BAB 2
LANDASAN TEORI

2.1 Penerapan

Implementasi adalah bermuara pada aktivitas, aksi, tindakan atau adanya mekanisme suatu sistem, implemantasi bukan sekedar aktivitas, tapi suatu kegiatan yang terencana dan untuk mencapai tujuan kegiatan[5].

Penerapan adalah suatu perbuatan mempraktekkan suatu teori, metode, dan hal lain untuk mencapai tujuan tertentu dan untuk suatu kepentingan yang diinginkan oleh suatu kelompok atau golongan yang telah terencana dan tersusun sebelumnya[6].

2.2 Face Recognition

Pengenalan wajah merupakan sebuah sistem identifikasi pribadi yang menggunakan karakteristik wajah seseorang. Pengenalan wajah sendiri merupakan suatu cabang ilmu biometrik, yaitu suatu bidang keilmuan yang menggunakan karakteristik fisik dari seseorang untuk menentukan atau mengungkapkan identitasnya.[7]

Evaluasi terbaru dari sistem pengenalan wajah komersial menunjukkan tingkat kinerja untuk verifikasi wajah dari sistem terbaik agar setara dengan pengenal sidik jari untuk wajah depan yang diterangi secara seragam[8].

2.3 Deteksi

Deteksi adalah suatu proses untuk memeriksa atau melakukan pemeriksaan terhadap sesuatu dengan menggunakan cara dan teknik tertentu. Deteksi digunakan untuk berbagai masalah, misalkan pendeteksi wajah, dimana sistem ini mengidentifikasi berbagai wajah.

Tujuan dari deteksi adalah memecahkan suatu masalah dengan berbagai cara tergantung metode yang diterapkan sehingga menghasilkan sebuah solusi.

2.4 Wajah

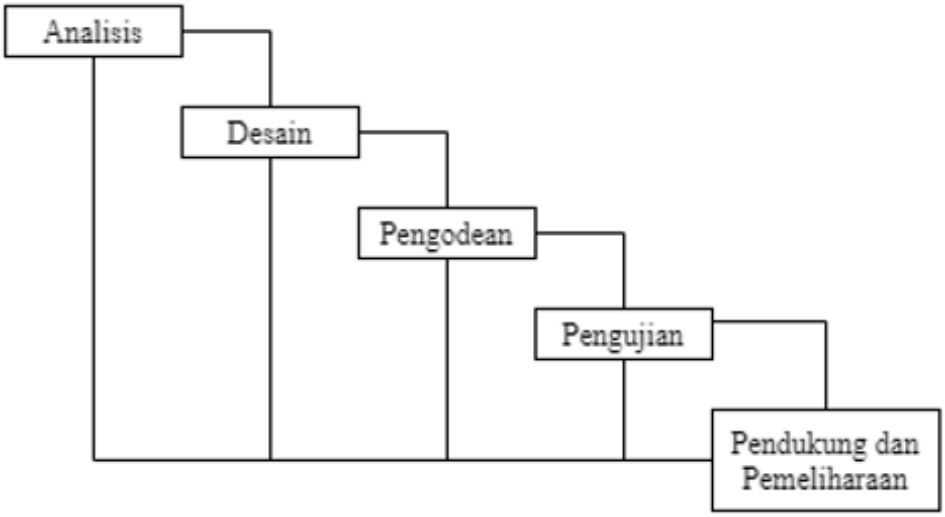
Wajah adalah bagian depan dari kepala, pada manusia meliputi wilayah dari dahi hingga dagu, termasuk rambut, dahi, alis, pelipis, mata, hidung, pipi, mulut, bibir, gigi, kulit, termasuk dagu. Wajah terutama digunakan untuk ekspresi wajah, penampilan, serta identitas.

Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpan[9]. Wajah dapat di deteksi melalui kontur atau struktur wajah dan tepi citra. Tepi citra merupakan perubahan signifikan dari nilai keabuan pada suatu citra.

2.5 Metode *Waterfall* – Roger S

Metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle*), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (*planning*), permodelan (*modeling*), konstruksi (*construction*), serta penyerahan sistem ke para pelanggan/pengguna (*deployment*), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2012).

Tahapan metode *waterfall* dapat dilihat pada gambar di bawah ini.



Gambar 0.1Metode Waterfall

Penjelasan dari setiap tahapannya adalah sebagai berikut :

1. Analisis

Tahap ini bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi diperoleh melalui jurnal dan studi literatur. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2. Desain

Pada tahap Design ini, aplikasi yang akan dibangun adalah berbasis *Web*site. Bahasa pemrograman yang digunakan dalam pembangunan aplikasi ini adalah Python. Database yang digunakan adalah MySQL.

3. Pengodean

Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian melibatkan verifikasi bahwa setiap fungsi memenuhi spesifikasinya

4. Pengujian

Tahap ini adalah tahap dimana aplikasi akan dilakukan pengujian internal apakah fungsionalitasnya sudah berjalan sesuai dengan yang diharapkan atau tidak.

5. Pendukung dan Pemeliharaan

Tahap akhir dalam model waterfall. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru. Kesimpulan dan Saran[10].

2.6 Flask

Flask sebagai kerangka aplikasi tampilan dari sebuah web yang menggunaan flask dan bahasa python, pengembang membuat sebuah *web* terstruktur dan mengatur suatu web dengan mudah. Flask termasuk pada jenis *microframework* karena tidak memerlukan suatu alat tertentu dalam penggunaannya. Hal ini dikarenakan fungsi komponen-komponen sudah disediakan oleh pihak ketiga dan flask menggunakan ekstensi membuat fitur dan komponen-komponen diimplementasikan oleh flask sendiri[11].

2.7 Python

Python merupakan salah satu bahasa pemograman tingkat tinggi yang bersifat *interpreter, interactive, objectoriented*, dan dapat beroperasi hampir di semua *platform: Mac, Linux*, dan *Windows*. *Python* termasuk bahasa pemograman yang mudah dipelajari karena *sintaks* yang jelas, juga dapat dikombinasikan dengan penggunaan modul siap pakai, dan struktur data tingkat tinggi yang efisien.

Distribusi *Python* dilengkapi dengan suatu fasilitas seperti *shell* di *Linux*. Lokasi penginstalan *Python* biasa terletak di "/usr/bin/Python", dan dapat juga berbeda. Menjalankannya cukup dengan pengetikan "*Phyton*:, lalu akan muncul tampilan ">>>", dan *Phyton* siap menerima perintah. Ada tanda "…" yang berarti baris berikutnya dalam suatu blok *prompt* '>>>'. *Text editor* digunakan untuk modus skrip[12].

Gambar 0.2 Python

2.8 Open CV

Open Computer Vision (*OpenCV*) sendiri merupakan *library open source* yang tujuannya dikhususkan untuk melakukan pengolahan citra. Maksudnya adalah agar komputer mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia. *OpenCV* telah menyediakan banyak algoritma visi komputer dasar. O*penCV* juga menyediakan modul pendeteksian objek yang menggunakan algoritma Viola Jones[12]. *OpenCV* digunakan untuk pemrosesan dan penglihatan gambar, GUI, Struktur data, Gambar, dan I/O[13].

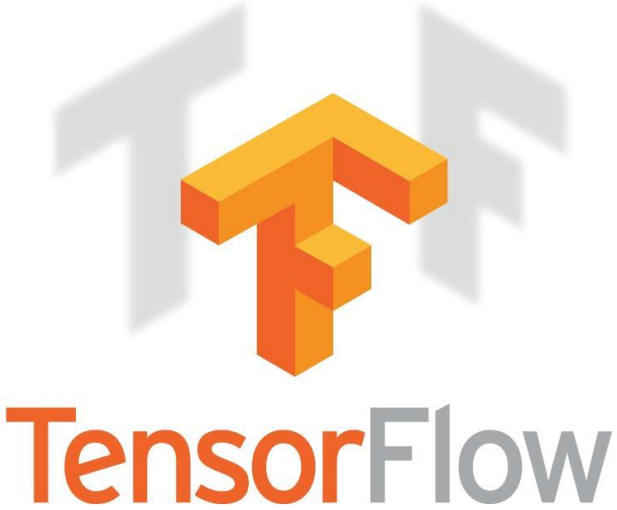
2.9 Image Processing

Image processing atau pengelohan citra merupakan suatu metode atau teknik yang digunakan untuk memproses citra atau gambar dengan memanipulasi menjadi data gambar untuk mendapatkan suatu informasi mengenai obyek yang sedang diamati[14].

Karena komputer merepresentasikan angka dengan menggunakan presisi hingga, angka-angka ini harus dikuantisasi agar dapat direpresentasikan secara digital. Pengolahan citra digital terdiri dari manipulasi angka presisi hingga tersebut[15].

2.10 Tensorflow

TensorFlow adalah pustaka perangkat lunak sumber terbuka dan gratis untuk pembelajaran mesin . Hal ini dapat digunakan di berbagai tugas tetapi memiliki fokus khusus pada pelatihan dan inferensi dari jaringan saraf yang mendalam.



Gambar 0.3 Logo Tensorflow

Tensorflow adalah pustaka matematika simbolis berdasarkan dataflow dan pemrograman yang dapat dibedakan . Ini digunakan untuk penelitian dan produksi di Google .

TensorFlow dikembangkan oleh tim Google Brain untuk penggunaan internal Google . Ini dirilis di bawah Lisensi Apache 2.0 pada tahun 2015.

TensorFlow adalah sistem generasi kedua Google Brain. Versi 1.0.0 dirilis pada 11 Februari 2017. Meskipun implementasi referensi berjalan pada satu perangkat, TensorFlow dapat berjalan pada beberapa CPU dan GPU (dengan ekstensi CUDA dan SYCL opsional untuk komputasi tujuan umum pada unit pemrosesan grafis).

TensorFlow tersedia di Linux 64-bit , macOS , Windows , dan platform komputasi seluler termasuk Android dan iOS .

Arsitekturnya yang fleksibel memungkinkan penerapan komputasi yang mudah di berbagai platform (CPU, GPU, TPU), dan dari desktop ke cluster server hingga perangkat seluler dan edge.

TensorFlow perhitungan dinyatakan sebagai stateful dataflow grafik . Nama TensorFlow berasal dari operasi yang dilakukan jaringan neural tersebut pada larik data multidimensi, yang disebut sebagai tensor . Selama Google I / O Conference pada Juni 2016, Jeff Dean menyatakan bahwa 1.500 repositori di GitHub menyebutkan TensorFlow, dimana hanya 5 dari Google.

2.11 Keras

Gambar 0.4 Logo Keras

Keras adalah pustaka perangkat lunak sumber terbuka yang menyediakan antarmuka Python untuk jaringan saraf tiruan . Keras bertindak sebagai antarmuka untuk pustaka TensorFlow .

Hingga versi 2.3 Keras mendukung beberapa backend, termasuk TensorFlow , Microsoft Cognitive Toolkit , R , Theano , dan PlaidML . Mulai versi 2.4, hanya TensorFlow yang didukung. Dirancang untuk memungkinkan eksperimen cepat dengan

jaringan neural dalam , ini berfokus pada ramah pengguna, modular, dan dapat diperluas. Ini dikembangkan sebagai bagian dari upaya penelitian proyek ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), dan penulis serta pemelihara utamanya adalah François Chollet, seorang Googleinsinyur. Chollet juga merupakan penulis model jaringan saraf dalam XCeption.

2.12 Dlib

Gambar 0.5 Logo Dlib

Dlib adalah pustaka perangkat lunak lintas platform tujuan umum yang ditulis dalam bahasa pemrograman C ++ . Desainnya sangat dipengaruhi oleh ide-ide dari desain kontrak dan rekayasa perangkat lunak berbasis komponen . Jadi, pertama dan terpenting, satu set komponen perangkat lunak independen. Ini adalah perangkat lunak sumber terbuka yang dirilis di bawah Lisensi Perangkat Lunak Boost .

Sejak pengembangan dimulai pada tahun 2002, Dlib telah berkembang dengan menyertakan berbagai macam alat. Pada 2016, ini berisi komponen perangkat lunak untuk menangani jaringan , utas , antarmuka pengguna grafis , struktur data , aljabar linier , pembelajaran mesin , pemrosesan gambar , penambangan data , XML dan penguraian teks, pengoptimalan numerik , jaringan Bayesian , dan banyak tugas lainnya. Dalam beberapa tahun terakhir, sebagian besar pengembangan telah difokuskan pada pembuatan seperangkat alat pembelajaran mesin statistik yang luas dan pada tahun 2009 Dlib diterbitkan diJurnal Penelitian Pembelajaran Mesin . Sejak itu telah digunakan di berbagai domain.

Dlib adalah suatu library berfungsi dengan cara menganalisis bagian wajah dengan mengekstrak nilai gambar. Dlib akan digunakan untuk membantu mengolah gambar wajah pada metode facial landmark.

2.13 Website

Website merupakan kumpulan dari halaman-halaman web yang berhubungan dengan file dile lain yang terkait. Home page adalah sebuah halaman yang pertama kali diliat ketika seseorang mengunjungi website. Dari home page, pengunjung dapat mengklik hyperlink untuk ke halaman lain yang terdapat dalam website tersebut[16].

Website juga dapat dijadikan sarana untuk menampung aplikasi. Kelebihan menggunakan aplikasi website yaitu tidak perlu mengunduh seperti pengunduhan aplikasi native.

2.14 PHP

PHP merupakan suatu bahasa pemrograman Open Source yang digunakan secara luas terutama untuk mengembangkan web dan disimpan dalam bentuk HTML. PHP bersifat server side yang di dalam HTML artinya dalam suatu dokumen HTML dapat dimasukan skrip PHP.[11]

Kelebihannya PHP lebih ringkas karena tidak perlu men*setting* secara berlebihan. Karena konfigurasi pada database juga dapat dilakukan dengan lebih mudah. Bahasa pemrograman PHP ini open source, maka pengguna dapat menggunakannya dengan bebas dan gratis.

2.15 MySQL

MySQL merupakan sebuah *database* yang dapat menerima dan mengirimkan data dengan sangat cepat, *multi-user*, serta menggunakan perintah standar SQL (*Structured Query Language*) baik diguakan *client server* mapun *server*

a. Kelebihan
    1. Bersifat *Open Source*.
    2. Mendukung *Multi-User.*
    3. Mendukung integrase dengan Bahasa pemrograman lain.
    4. Tidak membutuhkan RAM besar.

5. Struktur table yang fleksibel.
6. Tipe data yang bervariasi.

b. Kekurangan

1. *Technical Support* kurang bagus.
2. Sulit mengelola *database* besar.

2.16 Pycharm

PyCharm adalah Python Integrated Development Environment (IDE) khusus yang menyediakan berbagai alat penting untuk pengembang Python, terintegrasi erat untuk menciptakan lingkungan yang nyaman untuk pengembangan Python, web, dan ilmu data yang produktif.

Gambar 2.2 Pycharm

2.17 Visual Studio Code

Visual Studio Code atau biasa disebut VSCode merupakan source code yang dikembangkan oleh Microsoft untuk Windows, Linux dan MacOs. Ini termasuk dukungan untuk debugging, GIT Controlyang disematkan, penyorotan sintaks, penyelesaian kode cerdas, cuplikan, dan kode refactoring. Hal ini juga dapat disesuaikan, sehingga pengguna dapat mengubah tema editor, shortcut keyboard, dan preferensi.Visual Studio Code gratis dan open-source.

Gambar 0.6 Visual Studio Code

2.18 Analisa dan Perancangan Berorientasi Objek

Dalam melakukan analisa dan perancangan sistem berorientasi objek penulis menggunakan UML (*Unified Modelling Language*) untuk pemodelannya. Sedangkan alat (*tool*) visual modeling yang akan digunakan untuk menggambarkan model analisa dan perancangan adalah Microsoft Visio 2007. Implementasi perangkat lunak menggunakan bahasa pemrograman PHP.

2.18.1 Unified Modelling Language

Duet mereka pada Oktober 1995 menghasilkan Unified Method versi 0.8, yang menjadi cikal bakal dari UML sebagai bahasa pemodelan standar untuk aplikasi object oriented.

Pada tahun 2002 lahirlah UML versi 2.0 dengan penambahan dan penggantian diagram menjadi 13 buah diagram. Diagram-diagram ini terbagi menjadi 3 kategori, yaitu:

a. *Structural diagrams* : menggambarkan elemen dari spesifikasi yang mengabaikan waktu. Terdiri dari : *Class Diagram, Object Diagram, Component Diagram, Deployment Diagram, Composite Structure Diagram* dan *Package Diagram*.
b. *Behavior Diagram* : menggambarkan ciri-ciri *behavior/method/function* dari sebuah sistem atau *business process*. Terdiri dari : *Use case Diagram, Activity Diagram,* dan *State Machine Diagram*.

c. *Interaction Diagram* : bagian dari *behavior diagram* yang menggambarkan *object interactions*. Terdiri dari : *Communication Diagram, Interaction Overview Diagram, Sequence Diagram* dan *Timing Diagram*.

Karena UML sangat fleksibel, ada cara untuk melihat diagram UML berdasarkan kategori berikut :

a. *Static Diagram* : menunjukkan segi statik dari sistem. Kategori ini sama dengan *structural diagram*.

b. *Dynamic Diagram* : menunjukkan bagaimana sistem berkembang setiap waktu. Meliputi *state-machine diagram* dan *timing diagram.*

c. *Functional Diagram* : menunjukkan detail dari perilaku (*behavior*) dan algoritma bagaimana sistem memenuhi perilaku yang diinginkannya.

Kategori ini termasuk *Use Case, interaction* dan *activity diagram.*

2.18.2 Use Case Diagram

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang *user*. Digunakan untuk menggambarkan hubungan antara internal sistem dan eksternal sistem atau hubungan antara *use case* dan aktor[17].

1. Aktor

Aktor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam use case. Actor menggambarkan orang, sistem atau entitas eksternal yang secara khusus membangkitkan sistem dengan input atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. Actor dilukiskan dengan peran yang mereka mainkan dalam use case, seperti Staff, Kurir dan lain-lain.

Dalam use case diagram terdapat satu aktor pemulai atau initiator actor yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor

lain yang berpartisipasi atau participating actor. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.

2. *Use Case*

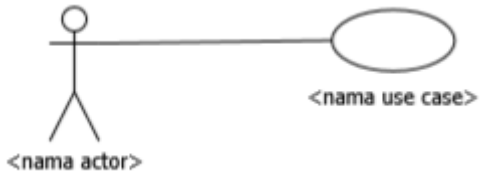
Use case yang dibuat berdasarkan keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor.

Dalam UML *Use Case* dinotasikan dengan gambar.

3. *Relationship*

Relasi atau *Relationship* digambarkan sebagai bentuk garis antara dua simbol dalam use case diagram. Relasi antara actor dan use case disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.

Relasi- relasi yang terjadi pada *use case diagram* bisa antara *actor* dengan *use case* atau *use case* dengan *use case*.



Relasi antara *Use Case* dengan *Use Case.*

a. *Include*, pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<include>>.

b. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak *control form* dan

mendeklarasikan *ekstension* pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<extend>>.

c. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base parent use case*. Digambarkan dengan garis berpanah tertutup dari *base use case* ke *parent use case*.

2.18.3 Activity Diagram

Diagram aktivitas menggambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung penggambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang (*branch*) dan penyatuan (*merge*).

Sebuah *branch* memiliki sebuah *transition* masuk atau yang disebut dengan *incoming transition* dan beberapa *transition* keluar atau yang disebut dengan *outgoing transition* dari branch yang berupa keputusan-keputusan. Hanya satu dari *outgoing transition* yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat *mutually exclusive*. [*else*] digunakan sebagai keterangan singkat yang menunjukkan bahwa *transition* "*else*" tersebut harus digunakan jika semua keputusan yang ada pada *branch* salah.

Sebuah *merge* memiliki banyak *input transition* dan sebuah *output*. *Merge* menandakan akhir dari suatu kondisi yang diawali dengan sebuah *branch*. Selain *branch* dan *merge*, di dalam diagram aktivitas terdapat pula *fork* dan *join*. *Fork* memiliki satu *incoming transition* dan beberapa *outgoing transition*. Sedangkan pada *join*, *outgoing transition* diambil atau digunakan hanya ketika semua *state* pada *incoming transition* telah menyelesaikan aktivitasnya.

2.19 Sequence Diagram

Diagram yang menggambarkan bagaimana objek berinteraksi dengan objek lainnya melalui pesan (*message*) yang disampaikan, disusun dalam urutan kejadian atau waktu dan secara khusus berasosiasi dengan *use case*[17].

Tujuan dari penggunaan *sequence diagram* ini untuk mengkomunikasikan requirement kepada tim teknis karena diagram ini dapat lebih mudah untuk dielaborasi menjadi model design, mengembangkan model deskripsi use-case menjadi spesifikasi design, dan memfokuskan pada identifikasi method di dalam sebuah system.

2.20 Class Diagram

Class diagram merupakan bagian yang paling penting dalam analisa dan perancangan berorientasi objek. Dalam UML diagram kelas digunakan untuk memodelkan *static structure* dari sistem informasi[17].

Kelas merupakan himpunan dari objek yang sejenis yang mempunyai atribut dan perilaku (*behaviors/method*) yang sama. Atribut adalah sebuah nilai data karakteristik yang dimiliki oleh objek sebuah kelas sedangkan *method* adalah perilaku atau operasi yang dikenakan oleh suatu kelas. Pada gambar kelas terdapat tiga bagiannya.

Diagram kelas menggambarkan struktur objek sistem, dimana diperlihatkan hubungan antar mereka. Diagram kelas (*Class Diagram*) merupakan pondasi untuk *component diagram* dan *deplo*elas merupakan himpunan dari obyek yang sejenis yang mempunyai atribut dan perilaku (behaviors/method) yang sama. Atribut adalah sebuah nilai*oyment diagram*.

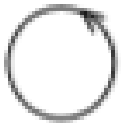
Secara garis besar terdapat 3 jenis *Class*. Ketiganya dikelompokkan berdasarkan fungsi dan karakternya masing-masing, yaitu :

a. *Entity Class Diagram*

Merupakan paket utama dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk sistem dan menjadi landasan untuk menyusun basis data pada model data konseptual.

b. *Control Class Diagram*

Berisi kumpulan kelas yang menjadi kontrol program termasuk koneksi dengan basis data dan merupakan kelas perantara atau penghubung antara *entity class* dengan kelas antar muka pemakai (*interface*).



c. *Boundary Class Diagram*

Berisi kumpulan kelas yang menjadi *interface* antara pemakai (*user*) dengan sistem, seperti tampilan form untuk pencetakan.