

BAB II

LANDASAN TEORI

2.1 Studi Pustaka

Pada bab 2 ini akan dijelaskan mengenai tinjauan pustaka yang menjadi acuan dalam melakukan penelitian mengenai “Perancangan Sistem Informasi Penyewaan Lapangan Futsal Berbasis Web pada Bywi Futsal”. Sistem informasi ini dirancang untuk mengelola data penyewaan lapangan dengan memberikan proses penyewaan, pembuatan bukti penyewaan, bukti pembayaran, dan pembuatan laporan pada *owner* (pemilik), menyiapkan dokumen-dokumen. (Ridwan, 2014)

Sudah banyak penelitian yang sebelumnya dilakukan mengenai sistem informasi penyewaan. Beberapa penelitian sebelumnya adalah sebagai berikut:

- a. Penelitian pertama yang telah dijalankan dengan judul ”Pengembangan Sistem Informasi Penyewaan Lapang Di Sudirman Futsal” yang diterbitkan oleh “Sekolah Tinggi Teknologi Garut” dalam “Jurnal Algoritma” dengan nomor ISSN 2302-7339. Penelitian ini membahas mengenai pembuatan sistem informasi penyewaan lapangan di Sudirman Futsal berbasis *desktop*. Dari pengembangan sistem informasi ini menghasilkan tampilan *menu* utama, *form* data lapang, *form* data jadwal lapang, *form* data penyewaan dan cetak kwitansi serta cetak laporan, *form* untuk menginput data penyewaan, *form* pelunasan, dan *form* untuk *edit user*. (Habil & Bunyamin, 2015)
- b. Penelitian kedua yang telah dijalankan dengan judul “Sistem Informasi Penyewaan Mobil Berbasis Web Di Jasa Karunia Tour And Travel” yang diterbitkan oleh “Sekolah Tinggi Teknologi Garut” dalam “Jurnal Algoritma” dengan nomor ISSN 2302-7339. Penelitian ini membahas mengenai sistem yang terkomputerisasi agar mempermudah mengelola data penyewa, data mobil yang tersedia, data mobil yang disewa pada rental mobil, pencatatan pemesanan yang cepat dan efektif serta memudahkan penyewa dengan adanya media promosi untuk penyewaan mobil. (Septavia, Gunadhi, & Kurniawati, 2015)
- c. Penelitian ketiga yang telah dijalankan dengan judul ”Rancang Bangun Aplikasi Penyewaan Dan Pengelolaan Data Alat Kemping Berbasis Desktop pada

Perusahaan Perorangan RZ Adventure” yang diterbitkan oleh “Sekolah Tinggi Teknologi Garut” dalam “Jurnal Algoritma” dengan nomor ISSN 2302-7339. Penelitian ini membahas mengenai pembuatan aplikasi yang digunakan untuk penyewaan dan pengelolaan data alat, memasukkan data penyewa, pengembalian dan penghapusan data barang secara efektif. Sedangkan bahasa pemrograman yang dipakai adalah *Java NetBeans* dan *database* menggunakan MySQL. Pembuatan Aplikasi Penyewaan di RZ Adventure mempermudah dalam aktifitas penyewaan dan proses pengembalian yang efektif serta efisien. (Frayoga As & Fitriani, 2016)

- d. Penelitian keempat yang telah dijalankan dengan judul ”Rancang Bangun Sistem Aplikasi Penyewaan Lapangan Futsal Berbasis Web” yang diterbitkan oleh “STMIK AMIKOM Yogyakarta” dalam jurnal “Seminar Nasional Teknologi Informasi dan Multimedia 2017” dengan nomor ISSN 2302-3805. Penelitian ini membahas mengenai pembuatan aplikasi penyewaan futsal berbasis *web* yang dibangun dengan menggunakan bahasa pemrograman PHP (*Personal Home Page*) dan untuk *database*-nya menggunakan Mysql dapat memudahkan *customer* dalam memesan lapangan futsal tanpa batasan jarak dan waktu, *customer* hanya datang ke tempat penyewaan pada saat jam dan waktu yang sudah ditentukan pada saat pemesanan. Pembuatan aplikasi penyewaan futsal dapat membantu pihak pengelola lapangan futsal dalam mengolah data penyewaan dan melihat data-data *customer* yang memesan lapangan secara otomatis tanpa harus menulis dan mencatat lagi. Selain itu pihak pengelola lapangan futsal dapat memperoleh laporan atau rekapan data pemesanan dengan cepat dan mudah. (Maimunah, Hariyansyah, & Jihadi, 2017)
- e. Penelitian kelima yang telah dijalankan dengan judul ”Pengembangan Aplikasi Penyewaan Lapangan Futsal Berbasis Android Studi Kasus: Hanggar Futsal Pancoran” yang diterbitkan oleh “Kalbiscentia” dalam “Jurnal Sains dan Teknologi” dengan nomor ISSN 2356-4393. Penelitian ini membahas mengenai pembuatan aplikasi penyewaan lapangan futsal berbasis *android* yang diharapkan dapat membuat pelanggan di Hanggar Futsal dalam melakukan penyewaan lapangan futsal dan mendapatkan jadwal nonton bersama pertandingan sepak bola dunia hanya dengan menggunakan *smartphone*

android yang dimiliki. Metode yang digunakan dalam pembuatan aplikasi penyewaan lapangan futsal di Hanggar Futsal adalah dengan metode penelitian perangkat lunak *Extreme Programming*. Hasil dari aplikasi penyewaan lapangan futsal, yaitu pelanggan dapat melakukan pemesanan, pembayaran, dan mengetahui informasi jadwal nonton bersama pertandingan sepak bola dengan menggunakan *smartphone android* yang dimiliki. (Nurchmawati & Lumba, 2017)

2.2 Sistem

Sistem merupakan sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama. (Pratama E. A., 2014)

Sistem juga dapat didefinisikan sebagai kumpulan/*group* dari subsistem/bagian/komponen apapun baik fisik ataupun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai satu tujuan tertentu. (Azhar, 2013)

Jadi dapat disimpulkan bahwa sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen elemen atau subsistem yang berinteraksi untuk mencapai suatu tujuan.

2.2.1 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu mempunyai komponen-komponen, batas sistem, lingkungan luar sistem, penghubung, masukan, keluaran, pengolah, dan sasaran atau tujuan. (Ladjamudin, 2013)

Adapun penjelasan dari masing-masing karakteristik sistem menurut tersebut adalah sebagai berikut:

a. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

b. Batasan Sistem

Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan dan menunjukkan ruang lingkup dari sistem tersebut.

c. Lingkungan Luar Sistem

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan juga merugikan.

d. Penghubung Sistem

Penghubung sistem merupakan media yang menghubungkan antara satu subsistem dengan subsistem yang lainnya. Melalui penghubung ini kemungkinan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

e. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan dan masukan sinyal *maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat berjalan. Sinyal *input* adalah energi yang diproses untuk mendapatkan keluaran dari sistem.

f. Keluaran Sistem

Keluaran sistem adalah energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran dapat merupakan masukan untuk subsistem yang lain.

g. Pengolahan Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan merubah masukan menjadi keluaran.

h. Sasaran Sistem

Suatu sistem mempunyai tujuan atau sasaran, kalau sistem tidak mempunyai sasaran maka sistem tidak akan ada. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya. Sasaran sangat berpengaruh pada masukan dan keluaran yang dihasilkan.

2.2.2 Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lainnya. Karena sistem memiliki sasaran yang berbeda untuk setiap kasus yang terjadi yang ada dalam sistem tersebut. Oleh karena itu sistem diklasifikasikan ke dalam berdasarkan kriteria tertentu.

Tabel 2.1 Pengklasifikasian Sistem

Kriteria	Klasifikasi	
Lingkungan	Sistem terbuka	Sistem tertutup
Asal pembuatnya	Buatan manusia	Buatan Allah/alam
Keberadaannya	Sistem berjalan	Sistem konsep
Kesulitan	Sulit/komplek	Sederhana
<i>Output</i> /kinerjanya	Dapat dipastikan	Tidak dapat dipastikan
Waktu keberadaannya	Sementara	Selamanya
Wujudnya	Abstrak	Ada secara fisik
Tingkatannya	Subsistem/Sistem	Supersistem
Fleksibilitas	Bisa beradaptasi	Tidak dapat beradaptasi

Adapun penjelasan lebih detail dan rinci dari tabel pengklasifikasian sistem di atas adalah sebagai berikut (Azhar, 2013):

a. Sistem Terbuka dan Tertutup

Sebuah sistem dikatakan terbuka bila aktivitas di dalam sistem tersebut dipengaruhi oleh lingkungannya. Sedangkan suatu sistem dikatakan tertutup bila aktivitas di dalam sistem tersebut tidak dipengaruhi oleh perubahan yang terjadi di lingkungannya.

b. Sistem Buatan Manusia dan Tuhan (Allah)

Suatu sistem bila diklasifikasikan berdasarkan pembuat sistem, bisa Tuhan (sistem alamiah) atau bisa juga manusia.

c. Sistem Berjalan dan Konseptual

Suatu sistem yang belum diterapkan disebut sebagai sistem konseptual. Bila kita merancang suatu sistem dan sistem tersebut belum diterapkan maka sistem tersebut hanyalah merupakan angan-angan atau masih berbentuk harapan yang

mungkin secara akal sehat (konsep) penyusunnya sistem sudah benar, dibuat berdasarkan kebutuhan dan situasi kondisi yang ada. Sistem berjalan adalah sistem yang digunakan saat ini. Sistem yang benar adalah sistem yang tepat guna dan dapat digunakan oleh pemakai sistem untuk meningkatkan pengendalian, efisiensi, dan kecepatan.

d. Sistem Sederhana dan Komplek

Dilihat dari tingkat kesulitannya, sebuah sistem dapat merupakan sebuah sistem yang sederhana atau sistem yang komplek. Sistem sederhana adalah sistem yang memiliki sedikit tingkatan dan subsistem. Sedangkan sistem komplek adalah sistem yang memiliki banyak tingkatan dan subsistem.

e. Kinerjanya Dapat dan Tidak Dapat Dipastikan

Suatu sistem dapat pula diklasifikasikan berdasarkan kepada kinerja yang dihasilkannya. Sebuah sistem yang dapat dipastikan artinya dapat ditentukan pada saat sistem akan dan sedang dibuat. Di lain pihak, sebuah sistem mungkin tidak dapat dipastikan yang artinya tidak dapat ditentukan dari awal tergantung kepada situasi yang dihadapi.

f. Sementara dan Selamanya

Suatu sistem mungkin digunakan untuk selamanya atau untuk periode waktu tertentu saja. Sementara artinya sistem hanya digunakan untuk periode waktu tertentu. Sebaliknya jika selamanya yang artinya sistem digunakan selamanya untuk waktu yang tidak ditentukan.

g. Ada Secara Fisik dan Abstrak/Non Fisik

Akhirnya sistem dapat dilihat dari wujudnya. Kendaraan bermotor bukan hanya merupakan sistem buatan manusia akan tetapi juga merupakan sistem yang ada secara fisik. Ada secara fisik artinya di sini dapat diraba. Perusahaan dan perguruan tinggi bukanlah organisasi yang dapat disentuh secara fisik. Kita dapat menyentuh foto, menunjuk apa yang ada di foto seperti mesin atau buku-buku, akan tetapi wujudnya adalah abstrak/non fisik. Abstrak artinya di sini tidak dapat diraba.

h. Sistem, Subsistem dan Supersistem

Berdasarkan tingkatannya/hierarki sebuah sistem bisa merupakan komponen dari sistem yang lebih besar. Sistem yang lebih kecil yang ada dalam sebuah

sistem disebut sebagai subsistem. Sedangkan sistem yang sangat besar dan kompleks adalah supersistem.

i. **Bisa Beradaptasi dan Tidak Bisa Beradaptasi**

Berdasarkan fleksibilitasnya kita dapat membedakan karakteristik suatu sistem tersebut dapat beradaptasi terhadap perubahan yang terjadi di lingkungannya atau tidak. Suatu sistem bisa beradaptasi artinya bisa menyesuaikan diri terhadap perubahan lingkungan, sebaliknya jika suatu sistem tidak bisa menyesuaikan diri terhadap perubahan lingkungan disebut tidak bisa beradaptasi.

2.3 Informasi

Informasi merupakan hasil pengolahan data dari satu atau berbagai sumber yang kemudian diolah, sehingga menghasilkan nilai, arti, dan manfaat. Jadi dapat disimpulkan bahwa informasi adalah data yang telah diolah menjadi sebuah bentuk yang lebih berguna dan bermanfaat bagi yang menerimanya. (Pratama E. A., 2014)

2.4 Sistem Informasi

Sistem informasi dapat didefinisikan sebagai berikut (Ladjamudin, 2013):

- a. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi.
- b. Sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan/atau untuk mengendalikan organisasi.
- c. Suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Sistem informasi adalah kumpulan sub-sub sistem baik fisik maupun non fisik yang saling berhubungan satu sama lain dan bekerjasama secara harmonis untuk mencapai satu tujuan yaitu mengolah data menjadi informasi yang berguna. (Azhar, 2013)

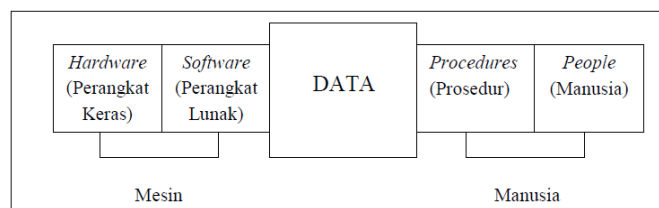
Sistem informasi merupakan empat gabungan bagian utama. Keempat bagian utama tersebut mencakup perangkat lunak (*software*), perangkat keras (*hardware*), infrastruktur, dan sumber daya manusia (SDM) yang terlatih. (Pratama E. A., 2014)

Dari beberapa uraian di atas dapat disimpulkan bahwa sistem informasi adalah suatu rangkaian komponen yang saling berkaitan untuk mengumpulkan, memproses serta menyimpan informasi yang mendukung fungsi operasi organisasi dalam pengambilan keputusan.

2.4.1 Komponen Sistem Informasi

Komponen sistem informasi terdiri dari beberapa bagian yang saling berintegrasi yang membentuk sebuah sistem. Terdapat lima komponen dalam sistem informasi dan kelima komponen tersebut dapat diklasifikasikan sebagai berikut (Ladjamudin, 2013):

- Hardware* dan *software* yang berfungsi sebagai mesin.
- People* dan *procedures* yang merupakan manusia dan tata cara menggunakan mesin.
- Data merupakan jembatan penghubung antara manusia dan mesin agar terjadi suatu proses pengolahan data.



Gambar 2.1 Komponen Sistem Informasi

2.5 Lapangan

Lapangan merupakan suatu bentuk ruang terbuka non hijau sebagai suatu pelataran dengan fungsi utama tempat dilangsungkannya aktivitas olahraga. Setiap jenis olahraga diperlukan sarana lapangan untuk tempat berlangsungnya aktivitas. Secara garis besar beberapa jenis olahraga yang membutuhkan sarana lapangan adalah Tenis, Futsal, Basket, dan Badminton. Untuk setiap jenis lapangan memiliki ukuran atau dimensi yang berbeda-beda. (Ramdani, 2016)

2.6 Futsal

Futsal adalah permainan olahraga yang dimainkan oleh dua tim yang berbeda, yang masing-masing beranggotakan lima orang pemain yang memainkan pertandingan dalam dua babak. Tujuannya adalah memasukkan bola ke gawang lawan, dengan memanipulasi bola dengan kaki. Selain lima pemain utama, setiap regu juga diizinkan memiliki pemain cadangan. Olahraga futsal diharapkan para pemain mampu mengasah kemampuan bermain bola, terutama teknik *dribbling* yang dapat diekspos dengan leluasa. Permainan ini juga memberikan manfaat bagi sistem ketahanan tubuh karena nyaris sepanjang permainan, seorang pemain akan berlari ke segala arah tanpa henti. (Mulyono, 2014)

Berdasarkan uraian di atas, futsal adalah permainan beregu yang dimainkan sangat cepat dan dinamis. Permainan ini dilakukan di lapangan yang sempit dengan tujuan dapat memasukkan bola ke gawang lawan dan mempertahankan gawang dari kemasukan bola.

2.7 Lapangan Futsal

Futsal merupakan versi mini dari olahraga sepak bola, namun menurut peraturan FIFA tahun 2010 ukuran lapangan futsal standar memiliki karakteristik yang berbeda dengan ukuran lapangan sepak bola, *indoorsoccer*, maupun *streetsoccer*. Lapangan futsal standar berbentuk persegi panjang dimana garis pembatas samping lapangan harus lebih panjang dari garis gawang, dengan ukuran panjang lapangan 38 – 42 m dan Lebar 18 – 25 m. Namun di Indonesia Ukuran Lapangan Futsal dengan Panjang 25 – 42 m dan Lebar: 15 – 25 m masih bisa digunakan dan tetap memenuhi syarat standar peraturan FIFA. (Ramdani, 2016)

2.8 Penyewaan

Penyewaan adalah suatu perjanjian atau kesepakatan di mana penyewa harus membayar atau memberikan imbalan dan manfaat dari benda atau barang yang dimiliki oleh pemilik barang yang dipinjamkan. Hukum dari penyewaan adalah mubah atau diperbolehkan. Contoh penyewaan dalam kehidupan kita sehari-hari misalkan seperti kontrak-mengontrak gedung kantor, kontrak rumah, sewa lahan

tanah untuk pertanian, menyewa/carter kendaraan, sewa-menyewa vcd dan dvd original, sewa lapangan futsal, dan lain-lain. (Bahtiar, 2011)

Dalam sewa-menyewa harus ada barang yang disewakan, penyewa, pemberi sewa, imbalan dan kesepakatan antara pemilik barang dan yang menyewa barang. Penyewa dalam mengembalikan barang atau aset yang disewa mengembalikan barang secara utuh seperti pertama kali tanpa kurang atau bertambah, kecuali ada kesepakatan lain yang disepakati saat sebelum barang berpindah tangan.

- a. Beberapa hal yang membuat sewa menyewa batal
 - 1) Barang yang disewa rusak
 - 2) Periode / masa pinjaman / kontrak sewa menyewa telah habis
 - 3) Barang yang disewakan cacat setelah berada di tangan penyewa.
- b. Manfaat Sewa-Menyewa
 - 1) Membantu orang lain yang tidak sanggup membeli barang
 - 2) Pemberi sewa mendapatkan manfaat dari yang penyewa

2.8.1 Faktor-Faktor Mempengaruhi Penyewaan

Faktor-faktor yang mempengaruhi penyewaan antara lain :

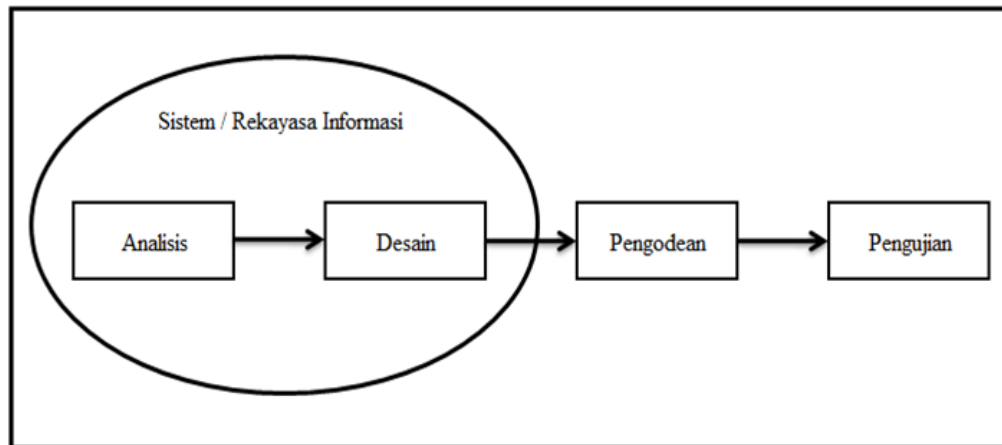
- a. Kondisi dan kemampuan penyewaan

Faktor yang perlu diperhatikan :

 - 1) Kondisi lapangan yang disewakan
 - 2) Harga sewa
 - 3) Syarat penyewaan, seperti kartu keluarga, katu tanda penduduk.
- b. Besarnya modal usaha yang dibutuhkan untuk pembuatan lapangan futsal.
- c. Bagaimana manajemen usaha lapangan futsal dengan baik.

2.9 Model Waterfall

Model air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun (*waterfall*) menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*). (Sukamto & Shalahuddin, 2016)



Gambar 2.2 Ilustrasi Model *Waterfall*

Tahapan-tahapan dari model *waterfall* adalah sebagai berikut (Sukamto & Shalahuddin, 2016):

a. Analisis kebutuhan perangkat lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

b. Desain

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

c. 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.

d. Pengujian

Pengujian fokus pada perangkat lunak secara dari segi logika 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.

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

Tidak menutup kemungkinan 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.

2.10 Basis Data (*Database*)

Basis data dapat didefinisikan sebagai himpunan kelompok data yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. (Hidayatullah & Kawistara, 2014)

Berikut ini adalah istilah-istilah yang digunakan dalam *database* (Enterprise, 2014):

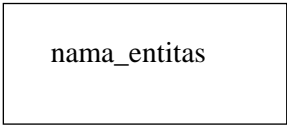
- a. *Database*: merupakan kumpulan tabel-tabel yang berisi data-data yang saling berkaitan.
- b. Tabel: merupakan matriks berisi data. Tabel dalam *database* terlihat seperti *spreadsheet* sederhana.
- c. Kolom: satu kolom (elemen data) mengandung data dengan satu jenis yang sama.
- d. Baris: sebuah baris (masukan atau rekaman data) merupakan sekumpulan data yang berhubungan.
- e. *Redudancy*: menyimpan data dua kali secara redundant untuk membuat sistem berjalan lebih cepat.
- f. *Primary Key*: *key* yang bersifat unik. Sebuah nilai *key* tidak dapat digunakan dua kali dalam satu tabel.
- g. *Foreign Key*: merupakan penghubung antara dua tabel.
- h. *Compound Key*: merupakan *key* yang terdiri dari beberapa kolom.
- i. Indeks: merupakan indeks dalam *database* yang menyerupai indeks pada buku.

- j. Integritas Referensial: digunakan untuk memastikan nilai *foreign* selalu mengacu pada suatu baris yang ada.

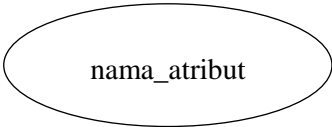
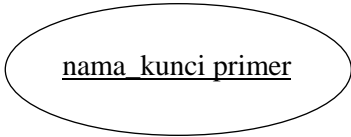
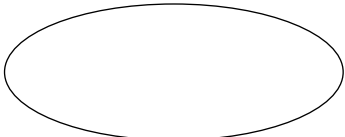
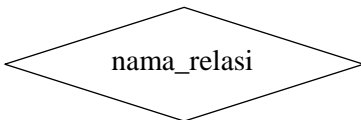
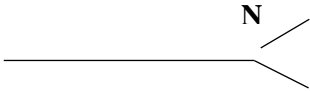
2.10.1 *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram (ERD) adalah sebuah diagram yang menggambarkan bagaimana struktur design database yang akan dibuat (Pratama A., 2017). *Entity Relationship Diagram (ERD)* merupakan bentuk paling awal dalam melakukan pemodelan basis data relasional. ERD 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 *ternary* atau *N-ary*. (Sukamto & Shalahuddin, 2016)

Tabel 2.2 Simbol-simbol *Entity Relationship Diagram (ERD)*

No.	Simbol	Keterangan
1.	<p>Entitas/<i>entity</i></p> 	Entitas merupakan data inti yang akan disimpan, bakal tabel pada basis data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer.

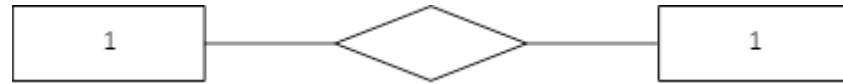
Lanjutan Tabel 2.2 Simbol-simbol *Entity Relationship Diagram* (ERD)

2.	<p>Atribut</p> 	<p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas.</p>
3.	<p>Atribut kunci primer</p> 	<p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses <i>record</i> yang diinginkan, biasanya berupa id. Kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).</p>
4.	<p>Atribut multivalai/<i>multivalue</i></p> 	<p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.</p>
5.	<p>Relasi</p> 	<p>Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja.</p>
6.	<p>Asosiasi/<i>association</i></p> 	<p>Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki <i>multiplicity</i> kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas.</p>

Relasi antara dua *file* atau tabel dapat dikategorikan menjadi tiga macam, antara lain:

- a. Relasi *one-to-one*

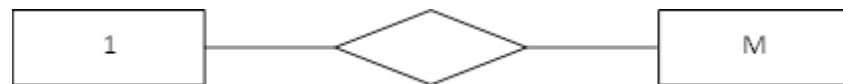
Yang berarti entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas B, dan begitu juga sebaliknya dengan satu entitas pada himpunan A.



Gambar 2.3 Relasi *One-to-One*

b. Relasi *one-to-many*

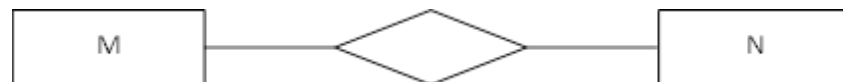
Hubungan satu ke banyak entitas atau atribut adalah jenis hubungan yang hanya dapat dilakukan satu entitas atau satu atribut dengan beberapa entitas atau atribut yang lainnya.



Gambar 2.4 Relasi *One-to-Many*

c. Relasi *many-to-many*

Hubungan banyak ke banyak entitas atau atribut adalah jenis hubungan yang hanya dapat dilakukan satu entitas atau satu atribut dengan beberapa entitas atau atribut yang lainnya dan satu entitas atau satu atribut dengan beberapa entitas atau atribut yang lainnya.



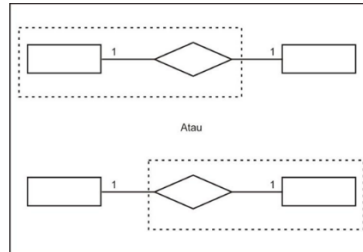
Gambar 2.5 Relasi *Many-to-Many*

2.10.2 Transformasi ERD ke LRS

Sebuah model sistem yang digambarkan dengan sebuah ERD akan mengikuti pola aturan pemodelan tertentu. Dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan-aturan berikut ini:

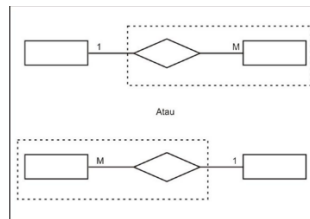
- a. Perhatikan kardinalitas karena sangat berpengaruh pada transformasi.

- b. Transformasi *one-to-one* (1 : 1). Pedoman relasinya adalah ke arah *weak entity*, dan ke entitas yang membutuhkan referensi atau ke entitas dengan jumlah atribut yang lebih sedikit.



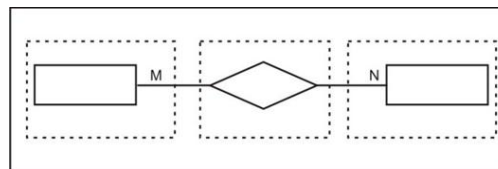
Gambar 2.6 Transformasi *One-to-One* (1 : 1)

- c. Transformasi *one-to-many* (1 : M). Pedoman relasinya adalah tidak perlu melihat jumlah atribut yang lebih sedikit, dan selalu digabung ke arah *many*.



Gambar 2.7 Transformasi *One-to-Many* (1 : M)

- d. Transformasi *many-to-many* (M : N). Relasinya berdiri sendiri atau membentuk tabel sendiri.



Gambar 2.8 Transformasi *Many-to-Many* (M : N)

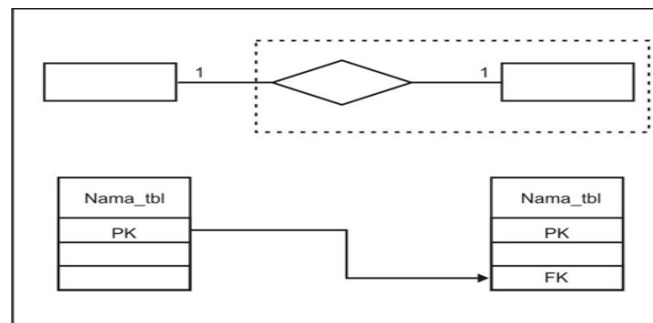
2.10.3 Logical Record Structure (LRS)

LRS adalah representasi dari struktur *record-record* pada *table-table* yang terbentuk dari hasil antar himpunan entitas. Setiap *table* harus memiliki paling sedikit satu *primary key*, dimana sebuah *primary key* merupakan bagian dari kelompok atribut yang memberikan nilai yang unik di dalam sebuah *table*.

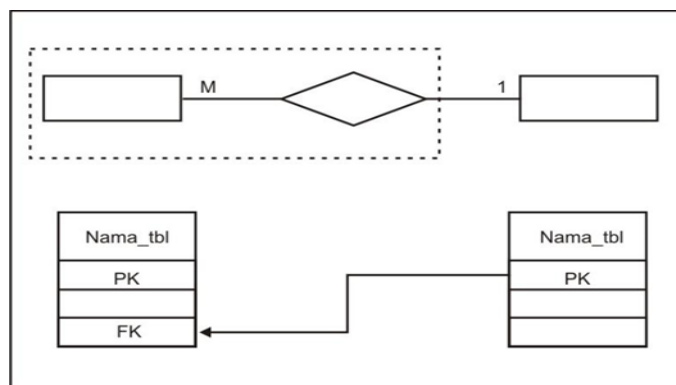
Pedoman membuat LRS, yaitu:

- Tiap entitas dan relasi (jika kardinalitas setiap entitas *many-to-many*) juga menjadi sebuah *Logical Record Structure* (LRS).
- Nama LRS menjadi nama tabel.
- Tiap satu atribut menjadi satu kolom.
- Nama atribut akan menjadi nama kolom.

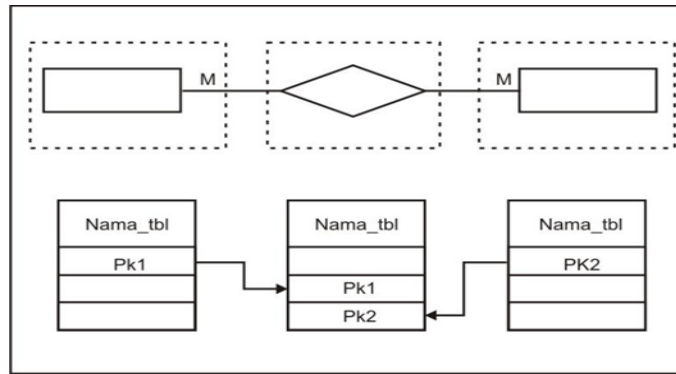
LRS ditentukan oleh relasi antar entitas, seperti berikut:



Gambar 2.9 LRS Transformasi 1:1



Gambar 2.10 LRS Transformasi 1:M



Gambar 2.11 LRS Transformasi M:N

2.11 *Unified Modeling Language (UML)*

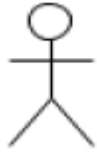
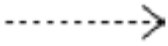

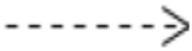
Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language (UML)*. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. (Sukamto & Shalahuddin, 2016)

Terdapat beberapa model diagram yang digunakan dalam *Unified Modeling Language (UML)*, diantaranya *use case diagram*, *activity diagram*, *sequence diagram*, dan *class diagram*.






2.11.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. (Sukamto & Shalahuddin, 2016)


Tabel 2.3 Simbol-simbol *Use Case Diagram*

No.	Gambar	Nama	Keterangan
1.		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2.		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya sebagai elemen yang tidak mandiri (<i>independent</i>).
3.		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / <i>ancestor</i>).
4.		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.

Lanjutan Tabel 2.3 Simbol-simbol *Use Case Diagram*

5.		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8.		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu <i>actor</i> .
9.		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).




Lanjutan Tabel 2.3 Simbol-simbol *Use Case Diagram*

10.		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.
-----	---	-------------	--

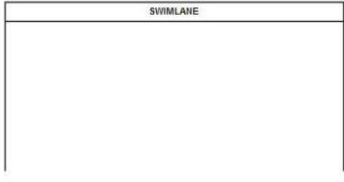


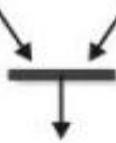
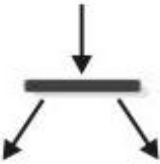
2.11.2 Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan adalah bahwa *activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. (Sukamto & Shalahuddin, 2016)

Tabel 2.4 Simbol-simbol *Activity Diagram*

No.	Simbol	Keterangan
1.		Start merupakan status awal sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Activity merupakan aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Decision merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

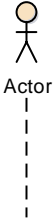
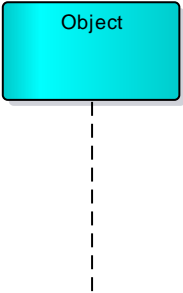

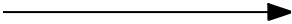
Lanjutan Tabel 2.4 Simbol-simbol *Activity Diagram*

4.		Swimlane digunakan untuk memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
5.		Finish merupakan status akhir sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6.		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
7.		Join digunakan untuk menunjukkan kegiatan yang digabungkan
8.		Fork digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel.



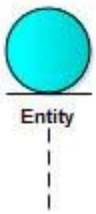
2.11.3 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case*. (Sukamto & Shalahuddin, 2016)

Tabel 2.5 Simbol-simbol *Sequence Diagram*

No.	Simbol	Keterangan
1.		Actor adalah pesan dari seseorang atau sistem lain yang bertukar informasi dengan sistem yang lainnya, kemudian <i>lifeline</i> berhenti atau mulai pada titik yang tepat.
2.		Object Life Line menunjukkan keberadaan dari sebuah objek terhadap waktu. Yaitu objek dibuat atau dihilangkan selama suatu periode waktu diagram ditampilkan, kemudian <i>lifeline</i> berhenti atau mulai pada titik yang tepat.
3.		Activation menampilkan periode waktu selama sebuah objek atau aktor melakukan aksi. Dalam <i>object lifeline</i> , <i>activation</i> berada diatas <i>lifeline</i> dalam bentuk kotak persegi panjang, bagian atas dari kotak merupakan inisialisasi waktu dimulainya suatu kegiatan dan yang dibawah merupakan akhir dari waktu.
4.		Message adalah komunikasi antar objek yang membawa informasi dan hasil pada sebuah aksi. <i>Message</i> menyampaikan dari <i>lifeline</i> sebuah objek kepada <i>lifeline</i> yang lain, kecuali pada kasus sebuah <i>message</i> dari objek kepada objek itu sendiri, atau dengan kata lain <i>message</i> dimulai dan berakhir pada <i>lifeline</i> yang sama.

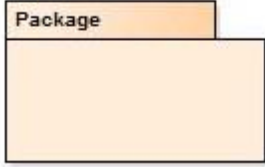
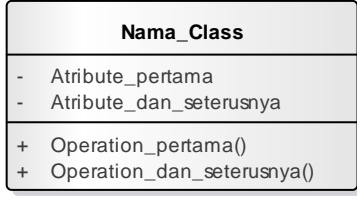




Lanjutan Tabel 2.6 Simbol-simbol *Sequence Diagram*

5.		<i>Boundary</i> digunakan untuk menggambarkan sebuah <i>form</i> .
6.		<i>Control</i> digunakan untuk menghubungkan <i>boundary</i> dengan tabel.
7.		<i>Entity</i> digunakan untuk menggambarkan hubungan kegiatan yang akan dilakukan.



2.11.4 Class Diagram

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. *Class diagram* dibuat agar pembuat program membuat kelas-kelas sesuai rancangan di dalam *class diagram* agar antara dokumentasi perancangan dan perangkat lunak sinkron. Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan *class diagram*. (Sukanto & Shalahuddin, 2016)

Tabel 2.7 Simbol-simbol *Class Diagram*

No.	Simbol	Keterangan
1.		Package merupakan sebuah bungkus dari satu atau lebih kelas.
2.		Class digambarkan dengan bentuk persegi panjang yang dibagi kedalam ruang-ruang terpisah yang terdiri dari nama <i>class</i> , atribut, dan operasi-operasinya.
3.		Associations adalah representasi / gambaran relasi statis diantara <i>class-class</i> .
4.		Interface sama dengan <i>interface</i> dalam pemrograman berorientasi objek.
5.		Associations berarah merupakan relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
6.		Generalisasi merupakan relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus).

Lanjutan Tabel 2.8 Simbol-simbol *Class Diagram*

7.		Depedency merupakan relasi antar kelas dengan makna kebergantungan antar kelas.
8.		Aggregation merupakan relasi antar kelas dengan makna semua-bagian (<i>whole-part</i>).

2.12 Perangkat Lunak Pendukung

Dalam perancangan sistem informasi, dibutuhkannya perangkat lunak pendukung agar perancangan sistem informasi dapat terlaksana.

2.12.1 *Hypertext Markup Language* (HTML)

HTML merupakan singkatan dari *Hypertext Markup Language*. Singkatan ini terdiri dari 3 komponen kata, yakni: *Hypertext*, *Markup* dan *Language*. Kata *Hypertext* dari HTML menekankan pengertian: teks yang lebih dari sekedar teks (*'hyper'-text*). Maksudnya selain berfungsi sebagai teks biasa, sebuah teks di dalam HTML juga bisa berfungsi sebagai penghubung ke halaman lain atau dikenal dengan istilah link. Tidak hanya teks saja yang bisa digunakan sebagai *link*, tetapi bisa berupa gambar. *Link* inilah yang menjadi inti dari HTML. Kata kedua dari singkatan HTML adalah *Markup*. *Markup* dapat diterjemahkan sebagai tanda atau penanda (bahasa inggris: *mark*). Di dalam HTML, kita akan menggunakan tanda-tanda khusus seperti `<p>`, `<a>`, atau ``. Tanda ini diperlukan untuk mengatur format dan membuat struktur halaman *web*. Bagian terakhir dari HTML adalah *Language*. Istilah *language* jika diterjemahkan berarti bahasa. Di sini HTML tidak menggunakan '*Programming Language*', tetapi hanya '*Language*' saja. Hal ini secara tidak langsung menyatakan bahwa HTML bukanlah sebuah bahasa pemrograman. HTML tidak memiliki struktur dasar seperti variabel, kondisi *if*, *function*, atau *class* seperti layaknya sebuah bahasa pemrograman komputer. (Pratama A. , 2018)

Merangkum penjelasan di atas, dapat disimpulkan bahwa HTML adalah sebuah bahasa khusus yang ditulis menggunakan tanda-tanda (*mark*) untuk membuat halaman *web*.

2.12.2 *Cascading Style Sheet (CSS)*

CSS merupakan singkatan dari *Cascading Style Sheet*. CSS digunakan untuk mengubah tampilan (*style*) dari halaman *web*. Sebagaimana yang kita ketahui, halaman web modern terdiri dari 3 komponen dasar: HTML untuk membuat struktur, CSS untuk tampilan, dan JavaScript untuk interaksi. Jika halaman *web* diibaratkan sebuah bangunan, CSS adalah tampilan luar dari bangunan tersebut, seperti warna dinding atau warna atap. Kerangka dasarnya dibuat dari HTML. Dengan demikian, kita bisa dengan mudah menukar warna dinding bangunan tanpa perlu mengubah struktur dasarnya. Begitu pula dengan halaman *web*. Menggunakan CSS, kita bisa mengubah tampilan *website* tanpa perlu menyentuh kode HTML. Apabila saat ini *website* kita memiliki warna mayoritas merah, minggu depan bisa menjadi biru hanya dengan menukar beberapa baris kode CSS. Kata *Cascade* dari kepanjangan CSS juga perlu kita bahas. Dalam bahasa Inggris, *cascade* berarti “air terjun kecil, riam, jeram, mengalir/berpancaran kebawah”. Dimana maknanya adalah: sesuatu yang mengalir dari atas ke bawah. Di dalam CSS, *style* atau aturan tampilan yang dibuat bisa saja saling menimpa satu sama lain, tergantung dari posisinya dan ke-spesifikan kode CSS tersebut. Sebagai contoh, jika pada baris pertama kode CSS kita membuat perintah untuk mengubah warna paragraf menjadi biru, di baris kedua kita bisa menulis kembali perintah yang sama, tetapi kali ini mengubah warna tersebut menjadi merah. (Pratama A. , 2016)

Sebagai kesimpulan, dalam pengertian sederhana CSS adalah format bahasa khusus yang digunakan untuk mengatur tampilan dari halaman *web*.

2.12.3 *PHP: Hypertext Preprocessor (PHP)*

PHP merupakan singkatan dari *PHP: Hypertext Preprocessor*. Singkatan ini disebut singkatan rekursif, yakni permainan kata dimana kepanjangannya juga terdiri dari singkatan PHP itu sendiri, yakni *PHP: Hypertext Preprocessor*. *Hypertext Preprocessor* bisa diterjemahkan sebagai ‘pemroses *hypertext*’, atau

‘pemroses HTML’. Jadi, PHP adalah bahasa pemrograman *web* yang digunakan untuk *men-generate* atau menghasilkan kode HTML. Sebenarnya PHP dapat melakukan lebih dari sekedar menghasilkan kode HTML. Kita bisa menggunakan PHP untuk pemrosesan *form*, mengakses *database*, *management session* dan *cookie*, membaca *file* teks, menangani *file upload*, membuat *file pdf*, membuat *file excel*, dan masih banyak lagi. Ini karena PHP adalah sebuah bahasa pemrograman *web server side* (*server side programming language*). Karena termasuk bahasa pemrograman berbasis *server*, kita harus menjalankan kode-kode PHP dari sebuah *server*. Dengan kata lain, kode PHP tidak bisa dijalankan tanpa *server*. (Pratama A. , 2016)

2.12.4 MariaDB

MariaDB adalah salah satu aplikasi RDBMS (*Relational Database Management System*). Pengertian sederhana RDBMS adalah aplikasi *database* yang menggunakan prinsip relasional. MariaDB juga bukan satu-satunya RDBMS, diantaranya yang banyak dikenal adalah: Oracle, Sybase, Microsoft Access, Microsoft SQL Server, dan PostgreSQL. MariaDB bersifat gratis dan *open source*, artinya setiap orang boleh menggunakan dan mengembangkan aplikasi ini. Namun walaupun gratis, MariaDB di *support* oleh ribuan programmer dari seluruh dunia, dan merupakan sebuah aplikasi RDBMS yang lengkap, cepat, dan reliabel. (Pratama A. , 2017)

2.12.5 XAMPP

XAMPP merupakan aplikasi yang mem-*bundle* paket *web server* + PHP + MariaDB ke dalam 1 kali proses instalasi. Dengan menginstall aplikasi paket ini, kita sudah mendapatkan 3 aplikasi lengkap (dan settingannya) sehingga sudah siap pakai. Nama aplikasi XAMPP terdiri dari paket yang ada di dalamnya. X (berarti *cross-platform*, maksudnya tersedia dalam berbagai sistem operasi), Apache Web Server, MariaDB Database Server, PHP dan Perl. Selain aplikasi ini, XAMPP juga menyertakan aplikasi pelengkap seperti phpMyAdmin, File Zilla FTP Server, Mercury Mail Server, dan lain-lain. XAMPP adalah salah satu aplikasi yang dikenal sebagai AMP Stack. AMP merupakan singkatan dari Apache Web Server,

MariaDB Database Server, dan PHP. Dalam pengembangan *web*, ketiga aplikasi ini sangat populer digunakan. Jika AMP Stack dijalankan di sistem operasi Windows, namanya menjadi WAMP (Windows-Apache-MariaDB-PHP). Jika dijalankan di OS Linux, menjadi LAMP (Linux-Apache-MariaDB-PHP). Untuk Mac OS dikenal sebagai MAMP (Mac-Apache-MariaDB-PHP). Selain XAMPP, aplikasi AMP Stack populer lain adalah WAMP Server, AMPPS, dan BitNami WAMPStack. (Pratama A., 2016)

2.13 Pengujian (*Testing*)

Sebuah perangkat lunak perlu dijaga kualitasnya bahwa kualitas bergantung kepada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan sebagai berikut (Sukamto & Shalahuddin, 2016):

- a. Agar dapat *survive* bertahan hidup di dunia bisnis perangkat lunak.
- b. Dapat bersaing dengan perangkat lunak yang lain.
- c. Penting untuk pemasaran global (*global marketing*).
- d. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran atau kegagalan produksi.
- e. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak sudah berada di tangan *user*. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan *bug*. Untuk menghindari banyaknya *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan. (Sukamto & Shalahuddin, 2016)

2.13.1 *Black-Box Testing* (Pengujian Kotak Hitam)

Black-box testing 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. (Sukamto & Shalahuddin, 2016)

Black-box testing 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 *black-box testing* harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses *login* maka kasus uji yang dibuat adalah:

- a. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
- b. 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.

2.13.2 White-Box Testing (Pengujian Kotak Putih)

White-box testing yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan. *White-box testing* dilakukan dengan memeriksa logika dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian dari standar pemrograman yang seharusnya. Pengujian terhadap dokumentasi yang dibuat juga harus dilakukan agar dokumentasi yang dibuat tetap konsisten dengan perangkat lunak yang dibuat. (Sukamto & Shalahuddin, 2016)

