BAB II

LANDASAN TEORI

2.1. Konsep Dasar *Web*

Dalam penulisan tugas akhir ini diperlukan teori yang relevan atau benar untuk mendukung kemudahan dalam mempelajari serta merancang *web* yang diharapkan dapat berjalan secara maksimal dan mudah dimengerti. Masing-masing bidang berlomba-lomba untuk memanfaatkan fasilitas yang ada, salah satu diantaranya adalah *internet*. Untuk dapat memanfaatkannya maka penulis memerlukan *web* yang mana adalah tempat untuk mewadahi dan melakukan banyak hal seperti menampilkan profil sebuah perusahaan hingga mempromosikan sebuah produk bahkan mengelola manajemen surat-menyurat khususnya pada arsip surat masuk - surat keluar, dan lain-lain. *Web* memiliki konsep dasar yang menjadi acuan dalam mempelajari suatu *web*. Pada bab ini penulis akan membahas beberapa teori, adapun teori yang akan dijelaskan untuk membantu dalam pembuatan *web*, sebagai berikut :

2.1.1. *Website*

Menurut Nurhadi (2017:8) mengemukakan bahwa :

> *Website* sering juga disebut *Web*, dapat diartikan suatu kumpulan-kumpulan halaman yang menampilkan berbagai macam informasi teks, data, gambar diam ataupun bergerak, data animasi, suara, video, maupun yang dinamis, yang dimana membentuk satu rangkaian bangunan yang saling berkaitan dimana masing-masing dihubungkan dengan jaringan halaman atau *hyperlink*.

Definisi lainnya dari *website* adalah kumpulan dari berbagai macam halaman situs, yang terangkum didalam sebuah *domain* atau juga *subdomain*, yang lebih

tempatnya berada di dalam WWW (*World Wide Web*) yang tentunya terdapat dalam *internet.*

1. *Web Browser*

Menurut Setiawan (2017:16) dalam bukunya "*Web Browser* ialah sebuah aplikasi yang digunakan untuk menjelajahi situs-situs di dunia maya".

Menurut Arief definisi dari "*Web browser* adalah aplikasi yang mampu menjalankan dokumen-dokumen *web* dengan cara diterjemahkan". Prosesnya dilakukan oleh komponen yang terdapat didalam aplikasi *browser* yang biasa disebut *Web Engine*. Semua dokumen *web* ditampilkan oleh *browser* dengan cara diterjemahkan. Beberapa jenis *browser* yang populer saat ini diantaranya adalah *Internet Explorer* yang diproduksi oleh *Microsoft, Mozilla Firefox, Opera,* dan *Safari* yang diproduksi oleh *Apple* (Fridayanthie & Tias Mahdiati, 2016).

2. *Web Server*

Menurut Fathansyah menerangkan bahwa pengertian *web server* adalah "*Server Web* (*Web Server*) merujuk pada perangkat keras (*server*) dan perangkat lunak yang menyediakan layanan akses kepada pengguna melalui protokol komunikasi HTTP ataupun variannya (seperti FTP dan HTTPS) atas berkas-berkas yang terdapat pada suatu URL ke pemakai" (Prayitno & Safitri, 2015).

3. *Internet*

Menurut Oneto dan Sugiarto mengatakan bahwa "*internet* adalah jaringan komputer". Ibarat jalan raya, *internet* dapat dilalui berbagai sarana transportasi, seperti bus, mobil dan motor yang memiliki kegunaan masing masing (Prayitno & Safitri, 2015).

Menurut Yuhefizar mendefinisikan "*Internet* adalah rangkaian jaringan komputer yang dapat diakses secara umum diseluruh dunia, yang mengirimkan data dalam bentuk paket data berdasarkan standar *Internet Protocol* (IP)" (Anwar & Irawan, 2017).

2.1.2. Bahasa Pemrograman

Bahasa pemrograman sangat membantu seorang *programmer* dalam menentukan data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan atau diteruskan, dan jenis langkah apa yang akan diambil dalam berbagai situasi.

1. HTML

Menurut (Setiawan (2017:16) "HTML atau *Hyper Text Markup Language* merupakan sebuah bahasa pemrograman terstruktur yang dikembangkan untuk membuat halaman *website* yang dapat diakses atau ditampilkan menggunakan *Web Browser"*. HTML sendiri secara resmi lahir pada tahun 1989 oleh Tim Berners Lee dan dikembangkan oleh *World Wide Web Consortium (W3C),* yang kemudian pada tahun 2004 dibentuklah *Web Hypertext Application Technology Working Group (WHATG)* yang hingga kini bertanggung jawab akan perkembangan bahasa HTML ini.

2. PHP

Menurut Setiawan (2017:54) "PHP sendiri sebenarnya merupakan singkatan dari "*Hypertext Preprocessor*", yang merupaakan sebuah bahasa scripting tingkat tinggi yang dipasang pada dokumen HTML". Sebagian besar sintaks dalam PHP mirip dengan bahasa C, Java dan Perl, namun PHP ada beberapa fungsi yang lebih

spesifik. Sedangkan tujuan utama dari penggunaan bahasa ini adalah untuk memungkinkan perancang *web* yang dinamins dan dapat bekerja secara otomatis.

3. CSS

Menurut Setiawan (2017:116) "CSS adalah kependekan dari *Cascading Style Sheet*, CSS merupakan salah satu kode pemrograman yang bertujuan untuk menghias dan mengatur gaya tampilan/*layout* halaman *web* supaya lebih elegan dan menarik".

CSS direkomendasikan oleh World Wide Web Consortium atau W3C pada tahun 1996. Awalnya CSS dikembangkan di SGML pada tahun 1970, dan terus berkembang hingga saat ini. CSS digunakan oleh *web programmer* dan juga *web designer* untuk menentukan warna, tata letak *font*, dan semua aspek lain dari presentasi dokumen di situs *web*.

4. Javascript

Menurut Setiawan (2017:194) mengemukakan bahwa "*JavaScript* adalah bahasa *scripting* yang populer di sebagian besar *browser*. *JavaScript* disisipkan pada halaman *web* menggunakan *tag* `<script>`.

Kegunaan *JavaScript* :

a. Untuk menambah interaktif suatu *website*

Beberapa hal tentang java script :

1) *Open Source* (semua orang dapat menggunakan secara gratis).
2) *JavaScript* merupakan bahasa *Scripting* yang ringan.
3) *JavaScript embedded* (disisipkan) dalam HTML
4) Dalam *JavaScript*, *script* akan langsung dieksekusi tanpa kompilasi.

Berikut contoh *script* pada *JavaScript* :

Kode Pembuatan :

```
<!DOCTYPE html>
<html>
<head>
      <!--CONTOH myBeta-->
      <title>JS</title>
</head>
<body>
<script type=”text/javascript”>
      document.write(“Hallo, Apa Kabar Bet?”);
</script>
</body>
</html>
```

2.1.3. Basis Data

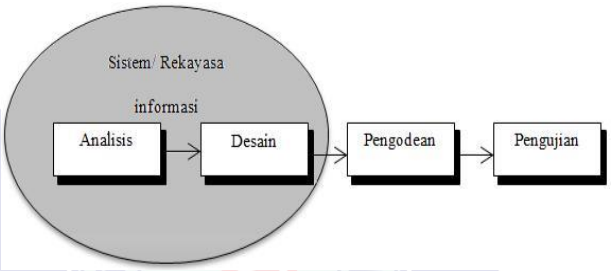
Menurut Lubis (2016:2-3) “Basis data (*database*) merupakan gabungan *file* data yang dibentuk dengan hubungan/relasi yang logis dan dapat diungkapkan dengan catatan serta bersifat independen”.

Arti lain dari sistem basis data adalah suatu sistem penyusunan dan pengelolaan *record-record* dengan menggunakan komputer, dengan tujuan untuk menyimpan atau merekam serta memelihara data secara lengkap pada sebuah organisasi/perusahaan.

Basis data akan menggunakan media penyimpanan (*storage*), yaitu berkaitan dengan setiap alat yang dapat menerima data yang dapat disimpan, dan dapat dipanggil kembali data itu pada waktu berikutnya atau setiap alat yang dapat menyimpan data.

2.1.4. Model Pengembangan Perangkat Lunak

Menurut Sukamto & Shalahuddin (2016:28) mengemukakan bahwa "Model SDLC *waterfall* disebut juga dengan model sekuensial *linier* (*sequential linear*) atau alur hidup klasik (*classic life cycle*)". Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support)* . Berikut adalah gambar model air terjun :



Sumber: Sukamto & Shalahuddin, (2016:29)

Gambar II.1
Ilustrasi Model *Waterfall*

1. Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*.

2. Desain

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya.

3. Pembuatan Kode Program

Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*support*) atau pemeliharaan (*maintenance*)

Sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2. *Tools* Program

Dalam pembuatan *website* ini juga tidak hanya memerlukan teori konsep dasar program saja, namun diperlukan teori pendukung untuk lebih memahami isi dari tulisan.

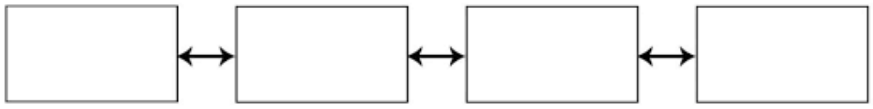
2.2.1. Struktur Navigasi

Menurut Andriansyah (2016:61) mengemukakan bahwa “Struktur navigasi dapat diartikan sebagai alur dari suatu program yang menggambarkan rancangan

hubungan antar area yang berbeda sehingga memudahkan proses pengorganisasian seluruh elemen *website*". Ada empat macam bentuk dasar struktur navigasi , adalah :

1. Struktur Navigasi *Linier*

Struktur navigasi linier merupakan struktur yang mempunyai satu rangkaian cerita berurutan. Struktur ini menampilkan satu demi satu tampilan *layer* secara berurutan menurut aturannya.

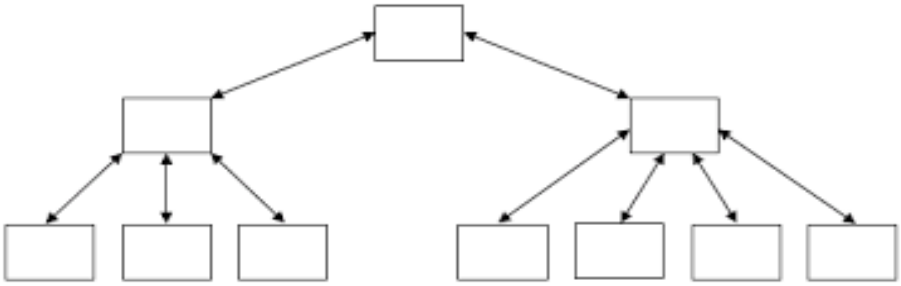


Sumber : Andriansyah (2016:62)

Gambar II.2
Struktur Navigasi *Linier*

2. Struktur Navigasi Hirarki

Struktur navigasi hirarki sering disebut struktur navigasi bercabang, yaitu merupakan suatu struktur yang mengandalkan percabangan untuk menampilkan data atau gambar pada *layer* dengan kriteria tertentu. Tampilan pada menu utama disebut *master page* (halaman utama satu), halaman tersebut mempunyai halaman percabangan yang disebut *slave page* (halaman pendukung) dan jika dipilih akan menjadi halaman kedua, begitu seterusnya.

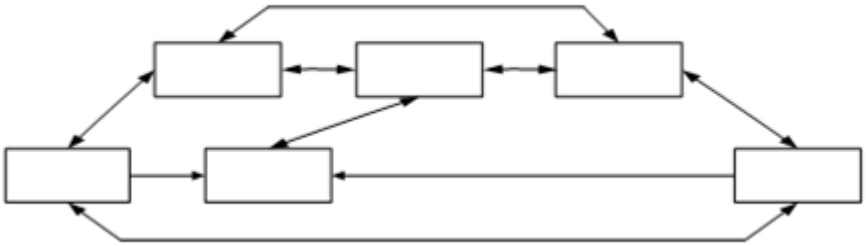


Sumber : Andriansyah (2016:62)

Gambar II.3
Struktur Navigasi Hirarki

3. Struktur Navigasi *Non Linier*

Struktur navigasi *non linier* (tidak terurut) merupakan pengembangan dari struktur navigasi *linier*, hanya saja pada struktur ini diperkenankan untuk membuat percabangan. Percabangan pada struktur *non linier* berbeda dengan percabangan pada struktur hirarki, pada struktur ini kedudukan semua *page* sama, sehingga tidak dikenal adanya *master* atau *slave page*.

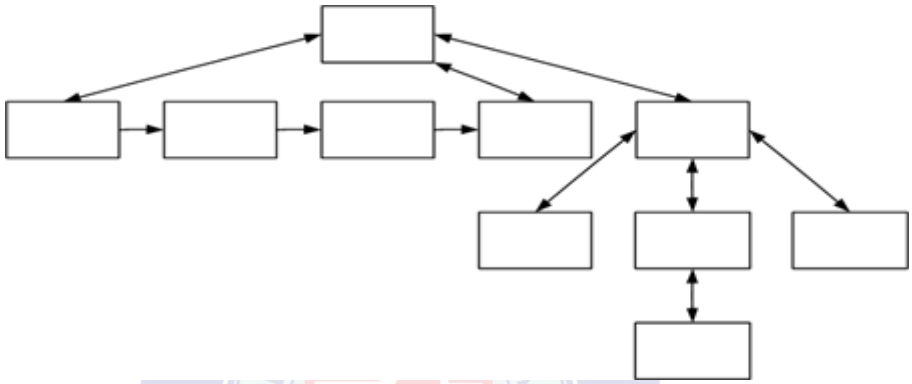


Sumber : Andriansyah (2016:62)

Gambar II.4
Struktur Navigasi *Non Linier*

4. Struktur Navigasi Campuran

Struktur navigasi campuran (*composite*) merupakan gabungan dari struktur sebelumnya dan disebut juga struktur navigasi bebas, maksudnya adalah jika suatu tampilan membutuhkan percabangan maka dibuat percabangan. Struktur ini paling banyak digunakan dalam pembuatan aplikasi multimedia.



Sumber : Andriansyah (2016:63)

Gambar II.5
Struktur Navigasi Campuran

2.2.2. *Entity Relationship Diagram*

Menurut Sukamto & Shalahuddin (2016:3) "ERD (*Entity Relationship Diagram*) digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS makan perancangan basis data tidak perlu menggunakan ERD".

Dengan diagram hubungan entitas ini kita dapat menguji model dengan mengabaikan proses yang harus dilakukan. Diagram hubungan entitas dapat membantu dalam menjawab persoalan tentang data yang diperlukan dan bagaimana data tersebut saling berhubungan. ERD terbagi atas empat komponen, yaitu:

1. Entitas (*Entity*)

Entitas merupakan benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer, penamaan entitas biasanya lebih kekata benda dan belum merupakan nama *table*. Entitas dinyatakan dengan simbol persegi panjang.

2. Atribut (*Atributte*)

Atribut (*Atributte*) merupakan *field* atau kolom data yang butuh di simpan dalam suatu entitas. Atribut digambarkan dalam bentuk lingkaran atau *elips*. Atribut yang menjadi kunci entitas atau *key* diberi garis bawah.

3. Relasi (*Relation*)

Relasi atau hubungan adalah kejadian atau transaksi yang terjadi diantara dua entitas yang keterangan nya perlu disimpan dalam basis data. Penghubung antara himpunan relasi dengan himpunan entitas dengan atribut dinyatakan dalam bentuk garis. Relasi biasanya diawali dengan kata kerja.

4. Kardinalitas (*Kardinality*)

Menunjukan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas Relasi :

a. Satu ke satu (*One to One*)

Setiap elemen dari entitas A berhubungan paling banyak dengan elemen pada entitas B. Demikian juga sebaliknya setiap elemen B berhubungan paling banyak satu elemen pada entitas A.

b. Satu ke banyak (*One to Many*)

Setiap elemen dari entitas A berhubungan dengan maksimal banyak elemen pada entitas B. Dan sebaliknya setiap elemen dari entitas B berhubungan dengan paling banyak satu elemen di entitas A.

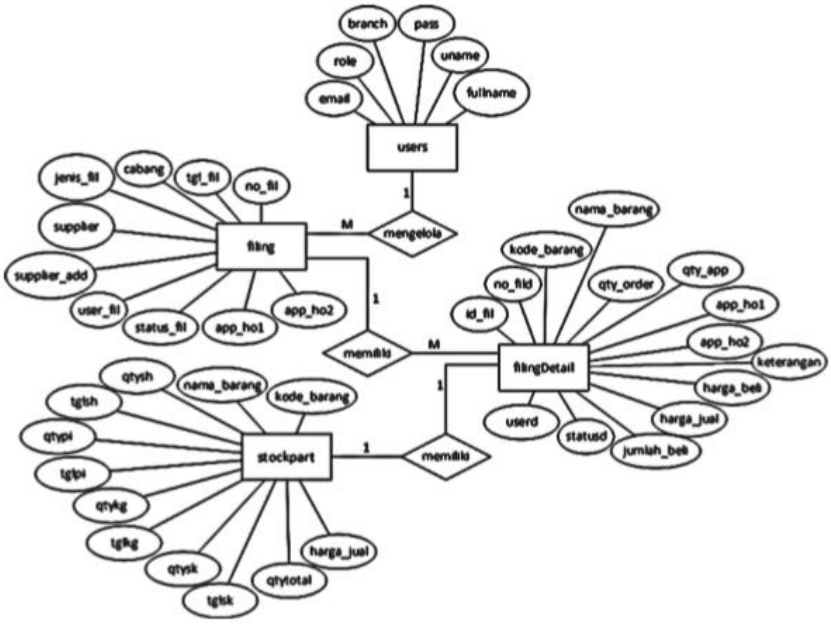
c. Banyak ke satu (*Many to One*)

Setiap elemen dari entitas A berhubungan paling banyak dengan satu elemen pada entitas B. Dan sebaliknya setiap elemen dari entitas B behubungan dengan maksimal banyak elemen di entitas A.

d. Banyak ke banyak (*Many to Many*)

Setiap elemen dari entitas A berhubungan maksimal banyak elemen pada entitas B demikian sebaliknya.

Berikut adalah contoh dari ERD (*Entity Relationship Diagram*) :

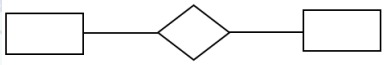
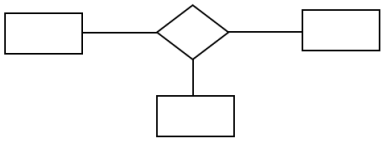
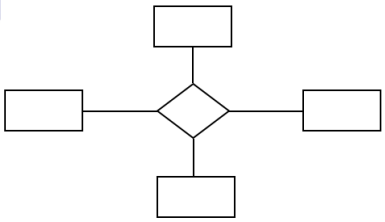


Sumber : (Anwar & Irawan, 2017)

Gambar II.6
ERD (*Entity Relationship Diagram*)

ERD (*Entity Relationship Diagram*) biasanya memiliki hubungan *binary* (satu relasi menghubungkan dua buah entitas. Beberapa metode perancangan ERD menoleransi hubungan relasi *ternary* (satu relasi menghubungkan tiga buah relasi) atau *N-ary* (satu relasi menghubungkan banyak entitas), tapi banyak metode perancangan ERD yang tidak mengizinkan hubungan *ternary* atau *N-ary*. Berikut adalah contoh bentuk hubungan relasi dalam ERD :

Tabel II.1
Bentuk Hubungan Relasi dalam ERD

Nama	Gambar
Binary	
Ternary	
N-ary	

Sumber : Sukamto & Shalahuddin (2018:52)

2.2.3. LRS (*Logical Record Structure*)

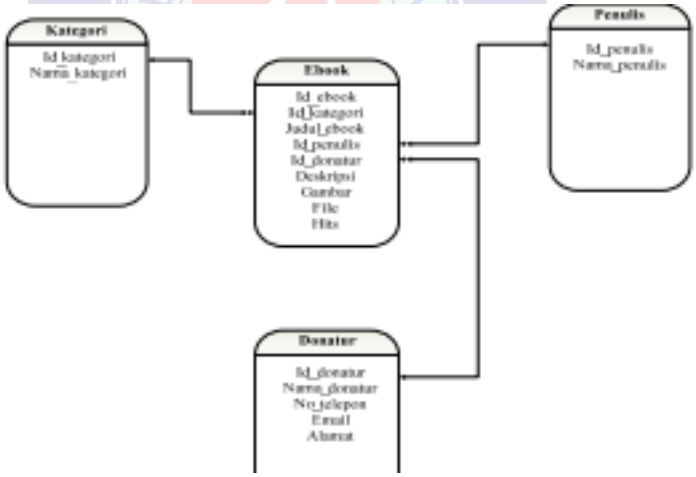
Menurut Andriansyah (2016:53) mengemukakan bahwa “LRS merupakan transformasi dari penggambaran ERD dalam bentuk yang lebih jelas dan mudah untuk dipahami. Penggambaran LRS hampir mirip dengan penggambaran

normalisasi *file*, hanya saja tidak digambarkan simbol *asterix* (*) sebagai simbol *primary key* (kunci utama) dan *foreign key* (kunci tamu)".

Berikut adalah cara membentuk skema database atau LRS (*Logical Record Strutured*) berdasarkan *Entity Relationship Diagram* :

1. Jika relasinya satu-ke-satu, maka *foreign key* diletakan pada salah satu dari dua entitas yang ada tau menyatukan kedua entitas tersebut.
2. Jika relasinya satu-ke-banyak, maka *foreign key* diletakan pada entitas *Many*.
3. Jika relasinya banyak-ke-banyak, maka dibua "*file* konektor" yang berisi dua *foreign key* yang berasal dari kedua entitas.

Berikut adalah contoh dari *Logical Record Structured* (LRS) :



Sumber : (Prayitno & Safitri, 2015)

Gambar II.7
Logical Record Structured (LRS)

2.2.4 Implementasi dan Pengujian *Web*

Menurut Sukamto & Shalahuddin (2018:275) "*Black-Box Testing* (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program". Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus uji yang dibuat adalah :

1. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar
2. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

Pengujian dengan metode *black-box testing* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program. Adapun beberapa kategori kesalahan yang diuji oleh *black-box testing*, diantaranya :

a. Fungsi-fungsi yang salah atau hilang
b. Kesalahan *interface*
c. Kesalahan dalam struktur data atau akses *database* eksternal
d. Kesalahan performa
e. Kesalahan inisialisasi dan terminal

BINA SARANA
INFORMATIKA
UNIVERSITAS