BAB 2
TINJAUAN PUSTAKA

2.1 Biometrik

Menurut pendapat Roethenbaugh dan Mansfield [8] biometrik adalah metode otomatisasi dari pengenalan ataupun verifikasi identitas seseorang berdasarkan pada sebuah karakteristik fisik ataupun tingkah laku. Karakteristik perilaku individu seperti sidik jari, iris, geometri tangan dan retina merupakan ciri khusus fisik setiap individu atau bisa disebut juga dengan biometrik.

2.2 Absensi

Suatu sistem formal dan terstruktur yang menilai, mengukur dan mempengaruhi sifat-sifat yang berkaitan dengan pekerjaan, perilaku, dan hasil termasuk ketidakhadiran. Absensi adalah suatu kegiatan mendokumentasikan kehadiran [8].

2.3 Deteksi Wajah (*Face Detection*)

Deteksi wajah adalah proses mendeteksi wajah manusia secara otomatis dalam media visual (gambar atau video digital). Wajah yang terdeteksi dilaporkan pada sebuah posisi beserta ukuran dan orientasi yang terkait. Setelah terdeteksi, ciri wajah seperti mata dan hidung didapatkan [9].

Faktor-faktor yang dihadapi pada proses pendeteksian wajah diantaranya adalah posisi wajah (tegak lurus, miring, dll), komponen wajah (kumis, jenggot, dll), ekspresi wajah (senyum, sedih, tertawa, dll), citra terhalang objek lain seperti kacamata, kondisi lingkungan (cahaya, kualitas kamera, dll). Pendeteksian wajah (*face detection*) adalah salah satu tahap awal yang harus dilakuakan sebelum dilakukannya proses pengenalan wajah (*face recognition*).

Metode yang digunakan untuk mendeteksi wajah lebih menekankan kepada teknik pembelajaran statistik dan penggunaan fitur penampilan, termasuk *real-time Viola-Jones face detector*, dimana metode ini dapat dikategorikan sebagai metode yang paling umum digunakan dalam mendeteksi wajah secara otomatis. *Viola-*

Jones face detector terdiri dari *classifier* yang di*training* oleh algoritma *AdaBoost* [10]. Sebuah *classifier* itu sendiri berasal dari hasil *training* ratusan sampel gambar terhadap objek tertentu. Setelah *classifier* di-*training*, *classifier* ini dapat diaplikasikan ke daerah-daerah dalam sebuah *input* gambar. Kemudian *classifier* ini akan menghasilkan *output* "1" pada daerah-daerah yang menyerupai objek yang digunakan |pada saat *classifier* di-*training*, dan akan menghasilkan *output* "0" pada daerah lain yang dianggap tidak menyerupai objek tersebut .

OpenCV hadir dengan *filter cascade classifier* yang terdiri dari lima jenis variasi pendeteksian wajah. Empat diantaranya dibuat berdasarkan *Viola-Jones*, menggunakan *Ada-Boost* dengan fitur *Haarlike*. *Classifier* yang kelima juga merupakan *classifier* yang di-*training* oleh *Ada-Boost*, tetapi menggunakan fitur *Local Binary Pattern* (LBP). Fitur LBP dikenalkan oleh Ojala et al untuk mendeskripsikan pola tekstur lokal. Fitur LBP membuat deskripsi dari *pixel* berukuran 3x3 dengan melakukan *threshold* pada *pixel* bagian luar menggunakan nilai pada *pixel* tengah.

2.4 Pengenalan Wajah (*Face Recognition*)

Pengenalan wajah adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem keamanan selain pengenalan retina mata, pengenalan sidik jari dan iris mata [11]. Dalam implementasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menambil citra wajah seseorang. Kemudian dibandingkan dengan wajah yang sebelumnya telah disimpan di dalam *database* model wajah. Terdapat banyak variabel pada pengenalan wajah, seperti citra sumber, citra hasil ekstraksi, cira hasil pengolahan dan data profil seseorang. Dibutuhkan juga alat 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.

2.5 Pengolahan Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra

digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai *real* maupun komplek yang direpresentasikan dengan deretan bit tertentu. Suatu citra dapat didefiisikan sebagai fungsi f(x,y) berkuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitude f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitude f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatan bahwa citra tersebut adalah citra digital [12].

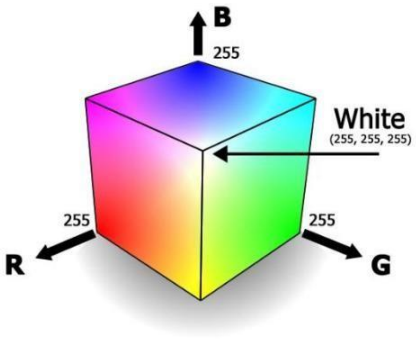
Ada tiga jenis citra yang umum digunakan dalam pemrosesan citra. Ketiga jenis citra tersebut yaitu citra berwarna, citra berskala keabuan, dan citra biner.

2.5.1 Citra Berwarna

Citra berwarna pada umumnya memiliki beberapa model warna diantaranya model warna RGB dan HSV :

2.5.1.1 Model Warna RGB

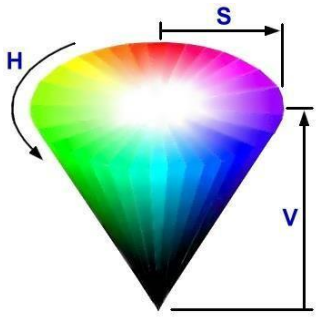
Citra berwarna, pada umumnya menggunakan model warna RGB yang terdiri dari 3 komponen warna primer, yakni R (merah), G (hijau), dan B (biru). Setiap komponen warna menggunakan delapan bit. (nilainya berkisar antara 0 sampai dengan 255). Dengan demikian, kemungkinan warna yang dapat disajikan mencapai 255 x 255 x 255 atau 16.581.375 warna [13].



Gambar 2.1 Model Warna RGB

2.5.1.2 Model Warna HSV

HSV adalah model warna yang lebih baik untuk digunakan untuk berbagai keperluan pengolahan citra dan *computer vision*. Misalnya saja pada *object tracking* berdasarkan warna, segmentasi citra dsb. Hue (H) adalah ukuran dari jenis warna seperti warna merah, kuning, hijau dan seterusnya. Representasinya dalam bentuk deraja dengan nilai 0 – 360. Saturasi (S) adalah keberwarnaan suatu warna. Semakin berwarna sebuah warna berarti semakin besar nilai saturasinya. Namun apabila suatu warna pucat, itu berarti saturasinya rendah. *Value* (V) adalah nilai kecerahan sebuah warna. Warna cerah memiliki nilai *Value* tinggi dan sebaliknya untuk warna yang gelap [13].



Gambar 2.2 Model Warna HSV

Konversi dari RGB ke HSV dapat dilakukan dengan persamaan **(2.1**)

$$V = \max R, G, B$$

$$V_m = \min R, G, B$$

$$S = \begin{cases} 0 \; jika \; V = 0 \\ \dfrac{V_m}{V} \quad jika \; V > 0 \end{cases}$$

$$H = \begin{cases} 0° \; jika \; S = 0 \\ 60° \times \left(\dfrac{G-B}{V_m} \; mod 6\right) \; jika \; V = R \\ 60° \times \left(2 + \dfrac{B-R}{V_m}\right) jika \; V = G \\ 60° \times \left(4 + \dfrac{R-G}{V_m}\right) jika \; V = B \end{cases} \quad \textbf{(2.1)}$$

2.5.2 Citra Berskala Keabuan

Sesuai dengan namanya, Citra jenis ini menangani gradasi warna hitam dan putih, yang tentu saja menghasilkan efek warna abu-abu. Pada jenis gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [13].

2.6 Metode Viola-Jones

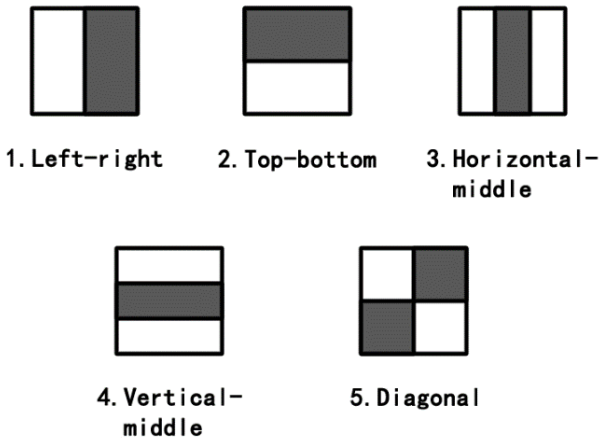
Saat ini telah banyak berkembang aplikasi-aplikasi yang menggunakan fitur deteksi wajah. salah satu algoritmanya adalah algoritma *Viola-Jones*. *Viola-Jones* telah memperkenalkan sebuah *framework* deteksi wajah yang mampu memproses gambar dengan sangat cepat dengan tingkat deteksi yang tinggi. *Viola-Jones* menerapkan algoritma *Adaboost* (*Adaptive Boosting*) algoritma yang dapat meningkatkan kinerja pendeteksian[14].

Dalam *framework Viola-Jones* disediakan banyak librari untuk melakukan proses seleksi fitur, fitur yang merupakan fungsi dasar untuk meningkatkan proses seleksi dikenal sebagai *Haar-Like Feature*. *Viola-Jones* terhambat oleh seleksi fitur yang harus berdasarkan pada setiap classifier yang lemah sehingga tergantung pada satu fitur tunggal. *Viola-Jones* memperkenalkan sebuah representasi *image* baru yang dikenal sebagai *Integral Image* yang tidak terpisahkan. Dengan metode ini *Haar-Like Feature* dapat dihitung pada setiap skala atau lokasi dalam waktu yang konstan[14].

Proses-proses yang ada pada *Viola-Jones* meliputi *Haar-Like Feature, Integral Image, Adaboost (Adaptive Boosting),* dan *Cascade Classifier*. Adapun detail dari setiap tahap yang dilalui dari sebuah citra saat proses pendeteksian wajah menggunakan algoritma *Viola-Jones* sebagai berikut:

2.6.1 Haar Like Feature

Haar Like Feature yaitu selisih dari jumlah piksel dari daerah di dalam persegi Panjang [14]. Contoh *Haar Like Feature* disajikan dalam gambar 2.3.



Gambar 2.3 Contoh *Haar Like Feature*

Nilai *Haar Like Feature* diperoleh dari selisih jumlah nilai piksel daerah gelap dengan jumlah nilai piksel daerah terang :

$$F(Haar) = \sum \boldsymbol{F_{White}} - \sum \boldsymbol{F_{Black}} \quad \textbf{(2.2)}$$

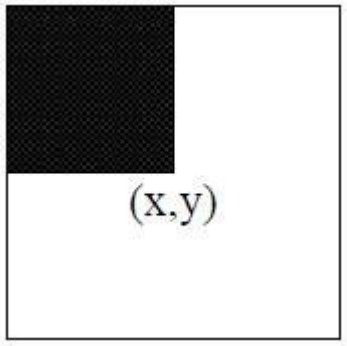
$F(Haar)$ = Nilai fitur total

$\sum \boldsymbol{F_{White}}$ = Nilai fitur pada daerah terang

$\sum \boldsymbol{F_{Black}}$ = Nilai fitur pada daerah gelap

2.6.2 Integral Image

Integral Image yaitu suatu teknik untuk menghitung nilai fitur secara cepat dengan mengubah nilai dari setiap piksel menjadi suatu representasi citra baru [14], sebagaimana disajikan dalam **Gambar 2.4**.



Gambar 2.4 ***Integral Image***

Berdasarkan **Gambar 2.4** citra integral pada titik (x,y) (ii(x,y)) dapat dicari menggunakan persamaan **(2.3)**

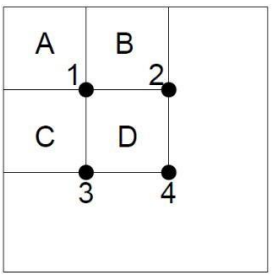
$$\boldsymbol{ii}(\boldsymbol{x},\boldsymbol{y}) = \sum x' \leq x, y' \leq y \; \boldsymbol{i}(\boldsymbol{x}',\boldsymbol{y}') \qquad \textbf{(2.3)}$$

Keterangan:

$\boldsymbol{ii}(\boldsymbol{x},\boldsymbol{y})$ = citra integral pada lokasi x,y

$\boldsymbol{i}(\boldsymbol{x}',\boldsymbol{y}')$ = Nilai piksel pada citra asli

Perhitungan nilai dari suatu fitur dapat dilakukan secara cepat dengan menghitung nilai citra integral pada empat buah titik sebagaimana disajikan dalam **Gambar 2.5**.



Gambar 2.5 Perhitungan Nilai Fitur

Jika nilai *integral image* titik 1 adalah A, titik 2 adalah A+B, titik 3 adalah A+C, dan di titiki 4 adalah A+B+C+D, maka jumlah piksel di daerah D dapat diketahui dengan cara 4 + 1 – (2 + 3).

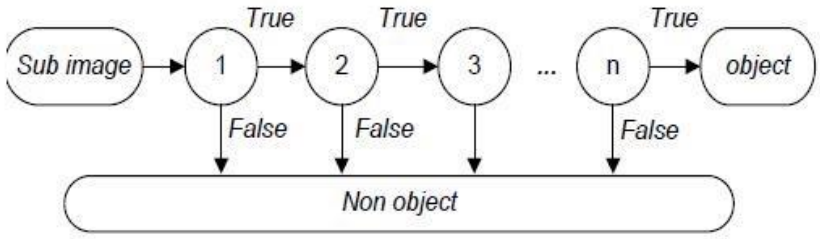
Algoritma *Adaboost learning,* digunakan untuk meningkatkan kinerja klasifikasi dengan pembelajaran sederhana untuk menggabungkan banyak *classifier* lemah menjadi satu *classifier* kuat. *Classifier* lemah adalah suatu jawaban benar dengan tingkat kebenaran yang kurang akurat.

2.6.3 Adaptive Boosting

Untuk memilih fitur yang spesifik yang akan digunakan dan untuk mengatur nilai ambangnya (*threshold*), *Viola-Jones* menggunakan sebuah *machine learning* yang disebut *AdaBoost*. *AdaBoost* menggabungkan banyak *classifier* lemah untuk membuat sebuah *classifier* kuat. Lemah disini berarti urutan *filter* pada classifier hanya mendapatkan jawaban benar lebih sedikit. Jika keseluruhan *classifier* lemah digabungkan maka akan menjadi *classifier* yang lebih kuat. *AdaBoost* memilih sejumlah *classifier* lemah untuk disatukan dan menambahkan bobot pada setiap *classifier*, sehingga akan menjadi *classifier* yang kuat. *Viola-Jones* menggabungkan beberapa *AdaBoost classifier* sebagai rangkaian *filter* yang cukup efisien yang cukup efisien untuk menggolongkan daerah *image*. Masing-masing *filter* adalah satu *AdaBoost classifier* terpisah yang terdiri *classifier* lemah atau satu *filter* fitur[14].

2.6.4 Cascade Classifier

Cascade classifier adalah sebuah metode untuk mengkombinasikan *classifier* yang kompleks dalam sebuah struktur bertingkat yang dapat meningkatkan kecepatan pendeteksian obyek dengan memfokuskan pada daerah citra yang berpeluang saja [14]. Struktur *cascade classifier* terdapat pada **Gambar 2.6**.



Gambar 2.6 *Cascade Classifier*

Gambar 2.6 menjelaskan proses penyeleksian keberadaan obyek. Diasumsikan suatu *sub image* di evaluasi oleh *classifier* pertama dan berhasil melewati *classifier* tersebut, hal ini mengindikasikan *sub image* berpotensi terkandung obyek dan dilanjutkan pada *classifier* ke dua sampai dengan ke-n, jika berhasil melewati keseluruhan *classifier*, maka disimpulkan terdapat obyek yang dideteksi. Jika tidak, proses evaluasi tidak dilanjutkan ke *classifier* berikutnya dan disimpulkan tidak terdapat obyek.

2.7 Image Scaling

Dalam komputer grafik, image scaling mengacu kepada perubahan ukuran pada citra digital. Saat menskala citra grafik vektor, grafik primitif yang membentuk citra dapat diskalakan menggunakan transformasi geometris, tanpa kehilangan kualitas citra. Saat menskala citra grafis raster, citra baru dengan jumlah piksel lebih tinggi atau lebih rendah harus dihasilkan. Dalam kasus penurunan jumlah piksel (penskalaan) ini biasanya menghasilkan penurunan kualitas yang terlihat. Dari sudut pandang pemrosesan sinyal digital, penskalaan grafik raster adalah contoh dua dimensi dari konversi laju sampel, konversi sinyal diskrit dari laju pengambilan sampel (dalam hal ini laju pengambilan sampel lokal) ke yang lain[15].

2.8 Local Binary Pattern

LBP adalah metode analisis tekstur yang menggunakan model statistika dan struktur [16]. LBP pertama kali diperkenalkan oleh Timo Ojala. Operator LBP menggunakan perbandingan nilai keabuan dari piksel-piksel ketetanggaan. Operator dasar LBP berukuran 3 x 3 menggunakan 8 piksel ketetanggan in dari sebuah piksel tengah ic. Piksel ketetanggaan ke-n tersebut di-*threshold* menggunakan nilai keabuan dari piksel tengah seperti yang ditunjukkan pada persamaan **(2.4)** dan fungsi *thresholding* s(x) seperti yang ditunjukkan pada persamaan **(2.5)**. Kode binary hasil operator LBP piksel ketetanggaan akan digunakan untuk merepresentasikan fitur dari piksel tengah ic [17].

$$LBP(xc, yc) = \sum s(in - ic)2^n \quad \textbf{(2.4)}$$

$$s(x) = \begin{cases} 1, if \ \ x \geq 0 \\ 0, if \ \ x < 0 \end{cases} \quad \textbf{(2.5)}$$

Menurut Biglari [18] kelebihan dari LBP adalah mudah diimplementasikan dan tingkat komputasinya lebih rendah sehingga tidak membutuhkan waktu yang lama dalam ekstraksi fitur.

Ahonen et al pada tahun 2004 menerapkan algoritma LBPH untuk melakukan pengenalan wajah dan mendapatkan hasil yang sangat baik dengan menggunakan database FERET. Dalam metode mereka, pertama-tama LBP histogram diekstrak terlebih dahulu, dan setelah itu gambar wajah dipecah menjadi bagian yang lebih kecil. Kemudian gambar wajah digabungkan kembali menjadi satu, dan dilakukan peningkatan histogram yang menggambarkan tekstur lokal dan bentuk wajah secara global. Pengenalan ini dilakukan menggunakan *classifier nearest-neighbor*[19]. Selanjutnya, Zhang et al [20] mengindentifikasi dua kelemahan dari pendekatan Ahonen. Pertama, ukuran dari wilayah pada metode Ahonen terbatas atas posisi dan ukuran dari area lokal. Kedua, bobot dari daerah dioptimalkan secara manual. Oleh karena itu, Zhang dan kawan-kawan mengusulkan untuk menggunakan classifier boosting untuk memilih histogram diskriminatif dari area yang diperoleh dengan mengekstrasi LBP histogram dengan cara menggeser dan mengatur ukuran area disekitar intra-personal dan inter-personal wajah. Jika dibandingkan dengan metode yang dilakukan oleh Ahonen, metode ini lebih sedikit menggunakan histogram dan menghasilkan akurasi yang sama.

2.9 Open*CV* (*Open Source Computer Vision Library*)

OpenCV (*Open Source Computer Vision Library*) adalah sebuah *library* perangkat lunak yang ditujukan untuk pengolahan citra yang biasa digunakan secara *real-time* (pada waktu itu juga). *Library* ini dibuat oleh Intel dan merupakan *library* yang bebas digunakan dan berada dalam naungan sumber terbuka (*Open source*) dari lisensi. *Library* ini juga bisa digunakan diberbagai platform dan didedikasikan sebagian besar untuk pengolahan citra secara real-

time. Umumnya library ini menggunakan bahasa pemrograman C/C++, tetapi akhir – akhir ini sudah dikembangkan keberbagai bahasa pemrograman seperti *Phyton, Javascript,* dan *Java* [21].

Secara garis besar *OpenCV* mempunyai modul/subrutin yang ada pada *library* yang akan dijelaskan sebagai berikut[22].

1. *Core* – sebuah modul/subrutin dasar dari struktur data, sudah termasuk *array* multi dimensi dan matriks.
2. *Imgproc* – sebuah modul/subrutin untuk pemprosesan citra seperti *image filtering*, *geometrical image*, *image transformation*, dan *color space conversion*.
3. *Video* – sebuah modul/subrutin untuk analisis video termasuk *motion*, *background subtraction* dan *object tracking algorithm*.
4. *Calib3d* – sebuah modul/subrutin untuk geometry algorithm, kalibrasi kamera dan elemen untuk membangun gambar 3D (3-Dimensional).
5. *Features2d* – sebuah modul/subrutin untuk perhitungan konvolusi dan ekstraksi fitur.
6. *Objdetect* – sebuah modul/subrutin untuk deteksi objek dari kelas yang sudah ditentukan
7. *Highgui* – sebuah modul/subrutin untuk menangkap kamera *webcam* dan *image* and *video codecs*.
8. *Ml* – sebuah modul/subrutin tambahan untuk melakukan perhitungan *Machine learning* seperti *K-nearest Neighbors*, *Support Vector Machine* dan *Decision Tree*.

2.10 UML (Unified Modelling Language)

UML adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Abstraksi konsep dasar UML terdiri dari structural classification, dynamic behavior, dan model management dapat kita pahami *main concepts* sebagai term yang akan muncul pada saat membuat diagram dan *view* adalah kategori dari diagram tersebut. UML mendefinisikan beberapa

diagram antara lain: *Class Diagram, Package Diagram, Use Case Diagram, Sequence Diagram, Communication Diagram, Statechart Diagram, Activity Diagram, Component Diagram* dan *Deployment Diagram*[23]. Dalam pengembangan aplikasi ini, hanya menggunakan 4 diagram UML yaitu *Use Case Diagram, Activity Diagram,* dan *Sequence Diagram* karena dapat membantu penelitian selanjutnya dalam mengembangkan sistem ini.

2.10.1 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu[23]. Adapun komponen komponen dalam *use case* diagram diantaranya:

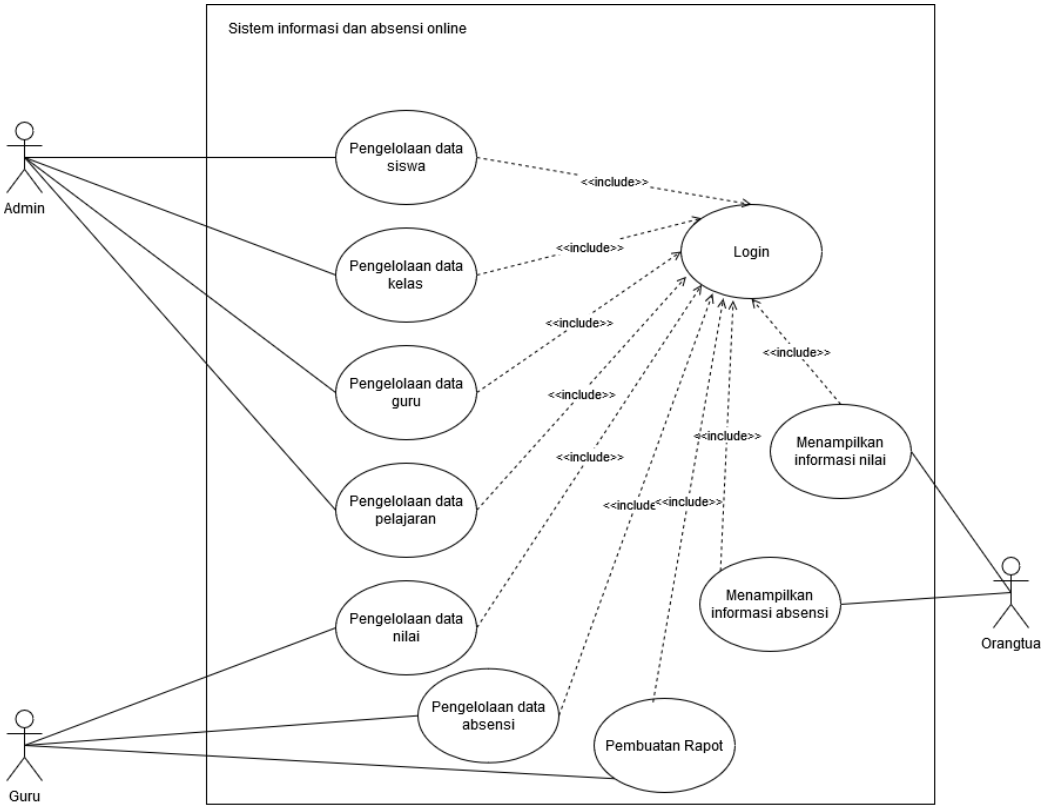
1. Aktor

   Aktor merupakan suatu entitas yang berkaitan dengan sistem tapi bukan dari bagian dalam sistem itu sendiri. Aktor berbeda diluar sistem namun berkaitan erat dengan fungsionalitas didalamnya. Aktor dapat memiliki hubungan secara langsung terhadap fungsi utama baik terhadap salah satu atau semua fungsionalitas utama. Aktor juga dapat dibagi terhadap berbagai jenis atau tingkatan dengan cara digeneralisasi atau dispesifikasi tergantung kebutuhan sistemnya. Aktor biasanya dapat berupa pengguna atau database yang secara pandang berada dalam suatu ruang lingkup sistem tersebut.

2. *Use Case*

   Use case merupakan gambaran umum dari fungsi proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *use case*, tetapi yang digambarkan hanya fungsionalitas utama yang berkaitan dengan sistem. *Use case*

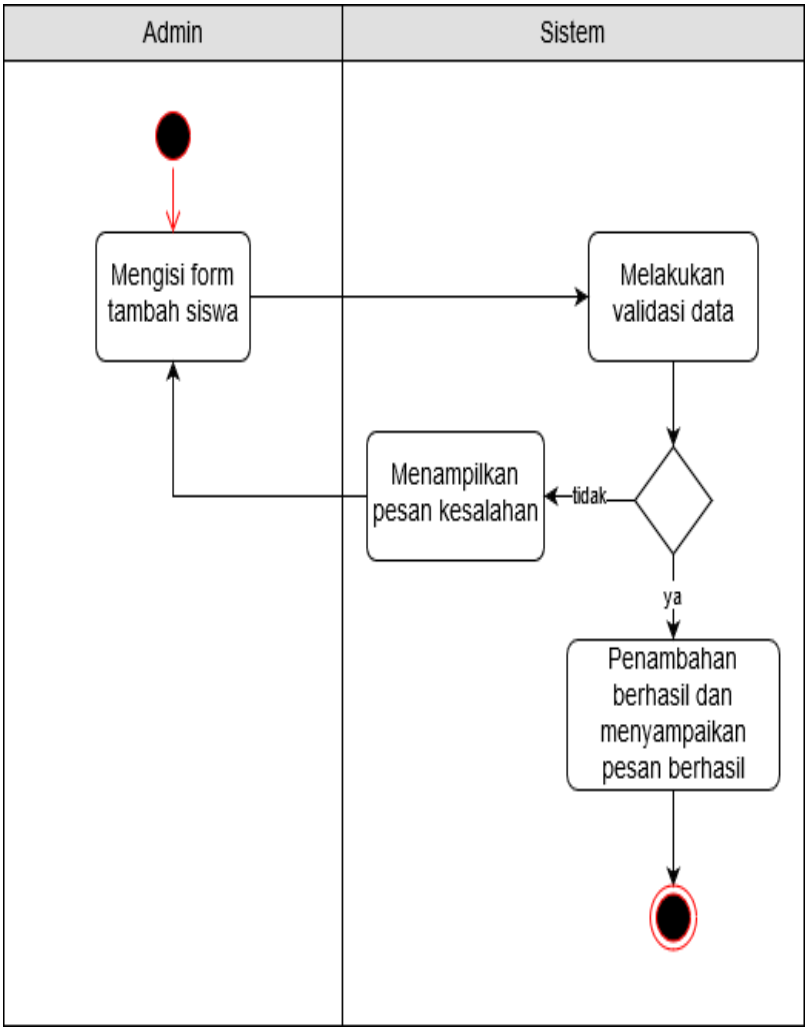
menitikberatkan bagaimana suatu sistem dapat berinteraksi baik antar sistem maupun di luar sistem.



Gambar 2.7 *Use Case Diagram*

2.10.2 Activity Diagram

Activity diagram menggambarkan berbagai alur aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, keputusan yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah aksi dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar sub sistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari level atas secara umum[23].



Gambar 2.8 *Activity Diagram*

2.10.3 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, tampilan, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktifitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan[23].

2.11 Python

Python adalah bahasa pemrograman interpretatif yang dianggap mudah dipelajari. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Dengan ata lain, *python* diklaim sebagai bahasa pemrograman yang memiliki kode-kode pemrograman yang sangat jelas, lengkap, dan mudah untuk dipahami. Python secara umum berbentuk pemrograman beriorientasi objek, imperatif, dan fungsional[24].

Python dikembangkan oleh Guido van Rossum (*programmer* kelahiran belanda) pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Sampai saat ini *Python* masih dikembangkan oleh *Python Software Foundation*. Bahasa *Python* mendukung hampir semua sistem operasi, bahkan untuk sistem operasi *Linux*, hampir semua distronya sudah menyertakan *Python* di dalamnya. Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari *syntax error* [24].

2.12 Extensible Markup Language (XML)

XML adalah data berformat hirarki yang berfungsi sebagai tempat pertukaran data di dalam *world wide web* (www). Sebuah dokumen XML terdiri dari struktur elemen yang berulang, mulai dari elemen yang paling awal atau biasa disebut root. Elemen data tersebut dapat diisi dengan atribut atau dapat juga diisi dengan sub-elemen[25].

```
<book>
  <booktitle> The Selfish Gene </booktitle>
  <author id = "dawkins">
        <name>
            <firstname> Richard </firstname>
            <lastname> Dawkins </lastname>
        </name>
        <address>
            <city> Timbuktu </city>
            <zip> 99999 </zip>
        </address>
  </author>
</book>
```

Gambar 2.9 Contoh Struktur XML

Pada **Gambar 2.9** elemen book merupakan elemen paling awal atau disebut *root*. Elemen *book* sendiri mempunyai dua sub-elemen *book title* dan *author*. Sama seperti HTML, XML merupakan kumpulan dari SGML. Namun, ketika tag HTML mempunyai tujuan utama untuk menampilkan data item, tag XML justru mendeskripsikan dirinya sendiri. Walaupun perbedaan ini sangat tipis, namun hal ini cukup penting, karena data XML berfokus pada pendeskripsian isi dari XML itu sendiri, oleh karena itu hal ini memungkinkan program untuk menginterpretasi data dari XML tersebut[25].