BAB II
TINJAUAN PUSTAKA

II.1 Profil Komunitas Indonesian Xperia Photography

IXP berawal dari sebuah grup Facebook yang dibuat pada 25 September 2012. Member yang bergabung pada beberapa bulan saat terbentuk hanya berkisar ratusan orang dan digunakan sebagai media *sharing* foto serta bertukar informasi. Sesuai namanya IXP atau singkatan dari Indonesia Xperia Photography menjadi wadah bagi pengguna *handphone* atau *smartphone* Xperia untuk menuangkan hobi mereka dalam dunia fotografi atau hanya sekedar berbagi bidikan terbaiknya.

Seiring berjalannya waktu, member yang bergabung dengan grup Facebook IXP semakin bertambah, sehingga pada penghujung tahun 2012 telah mencapai seribu lebih. Selama itu juga telah dibuat beberapa peraturan dan konsep dasar dalam IXP, diantaranya pembuatan tema khusus untuk satu bulan. Jadi dalam satu bulan member hanya boleh mengupload foto sesuai tema yang telah ditentukan. Tema yang diberikan cukup menarik diantaranya, *landscape*, *bokeh*, *macro*, dan lain sebagainya. Pada penghujung bulan akan dipilih foto terbaik yang mana dijadikan sebagai foto sampul pada grup facebook IXP. Kemudian pada beberapa bulan setelahnya dilakukan perubahan terhadap tema yang awalnya satu bulan menjadi satu minggu. Perubahan ini juga menandai kematangan peraturan IXP, sehingga member bisa lebih nyaman dan lebih mudah untuk membaca sebuah foto, diantaranya format (judul, tema, lokasi, *editing*, dan lain sebagainya). Dengan format yang tepat maka dapat dengan mudah informasi itu tersampaikan sesuai apa yang diharapkan, semisal cara pengambilan foto dan cara *editing* apabila diperlukan.

Pada pertengahan tahun jumlah member terus bertambah mencapai 3.000 lebih, sehingga perlu dilakukan penambahn admin yang awalnya dua orang menjadi empat orang. IXP juga telah membuat banyak perubahan dari sejak awal terbentuk, diantaranya pembuatan *Fanpage*, Instagram, Twitter, dan juga Blog yang saat ini diterbitkan. Juga tak lupa pembentukan dari sekedar grup Facebook menjadi

komunitas yang anggotanya tersebar dari sabang sampai merauke. *Gathering* pertama IXP diadakan pada 28 April 2013 yang dihadiri saat itu enam orang dari area Jawa Timur tepatnya di Surabaya dan sekitarnya. *Gathering* ini juga yang menjadi cikal bakal diadakannya *Gathering* selanjutnya, yang mana semakin hari semakin banyak yang ikut bergabung. Sejauh ini telah terbentuk regional yang telah matang dan diresmikan antara lain Regional Jatim, Regional Jakarta, Jogja, Jateng, Jabar, Sulawesi Selatan, Sumatra Selatan, Lampung. Dan member yang bergabung pun kini telah semakin bertambah yaitu sebanyak 23.758 orang.

Logo dari komunitas Indonesia Xperia Photography ini dapat dilihat pada Gambar II.1.



Gambar II.1 Logo Komunitas

II.1.1 Gambaran Umum Komunitas

Komunitas ini memiliki *rule* atau aturan yang harus diikuti agar komunitas ini dapat berjalan dengan teratur dan juga dapat lebih mudah untuk dikelola. Segala sesuatu yang melanggar aturan tersebut akan dikenai sanksi dari admin komunitas.

Dalam komunitas IXP ini para anggotanya dapat berbagi ilmu seputar fotografi dan juga dalam komunitas ini diadakan *event* dengan tema tertentu yang diberikan oleh admin grup setiap minggunya dan para anggota komunitas dapat mengikuti *event* tersebut dengan mem-*posting* hasil foto terbaik mereka, dan anggota lain dapat melakukan proses voting untuk nantinya ditentukan pemenang dari *event* yang sedang berjalan tersebut yang mana akan diumumkan pada akhir periode *event*, hal ini tentu akan memberikan kebanggan dan kesenangan tersendiri

bagi pemenang *event* yang mana akan memotivasi mereka untuk terus meningkatkan kreativitas mereka dalam mengambil foto.

Gambaran umum aktivitas *posting* foto dari komunitas Indonesia Xperia Photography dapat dilihat pada Gambar II.2.



Gambar II.2 Gambaran Umum Aktivitas *Posting* Foto Dari Komunitas IXP

II.1.2 Visi dan Misi Komunitas

Visi dan misi dari komunitas Indonesia Xperia Photography adalah:

1. Visi

Visi dari komunitas IXP ini adalah untuk menjadi wadah bagi pencinta fotografi ponsel Sony di Indonesia.

2. Misi

Misi dari komunitas IXP ini adalah:

a. Membuktikan bahwa dengan menggunakan kamera dari ponsel dapat menghasilkan foto yang dapat menandingi foto yang dihasilkan oleh kamera profesional.
b. Bersama-sama berbagi pengetahuan dan ilmu mengenai seputar fotografi.

II.1.3 Struktur Organisasi Komunitas

Struktur organisasi komunitas IXP ini memiliki struktur yang sejajar yang terdiri atas masing-masing ketua regional yang saling berkolaborasi. Dan setiap ketua regional tugasnya pun serupa dengan ketua regional lain. Para ketua regional inilah yang akan bertindak sebagai administrator dari *web*site IXP.

Berikut ini adalah daftar ketua dari masing masing regional yang ada saat ini:

Tabel II.1 Ketua Regional Komunitas IXP

No	Regional	Nama Ketua Regional
1	Jogja – Jateng	Fadholi Aprilian
2	Jatim	M. Mushonnif
3	Jabar	Leny
4	Jakarta	Dafit Ariyanto
5	Sulawesi Selatan	Ochie
6	Sumatra Selatan	M Roppiansyah
7	Lampung	Lukman Hakim

II.1.4 Deskripsi Kerja Jabatan

Deskripsi kerja dari jabatan yang akan ikut serta dalam penggunaan perangkat lunak ini adalah Ketua Regional dimana deskripsi pekerjaan yang dilakukan adalah:

1. Mengelola *web*site IXP.
2. Mengelola *posting* foto.
3. Mengelola *event* mingguan dan *Gathering event*.
4. Mengelola data anggota dan juga data admin *web*site IXP.

II.2 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan perangkat lunak bagi komunitas IXP ini. Beberapa teori terkait dengan pembangunan perangkat lunak ini adalah konsep fotografi, citra, pengolahan citra, operasi pengolahan citra, pengolahan citra yang akan digunakan, konvolusi, histogram, teknik *watermarking*, android, android GPUImage *library*, *structure analysis and design, object oriented analysis and design, web service*, JSON, pengujian *black box*, dan pengujian kuesioner.

II.2.1 Fotografi

Fotografi berasal dari bahasa Yunani, *Photos* dan *Graphos* berarti menggambar atau melukis. Jadi, arti fotografi sesungguhnya adalah menggambar/melukis dengan cahaya. Dari hakekat fotografi sesungguhnya kamera adalah sebuah alat yang berfungsi merekam cahaya kedalam film pada kamera analog, atau merekam cahaya ke dalam kartu memori pada kamera digital.

Dalam fotografi canggihnya peralatan yang digunakan untuk mengambil gambar tidak menjadi jaminan untuk mendapatkan foto yang baik, seorang fotografer juga harus memiliki keahlian dan pengalaman di bidang fotografi. Foto tidak sekedar kertas yang bergambar. Foto dapat memberikan ungkapan, cerita atau perasaan kita terhadap orang lain. Dengan foto kita dapat menceritakan suatu kejadian yang telah berlangsung dan kita alami. Dengan foto, kita juga dapat memberikan suatu informasi kepada orang lain, seperti keindahan alam, kejadian peristiwa, produk dan lain sebagainya [6].

II.2.2 Citra

Secara harfiah, citra (*image)* adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam [7].

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat [8]:

1. Optik berupa foto.
2. Analog berupa sinyal *video* seperti gambar pada monitor televisi.

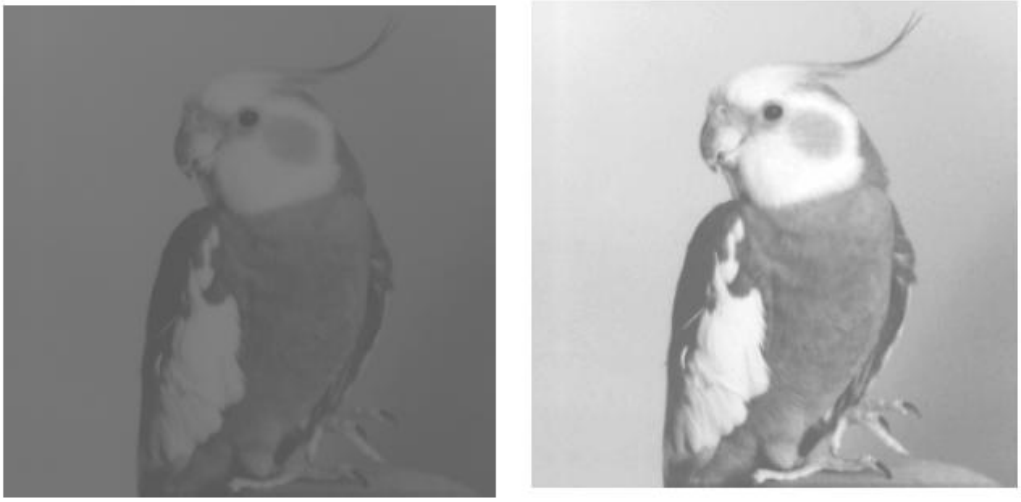
II.2.2.1 Pengolahan Citra

Meskipun sebuah citra kaya informasi, namun seringkali citra yang kita miliki mengalami penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang.

Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra (*image processing*).

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Sebagai contoh, citra burung nuri pada Gambar II.3 (a) tampak agak gelap, lalu dengan operasi pengolahan citra kontrasnya diperbaiki sehingga menjadi lebih terang dan tajam (b) [7]. Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila [9]:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra,
2. elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur,
3. sebagian citra perlu digabung dengan bagian citra yang lain.



(a) Sebelum (b) Sesudah

Gambar II.3 Contoh Hasil Perbaikan Citra

II.2.2.2 Operasi Pengolahan Citra

Operasi-operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut:

1. Perbaikan kualitas citra (*image enhancement*).

Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan.

Contoh-contoh operasi perbaikan citra:

a. Perbaikan kontras gelap/terang
b. Pencahayaan (*Brightness*)
c. Perbaikan tepian objek (*edge enhancement*)
d. Penajaman (*sharpening*)
e. Pemberian warna semu (*pseudocoloring*)
f. Penapisan derau (*noise filtering*)

2. Pemugaran citra (*image restoration*)

Operasi ini bertujuan menghilangkan/meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui.

Contoh-contoh operasi pemugaran citra:

a. Penghilangan kesamaran (*deblurring*).
b. Penghilangan derau (*noise*)

Kekaburan gambar mungkin disebabkan pengaturan fokus lensa yang tidak tepat atau kamera bergoyang pada pengambilan gambar. Melalui operasi *deblurring*, kualitas citra dapat diperbaiki sehingga tampak lebih baik.

3. Pemampatan citra (*image compression*)

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus. Contoh metode pemampatan citra adalah metode *JPEG*.

4. Segmentasi citra (*image segmentation*).

Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Pengorakan citra (*image analysis*)

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri

tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.

Contoh-contoh operasi pengorakan citra:

a. Pendeteksian tepi objek (*edge detection*)
b. Representasi daerah (*region*)
c. Ekstraksi batas (*boundary*)

6. Rekonstruksi citra (*image reconstruction*)

Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh [7].

II.2.2.3 Operasi Pengolahan Citra yang Akan Digunakan

Berikut ini adalah beberapa teknik *Image Enchanchement* yang akan di gunakan [7]:

1. *Brightness*

Untuk membuat citra lebih terang atau lebih gelap, kita melakukan pengubahan kecerahan gambar. Kecerahan/kecemerlangan gambar dapat diperbaiki dengan menambahkan (atau mengurangkan) sebuah konstanta kepada (atau dari) setiap *pixel* di dalam citra. Akibat dari operasi ini, histogram citra mengalami pergeseran.

Secara matematis operasi ini ditulis sebagai:

$$f(x, y)' = f(x, y) + b \qquad ..(II.3)$$

Jika *b* positif, kecerahan gambar bertambah, sebaliknya jika *b* negatif kecerahan gambar berkurang. Algoritma pengubahan kecerahan gambar ditunjukkan pada Algoritma 7.1. Citra masukan mempunyai 256 derajat keabuan yang nilai-nilainya dari 0 sampai 255. Intensitas *pixel* disimpan di dalam *Image* [0..N-1,0..M-1], sedangkan hasil pengubahan tetap disimpan di dalam citra *Image*. Nilai pixel hasil pengubahan mungkin ≤ derajat keabuan minimum (0) atau ≥

derajat keabuan maksimum (255). Karena itu, pixel tersebut perlu dilakukan *clipping* ke nilai keabuan minimum atau ke nilai keabuan maksimum.

2. *Contrast*

Kontras menyatakan sebaran terang (*lightness*) dan gelap (*darkness*) di dalam sebuah gambar. Citra dapat dikelompokkan ke dalam tiga kategori kontras yaitu citra kontras-rendah (*low contrast*), citra kontras-bagus (*good contrast* atau *normal contrast*), dan citra kontras-tinggi (*high contrast*). Ketiga kategori ini umumnya dibedakan secara intuitif.

Citra dengan kontras-rendah dapat diperbaiki kualitasnya Teknik yang lazim dipakai adalah perataan histogram (*histogram equalization*). Tujuan dari perataan histogram adalah untuk memperoleh penyebaran histogram yang merata, sedemikian sehingga setiap derajat keabuan memiliki jumlah *pixel* yang relatif sama.

Karena histogram menyatakan peluang *pixel* dengan derajat keabuan tertentu, maka rumus menghitung histogram ditulis kembali sebagai fungsi peluang:

$$p_r(r_k) = \frac{n_k}{n} \qquad ..(II.4)$$

Yang dalam hal ini,

$$n_k = \frac{k}{L-1}\ , 0\ \leq k\ \geq L-1 \qquad ..(II.5)$$

Yang dimaksud dengan perataan histogram adalah mengubah derajat keabuan suatu pixel (r) dengan derajat keabuan yang baru (s) dengan suatu fungsi transformasi *T*, yang dalam hal ini s = *T(r).*

Untuk mencari nilai derajat keabuan baru dapat menggunakan persamaan berikut:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} \sum_{j=0}^{k} P_r(r_j) \qquad ..(II.6)$$

Setelah didapat S_k kelompokan ke nilai r terdekat untuk memperoleh, hitung kembali nilai n_k yang baru dan hitunglah masing-masing $P_s(s_k)$.

3. *Sharpening*

Operasi penajaman citra bertujuan memperjelas tepi pada objek di dalam citra. Penajaman citra merupakan kebalikan dari operasi pelembutan citra karena operasi ini menghilangkan bagian citra yang lembut. Operasi penajaman dilakukan dengan melewatkan citra pada penapis lolos-tinggi (*high-pass filter*).

Penapis lolos-tinggi akan meloloskan (atau memperkuat) komponen yang berfrekuensi tinggi (misalnya tepi atau pinggiran objek) dan akan menurunkan komponen berfrekuensi rendah. Akibatnya, pinggiran objek telihat lebih tajam dibandingkan sekitarnya.

Aturan penapis lolos-tinggi:

a. koefisien penapis boleh positif, negatif, atau nol
b. jumlah semua koefisien adalah 0 atau 1

Jika jumlah koefisien = 0, maka komponen berfrekuensi rendah akan turun nilainya, sedangkan jika jumlah koefisien sama dengan 1, maka komponen berfrekuensi rendah akan tetap sama dengan nilai semula.

Contoh-contoh penapis lolos-tinggi:

$$\text{(i)}\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}\ \Sigma = 0 \qquad \text{(ii)}\begin{bmatrix}-1 & -1 & -1\\ -1 & 9 & -1\\ -1 & -1 & -1\end{bmatrix}\ \Sigma = 1 \qquad \text{(iii)}\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}\ \Sigma = 1$$

$$\text{(iv)}\begin{bmatrix}1 & -2 & 1\\ -2 & 5 & -2\\ 1 & -2 & 1\end{bmatrix}\ \Sigma = 1 \qquad \text{(v)}\begin{bmatrix}1 & -2 & 1\\ -2 & 4 & -2\\ 1 & -2 & 1\end{bmatrix}\ \Sigma = 0 \qquad \text{(vi)}\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}\ \Sigma = 0$$

Gambar II.4 Contoh Kernel Penapis Lolos-Tinggi [7]

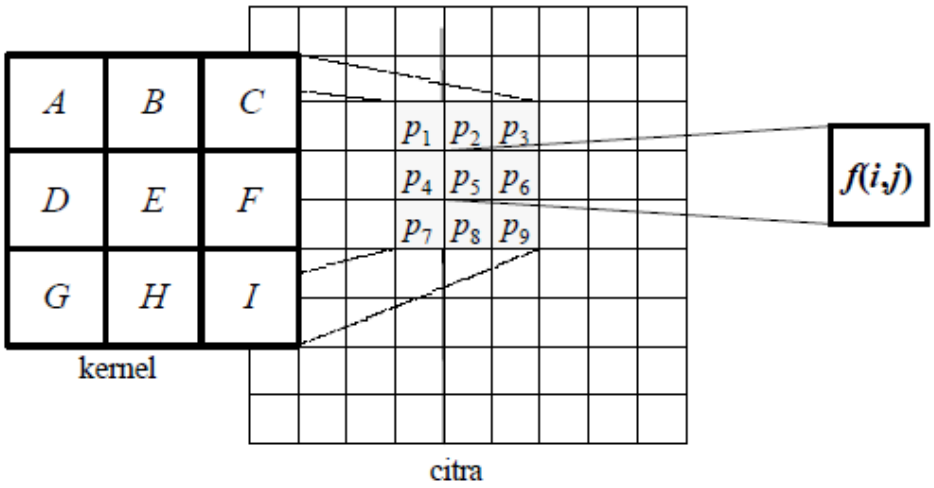
II.2.2.4 Konvolusi

Konvolusi adalah sebuah operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah *mask* atau *kernel*. Untuk fungsi dengan dua peubah (fungsi dua dimensi atau dwimatra), operasi konvolusi didefinisikan sebagai berikut:

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b)g(x-a, y-b) \quad ..(II.1)$$

Fungsi penapis *g(x,y)* disebut juga *convolution filter*, *convolution mask*, *convolution kernel*, atau *template*. Dalam ranah diskrit *kernel* konvolusi dinyatakan dalam bentuk matriks (umumnya 3 ´ 3, namun ada juga yang berukuran 2 ´ 2 atau 2 ´ 1 atau 1 ´ 2). Ukuran matriks ini biasanya lebih kecil dari ukuran citra. Setiap elemen matriks disebut koefisien konvolusi.

Ilustrasi konvolusi ditunjukkan pada Gambar II.5 [7]:



$$f(i,j) = Ap1 + Bp2 + Cp3 + Dp4 + Ep5 + Fp6 + Gp7 + Hp8 + Ip9$$

Gambar II.5 Ilustrasi Konvolusi

II.2.2.5 Histogram

Informasi penting mengenai isi citra digital dapat diketahui dengan membuat histogram citra. Histogram citra adalah grafik yang menggambarkan penyebaran nilai-nilai intensitas *pixel* dari suatu citra atau bagian tertentu di dalam citra. Dari sebuah histogram dapat diketahui frekuensi kemunculan nisbi (*relative*) dari intensitas pada citra tersebut. Histogram juga dapat menunjukkan banyak hal tentang kecerahan (*brightness*) dan kontras (*contrast*) dari sebuah gambar. Karena

itu, histogram adalah alat bantu yang berharga dalam pekerjaan pengolahan citra baik secara kualitatif maupun kuantitatif [7].

Secara matematis histogram citra dihitung dengan rumus:

$$h_i = \frac{n_i}{n}, i = 0, 1, \ldots, L-1 \quad \text{..(II.2)}$$

yang dalam hal ini,

ni = jumlah pixel yang memiliki derajat keabuan *i*

n = jumlah seluruh pixel di dalam citra

II.2.2.6 Teknik *Watermarking*

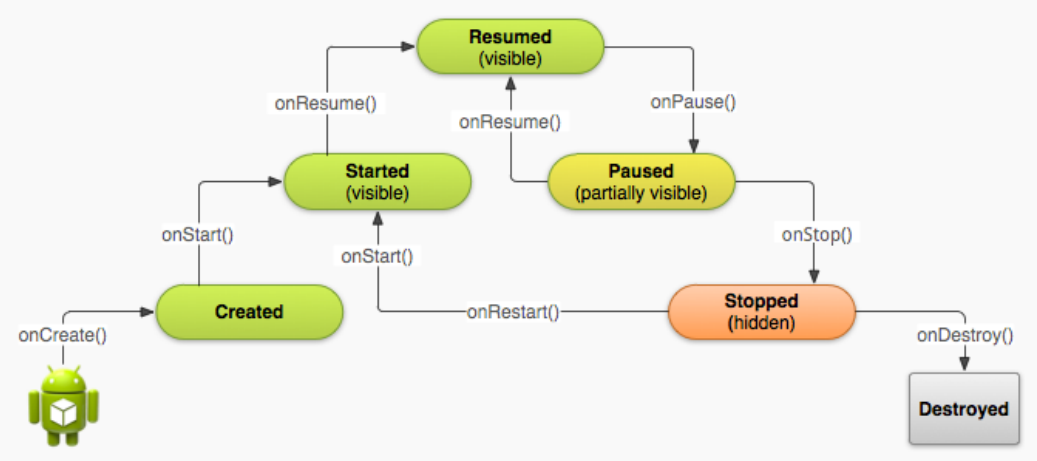
Pada dasarnya, teknik *watermark*ing adalah proses menambahkan kode identifikasi secara permanen ke dalam data digital. Kode identifikasi tersebut dapat berupa teks, gambar, suara, atau *video*. Selain tidak merusak data digital produk yang akan dilindungi, kode yang disisipkan seharusnya memiliki ketahanan (*robustness*) dari berbagai pemrosesan lanjutan seperti pengubahan, transformasi geometri, kompresi, enkripsi, dan sebagainya. Sifat *robustness* berarti data *watermark* tidak terhapus akibat pemrosesan lanjutan tersebut [7]. Namun pada penelitian ini teknik pemasangan *watermark* nya berbeda, tanpa memasukan suatu kode kedalam gambar, hanya menggabungkan gambar utama dengan gambar *watermark* sesuai dengan kordinat *x* dan *y* yang ditentukan.

II.2.3 Android

Android adalah sistem operasi *smartphone* layar sentuh seperti iOS iPhone dan OS BlackBerry yang dalam pengembangannya dipimpin oleh google. Sistem operasi ini bersifat *Open Source* dan dikembangkan berdasarkan kernel Linux yang disematkan pada gadget, baik itu *handphone* atau tablet [10]. Android memiliki OS yang sangat baik, cepat dan kuat serta memiliki antarmuka pengguna intuitif yang dikemas dengan pilihan dan fleksibilitas. Sedangkan android *SDK* (*Software Development* Kit) menyediakan *tools* dan *API* yang diperlukan untuk

mengembangkan aplikasi pada platform android dengan menggunakan bahasa pemrograman Java.

Di dalam android terdapat *activity* dimana komponen ini memberi interaksi antara user dan aplikasi yang dibangun melalui user *interface*. *Activity* ini memiliki Siklus hidup yang dinamakan *Android Life Cycle*. Flowchart siklus hidup tersebut dapat dilihat pada Gambar II.6.



Gambar II.6 *Android Life Cycle*

Gambar II.6 *Android Life Cycle* menjelaskan mengenai sebuah ilustrasi sederhana dari siklus hidup *activity* yang dinyatakan sebagai langkah piramida. Hal ini menunjukkan bagaimana untuk setiap *callback* yang digunakan untuk mengambil aktivitas langkah menuju *state* atas kemudian dilanjutkan ada metode *callback* yang mengambil langkah menurun. Kegiatan ini juga dapat kembali ke keadaan dari *statepause* dan *stop*. Siklus hidup *activity* ini menjadi sangat penting ketika kita hendak membuat aplikasi berbasis android. Gunanya adalah agar aplikasi yang dibangun berjalan dengan baik. Berikut penjelasan pada masing-masing *state*:

1. onCreate(), Method ini dipanggil ketika *activity* pertama kali dibuat.
2. onStart(), Method ini dipanggil ketika sebuah *activity* tampil ke pengguna
3. onResume(), Method ini dipanggil ketika *activity* yang berjalan pada saat itu dihentikan sementara (*paused*) dan *activity* sebelumnya dijalankan kembali(*resumed*). (Hasil dari method OnRestart())

4. onPause(), Method ini di panggil ketika *activity* di hentikan sementara (*pause*) dan berikutnya ketika dijalankan kembali akan berada dalam posisi *resume* dan memanggil method OnResume()
5. onStop(), Method ini dipanggil ketika *activity* tidak lagi tampak kepada pengguna
6. onDestroy(), Method ini dipanggil sebelum *activity* dihancurkan (*destroy*) oleh sistem (baik secara manual maupun untuk kepentingan pelonggaran memori).

Banyak sekali fitur yang dapat dimanfaatkan pada platform android ini. Diantaranya adalah *Location Based Services* (*LBS*) dan *Google Cloud Messaging* (GCM). *Location Based Services* (*LBS*) dapat dimanfaat untuk layanan yang menggunakan informasi geografis dalam memberikan informasi lokasi kepada pengguna, sebagai petunjuk posisi atau lokasi piranti *mobile* pengguna berada, dan menemukan rute jalan sesuai permintaan. *LBS* dapat digambarkan sebagai satu layanan yang berada pada pertemuan tiga teknologi yaitu: *Geographic Information System* (*GIS*), *Internet Service*, dan *Mobile Devices* [11]. *Google Cloud Messaging* atau disingkat *GCM* merupakan suatu layanan yang disediakan oleh google, dengan layanan ini kita dapat mengirimkan data dari *server* ke user yang pada aplikasinya menggunakan layanan ini. *Google Cloud Messaging* dapat digunakan *developer*, khususnya *developer* android untuk mengirimkan *push notificationication.*

II.2.4 Android GPUImage *Library*

Library ini adalah *library* yang di buat oleh Cyber Agent pada tahun 2012 yang dapat di unduh pada situs https://github.com/CyberAgent/android-gpuimage dimana *library* ini menyediakan berbagai macam fungsi dalam memperbaiki kualitas citra. Dalam penelitian ini yang akan digunakan diantaranya:

1. Pengaturan *contras*.
2. Pengaturan *Brightness*.
3. *Sharpening*.

Berikut ini adalah Cara pemanggilan fungsi dari *library* GPUImage ini:

```
    mGPUImage = new GPUImage(this);
    mGPUImage.setGLSurfaceView((GLSurfaceView)
findViewById(R.id.surfaceView));
    mGPUImage.setImage(imageUri); // this loads image on the
current thread, should be run in a thread
    mGPUImage.setFilter(new GPUImageSepiaFilter());
```

II.2.5 *Structure Analysis and Design*

Struktur analisis dan desain adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem kearah fungsional. Pendekatan ini memecahkan masalah-masalah dalam aktifitas bisnis menjadi bagian-bagian kecil yang dapat disatukan kembali menjadi satu kesatuan yang utuh untuk memecahkan masalah [12].

Tools yang dapat digunakan pada pendekatan analisis pengembangan sistem secara terstruktur adalah:

1. Diagram Konteks adalah diagram yang menggambarkan sebuah sistem secara menyeluruh yang selanjutnya akan lebih didetailkan oleh DFD. Konteks diagram juga biasa disebut sebagai DFD *level* 0.
2. *Data Flow Diagram* atau biasa di singkat DFD merupakan serangkaian diagram yang menggambarkan kegiatan-kegiatan yang ada dalam satu sistem. Teknik pembangunan DFD dimulai dengan menggambarkan sistem secara global dan dilanjutkan dengan analisis masing-masing bagian. Pada awalnya digambarkan konteks diagram yang menggambarkan sebuah sistem secara menyeluruh yang akan diinvestigasi. Konteks diagram tersebut dapat dikatakan sebagai DFD *level* 0. Analisis sistem yang lebih detail selanjutnya dapat dilakukan dengan menggambarkan DFD *level* 1,2 dan seterusnya.
3. Spesifikasi Proses merupakan tabel yang berisi keterangan deskripsi dari semua proses yang terdapat di DFD. Logika proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau *pseudo code* (tidak boleh campuran.
4. Kamus Data (*Data Dictionary*) merupakan fakta tentang data dan kebutuhan-kebutuhan informasi dari sistem informasi. Dengan menggunakan *data dictionary*, analis sistem dapat mendefinisikan data yang mengalir dalam sistem dengan lengkap [12].

II.2.6 *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 [13]. *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 [14]. 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* [15].

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* [15]. Dasar pembangunan *use case scenario* dapat dilihat pada Tabel II.2.

Tabel II.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 [14]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*.

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. *Class* memiliki tiga area pokok:

1. Nama dan stereotype
2. Atribut
3. 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 :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.
3. 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.
4. 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.
5. 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

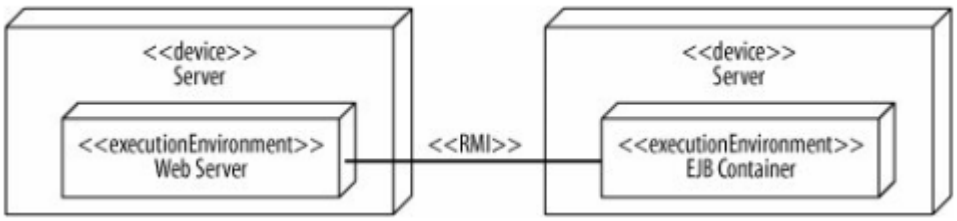
5. *Activity Diagram*

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 [14].

6. *Depeloyment Diagram*

Deployment diagram merupakan diagram yang menunjukkan susunan fisik suatu sistem yang dibangun. *Deployment diagram* menunjukkan bagian perangkat lunak mana yang berjalan pada suatu perangkat keras tertentu yang digunakan pada sistem. Hal yang penting dalam *deployment diagram* adalah pusat – pusat yang dihubungkan oleh suatu jalur komunikasi dimana pusat tersebut merupakan sebuah titik yang mengumpulkan beberapa perangkat lunak. Berikut contoh bentuk *deployment diagram*:



Gambar II.7 Contoh *Deployment Diagram*

II.2.7 *Web Service*

Web Service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya yang terletak di satu *server* yang terhubung ke internet [16]. Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah

organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain XML, terdapat pula format pertukaran data lain yaitu JSON.

II.2.8 JSON (*Java Sript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer [17]. JSON terbuat dari dua struktur yaitu:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.
2. Daftar nilai terurutkan (*list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*Sequence*).

JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan[(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

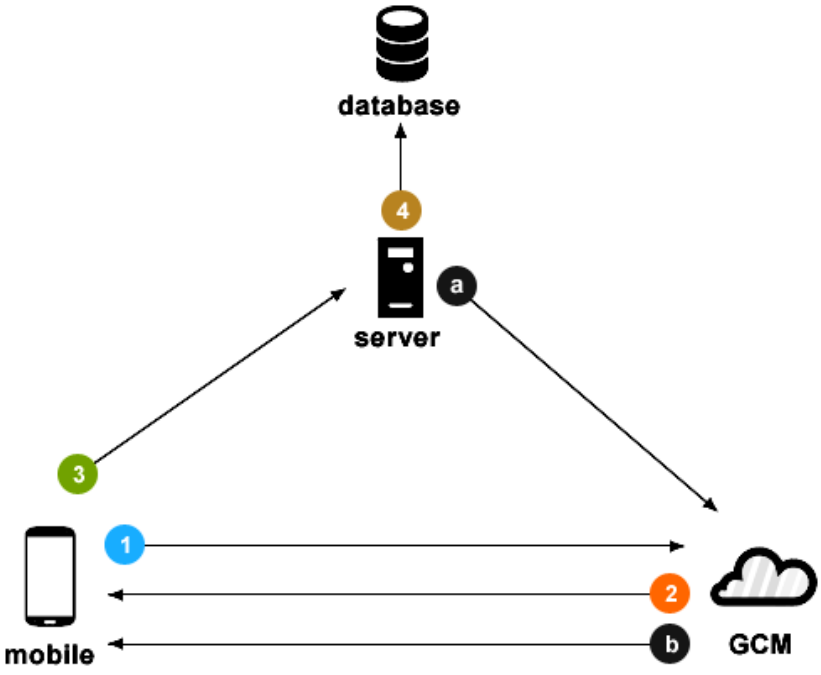
4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka, Format oktal dan heksadesimal tidak digunakan.

II.2.9 *Google Cloud Messaging*

Google Cloud Messaging adalah teknologi untuk membantu pengembang mengirim data dari *server* menuju *device* android secara langsung melalui *server* google, pemanfaatan teknologi ini biasanya digunakan untuk *push messaging.* Pengiriman pesan menggunakan teknologi ini mengharuskan adanya API key, key ini terkait langsung dengan akun google yang digunakan untuk mempublikasikan aplikasi melalui google play [18].



Gambar II.8 GCM Model [19]

Berdasarkan ilustrasi dari Gambar II.8 terdapat dua *server* yang terlibat dalam proses *push messaging* , pada setiap *server* yang terlibat terdapat proses pengiriman data yang berbeda-beda diantaranya [18]:

1. *Device* android mengirimkan sender id dan application id menuju GCM *Server* untuk *registrasi.*
2. Setelah berhasil registrasi, GCM *server* akan mengirimkan *registration* id menuju *device* android.
3. Setelah *device* android mendapatkan *registration* id, *device* tersebut akan mengirimkan *registration* id menuju *server* aplikasi.
4. *Server* aplikasi akan menyimpan registrasi id yang dikirimkan untuk digunakan sebagai id saat menggunakan *push messaging*.

II.2.10 Pengujian *Black Box*

Metode 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 black box 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 digunakan adalah *Equivalence partitioning*.

Equivalence partioning merupakan metode ujicoba *blackbox* 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.

II.2.11 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 (ST), Tidak Setuju (T), Tidak Memutuskan (N), 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 II.4.

Tabel II.3 Panduan pemberian skor

Bobot pendapat				
SS	S	N	T	TS
5	4	3	2	1

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu total kan 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 skala netral, maka dapat diartikan responden bersikap negatif, jika nilai rata-rata lebih dari skala netral, maka dapat diartikan responden bersikap positif, dan jika nilai rata-rata sama dengan skala netral maka responden bersikap netral terhadap tujuan yang ingin peneliti capai [20]. Untuk lebih jelasnya dapat dilihat pada rumus seperti berikut.

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

Dimana,

$\overline{X} > 3$ bersikap positif

$\overline{X} < 3$ bersikap negatif

$\overline{X} = 3$ bersikap netral

Keterangan :

$\overline{X}$ = rata-rata

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

n = total responden