat lunak sebagai "kotak hitam" tanpa pengetahuan tentang pelaksanaan internal.