BAB 2
TINJAUAN PUSTAKA

2.1 Citra

Definisi citra menurut Kamus Webster adalah suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda [6]. Secara umum citra dapat dibagi menjadi dua macam, yaitu :

a. Citra kontiyu, yaitu citra yang dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog.
b. Citra diskrit, yaitu citra yang dihasilkan melalui proses digitalisasi terhadap citra kontiyu dan direpresentasikan sebagai sebuah matrik yang masing-masing elemennya merepresentasikan nilai intensitas.

Pada bidang komputer, ada tiga bidang studi yang berkaitan dengan data citra, yaitu:

2.1.1 Grafika Komputer

Grafika komputer dapat diartikan sebagai separangkat alat yang terdiri dari hardware dan software untuk membuat gambar, grafik atau citra realistik untuk seni, game komputer, foto dan animasi komputer.

Grafika komputer bertujuan menghasilkan citra dengan bentuk-bentuk primitif geometri seperti garis, lingkaran, dan sebagainya. Primitif-primitif geometri tersebut memerlukan data deskriptip untuk melukiskan elemen-elemen gambar. Contoh data deskriptif adalah koordinat titik, panjang garis, jari-jari, tebal garis dan sebagainya. Grafika komputer mempuyai peranan penting dalam visualisasi dan virtual reality.



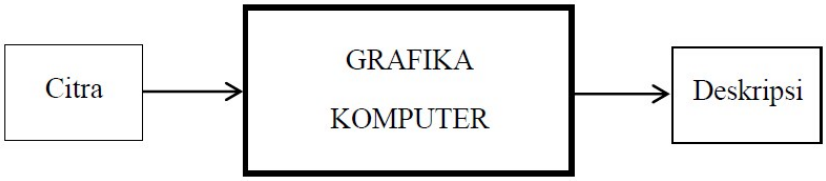
Gambar 2. 1 Alur Grafika Komputer [6]

2.1.2 Pengolahan Citra

Secara umum, pengolahan citra digital menunjukan pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. sudah diimplementasikan secara luas, khususnya dibidang keilmuan maupun industri seperti kedokteran, keamanan, pertahanan, geologi, biologi, sistem kontrol fabrikasi, dan lain sebagainya.

2.1.3 Pengenalan Pola

Mengelompokan data numerik dan simbolik (citra) secara otomatis oleh mesin (komputer). Tujuan pengelompokan adalah untuk mengidentifikasikan suatu objek berdasarkan citra yang dihasilkan.



Gambar 2. 2 Alur Pengenalan Pola [6]

2.2 Digitalisasi Citra

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari fungsi malar (kontiyu) menjadi nilai-nilai diskrit disebut digitalisasi. Citra yang dihasilkan inilah yang disebut citra digital (Digital Image). Pada umumnya citra digital berbentuk empat persegipanjang dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (atau lebar x panjang). Citra digital tingginya N, lebarnya M, dan memiliki L derajat dapat dianggap sebagai fungsi:

$$f(x,y) \quad \begin{matrix} 0 \le x \le N \\ 0 \le y \le M \\ 0 \le f \le L \end{matrix}$$

Citra digital yang berukuran N x M lazim dinyatakan dengan matriks yang berukuran N baris dan M kolom sebagai berikut :

$$(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M) \\ f(1,0) & f(1,1) & \dots & f(1,M) \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Indek baris (i) dan index kolom (j) menyatakan suatu koordinat titik pada citra, sedangkan f(i,j) merupakan intensitas (derajat keabuan) pada titik (i,j).

Masing-masing elemen pada citra digital (berarti elemen matriks) disebut image element, picture element atau pixel atau pel. Jadi, citra yang berukuran N x M mempuyai NM buah pixel. Sebagai contoh, misalkan sebuah citra berukuran 256 x 256 pixel dan direpresentasikan secara numeric dengan matriks yang terdiri dari 256 buah baris (di-indeks dari 0 sampai 255) dan 256 buah kolom (di-indeks dari 0 sampai 255) seperti contoh berikut:

$$\begin{bmatrix} 0 & 134 & 145 & \dots & \dots & 231 \\ 0 & 167 & 201 & \dots & \dots & 197 \\ 220 & 187 & 189 & \dots & \dots & 120 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 221 & 219 & 210 & \dots & \dots & 156 \end{bmatrix}$$

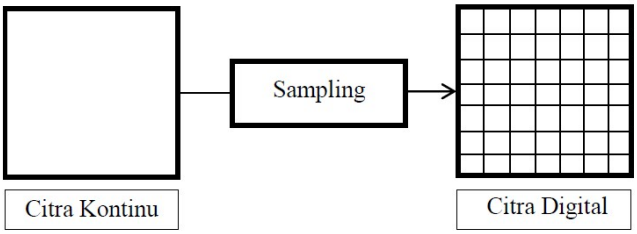
Pixel pertama pada koordinat (0,0) mempuyai nilai intensitas 0 yang berarti warna pixel tersebut hitam dan putih, dan seterusnya.

Proses digitalisasi citra ada dua macam:

a. Digitalisasi spasial (x,y), sering disebut sebagai penerokan (Sampling).
b. Digitalisasi intensitas f(x,y), sering disebut sebagai kuantisasi.

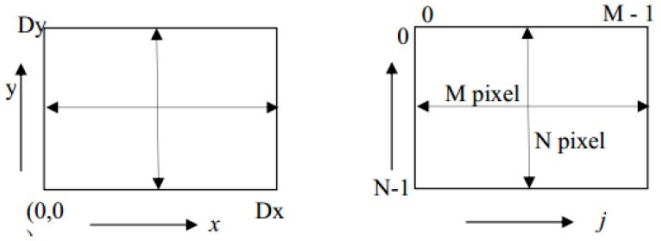
2.3 Sampling

Sampling, merupakan proses pengambilan informasi dari suatu titik pixel pada grid-grid yang berbentuk bujursangkar (kisi-kisi dalam horizontal dan vertikal). Diperlihatkan dalam gambar dibawah ini.



Gambar 2. 3 Sampling Terhadap Suatu Citra [6]

Terdapat perbedaan antara koordinat gambar (yang di sampling) dengan koordinat matriks (hasil digitalisasi). Titik asal (0,0) pada gambar dan elemen (0,0) pada matriks tidak sama. Koordinat x dan y pada gambar dimulai dari sudut kiri bawah, sedangkan penomoran pixel pada matriks dimulai dari sudut kiri atas.



Gambar 2. 4 Koordinat Pixel [6]

Dalam hal ini :

$i = x, 0 \leq j \leq N - 1$

$j = (M - y), 0 \leq j \leq M - 1$

$x = D_x/N\ increment$

$y = D_y/N\ increment$

$N = jumlah\ maksimum\ pixel\ dalam\ satu\ baris$

$M = jumlah\ maksimum\ pixel\ dalam\ satu\ kolom$

$D_x = lebar\ gambar\ (dalam\ inchi)$

$D_y = tinggi\ gambar\ (dalam\ inchi)$

Elemen (i,j) didalam matriks menyatakan rata-rata intensitas cahaya pada area citra yang direpresentasikan oleh pixel. Sebagai contoh, tinjau citra binear yang hanya mempuyai 2 derajat keabuan, 0 (hitam) dan 1 (putih). Sebuah gambar yang berukuran 10x10 inchi dinyatakan dalam matriks yang berukuran 5x4, yaitu lima baris dan 4 kolom. Tiap elemen gambar lebarnya 2.5 inchi dan tingginya 2 inchi akan diisi dengan sebuah nilai bergantung pada rata-rata intensitas cahaya pada area tersebut.

Area 2.5 x 2.0 inchi pada sudut kiri atas gambar dinyatakan dengan lokasi (0,0) pada matriks 5 x 4 yang mengandung nilai 0 (yang berarti tidak ada intensitas cahaya). Area 2.5 x 2.0 inchi pada sudut kanan bawah gambar dinyatakan dengan lokasi (4,3) pada matriks 5 x 4 yang mengandung nilai 1 (yang berarti iluminasi maksimum).

Untuk memudahkan implementasi, jumlah sampling biasanya diasumsikan perpangkatan dari dua yang dalam hal ini,

$N = jumlah\ sampling\ pada\ suatu\ baris/kolom$

$n = bilangan\ bulat\ positif$

Contoh ukuran penerokan: 256 x 256 pixel, 128 x 256 pixel. Pembagian gambar menjadi ukuran tertentu menentukan resolusi (yaitu derajat rincian yang dapat dilihat) spasial yang diperoleh. Semakin tinggi resolusinya yang berarti semakin kecil ukuran pixel (atau semakin banyak jumlah pixel-nya), semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokan derajat keabuan pada penerokan semakin kecil.

2.4 Elemen-elemen Citra Digital

Citra digital mengandung sejumlah elemen-elemen dasar. Elemen-elemen dasar tersebut dimanipulasi dalam pengolahan citra dan dieksploitasi lebih lanjut dalam computer vision. Elemen-elemen dasar yang penting diantaranya adalah:

2.4.1 Kecerahan (*brightness)*

Kecerahan adalah kata lain untuk intensitas cahaya. Sebagaimana telah dijelaskan pada bagian penerokan, kecerahan pada sebuah titik (pixel) di dalam citra bukanlah intensitas yang sebenarnya, tetapi sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya. Sistem visual manusia mampu menyesuaikan dirinya dengan tingkat kecerahan (brightness level) mulai dari yang paling rendah sampai yang paling tinggi dengan jangkauan sebesar .

2.4.2 Kontras (*contrast)*

Kontras menyatakan sebaran terang (lightness) dan gelap (darkness) di dalam sebuah gambar. Citra dengan kontras rendah dicirikan oleh sebagian besar komposisi citranya adalah terang atau sebagian besar gelap. Pada citra dengan kontras yang baik, komposisi gelap dan terang tesebar secara merata.

2.4.3 Kontur (*countour)*

Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada pixel-pixel yang bertentangga. Karena adanya perubahan intensitas inilah mata kita mampu mendeteksi tepi-tepi (edge) objek di dalam citra.

2.4.4 Warna (*color*)

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempuyai panjang gelombang (λ) yang berbeda. Warna merah mempuyai panjang gelombang paling tinggi, sedangkan warna ungu (violet) mempuyai panjang gelombang paling rendah.

Warna-warna yang diterima oleh mata (sistem visual manusia) marupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah red (R), green (G), blue (B).

Persepsi sistem visual manusia terhadap warna sangat relatif sebab dipengaruhi oleh banyak kriteriam, salah satunya disebabkan oleh adaptasi yang menimbulkan distorsi. Misalnya bercak abu-abu disekitar warna hijau akan tampak keungu-unguan (distorsi terhadap ruang), atau jika mata melihat warna hijau lalu langsung dengan cepat melihar warna abu-abu, maka mata menangkap kesan warna abu-abu tersebut sebagai warna ungu (distor terhadap waktu).

2.4.5 Bentuk (*shape*)

Shape berarti properti intrinsic dari objek tiga dimensi dengan pengertian bahwa shape merupakan properti intrinsik utama untuk sistem visual manusia. Manusia lebih sering mengasosiasikan objek dengan bentuknya ketimbang elemen lainnya (warna misalnya). Pada umumnya, citra yang dibentuk oleh mata merupakan citra dwimatra (2 dimensi), sedangkan objek yang dilihat umumnya berbentuk trimata (3 dimensi).

Informasi bentuk objek dapat diekstrasikan dari citra pada permulaan pra-pengolahan dan segmentasi citra. Salah satu tantangan utama pada computer vision adalah merepresentasikan bentuk, atau aspek-aspek penting dari bentuk.

2.4.6 Tekstur (*texture)*

Tekstur dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan pixel-pixel yang bertetanggaan. Jadi, tekstur tidak dapat didefinisikan untuk sebuah pixel. Sistem visual manusia pada hakekatnya tidak menerima informasi citra sebagai independen pada setiap pixel, melainkan suatu citra dianggap sebagai suatu kesatuan. Resolusi citra yang diamati ditemtukan oleh skala pada mana tekstur tersebut dipersepsi.

Sebagai contoh, jika kita mengamati citra lantai berubin dari jarak jauh, maka kita mengamati bahwa tekstur terbentuk oleh penempatan ubin-ubin secara keseluruhan, bukan dari persepsi pola di dalam ubin itu sendiri. Tetapi, jika kita mengamati citra yang sama dari jarak yang dekat, maka hanya beberapa ubin yang tampak dalam bidang pengamatan, sehingga kita mempersepsi bahwa teksture terbentuk oleh penempatan pola-pola rinci yang menyusun tiap ubin.

2.5 Representasi Gerak

Gerak adalah perubahan posisi suatu benda terhadap titik acuan. Titik acuan sendiri didefinisikan sebagai titik awal atau titik tempat pengamatan.

Berdasarkan definisi diatas dapat diartikan bahwa gerak merupakan perpindahan objek dari satu titik ke titik lainnya berdasarkan batas nilai tertentu, sebagai contoh sebagai berikut :

Objek yang berada di dalam lingkarang, jika objek berpindah posisi keluar dari lingkarang tersebut maka dikatakan objek tersebut bergerak.

2.6 Deteksi Gerak

Fungsi utama dari motion detector / deteksi gerakan adalah untuk mengidentifikasikan apakah benda tersebut dalam keadaan bergerak atau diam.

Secara sederhana deteksi gerakan dapat dilakukan dengan mencari perbedaan antara dua citra yang berurutan pada hasil pencitraan menggunakan webcam. Metode perhitungan didasarkan pada perbandingan pixels citra saat ini dengan citra sebelumnya. Bila terdapat perbedaan pixels pada kedua gambar yang melebihi ambang batas yang sudah ditetapkan maka dapat disimpulkan adanya gerakan

2.7 Metode Deteksi Gerak *Spatial Domain*

Deteksi gerakan yang digunakan diaplikasi pengawas pendeteksi gerakan ini adalah pemrosesan citra spatial domain dengan point processing. Motede ini berkerja dengan cara sebagai berikut :

a. Penetapan sebuah citra referensi

   Sebuah citra yang diambil oleh kamera pada saat situasi lingkungan yang stabil ditetapkan sebagai citra referensi. Citra ini kemudian disimpan sebagai acuan dalam membandingkan citra.

b. Pengambilan citra secara periodik

   Setiap jangka waktu tertentu, sebuah citra diambil oleh kamera. Citra ini kemudian disimpan untuk perhitungan selanjutnya.

c. Pendeteksi Gerak

   Citra yang diambil secara periodik tadi dibandingkan dengan referensi dengan cara membandingkan setiap pixel yang berada dilokasi yang sama. Perhitungan perbandingan dilakukan dengan cara mengambil nilai absolute dari hasil pengurangan nilai RGB dari 2 pixel yang berbeda dilokasi yang sama. Jika nilai dari perhitungan tersebut melewati suatu batas nilai tertentu maka pixel tersebut dinyatakan sebagai pixel yang terdeteksi gerakan.

$$f(x,y) = \begin{cases} 1 \; if \; T1 < ABS[p(x,y) - q(x,y)] \\ 0 \; if \; T1 \geq ABS[p(x,y) - q(x,y)] \end{cases}$$

Persamaan pendeteksi gerak pada metode spatial domain dimana :

f = *Array 2* dimensi yang digunakan untuk menyampaikan hasil dari perhitungan

x = Posisi *pixel* terhadap sumbu x.

y = Posisi *pixel* terhadap sumbu y.

T1 = Batas nilai untuk membatasi banyaknya perubahan nilai *pixel* tersebut dinyatakan terhadap gerakan (*Thresold).*

ABS = Fungsi absolute.

p = Citra referensi

q = Citra yang diambil secara periodik (citra yang dibandingkan).

Variabel T1 merupakan sebuah nilai batas ambang perubahan 2 buah pixel diletak yang sama. Penggunaan variabel T1 ini memiliki dua tujuan. Tujuan yang pertama adalah untuk mengeliminasi noise yang terjadi, sedangkan tujuan yang kedua adalah untuk memberikan batas ambang perubahan intensitas warna antara dua buah pixel. Noise terjadi karna suatu lingkungan yang pencahayaannya labil. Hal ini dapat terjadi karena intensitas cahaya yang datang tidak selalu sama dari waktu ke waktu.

Sebuah array boolean f(x,y) digunakan untuk menyimpan nilai dari perhitungan. Jika nilai f(x,y) bernilai true berarti dua buat pixel dari 2 buah citra yang berbeda dengan letak yang sama terjadi perubahan warna yang melewati batas ambang T1. Sebaliknya, jika f(x,y) bernilai false berarti kedua buah pixel tersebut tidak terjadi perubahan warna atau perubahan yang terjadi tidak melewati batas ambang T1. Array ini digunakan untuk menghitung jumlah pixel yang berubah. Jika jumlah pixel yang berubah melebihi nilai batas ambang T2 maka citra tersebut dikatakan terdapat gerakan.

$$res = f(x) = \begin{cases} 1 \; if \; T2 < \sum_x \sum_y f(x,y) \\ 1 \; if \; T2 \geq \sum_x \sum_y f(x,y) \end{cases}$$

Persamaan perhitungan jumlah pixel pada metode spatial domain dimana :

res = Hasil deteksi gerakan, bernilai 1 jika terdapat gerakan, dan bernilai 0 jika tidak terdapat gerakan.

T2 = Batas nilai untuk membatasi banyak jumlah pixel yang berubah sebelum sebuah citra dinyatakan terdapat gerakan (*Thresold).*

x = Posisi *pixel* terhadap sumbu x.

y = Posisi *pixel* terhadap sumbu y

f = *Array* hasil perhitungan pendeteksi citra.

Terdapat dua metode yang dapat digunakan untuk mendapatkan sebuah citra sebagai citra referensi. Metode yang pertama, citra referensi hanya diambil satu kali untuk pendeteksian seterusnya, sedangkan metode kedua, citra referensi diambil dari citra pembanding setiap kali pembanding telah lolos dari deteksi gerakan.

Metode yang pertama digunakan apabila lingkungan yang dideteksi tidak ada perubahan cahaya selama proses deteksi gerakan berlangsung, misalnya di dalam ruangan. Metode kedua digunakan apabila pencahayaan lingkungan yang dideteksi dapat berubah – ubah. Metode ini dapat digunakan pada lingkungan yang terpengaruhi oleh cahaya matahari. Perubahan sinar matahari tidak dianggap sebagai gerakan karena perubahan yang sangat lambat dank arena adanya variabel T1. Penggunaan metode kedua ini dapat berakibat tak terdeteksinya sebuah gerakan yang sangat lambat, metode ini hanya dapat mengenali perubahan yang ekstrim. Apabila proses pendeteksian termasuk untuk mendeteksi gerakan yang sangan lambat maka merode kedua ini tidak dapat digunakan, atau dapat digunakan tetapi dengan mengurangi batas ambang T1 , sehingga perubahan kecil dapat terdeteksi oleh sistem.

2.8 Video Digital

Video digital pada dasarnya tersusun atas rangkaian frame. Rangkaian frame tersebut ditampilkan pada layar dengan keceparan tertentu, tergantung pada frame rare yang diberikan (dalam frame/second). Jika frame rare cukup tinggi, mata manusia tidak dapat menangkap gambar atau frame, melainkan menangkapnya sebagai rangkaian yang kontinu (video).

Masing - masing frame merupakan citra digital. Suatu citra digital direpresentasikan dengan sebuah matriks yang masing – masing elemennya merepresentasikan nilai intensitas. Jika I adalah matriks dua dimensi, I(x,y) adalah nilai intensitas yang sesuai pada posisi baris x dam kolom y pada matriks tersebut. Titik – titik ditempatkan image di sampling disebut picture elements, atau sering dikenal sebagai piksel.

Karakteristik suatu video digital ditentukan oleh resolusi (resolution) atau frame dimention, pixel depth, dan frame rate. Karakteristik – karakteristik ini akan menetukan tawar menawar antara kualitas video jumlah bit yang dibutuhkan untuk menyimpan atau mentransmisikannya.

2.8.1 Resolusi

Resolusi (resolution) atau frame dimention adalah ukuran sebuah frame. Resolusi dinyatakan dalam piksel x piksel. Semakin tinggi resolusi, semakin baik kualitas video yang dihasilkan, artian bahwa ukuran fisiknya sama, video dengan resolusi tinggi akan lebih detail. Namun resolusi yang tinggi akan mengakibatkan jumlah bit yang diperlukan untuk menyimpan atau mentrasmisikannya meningkat.

2.8.2 Bit Depth

Bit depth menentukan jumlah bit yang digunakan untuk merepresentasikan tiap piksel pada sebuah frame. Kedalaman bit dinyatakan dalam bit/pixel. Semakin banyak jumlah bit yang digunakan untuk merepresentasikan sebuah piksel, yang berarti semakin tinggi kedalaman pikselnya, maka semakin baik pula kualitasnya, dengan konsekuensi jumlah bit yang diperlukan menjadi lebih tinggi. Dengan 1 byte (8bit) untuk setiap piksel, diperoleh 28 atau 256 level intensitas. Dengan level intensitas sebanyak itu, umumnya mata manusia sudah dapat dipuaskan. Kedalaman piksel paling rendah terdapat pada binary value image yang hanya menggunakan 1 bit/pixel, sehingga hanya ada dua kemungkinan bagi tiap piksel, yaitu 0 untuk hitam dan 1 untuk putih. Pada kenyataannya semakin sedikit jumlah bit yang digunakan untuk tiap piksel, maka kualitas gambar akan semakin turun.

2.8.3 Frame Rate

Frame rate menunjukan jumlah frame yang digambarkan tiap detik, dan dinyatakan dengan frame/second. Sehubungan dengan frame rate ini ada dua hal yang perlu diperhatikan, yaitu kehalusan gerakan (smooth motion) dan kilatan (flash). Kehalusan gerakan ditentukan oleh jumlah frame yang berbeda per detik. Untuk mendapatkan gerakan yang halus, video setidaknya harus menampilkan sedikitnya 25 frame/second. Kilatan ditentukan oleh jumlah berapa kali layar digambar perdetik (frame rate), dengan 29 frame/second kilatan sudah dapat dilenyapkan.

Video yang berkualitas baik akan memiliki frame rate yang tinggi, setidaknya sesuai dengan mata manusia, yang berarti membutuhkan jumlah bit yang lebih tinggi.

2.10 Webcam

Webcam adalah kamera yang mampu menangkap data berupa gambar dan suara. *Webcam* adalah sebuah kamera video digital kecil yang dihubungkan ke komputer melalui *port* USB ataupun port COM. Istilah *web* mengacu pada teknologi secara umum [6]. Oleh karena itu, kata *web* dapat diganti dengan kata lain sesuai pemandangan yang ditampilkan kamera. Misalnya, *StreetCam* yang

memperlihatkan pemandangan jalan. Ada *Metrocam* yang memperlihatkan pemandangan panorama kota dan pedesaan. *TraffiCam* yang digunakan untuk memantau jalan raya. Memantau cuaca dengan *WeatherCam*. Memantau keadaan gunung berapi dengan *VolcanoCam*.

2.10.1 Sejarah Webcam

Pada awalnya, bentuk *webcam* terbatas pada bentuk-bentuk standar yang hanya terdiri dari lensa dan papan sirkuit serta casing yang biasa. Namun seiring dengan perkembangan teknologi, bentuk *webcam* pun sudah makin bervariasi dengan fitur-fitur yang makin canggih [6]. Sebuah penemuan oleh Microsoft pada tahun 2004 menggambarkan kemajuan perkembangan teknologi *webcam*. i2i adalah sebuah sistem dua-kamera yang dengan sangat hati-hati mengikuti pergerakan individu. Kamera ini menggunakan perhitungan algoritma yang secara spesial dikembangkan untuk memfusikan apa yang setiap kamera lihat untuk membuat gambar '*cyclopean' stereo* yang akurat. Kamera ini juga dapat menampilkan *emoticon 3D* yang melayang. Sistem i2i dapat juga menghasilkan gambar background yang realistis sehingga pengguna dapat berpura-pura berada di tempat lain. Kemampuan sistem i2i ini, diantaranya yaitu kemampuan tracking (disebut *smart framing*) dan juga kemampuan *smart focusing*, dapat menambah pengalaman berkonferensi bagi pengguna.

Teknologi *webcam* pada awalnya mendapat dukungan komersial dari industri pornografi. Industri ini membutuhkan gambar-gambar '*live'* dan meminta pembuatan software yang mampu melakukannya tanpa *web browser* plugins. Hal ini melahirkan teknologi *live streaming webcam* yang masih tetap ada hingga sekarang. Sekarang ini web camera yang ada di pasaran pada umumnya terbagi ke dalam dua tipe: *webcam* permanen (*fixed*) dan *revolving webcam*. Pada *webcam* permanen terdapat pengapit untuk mengapit lensa standar di posisi yang diinginkan untuk menangkap gambar pengguna. Sedangkan pada *revolving webcam* terdapat landasan dan lensa standar dipasang di landasan tersebut sehingga dapat disesuaikan ke sudut pandang yang terbaik untuk menangkap gambar pengguna. Penggunaan *webcam* mencakup *video conferencing, internet dating, video*

messaging, home monitoring, images sharing, video interview, video phone-call, dan banyak hal lain. Kamera untuk *video conference* biasanya berbentuk kamera kecil yang terhubung langsung dengan komputer. Kamera analog juga kadang-kadang digunakan, kamera ini terhubung dengan *video capture card* dan tersambung dengan internet (baik langsung maupun tidak langsung). Saat ini kamera untuk *video conference* sudah makin maju, sudah ada *webcam* yang di dalamnya terdapat *microphone* maupun *noise cancellation* untuk memfokuskan audio ke *speaker* yang terletak di depan kamera sehingga *noise* yang ada tidak mengganggu jalannya konferensi.

2.11 Sistem Operasi Android

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan sistem operasi Android, diantaranya pengenalan sistem operasi Android, arsitektur Android, dan Android SDK (*Software Development Kit*).

2.11.1 Pengenalan Sistem Operasi Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi [11]. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka [11]. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada masa saat ini, sebagian besar vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis Android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi Android. Hal ini karena Android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun [11].

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi tablet PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan di atas adalah karena Android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, aplikasi dan tool pengembangan, *market* aplikasi Android, serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga Android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia [11].

Android telah melalui cukup banyak pembaruan sejak pertama rilis. Tabel 2.1 menunjukkan berbagai versi Android beserta dengan tanggal rilis dan *codename* (nama kode).

Tabel 2.1 Versi Android

Versi Android	Tanggal Rilis	Nama Kode
1.0	23 September 2008	-
1.1	9 February 2009	-
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 Oktober 2009	Éclair
2.2	20 Mei 2010	Froyo
2.3	6 Desember 2010	Gingerbread
3.0/3.1/3.2	22 Februari 2011	Honeycomb
4.0	19 Oktober 2011	Ice Cream Sandwich
4.1/4.2/4.3	9 Juli 2012	Jelly Bean
4.4	31 Oktober 2013	KitKat
5.0	3 November 2014	Lollipop

2.11.2 Arsitektur Android

Secara garis besar, arsitektur Android terdiri atas 5 bagian, diantaranya sebagai berikut [11]:

1. *Application* dan *Widgets*

   Merupakan *layer* (lapis), dimana pengguna berhubungan dengan aplikasi saja.

2. *Application Frameworks*

   Merupakan *Open Development Platform* yang ditawarkan Android untuk dapat dikembangkan guna membangun aplikasi. Pengembang memiliki akses penuh menuju API *Frameworks* seperti yang di lakukan oleh aplikasi kategori inti. Komponen-komponen yang termasuk di dalam *Applications Frameworks*

adalah *Views*, *Content Provider*, *Resource Manager*, *Notification Manager*, dan *Activity Manager*.

3. *Libraries*

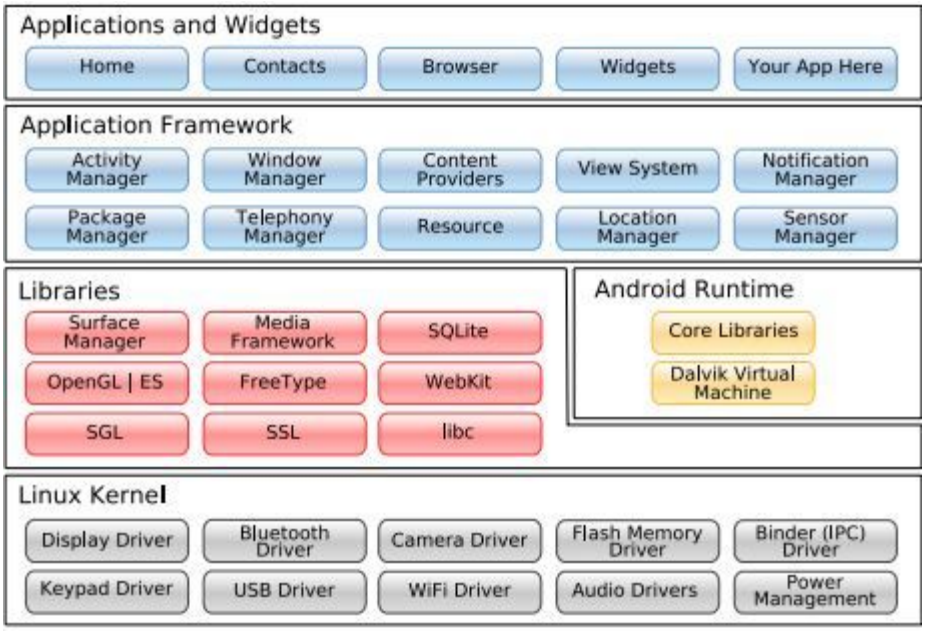
   Merupakan *layer*, dimana fitur-fitur Android berada.

4. *Android Run Time*

   Merupakan *layer* yang membuat aplikasi Android dapat dijalankan, dimana dalam prosesnya menggunakan implementasi Linux.

5. Linux Kernel

   Merupakan *layer* inti dari sistem operasi Android berada.



Gambar 2. 5 Arsitektur Android

2.11.3 Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middeware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform*

aplikasi netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan yang bukan merupakan aplikasi bawaan *handphone*/*smartphone* [11].

2.12 Object Oriented Analysis and Design

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan *user* [12]. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

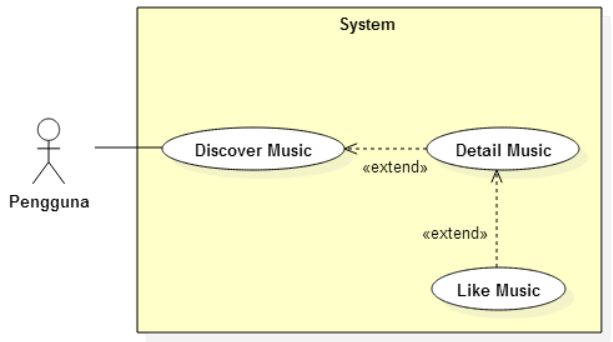
Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek [13]. Dalam membangun block UML ada 3 hal yang harus diperhatikan, yaitu *object* (memodelkan konsep), *relationship* (mengkoneksikan *object*), dan *diagram* (*grouping* yang saling mengkoneksikan antara *object* dan *relationship.* Diagram yang umum dipakai dalam analisis dan desain adalah:

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. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah use case dapat di- include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang serupa. Sebuah *use case* juga dapat meng-*extend usecase* lain dengan *behaviour*-nya sendiri. Sementara hubungan

generalisasi antar *use case* menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

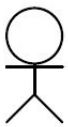
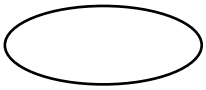
Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, *tester*, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case* [13]. Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar 2.6 Contoh *Use Case Diagram* dan simbol yang terdapat pada *use case diagram* dapat dilihat pada Gambar 2.6 Simbol *Use Case Diagram*.

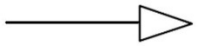


Gambar 2. 6 Contoh *Use Case Diagram*

Simbol yang terdapat pada *use case diagram* dapat dilihat pada **Tabel 2.1 Simbol *Usecase Diagram***.

Tabel 2.1 Simbol *Usecase Diagram*

Simbol	Nama Simbol	Keterangan
	Actor	Menggambarkan pihak yang berhubungan dengan sistem baik itu merupakan pengguna atau sistem lainnya yang berada dari sistem yang sedang dibahas.
	Use Case	Simbol yang menggambarkan suatu kegiatan (aktifitas) yang terjadi pada sistem
	Asosiasi	Menghubungkan keterkaitan antara aktor dengan use case

Simbol	Nama Simbol	Keterangan
	Include	Relasi use case dimana proses bersangkutan akan dilanjutkan ke proses yang dituju
	*Extend*	Relasi use case tambahan ke sebuah use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu
	Generalisasi	Sebuah relasi di mana fungsi yang satu adalah fungsi yang umum dari yang lainnya.

2. *Use Case Scenario*

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per *use case*nya. Berikut adalah dasar format penulisan *use case scenario* [13]. Dasar pembangunan *use case scenario* dapat dilihat pada **Tabel 2.2** Dasar Pembangunan *Use Case Scenario.*

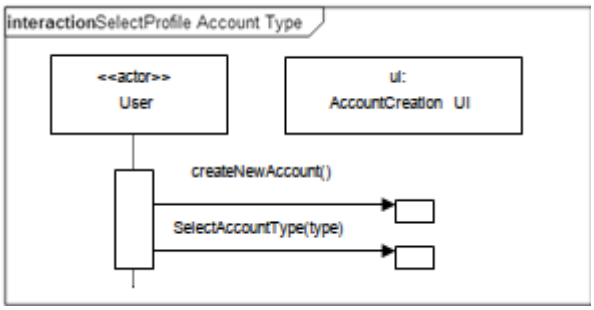
Tabel 2.2 Dasar Pembangunan *Use Case Scenario*

Use Case Name	Berisi nama dari *Use case* yang akan digunakan	
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari *Use case*	
Description	Menjelaskan gambaran dari *Use case*	
Related Use Case	Daftar *Use case* yang berhubungan dengan *Use case* tersebut	
Successful End Condition	Kondisi *Use case* jika berhasil	
Failed End Condition	Kondisi *Use case* jika gagal	
Actors	Daftar aktor yang dapat mengakses *Use case*	
Trigger	Aktifitas yang dilakukan untuk mengawali *Use case*	
Main Flow	Step	Action
	1	Deskripsi urutan aksi dari aktifitas *Use case*
	2	
	3	
Extension	Step	Branching Action
	2.1	Deskripsi urutan aksi lain selain urutan aksi utama
	2.2	

3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan *Output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *Output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message [13]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*. Comtoh dari sequence diagram dapat dilihat pada Gambar 2.7

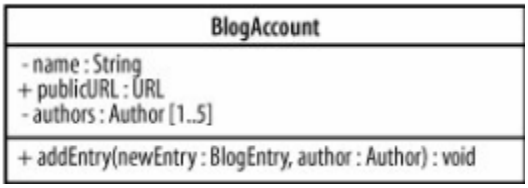


Gambar 2. 7 Contoh *Sequence Diagram*

4. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) satu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class*

diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.



Gambar 2. 8 Contoh *Class Diagram*

Class memiliki tiga area pokok:

a. Nama dan stereotype
b. Atribut
c. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Class memiliki tipe-tipe *relationship*, diantaranya :

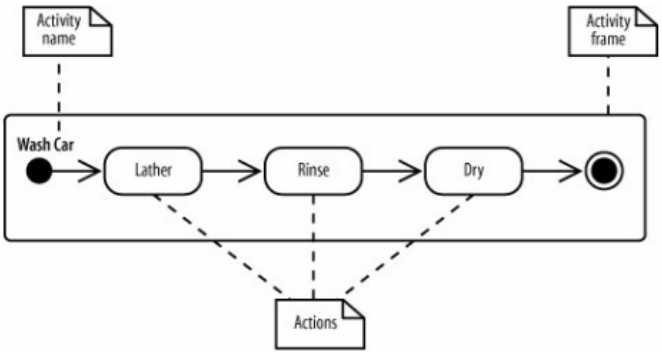
a. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
b. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.
c. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan

menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Komposisi, yaitu jenis relasi *class diagram* yang kuat dimana jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/*solid*.
e. Depedensi, salah satu jenis relasi *class diagram* yang lemah dimana objek dalam suatu class akan bekerja sangat singkat dengan objek yang ada pada *class* lain.

5. *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis *horizontal* atau *vertikal*. *Activity diagram* dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [13]. Contoh dari *activity diagram* dapat dilihat pada Gambar 2.9



Gambar 2. 9 Contoh *Activity Diagram*

2.13 *Web Services*

Perangkat peranti lunak integrasi aplikasi perusahaan sifatnya spesifik terhadap suatu produk, artinya perangkat ini hanya dapat bekerja dengan bagian-bagian peranti lunak dan sistem operasi tertentu. Contoh, suatu perangkat EAI untuk menghubungkan bagian tertentu dari peranti lunak pencatatan penjualan untuk aplikasi produksi, pengangkutan, dan penagihan, tidak dapat bekerja dengan peranti lunak pencatatan penjualan dari vendor lainnya. Program *middleware* yang dikembangkan oleh sistem BEA, sebuah integrator yang besar, tidak dapat berhubungan dengan aplikasi *middleware* vendor lainnya yang dibeli pada tahun-tahun sebelumnya tanpa pengeluaran yang besar untuk pemrograman dan perancangan. Layanan *web* (*web services*) mencoba memberikan alternatif standar untuk menghadapi masalah-masalah integrasi seperti itu dengan menciptakan sebuah lingkungan komunikasi yang netral [14].

Layanan *web* (*web services*) adalah serangkaian komponen peranti lunak yang bertukar informasi antara satu sama lain dengan bebas menggunakan komunikasi *web* dan bahasa yang standar. Layanan web dapat bertukar informasi antara dua sistem yang berbeda, terlepas dari sistem operasi atau bahasa pemrograman sistem tersebut. Layanan web dapat dipadukan untuk membangun sistem penghubung aplikasi berbasis *web* standar terbuka dari dua organisasi yang berbeda, dan juga dapat digunakan untuk membuat aplikasi yang menghubungkan sistem terpisah dalam satu perusahaan. Layanan *web* tidak terikat dengan sistem operasi atau bahasa pemrograman apa pun, dan aplikasi-aplikasi yang berbeda dapat digunakan

untuk berkomunikasi satu sama lain dengan cara standar tanpa penulisan kode khusus yang menghabiskan waktu [14].

2.14 Pemrograman *Web*

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan pemrograman web, diantaranya HTML, CSS, PHP dan JavaScript.

2.14.1 HTML

HTML adalah singkatan dari *Hyper Text Markup Language*. HTML digunakan untuk membuat halaman *web*. Sebuah *file* dokumen yang ditulis dalam format HTML akan dibaca dan diterjemahkan oleh *web browser* (misal Internet Explorer) untuk kemudian disajikan dalam bentuk *web* [15].

File-file yang ditulis dalam format HTML disimpan dengan ekstensi *.htm* atau *.html*. *File-file* tersebut dapat ditulis atau diketik menggunakan berbagai macam teks editor, misalkan Notepad, Wordpad, dan lain sebagainya [15].

HTML terdiri atas berbagai macam *tag* yang digunakan untuk menandai dan mengatur tampilan halaman *web* yang dibuat. Misalkan jika ingin membuat sebuah kalimat bercetak tebal di *web*, maka gunakan *tag* yang cocok untuk itu [15]

Tag biasanya dituliskan berpasangan di awal dan di akhir bagian *file* yang akan ditandai. Untuk *tag* yang diletakkan di awal bagian ditulis dengan tanda lebih kecil (<), nama *tag*, dan tanda lebih besar (>). Dan untuk *tag* yang diletakkan di akhir bagian ditulis dengan tanda lebih kecil (<), garis miring (/), nama *tag*, dan tanda lebih besar (>) [15].

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Gambar 2. 10 Contoh *Script* HTML

2.14.2 CSS

CSS (*Cascading Style Sheet*) adalah suatu teknologi yang digunakan untuk memperindah tampilan halaman *website* (situs). Singkatnya dengan menggunakan metode CSS ini dapat dengan mudah mengubah secara keseluruhan sekaligus memformat ulang situs. CSS dapat digunakan untuk menghindari pekerjaan berulang-ulang [15].

CSS mempunyai dua bagian utama yaitu *selector* dan deklarasi. Yang disebut *selector* biasanya elemen html yang ingin di ubah, sedangkan deklarasi biasanya terdiri dari *property* dan nilai, *property* sendiri adalah atribut *style* yang ingin di ubah, dan setiap *property* memiliki nilai. Setiap *syntax* css selalu diakhiri dengan titik koma (;), dan kelompok deklarasi dikelilingi oleh kurung kurawal ({}) [15].

Ada dua cara untuk membuat aturan *style*, yaitu dengan menuliskannya langsung dalam dokumen HTML atau dengan memisahkannya dalam sebuah *file* khusus yang berekstensi *.css*. Cara yang pertama dapat dilakukan dengan menambahkan *tag* <STYLE> diantara *tag* <HEAD> dan </HEAD>. Kemudian tuliskan aturan *style*-nya di dalam *tag* <STYLE> tersebut. Cara pertama sudah jarang dilakukan. Saat ini, para pembuat *web* biasanya menggunakan cara yang kedua, yaitu membuat aturan *style* dalam *file* terpisah [15].

```
body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}

p {
    font-family: "Times New Roman";
    font-size: 20px;
}
```

Gambar 2. 11 Contoh *Script* CSS

2.14.3 PHP

PHP adalah singkatan dari PHP: *Hypertext Preprocessor*. PHP merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting* [14]. PHP memungkinkan untuk membuat halaman *web* yang bersifat dinamis.

PHP dapat dijalankan pada berbagai macam sistem operasi, misalnya Windows, Linux, dan Mac OS. Selain Apache, PHP juga mendukung beberapa *web server* lain, misalnya Microsoft IIS, Caudium, PWS, dan lain-lain [14].

PHP dapat memanfaatkan *database* untuk menghasilkan halaman *web* yang dinamis [14]. Sistem manajemen *database* yang sering digunakan bersama PHP adalah MySQL. Namun, PHP juga mendukung sistem manajemen *database* Oracle, Microsoft Access, Interbase, dBase, PostgreSQL, dan lain-lain [14].

Hingga kini, PHP sudah berkembang hingga versi 5. PHP 5 mendukung penuh *Object-Oriented Programming* (OOP), integrasi XML, mendukung semua ekstensi terbaru MySQL, pengembangan *web services* dengan SOAP dan REST, serta ratusan peningkatan lainnya dibandingkan versi sebelumnya. PHP juga bersifat *open source* sehingga setiap orang dapat menggunakannya secara gratis [14].

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Gambar 2. 12 Contoh *Script* PHP

2.14.4 JavaScript

JacaScript merupakan bahasa *scripting* (bahasa pemrograman yang ringan) yang popular di Internet dan dapat bekerja di sebagian besar *browser* popular seperti Internet Explorer (IE), Mozilla FireFox, Netscape, Google Chrome, dan Opera. Kode JavaScript dapat disisipkan dalam halaman *web* menggunakan *tag* <SCRIPT> [15].

JavaScript adalah bahasa interpreter yang berarti skrip dieksekusi tanpa proses kompilasi [11]. JavaScript berisi baris kode yang dijalankan di komputer (*web browser*). JavaScript didesain untuk menambah interaktif suatu *web* [15]

Ada beberapa cara untuk menuliskan JavaScript, yaitu di bagian <HEAD>, <BODY> atau *external*. JavaScript dalam sebuah halaman *web* akan dieksekusi saat halaman di-*load* oleh *browser*. Namun, hal ini kadang tidak sesuai dengan keinginan karena suatu saat akan mengeksekusi skrip saat halaman di-*load*, tetapi kadang JavaScript dijalankan melalui *trigger* sebuah *event* [15]

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Gambar 2. 13 Contoh *Script* JavaScirpt

2.15 Pengujian *Black Box*

Pengujian *black box* fokus pada keperluan penelusuran kesalahan fungsional dari *software*. Ujicoba *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan *interface.*
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan performa.
5. Kesalahan inisialisasi dan terminasi.

Teknik pengujian *blackbox* terdiri dari 10 jenis diantaranya *Equivalence Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing.* Salah satunya yang akan dibahas adalah *Equivalence partitioning*.

Equivalence partioning merupakan metode ujicoba *black box* yang membagi domain *input* dari program menjadi beberapa kelas data dari kasus ujicoba yang dihasilkan. Kasus uji penanganan *single* yang ideal menemukan sejumlah kesalahan

(misalnya: kesalahan pemrosesan dari seluruh data karakter) yang merupakan syarat lain dari suatu kasus yang dieksekusi sebelum kesalahan umum diamati.

2.16 Pengujian Kuesioner

Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian ia diminta untuk menilai pernyataan-pernyataan tersebut.

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (STS), Tidak Setuju (TS), Ragu-Ragu (R), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel 2.7.

Tabel 2.7 Panduan Pemberian Skor

Jenis Pernyataan	Bobot pendapat				
	SS	S	R	TS	STS
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu, totalkan seluruh bobot jawaban tersebut kemudian bagi dengan total responden yang nantinya menjadi nilai rata-rata. Nilai rata-rata inilah yang diambil sebagai acuan sikap dimana jika nilai rata-rata kurang dari 3, maka dapat diartikan responden bersikap negatif dan jika nilai rata-rata lebih dari sama dengan 3, maka dapat diartikan responden bersikap positif terhadap tujuan yang ingin dicapai. Untuk lebih jelasnya dapat dilihat pada rumus dibawah ini.

$$\overline{X} = \frac{\sum Total}{n}$$

Dimana,

$\overline{X} \geq 3\ bersikap\ positif$

$\overline{X} < 3\ bersikap\ negatif$

Keterangan:

$\overline{X}$ = rata-rata

$\sum Total$ = jumlah seluruh nilai setwlah dikalikan dengan bobot

n = total responden