BAB II
TINJAUAN PUSTAKA

2.1 Sertifikat

Sertifikat adalah tanda atau surat keterangan (pernyataan) tertulis atau tercetak dari orang yang berwenang yang dapat digunakan sebagai bukti kepemilikan atau suatu kejadian(Kamus Besar Bahasa Indonesia 2016).Dalam konteks kegiatan seperti seminar atau workshop, maka sertifikat dikeluarkan oleh suatu instansi atau panitia penyelenggara yang menerangkan bahwa nama orang yang disebut dalam sertifikat telah berpartisipasi pada kegiatan seminar atau *workshop* yang diselenggarakannya baik sebagai peserta maupun pembicara/narasumber.

2.2 *QR Code (Quick Response Code)*

QR Code, kependekan dari *Quick Response Code*, merupakan gambar dua dimensi yang memiliki kemampuan untuk menyimpan data. *QR Code* biasa digunakan untuk menyimpan data berupa teks, baik itu numerik, alfanumerik, maupun kode biner. *QR Code* banyak digunakan untuk keperluan komersial, khususnya di Jepang, biasanya berisi *link url* ke alamat tertentu atau sekedar teks berisi iklan, promosi, dan lain-lain (Ani et al. 2011).

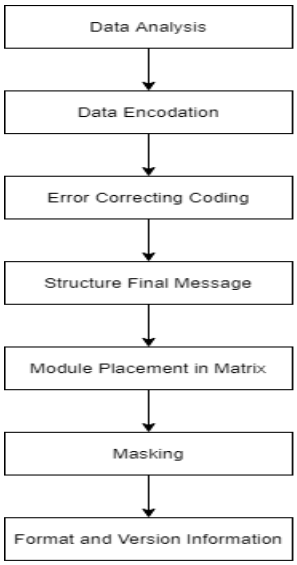
Untuk membaca sebuah pesan yang tersembunyi yang ada pada *QR Code* pengguna dapat menggunakan sebuah aplikasi *QR Code scanner* yang ada di *Play Store* bagi pengguna *Android* dan *App Store* untuk pengguna *iPhone* atau *device* (Febriyanto et al. 2019).

Gambar 2.1 Contoh *qr code*

QR Code mampu menyimpan semua jenis data, seperti data angka/numerik, alphanumerik, biner, kanji/kana. Selain itu *QR Code* memiliki tampilan yang lebih kecil dari pada *barcode*. Hal ini dikarenakan *QR Code* mampu menampung data

secara horizontal dan vertikal, jadi secara otomatis ukuran dari tampilannya gambar *QR Code* bisa hanya sepersepuluh dari ukuran sebuah *barcode*. Tidak hanya itu *QR Code* juga tahan terhadap kerusakan, sebab *QR Code* mampu memperbaiki kesalahan sampai dengan 30% tergantung dengan ukuran atau versinya. Oleh karena itu, walaupun sebagian simbol *QR Code* kotor ataupun rusak, data tetap dapat disimpan dan dibaca (Ardhianto and Wakhidah 2016).

Prosedur pembangkitan QR Code dari sebuah teks dapat dijelaskan dengan diagram alir pada gambar 5.2 berikut ini (Ani et al. 2011) :

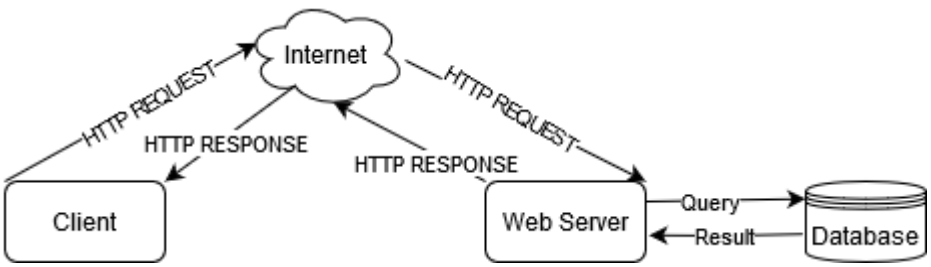


Gambar 2.2 Prosedur pembangkitan *qr code* dari sebuah teks

2.3 Sistem Berbasis *Website*

2.3.1 *Website*

Website merupakan suatu metode untuk menampilkan informasi di internet, baik berupa teks, gambar, suara maupun video yang interaktif dan mempunyai kelebihan untuk menghubungkan (*link*) satu dokumen dengan dokumen lainnya (*hypertext*) yang dapat diakses melalui sebuah *browser*(Yuhefizar, Mooduto, and Hidayat 2012).



Gambar 2.3 Arsitektur *website*

2.3.2 *PHP (HyperText Preprocessor)*

PHP Pertama kali dibuat pada musim gugur tahun 1994 oleh Rasmus Lerdof (rasmus@php.net) , awalnya digunakan pada *website* nya untuk mencatat siapa saja yang berkunjung dan melihat biodatanya. Versi pertama yang di-*release* tersedia pada awal tahun 1995, dikenal sebagai *tool Personal Home Page*¸ yang terdiri atas *engine parser* yang sangat sederhana yang hanya mengerti beberapa makro khusus dan sejumlah utilitas yang sering digunakan pada halaman-halaman *web*, seperti buku tamu, *counter* pengunjung, dan lainnya. *Parser* diprogram ulang pada pertengahan 1995 dan diberi nama PHP/FI (*Personal Home Page Form Interface*) versi 2.0 FI berasal dari paket Resmus lainnya yang ditulis untuk menginterpretasikan data dari form, yang kemudian dikombinasikan dengan *tool Personal Home Page* dan ditambahkan dukungan untuk *database* mSQL *(Mini SQL).*

Dikutip dari laman resmi PHP, PHP (*HyperText Preprocessor*) merupakan bahasa pemrograman *server side* yang disisipkan pada dokumen HTML yang dijalankan di *server* untuk pengembangan *web*. PHP pertama kali dibuat oleh Rasmus Lerdof tahun 1994 dan dilisensikan sebagai *software* yang bersifat *Open Source*(Group 2020).

PHP dikenal sebagai bahasa pemrograman script-*script* yang membuat dokumen HTML secara *on- the fly* yang dieksekusi di *server web*. Dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan teks editor. Dikenal sebagai bahasa pemrograman *server-side*(Sidik 2017). Berikut adalah contoh sederhana dari kode PHP.

1. Koneksi ke *MySQL*

Untuk menghubungkan *PHP* dan *MySQL* menggunakan fungsi yang telah disediakan yaitu *mysqli_connect()* dimana fungsi ini memiliki 4 (empat) parameter yaitu *host,username,password* dan nama *database*. Berikut adalah contoh kode program menghubungkan ke *MySQL:*

Kode Program 2.1 Koneksi ke *database MySQL*

```
01. <?php
02.     $db_host = "localhost"  // host
03.     $db_user = "root"       // username mysql
04.     $db_pass = ""           // password mysql
05.     $db_name = "mydb"       // nama databse
06.
07.     $conn = mysqli_connect($db_host,$db_user,$db_pass,$db_name) or die mysqli_error($conn);
08. ?>
```

2. Mengambil data dari *database* dengan perintah *SQL*

Mengambil data dari *database* dengan perintah *SQL* dapat menggunakan fungsi *mysqli_query()* dimana fungsi ini memiliki 2 (dua) parameter yakni variabel koneksi dan perintah *sql*. Berikut adalah contoh kode program mengambil data dari database dengan perintah *sql*.

Kode Program 2.2 Mengambil data dari *database* dengan perintah SQL

```php
<?php
    $db_host = "localhost"  // host
    $db_user = "root"       // username mysql
    $db_pass = ""           // password mysql
    $db_name = "mydb"       // nama databse

    $conn = mysqli_connect($db_host,$db_user,$db_pass,$db_name) or die mysqli_error($conn);

    $results = mysqli_query($conn, "SELECT * FROM users");
?>
```

2.3.3 Basis Data *(Database)*

Sistem basis data adalah sistem yang terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan(A.S. and Shalahuddin 2018). Dalam pengembangan sebuah perangkat lunak pasti ada yang namanya basis data, basis data menjadi peran penting untuk menyimpan suatu data, dengan adanya basis data pengguna dapat mengakses data secara mudah dan cepat. Secara teori Basis Data adalah suatu sistem yang memproses *input* berupa data menjadi *output* yaitu informasi yang diinginkan. Sedangkan menurut fungsinya adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

2.3.4 *MySQL*

Dikutip dari laman resmi MySQL, MySQL merupakan sistem manajemen basis data relasional (RDBMS) yang bersifat *open source* yang paling populer dikembangkan, didistribusikan dan didukung oleh *Oracle Corporation*(Oracle 2020). MySQL menggunakan arsitektur *client-server*, di mana yang bertugas untuk memanipulasi basis data adalah *server*. Sedangkan *client* digunakan untuk berkomunikasi dengan *server* dengan menggunakan perintah yang ditulis dalam bahasa SQL (*Structured Query Language*). Pernyataan *(statement)* SQL dapat digolongkan atas tiga golongan, yaitu:

1. *Data Definition Language* (DDL) yang mendefinisikan struktur data. Perintah-perintah SQL yang termasuk DDL antara lain *create, alter,* dan *drop.*

   a. *Create*

      Perintah *create* pada *Data Definition Language* meliputi *create database,* dan *create table.*

Kode Program 2.3 Perintah *sql create database*

```
MariaDB [(none)]> create database informatika;
Query OK, 1 row affected (0.008 sec)
```

   b. *Alter*

      Perintah *alter* pada *Data Definition Language* meliputi *change, modify, add, drop* dan *rename.*

Kode Program 2.4 Perintah *sql alter table*

```
MariaDB [informatika]> alter table mhs add email varchar(50);
Query OK, 0 rows affected (0.045 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   c. *Drop*

      Perintah *drop* pada *Data Definition Language* meliputi *drop database,* dan *drop table.*

Kode Program 2.5 Perintah *sql drop table*

```
MariaDB [informatika]> drop table mhs;
Query OK, 0 rows affected (0.109 sec)
```

2. *Data Manipulation Language* (DML) yang mencari *query* dan mengubah *(modify)* suatu tabel. Perintah-perintah SQL yang termasuk DML antara lain *select, insert, update,* dan *delete*.

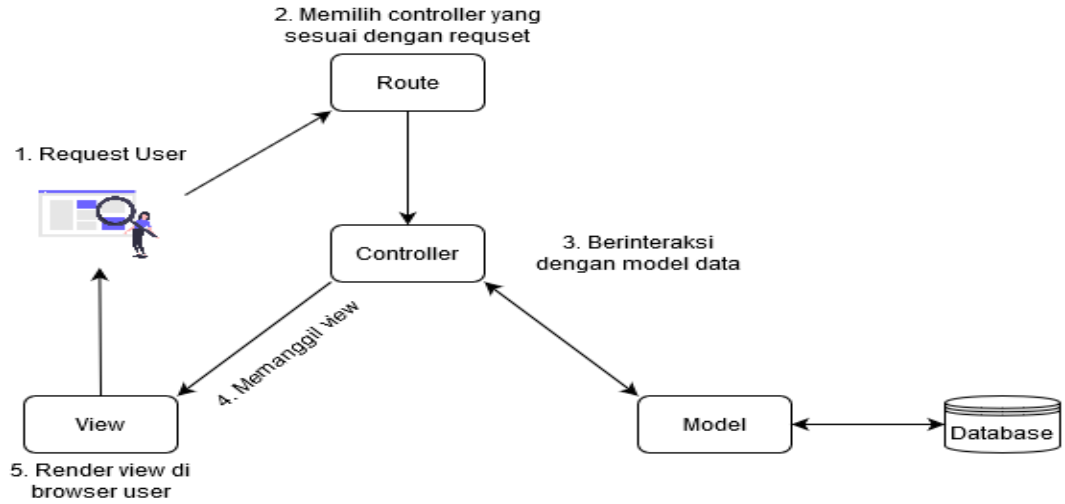
Dalam MySQL, sebuah *database* terdiri dari satu atau lebih tabel, dan setiap tabel terdiri dari baris dan kolom. Dalam bahasa SQL, pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi yang terdiri atas baris-baris data yang berada dalam satu atau lebih kolom.

2.3.5 *Framework* Laravel

F*ramework* adalah suatu struktur konseptual dasar yang digunakan untuk memecahkan atau menangani suatu masalah yang kompleks(Naista 2017). Singkatnya, *framework* adalah wadah atau kerangka kerja dari sebuah *website* yang akan dibangun. Dengan menggunakan kerangka tersebut waktu yang digunakan

dalam membuat *website* lebih singkat dan memudahkan dalam melakukan perbaikan.

Salah satu *framework* yang digunakan untuk membuat aplikasi berbasis *website* yaitu laravel. Laravel sebuah kerangka kerja *(framework) web* gratis dengan kode sumber terbuka yang menggunakan bahasa pemrograman PHP, dirancang oleh *Taylor Otwell*, Laravel adalah sebuah MVC *web development framework* PHP yang didesain untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan dan perbaikan serta meningkatkan produktifitas pekerjaan dengan sintak yang bersih dan fungsional set yang dapat mengurangi banyak waktu untuk implementasi (Widodo and Purnomo 2016) . Beberapa fitur Laravel adalah sistem pengemasan paket modular dengan *dependencies* manager khusus, berbagai cara untuk mengakses basis data relasional, utilitas yang membantu dalam penerapan dan pemeliharaan aplikasi, *templating engine* untuk menampilkan halaman *web* dinamis menggunakan *Blade*, dan berorientasi ke *syntactic sugar*. Laravel dapat digunakan untuk membangun aplikasi web apapun, mulai dari *web* sederhana hingga aplikasi level *enterprise*(Korop 2018).



Gambar 2.4 Konsep mvc *framework* laravel

1. Model

   Model mewakili struktur data. Biasanya *model* berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data. Berikut adalah contoh kode untuk *model* pada *framework* laravel.

Kode Program 2.6 Contoh *model framework* laravel

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The table associated with the model.
     *
     * @var string
     */
    protected $table = 'users';
}
```

2. Controller

   Controller merupakan bagian yang menjembatani *model* dan *view*. Berikut adalah contoh kode untuk *controller* pada *framework* laravel.

Kode Program 2.7 Contoh *controller framework* laravel

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\User;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
     * @param  int  $id
     * @return View
     */
    public function show($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
}
```

3. View

   View adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web. Berikut adalah contoh kode program *view* pada *framework* laravel.

Kode Program 2.8 Contoh *view framework* laravel

```php
<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

4. Route

   Route merupakan pendaftaran setap *URL* yang akan diakses oleh pengguna. Berikut adalah contoh kode program *route* pada *framework*.

Kode Program 2.9 Contoh *route framework* laravel

```php
<?php
    Route::get('user/{id}', 'UserController@show');
```

Berdasarkan situs resmi laravel versi yang rilis saat ini adalah laravel versi 8 dan pada versi 8 ini melanjutkan peningkatan yang dibuat di Laravel 7.x dengan

memperkenalkan versi laravel *jetstream*, *model factory classes*, *migration squashing*, pengelompokan pengerjaan, pembatasan tingkat yang ditingkatkan, peningkatan antrian, komponen Blade dinamis, tampilan paginasi Tailwind, pembantu pengujian waktu, peningkatan artisan serve, pendengar acara perbaikan, dan berbagai perbaikan bug lainnya dan peningkatan kegunaan(Otwell 2020). Laravel berada di bawah lisensi *MIT License* dengan menggunakan *Github* sebagai tempat berbagi *code* menjalankannya (Mediana 2018).

2.3.6 *Unified Modeling Language*

Menurut Rosa dan M. Shalahuddin (2018) *Unified Modelling Language* (UML) adalah salah satu standar bahasa yang banyak digunakan didunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

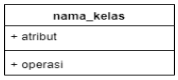
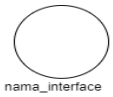
Unified Modelling Language (UML) terbagi ke dalam 3 (tiga) kategori yaitu diagram struktur *(structure diagrams)*, diagram kelakuan sistem *(behavior diagrams)*, dan diagram interaksi sistem *(interaction diagrams).*

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antara subsistem pada suatu sistem.

2.3.6.1 *Class Diagram*

Menurut Rosa dan M. Shalahuddin (2018) diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas dibuat agar pembuat program atau programmer membuat kelas sesuai rancangan di dalam diagram kelas agar dokumentasi perancangan dan perangkat lunak sinkron. Berikut adalah simbol yang ada pada diagram kelas.

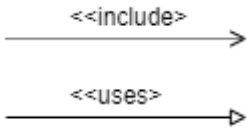
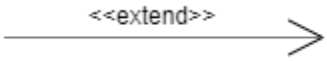
Tabel 2.1 Simbol *Class Diagram*

No	Simbol	Deskripsi
1.	Kelas 	Kelas pada struktur sistem.
2.	Antarmuka/*interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3.	Asosiasi/*association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
4.	Asosiasi berarah/*directed/association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Kebergantungan/*dependency* 	Kebergantungan antar kelas
7.	Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua-bagian *(whole-part).*

2.3.6.2 *Use Case*

Menurut Rosa dan M. Shalahuddin (2018) *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. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 2.2 Simbol *Use Case*

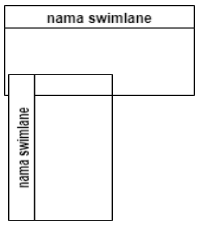
No	Simbol	Deskripsi
1.	*Use Case* 	Fungsionalitas yang disediakan dalam sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Biasanya dinyatakan dengan menggunakan kata kerja di awal frase name *use case*.
2.	Menggunakan/*include*/*uses* 	Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case ini*.
3.	Aktor/*actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat. Simbol dari aktor adalah orang tetapi aktor belum tentu merupakan orang, biasanya menggunakan kata benda di awal frase nama aktor.
4.	Ekstensi/*extend* 	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5.	Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum digunakan dari lainnya .

No	Simbol	Deskripsi
6.	Asosiasi/*association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

2.3.6.3 *Activity Diagram*

Menurut Rosa dan M. Shalahuddin (2018) Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada di dalam perangkat lunak. Berikut adalah simbol yang ada pada diagram aktivitas.

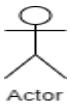
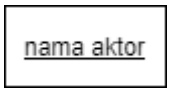
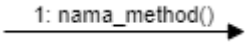
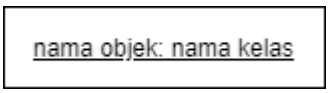
Tabel 2.3 Simbol *Activity Diagram*

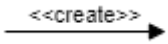
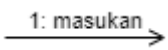
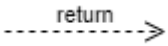
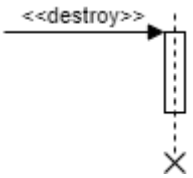
No	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6.	*Swimlane*  atau	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.3.6.4 *Sequence Diagram*

Menurut Rosa dan M. Shalahuddin (2018) Diagram *Sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki di kelas yang diinstansiasi menjadi objek. Untuk membuat *sequence* diagram perlu diketahui terlebih dahulu objek – objek yang terlibat dalam *use case* dan skenario perannya. Berikut adalah simbol yang ada pada *sequence diagram.*

Tabel 2.4 Simbol *Sequence Diagram*

No	Simbol	Deskripsi
1.	Aktor  atau  tanpa waktu aktif	Orang,proses,atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri, jadi walaupun simbol dari aktor adalah orang, tapi aktor belum tentu merupakan orang. Biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2.	Pesan tipe *call*  	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode. maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan objek yang berinteraksi.
3.	Objek  	Menyatakan objek yang berinteraksi

No	Simbol	Deskripsi
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
5.	Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6.	Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek.
7.	Pesan tipe *send* 	Menyatakan suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8.	Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9.	Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

2.3.7 Pengujian Perangkat Lunak

2.3.7.1 *Black Box Testing*

Black box testing berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program (Pressman 2010). *Black box* testing juga disebut *functional testing*, sebuah teknik

pengujian fungsional yang merancang *test case* berdasarkan informasi dari spesifikas(Nidhra 2012).

Pengujian *black box* atau *black box testing* merupakan strategi pengujian yang memperhatikan atau memfokuskan kepada faktor fungsionalitas dan spesifikasi perangkat lunak(Setiyani 2019). Pada pengujian ini tidak membutuhkan pengetahuan internal, struktur atau implementasi dari *software under test.* Kategori – kategori kesalahan yang diuji oleh *black box testing* adalah fungsi – fungsi yang salah atau hilang, kesalahan *interface*, kesalahan dalam struktur data atau akses *database* eksternal , kesalahan performa, kesalahan inisialisasi dan terminasi.

Pengujian *black box testing* dapat dilakukan pada setiap level pembangunan sistem yaitu mulai dari unit,*integration*, *system* dan *acceptance*.Di dalam pengujian *black box* terdapat 3 (tiga) teknik pengujian, yaitu *equivalence partitioning, boundary value analysis,* dan *cause effect graphic.*

1. *Equivalence partitioning*

   Teknik ini merupakan teknik pengujian *software* yang melibatkan pembagian nilai input ke dalam bagian nilai valid dan tidak valid dan memilih perwakilan dari masing – masing data tes.

2. *Boundary value analysis*

   Teknik ini merupakan teknik pengujian *software* yang melibatkan penentuan penentuan inputan nilai input dan memilih beberapa nilai dari batasan-batasan tersebut sebagai data tes.

3. *Cause effect graphic*

   Teknik ini merupakan teknik pengujian *software* yang melibatkan pengidentifikasian sebab-sebab (kondisi *input*) dan akibat-akibat (kondisi *input*) menghasilkan kasus tes.