BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan proses penguraian sistem informasi ke dalam komponen-komponen yang sederhana yang memiliki tujuan untuk mengidentifikasi serta mengevaluasi masalah-masalah dan hambatan-hambatan yang mungkin terjadi serta kebutuhan-kebutuhan yang diharapkan menjadi acuan yang dapat mengarah kepada solusi yang terbaik. Pada analisis sistem ini akan dijelaskan beberapa macam analisis yang berhubungan dengan aplikasi yang akan dibangun seperti analisis masalah, analisis data masukkan, analisis proses, analisis metode yang digunakan, analisis kebutuhan fungsional dan non fungsional.

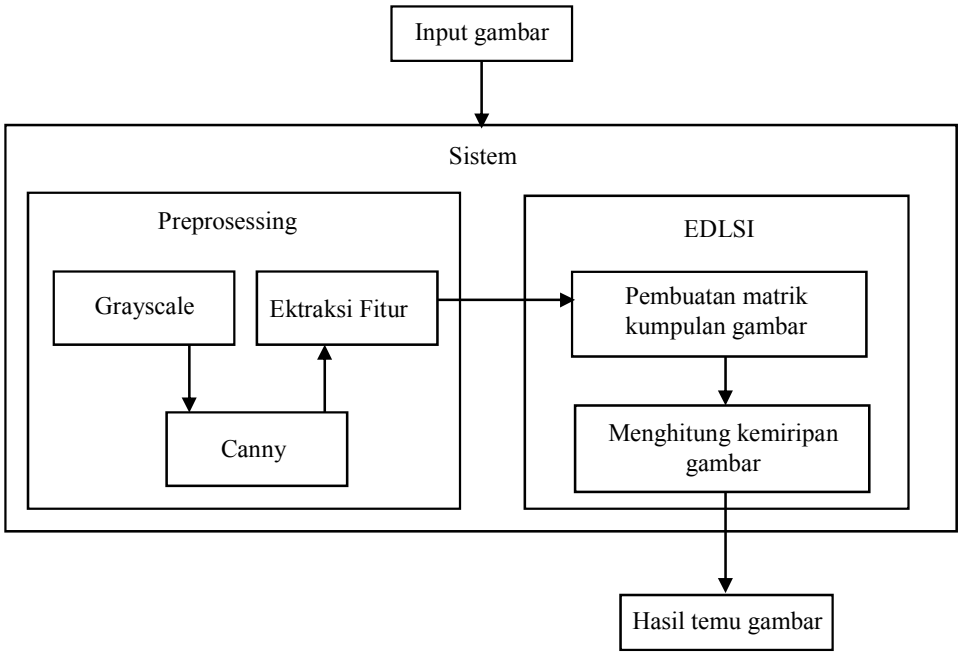
3.1.1 Analisis Masalah

Content Based Image Retrieval (CBIR) atau dikenal sebagai *query* dengan konten *image* dan pengambilan informasi visual berbasis konten merupakan penerapan teknik *Computer Vision* untuk pengambilan gambar atau pencarian gambar digital dalam *database*. *Content based* berarti pencarian akan menganalisa isi sebenarnya dari gambar. Menganalisa dalam konteks dapat berupa warna, bentuk, atau informasi lain yang terdapat pada gambar. Penggunaan metode warna dan bentuk masih memiliki akurasi yang kurang baik sehingga perlu dicari pendekatan lain seperti semantik gambar yang dapat memaksimalkan hasil pencarian. Oleh karena itu dibuat *prototype* CBIR dengan menggunakan semantik gambar

Metode yang diterapkan pada penelitian ini adalah EDLSI yang mana merupakan pengembangan dari LSI. Metode ini dinilai dapat menganalisa kemiripan dari gambar berdasarkan nilai semantik yang ada pada gambar tersebut.

3.1.2 Analisis Sistem Yang Akan Dibangun

Sistem yang akan dibangun adalah sistem pencarian gambar berdasarkan konten bentuk. Sistem ini membandingkan kemiripan bentuk *query* pengguna dengan kumupulan data gambar yang berada di database. Berikut merupakan gambaran umum tentang sistem yang akan dibuat :



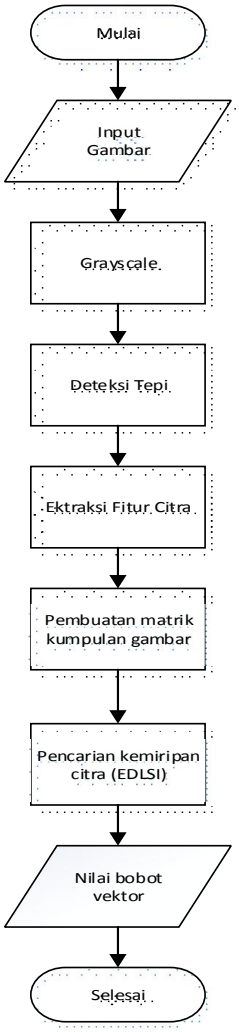
Gambar 3.1 Gambaran Umum Sistem

adalah gambaran umum sistem dan proses – proses yang akan dilakukan untuk pembuatan *prototype.* Proses –proses utama tersebut meliputi *image processing* dan *Essensial Dimension of Latent Semantic Indexing* (EDLSI). Pada proses *image processing*, tahapan pertama yang dilakukan yaitu *grayscale*, grayscale diperlukan untuk proses deteksi tepi, tahap kedua yaitu deteksi tepi *Canny,* pada tahap ini gambar hasil konversi *grayscale* dilakukan pendeteksian tepi guna mempermudah dalam proses pencarian kemiripan gambar dan tahap terakhir yaitu Ektraksi fitur gambar, setelah hasil dari deteksi tepi didapatkan tahap selanjutnya dilakukan pengektraksian gambar untuk nantinya dapat digunakan dalam pembentukan matrik kumpulan gambar, hasil dari ekstraksi fitur gambar tersebut kemudian disimpan di *database*. Kemudian pada proses EDLSI, proses

yang dilakukan meliputi pembuatan matrik kumpulan gambar dan proses perhitungan kemiripan gambar. Pembuatan matrik kumpulan gambar dibuat dari hasil setiap ekstraksi fitur gambar. Ukuran matrik tersebut dinotasikan m x n, di mana m merupakan hasil ekstraksi fitur dari setiap gambar dan n merupakan banyaknya gambar. Proses selanjutnya merupakan proses mencari kemiripan dengan EDLSI, hasil dari proses ini merupakan visualisasi gambar yang dianggap memiliki kemiripan dengan gambar *query*.

3.2 Analisis Proses

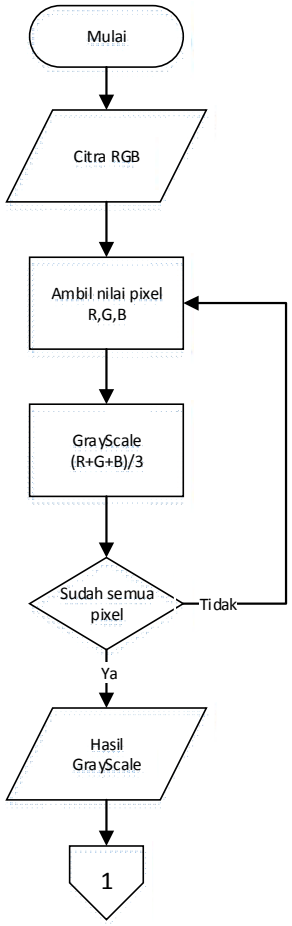
Berikut merupakan *flowchart* dari sistem yang akan dibangun :



Gambar 3.2 *Flowchart* Proses Aplikasi

3.2.1 Gray Scale

Gray Scale merupakan proses konversi gambar normal (RGB) menjeadi gambar abu – abu. Berikut alur dari proses konversi gambar normal ke abu – abu :



Berikut merupakan sampel gambar normal yang akan diolah menjadi abu – abu :



Gambar 3.3 Sampel Gambar Uji

Sampel matrik yang diambil merupakan matrik dengan ukuran 5 x 5. Tahap pertama dalam mengkonversi matrik yaitu mengambil nilai pixel dari *Red, Green, Blue* (RGB). Berikut ini merupakan nilai RGB dari sampel pada Gambar 3.3 :

$$R = \begin{bmatrix} 64 & 63 & 63 & 62 & 62 \\ 64 & 63 & 63 & 62 & 62 \\ 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \end{bmatrix} \quad G = \begin{bmatrix} 42 & 41 & 41 & 40 & 40 \\ 42 & 41 & 41 & 40 & 40 \\ 41 & 41 & 41 & 40 & 40 \\ 41 & 41 & 41 & 40 & 40 \\ 41 & 41 & 41 & 40 & 40 \end{bmatrix} \quad B = \begin{bmatrix} 29 & 28 & 28 & 27 & 27 \\ 29 & 28 & 28 & 27 & 27 \\ 28 & 28 & 28 & 27 & 27 \\ 28 & 28 & 28 & 27 & 27 \\ 28 & 28 & 28 & 27 & 29 \end{bmatrix}$$

Setelah nilai RGB dari tiap pixel diperoleh, proses selanjutnya untuk mengkonversi ke *grayscale* digunakan rumus :

$$grayscale = \frac{(R+G+B)}{3} = \frac{(64+42+29)}{3} = 45 \qquad (2.1)$$

Sehingga didapatkan nilai *grayscale* sebagai berikut.

$$\begin{bmatrix} 45 & 44 & 44 & 43 & 43 \\ 45 & 44 & 44 & 43 & 43 \\ 44 & 44 & 44 & 43 & 43 \\ 44 & 44 & 44 & 43 & 43 \\ 44 & 44 & 44 & 43 & 44 \end{bmatrix}$$

Gambar 3.4 Nilai Hasil Matriks Grayscale

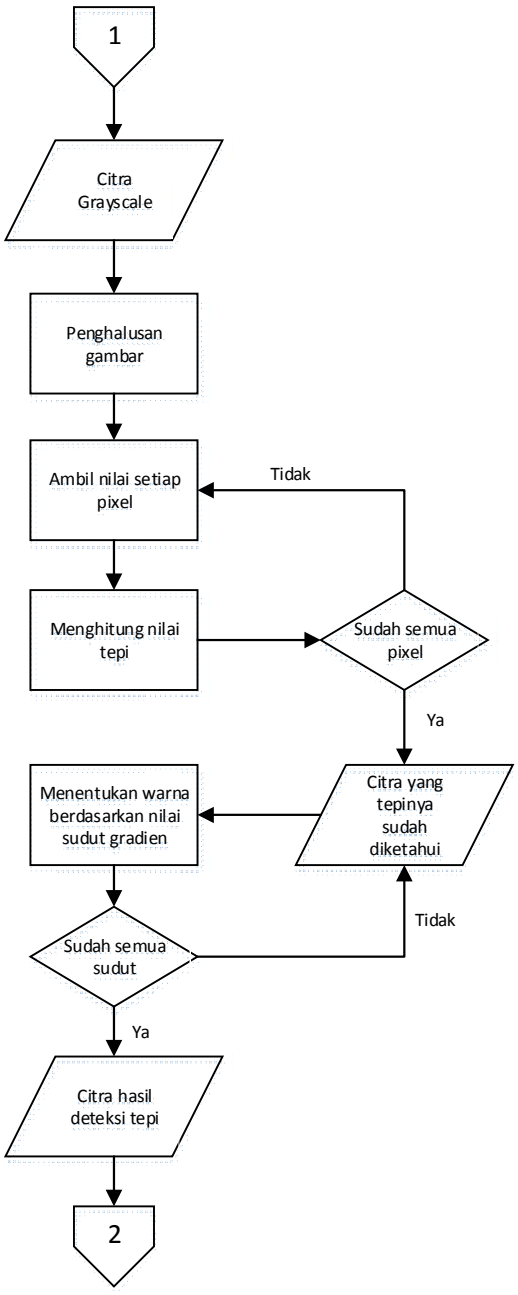
Berikut merupakan hasil konversi gambar RGB pada Gambar 3.3 menjadi gambar grayscale :



Gambar 3.5 Grayscale Sampel Gambar Uji

3.2.2 Deteksi Tepi Canny

Deteksi tepi *Canny* digunakan untuk mendeteksi tepi yang diambil dari gambar yang telah dilakukan proses *grayscale*. Berikut merupakan *flowchart* dari deteksi tepi *Canny* :



Gambar 3.6 Proses Deteksi Tepi *Canny*

Adapun tahapan dari proses deteksi tepi canny adalah sebagai berikut :

1. Penghalusan Gambar (*Smoothing*)

   Proses *smoothing* dilakukan terhadap gambar hasil *grayscale* dengan tujuan untuk menghilangkan noise pada gambar. *Gaussian filter* merupakan metode untuk memperhalus gambar. Pada penelitian ini ukuran filter yang digunakan adalah 5x5 dan standar deviasi σ = 1 [10]. Untuk memperoleh nilai nilai bobotnya di tentukan pada persamaan 2.3. Berikut perhitungan bobot gaussian filter:

A[z]	B[z]	C[z]	D[z]	E[z]
G(-2,-2)	G(-2,-1)	G(0,-2)	G(1,-2)	G(2,-2)
F[z]	G[z]	H[z]	I[z]	J[z]
G(-2,-1)	G(-1,-1)	G(0,-1)	G(1,-1)	G(2,-1)
K[z]	L[z]	M[z]	N[z]	O[z]
G(-2,0)	G(-1,0)	G(0,0)	G(1,0)	G(2,0)
P[z]	Q[z]	R[z]	S[z]	T[z]
G(-2,1)	G(-1,1)	G(0,1)	G(1,1)	G(2,1)
U[z]	V[z]	W[z]	X[z]	Y[z]
G(-2,2)	G(-1,2)	G(0,2)	G(1,2)	G(2,2)

Gambar 3.7 Ilustrasi Koordinat Pada Gaussian Filter

z = merupakan nilai yang akan dicari dengan persamaan 2.3.

$G(x,y) = e-\frac{x^2+y^2}{2\sigma^2}$ (2.3)

G(0,0)=$e-\frac{0^2+0^2}{2\,(1)^2} = e-^0$=1

G(1,0)=G(0,1)=G(-1,0)=G(0,-1)= $e-\frac{1^2+0^2}{2\,(1)^2} = e^{-\frac{1}{2}}$ = 0,6065

G(1,1)=G(1,-1)=G(-1,1)=G(-1,-1)= $e-\frac{1^2+1^2}{2\,(1)^2} = e^{-1}$ = 0,3679

G(2,1)=G(1,2)=G(-2,1)=G(-2,-1)= $e-\frac{2^2+1^2}{2\,(1)^2} = e^{-\frac{5}{2}}$ = 0,0821

G(2,0)=G(0,2)=G(-2,0)=G(0,-2)= $e-\frac{2^2+0^2}{2\,(1)^2} = e^{-2}$=0,1353

G(2,2)=G(-2,-2)=G(-2,2)=G(2,-2)= $e-\frac{2^2+2^2}{2\,(1)^2} = e^{-4}$=0,0183

Setelah semua nilai koordinat ditemukan, maka tahap selanjutnya menjadikan nilai koordinat terkecil (G(2,2) = 0,0183) menjadi 1, maka semua nilai G(x,y) dibagi dengan 0,0183 dan hasilnya dibulatkan ke atas.

$G(0,0) = \frac{1}{0,0183} = 55$

$G(1,0) = G(0,1) = G(-1,0) = G(0,-1) = \frac{0,6065}{0,0183} = 33$

$G(1,1) = G(1,-1) = G(-1,1) = G(-1,-1) = \frac{0,3679}{0,0183} = 20$

$G(2,1) = G(1,2) = G(-2,1) = G(-2,-1) = \frac{0,0821}{0.0183} = 5$

$G(2,0) = G(0,2) = G(-2,0) = G(0,-2) = \frac{0,1353}{0.0183} = 7$

$G(2,2) = G(-2,-2) = G(-2,2) = G(2,-2) = \frac{0,0183}{0,0183} = 1$

Berikut merupakan hasil perhitungan Gaussian filter pada setiap koordinat.

1	5	7	5	1
5	20	33	20	5
7	33	55	33	7
5	20	33	20	5
1	5	7	5	1

Setelah itu dilakukan normalisasi, normalisasi dilakukan dengan cara membagi nilai pada setiap G(x,y) dengan hasil dari penjumlahan semua nilai G(x,y) pada setiap koordinat sehingga diperoleh filter sebagai berikut.

$\frac{1}{339}$	$\frac{5}{339}$	$\frac{7}{339}$	$\frac{5}{339}$	$\frac{1}{339}$
$\frac{5}{339}$	$\frac{20}{339}$	$\frac{33}{339}$	$\frac{20}{339}$	$\frac{5}{339}$
$\frac{7}{339}$	$\frac{33}{339}$	$\frac{55}{339}$	$\frac{33}{339}$	$\frac{7}{339}$
$\frac{5}{339}$	$\frac{20}{339}$	$\frac{33}{339}$	$\frac{20}{339}$	$\frac{5}{339}$
$\frac{1}{339}$	$\frac{5}{339}$	$\frac{7}{339}$	$\frac{5}{339}$	$\frac{1}{339}$

Atau hasil *Gaussian filter* dapat ditulis sebagai berikut.

$$\begin{bmatrix} 0.0029 & 0.0147 & 0.0206 & 0.0147 & 0.0029 \\ 0.0147 & 0.0590 & 0.0973 & 0.0590 & 0.0147 \\ 0.0206 & 0.0973 & 0.1622 & 0.0973 & 0.0206 \\ 0.0147 & 0.0590 & 0.0973 & 0.0590 & 0.0147 \\ 0.0029 & 0.0147 & 0.0206 & 0.0147 & 0.0029 \end{bmatrix}$$

Gambar 3.8 Gaussian Filter σ = 1

Selanjutnya dilakukan konvolusi dengan menggunakan *Gaussian filter* ke *image grayscale* pada Gambar 3.5 dengan ukuran gambar 384 x 256 menggunakan persamaan 2.2.

Berikut sampel nilai pixel dengan ukuran 5x5 yang diperoleh dari grayscale uji gambar dengan menggunakan penghalusan Gaussian filter :

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

Gambar 3.9 Hasil proses Gaussian Filter

2. Menghitung nilai tepi

   Untuk menemukan gradien digunakan operator sobel dengan menggunakan rumus 2.4. Setelah Gx dan Gy diperoleh langkah selanjutnya Gx dan Gy digabungkan dengan menggunakan rumus : G = |Gx|+|Gy|. Berikut ilustrasi perhitungan gradien pada gambar grayscale sampel gambar uji (Gambar 3.9):

$$\begin{bmatrix} \mathbf{44} & \mathbf{43} & \mathbf{43} & \mathbf{43} & \mathbf{43} \\ \mathbf{44} & \mathbf{43} & \mathbf{43} & \mathbf{43} & \mathbf{43} \\ \mathbf{44} & \mathbf{44} & \mathbf{43} & \mathbf{43} & \mathbf{43} \\ \mathbf{44} & \mathbf{44} & \mathbf{44} & \mathbf{44} & \mathbf{43} \\ \mathbf{44} & \mathbf{44} & \mathbf{44} & \mathbf{44} & \mathbf{44} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{44} & \mathbf{43} & \mathbf{43} & 43 & 43 \\ \mathbf{44} & \mathbf{43} & \mathbf{43} & 43 & 43 \\ \mathbf{44} & \mathbf{44} & \mathbf{43} & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

Gambar 3.10 Proses Sobel

Nilai Gx diperoleh dengan melakukan konvolusi pada Gambar 3.9 dengan menggunakan *filter* Mx = $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, sedangkan untuk nilai Gy diperoleh dengan melakukan konvolusi menggunakan filter My = $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$, proses konvolusi dilakukan dengan menggunakan persamaan 2.2. Berikut merupakan hasil perhitungan nilai *gradient*.

$h(x,y) = f(x,y) * g(x,y) = \sum_{a=\infty}^{\infty} \sum_{a=\infty}^{\infty} f(a,b).g(x-a,y-b)$ (2.2)

1. Gx = (-1x44) + (0x43) + (1x43) + (-2x44) + (0x43) + (2x43) + (-1x44) + (0x44) + (1x43) = 4

   Gy = (1x44) + (2x43) + (1x43) + (0x44) + (0x43) + (0x43) + (-1x44) + (-2x44) + (1x43) = 2

   |G| = |Gx|+|Gy| → |G|=4+2= 6

   maka nilai 43 diubah menjadi 6

   $\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$

2. Gx = (-1x43) + (0x43) + (1x43) + (-2x43) + (0x43) + (2x43) + (-1x44) + (0x43) + (1x43) = -1

   Gy = (1x43) + (2x43) + (1x43) + (0x43) + (0x43) + (0x43) + (-1x44) + (-2x43) + (1x43) = -1

   |G| = |Gx|+|Gy| → |G|=1+1= 2

   maka nilai 43 diubah menjadi 2

   $\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$

3. Gx = (-1x43) + (0x43) + (1x43) + (-2x43) + (0x43) + (2x43) + (-1x43) + (0x43) + (1x43) = 0

Gy = (1x43) + (2x43) + (1x43) + (0x43) + (0x43) + (0x43) + (-1x43) + (-2x43) + (1x43) = 0

|G| = |Gx|+|Gy| → |G|= 0+0= 0

maka nilai 43 diubah menjadi 0

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

4. Gx = (-1x44) + (0x44) + (1x43) + (-2x44) + (0x44) + (2x43) + (-1x44) + (0x44) + (1x44) = -3

   Gy = (1x44) + (2x43) + (1x43) + (0x44) + (0x44) + (0x43) + (-1x44) + (-2x44) + (-1x44) = -3

   |G| = |Gx|+|Gy| → |G|= 3+3= 6

   maka nilai 44 diubah menjadi 4

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 4 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

5. Gx = (-1x44) + (0x43) + (1x43) + (-2x44) + (0x43) + (2x43) + (-1x44) + (0x44) + (1x44) = -3

   Gy = (1x44) + (2x43) + (1x43) + (0x44) + (0x43) + (0x43) + (-1x44) + (-2x44) + (-1x44) = -3

   |G| = |Gx|+|Gy| → |G|= 3+3=6

   maka nilai 44 diubah menjadi 6

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 3 & 6 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

6. Gx = (-1x43) + (0x43) + (1x43) + (-2x43) + (0x43) + (2x43) + (-1x44) + (0x44) + (1x43) = -1

   Gy = (1x43) + (2x43) + (1x43) + (0x43) + (0x43) + (0x43) + (-1x44) + (-2x44) + (-1x43) = -3

|G| = |Gx|+|Gy| → |G|= 1+3 = 4

maka nilai 43 diubah menjadi 4

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 3 & 6 & 4 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

7. Gx = (-1x44) + (0x44) + (1x43) + (-2x44) + (0x44) + (2x44) + (-1x44) + (0x44) + (1x44) = -1

   Gy = (1x44) + (2x44) + (1x43) + (0x44) + (0x44) + (0x44) + (-1x44) + (-2x44) + (-1x44) = -1

   |G| = |Gx|+|Gy| → |G|= 1+1 = 2

   maka nilai 44 diubah menjadi 2

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 3 & 6 & 4 & 43 \\ 44 & 2 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

8. Gx = (-1x44) + (0x43) + (1x43) + (-2x44) + (0x44) + (2x44) + (-1x44) + (0x44) + (1x44) = -1

   Gy = (1x44) + (2x43) + (1x43) + (0x44) + (0x44) + (0x44) + (-1x44) + (-2x44) + (-1x44) = -3

   |G| = |Gx|+|Gy| → |G|= 1+3 = 4

   maka nilai 44 diubah menjadi 4

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 3 & 6 & 4 & 43 \\ 44 & 2 & 4 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

9. Gx = (-1x43) + (0x43) + (1x43) + (-2x44) + (0x44) + (2x43) + (-1x44) + (0x44) + (1x44) = -2

   Gy = (1x43) + (2x43) + (1x43) + (0x44) + (0x44) + (0x43) + (-1x44) + (-2x44) + (-1x44) = -4

   |G| = |Gx|+|Gy| → |G|= 2+4 = 6

   maka nilai 44 diubah menjadi 4

$$\begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 43 & 43 & 43 & 43 \\ 44 & 44 & 43 & 43 & 43 \\ 44 & 44 & 44 & 44 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix} \rightarrow \begin{bmatrix} 44 & 43 & 43 & 43 & 43 \\ 44 & 6 & 2 & 0 & 43 \\ 44 & 3 & 6 & 4 & 43 \\ 44 & 2 & 4 & 6 & 43 \\ 44 & 44 & 44 & 44 & 44 \end{bmatrix}$$

nilai tepi tidak dilakukan konvolusi sehingga nilai tepian diabaikan (diberi tanda *) . Setelah dilakukan proses sobel, maka didapatkan nilai pixel baru sebagai berikut :

*	*	*	*	*
*	6	2	0	*
*	6	6	4	*
*	2	4	6	*
*	*	*	*	*

Gambar 3.11 Hasil Dari Proses Sobel

Sebelumnya matrik mempunyai ukuran 5x5, setelah melalui proses konvolusi dengan operator sobel maka ukuran matrik menjadi 3x3. Pada proses sobel nilai tepi matrik sebelumnya tidak digunakan dikarenakan tidak mempengaruhi hasil dari proses selanjutnya.

Setelah menemukan nilai tepi sobel, maka langkah selanjutnya adalah menentukan arah tepi dengan menggunakan persamaan 2.6. Berikut ditunjukan sample perhitungan menentukan arah tepi :

$$\boldsymbol{\theta = \arctan\left(\frac{|Gy|}{|Gx|}\right)}$$

1. $\theta = \left(\left(\tan^{-1}\left(\frac{2}{4}\right)\right) * 180\right)/3.14 = 26{,}57°$
2. $\theta = \left(\left(\tan^{-1}\left(\frac{1}{1}\right)\right) * 180\right)/3.14 = 45°$
3. $\theta = \left(\left(\tan^{-1}\left(\frac{0}{0}\right)\right) * 180\right)/3.14 = \infty°$
4. $\theta = \left(\left(\tan^{-1}\left(\frac{3}{3}\right)\right) * 180\right)/3.14 = 45°$

5. $\theta = (\left(\tan^{-1}\left(\frac{3}{3}\right)\right) * 180)/3.14 = 45°$
6. $\theta = (\left(\tan^{-1}\left(\frac{3}{1}\right)\right) * 180)/3.14 = 71{,}56°$
7. $\theta = (\left(\tan^{-1}\left(\frac{1}{1}\right)\right) * 180)/3.14 = 45°$
8. $\theta = (\left(\tan^{-1}\left(\frac{3}{1}\right)\right) * 180)/3.14 = 71{,}56°$
9. $\theta = (\left(\tan^{-1}\left(\frac{4}{2}\right)\right) * 180)/3.14 = 63{,}43°$

sehingga hasil arah tepi(θ) ditampilkan dalam tabel berikut :

*	*	*	*	*
*	6	2	0	*
*	6	6	4	*
*	2	4	6	*
*	*	*	*	*

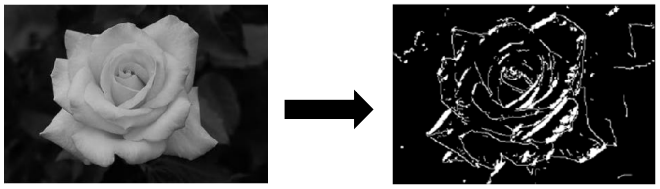
→

*	*	*	*	*
*	26,57°	45°	∞°	*
*	45°	45°	71,56°	*
*	45°	71,56°	63,43°	*
*	*	*	*	*

Gambar 3.12 Hasil Penentuan Arah Tepi

Setelah mengetahui arah tepi langkah selanjutnya menghubungkan arah tepi ke arah yang dapat ditelusuri dalam gambar berdasarkan 4 warna untuk mengetahui arah suatu tepi sehingga apabila garis yang arahnya berbeda memiliki warna yang berbeda [11].

Berikut ini merupakan hasil dari deteksi tepi menggunakan canny :



Gambar 3.13 Sampel Gambar Hasil Deteksi Canny

3.2.3 Ekstraksi

Ekstraksi atau *feature extraction* merupakan pengambilan nilai pixel semua gambar dari hasil proses *image processing*. Berikut merupakan sampel gambar yang diambil :



Gambar 3.14 Sampel Ekstraksi

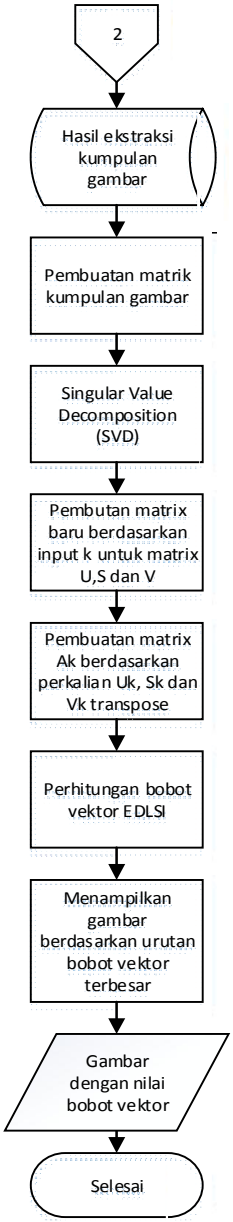
Gambar 3.13 merupakan sampel gambar hasil dari deteksi tepi *canny*. Pada sampel tersebut diambil matrik berukuran 5x5. Berikut merupakan sampel matrik yang diambil dari Gambar 3.14.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Gambar 3.15 Sampel Matriks 5x5

3.2.4 Proses EDLSI

Proses EDLSI digunakan untuk melihat kemiripan antar gambar berdasarkan pada nilai semantik gambar.



Gambar 3.16 Proses Perhitungan EDLSI

Adapun tahapan yang dilakukan adalah sebagai berikut :

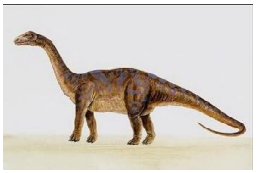
1. Pembuatan Kumpulan Matrik Gambar

Pada tahap ini, nilai pixel gambar hasil proses ekstraksi dijadikan matrik. Matrik dari setiap gambar berukuran m x 1, di mana m merupakan baris matrik yang berisikan nilai dari ekstraksi gambar. Setiap matrik gambar disatukan

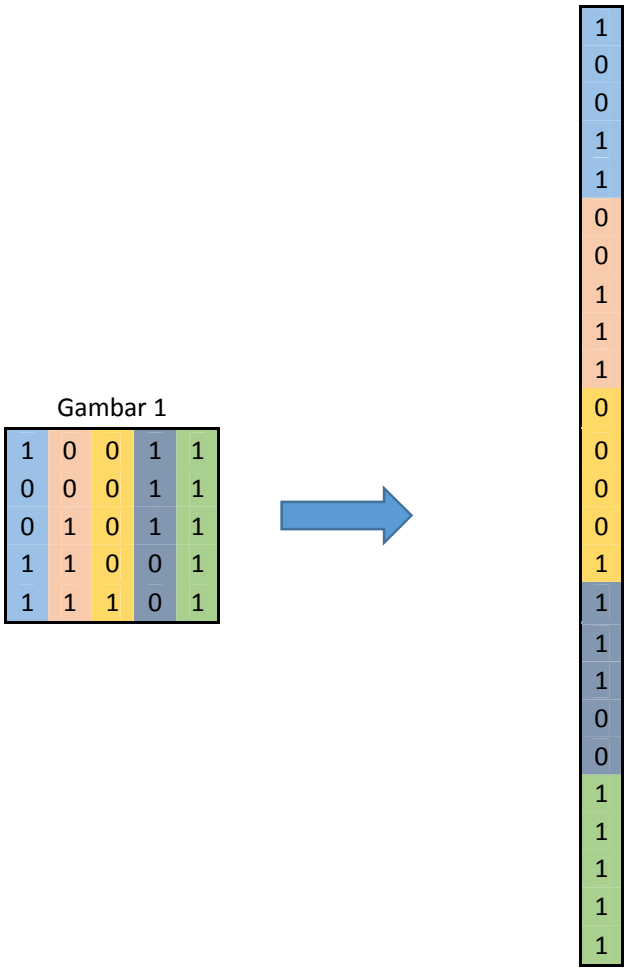
sehingga memperoleh satu matrik berukuran m x n, dimana m merupakan baris matriks berisikan nilai ekstraksi gambar dan n merupakan kolom dari kumpulan gambar.

Berikut merupakan contoh gambar dan nilai pixel yang akan dilakukan pembuatan kumpulan matrik gambar dengan ukuran gambar 100 x 100 pixel. pada simulasi ini matriks yang digunakan berukuran 5x5 .

Tabel 3-1 Sampel Gambar Uji

No	Gambar	Nama Gambar	Matriks gambar
1		Gambar 1.jpg	$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$
2		Gambar 2.jpg	$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$
3		Gambar 3.jpg	$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$
4		Gambar 4.jpg	$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$
5		Gambar 5.jpg	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
6		Gambar 6.jpg	$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$

Berikut merupakan sampel hasil dari gambar 1 yang ukurannya akan diubah menjadi m x 1.



Gambar 3.17 Sampel Perubahan Ukuran Matrik Gambar

Berikut merupkan hasil dari penggabungan setiap gambar yang dibentuk dalam satu matrik :

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

2. Pembentukan Singular Value Decomposition (SVD)

Pembentukan dilakukan dengan menggunakan persamaan 2.8. untuk mendapatkan matriks U,S,V^T dilakukan tahapan sebagai berikut :

a. Matrik U

Untuk mendapatkan matrik U dilakukan perhitungan untuk mendapatkan nilai eigen dan vektor eigen dengan menggunakan persamaan 2.9.(detail matrik dapat dilihat pada lampiran B)

$$A^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 3 & 2 & 2 & 1 & 1 & 1 & 0 & 1 & 3 & 3 & 2 & 0 & 0 & 2 & 1 & 2 & 2 & 3 & 1 & 0 & 2 & 3 & 3 & 3 & 2 \\ 2 & 3 & 2 & 0 & 1 & 1 & 1 & 1 & 3 & 3 & 2 & 0 & 1 & 2 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 3 & 3 & 3 & 2 \\ 2 & 2 & 3 & 0 & 1 & 2 & 0 & 1 & 2 & 3 & 3 & 0 & 0 & 3 & 1 & 1 & 2 & 2 & 2 & 0 & 2 & 3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 3 & 1 & 1 & 3 & 2 & 3 & 1 & 0 & 1 & 1 & 3 & 1 & 3 & 1 & 2 & 1 & 2 & 3 & 3 & 3 & 2 \\ 1 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 3 & 1 & 2 & 4 & 2 & 3 & 2 & 1 & 1 & 1 & 4 & 2 & 3 & 1 & 3 & 2 & 3 & 4 & 4 & 3 & 2 \\ 3 & 3 & 2 & 1 & 2 & 1 & 1 & 2 & 4 & 4 & 2 & 0 & 1 & 2 & 2 & 2 & 3 & 3 & 2 & 1 & 2 & 4 & 4 & 4 & 3 \\ 3 & 3 & 3 & 1 & 3 & 2 & 1 & 3 & 4 & 5 & 3 & 0 & 1 & 3 & 3 & 2 & 4 & 3 & 3 & 1 & 3 & 5 & 5 & 5 & 3 \\ 2 & 2 & 3 & 0 & 1 & 2 & 1 & 2 & 2 & 3 & 4 & 1 & 0 & 3 & 2 & 2 & 2 & 2 & 3 & 1 & 3 & 4 & 4 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 2 & 2 & 3 & 0 & 1 & 2 & 0 & 1 & 2 & 3 & 3 & 0 & 0 & 3 & 1 & 1 & 2 & 2 & 2 & 0 & 2 & 3 & 3 & 3 & 1 \\ 1 & 1 & 1 & 1 & 3 & 1 & 2 & 4 & 2 & 3 & 2 & 1 & 1 & 1 & 4 & 2 & 3 & 1 & 3 & 2 & 3 & 4 & 4 & 3 & 2 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 1 & 0 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 2 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 & 3 & 1 & 1 & 3 & 3 & 4 & 2 & 0 & 1 & 2 & 3 & 1 & 4 & 2 & 2 & 1 & 3 & 4 & 4 & 4 & 3 \\ 3 & 2 & 2 & 1 & 1 & 1 & 0 & 1 & 3 & 3 & 2 & 0 & 0 & 2 & 1 & 2 & 2 & 3 & 1 & 0 & 2 & 3 & 3 & 3 & 2 \\ 1 & 2 & 2 & 0 & 2 & 2 & 2 & 3 & 2 & 3 & 3 & 1 & 1 & 2 & 3 & 2 & 2 & 1 & 4 & 2 & 2 & 4 & 4 & 3 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 1 & 3 & 2 & 3 & 3 & 1 & 0 & 2 & 3 & 2 & 3 & 2 & 2 & 1 & 4 & 4 & 4 & 3 & 2 \\ 3 & 3 & 3 & 1 & 3 & 2 & 2 & 4 & 4 & 5 & 4 & 1 & 1 & 3 & 4 & 3 & 4 & 3 & 4 & 2 & 4 & 6 & 6 & 5 & 3 \\ 3 & 3 & 3 & 1 & 3 & 2 & 2 & 4 & 4 & 5 & 4 & 1 & 1 & 3 & 4 & 3 & 4 & 3 & 4 & 2 & 4 & 6 & 6 & 5 & 3 \\ 3 & 3 & 3 & 1 & 3 & 2 & 1 & 3 & 4 & 5 & 3 & 0 & 1 & 3 & 3 & 2 & 4 & 3 & 3 & 1 & 3 & 5 & 5 & 5 & 3 \\ 2 & 2 & 1 & 1 & 2 & 0 & 1 & 2 & 3 & 3 & 1 & 0 & 1 & 1 & 2 & 1 & 3 & 2 & 1 & 1 & 2 & 3 & 3 & 3 & 3 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix}
\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda
\end{bmatrix}$$

Dari hasil perhitungan dengan menggunakan persamaan 2.9 diperoleh nilai eigen yaitu $\lambda_1 = 54.31$, $\lambda_2 = 10.38$, $\lambda_3 = 7.09$, $\lambda_4 = 4.62$, $\lambda_5 = 4.27$, $\lambda_6 = 2.32$. selanjutnya dengan menggunakan persamaan 2.10 diperoleh vektor eigen dari matrik U adalah sebagai berikut (detail matriks U dapat dilihat pada lampiran B):

$$U = \begin{bmatrix} -0.17 & -0.28 & -0.12 & 0.23 & -0.20 & & -0.17 & -0.17 & -0.18 & -0.27 \\ -0.17 & -0.17 & -0.5 & -0.38 & -0.31 & & -0.23 & -0.23 & -0.03 & -0.01 \\ -0.17 & -0.28 & 0.23 & -0.20 & 0.17 & & -0.06 & -0.06 & -0.20 & -0.04 \\ -0.06 & 0.02 & -0.21 & 0.29 & 0.03 & & -0.11 & -0.11 & -0.06 & 0.21 \\ -0.18 & 0.18 & -0.21 & -0.06 & 0.34 & & -0.03 & -0.03 & -0.25 & -0.22 \\ -0.11 & -0.13 & 0.27 & -0.10 & 0.22 & & -0.01 & -0.01 & -0.10 & 0.38 \\ -0.10 & 0.31 & 0.06 & -0.15 & -0.27 & & -0.06 & -0.06 & -0.01 & -0.07 \\ -0.22 & 0.36 & -0.01 & 0.10 & 0.15 & & -0.11 & -0.11 & -0.01 & 0.001 \\ -0.23 & -0.14 & -0.26 & -0.09 & -0.28 & & 0.02 & 0.02 & -0.01 & -0.14 \\ -0.29 & -0.12 & -0.12 & -0.13 & 0.11 & & -0.03 & -0.03 & -0.11 & -0.05 \\ -0.21 & -0.11 & -0.42 & 0.06 & -0.02 & & -0.10 & -0.10 & 0.01 & 0.06 \\ -0.04 & 0.17 & 0.20 & 0.16 & -0.19 & & -0.07 & -0.07 & 0.06 & 0.01 \\ -0.06 & 0.14 & -0.14 & -0.31 & 0.08 & ... & 0.004 & 0.004 & -0.07 & -0.08 \\ -0.17 & -0.28 & 0.23 & -0.10 & 0.17 & & -0.04 & -0.04 & -0.05 & 0.05 \\ -0.22 & 0.36 & -0.01 & 0.10 & 0.15 & & -0.11 & -0.11 & -0.01 & 0.001 \\ -0.15 & 0.04 & 0.12 & 0.39 & -0.34 & & -0.06 & -0.06 & 0.05 & 0.08 \\ -0.23 & 0.03 & -0.26 & -0.06 & 0.29 & & -0.05 & -0.05 & -0.08 & -0.14 \\ -0.17 & -0.28 & -0.12 & 0.23 & -0.20 & & 0.01 & 0.01 & -0.03 & 0.06 \\ -0.21 & 0.18 & 0.33 & -0.25 & -0.06 & & -0.10 & -0.10 & -0.05 & 0.12 \\ -0.10 & 0.31 & 0.06 & -0.15 & -0.27 & & -0.06 & -0.06 & -0.01 & -0.07 \\ -0.22 & 0.07 & 0.08 & 0.42 & 0.19 & & -0.12 & -0.12 & 0.05 & -0.05 \\ -0.33 & 0.05 & 0.08 & 0.04 & -0.07 & & 0.90 & -0.10 & -0.05 & -0.04 \\ -0.33 & 0.05 & 0.08 & 0.04 & -0.07 & & -0.10 & 0.90 & -0.05 & -0.04 \\ -0.29 & -0.12 & -0.12 & -0.13 & 0.11 & & -0.03 & -0.03 & 0.89 & -0.05 \\ -0.18 & 0.01 & -0.39 & -0.02 & -0.10 & & 0.003 & -0.003 & -0.07 & 0.77 \end{bmatrix}$$

b. Matriks V

Untuk mendapatkan matriks V, dilakukan perhitungan untuk mendapatkan vektor eigen dengan menggunakan persamaan 2.11. Langkah selanjutnya sama seperti pembuatan matrik U. setelah diperoleh matriks V langkah selanjutnya matriks V tersebut ditranspose. Berikut ini merupakan hasil matrik V dan V^T yang diperoleh :

$$A^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 15 & 6 & 10 & 9 & 10 & 8 \\ 6 & 11 & 4 & 7 & 7 & 5 \\ 10 & 4 & 14 & 9 & 8 & 11 \\ 9 & 7 & 9 & 14 & 9 & 9 \\ 10 & 7 & 8 & 9 & 15 & 7 \\ 8 & 5 & 11 & 9 & 7 & 14 \end{bmatrix} \qquad \lambda I = \begin{bmatrix} \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda \end{bmatrix}$$

Dari hasil perhitungan persamaan 2.10 diperoleh nilai eigen yaitu λ_1 = 54.31, λ_2 = 10.38, λ_3= 7.09, λ_4 = 4.62, λ_5 = 4.27, λ_6 = 2.32. Selanjutnya dengan menggunakan persamaan 2.12 diperoleh vektor eigen dari matrik V adalah sebagai berikut :

$$V = \begin{bmatrix} -0.444 & 0.068 & -0.560 & 0.626 & 0.064 & 0.299 \\ -0.287 & 0.559 & 0.521 & 0.354 & -0.391 & -0.237 \\ -0.431 & -0.488 & -0.125 & 0.000 & -0.098 & -0.742 \\ -0.432 & 0.076 & 0.365 & -0.086 & 0.816 & 0.031 \\ -0.424 & 0.445 & -0.365 & -0.676 & -0.175 & 0.038 \\ -0.411 & -0.491 & 0.365 & -0.136 & -0.370 & 0.549 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.444 & -0.287 & -0.431 & -0.432 & -0.424 & -0.411 \\ 0.068 & 0.559 & -0.488 & 0.076 & 0.445 & -0.491 \\ -0.560 & 0.521 & -0.125 & 0.365 & -0.365 & 0.365 \\ 0.626 & 0.354 & 0.000 & -0.086 & -0.676 & -0.136 \\ 0.064 & -0.391 & -0.098 & 0.816 & -0.175 & -0.370 \\ 0.299 & -0.237 & -0.742 & 0.031 & 0.038 & 0.549 \end{bmatrix}$$

c. Matriks S

Untuk mendapatkan matriks S didapatkan dari diagonal akar dari nilai eigen yang sudah diurutkan dari yang terbesar dan bukan nol.

$$S = \begin{bmatrix} 7.370 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.222 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.663 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.065 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.524 \end{bmatrix}$$

3. Pembentukan Matrik Baru Berdasarkan K

Pembentukan matrik baru dilakukan dengan mereduksi baris, kolom maupun baris dan kolom. Pembentukan tersebut ditentukan sesuai dengan k, pada penelitian ini nilai k = 2. Pada matrik U kolom diubah sesuai dengan k sehingga matrik U hanya memiliki 2 kolom, pada matrik S baris dan kolom diubah sesuai k sehingga mempunyai 2 baris dan 2 kolom sedangkan pada matrik V^T baris di ubah sesuai k sehingga mempunyai 2 baris. Berikut hasil dari pembentukan matriks baru :

$$U_2 = \begin{bmatrix} -0.17 & -0.28 \\ -0.17 & -0.17 \\ -0.17 & -0.28 \\ -0.06 & 0.02 \\ -0.18 & 0.18 \\ -0.11 & -0.13 \\ -0.10 & 0.31 \\ -0.22 & 0.36 \\ -0.23 & -0.14 \\ -0.29 & -0.12 \\ -0.21 & -0.11 \\ -0.04 & 0.17 \\ -0.06 & 0.14 \\ -0.17 & -0.28 \\ -0.22 & 0.36 \\ -0.15 & 0.04 \\ -0.23 & 0.03 \\ -0.17 & -0.28 \\ -0.21 & 0.18 \\ -0.10 & 0.31 \\ -0.22 & 0.07 \\ -0.33 & 0.05 \\ -0.33 & 0.05 \\ -0.29 & -0.12 \\ -0.18 & 0.01 \end{bmatrix} \quad S_2 = \begin{bmatrix} 7.370 & 0 \\ 0 & 3.222 \end{bmatrix}$$

$$V^T_2 = \begin{bmatrix} -0.444 & -0.287 & -0.431 & -0.432 & -0.424 & -0.411 \\ 0.068 & 0.559 & -0.488 & 0.076 & 0.445 & -0.491 \end{bmatrix}$$

4. Pembentukan Matrik A_k

Pembentukan matrik Ak dilakukan untuk menghitung kemiripan gambar yang dibutuhkan untuk persamaan 2.8. Untuk mendapatkan matrik A_k dilakukan perkalian matrik U_2,S_2 dan V^T_2 . Berikut hasil perkalian yang membentuk nilai dari matrik A_k :

$$A_k = \begin{bmatrix} 0.508 & -0.140 & 0.999 & 0.486 & 0.140 & 0.976 \\ 0.525 & 0.065 & 0.806 & 0.506 & 0.299 & 0.783 \\ 0.503 & -0.139 & 0.990 & 0.481 & 0.138 & 0.967 \\ 0.202 & 0.166 & 0.158 & 0.197 & 0.218 & 0.149 \\ 0.617 & 0.702 & 0.272 & 0.606 & 0.813 & 0.245 \\ 0.346 & 0.010 & 0.566 & 0.332 & 0.173 & 0.550 \\ 0.384 & 0.766 & -0.183 & 0.383 & 0.748 & -0.201 \\ 0.782 & 1.097 & 0.123 & 0.772 & 1.183 & 0.088 \\ 0.727 & 0.231 & 0.964 & 0.702 & 0.517 & 0.932 \\ 0.924 & 0.397 & 1.113 & 0.894 & 0.734 & 1.072 \\ 0.669 & 0.256 & 0.841 & 0.647 & 0.508 & 0.811 \\ 0.166 & 0.395 & -0.149 & 0.167 & 0.370 & -0.156 \\ 0.218 & 0.370 & -0.034 & 0.217 & 0.378 & -0.044 \\ 0.503 & -0.139 & 0.990 & 0.481 & 0.138 & 0.967 \\ 0.782 & 1.097 & 0.123 & 0.772 & 1.183 & 0.088 \\ 0.516 & 0.404 & 0.426 & 0.503 & 0.545 & 0.403 \\ 0.775 & 0.553 & 0.696 & 0.754 & 0.778 & 0.661 \\ 0.508 & -0.140 & 0.999 & 0.486 & 0.140 & 0.976 \\ 0.730 & 0.775 & 0.382 & 0.715 & 0.920 & 0.349 \\ 0.384 & 0.765 & -0.183 & 0.383 & 0.748 & -0.201 \\ 0.722 & 0.578 & 0.582 & 0.704 & 0.771 & 0.549 \\ 1.089 & 0.792 & 0.964 & 1.061 & 1.104 & 0.915 \\ 1.089 & 0.792 & 0.964 & 1.061 & 1.104 & 0.915 \\ 0.924 & 0.397 & 1.113 & 0.894 & 0.734 & 1072 \\ 0.578 & 0.387 & 0.547 & 0.562 & 0.562 & 0.521 \end{bmatrix}$$

5. Perhitungan EDLSI

Perhitungan kemiripan dilakukan dengan menggunakan persamaan 2.13. query (q) yang diinputkan merupakan data Gambar 1 yang sebelummnya sudah dilakukan perubahan menjadi matrik berdimensi mx1 (Gambar 3.17), kemudian dilakukan transpose. Pada penelitian ini nilai x = 0,7 karena pada percobaan penelitian sebelumnya nilai tersebut menghasilkan MAP terbesar [3]. Akan tetapi sejauh ini pemilihan nilai x masih dilakukan pada teks dan belum diujikan

pada gambar. Berikut merupakan hasil dari perhitungan kemiripan antar gambar :

$$W = (x)(q^T . A_k) + (1 - x)(q^T . A) \qquad (2.13)$$

W = [12.131 6.526 9.913 9.995 10.167 9.604]

Hasil dari perhitungan ini merupakan bobot vektor dari masing – masing gambar, bobot vektor tersebut menunjukan kemiripan query dengan data gambar.

Tabel 3-2 Hasil Perhitungan EDLSI

No Urut	Gambar	Query	Bobot Vektor
1	Gambar 1	Gambar 1	12.131
2	Gambar 5		10.167
3	Gambar 4		9.995
4	Gambar 3		9.913
5	Gambar 6		9.604
6	Gambar 2		6.526

Kesimpulan :

Pada simulasi ini query yang di masukan merupakan data Gambar 1, sehingga Gambar 1 tidak dimasukan sebagai data pembanding pada bobot vektor. Hasil perhitungan dengan menggunakan EDLSI menunjukan bobot vektor terbesar yaitu pada data Gambar 5.

3.3 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan kebutuhan dari simulasi sistem yang akan dibangun. Spesifikasi tersebut akan berisikan hal – hal yang harus ada pada simulasi yang akan diimplementasikan.

3.3.1 Analisis kebutuhan perangkat keras

Analisis kebutuhan perangkat keras dilakukan untuk mempermudah proses perancangan dan implementasi. Berdasarkan hasil perkiraan, spesifikasi minimal perangkat keras yang direkomendasikan adalah sebagai berikut :

1. Processor 1.6 GHz
2. VGA 512 MB
3. Hard disk 500 GB
4. RAM 1 GB
5. Monitor
6. Keyboard
7. Mouse

3.3.2 Analisis kebutuhan perangkat lunak

Analisis kebutuhan perangkat lunak dilakukan untuk mempermudah proses perancangan dan implementasi. Perangkat lunak yang dibutuhkan dalam pembangunan simulasi ini adalah sebagai berikut :

1. Sistem Operasi : Windows 7
2. Visual Studio 2012

3.3.3 Analisis pengguna

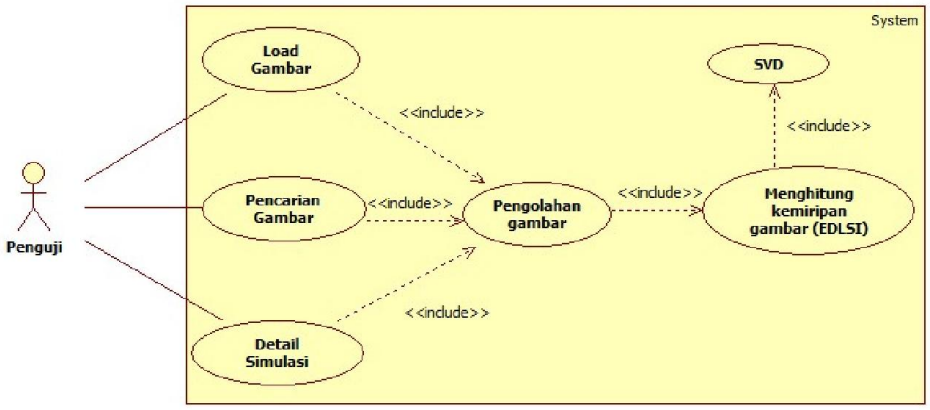
Untuk menjalankan simulasi yang dibangun dibutuhkan spesifikasi dari pengguna agar dapat menjalankan fungsionalitas yang terdapat pada simulasi. Hanya terdapat satu jenis pengguna yang dibutuhkan yaitu penguji. Adapun pengguna yang dibutuhkan untuk menjalankan simulasi yaitu dapat mengoprasikan komputer dan dapat memahami cara menggunakan aplikasi yang diberikan.

3.4 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah simulasi dan menjelaskan kebutuhan yang diperlukan simulasi agar berjalan dengan baik. Kebutuhan fungsional dalam pembangunan aplikasi ini dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Adapun tahapan pemodelan dalam analisis ini sebagai berikut [16]:

3.4.1 Use Case Diagram

Use case diagram merupakan gambaran umum dari perancangan sistem yang dibuat untuk membantu pengguna untuk mendapatkan pemahaman yang utuh mengenai sistem yang dibangun. Berikut merupakan gambaran untuk use case diagram sistem yang di bangun :



Gambar 3.18 *Use Case Diagram*

3.4.1.1 Definisi Aktor

Pada use case diagram ini aktor merepresentasikan seseorang yang berinteraksi dengan sistem.

Tabel 3-3 Definisi Aktor

No	Aktor	Deskripsi
1	Penguji	Merupakan aktor yang mempunyai peran untuk melakukan semua proses yang ada pada sistem.

3.4.1.2 Definisi Use Case

Pada bagian ini diisi dengan daftar use case dan deskripsi singkat mengenai use case yang ada pada sistem yang akan dibangun. Deskripsi tersebut dapat dilihat pada tabel 3-4.

Tabel 3-4 Definisi *Use Case*

No	Use Case	Deskripsi
1	Load Gambar	Merupakan fungsionalitas yang digunakan oleh penguji untuk menginput data gambar ke dalam sistem.
2	Pencarian Gambar	Merupakan fungsionalitas yang digunakan oleh penguji untuk mencari gambar dalam sistem.
3	Detail Simulasi	Merupakan fungsionalitas yang digunakan untuk menampilkan alur contoh simulasi secara keseluruhan.
4	Pengolahan Gambar	Merupakan fungsionalitas yang digunakan untuk mengolah gambar RGB menjadi gambar biner dengan menggunakan metode grayscale dan deteksi tepi *canny.*
5	SVD	Merupakan fungsionalitas yang digunakan untuk mendekomposisi matriks dari kumpulan gambar.
6	Menghitung kemiripan gambar (EDLSI)	Merupakan fungsionalitas yang digunakan untuk menghitung kemiripan gambar.

3.4.1.3 Use Case Skenario

Pada bagian ini diisi dengan skenario untuk setiap *use case* pada rancangan yang menggambarkan urutan interaksi aktor dengan sistem.

1. *Use Case Secenario* Load Gambar

   Load Gambar digunakan untuk memasukan gambar ke dalam sistem yang nantinya akan digunakan pada tahap perhitungan kemiripan antara *query* dengan gambar yang sudah ada pada sistem. Use case scenario dapat dilihat pada tabel 3-5.

Tabel 3-5 Load Gambar

Use Case Name	Load Gambar	
Goal In Context	User dapat memasukan data gambar ke dalam sistem	
Related Use Case	Pengolahan gambar, SVD	
Preconditions	-	
Successful End Condition	Data gambar berhasil di masukan ke dalam sistem	
Failed End Condition	Data gambar gagal di masukan ke dalam sistem	
Actors	Penguji	
Trigger	Penguji telah membuka aplikasi	
Include Cases	Pengolahan Gambar, SVD	
Main Flow	**Step**	**Action**
	1	Penguji meminta sistem menampilkan tab Load Gambar
	2	Sistem akan menampilkan halaman Load Gambar
	3	Penguji melakukan penentuan k-rank dan load data gambar
	4	Sistem menampilkan halaman explorer
	5	Penguji memilih folder yang berisi gambar dan menekan tombol pilih
	6	Sistem memasukan data gambar yang ada pada folder yang dipilih
	7 **Include :: Pengolahan Gambar**	Sistem melakukan pengolahan gambar

	8 **Include :: SVD**	Sistem melakukan perhitungan SVD dengan data masukan berupa matriks dari kumpulan gambar
	9	Sistem menampilkan gambar asli dan pengolahan gambar berupa matrik dari tiap gambar.
Extension	**Step**	**Branching Action**
	6.1	Sistem gagal memasukan data
	7.1	Sistem tidak dapat melakukan pengolahan gambar
	8.1	Sistem gagal menghitung SVD
	9.1	Sistem tidak dapat menampilkan gambar dan hasil pengolahan gambar

2. *Use Case Scenario* Pencarian Gambar

   Cari gambar digunakan oleh penguji untuk mencari gambar dalam sistem dengan menggunakan *query* gambar yang di inputkan oleh penguji. *Use case scenario* cari gambar dapat dilihat pada tabel 3-6.

Tabel 3-6 Pencarian Gambar

Use Case Name	Pencarian gambar
Goal In Context	Penguji dapat mencari gambar yang sesuai dengan *query* yang di inputkan
Related Use Case	*Image processing*, SVD, menghitung kemiripan gambar (EDLSI), menampilkan gambar hasil perhitungan EDLSI
Preconditions	-
Successful End Condition	Sistem dapat menampilkan kumpulan gambar yang dinilai memiliki kemiripan dengan *query* yang sudah di inputkan oleh penguji

Failed End Condition	Sistem tidak dapat menampilkan kumpulan gambar.	
Actors	Penguji	
Trigger	Penguji telah melakukan Load Gambar	
Include Cases	Pengolahan Gambar, SVD, Menghitung kemiripan gambar (EDLSI)	
Main Flow	**Step**	**Action**
	1	Penguji meminta sistem memasukan data *query* gambar
	2	Sistem menampilkan *file directori*
	3	Penguji memilih *file direktori* dan gambar dan memilih satu gambar
	4 **Include :: Pengolahan gambar**	Sistem melakukan proses pengolahan gambar.
	5 **Include :: SVD**	Sistem melakukan proses perhitungan SVD dengan data hasil dari pengolahan gambar.
	6 **Include :: Menghitung kemiripan gambar (EDLSI)**	Sistem melakukan proses perhitungan kemiripan gambar (EDLSI)
	7 **Include :: Menampilkan hasil kemiripan gambar**	Sistem menampilkan gambar hasil dan perhitungan kemiripan gambar

3. *Use Case Scenario* Detail Simulasi

   Detail simulasi merupakan gambaran dari proses yang terjadi pada simulasi yang dibangun. *Use case scenario* pengolahan gambar dapat dilihat pada tabel 3-7.

Tabel 3-7 Detail Simulasi

Use Case Name	Detail Simulasi	
Goal In Context	Sistem dapat menunjukan proses dari simulasi yang dibangun	
Related Use Case	Pengolahan gambar, SVD, Menghitung kemiripan gambar (EDLSI).	
Preconditions	-	
Successful End Condition	Sistem dapat menunjukan hasil dari setiap proses simulasi	
Failed End Condition	Sistem tidak dapat menunjukan hasil proses dari setiap proses simulasi	
Actors	Penguji	
Trigger	-	
Main Flow	**Step**	**Action**
	1	Penguji melakukan load gambar
	2 **Include :: Pengolahan Gambar**	Sistem melakukan pengolahan gambar
	3	Sistem menampilkan gambar asli, gambar *grayscale*, gambar deteksi tepi *canny*, dan nilai ekstraksi gambar
	4 **Include :: SVD**	Sistem melakukan perhitungan SVD dengan input kumpulan

		gambar yang dijadikan satu matriks.
	5	Sistem menampilkan hasil perhitungan SVD yang berupa matrik U, S, V^T, U_k, S_k, dan V^T_k .
	6	Penguji memasukan *query* yang berupa gambar.
	7 **Include :: Pengolahan Gambar**	Sistem melakukan pengolahan gambar
	8	Sistem menampilkan hasil dari pengolahan gambar
	9 **Include :: Menghitung kemiripan gambar (EDLSI)**	Penguji meminta sistem untuk melakukan perhitungan kemiripan gambar
	10 **Include :: Menampilkan hasil kemiripan gambar**	Sistem menampilkan gambar dan nilai hasil perhitungan kemiripan pada setiap gambar.

4. *Use Case Scenario* Pengolahan gambar

   Pengolahan gambar digunakan sistem untuk mengolah data gambar baik itu data uji (Load Gambar) maupun gambar *query* (Pencarian Gambar) menjadi data yang bisa dihitung kemiripanya. *Use case scenario* pengolahan gambar dapat dilihat pada tabel 3-8.

Tabel 3-8 Use Case Scenario Pengolahan Gambar

Use Case Name	Pengolahan Gambar	
Goal In Context	Mendapatkan nilai *pixel* dari setiap gambar	
Related Use Case	Load Gambar, Pencarian Gambar, Detail Simulasi	
Preconditions	Load Gambar, Cari Gambar	
Successful End Condition	Data gambar menjadi data yang dapat diambil nilai *pixel* dari setiap gambar dan dijadikan matriks	
Failed End Condition	Nilai pixel gambar tidak dapat diambil.	
Actors	Sistem	
Trigger	-	
Main Flow	**Step**	**Action**
	1	Sistem mengambil data gambar (Load Gambar) maupun (Cari Gambar).
	2	Sistem melakukan konversi gambar dari RGB menjadi *Grayscale*
	3	Sistem melakukan pendeteksian tepi dari gambar *grayscale* dengan menggunakan metode deteksi tepi *canny*
	4	Sistem melakukan ekstraksi pada setiap gambar yang sudah melewati proses *grayscale* dan *canny*, kemudian hasil tersebut dijadikan matriks

5. *Use Case Scenario* SVD

   SVD digunakan untuk mendekomposisikan matriks kumpulan gambar hasil pengolahan gambar yang dibutuhkan untuk proses perhitungan kemiripan gambar. *Use case scenario* SVD dapat dilihat pada tabel 3-9.

Tabel 3-9 Use Case Scenario SVD

Use Case Name	SVD	
Goal In Context	Sistem dapat mendekomposisikan matrik dari kumpulan gambar	
Related Use Case	Load Gambar, Pengolahan gambar, Detail Simulasi	
Preconditions	Load Gambar, Pengolahan gambar	
Successful End Condition	Sistem dapat mendekomposisi matrik kumpulan gambar dan menhasilkan matrik U,S,V, U_k, S_k , dan V_k	
Failed End Condition	Sistem tidak mendekomposisikan matrik kumpulan gambar.	
Actors	Sistem	
Trigger	-	
Main Flow	**Step**	**Action**
	1	Sistem mengambil data hasil proses pengolahan gambar dari data gambar pada sistem.
	2	Sistem melakukan pembuatan matriks kumpulan gambar
	3	Sistem melakukan perhitungan Singular Value Decomposition
	4	Sistem mengeluarkan hasil perhitungan berupa matrik U,S,V^T, U_k, S_k , dan V^T_k
Extension	**Step**	**Branching Action**
	4.1	Sistem gagal mengeluarkan hasil perhitungan.

6. *Use Case Scenario* Menghitung Kemiripan Gambar

   Menghitung kemiripan gambar EDLSI digunakan untuk menghitung kemiripan gambar yang sebelumnya sudah melewati proses *image processing*. *Use case scenario* menghitung kemiripan gambar dapat dilihat pada tabel 3-10.

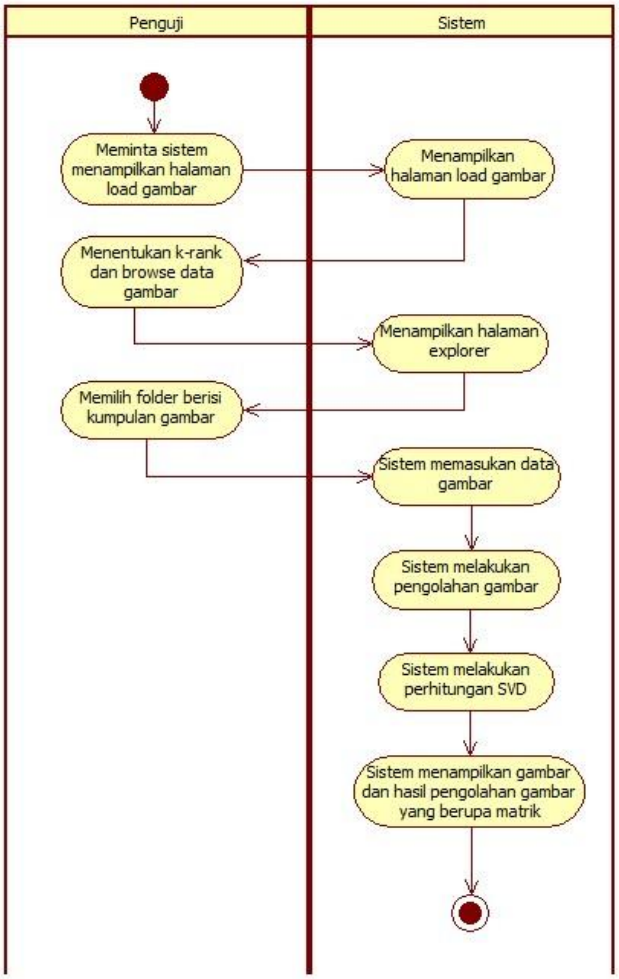
Tabel 3-10 Use Case Scenario Kemiripan Gambar

Use Case Name	Menghitung kemiripan gambar (EDLSI)	
Goal In Context	Sistem dapat menentukan kemiripan gambar query dengan gambar yang sudah ada pada sistem	
Related Use Case	Pencarian gambar, Pengolahan gambar, SVD	
Preconditions	Pencarian gambar	
Successful End Condition	Sistem dapat menghitung kemiripan gambar dalam sistem dengan gambar query yang diinputkan oleh penguji	
Failed End Condition	Sistem tidak dapat menghitung kemiripan gambar	
Actors	Sistem	
Trigger	-	
Include Cases	Pengolahan gambar, SVD	
Main Flow	**Step**	**Action**
	1 **Include :: Pengolahan gambar**	Sistem mendapatkan data hasil proses pengolahan gambar dari data gambar pada sistem.
	2 **Include :: SVD**	Sistem melakukan perhitungan SVD
	3	Sistem memasukan *query* gambar yang sudah melewati *image processing.*
	4	Sistem melakukan perhitungan kemiripan gambar dengan menggunakan metode EDLSI
	5	Sistem menghasilkan nilai hasil perhitungan dari setiap kemiripan gambar

3.4.2 Activity Diagram

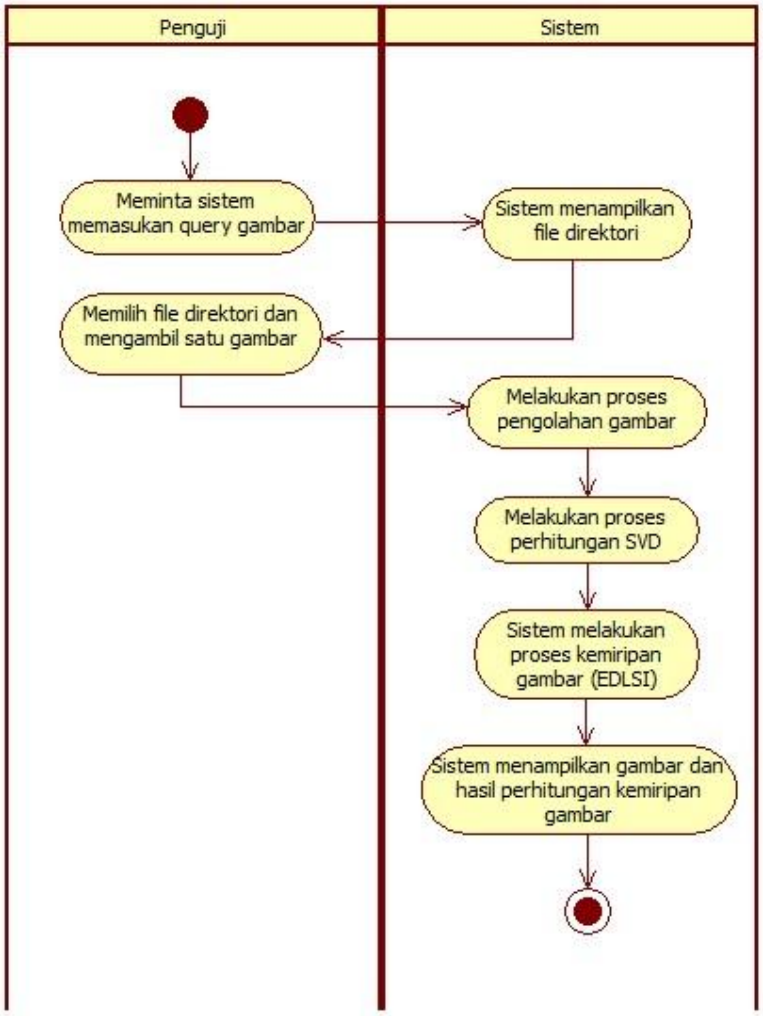
Pada bagian ini menjelaskan tentang alur kerja tahapan – tahapan aktivitas dari use case yang akan dibangun. Adapun *activity diagram* dari setiap use case adalah sebagai berikut.

1. Activity Diagram Load Gambar



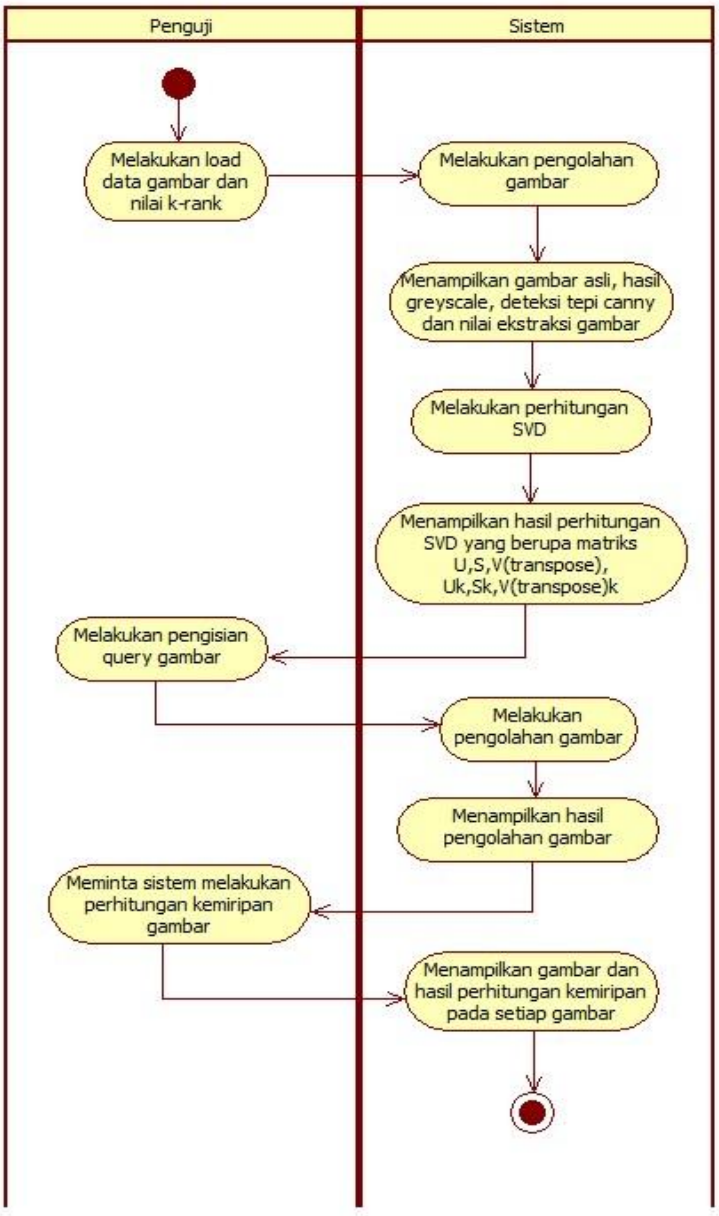
Gambar 3.19 Activity Diagram Load Gambar

2. Activity Diagram Pencarian Gambar



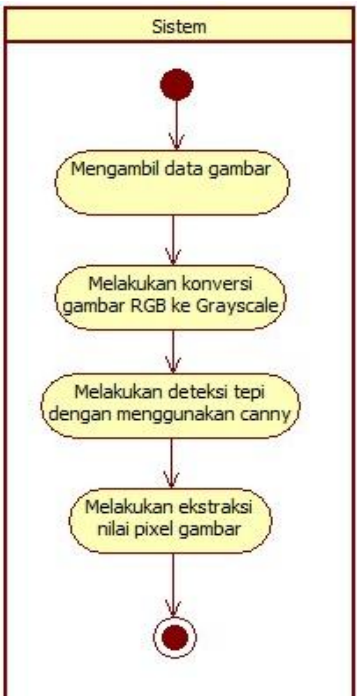
Gambar 3.20 Activity Diagram Cari Gambar

3. Activity Diagram Detail Simulasi



Gambar 3.21 Activity Diagram Detail Simulasi

4. Activity Diagram Pengolahan Gambar



Gambar 3.22 Activity Diagram Image Processing

5. Activity Diagram SVD



Gambar 3.23 Activity Diagram SVD

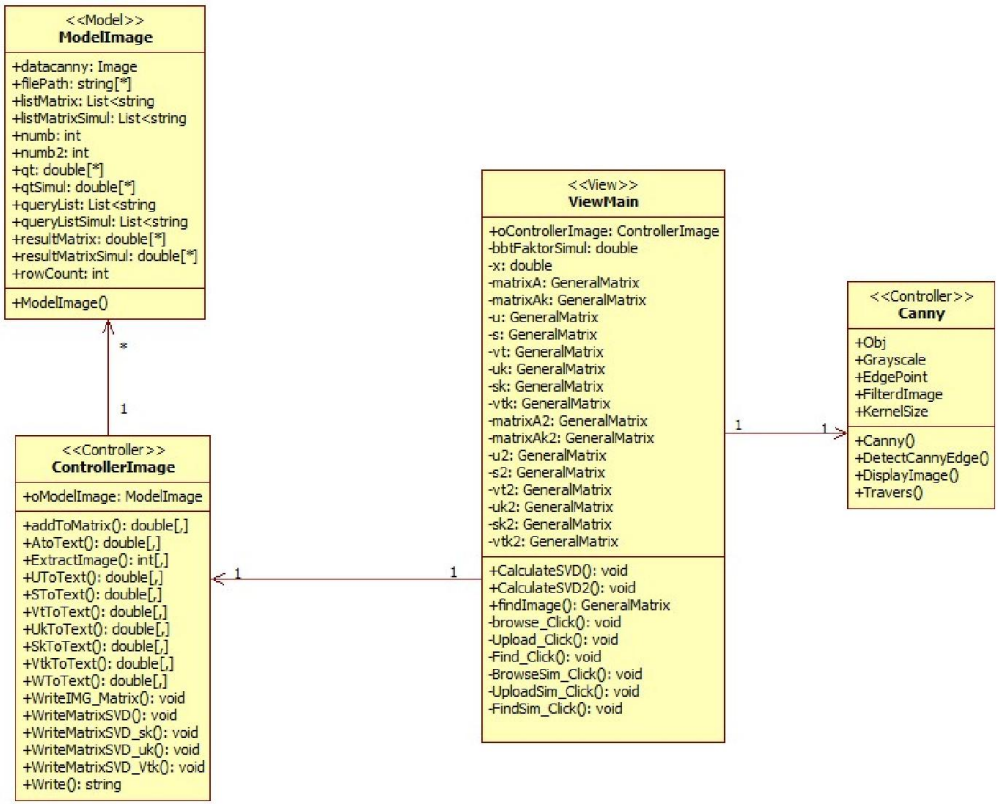
6. Activity Diagram Menghitung Kemiripan Gambar



Gambar 3.24 Activity Diagram Menghitung Kemiripan Gambar

3.4.3 Class diagram

Berikut merupakan class diagram pada simulasi yang akan dibuat yaitu :

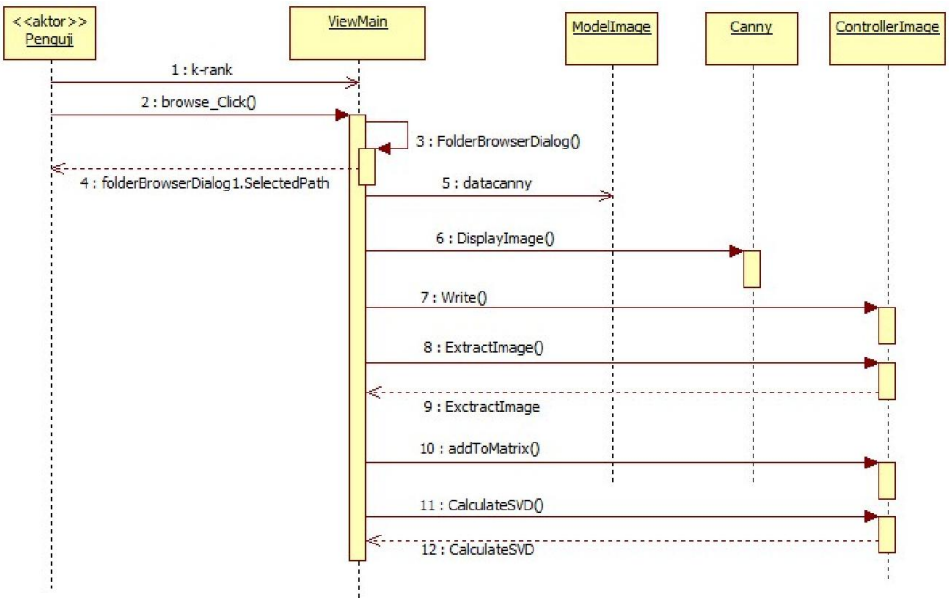


Gambar 3.25 Class Diagram

3.4.4 Sequence diagram

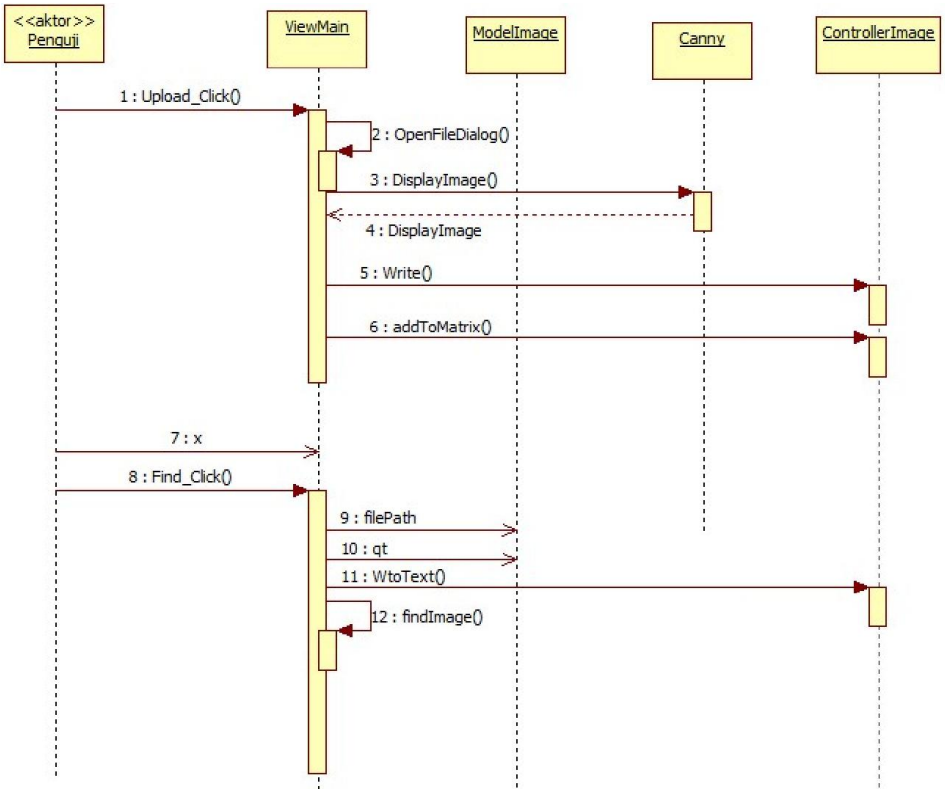
Sequence diagram menggambarkan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Adapun *sequence diagram* ditampilkan sebagai berikut.

1. Sequence Diagram Load Gambar



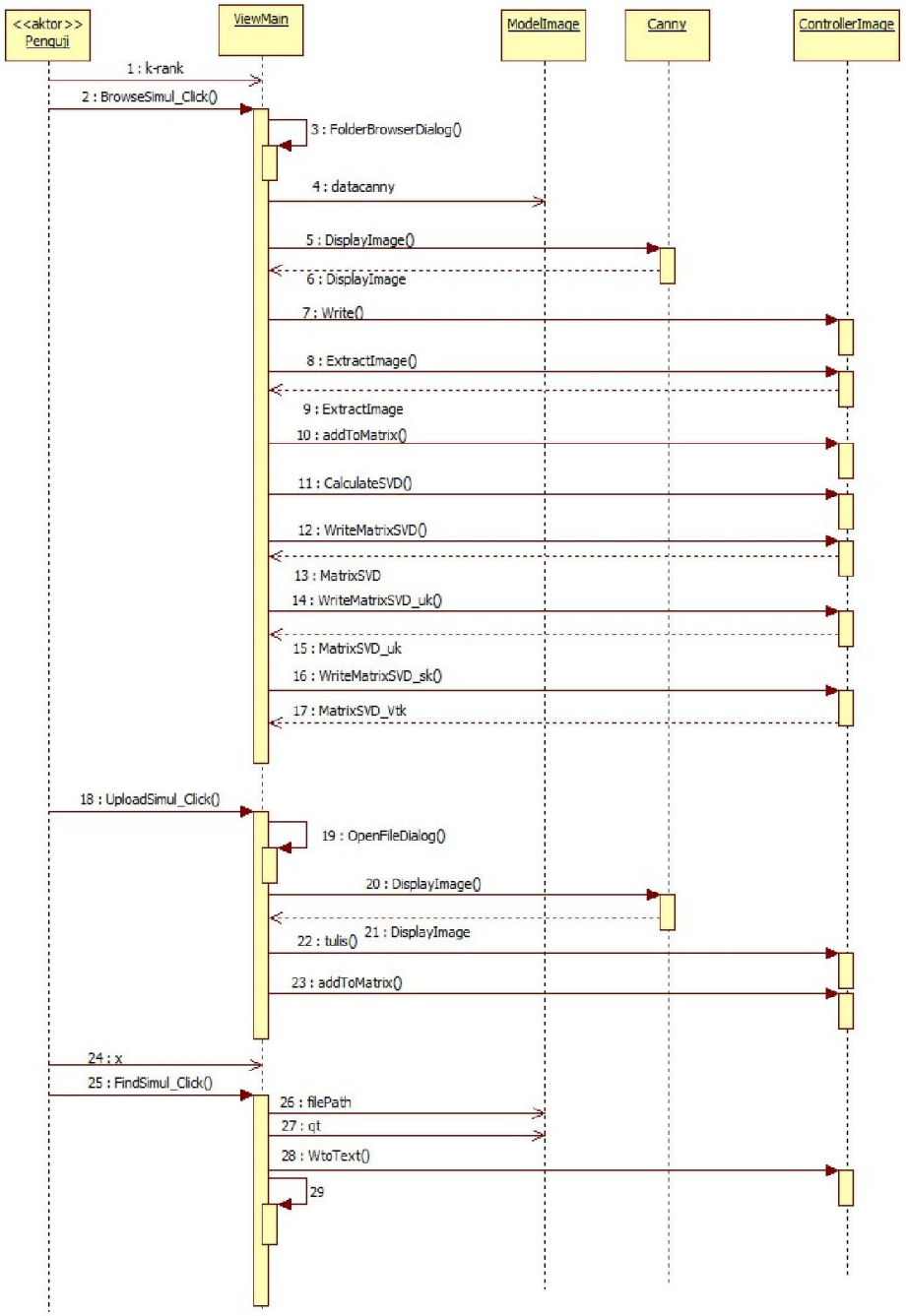
Gambar 3.26 Sequence Diagram Load Gambar

2. Sequence Diagram Pencarian Gambar



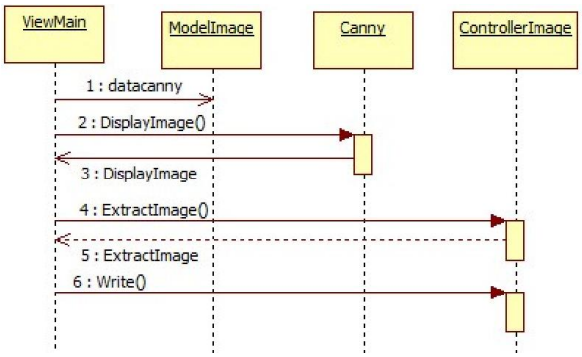
Gambar 3.27 Sequence Diagram Pencarian Gambar

3. Sequence Diagram Detail Simulasi



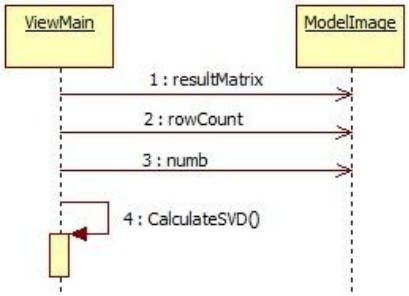
Gambar 3.28 Sequence Diagram Detail Simulasi

4. Sequence Diagram Pengolahan Gambar



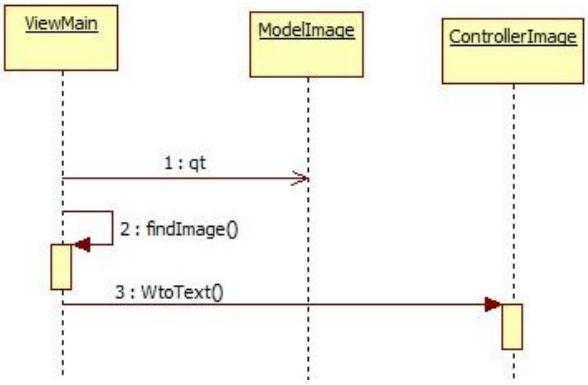
Gambar 3.29 Sequence Diagram Pengolahan Gambar

5. Sequence Diagram SVD



Gambar 3.30 Sequence Diagram SVD

6. Sequence Diagram Menghitung Kemiripan EDLSI



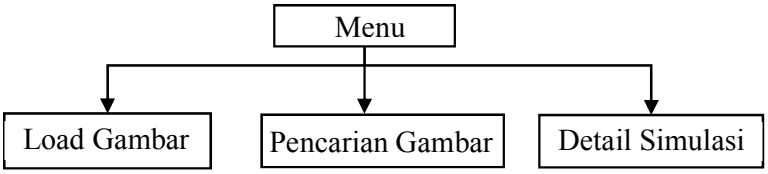
Gambar 3.31 Sequence Diagram Menghitung Kemiripan EDLSI

3.5 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem merupakan perencanaan, penggambaran, pembuatan sketsa dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh. Pada penelitian ini perancangan terdapat pada perancangan antarmuka, perancangan pesan dan perancangan jaringan semantik.

3.5.1 Pencancangan Struktur Menu

Berikut merupakan struktur menu dari simulator yang akan dibangun :

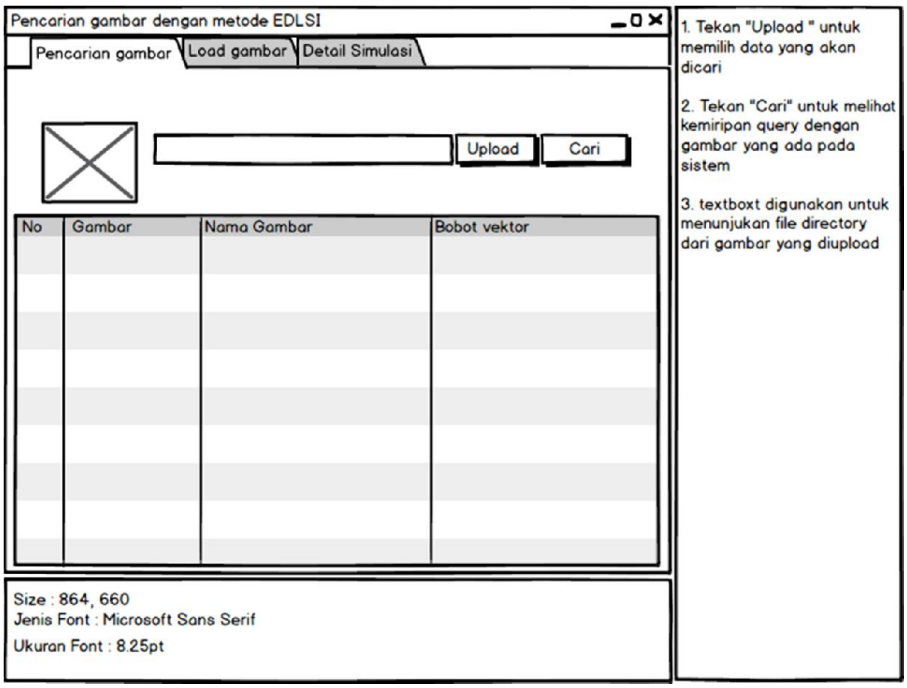


Gambar 3.32 Struktur Menu

3.5.2 Perancangan Antarmuka

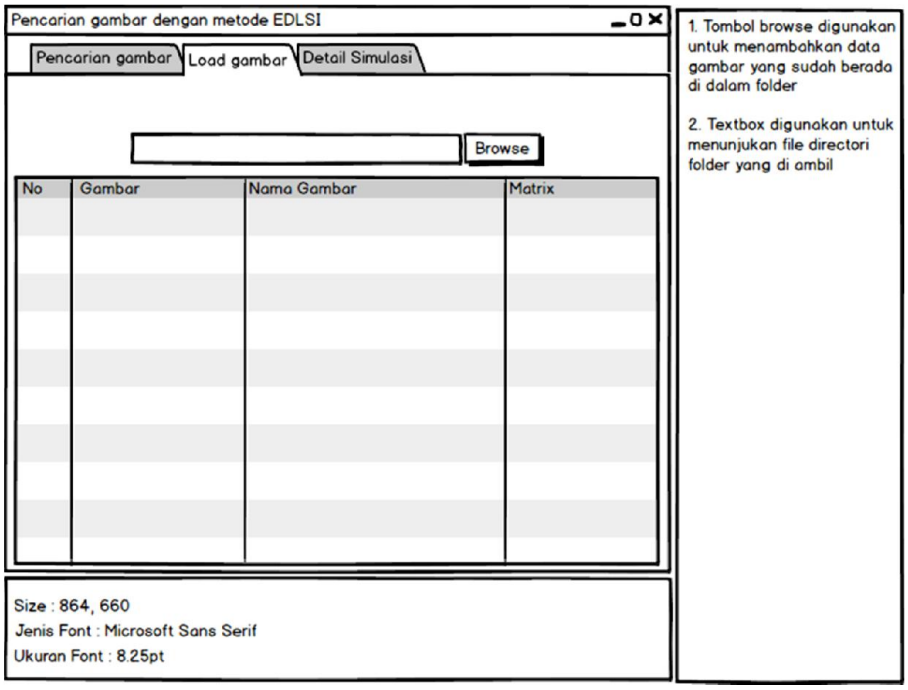
Rancangan antarmuka dari aplikasi yang akan dibangun adalah sebagai berikut.

1. Antarmuka menu pencarian gambar (F01)



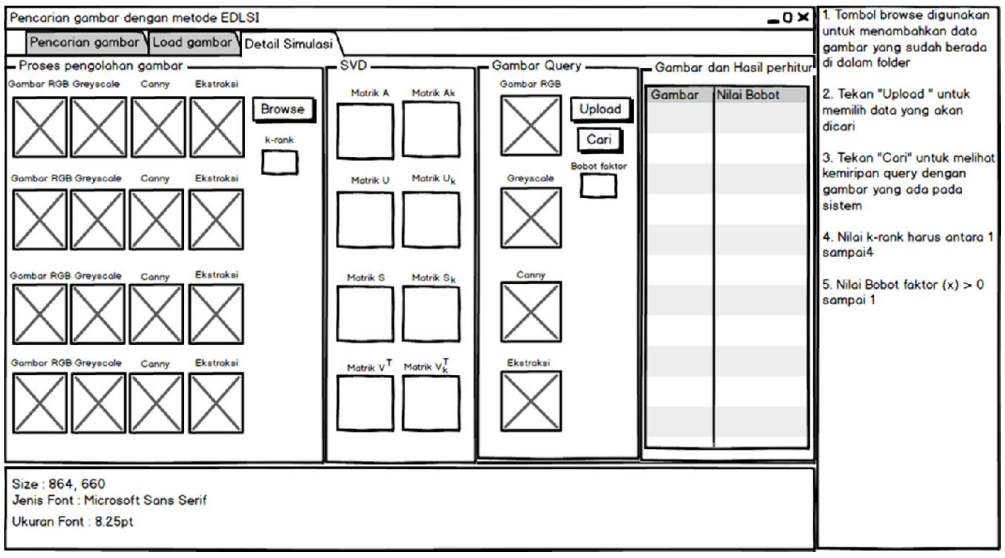
Gambar 3.33 Antarmuka Tab Pencarian Gambar

2. Antarmuka menu load gambar (F01)



Gambar 3.34 Antarmuka Tab Load Gambar

3. Antarmuka menu detail simulasi (F01)

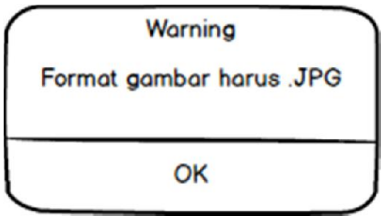


Gambar 3.35 Antarmuka Tab Detail Simulasi

3.5.3 Perancangan Pesan

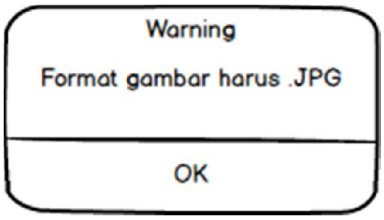
Rancangan menu pesan yang akan dibangun pada simulator ini adalah sebagai berikut :

1. Peringatan format gambar yang diinputkan pada pencarian gambar (M01)



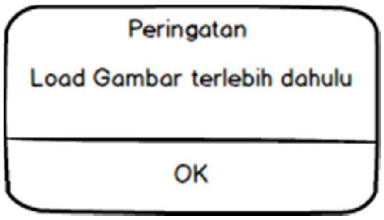
Gambar 3.36 Pesan Format gambar pada pencarian gambar

2. Peringatan format gambar yang di inputkan pada load gambar (M02)



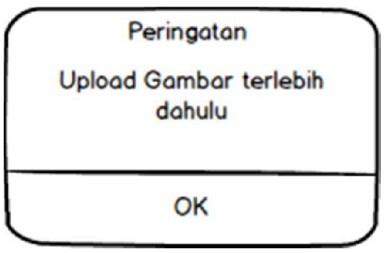
Gambar 3.37 Pesan Format Gambar Pada Load Gambar

3. Peringatan melakukan pencarian gambar tanpa load gambar (M03)



Gambar 3.38 Pesan Pencarian Tanpa Load Gambar

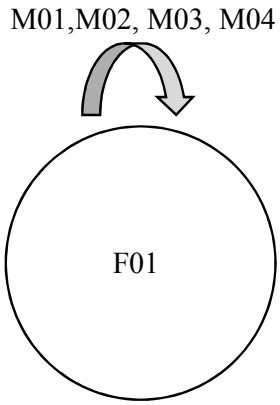
4. Peringatan melakukan pencarian tanpa upload gambar (M04)



Gambar 3.39 Pesan Pencarian Gambar Tanpa Upload Gambar

3.5.4 Perancangan Jaringan Semantik

Jaringan semantik menggambarkan keterhubungan dari navigasi menu dari saru form ke form lain. Berikut merupakan jaringan semantik yang akan dibangun pada simulator :



Gambar 3.40 Jaringan Semantik