BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis

Analisis bertujuan untuk mengidentifikasi permasalahan-permasalahan yang terdapat pada sistem serta menentukan kebutuhan-kebutuhan dari sistem yang dibangun. Analisis tersebut meliputi analisis masalah, analisis kebutuhan data, analisis kebutuhan fungsional, dan analisis sistem.

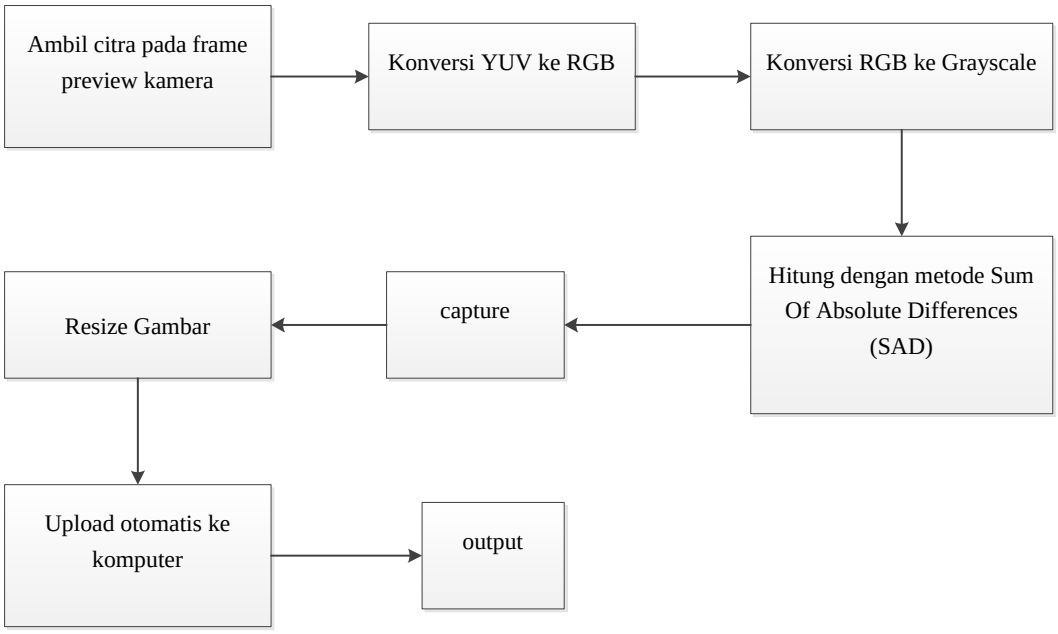
3.1.1 Analisis Masalah

Pentingnya keamanan terhadap tindak pencurian barang berharga maka diperlukannya suatu aplikasi yang dapat digunakan untuk memantau keadaaan suatu ruangan menggunakan perangkat yang umum dimiliki masyarakat yaitu *smartphone*. Banyak aplikasi pengawasan atau monitoring yang memonitor situasi memiliki sistem kerja merekam situasi terus menerus tanpa henti, walaupun ruangan yang dimonitor tidak terdapat aktifitas. Sehingga akan lebih efektif ketika aplikasi dapat mendeteksi sebuah pergerakan pada *preview* kamera di android secara *real-time*, aplikasi dapat meng-*capture* pergerakan tersebut dan aplikasi dapat meng-upload gambar hasil *capture* secara otomatis ke komputer *server*.

Metode yang digunakan untuk mendeteksi gerak adalah *Sum Of Absolute Difference* (SAD). Diawali dengan mengambil citra pada frame *preview* kamera, karena citra yang didapat adalah format YUV maka ubah kedalam format RGB lalu mengubahnya menjadi *grayscale*. Setelah itu ambil nilai *pixel* pada frame kamera, bandingkan antara frame 1 dan selanjutnya, diambil selisihnya dan di *absolute*-kan. Agar kamera tidak meng-*capture* sembarang pergerakan maka dilakukan proses *thresholding* dimana jika nilai pergerakan tersebut kurang dari nilai threshold maka aplikasi tidak akan meng-*capture* pergerakan tersebut dan aplikasi kembali memantau ruangan. Dan jika pergerakan itu dianggap sebuah pergerakan maka aplikasi akan meng-upload secara otomatis ke komputer *server* dengan menggunakan jaringan *wireless fidelity (wi-fi)*

3.1.2 Analisis Sistem yang akan dibangun

Sistem yang akan dibangun adalah sistem aplikasi pengawasan atau monitoring sebuah ruangan. Aplikasi ini dapat mengenali sebuah pergerakan dan akan meng-*capture* secara otomatis ketika sebuah pergerakan terdeteksi. adapun gambaran umum tentang sistem yang akan dilihat pada Gambar 3.1.



Gambar 3.1 Analisis Sistem yang akan dibangun

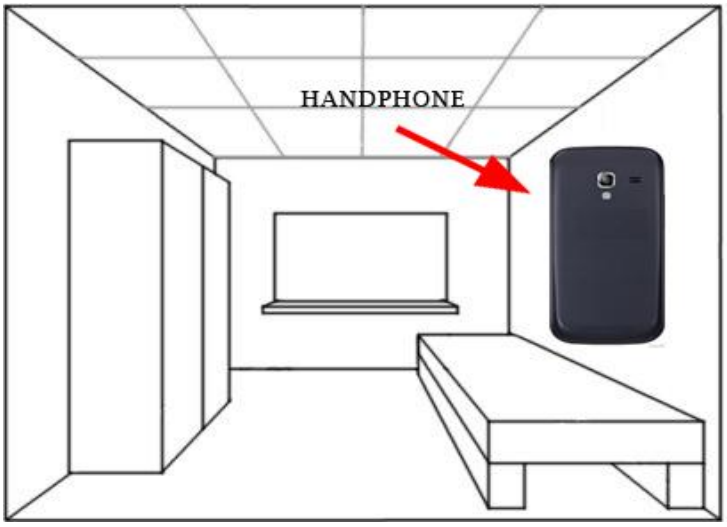
Berikut penjelasan alur proses sistem yang akan dibangun :

1. Kamera akan mengambil citra pada frame – frame preview kamera dan citra yang didapat adalah format YUV.
2. Format citra YUV dikonversikan ke jenis citra warna yaitu format RGB.
3. Suatu citra warna di ubah menjadi citra *grayscale* untuk memperoleh informasi intensitas dari gambar, sehingga dapat di sortir mulai dari hitam untuk intensitas yang paling lemah samapai dengan putih yang intensitas yang paling kuat. Dan mendapatkan nilai *grayscale* nya
4. *Sum of absolute differences* (SAD) pada tahap ini sistem mendeteksi sebuah pergerakan jika ada perubahan antar *pixel*. Dengan cara membandingkan frame-frame yang kamera tangkap.

5. *Capture* Pada tahap ini sistem akan meng-*capture* secara otomatis ketika ada sebuah pergerakan terdeteksi.
6. *Resize Gambar* pada tahap ini resolusi gambar akan dirubah ukurannya untuk mempercepat proses upload.
7. *Upload* pada tahap ini sistem akan meng-upload secara otomatis menggunakan jaringan *wireless Fidelity (Wi-Fi)* ketika pergerakan tersebut telah di *capture* oleh aplikasi.
8. *Output* pada tahap ini gambar hasil *capture* telah berada di folder komputer.

3.1.2.1 Analisis *Preview* Kamera

Pada tahap ini merupakan proses yang penting, karena melalui proses inilah data input akan didapatkan untuk dilakukan proses penghitungan. Pada proses ini kamera akan mengambil nilai-nilai *pixel* pada frame-frame yang kamera tangkap. Kecepatan kamera untuk proses *preview* kamera yang digunakan adalah 24 fps. Adapun posisi untuk penempatan kamera *smartphone* yang digunakan pada penelitian ini adalah berada di tepi ruangan dengan kemiringan sekitar 75 derajat.



Gambar 3.2 Penempatan posisi *handphone*.

3.1.2.2 Analisis Model Warna YUV ke RGB

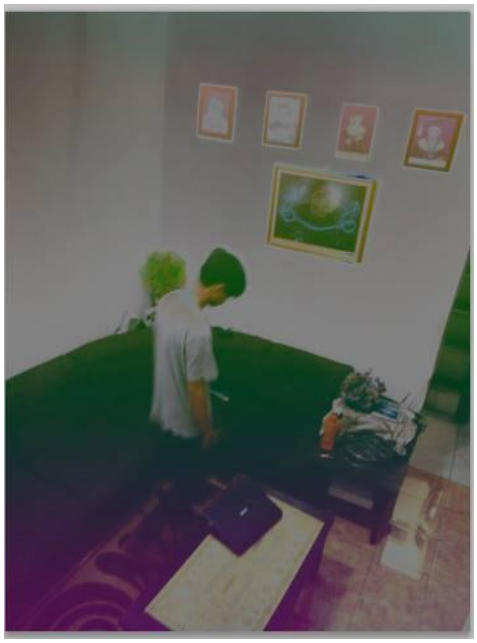
Proses ini dilakukan karena kamera mendapatkan citra dengan format YUV. RGB merupakan suatu model warna yang terdiri atas tiga warna yaitu R(*Red*), G(*Green*), B(*Blue*). Pada format RGB, suatu warna didefinisikan sebagai kombinasi (campuran) dari komponen warna R, G dan B. Proses ini dilakukan karena di tahapan berikutnya penulis akan melakukan proses *grayscaling* guna mempermudah proses perhitungan. Formulasi konversi dari YUV ke RGB adalah:

R = Y + (1.4075 * (V-128));

G = Y – (0.3455 * (U-128) – (0.7169 * (V-128));

B = Y + (1.7790 * (U-128));

Contoh citra YUV:



Gambar 3.3 Citra YUV

Sedangkan Algoritma mengubah YUV ke RGB untuk seluruh *pixel* adalah:

```
Kamus
      ImageHeight : integer
      ImageWidth  : integer
      Image : array[imageHeight][ImageWidth]
      YUVImage : array[ImageHeight][ImageWidth] of integer
      RGBImage : array[ImageHeight][ImageWidth] of integer
Algoritma
      For i← 0 to ImageWidth do
            For j ← 0 to ImageHeight do
                  YUVImage[i][j] ← Image[i][j]
                  RGBImage[i][j] ←
R ← YUVImage[i][j].Y+(1.4075*(YUVImage[i][j].V-128));
G ← YUVImage[i][j].Y-(0.3455*(YUVImage[i][j].U-128)-
(0.716*(YUVImage[i][j].V-128));
B ← YUVImage[i][j].Y+(1.7790*(YUVImage[i][j].U-128));
            Endfor
      Endfor
End
```

Gambar 3.4 Algoritma YUV ke RGB

3.1.2.3 Analisis *Grayscaling*

Tahap selanjutnya adalah tahap *grayscaling* yaitu mengubah citra warna menjadi citra grayscale, hal ini digunakan untuk menyederhanakan model citra dan guna mempermudah proses perhitungan. Untuk memperoleh informasi intensitas dari citra tersebut, citra dapat di sortir secara ekslusif mulai dari hitam untuk intensitas yang paling lemah sampai dengan putih untuk intensitas yang paling kuat. Citra warna terdiri dari 3 layer matrik yaitu R-layer, G-layer, B-layer. Jadi dalam proses ini akan mengubah 3 layer matriks citra berwarna menjadi 1 layer matriks *grayscale* dan hasilnya citra *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing R, G dan B menjadi citra *grayscale* dengan nilai I. Berikut adalah perhitungan untuk mendapatkan *grayscale* :

$$Grayscale = (\mathrm{R} \times 0{,}21) + (\mathrm{G} \times 0{,}77) + (\mathrm{B} \times 0{,}02)$$

Contoh citra berwarna (RGB):



Gambar 3.5 Citra Berwarna RGB

Contoh citra yang telah menjadi *grayscale*:



Gambar 3.6 Citra *Grayscale*

Sedangkan algoritma *grayscale* adalah sebagai berikut:

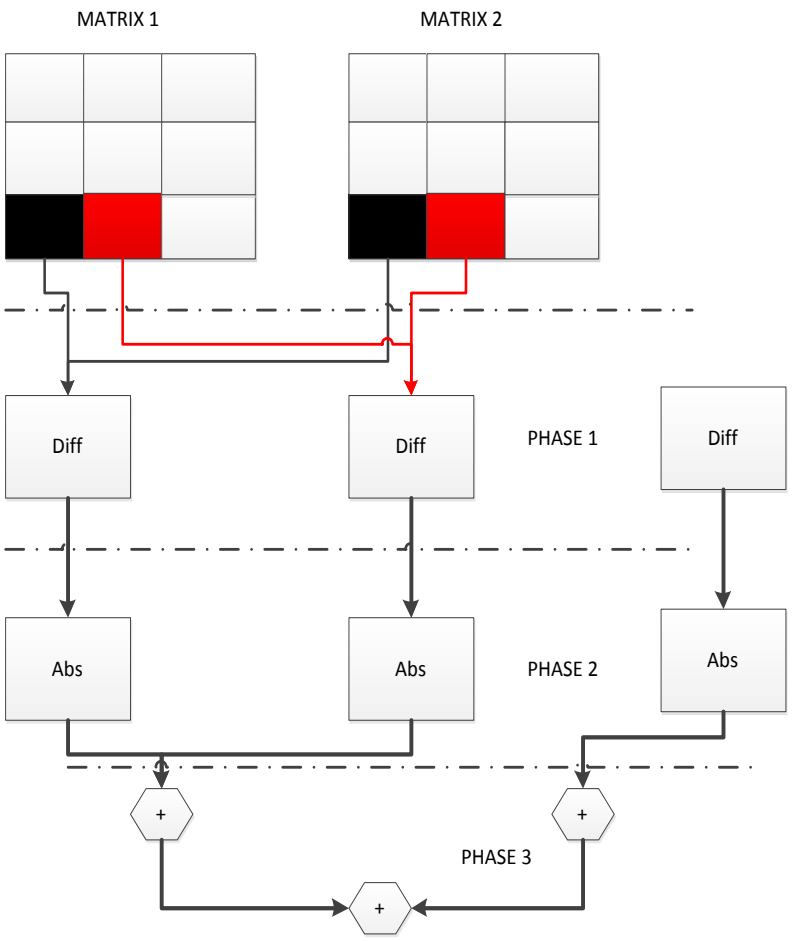
```
Kamus
      ImageHeight : integer
      ImageWidth  : integer
      Image : array[imageHeight][ImageWidth]
      RGBImage : array[ImageHeight][ImageWidth] of integer
      GrayImage : array[ImageHeight][ImageWidth] of integer
```

```
Algoritma
       For i← 0 to ImageWidth do
              For j ← 0 to ImageHeight do
                     RGBImage[i][j] ← Image[i][j]
                     GrayImage[i][j] ←
                            0.21 * RGBImage[i][j].R +
                            0.77 * RGBImage[i][j].G +
                            0.02 * RGBImage[i][j].B
              Endfor
       Endfor
End
```

Gambar 3.7 Algoritma RGB to *Grayscale*

3.1.2.4 Analisis *Sum of absolute differences* (SAD)

Setelah melakukan proses preview kamera, mengubah citra YUV ke citra RGB dan mengubah citra RGB menjadi citra *gray*, langkah selanjutnya adalah menghitung nilai *Sum of absolute differences* (SAD) pada setiap frame yang telah dikonversi menjadi *grayscale*. Pada tahap ini, tiap *pixel* akan dibandingkan dengan *pixel* lain. *Sum of absolute differences*(SAD) ini bekerja dengan cara mengambil prebedaan mutlak (*Absolute Differences*) antara setiap *pixel* di citra awal dengan citra-citra berikutnya. Illustrasi langkah-langkah metode SAD adalah sebagai berikut:



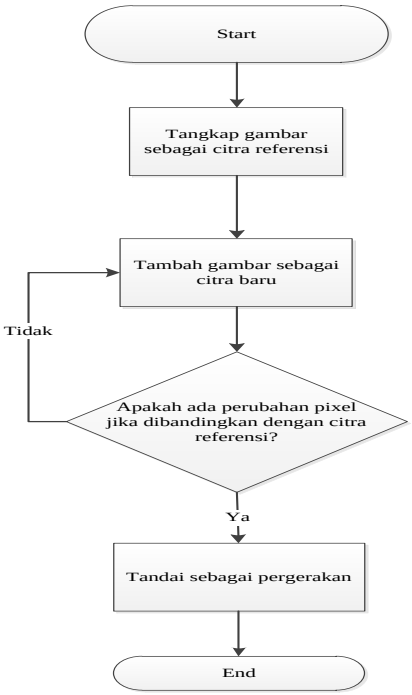
Gambar 3.8 Ilustrasi menghitung *pixel* antar frame

Berikut penjelasan ilustrasi menghitung antar frame :

1. Bandingkan selisih *pixel* 1 di matrix 1 dan *pixel* 1 di matrix 2, *pixel* 2 pada matrix 1 dan *pixel* 2 pada matrix 2 dan begitu selanjutnya. Matrix disini dianalogikan sebagai frame.
2. *Absolute*-kan nilai hasil selisih *pixel* antar frame.
3. Jumlahkan seluruh selisih *pixel* tersebut sehingga mendapatkan nilai SAD nya.

Proses pendeteksian gerak dengan menggunakan *Sum of absolute differences* (SAD) dimulai dengan menangkap gambar sebagai citra referensi atau citra awal. Karena pada penelitian ini menggunakan kamera secara *real-time*, citra yang menjadi referensi adalah citra yang pertama kali kamera tangkap. Langkah

selanjutnya kamera akan menambah gambar sebagai citra baru yang akan dibandingkan dengan citra awal. Jika tidak ada perubahan nilai *pixel* dibandingkan dengan citra awal maka kamera akan menambah citra baru untuk kembali dibandingkan dan jika ada perubahan *pixel* maka akan ditandai sebagai pergerakan. Berikut ini merupakan *flowchart* deteksi gerak dengan menggunakan *Sum of absolute differences* (SAD).



Gambar 3.9 *Flowchart* deteksi gerak dengan SAD

Untuk Setiap frame, lakukan pemrosesan setiap *pixel* yang menggunakan persamaan *Sum of absolute differences* (SAD) berikut:

$$\text{SAD} = \sum_{i=1}^{n} \sum_{j=1}^{n} |i - j|$$

Dimana:

i= frame i.

j= frame j.

n= jumlah *pixel* pada frame.

Berikut algoritma proses perbandingan antar frame dengan *Sum of absolute differences* :

```
Procedure getFrameDifference()
Kamus
      i,j,f1,f2,f3,Fd : integer
      pixel : array [ih][iw] of double
Algoritma
      Pixel  = new double [ih][iw];
                  For i ← 0 to i < ih do
                              For j ← 0 to j <iw do
                  Fd ← ((absolute)(f1[i][j] – f2[i][j])) +
                  ((absolute) (f2[i][j] – f3[i][j]));
                               Endfor
                    Endfor
```

Gambar 3.10 Algoritma proses perbandingan antar frame SAD

3.1.2.4.1 Contoh penerapan Pemrosesan SAD

Contoh penerapan permrosesan SAD dapat dilihat pada Gambar 3.11 dan Gambar 3.13. Dengan membandingkan frame 1 sebagai citra awal dan frame 2 sebagai citra baru.



Gambar 3.11 Citra awal Frame 1 RGB



Gambar 3.12 Citra awal Frame 1 *Grayscale*



Gambar 3.13 Citra Frame 2



Gambar 3.14 Citra Frame 2 *Grayscale*

Contoh pemrosesan SAD adalah sebagai berikut. yang disebut pergerakan pada aplikasi ini adalah ketika nilai pixel antar frame berubah. Berikut adalah contoh nilai pixel dengan luas sebesar 20x15 pixel.

x/y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	103	115	90	94	101	89	100	101	115	123	127	115	171	115	128
2	105	115	100	110	115	114	116	120	150	123	124	90	157	143	151
3	121	125	134	120	134	115	120	115	140	121	137	145	134	115	115
4	132	132	115	132	131	129	129	131	124	242	141	232	157	143	102
5	140	134	122	123	142	131	223	220	151	123	128	231	252	115	143
6	150	151	125	144	151	127	204	206	132	123	127	232	127	128	115
7	166	167	145	123	121	120	205	205	141	123	128	142	123	115	151
8	128	187	152	141	132	134	206	210	221	125	141	148	136	128	115
9	161	123	132	161	122	124	214	220	213	126	134	134	138	102	115
10	161	162	121	123	123	122	102	105	123	127	143	123	138	115	102
11	161	167	142	141	145	147	151	157	210	125	134	132	90	115	115
12	165	168	143	156	156	158	162	168	123	225	123	152	100	102	143
13	164	154	123	176	171	178	181	187	213	242	242	241	109	115	115
14	128	178	163	189	181	185	190	200	123	123	123	232	106	143	141
15	133	127	164	123	121	125	120	127	123	213	125	213	117	115	115
16	180	127	167	143	143	147	136	138	125	128	131	241	114	115	143
17	140	115	168	123	144	150	151	162	172	157	135	232	126	143	115
18	142	167	165	123	123	128	157	181	123	124	137	123	129	128	102
19	132	144	123	121	151	155	156	128	123	125	139	134	125	128	115
20	156	115	115	131	167	168	169	232	141	131	123	156	138	120	115

Gambar 3.15 Nilai pixel citra frame 1

x/y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	107	120	100	80	101	75	121	104	115	123	127	115	171	115	128
2	105	115	100	110	115	114	116	120	150	123	124	90	157	143	151
3	124	125	134	120	134	115	120	115	140	121	137	145	134	115	115
4	135	132	115	141	125	129	130	131	124	242	141	232	157	143	102
5	142	134	122	123	142	131	223	220	151	123	128	231	252	115	143
6	150	151	125	144	152	127	204	206	132	123	127	232	127	128	115
7	166	167	145	123	121	120	205	205	141	123	128	142	123	115	151
8	128	187	152	141	142	134	206	210	221	125	141	148	136	128	115
9	161	123	132	161	122	124	214	220	213	126	134	134	138	102	115
10	161	162	121	123	123	122	102	105	123	127	143	123	138	115	102
11	161	167	142	141	145	147	151	157	210	125	134	132	90	115	115
12	165	168	143	156	156	158	162	168	123	225	123	152	100	102	143
13	164	154	123	176	171	178	181	187	213	242	242	241	109	115	115

14	128	178	163	189	181	185	190	200	123	123	123	232	106	143	141
15	133	127	164	123	121	125	120	127	123	213	125	213	117	115	115
16	180	127	167	143	143	147	136	138	125	128	131	241	114	115	143
17	140	115	168	123	144	150	151	162	172	157	135	232	126	143	115
18	142	167	165	123	123	128	157	181	123	124	137	123	129	128	102
19	132	144	123	121	151	155	156	128	123	125	139	134	125	128	115
20	156	115	115	131	167	168	169	232	141	131	123	156	143	115	141

Gambar 3.16 Nilai pixel Citra frame 2

Nilai SAD dihitung melalui perbandingan antara citra frame 1 sebagai citra awal dan frame 2 sebagai citra baru. Pada citra frame 1 dan frame 2 memiliki nilai pixel-pixel. Perhitungan SAD dilakukan terhadap pixel-pixel ini. Jumlah pixel dinyatakan dengan n. berikut perhitungannya:

$SADf_{1,2}$ = |103-137| + |115-120| + |90-100| +

|94-80| + |101-101| + |89-75| +

……

|138-143| + |102-141| + |115-141|

$SADf_{1,2}$ = 2034

Resolusi pixel yang digunakan adalah sesuai resolusi kamera pada perangkat. Untuk mempercepat proses perhitungan deteksi gerak menggunakan *Sum of absolute differences* (SAD) maka pada tahap ini dilakukan normalisasi nilai. Dari nilai *pixel* citra yang ditampung ke dalam array, akan dilakukan penjumlahan selisih *pixel* antar frame. Bila nilai selisih *pixel* antar frame telah didapat dan telah di-*absolute*-kan, maka nilai tersebut akan dibagi 255. Misalkan selisih 1 *pixel* antar frame adalah 80 setelah di-*absolute*-kan, nilai 80 akan dibagi 255 dan menghasilkan nilai 0.31, maka nilai 0.31 inilah yang akan disimpan ke array menggantikan nilai 80. Setelah proses tersebut dilakukan berulang sampai selisih setiap *pixel* dijumlahkan, maka akan didapat nilai SAD yang lebih kecil. Berikut perhitungannya:

$SADf_{1,2} = \left|\frac{103-137}{255}\right| + \left|\frac{115-120}{255}\right| + \left|\frac{90-100}{255}\right| +$

$$\left|\frac{138-143}{255}\right| + \left|\frac{102-141}{255}\right| + \left|\frac{103-137}{255}\right|$$

……

$$\left|\frac{103-137}{255}\right| + \left|\frac{103-137}{255}\right| + \left|\frac{115-141}{255}\right|$$

$SADf_{1,2} = 7$

3.1.2.5 *Threshold*

Tahapan selanjutnya adalah *thresholding* atau penentuan nilai ambang batas untuk menentukan ada atau tidaknya pergerakan pada suatu daerah pemantauan. Pergerakan yang akan di *capture* secara otomatis adalah apabila terdapat gerakan dengan nilai hasil persamaan SAD antar frame yang melebihi nilai *threshold* yang telah ditentukan. Dalam hal ini beberapa nilai *threshold* diujicobakan dari nilai 100 – 600. Penentuan nilai *threshold* dilihat dengan cara melihat hasil persamaan SAD antar frame pada saat *capture* otomatis pada saat kamera di arahkan pada suatu ruangan yang kosong untuk mendeteksi pergerakan. Proses ini dilakukan berulang-ulang dan menggunakan jarak pantau yang berbeda serta merubah nilai *threshold* yang berulang-ulang juga untuk mendapatkan nilai yang cocok. Sehingga diperoleh nilai threshold yang cocok. Nilai threshold yang cocok adalah 300 karena nilai 300 adalah nilai tengah dari nilai yang diujicobakan dari batas minimum 100 – 600 dengan nilai 300 aplikasi dapat mendeteksi pergerakan orang berjalan dengan jarak maksimal 6 meter ruangan yang diawasi. Jika nilai hasil persamaan SAD antar frame yang didapat > 300 maka dipastikan terdapat objek yang bergerak dan akan di *capture* secara otomatis.

Berikut contoh deteksi gerak dengan menentukan nilai *threshold* pada 4 frame:



Gambar 3.17 SAD Frame 1 dan Frame 2

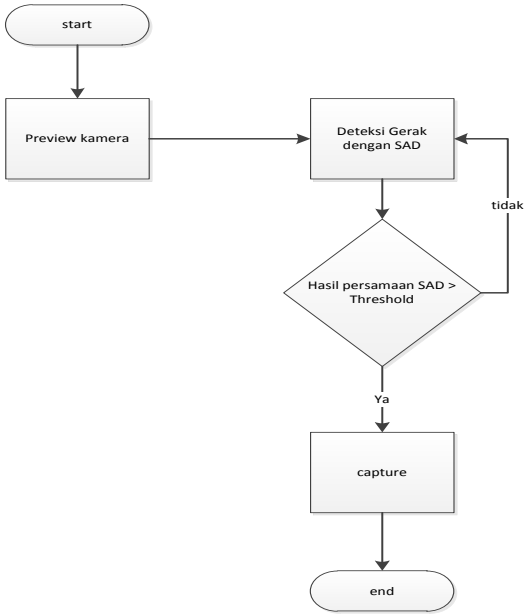


Gambar 3.18 SAD Frame 1 dan Frame 2

Pada gambar 3.17 terdapat nilai SAD antar frame kurang dari 300, jika kondisi seperti itu maka aplikasi akan mengabaikan pergerakan tersebut karena aplikasi menganggap bahwa pada ruangan yang diawasi tidak terdapat pergerakan. Pada gambar 3.18 terdapat nilai SAD antar frame lebih dari 300, jika kondisi seperti itu maka aplikasi akan meng-*capture* secara otomatis pergerakan tersebut dan menganggap bahwa pada ruangan yang diawasi terdapat pergerakan.

3.1.2.6 Proses *Capture*

Proses *capture* merupakan proses menangkap gambar secara otomatis setelah pergerakan terdeteksi. Proses *capture* ini dilakukan dengan pengambilan gambar pada *preview* kamera secara *real-time*. Selanjutnya dilakukan proses deteksi gerak dengan SAD. Jika hasil persamaan SAD melebihi nilai *threshold* yang telah ditentukan maka aplikasi akan maka aplikasi akan meng-*capture* secara otomatis pergerakan tersebut.



Gambar 3.19 *Flowchart* proses *capture*

Gambar hasil *capture* kamera:



Gambar 3.20 Gambar hasil *capture*

3.1.2.7 Proses *resize* gambar

Setelah gambar di *capture* oleh aplikasi tahapan selanjutnya adalah *resize* gambar agar mempercepat proses upload. Gambar yang didapat dari hasil *capture* kamera adalah gambar dengan resolusi 2448x3264 yang akan di *resize* menjadi 256x361. Pada penilitian proses merubah ukuran gambar dilakukan dengan

menggunakan salah satu *built in* di android programing yaitu *resize bitmap*. Berikut adalah tahap tahapannya:

1. Inisialisasi *bitmap*

```
Bitmap result = Bitmap.createScaledBitmap(source, newWidth, newHeight,
filter);
```

Gambar 3.21 Inisialisasi *bitmap*

Dari syntax diatas hasil *resize* akan ditampung di result. Parameter yang dikirim ke fungsi *createScaledBitmap* adalah *Bitmap source* atau gambar yang mau di *resize*, *width* dan *height* yang diinginkan.

2. *Resize image*

Selanjutnya setelah inisialisasi bitmap yang bertujuan untuk menambah gambar baru untuk hasil *resize*. Gambar yang akan di *resize* pada penelitian ini adalah gambar ukuran 2448x3264 menjadi 256x341.

```
public static Bitmap getResizedBitmap(Bitmap bm){
float height2;
float width2;

int width = bm.getWidth();
int height = bm.getHeight();

Bitmap resizeBitmap = null;

//potrait
width2 = 256;
height2 = ((float)256/width)*height;
resizeBitmap = Bitmap.createScaledBitmap(bm, Math.round(width2),
Math.round(height2), true);

}
```

Gambar 3.22 *Resize Image*

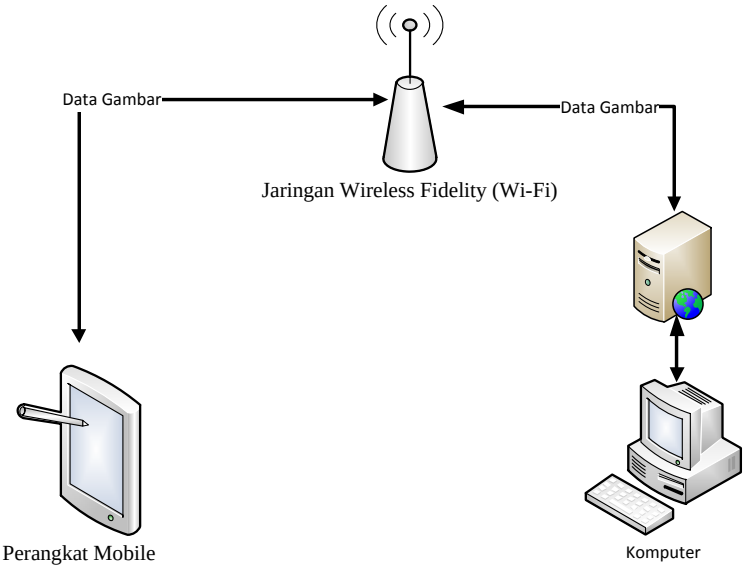
Penjelasan dari fungsi diatas adalah pertama buat function get*Resize*dBitmap, lalu inisialisasi variable *width*, *height* itu ukuran *bitmap source.* Width2, height2 itu adalah ukuran bitmap yang akan dicapai. Karena pada penelitian ini akan me-*resize* dalam bentuk portrait, maka width atau lebarnya kita buat 256 dan height nantinya akan lebih dari 341. Cara menghitung height barunya adalah dengan (256/width lama) x height lama. Lalu *resize* berdasarkan height dan width yang sudah di dapatkan.



Gambar 3.23 *resize* ukuran gambar.

3.1.2.8 Proses Upload

Setelah gambar di *capture* oleh aplikasi, maka aplikasi akan mengupload gambar hasil *capture* secara otomatis ke komputer *server* dengan cara menghubungkan perangkat handphone dan komputer ke dalam satu jaringan *wireless Fidelity (Wi-Fi).*

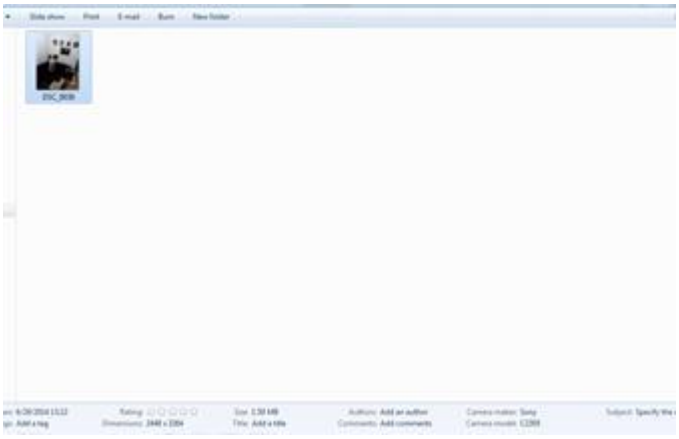


Gambar 3.24 Proses Upload ke komputer menggunakan wireless Fidelity (wi-fi)

Berikut penjelasan mengenai langkah-langkah upload ke komputer *server*:

1. Hubungkan perangkat handphone dan komputer ke dalam suatu jaringan.
2. Konfigurasikan IP-*address* komputer *server* pada aplikasi. Jika perangkat mobile dapat mengakses web *server* maka komputer dan perangkat mobile dapat melakukan proses komunikasi data.
3. Ketika aplikasi meng-*capture* pergerakan tersebut maka aplikasi akan secara otomatis meng-*upload* gambar hasil *capture* tersebut ke komputer**.**

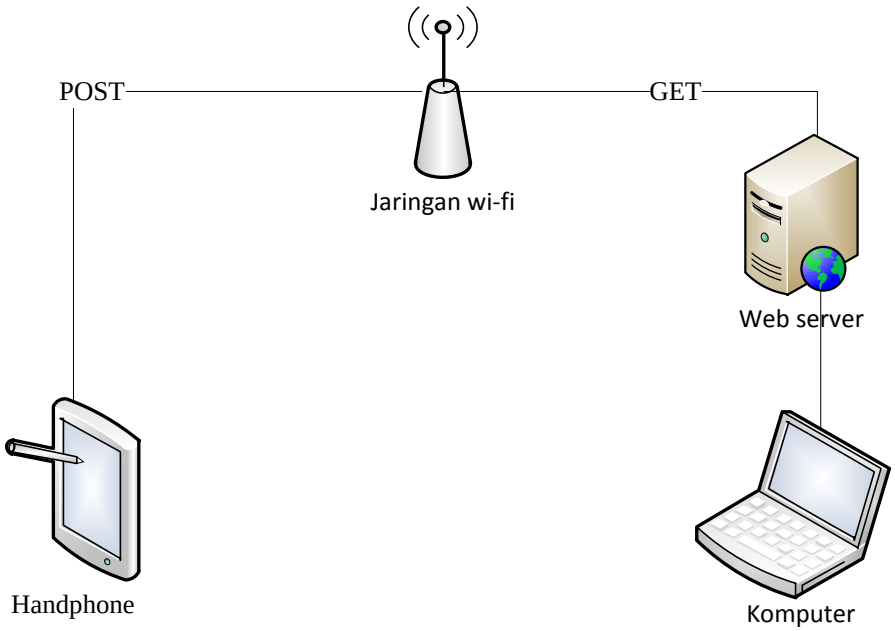
Setelah proses-proses tersebut maka akan menghasilkan output berupa gambar yang telah berada di *folder* komputer.



Gambar 3.25 *Output* gambar telah berada di komputer

3.1.3 Analisis Komunikasi Data

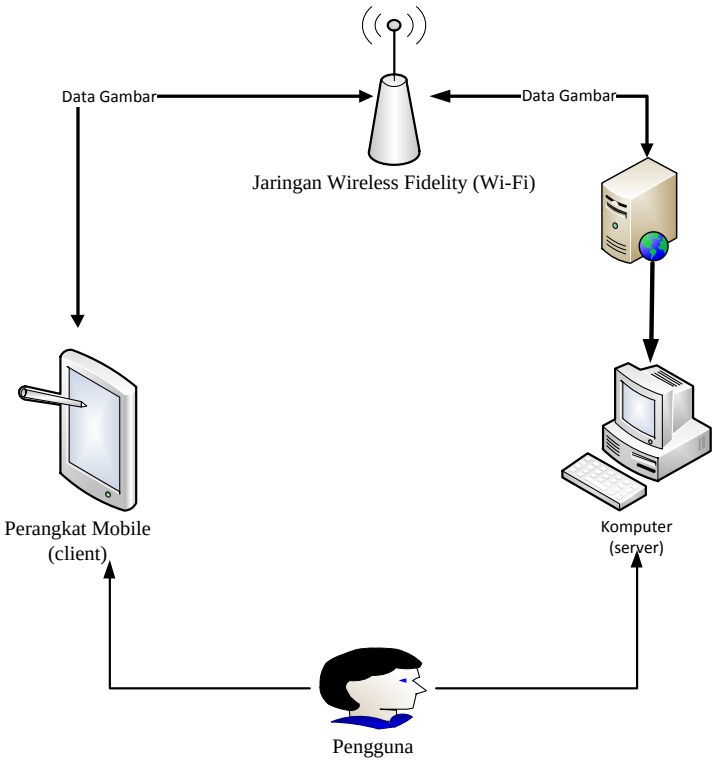
Analisis komunikasi data ini merupakan tahapan analisis untuk proses komunikasi data antara perangkat mobile atau smartphone sebagai *client* dan komputer sebagai *server*. Topologi jaringan yang digunakan dalam aplikasi pemantauan atau monitoring pada suatu ruangan diperlihatkan pada gambar. Pada topologi jaringan ini perangkat mobile dan komputer akan dihubungkan dengan menggunakan jaringan Wireless Fidelity (Wi-Fi). perangkat mobile akan mengirimkan data gambar dan komputer akan mengambil data gambar tersebut untuk ditampilkan. Namun, jika jaringan wi-fi mati atau terganggu maka semua perangkat yang dihubungkan ke jaringan wi-fi tersebut tidak dapat saling berkomunikasi. Client akan menjalankan aplikasi *auto capture* dan mengirimkan data gambar ke komputer *server* untuk ditampilkan. Instruksi permintaan data oleh web *server* dikemas dalam TCP yang merupakan *protocol transport* dan dikirim ke *protocol* HTTP. Web *server* akan mencari data *byte* yang dikemas dalam TCP *protocol*. Jika perangkat mobile tidak dapat mengakses url ip *address* komputer, maka perangkat yang dihubungkan tidak dapat saling berkomunikasi.



Gambar 3.26 Analisis Komunikasi Data

3.1.4 Arsitektur Sistem

Sistem yang akan dibangun adalah sistem aplikasi pengawasan atau monitoring pada suatu ruangan dengan menggunakan perangkat *mobile* atau *smartphone*. Adapun arsitektur sistem yang akan dibangun yang dapat dilihat pada Gambar 3.27:



Gambar 3.27 Arsitektur sistem

Penjelasan dari arsitektur sistem yang akan dibangun adalah:

1. Perangkat Mobile

   Perangkat mobile dalam aplikasi *auto capture* berdasarkan pergerakan objek berfungsi sebagai perangkat yang digunakan untuk memonitor sebuah ruangan. Perangkat mobile ini dapat mengenali sebuah pergerakan, akan meng-*capture* setelah pergerakan terdeteksi dan akan mengirim gambar hasil *capture* ke komputer.
2. Jaringan *Wireless Fidelity (Wi-Fi)*

Jaringan *Wireless Fidelity (Wi-Fi)* dalam aplikasi ini adalah sebuah jaringan yang menghubungkan antara perangkat mobile dan komputer.

3. Web *Server*

   Web *Server* dalam aplikasi auto capture berdasarkan pergerakan objek berfungsi sebagai sebuah aplikasi untuk proses komunikasi data antara perangkat *mobile* dan komputer.

4. Komputer

   Komputer dalam aplikasi *auto capture* berdasarkan pergerakan objek berfungsi sebagai *server* yang didalamnya terdapat aplikasi web *server*. Agar komputer ini dapat diakses oleh *client* maka komputer *server* juga harus terhubung ke jaringan. Komputer yang berperan sebagai web *server* memiliki IP *address* sehingga ketika komputer dan perangkat mobile terhubung ke jaringan maka komputer dapat menampilkan gambar hasil capture.

3.1.5 Analisis Kebutuhan Sistem

Kebutuhan sistem yang diperlukan dan yang digunakan dalam pembuatan sistem ini terdiri dari beberapa bagian, yaitu: kebutuhan fungsional dan kebutuhan non fungsional.

Tabel 3.1 Kebutuhan Fungsional

Nomor	Keterangan
SKPL-F-01	Sistem dapat mendeteksi pergerakan pada *preview* kamera.
SKPL-F-02	Sistem dapat meng-*capture* otomatis setelah pergerakan terdeteksi.
SKPL-F-03	Sistem dapat meng-*upload* otomatis gambar hasil *capture* ke komputer.

Tabel 3.2 Kebutuhan Non-Fungsional

Nomor	Keterangan
SKPL-NF-01	Sistem dibangun dalam bentuk aplikasi *mobile.*
SKPL-NF-02	Sistem dapat dijalankan pada ponsel android minimal versi 2.2 (*froyo*).

SKPL-NF-03	Pada perangkat ponsel harus memiliki kamera belakang minimal 2 *megapixel*.

3.1.6 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsionalitas merupakan analisis yang yang dibutuhkan untuk menentukan spesifikasi yang diperlukan. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk aplikasi yang akan dibangun. Kebutuhan non-fungsionalitas yang dibutuhkan untuk membangun aplikasi ini terdiri dari tiga hal, yaitu : analisis pengguna, analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

3.1.6.1 Analisis Pengguna

Pengguna aplikasi ini adalah *user* atau masyarakat umum yang terbiasa mengoperasikan komputer dan aplikasi *mobile* serta, terbiasa dengan perangkat *mobile* dengan *display* layar sentuh, karakteristik pengguna pada aplikasi ini dapat dilihat pada Tabel 3.3.

Tabel 3.3 Analisis Pengguna

No	Tipe	Hak Akses	Tingkat Keterampilan
1	Pengguna Aplikasi *mobile*	Memulai pengawasan atau monitoring pada suatu ruangan	Mampu mengoperasikan smartphone android.
2	Pengguna Komputer	Melihat Gambar hasil *capture* dari perangkat mobile	Mampu mengoperasikan komputer.

3.1.6.2 Analisis Kebutuhan Perangkat keras

Perangkat keras yang dibutuhkan berdasarkan kebutuhan minimum yang harus terpenuhi yaitu perangkat keras *handphone* yang digunakan untuk menjalankan aplikasi dan perangkat keras komputer yang digunakan untuk membangun dan sebagai *output* aplikasi.

Tabel 3.4 Perangkat Keras Handphone

No	Perangkat Keras	Spesifikasi

1	Processor	600 Mhz
2	RAM	512 MB
3	Memori Internal	4 GB
3	Kamera	8 MP
5	Wi-fi	802.11 b/g/n, DLNA, Wi-Fi hotspot
6	layar	480 x 800 *pixel*s, 4 inches

Tabel 3.5 Perangkat keras Komputer

No	Perangkat Keras	Spesifikasi
1	Processor	Intel Core i3
2	RAM	2 GB
3	Hard Disk	500 GB
4	Monitor	14 inch

3.1.6.3 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak terdiri dari spesifikasi minimum perangakat lunak yang dibutuhkan untuk membangun aplikasi *Auto capture* berdasarkan pergerakan objek. Spesifikasi minimum yang dimaksud dapat dilihat pada Tabel 3.6.

Tabel 3.6 Analisis Kebutuhan Perangkat Lunak

No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 7 32 bit
2	JDK (Java Development Kit)	JDK 4.2.2
3	IDE	Eclipse Juno
4	SDK	Android SDK Windows
5	ADT	ADT v.10.0.0
6	Sistem Operasi Mobile	Android 4.0 ICS
7	Web *Server*	Apache

3.1.7 Analisis Kebutuhan Fungsional

Kebutuhan fungsional berhubungan dengan fitur software yang akan dibuat atau dikembangkan. Berikut adalah tahapan analisis kebutuhan fungsional aplikasi *auto capture* berdasarkan pergerakan objek. Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik. Analisis yang dilakukan dimodelkan dengan menggunakan

UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis tersebut antara lain identifikasi aktor, *Use Case diagram, skenario, sequence diagram, activity diagram, dan class diagram.*

3.1.7.1 Definisi Aktor

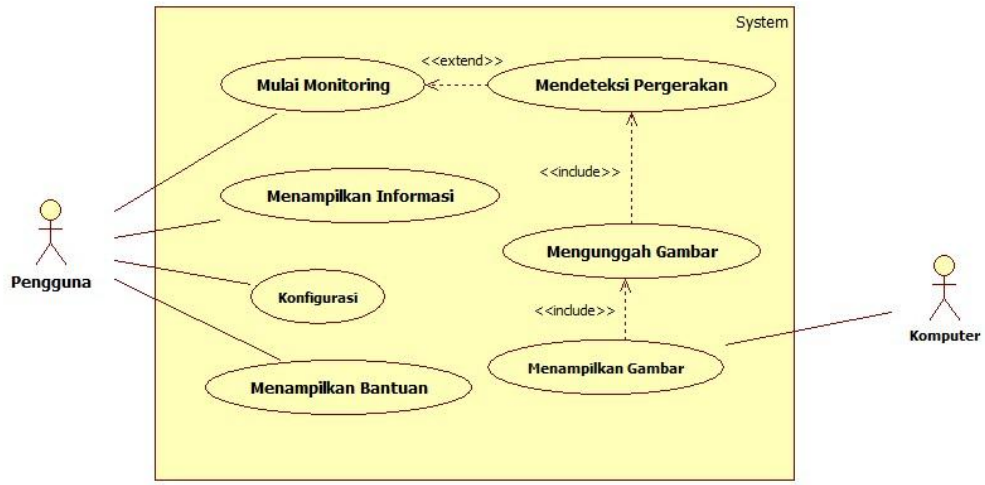
Definisi aktor dimaksudkan untuk mengetahui siapa saja aktor yang terlibat di dalam sistem aplikasi ini.Deskripsi dari tiap aktor yang terlibat dapat dilihat pada Tabel 3.7.

Tabel 3.7 Definisi Aktor

No	Aktor	Deskripsi
1	Pengguna	Pengguna ini merupakan pengguna aplikasi pada platform android. Dimana dalam menggunakan aplikasi *user* dapat menjalankan aplikasi dan *user* dapat melihat hasil *capture* yang telah disimpan di komputer.
2	Komputer	Menampilkan gambar hasil *capture*.

3.1.7.2 *Use Case* Diagram

Use case Diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan sistem yang akan dibangun. Dari hasil analisis aplikasi yang ada maka *use case diagram* pada aplikasi ini dapat dilihat pada Gambar 3.28.



Gambar 3.28 *Use Case* Diagram Aplikasi

3.1.7.3 Definisi *Use Case*

Use case diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan yang disepakati antara pemakai dan pengembang.

Tabel 3.8 Definisi *Use Case*

No	*Use Case*	Deskripsi
UC-01	Mulai Monitoring	Merupakan aksi untuk memulai monitoring.
UC-02	Mendeteksi Pergerakan	Merupakan proses untuk deteksi gerak dan *capture* otomatis setelah pergerakan terdeteksi.
UC-03	Mengunggah Gambar	Merupakan proses upload otomatis gambar hasil *capture*.
UC-04	Menampilkan Informasi	Merupakan aksi untuk menampilkan informasi aplikasi.
UC-05	Konfigurasi	Merupakan aksi untuk menampilkan konfigurasi IP *address*.
UC-06	Menampilkan Bantuan	Merupakan aksi untuk menampilkan bantuan menggunakan aplikasi.
UC-07	Menampilkan Gambar	Merupakan proses untuk menampilkan gambar.

3.1.7.4 Skenario *Use Case*

Skenario *Use Case* mendeskripsikan urutan langkah-langkah dalam proses bisnis, baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor.

3.1.7.4.1 *Use Case* Skenario Mulai Monitoring

Interaksi antara aktor *user* dengan *use case* Mendeteksi Pergerakan dijelaskan dalam Tabel 3.9.

Tabel 3.9 *Use Case* Skenario Mulai Monitoring

Identifikasi		
Nomor	**UC-01**	
Nama *Use Case*	Mulai Monitoring	
Tujuan	Memulai monitoring	
Aktor	Pengguna	
Skenario Utama		
Kondisi Awal	Pengguna Menjalankan Aplikasi	
Aksi Aktor		**Reaksi Sistem**
		1. Menampilkan menu utama aplikasi
2. Menekan tombol mulai		
		3. Menampilkan *preview* kamera.
Skenario Alternatif		
Kondisi Akhir	Memulai monitoring	

3.1.7.4.2 Use Case Skenario Mendeteksi Pergerakan

Interaksi antara aktor *user* dengan *use case* mendeteksi pergerakan dijelaskan dalam Tabel 3.10.

Tabel 3.10 *Use Case* Skenario Mendeteksi Pergerakan

Identifikasi		
Nomor	**UC-02**	
Nama *Use Case*	Mendeteksi Pergerakan	
Tujuan	Mendeteksi pergerakan pada *preview* kamera	
Aktor	Pengguna	
Skenario Utama		
Kondisi Awal	Pengguna berada pada tampilan *preview* kamera.	
Aksi Aktor		**Reaksi Sistem**

	1. Ambil Frame sebagai kondisi awal pada preview kamera.
	2. Konversi dari YUV ke RGB
	3. Konversi dari RGB ke *Grayscale*
	4. Hitung *difference* antar frame
	5. Cek *threshold*
	6. Tandai bahwa pergerakan terdeteksi.
Skenario Alternatif	
	Jika nilai SAD antar frame < *threshold* maka kembali ke ambil frame pada *preview* kamera.
Skenario Alternatif	
Kondisi Akhir	Mendeteksi pergerakan pada *preview* kamera

3.1.7.4.3 *Use Case* Skenario Mengunggah Gambar

Interaksi antara aktor *user* dengan *use case* Mengunggah gambar dijelaskan dalam Tabel 3.11.

Tabel 3.11 *Use Case* Skenario Mengunggah gambar

Identifikasi	
Nomor	**UC-03**
Nama *Use Case*	Mengunggah gambar
Tujuan	Gambar telah berada pada *folder* di komputer.
Aktor	Pengguna
Skenario Utama	
Kondisi Awal	Pergerakan terdeteksi
Aksi Aktor	**Reaksi Sistem**
	1. *Capture* Pergerakan pada frame.
	2. Upload gambar secara otomatis ke komputer.
Skenario Alternatif	
	1. Jika handphone dan komputer tidak terhubung pada satu jaringan maka aplikasi tidak akan meng-*upload* gambar hasil *capture* dan menampilkan pesan salah konfigurasi IP.
Kondisi Akhir	Gambar telah berada pada *folder* di komputer.

3.1.7.4.4 *Use Case* Skenario Informasi

Interaksi antara aktor *user* dengan *use case* informasi dijelaskan dalam Tabel 3.12.

Tabel 3.12 *Use Case* Skenario Informasi

Identifikasi		
Nomor	**UC-04**	
Nama *Use Case*	Menampilkan Informasi	
Tujuan	Menampilkan Informasi aplikasi	
Aktor	Pengguna	
Skenario Utama		
Kondisi Awal	Pengguna berada di tampilan menu aplikasi	
Aksi Aktor		**Reaksi Sistem**
1. Menekan tombol tentang.		
		2. Menampilkan halaman informasi aplikasi.
Kondisi Akhir	Menampilkan Informasi aplikasi	

3.1.7.4.5 Use Case Skenario Konfigurasi

Interaksi antara aktor *user* dengan *use case* bantuan dijelaskan dalam tabel Tabel 3.13.

Tabel 3.13 Skenario Konfigurasi

Identifikasi		
Nomor	**UC-05**	
Nama *Use Case*	Konfigurasi	
Tujuan	Menampilkan alamat Ip.	
Aktor	Pengguna	
Skenario Utama		
Kondisi Awal	Pengguna berada di tampilan menu aplikasi	
Aksi Aktor		**Reaksi Sistem**
1. Menekan tombol konfigurasi		
		2. Menampilkan alamat Ip
		3. Menampilkan nilai *Threshold*
Kondisi Akhir	Menampilkan alamat Ip dan nilai threshold	

3.1.7.4.6 Use Case Skenario Bantuan

Interaksi antara aktor *user* dengan *use case* bantuan dijelaskan dalam Tabel 3.14.

Tabel 3.14 Skenario Bantuan

Identifikasi		
Nomor	**UC-06**	
Nama *Use Case*	Menampilkan Bantuan	
Tujuan	Menampilkan Informasi bantuan menggunakan aplikasi	
Aktor	Pengguna	
Skenario Utama		
Kondisi Awal	Pengguna berada di tampilan menu aplikasi	
Aksi Aktor		**Reaksi Sistem**
1. Menekan tombol bantuan.		
		2. Menampilkan halaman bantuan aplikasi.
Kondisi Akhir	Menampilkan bantuan menggunakan aplikasi	

3.1.7.4.7 *Use Case* Skenario Menampilkan Gambar

Interaksi antara aktor Komputer dengan *use case* menampilkan gambar dijelaskan dalam Tabel 3.15 .

Tabel 3.15 Skenario Menampilkan Gambar

Identifikasi		
Nomor	**UC-07**	
Nama *Use Case*	Menampilkan Gambar	
Tujuan	Menampilkan Gambar di folder komputer	
Aktor	Komputer	
Skenario Utama		
Kondisi Awal	Sistem mengupload gambar otomatis	
Aksi Aktor		**Reaksi Sistem**
		1. Upload Gambar hasil *capture*
2. Menyimpan gambar hasil *capture*.		
Kondisi Akhir	Menampilkan gambar hasil *capture* di folder komputer.	

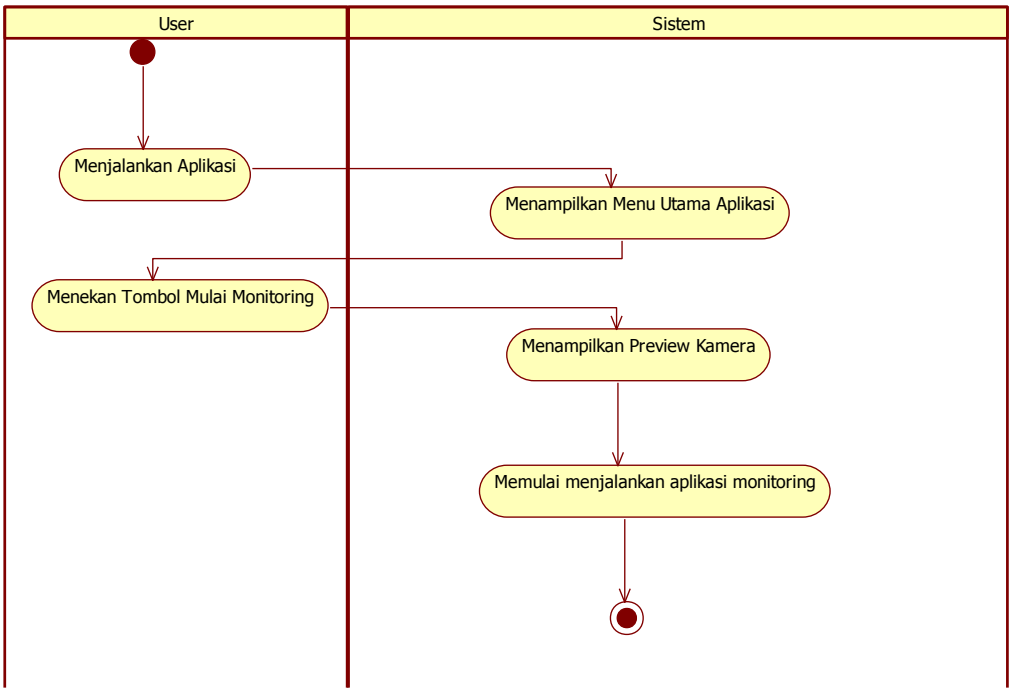
3.1.7.5 *Activity* Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin

terjadi dan bagaimana mereka berakhir. Penggambaran *activity diagram* memiliki kemiripan dengan *flowchart diagram*. *Activity diagram* memodelkan event-event yang terjadi pada *Use Case* dan digunakan untuk pemodelan aspek dinamis dari sistem.

3.1.7.5.1 *Activity* Diagram Mulai Monitoring

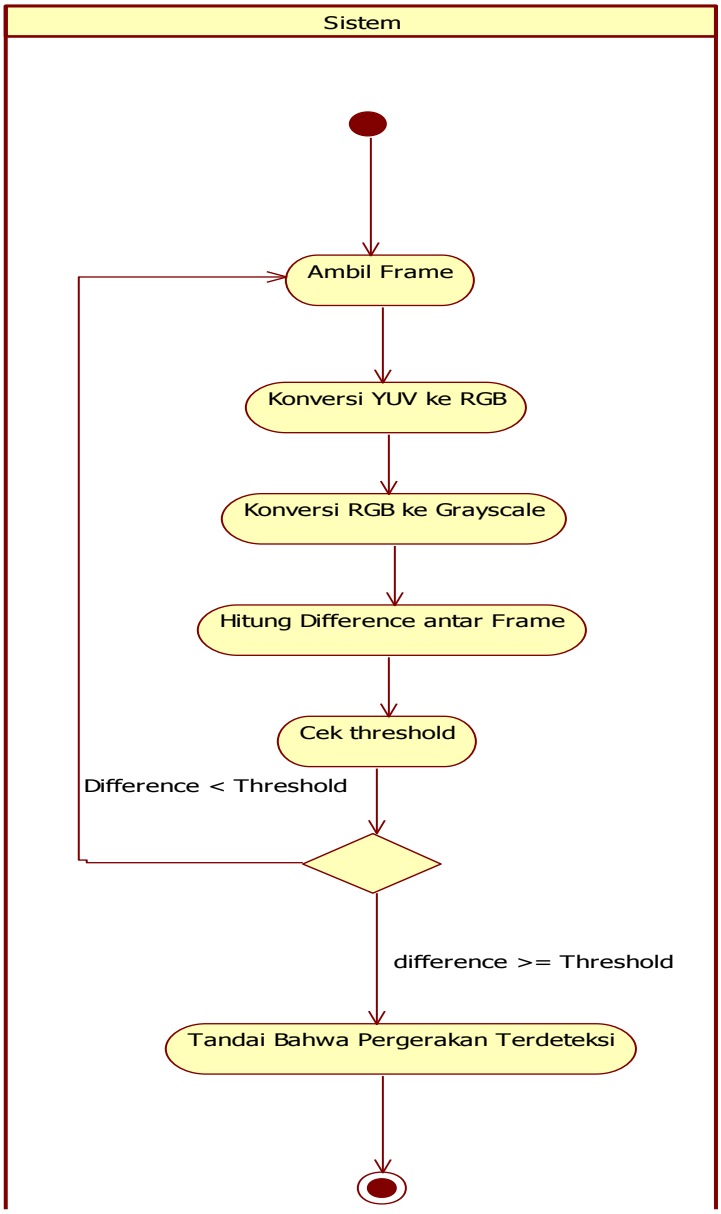
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas mulai monitoring pada aplikasi yang dapat dilihat pada Gambar 3.29.



Gambar 3.29 Activity Diagram Mulai

3.1.7.5.2 *Activity* Diagram Mendeteksi Pergerakan

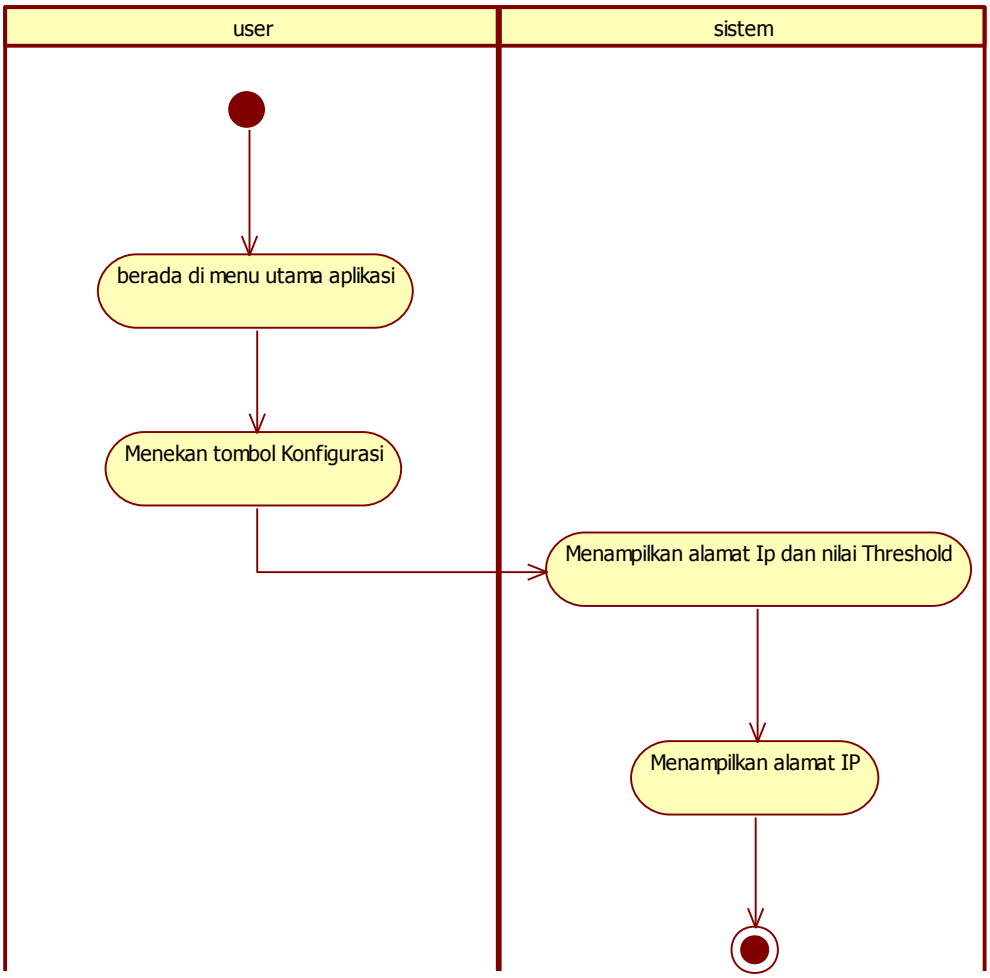
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas mendeteksi pergerakan yang dapat dilihat pada Gambar 3.30.



Gambar 3.30 *Activity* Mendeteksi pergerakan

3.1.7.5.3 *Activity* Diagram Mengunggah Gambar

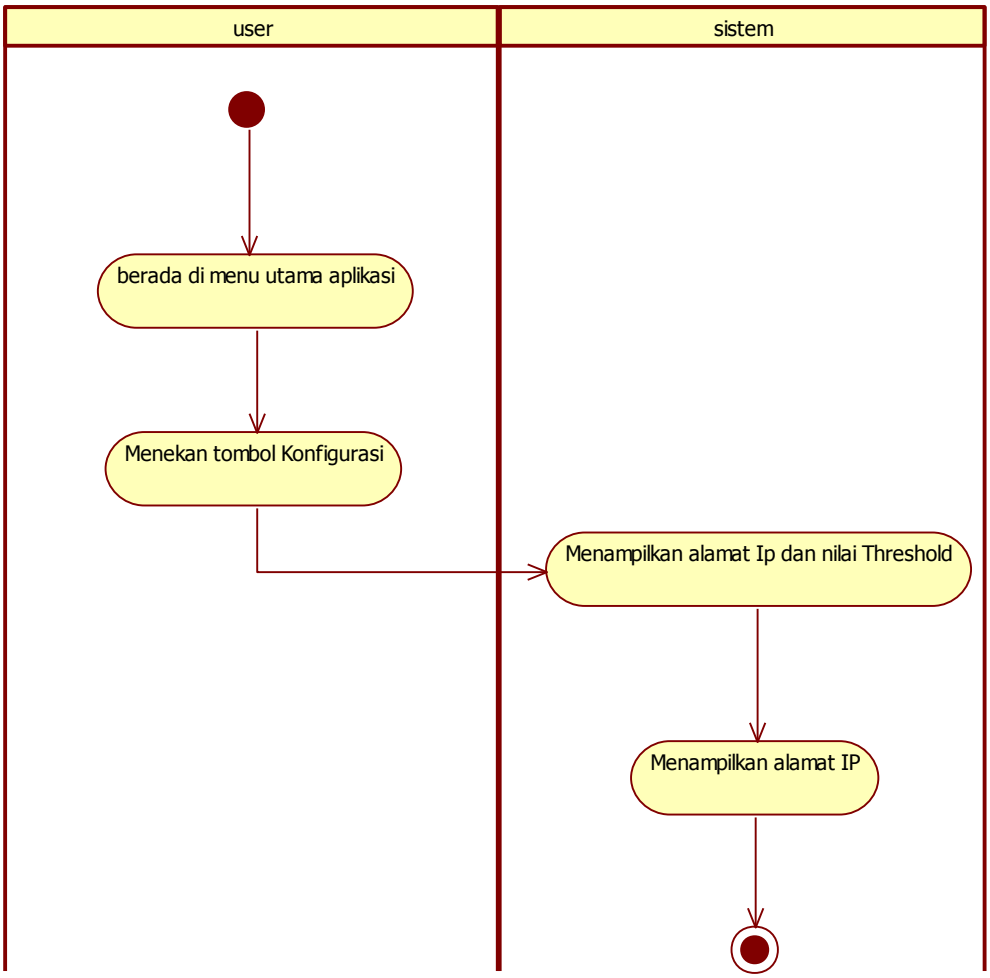
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas mengunggah gambar yang dapat dilihat pada Gambar 3.31.



Gambar 3.31 *Activity* Mengunggah gambar

3.1.7.5.4 Activity Konfigurasi

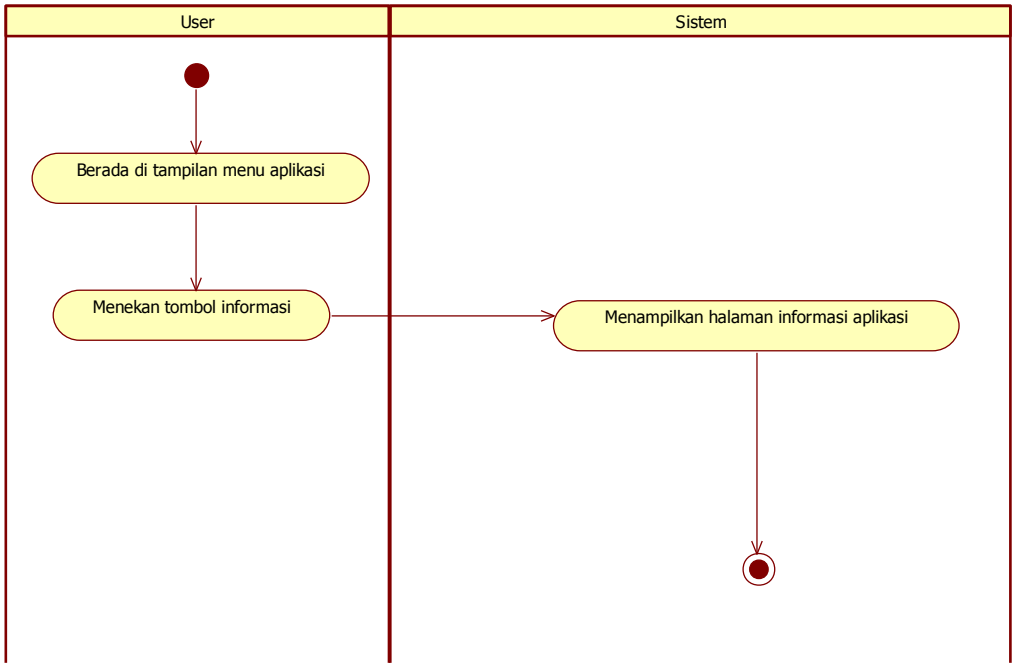
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas konfigurasi yang dapat dilihat pada Gambar 3.32.



Gambar 3.32 *Activity* Diagram Konfigurasi

3.1.7.5.5 Activity Informasi

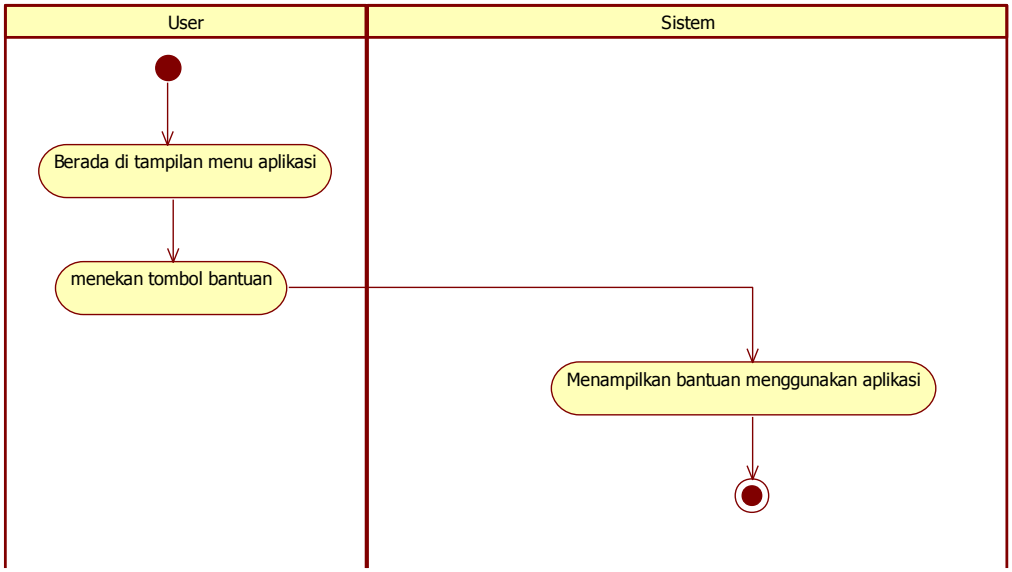
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas menampilkan informasi yang dapat dilihat pada Gambar 3.33.



Gambar 3.33 *Activity* Diagram Informasi

3.1.7.5.6 *Activity* Bantuan

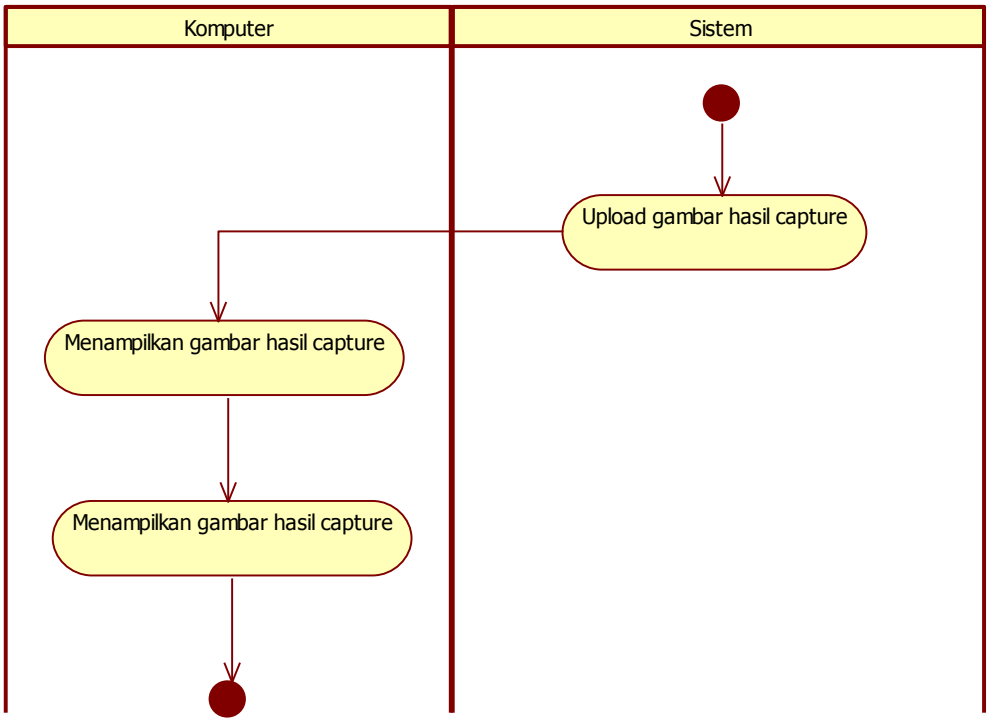
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas menampilkan bantuan menggunakan aplikasi yang dapat dilihat pada Gambar 3.34.



Gambar 3.34 *Activity* Diagram Bantuan

3.1.7.5.7 *Activity* Menampilkan Gambar

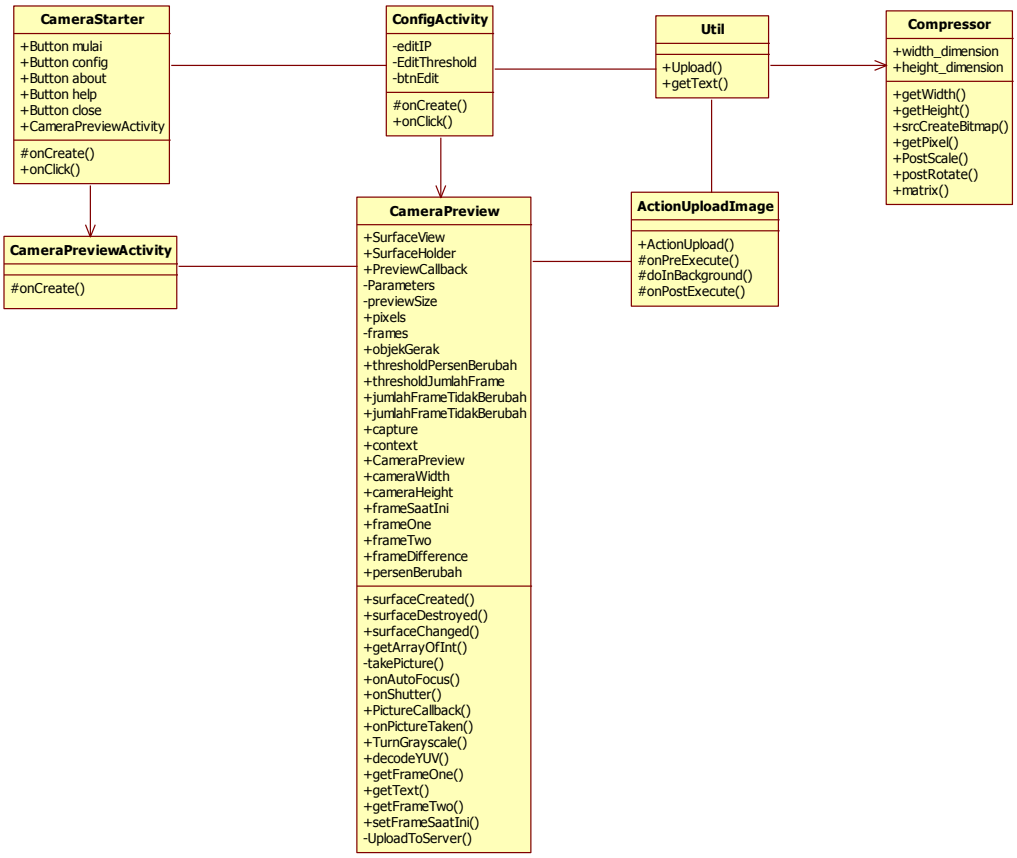
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas menampilkan gambar hasil *capture* di komputer dapat dilihat pada Gambar 3.35.



Gambar 3.35 *Activity* Diagram Menampilkan Gambar

3.1.7.6 *Class* Diagram

Diagram kelas (*Class Diagram*) merupakan diagram structural yang memodelkan sekumpulan kelas, interface, kolaborasi dan relasinya. Diagram kelas digambarkan dengan kotak yang pada dasarnya terbagi atas tiga bagian yaitu: nama kelas, atribut dan operasi. Diagram kelas digunakan untuk menggambarkan proses static dari aplikasi pengukuran tinggi objek ini. berikut ini adalah class diagram dari aplikasi *auto capture* berdasarkan pergerakan objek dapat dilihat pada Gambar 3.36.



Gambar 3.36 *Class* Diagram

Tabel 3.16 Deskripsi *Class* Diagram

No	Kelas	Jenis Kelas	Deskripsi
1	*CameraStarter*	*Interface*	Merupakan kelas yang digunakan untuk menampilkan halaman utama.
2	*CameraPreviewActivity*	*Interface*	Merupakan kelas yang digunakan untuk menampilkan preview kamera.
3	*ConfigActivity*	*Interface*	Merupakan kelas yang digunakan untuk menampilkan konfigurasi.
4	*CameraPreview*	*Entity*	Merupakan kelas yang digunakan untuk memproses citra dan perhitungan SAD
5	*ActionUploadImage*	*Control*	Merupakan kelas yang

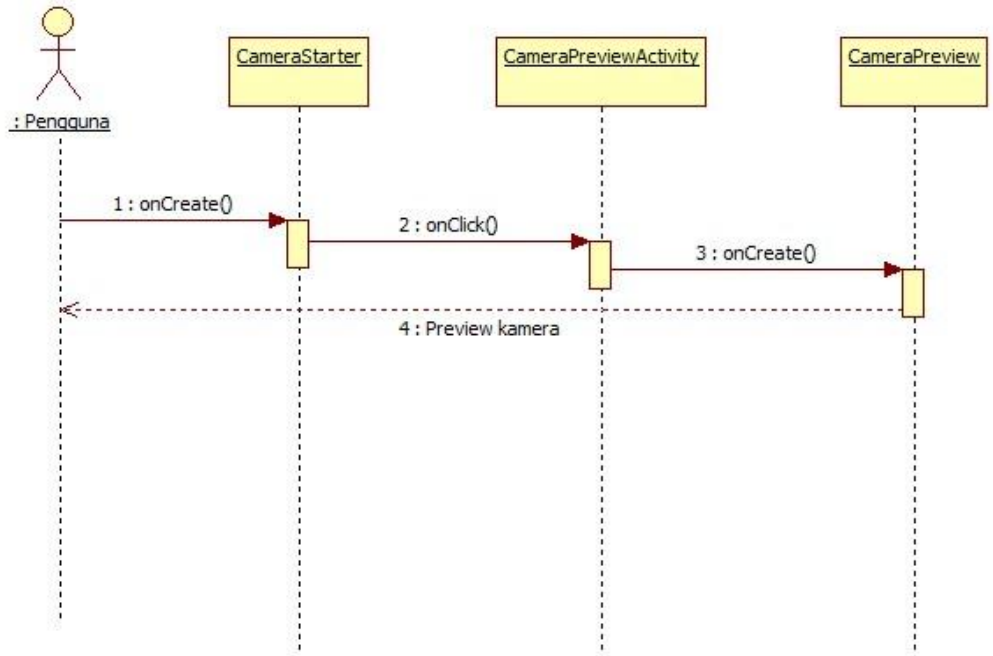
			berisi method untuk *upload* ke komputer.
6	*Util*	*Control*	Merupakan kelas yang berisi method *utilitas* untuk konfigurasi dengan komputer.
7	*Compressor*	*Control*	Merupakan kelas yang berfungsi untuk me-*resize* ukuran gambar.

3.1.7.7 *Sequence* Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan message yang diletakkan diantara obyek-obyek ini didalam *usecase*. Biasa digunakan untuk menggambarkan scenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari sebuah aktivitas tertentu, kemudian berproses mengikuti urutan tertentu yang bisa terlihat melalui *message* antar objeknya.

3.1.7.7.1 *Sequence* Diagram Mulai Monitoring

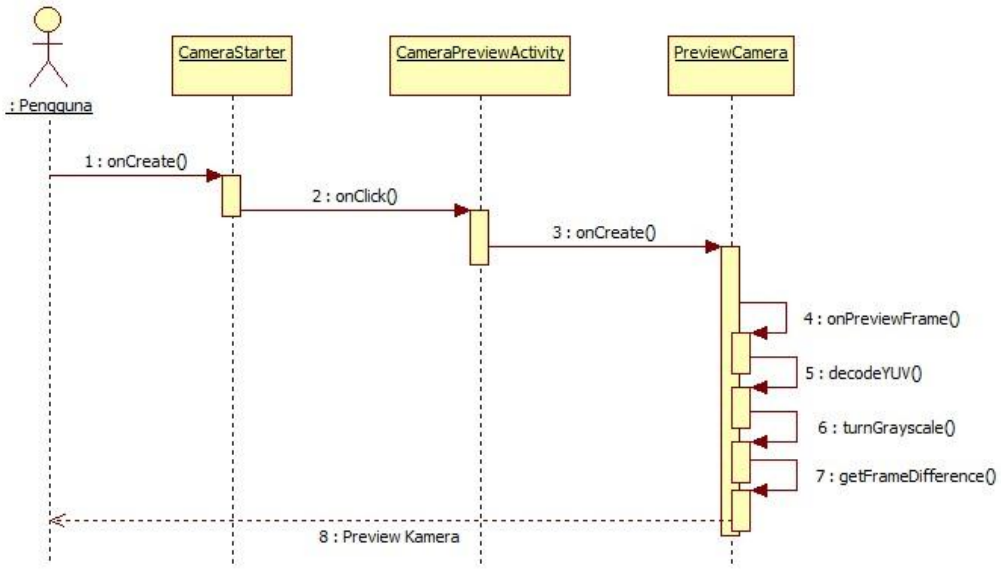
Sequence diagram Mulai merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam mulai aplikasi. *Sequence* diagram mulai terlihat seperti pada Gambar 3.37.



Gambar 3.37 *Sequence* Diagram Mulai Monitoring

3.1.7.7.2 *Sequence* Diagram Mendeteksi Pergerakan

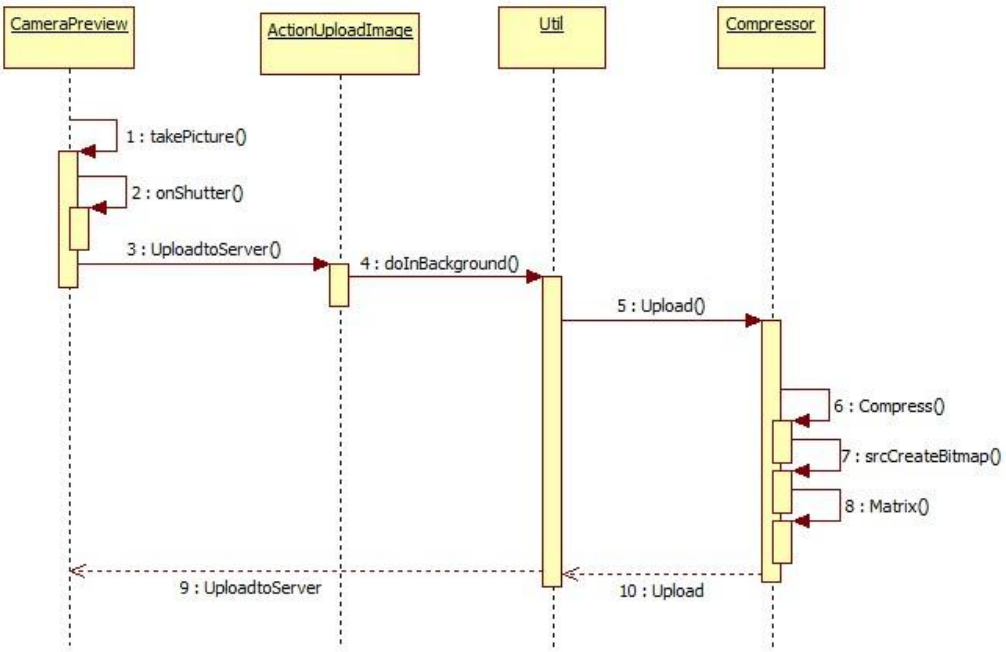
Sequence diagram mendeteksi pergerakan merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam deteksi gerak. *Sequence* diagram mendeteksi pergerakan terlihat seperti pada Gambar 3.38 .



Gambar 3.38 *Sequence* Mendeteksi Pergerakan

3.1.7.7.3 *Sequence* Diagram Mengunggah gambar

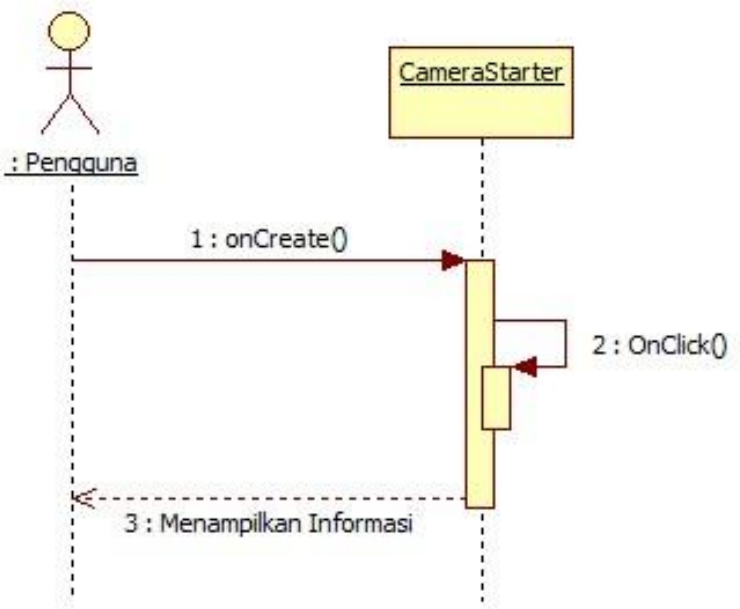
Sequence diagram mengunggah gambar merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam mengunggah gambar. *Sequence* diagram mengunggah gambar terlihat seperti pada Gambar 3.39.



Gambar 3.39 *Sequence* Mengunggah gambar

3.1.7.7.4 *Sequence* Diagram Menampilkan Informasi

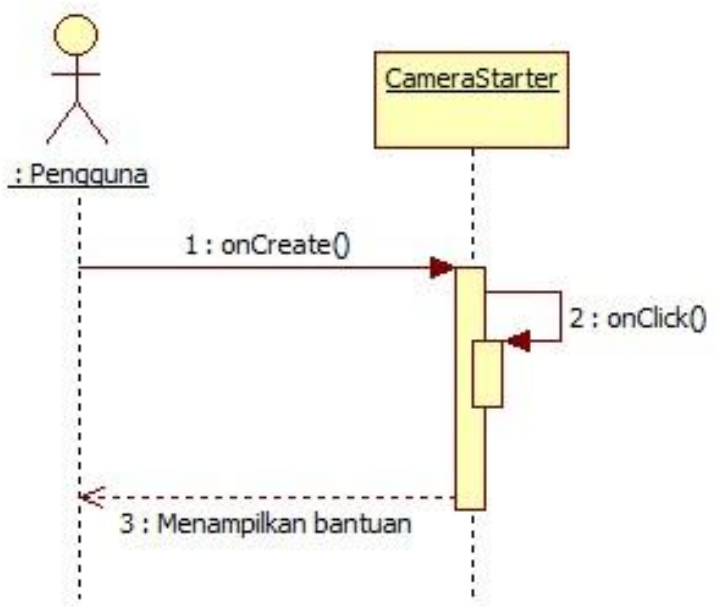
Sequence diagram informasi merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam menampilkan informasi aplikasi. *Sequence* diagram informasi terlihat seperti pada Gambar 3.40.



Gambar 3.40 *Sequence* Diagram Menampilkan Informasi

3.1.7.7.5 *Sequence* Diagram Menampilkan Bantuan

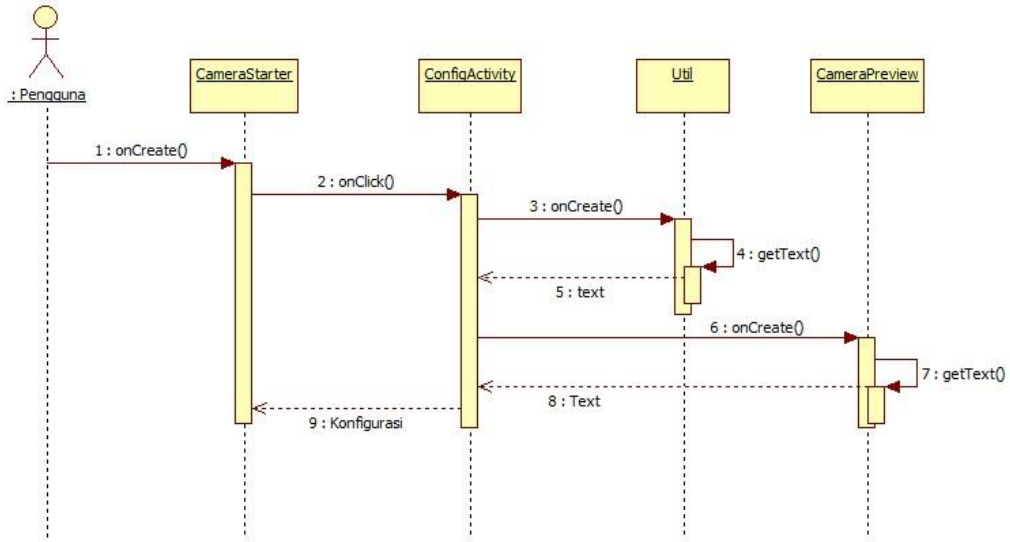
Sequence diagram informasi merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam menampilkan bantuan aplikasi. *Sequence* diagram bantuan terlihat seperti pada Gambar 3.41.



Gambar 3.41 *Sequence* Diagram Menampilkan Bantuan

3.1.7.7.6 *Sequence* Diagram konfigurasi

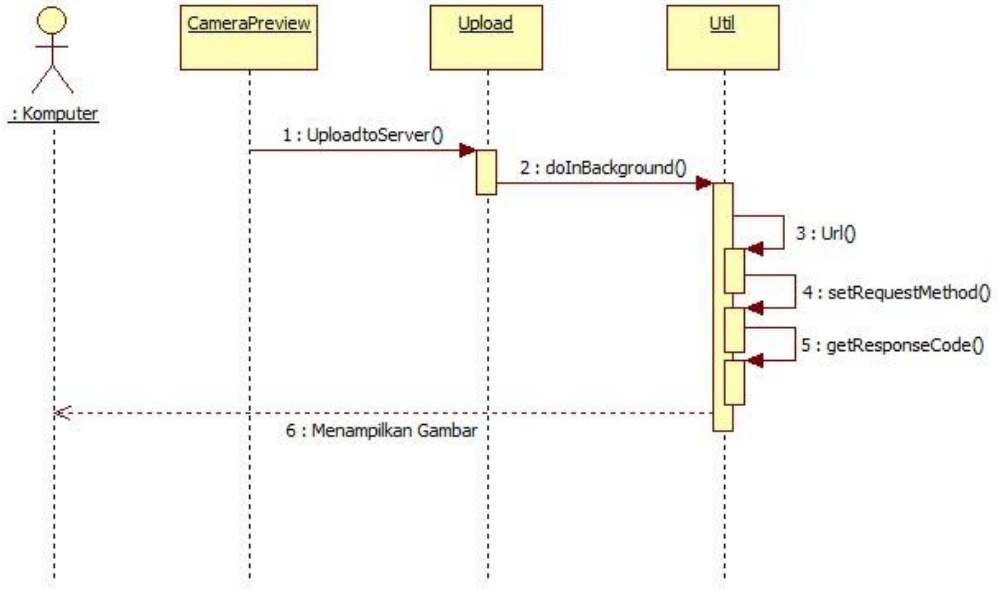
Sequence diagram konfigurasi merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam menampilkan konfigurasi IP address aplikasi. *Sequence* diagram konfigurasi terlihat seperti pada Gambar 3.42.



Gambar 3.42 *Sequence* Diagram Konfigurasi

3.1.7.7.7 *Sequence* Diagram Menampilkan gambar

Sequence diagram menampilkan gambar merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam menyimpan gambar. *Sequence* diagram menampilkan gambar terlihat seperti pada Gambar 3.43.



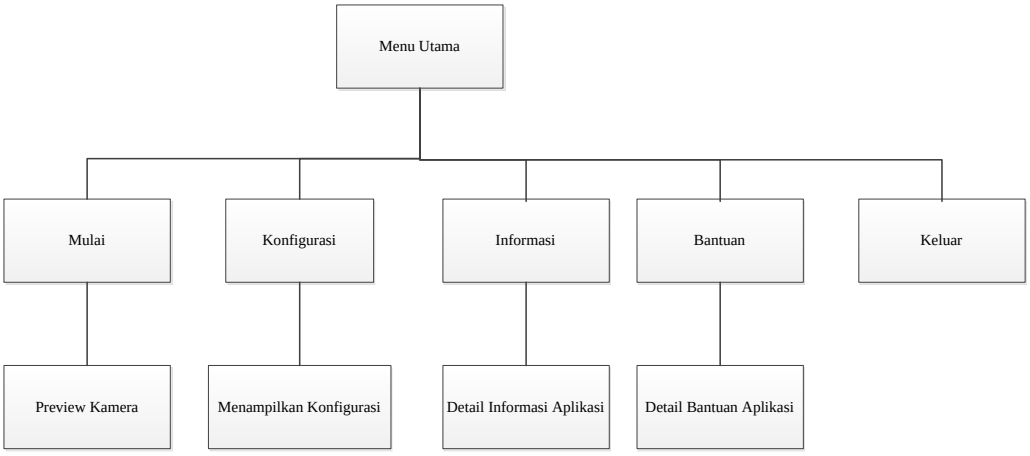
Gambar 3.43 *Sequence* Diagram Menampilkan Gambar

3.2 Perancangan Sistem

Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis.

3.2.1 Perancangan Struktur Menu

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi. Perancangan struktur menu dari aplikasi ini dapat dilihat pada Gambar 3.44.



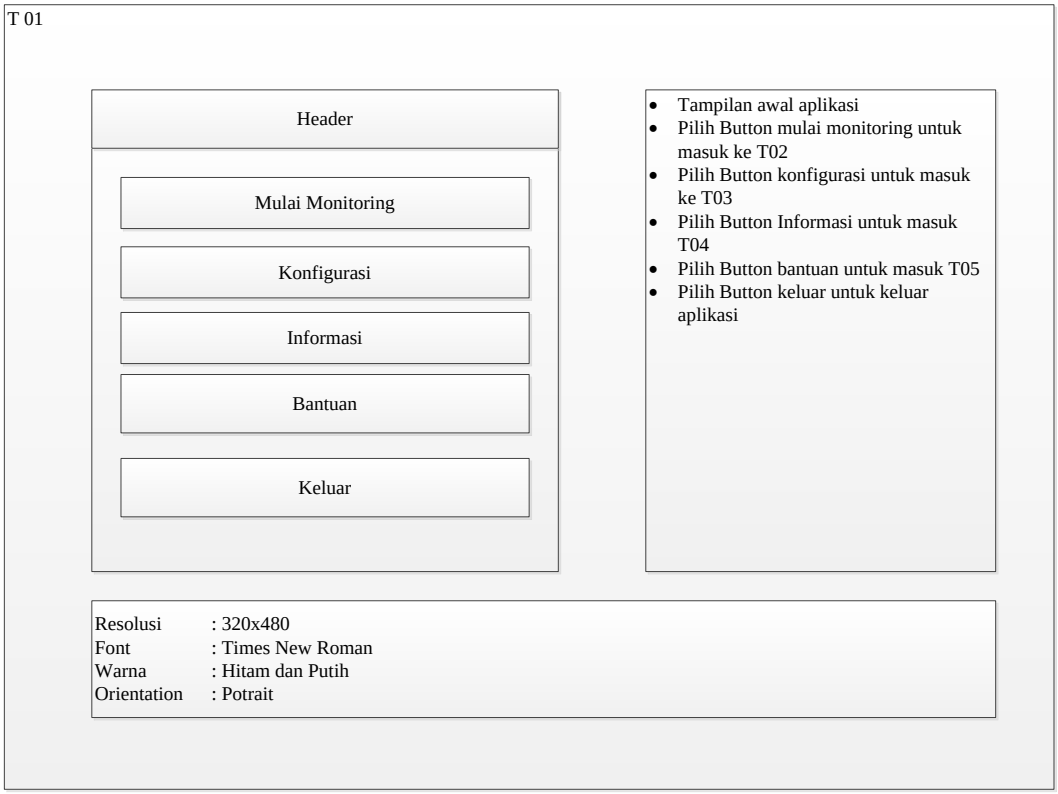
Gambar 3.44 Perancangan Struktur Menu

3.2.2 Perancangan Antarmuka

Tahap perancangan antarmuka sistem bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan faktor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan pada tahap analisis. Dalam tahap ini upaya yang dilakukan yaitu dengan cara mengkombinasikan penggunaan teknologi perangkat keras dan perangkat lunak yang tepat sehingga diperoleh yang optimal dan mudah diimplementasikan.

3.2.2.1 Perancangan antarmuka tampilan awal aplikasi

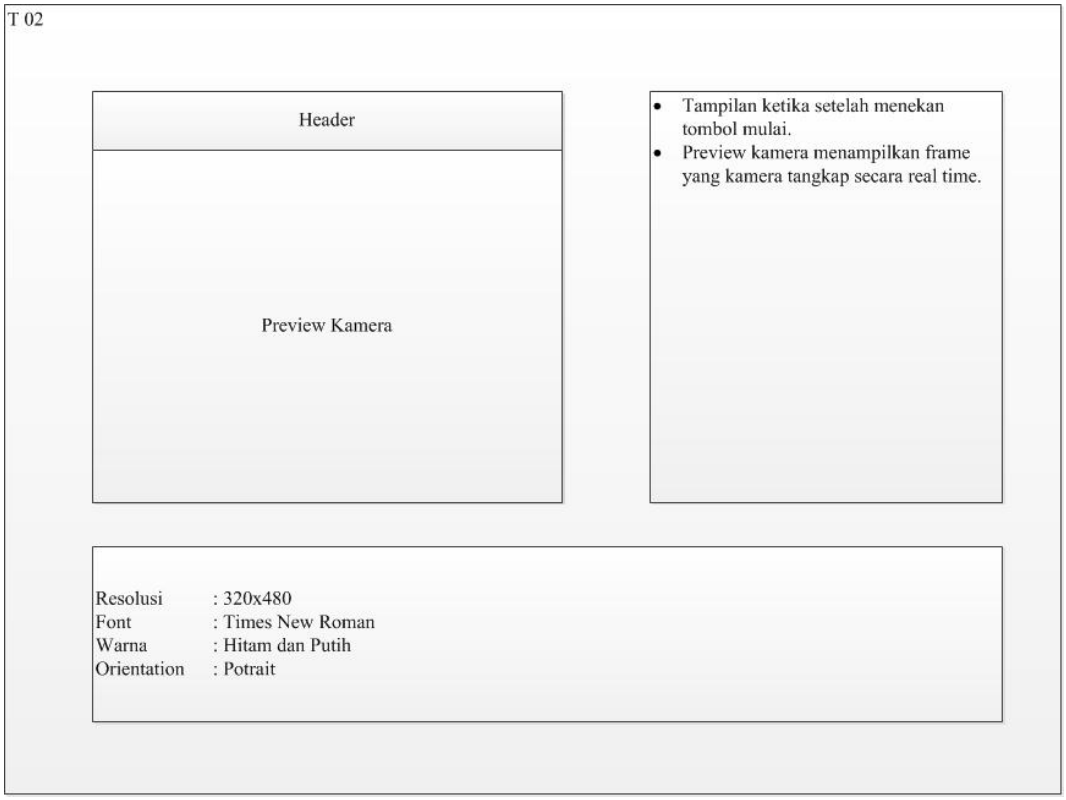
Perancangan antarmuka tampilan awal merupakan tahapan rancangan antarmuka dari awal aplikasi dijalankan. Perancangan antarmuka tampilan awal aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.45.



Gambar 3.45 Tampilan awal aplikasi

3.2.2.2 Perancangan antarmuka *preview* kamera

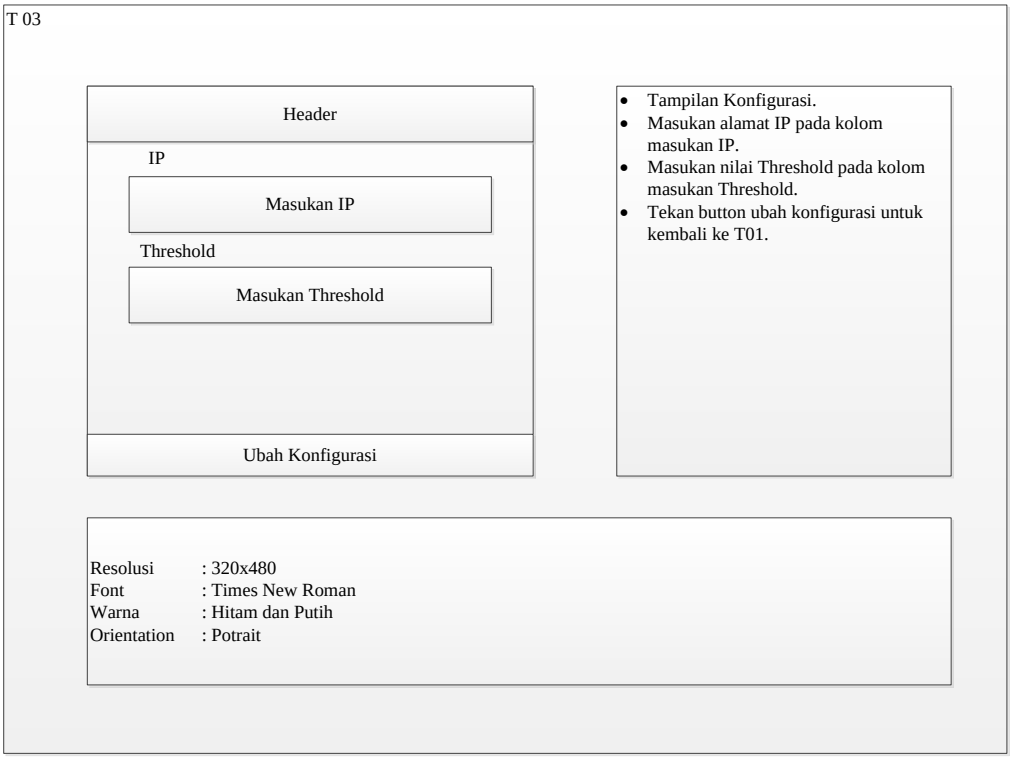
Perancangan antarmuka *preview* merupakan tahapan rancangan antarmuka setelah *user* menekan tombol mulai. Perancangan antarmuka tampilan *preview* kamera aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.46.



Gambar 3.46 Tampilan *Preview* kamera

3.2.2.3 Perancangan antarmuka konfigurasi

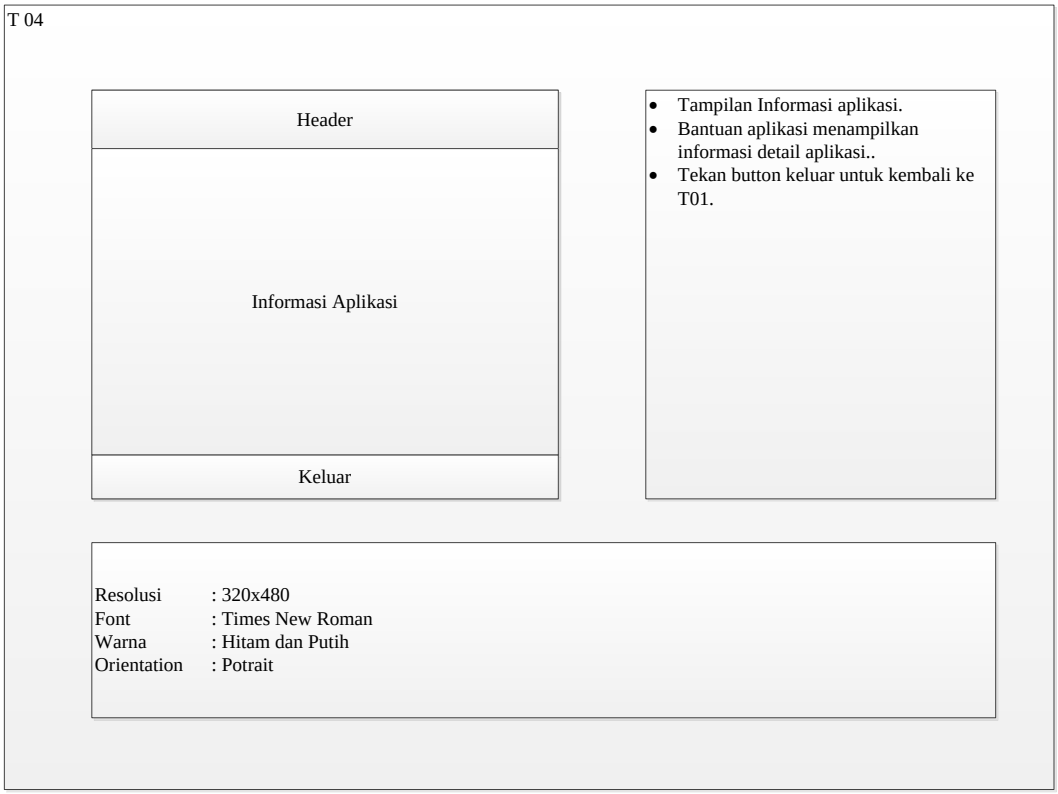
Perancangan antarmuka konfigurasi merupakan tahapan rancangan antarmuka setelah *user* menekan tombol konfigurasi. Perancangan antarmuka tampilan informasi aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.47.



Gambar 3.47 Tampilan Konfigurasi

3.2.2.4 Perancangan antarmuka informasi aplikasi

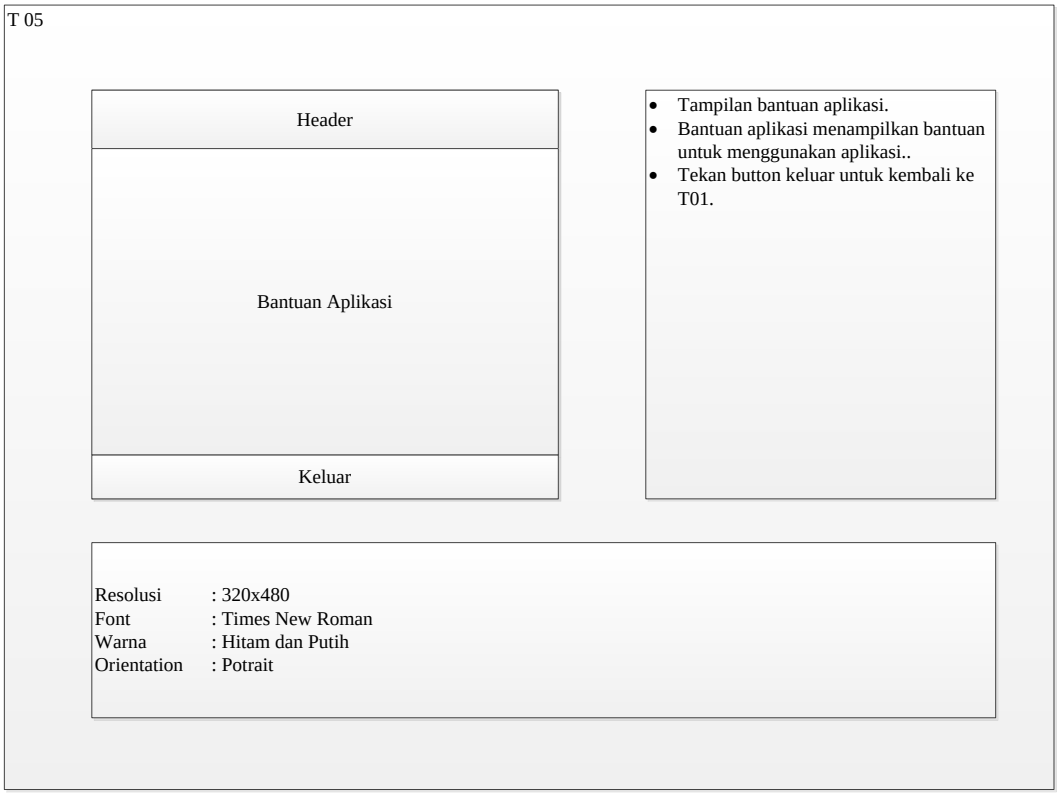
Perancangan antarmuka informasi aplikasi merupakan tahapan rancangan antarmuka setelah *user* menekan tombol informasi. Perancangan antarmuka tampilan informasi aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.48.



Gambar 3.48 Tampilan Informasi Aplikasi

3.2.2.5 Perancangan antarmuka bantuan aplikasi

Perancangan antarmuka bantuan aplikasi merupakan tahapan rancangan antarmuka setelah *user* menekan tombol bantuan. Perancangan antarmuka tampilan bantuan aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.49.



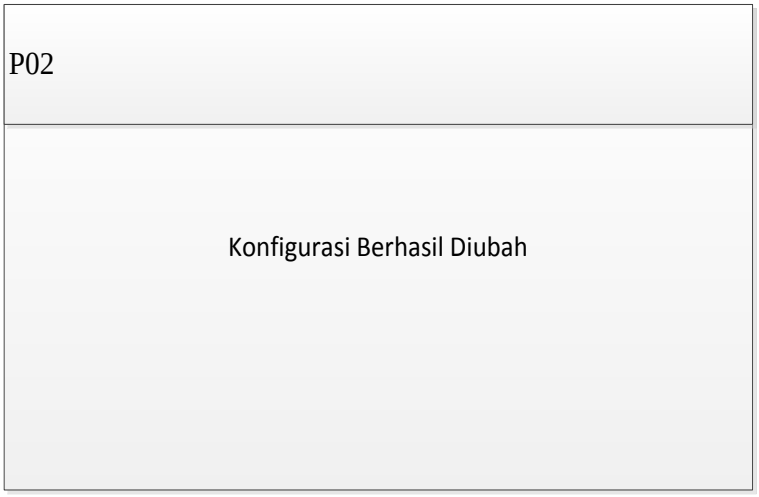
Gambar 3.49 Tampilan Bantuan Aplikasi

3.2.2.6 Perancangan Antarmuka pesan

Perancangan antarmuka pesan merupakan tahapan rancangan antarmuka ketika perangkat *mobile* dan komputer tidak terhubung pada satu jaringan dan tahapan rancangan antarmuka ketika konfigurasi berhasil diubah. Perancangan anarmuka pesan aplikasi *auto capture* berdasarkan pergerakan objek ini terlihat pada Gambar 3.50 dan Gambar 3.51.



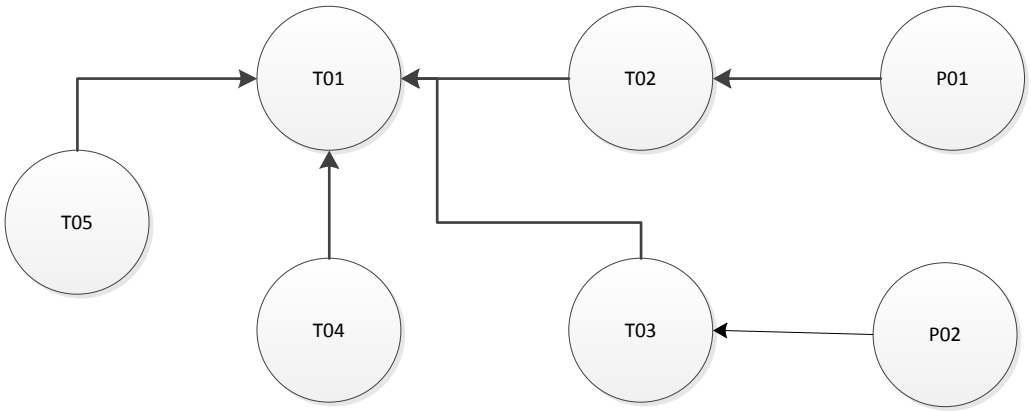
Gambar 3.50 Antarmuka Pesan Salah Konfigurasi



Gambar 3.51 Antarmuka Pesan Konfigurasi Berhasil Diubah

3.2.3 Perancangan Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan alur-alur dari menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi *auto capture* berdasarkan pergerakan objek yang dibangun yaitu seperti pada Gambar 3.52.



Gambar 3.52 Jaringan Semantik Aplikasi *Auto Capture* Berdasarkan Pergerakan Objek