BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Adapun penelitian terkait yang akan digunakan sebagai sumber acuan yang relevan dan terkini yaitu:

Berdasarkan penelitian dari Linda Rahmayanti, dkk, (2019) dengan judul "Sistem Absensi Mahasiswa Berdasarkan Citra Wajah Menggunakan Metode *Principal Component Analysis* (Pca)" Salah satu penggunaan deteksi wajah adalah absensi wajah, dimana absensi merupakan catatan kehadiran yang sangat penting dalam dunia pendidikan. Di Universitas Islam Majapahit, absensi memiliki kontribusi nilai yang cukup besar dalam kontrak kuliah. Oleh karena itu keaslian data absensi sangat diperlukan. Metode *Principal Component Analysis* (PCA) adalah salah satu metode yang banyak digunakan dalam ektrasi ciri citra, dimana pada proses deteksi maupun pengenalan dapat mengenali bagian wajah terlepas dari *background* yang digunakan. Untuk proses deteksi wajah menggunakan metode *Haar Cascade Classifier*, dimana metode ini mampu mengenali bagian wajah dan dengan bantuan *Library* EmguCV proses ini dapat dilakukan dengan cepat.

Berdasarkan penelitian dari Fahmi Syuhada, dkk, (2018), dengan judul "Pengenalan Wajah Untuk Sistem Kehadiran Menggunakan Metode *Eigenface* dan *Euclidean Distance"* bahwa Permasalahan yang diangkat yaitu bagaimana komputer yang sudah terkoneksi kamera *webcam* dapat mengenali wajah

seseorang walaupun orang tersebut tidak secara langsung melakukan proses absensi. Hal ini dilakukan melalui kamera webcam yang terkoneksi akan melakukan proses perekam dan pengenalan terhadap objek wajah yang dideteksi secara *real-time*. Kemudian dari hasil pengenalan wajah ini, digunakan untuk memperbaharui status kehadiran seseorang pada suatu tempat atau ruangan. Model pengenalan seperti ini sangat cocok diaplikasi untuk absensi kelas pengajaran.

Berdasarkan penelitian dari Muhammad Yusuf, dkk, (2016) dengan judul "Rancang Bangun Aplikasi Absensi Perkuliahan Mahasiswa Dengan Pengenalan Wajah" bahwa Proses absensi yang dilakukan secara manual dinilai kurang efektif karena terbukanya kesempatan melakukan kecurangan. Selain itu, proses rekapitulasi manual membutuhkan waktu yang lama. Sistem absensi dengan teknologi dapat diterapkan untuk membantu proses absensi dan rekapitulasi yang efektif. Pada tugas akhir ini, teknologi yang digunakan adalah sistem pengenalan wajah. Pembuatan aplikasi absensi dengan pengenalan wajah ini menggunakan metode *Eigenface* untuk melakukan proses pengenalan wajah.

Berdasarkan penelitian dari Muhammmad Rizki, dkk, (2015) dengan judul "Implementasi Pengenalan Wajah Dengan Metode Eigenface Pada Sistem Absensi" bahwa Secara umum proses absensi menggunakan pengenalan wajah ini dilakukan dengan memasukkan data wajah terlebih dahulu beserta password dari masing-masing orang, setelah itu dilakukan proses pemindaian untuk proses absensi. Metode eigenface dari opencv ini mencari data wajah yang mendekati dengan data wajah yang ada di database. Pada pengujian penelitian ini hasil yang

didapat berbeda-beda antara wajah satu dengan wajah yang lainnya, pada saat database berisi 10 data wajah, hasil rata-rata persentase kecocokan mencapai 88%, sedangkan pada saat database berjumlah 20 data wajah, hasil rata-rata persentase kecocokan mencapai 52%. Penyebab dari perbedaan hasil tersebut adalah karena faktor pencahayaan, jarak, bentuk wajah, serta jumlah data yang tersedia.

II.2. Landasan Teori

II.2.1. Aplikasi

Istilah aplikasi berasal dari bahasa Inggeris *application* yang berarti penerapan, lamaran ataupun penggunaan. Sedangkan secara umum, pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta jasa pengguna aplikasi lain yang dapat digunakan oleh pengguna yang akan dituju. Menurut kamus komputer eksekutif, pengertian aplikasi merupakan pemecahan masalah yang biasanya berpacu pada sebuah komputasi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Aplikasi biasanya berupa perangkat lunak yang berbentuk *Software* yang berisi kesatuan perintah atau program yang dibuat untuk melaksanakan sebuah pekerjaan yang diinginkan. (Inayah, 2015 : 3).

II.2.2. Pengenalan Pola

Pola adalah suatu bentuk dimana masing-masing pola memiliki ciri-cirinya. Ciri-ciri tersebut digunakan utuk mem-bedakan suatu pola dengan pola yang lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda yang tinggi,

se-hingga pengelompokkan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi. (Muhammad Rizki : 2015)

II.2.3. Pengenalan Wajah

Pengenalan wajah adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem keamanan selain pengenalan retina mata, pengenalan sidik jari dan iris mata. Dalam aplikasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah yang sebelumnya telah disimpan di dalam *database* tertentu. Pengenalan wajah melibatkan banyak variabel, misalnya citra sumber, cira hasil pengolahan citra, citra hasil ekstraksi dan data profil seseorang. Dibutuhkan juga alat pengindra berupa sensor kamera dan metode untuk menentukan apakah citra yang ditangkap oleh webcam tergolong wajah manusia atau bukan, sekaligus untuk menentukan informasi profil yang sesuai dengan citra wajah yang dimaksud. (Muhammad Rizki : 2015)

Pengenalan wajah merupakan salah satu pendekatan pengenalan pola untuk keperluan identifikasi wajah seseorang dengan pendekatan biometrik. Suatu biometrik bersifat unik sehingga dapat digunakan untuk mengenali identitas seseorang. Proses pengenalan biometrik dapat dibagi menjadi dua karakteristik, yaitu secara fisik dan secara perilaku. Biometrik fisik berasal dari pengukuran dan data yang ada langsung dari bagian manusia misalnya pengenalan sidik jari, pengenalan wajah, iris, retina, dan tangan. Sedangkan biometrik perilaku berasal dari pengukuran dan data yang berasal dari tindakan seperti suara, tanda tangan,

dan *keystrokes*. Sistem biometrik mengacu pada terintegrasinya antara perangkat keras dan perangkat lunak untuk melakukan proses identifikasi dan verifikasi. (Muhammad Yusuf : 2016)

II.2.4. Eigenface

Kata *eigenface* sebenarnya berasal dari bahasa Jerman *"eigenwert"* dimana *"eigen"* artinya karakteristik dan *"wert"* artinya nilai. Menurut Hanif *Eigenface* adalah salah satu *algoritma pengenalan pola wajah* yang berdasarkan pada *Principle Component Analysis* (PCA). Prinsip dasar dari penge-nalan wajah adalah dengan mengutip informasi unik wajah tersebut kemudian di-*encode* dan dibandingkan dengan hasil *de-code* yang sebelumnya dilakukan. Dalam metode *eigenface*, *decoding* dilakukan dengan menghitung *eigenvector* kemudian direpresentasikan dalam sebuah matrikss yang berukuran besar. *Eigenvector* juga dinyatakan sebagai karakteristik wajah oleh karena itu metode ini disebut dengan *eigenface*. Setiap wajah direpresentasikan dalam kombinasi linear *eigenface*. Metode *eigenface* pertama kali dikembangkan oleh Matthew Turk dan Alex Pentland dari Vision and Modeling Group, The Media Laboratory, Massachusetts Institute of Technology pada tahun 1987. Metode ini disempurnakan lagi oleh Turk dan Pentland pada tahun 1991. (Muhammad Rizki : 2015)

Eigenface adalah nama yang diberikan untuk satu set *eigenvector* ketika digunakan dalam pengenalan wajah pada bidang visi komputer. Dalam istilah Layman, *Eigenface* adalah sekumpulan *standardized face ingredient* dari analisis statistik dari banyak gambar wajah. Pendekatan *Eigenfaces* untuk pengenalan

wajah dikembangkan oleh Sirovich dan Kirby dan digunakan oleh Matthew Turk dan Alex Pentland pada klasifikasi wajah. Selain merancang sistem untuk pengenalan wajah, Matthew Turk dan Alex Pentland juga menunjukkan cara menghitung *eigenvector* untuk melakukan *eigendecomposition* pada sebagian besar gambar wajah. *Eigenvector* berasal dari kovarian matriks distribusi probabilitas pada ruang vektor dari gambar wajah. *Dataset* wajah yang digunakan harus diambil dalam kondisi pencahayaan dan resolusi yang sama dengan saat melakukan pengenalan wajah baru. Algoritma pengenalan wajah dengan metode *Eigenface* dilakukan melalui beberapa tahapan, yaitu:

1. Menyusun *Flatvector* Matriks Citra
2. Menghitung Nilai Tengah atau *Mean (Ψ)*
3. Menghitung Selisih antara *Training Image* dengan Nilai Tengah atau *Mean (Ψ)*
4. Menghitung Nilai Matriks Kovarian
5. Menghitung Nilai *Eigenvalue* dan *Eigenvector*
6. Mencari Nilai *Eigenface*
7. Proses Identifikasi (Muhammad Yusuf : 2016)

S = (Γ1, Γ2, …, ΓM)……..…(1)

Cari nilai nilai eigenvalue (λ) dan eigen-vector (ν).

L x ν = λ x ν ……..……(2)

L x ν = λI x ν ……(3)

$L - \lambda I = 0$ $atau$ $\lambda I - L = 0$……(4)

$\mu i = \sum_{k=1}^{m} \text{vikϕk}$ ……...……(5)

$\varepsilon k = \Omega - \Omega k$……..……(6)

II.2.5. Basis Data (*Database*)

Pangkalan data atau basis data (*database*) adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi. Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Model yang umum digunakan sekarang adalah model relasional, yang mewakili semua informasi dalam bentuk tabel-tabel yang saling berhubungan dimana setiap tabel terdiri dari baris dan kolom. Model yang lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar tabel (Neni Purwati dan Hendra Kurniawan, 2015 : 50).

II.2.6. Normalisasi

Normalisasi adalah proses pembentukan struktur basis data sehingga sebagian besar ambiguity bisa dihilangkan. Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data relasional yang mengelompokkan atribut dari suatu tabel sehingga membentuk struktur tabel yang normal. Adapun kriteria tabel dikatakan normal adalah ketika tidak ada kerangkapan data (redudansi data). (Puspitasari et al., 2016)

Tujuan dari normalisasi adalah untuk :

1. Untuk menghilangkan kerangkapan data sehingga meminimumkan pemakaian *storage* yang dipakai oleh *base relations* (*file*).
2. Untuk mengurangi kompleksitas.
3. Untuk mempermudah pemodifikasian data.

II.2.6.1. Proses Normalisasi

1. Proses Normalisasi
   a. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu kebeberapa tingkat.
   b. Apabila tabel yang diuji belum memenuhi persyaratan tertentu maka tabel tersebut pelu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.
2. Tahapan Normalisasi :
   1) Bentuk tidak normal : Menghilangkan perulangan *grup*.

Tabel II.2. Contoh bentuk tidak normal (Unnormal)

No-Mhs	Nama Mhs	Jurusan	Kode-MK	Nama-MK	Kode Dosen	Nama Dosen	Nilai
2683	Welli	MI	M1350	Manajemen DB	B104	Ati	A
			M1465	Analisis Perc. Sistem	B317	Dita	B
5432	Bakti	AK	M1350	Manajemen DV	B104	Ati	C
			Akn201	Akuntansi	D310	Lia	B
			MKT300	Dasar Pemasaran	B212	Lola	A

Sumber : Mukhlisulfatih Latief : 2016

2) Bentuk Normal pertama (1NF) : Menghilangkan ketergantungan sebagian.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kesatu bila setiap data bersifat atomik yaitu setiap irisan baris dan kolom hanya mempunyai satu nilai data.

Tabel II.3. Contoh Bentuk Normal Pertama (1NF)

No-Mhs	Nama Mhs	Jurusan	Kode-MK	Nama-MK	Kode Dosen	Nama Dosen	Nilai
2683	Welli	MI	M1350	Manajemen DB	B104	Ati	A
2683	Welli	MI	M1465	Analisis Perc. Sistem	B317	Dita	B
5432	Bakti	AK	M1350	Manajemen DV	B104	Ati	C
5432	Bakti	AK	Akn201	Akuntansi	D310	Lia	B
5432	Bakti	AK	MKT300	Dasar Pemasaran	B212	Lola	A

Sumber : Mukhlisulfatih Latief : 2016

3) Bentuk Normal kedua (2NF) : Menghilangkan ketergantungan transitif.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kedua bila relasi tersebut sudah memenuhi bentuk normal kesatu dan atribut yang bukan *key* sudah tergantung penuh terhadap *key*-nya.

Tabel II.4. Contoh Bentuk Normal Kedua (2NF)

Kode-MK	Nama-MK	Kode Dosen	Nama Dosen
M1350	Manajemen DB	B104	Ati
M1465	Analisis Perc. Sistem	B317	Dita
M1350	Manajemen DV	B104	Ati
Akn201	Akuntansi	D310	Lia
MKT300	Dasar Pemasaran	B212	Lola

Sumber : Mukhlisulfatih Latief : 2016

4) Bentuk Normal ketiga (3NF) : Menghilangkan anomali-anomali hasil dari ketergantungan fungsional. Yaitu : suatu relasi dikatakan sudah memenuhi bentuk

normal ketiga bila relasi tersebut sudah memenuhi bentuk normal kedua dan atribut yang bukan key tidak tergantung transitif terhadap *key*-nya.

Tabel II.5. Contoh Tabel Mahasiswa Dan Tabel Kuliah (3NF)

No_Mhs	Nama Mhs	Jurusan
2683	Welli	MI
5432	Bakti	AK

Sumber : Mukhlisulfatih Latief : 2016

Normalisasi adalah proses penyusunan *table-table* yang tidak redundan (*double*) yang dapat menyebabkan anomali pada saat terjadi manipulasi data seperti tambah, ubah dan hapus. Tujuan dari normalisasi adalah:

a. Untuk meghilangkan kerangkapan data.

b. Untuk mengurangi, kompleksitas.

c. Untuk mempermudah pemodifikasian data. (Trisnawati, 2016)

II.2.7. MySQL

MySQL (*My Structure Query Language*) merupakan sebuah program pembuat *database* yang bersifat *Open Source,* artinya semua orang dapat menggunakannya dan dapat dijalankan pada semua *platform* baik *Windows* maupun linux. *MySQL* juga merupakan sebuah perangkat lunak sistem manajemen basis data *SQL* yang bersifat jaringan sehingga dapat digunakan intuk aplikasi multi *user*. *MySQL* juga sering dikenal dengan nama sistem manajemen *database* relasional. Suatu *database* relasional menyimpan data dalam *table* yang terpisah. *Tabel -table* tersebut terhubung oleh suatu relasi terdefinisi yang memungkinkan memperoleh kombinasi data dari beberapa table dalam suatu permintaan. Untuk administrasi *database*, seperti pembuatan *database*, pembuatan tabel, dan

sebagainya dapat digunakan aplikasi berbasis web seperti *PHP MyAdmin* dengan aplikasi *XAMPP*. (Saipul Anwar, 2016 : 96**)**

II.2.8. XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *XAMPP* merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, *MySQL*, *PHP* dan *Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis. Untuk mendapatkanya dapat men-*download* langsung dari *web* resminya. (Randi V Palit, 2015 : 2).

II.2.9. PHP

PHP atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhususkan untuk pengembangan *web* dan dapat ditanamkan pada sebuah skripsi HTML. Bahasa PHP dapat dikatakan menggambarkan beberapa bahasa pemrograman seperti C, *Java*, dan *Perl* serta mudah untuk dipelajari. PHP merupakan bahasa *scripting server – side*, dimana pemrosesan datanya dilakukan pada sisi *server*. Sederhananya, *server* yang akan menterjemahkan skrip program, baru kemudian

hasilnya akan dikirim kepada *client* yang melakukan permintaan. Adapun pengertian lain PHP adalah *akronim* dari *Hypertext Preprocessor*, yaitu suatu bahasa pemrograman berbasiskan kode – kode *(script)* yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke *web browser* menjadi kode HTML". (Astria Firman, 2016 : 30).

II.2.10. *UML* (*Unified Modelling Language*)

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan Bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi.

Unified Modeling Language (UML) biasa digunakan untuk :

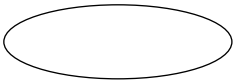
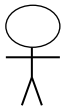
a. Menggambarkan batasan sitem dan fungsi - fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
b. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
c. Menggambarkan representasi struktur *static* sebuah sistem dalam betuk *class diagrams*.

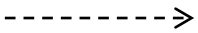
d. Membuat model *behavior* "yang menggambarkan kebiasaan atau sifat sebuah sistem" dengan *state transition diagrams*.
e. Menyatakan arsitektur implementasi fisik menggunakan *component and development.*
f. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*. (Ommi Alfina dan Fitriana Harahap : 2019 : 36)

II.2.10.1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.6 dibawah ini :

Tabel II.6. Simbol *Use Case*

Gambar	Keterangan	Deskripsi
	*Use case*	Menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor	Sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem.
	Asosiasi	Penghubung antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung

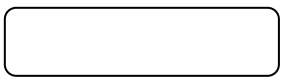
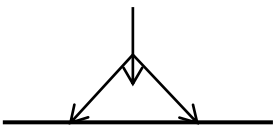
		dan bukannya mengidikasikan aliran data.
	Asosiasi	Penghubung antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*	Merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*	Merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

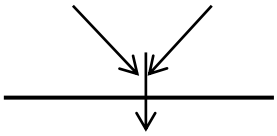
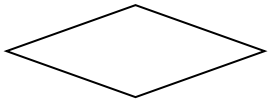
(Sumber : Ade Hendini, 2016)

II.2.10.2. *Activity* Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.7 dibawah ini:

Tabel II.7. Simbol *Activity Diagram*

Gambar	Keterangan	Deskripsi
	*Start point*	Diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*	Akhir aktifitas.
	*Activites*	Menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan).	Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.

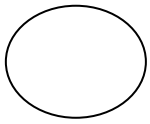
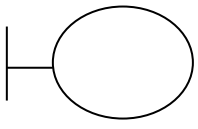
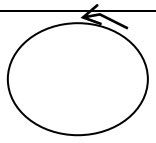
	*Join* (penggabungan)	Digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*	Menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*	Pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

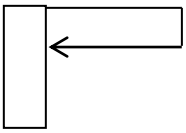
(Sumber : Ade Hendini, 2016 : 109)

II.2.10.3. *Sequence* Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.8 dibawah ini :

Tabel II.8. Simbol *Sequence Diagram*

Gambar	Keterangan	Deskripsi
	*Entity Class*	Merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*	Berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*	Suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan

		berbagai objek.
	*Message*	Simbol mengirim pesan antar *class*.
	*Recursive*	Menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*	Mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline	Garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Ade Hendini, 2016 : 110)

II.2.10.4. *Class* Diagram

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class* diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class* diagram secara khas meliputi : Kelas (*Class*), Relasi *Assosiations*, *Generalitation* dan *Aggregation*, attribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality* (Ade Hendini, 2016 : 111).

Tabel II.9. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Ade Hendini, 2016 : 110)