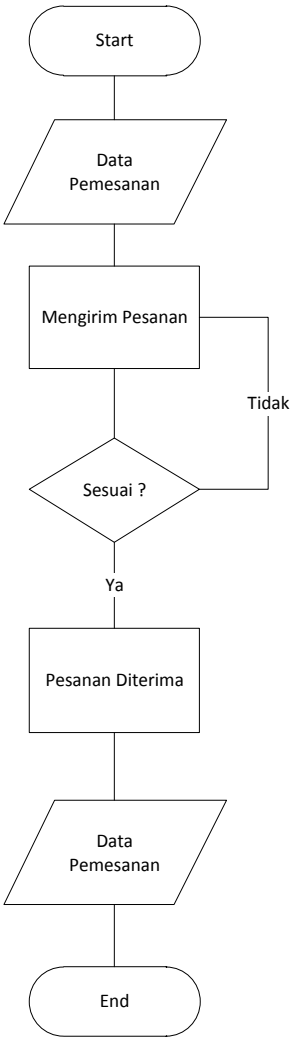
BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Proses Bisnis

Proses bisnis yang sedang berjalan pada SUI Warehouse Production memiliki satu proses bisnis yang utama. Proses bisnis tersebut pemesanan pembuatan produk dari konsumen, alur dari proses bisnis dari proses bisnis ini dapat dilihat pada gambar 3.1



Gambar 3.1 Alur bisnis pemesanan produk dari konsumen

3.2 Analisis Sistem

Analisis sistem merupakan suatu proses penguraian konsep kedalam bagian-bagian yang lebih sederhana dengan maksud untuk mengetahui, mengidentifikasi, dan mengevaluasi masalah-masalah yang ada, hambatan-hambatan, kemungkinan-kemungkinan dari setiap solusi dan kebutuhan dari sistem atau aplikasi yang akan dibuat. Analisis sistem terdiri dari beberapa tahapan, diantaranya analisis masalah, analisis metode, analisis kebutuhan non-fungsional, dan analisis kebutuhan fungsional.

3.3 Analisis Masalah

Perusahaan ini memproduksi t-shirt hampir ratusan lebih dari beberapa distro. Pada saat aktifitas seperti pengiriman, atau pun pengembalian barang perusahaan ini sangat kebingungan mencari informasi dari sebuah t-shirt dikarenakan banyaknya t-shirt yang diproduksi.

Berdasarkan hasil wawancara yang telah dilakukan dengan pihak SUI Warehouse didapatkan hasil sebagai berikut :

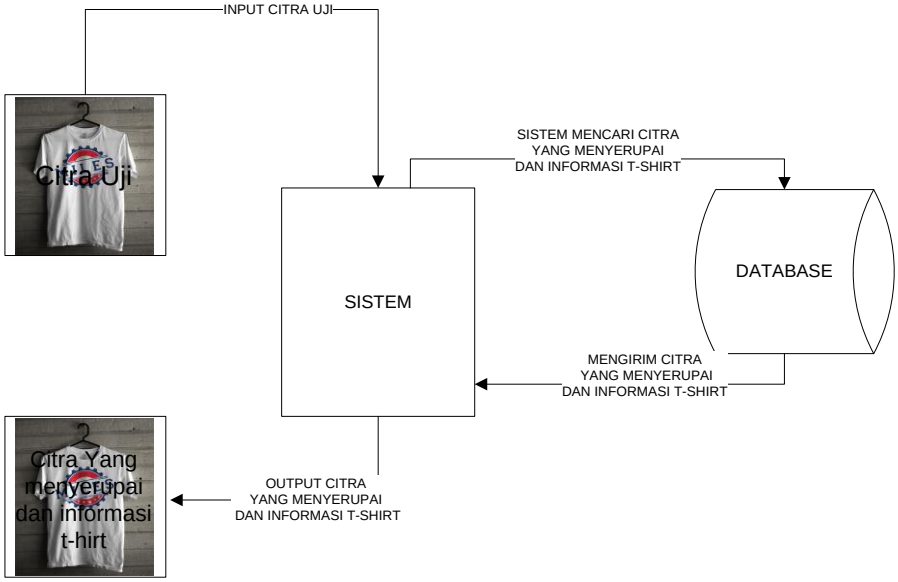
perusahaan ini memproduksi t-shirt dalam jumlah ratusan dari beberapa distro. Untuk membuat t-shirt yang akan dipesan oleh distro, distro diharuskan untuk memberikan gambar kepada perusahaan.

1. Gambar yang diberikan oleh pihak distro akan dijadikan acuan untuk mendapatkan informasi seperti nama artikel t-shirt, harga t-shirt, qty t-shirt.
2. Untuk mencari informasi dalam pengiriman dan pengembalian t-shirt pihak perusahaan mencari gambar dan informasinya secara manual.
3. Jumlah gambar yang dihasilkan sangat banyak, maka perusahaan kesulitan untuk mendapatkan informasi dari sebuah t-shirt, dan membutuhkan waktu yang lama untuk mendapatkan informasi dari gambar yang dimaksud.
4. Perusahaan memerlukan informasi yang cepat dari sebuah t-shirt tersebut.

3.4 Gambaran Sistem

Arsitektur sistem adalah gambaran proses dari sistem yang akan dibangun

dalam penelitian ini. Adapun arsitektur sistemnya dapat dilihat pada gambar 3.1.



Gambar 3.2 Arsitektur Sistem

3.5 Arsitektur Sistem

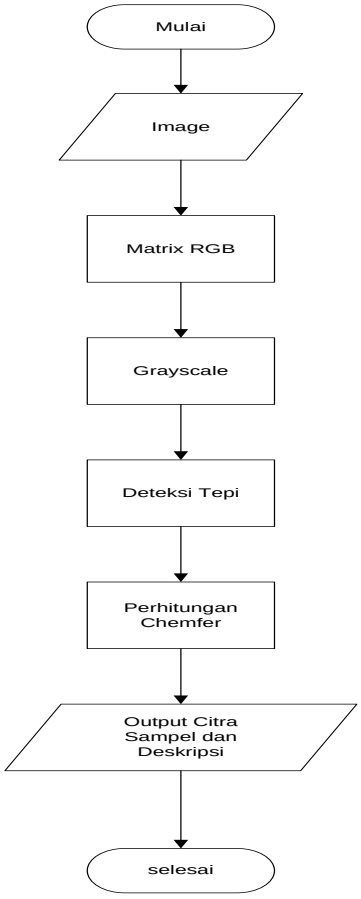
1. Pada tahap awal, citra yang akan dicari informasinya di-*input*-kan kedalam sistem.
2. Selanjutnya sistem akan melakukan proses dan mencari citra yang menyerupai yang ada didalam *database* kemudian mengambil informasi yang menyangkut dengan citra tersebut.
3. Kemudian sistem akan mengambil citra dan informasi yang telah didapatkan.
4. Selanjutnya sistem akan menampilkan *output* citra yang menyerupai dan informasi *t-shirt* mengenai citra tersebut.

3.6 Analisis Metode

Analisis metode digunakan untuk mengetahui alur proses dari sebuah metode yang digunakan dapat diterapkan ke dalam aplikasi yang dibangun dan contoh penerapannya pada beberapa gambar sebagai berikut :

3.7 Tahapan Metode

Pada bagian ini akan dijelaskan tahapan-tahapan yang akan dilakukan dalam melakukan proses pengidentifikasian citra oleh aplikasi yang dibangun. Tahapan metode aplikasi ini dapat dilihat pada gambar 3.3



Gambar 3.3 Tahapan Metode Aplikasi

3.7.1 Citra Uji Dan Citra Latih

Citra yang dimasukan disebut dengan citra uji dan citra latih, dimisalkan adalah gambar dengan ukuran 5 x 5 berikut

Citra yang akan diuji pada gambar 3.4 :



Gambar 3.4 Gambar Uji

Gambar 3.5 adalah beberapa citra latihnya :







Citra latih 1 Citra latih 2 Citra latih 3

Gambar 3.5 Citra Latih

3.7.2 Matriks RGB

Citra masukan kemudian diubah menjadi sebuah matriks yang memiliki 3 matriks warna, yaitu *Red, Green* dan *Blue.*Untuk mengubah kedalam matrik RGB menggunakan Aplikasi Matlab, dengan mengetikan perintah :

>>c=imread(‘C:/uji.jpg’);

Kemudian hasilnya seperti dibawah ini, karena diambil semple maka matriks menjadi dibawah ini, pada Gambar 3.6 terdapat asci (;,;,1), asci (;,;,2),

asci (;,;,3) menunjukan bahwa matriks yang ditampilkan adalah komponen matriks dari warna yaitu merah *(Red)*, hijau *(Green)*, biru *(Blue)*.

Matrik RGB citra uji :

80	75	55	109	112
56	172	161	198	119
68	170	117	164	169
58	142	134	190	128
42	131	163	175	117

Matriks R

78	74	51	106	109
54	171	157	195	116
66	169	116	164	166
56	141	133	190	125
40	130	162	175	114

Matriks G

79	72	48	99	102
55	169	154	188	109
67	167	112	156	159
57	139	129	182	118
41	128	158	167	107

Matriks B

Gambar 3.6 Matriks R, G, B Citra Uji

Matrik RGB citra latih 1 :

80	75	55	109	112
56	172	161	198	119
68	170	117	164	169
58	142	134	190	128
42	131	163	175	117

Matriks R

78	74	51	106	109
54	171	157	195	116
66	169	116	164	166
56	141	133	190	125
40	130	162	175	114

Matriks G

79	72	48	99	102
55	169	154	188	109
67	167	112	156	159
57	139	129	182	118
41	128	158	167	107

Matriks B

Gambar 3.7 Matriks R, G, B Citra Latih 1

Matrik RGB citra latih 2 :

76	77	58	112	108
50	143	133	146	137
66	104	81	125	145
52	104	129	143	120
28	68	94	100	82

Matriks R

76	77	57	111	107
50	143	132	145	136
66	104	80	124	144
52	104	128	142	119
28	68	93	99	81

Matriks G

74	75	53	106	102
48	141	128	140	131
64	102	76	119	139
50	102	124	138	114
26	66	89	95	76

Matriks B

Gambar 3.8 Matriks R, G, B Citra Latih 2

Matrik RGB citra latih 3 :

253	255	233	254	255
250	170	149	177	255
242	137	125	171	250
253	157	160	178	254
255	172	181	164	254

Matriks R

253	255	233	254	255
250	170	149	177	255
242	137	125	171	250
253	157	160	178	254
255	172	181	164	254

Matriks G

253	255	233	254	255
250	170	149	177	255
242	137	125	171	250
253	157	160	178	254
255	172	181	164	254

Matriks B

Gambar 3.9 Matriks R, G, B Citra Latih 3

3.7.3 Matriks Grayscale

Grayscale merupakan citra yang mempunyai nilai dari setiap pixel sampel tunggal. Citra yang ditampilkan yaitu warna abu-abu, variasi warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat.

Kemudian analisis yang dilakukan yaitu membuat matriks *grayscale* menggunakan rumus (2.7). Contoh perhitungan baris 1 kolom 1:

$$grayscale[1,1] = \frac{(R[1,1] + G[1,1] + B[1,1])}{3}$$

$$grayscale[1,1] = \frac{(80 + 78 + 79)}{3}$$

$$grayscale[1,1] = \frac{237}{3}$$

$$grayscale[1,1] = 79$$

Tabel 3-1 Hasil Grayscale Citra Uji

79	74	51	105	108
55	171	157	194	115
67	169	115	161	165
57	141	132	187	124
41	130	161	172	113

Tabel 3-2 Hasil Grayscale Citra Latih 1

79	74	51	105	108
55	171	157	194	115
67	169	115	161	165
57	141	132	187	124
41	130	161	172	113

Tabel 3-3 Hasil Grayscale Citra Latih 2

75	76	56	110	106
49	142	131	144	135
65	103	79	123	143
51	103	127	141	118
27	67	92	98	80

Tabel 3-4 Hasil Grayscale Citra Latih 3

253	255	233	254	255
250	170	149	177	255
242	137	125	171	250

253	157	160	178	254
255	172	181	164	254



Gambar 3.10 Citra Grayscale

3.7.4 Analisis Deteksi Tepi Canny

Langkah pertama pada deteksi tepi *canny* adalah mengkonvolusikan hasil matriks *grayscale* dengan tapis *Gaussian*, seperti pada metode *Gaussian* (2.8). menghasilkan Tabel 3.5.

Algoritma deteksi tepi menggunakan canny

Langkah pertama pada deteksi tepi canny adalah mengkonvolusikan hasil matriks *grayscale* dengan tapis *Gaussian*, dengan persamaan tapis gausian (3.5).

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad x$$

79	74	51	105	108
55	171	157	194	115
67	169	115	161	165
57	141	132	187	124
41	130	161	172	113

Kemudian dihitung seperti perhitungan matrik seperti biasanya, sebagai salah satu contoh perhitungan konvolusi gaussian sebagai berikut :

= (2x79) + (4x55) + (5x67) + (4x57) + (2x41) / 115

= 8,895

Tabel 3-5 Hasil Gaussian Citra Uji

9	22	19	25	19
20	49	42	56	43
26	63	54	73	56
20	49	42	56	43
9	22	19	25	19

Kemudian pencarian dilakukan secara horizontal (G_x) dan secara vertical (G_y) yaitu mengkonvolusikan dengan kernel arah horizontal (Gx) dan vertical(Gy). Sehingga pada proses ini citra *grayscale* ditentukan sesuai dengan baris dan kolom yang disoroti untuk dikalikan dengan kernel *gaussian x* dan *y*. Dalam perhitungan *gaussian* didapatkan dari proses sebelumnya :

$$GaussianX = \begin{bmatrix} 9 & 22 & 19 & 25 & 19 \\ 20 & 49 & 42 & 56 & 43 \\ 26 & 63 & 54 & 73 & 56 \\ 20 & 49 & 42 & 56 & 43 \\ 9 & 22 & 19 & 25 & 19 \end{bmatrix} \text{x} \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Sehingga dapat dijabarkan perhitungan konvolusi tersebut seperti berikut :

Perhitungan Sobel Gx :

$$\text{Sobel Gx(1,1)} = \begin{bmatrix} 9 & 22 & 19 \\ 20 & 49 & 42 \\ 26 & 63 & 54 \end{bmatrix} \text{x} \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

$$\text{Sobel Gx(1,2)} = \begin{bmatrix} 22 & 19 & 25 \\ 49 & 42 & 56 \\ 63 & 54 & 73 \end{bmatrix} \text{x} \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

.
.
.

$$\text{Sobel Gx(3,3)} = \begin{bmatrix} 54 & 73 & 56 \\ 42 & 56 & 43 \\ 19 & 25 & 19 \end{bmatrix} \text{x} \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Cara menghitung *gaussianX* tersebut dapat dijelaskan pada contoh *GaussianX*[1,1] dimana hasil dari perhitungan tersebut berikut :

GaussianX [1,1] = 9*(-1) + 19*1 + 20*(-2) + 42*2 + 26*(-1) + 54*1

GaussianX [1,1] = 83

Setelah dihitung semua baris dan kolom, maka dihasilkan nilai seperti dilihat pada matriks berikut :

Tabel 3-6 Hasil Perhitungan GaussianX Citra Uji

0	0	0	0	0
0	83	27	5	0
0	102	34	6	0
0	83	27	5	0
0	0	0	0	0

Dari hasil perhitungan konvolusi tersebut, ada beberapa kondisi yang harus dipenuhi yaitu :

Jika nilai piksel (-), maka nilai tersebut dijadikan 0, jika nilai maksimal *grayscale* level yaitu dalam studi ini bernilai 252 maka dilakukan *clipping.*

Untuk piksel pinggir diabaikan atau tidak dikonvolusi, diasumsikan bernilai 0 atau konstanta yang lain.

Konvolusi piksel pinggir tidak memperlihatkan efek yang kasat mata.

Sehingga dari ketentuan tersebut, didapatkan hasil dari GaussianX sebagai berikut:

Tabel 3-7 Hasil GaussianX Citra Uji

0	0	0	0	0
0	83	27	5	0
0	102	34	6	0
0	83	27	5	0
0	0	0	0	0

Adapun perhitungan untuk Sobel Gy yaitu :

$$\text{Sobel y(1,1)} = \begin{bmatrix} 9 & 22 & 19 \\ 20 & 49 & 42 \\ 26 & 63 & 54 \end{bmatrix} \text{x} \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

$$\text{Sobel y(1,2)} = \begin{bmatrix} 22 & 19 & 25 \\ 49 & 42 & 56 \\ 63 & 54 & 73 \end{bmatrix} \text{x} \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

. . .

$$\text{Sobel y(3,3)} = \begin{bmatrix} 54 & 73 & 56 \\ 42 & 56 & 43 \\ 19 & 25 & 19 \end{bmatrix} \text{x} \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

Contoh perhitungan baris 1 kolom 1 :

Sobel Gy (1,1) = 9*1 + 22*2 + 19*1 + 26*(-1) + 63*(-2) + 54*(-1)

Sobel Gy (1,1) = -136

Tabel 3-8 Hasil Perhitungan GaussianY Citra Uji

0	0	0	0	0
0	-136	-161	-168	0
0	0	0	0	0
0	136	161	168	0
0	0	0	0	0

Kemudian di*check* kembali seperti pada ketentuan Gx, menjadi :

Tabel 3-9 Hasil GaussianY Citra Uji

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	136	161	168	0
0	0	0	0	0

Setelah diketahui semua hasil dari Gx dan Gy, maka bisa dihitung langkah berikutnya yaitu menjumlahkannya menggunakan rumus 2.11 sebagai berikut :

$$|G| = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 83 & 27 & 5 & 0 \\ 0 & 102 & 34 & 6 & 0 \\ 0 & 83 & 27 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 136 & 161 & 168 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$|G| = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 83 & 27 & 5 & 0 \\ 0 & 102 & 34 & 6 & 0 \\ 0 & 219 & 188 & 173 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Kemudian menemukan nilai tepi sobel, maka langkah selanjutnya adalah menentukan arah tepi dengan menggunakan rumus $\theta = \boldsymbol{arctan}\left(\frac{Gy}{Gx}\right)$, pada contoh perhitungan berikut menggunakan *GaussianX[1,1]* dan *GaussianY[1,1*] sebagai berikut :

$$\theta = \tan^{-1}\left(\frac{0}{0}\right)$$

$$\theta = \tan^{-1}0$$

$$\theta = 0\,, \text{radian} = 0°$$

Akhirnya dapat dilakukan *Thresholding,* yang merupakan proses pemisahan pikse-piksel berdasrkan derajat kebuan lebih kecil dari nilai batas yang dimilikinya. Pada perhitungan penelitian ini menggunakan piksel dengan ketentuan 20 sampai 80 sebagai batas penentuan matriks menjadi 1, selain *range* tersebut matriks diubah menjadi 0. [8]

Metode ini digunakan untuk mengkonversi data image menjadi data biner dengan tujuan agar proses selanjutnya menjadi lebih mudah. Maka proses awal *thresholding* pada matriks dengan dua buah nilai *tresholding* nilai 0 dan 1, sebagai berikut :

Tabel 3-10 Hasil Thresholding Citra Uji

0	0	0	0	0
0	1	1	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Tabel 3-11 Hasil Thresholding Latih 1

0	0	0	0	0
0	1	1	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Tabel 3-12 Hasil Thresholding Latih 2

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0
0	0	0	0	0

Tabel 3-13 Hasil Thresholding Latih 3

0	0	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Hasil citra dari deteksi tepi canny dapat dilihat pada gambar berikut :



Gambar 3.11 Citra Hasil Deteksi Tepi Citra Uji

3.8 Analisis Proses Pencocokan

Matching Process digunakan untuk menghitung jarak antar tepi*(edge)*. Dimana dalam tahapan ini menentukan tingkat kemiripan dari tepi citra uji dan tepi citra latih. Semakin kecil jaraknya maka citra tersebut semakin mirip dengan citra uji. Pada penelitian ini, proses pencocokkan menggunakan perhitungan tepi. Perhitungan ini diimplementasikan dengan persamaan euclidean sebagai berikut (2.13)

$$d = \sqrt{(j_1 - j_2)^2 + (i_1 - i_2)^2} \text{.....(2.13)}$$

Dengan :

= Jarak dari dua citra *Grayscale* dengan perhitungan *Chamfer*.

j_1 = data koordinat ke-j_1 pada tepi (*Edge*) citra uji.

j_2 = data koordinat ke-j_2 pada tepi (*Edge*) citra yang ada di latih.

i_1 = data koordinat ke-i_1 pada tepi (*Edge*) citra uji.

i_2 = data koordinat ke-i_2 pada tepi (*Edge*) citra yang ada di latih.

Pengenalan diperoleh dengan menghitung jarak terdekat, yaitu nilai jarak yang paling kecil. Pada tiap nilai *edge (i, j)* yang dideteksi pada piksel citra uji maupun citra latih dilakukan pengurangan dan dikuadratkan, setelah itu hasil dari masing-masing nilai *i* dan *j* akan dilakukan penambahan, kemudian dilakukan penjumlahan dari hasil tiap-tiap nilai. Perhitungan terpusat pada citra uji dari awal koordinat, jadi jika pada citra latih tidak ditemukan nilai yang sama maka akan diganti dengan nilai yang terdekat.

Cara untuk mencocokkan titik *edge* dari citra uji dan citra latih adalah sebagai berikut, diketahui nilai dari citra biner dari citra latih yaitu :

Citra Biner Dari Citra Uji

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

Citra biner dari citra uji yang sudah dihitung masukan ke dalam table diubah menjadi data *sequance*. Dan diketahui bahwa citra uji yang masukan adalah lebih dari satu, karena digunakan untuk membandingkan dengan citra uji yang digunakan. Citra uji hanya 1, jadi alurnya adalah ketika sudah terdapat tabel *sequance* dari citra latih. Kemudian lakukan pengujian memasukan 1 citra latih, proses yang digunakan adalah sama yaitu grayscaling sampai ke deteksi tepi.

Setelah citra uji terbentuk menjadi angka *binary*, kemudian dijadikan data *sequance* juga karena akan dicocokan dengan data *sequance* citra latih yang lainnya. Dan dalam analisis metode ini akan ditunjukan pencocokan salah satu citra uji terhadap citra latih yang sudah dijelaskan pada sub bab sebelumnya. Berikut merupakan contoh matriks sampai pada tahapan deteksi tepi yang akan diuji untuk menentukan kecocokan citra dengan citra latih :

Biner Citra Latih 1

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

Biner Citra Latih 2

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

Biner Citra Latih 3

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

Pemilihan data matching adalah sebagai berikut :

Tabel 3-14 Perhitungan Euclidean Distance Citra Uji dengan Citra Latih 1

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 1)	$\lvert j1^2 - j2^2 \rvert$
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	1	0
8	1	1	0

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 1)	$\lvert j1^2 - j2^2 \rvert$
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	1	1	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
		Total	0
		Nilai Euclidean distance $\sqrt{\lvert j1^2 - j2^2 \rvert}$	0

Tabel 3-15 Perhitungan Euclidean Citra Uji dengan Citra Latih 2

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 2)	$\lvert j1^2 - j2^2 \rvert$
1	0	0	0

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 2)	$\lvert j1^2 - j2^2 \rvert$
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	1	0
8	1	1	0
9	0	1	1
10	0	0	0
11	0	0	0
12	0	1	1
13	1	1	0
14	0	1	1
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 2)	$\lvert j1^2 - j2^2 \rvert$
		Total	3
		Nilai Euclidean distance $\sqrt{\lvert j1^2 - j2^2 \rvert}$	1,27

Tabel 3-16 Perhitungan Euclidean Citra Uji dengan Citra Latih 3

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 3)	$\lvert j1^2 - j2^2 \rvert$
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	0	1
8	1	1	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	1	1	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0

Titik	j1 (Sequence Citra Uji)	j1 (Sequence Citra Latih 3)	$\lvert j1^2 - j2^2 \rvert$
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
		Total	1
		Nilai Euclidean distance $\sqrt{\lvert j1^2 - j2^2 \rvert}$	1

Tabel 3-17 Hasil Perhitungan Euclidean Distance Citra Uji dengan Citra Latih

Citra Latih	Nilai Euclidean Distance
1	0
2	1,27
3	1

Setelah diketahui nilai euclidean dari perbandingan Citra uji dengan citra latih, langkah berikutnya mencari citra yang menyerupai dengan cara metode chamfer matching dimana nilai yang paling terkecil merupakan citra yang menyerupai. Perhitungan ini diimplementasikan dengan persamaan sebagai berikut :

$$n \sum_{i=1} \min(\|p_i,q_j\|,\|p_i,q_j\|,\cdots\|p_i,q_m\|) \qquad (2.14)$$

Dengan :

p_1= titik yang ada pada citra uji.

q_j = titik yang ada pada citra latih.

$n \sum_{i=1} \min(||0||,||1,27||,||1||) = 0$

Dari data yang diperoleh, dapat dilihat bahwa citra latih 1 memiliki jarak euclidean distance yang paling kecil yaitu 0. Dengan demikian disimpulkan bahwa citra latih 1 adalah citra yang menyerupai citra uji.

3.9 Analisis Kebutuhan Sistem

Kebutuhan sistem yang diperlukan dan yang digunakan dalam pembuatan sistem ini terdiri dari beberapa bagian, yaitu: kebutuhan fungsional (Tabel 3.12) dan kebutuhan non fungsional (Tabel 3.13).

Tabel 3-18 Kebutuhan Fungsional

Nomor	Keterangan
SKPL-F-01	Sistem dapat menganalisa citra yang di-*input*-kan
SKPL-F-02	Sistem dapat memberikan informasi tentang citra yang Dianalisis

Tabel 3-19 Kebutuhan Non-Fungsionals

Nomor	Keterangan
SKPL-NF-01	Sistem dibangun dalam bentuk aplikasi *desktop.*
SKPL-NF-02	Sistem dapat dijalankan pada perangkat komputer dengan sistem operasi minimal windows 7.

3.9.1 Analisis Kebutuhan Non – Fungsional

Analisis kebutuhan nonfungsional adalah sebuah langkah dimana seorang

pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk menggunakan aplikasi yang akan dibangun. Analisis kebutuhan non-fungsional yang dilakukan dibagi dalam dua tahap, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

3.9.2 Analisis Kebutuhan Perangkat Keras

Analisis perangkat keras dimaksudkan untuk mengetahui spesifikasi perangkat keras yang digunakan oleh aplikasi ini. Spesifikasi perangkat keras yang digunakan dalam pembangunan aplikasi ini dapat dilihat pada Tabel 3-20.

Tabel 3-20 Spesifikasi Perangkat Keras Yang Digunakan Dalam Pembangunan Aplikasi

No	Nama Perangkat	Spesifikasi
1	Prosessor	Intel Core 2 duo 2.00 GHz
2	Monitor	Monitor 14 inch (1366x768)
3	Memori	RAM 3 GB DDR2
4	Harddisk	256 GB SATA 7200rpm

Sedangkan spesifikasi kebutuhan perangkat keras yang dibutuhkan untuk menjalankan aplikasi yang dibangun, yang harus dipenuhi dapat dilihat pada Tabel berikut :

Tabel 3-21 Spesifikasi Kebutuhan Perangkat Keras Yang Dibutuhkan Untuk Menjalankan Aplikasi

No	Nama Perangkat	Spesifikasi
1	Prosessor	Intel Core 2 duo 2.00 GHz
2	Monitor	Monitor 14 inch (1366x768)
3	Memori	RAM 1 GB DDR2

4	Harddisk	80 GB SATA 7200rpm

3.9.3 Analisis Kebutuhan Perangkat Lunak

Adapun spesifikasi perangkat lunak yang digunakan dalam pembangunan aplikasi *information retrieval* berbasis citra berdasarkan *screen printing* perusahaan dengan menggunakan metode *Edge Matching* ini dapat dilihat pada tabel berikut :

Tabel 3-22 Spesifikasi Perangkat Lunak Yang Digunakan Dalam Pembangunan Aplikasi

No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 7 Professional
2	Visual Studio	Visual Studio 2012

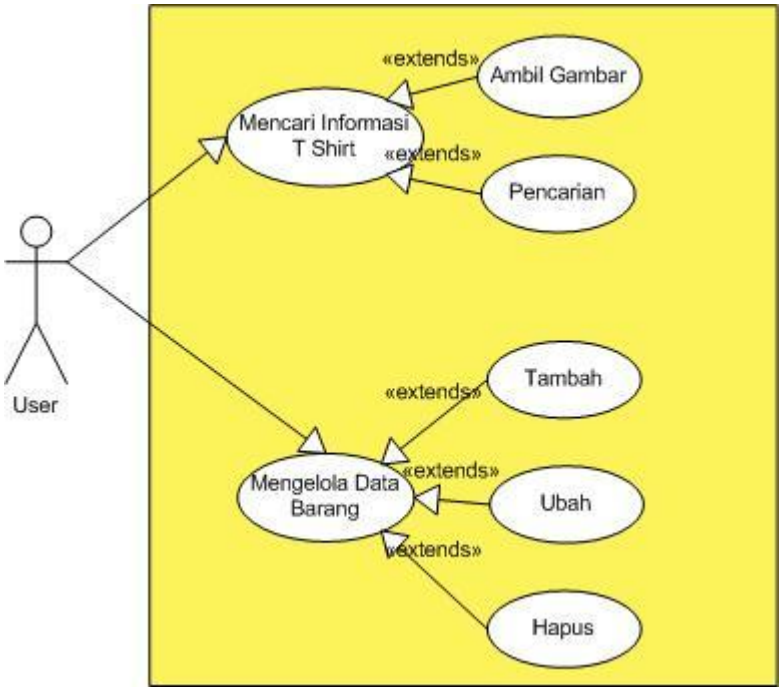
Sedangkan spesifikasi kebutuhan perangkat lunak yang dibutuhkan untuk menjalankan aplikasi yang dibangun, yang harus dipenuhi dapat dilihat pada Tabel berikut :

Tabel 3-23 Spesifikasi Kebutuhan Perangkat Lunak Yang Dibutuhkan Untuk Menjalankan Aplikasi

No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 7
2	Visual Studio	Visual Studio 2012

3.10 Diagram Use Case

Use case adalah konstruksi untuk mendeskripsikan bagaimana sistem terlihat di mata pengguna. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan.



Gambar 3.12 Gambar *Usecase*

3.11 Use Case Identification

Berikut table indentifikasi use case yang terdapat dalam aplikasi :

Tabel 3-24 Identifikasi Use Case

Use Case Indetification		
No	Use case	Deskripsi
UC-01	Mengelola Pencarian	Fungsional untuk pengolahan citra yang dicari dalam aplikasi
UC-02	Mengelola Data Barang	Fungsional yang dilakukan pengguna untuk mengolah data

Use Case Indetification		
No	Use case	Deskripsi
		kaos dan identifikasinya
UC-03	Training	Fungsionalitas untuk melakukan pelatihan terhadap citra masukan
UC-04	Testing	Fungsionalitas untuk melakukan pengujian terhadap citra hasil training.
UC-05	Menampilkan Data Pencarian	Proses mencari dan menampilkan pencarian yang dimaksud
UC-06	Mengambil Foto	Fitur yang digunakan untuk mengupload
UC-07	Menambah Data	Proses menambah data kaos
UC-08	Mengedit Data	Proses mengedit data yang sudah ada.
UC-09	Menghapus Data	Proses menghapus data yang ingin dihapus

3.12 Use Case Scenario

Skenario Use Case berfungsi untuk menjelaskan lebih detail mengenai modul-modul yang terdapat dalam aplikasi. Adapun penjelasan mengenai keterangan tabel skenario Use Case sebagai berikut :

Tabel 3-25 Skenario Use Case Mengelola Pencarian

Nama Use Case	Melakukan Pencarian
Deskripsi	Untuk proses pengolahan citra
Aktor	Pengguna
Kondisi Awal	Citra Masukan

Skenario Utama	
Aksi Aktor	Reaksi Sistem
1. Mengambil foto dengan cara upload	
2. Pengguna mengklik “ambil foto”	
	3. Aplikasi menampilkan citra hasil upload
	4. Memproses citra RGB menjadi *greyscale*
	5. Mengubah citra *greyscale* menjadi *canny* deteksi tepi
	6. Memproses citra hasil canny ke dalam *chamfer matching* dan melakukan pencocokan terhadap citra yang ada di database.
	7. Menampilkan hasil dari pencocokan
8. Mendapatkan pencarian yang dilakukan.	

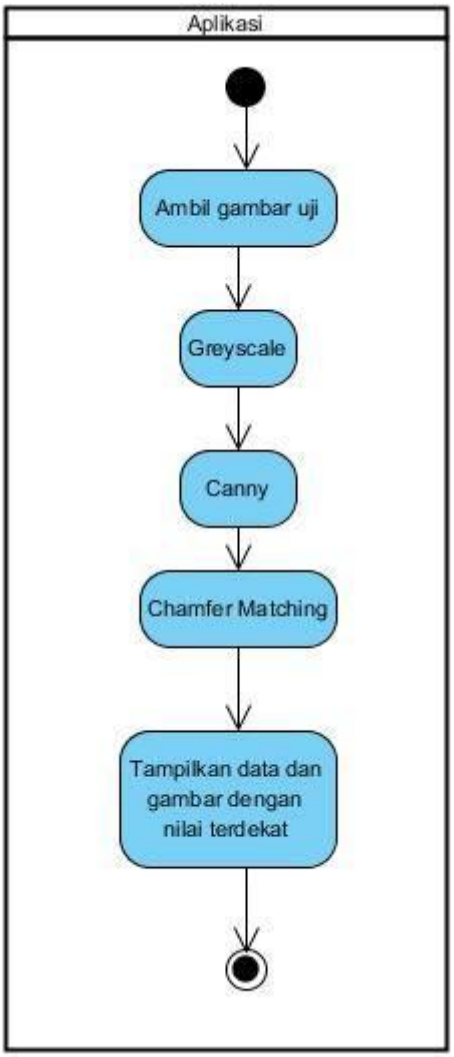
Tabel 3-26 Skenario Use Case Mengelola Data Barang

Nama Use Case	Mengelola Data Barang
Deskripsi	Untuk proses pengolahan barang
Aktor	User
Kondisi Awal	Citra Inputan Webcam
Kondisi Akhir	Aplikasi menyimpan data ke dalam database
Skenario Utama	
Aksi Aktor	Reaksi Sistem
1. Membuka menu mengelola Barang	
2. Melakukan penambahan barang.	
	3. Button upload untuk mendapatkan citra masukan.

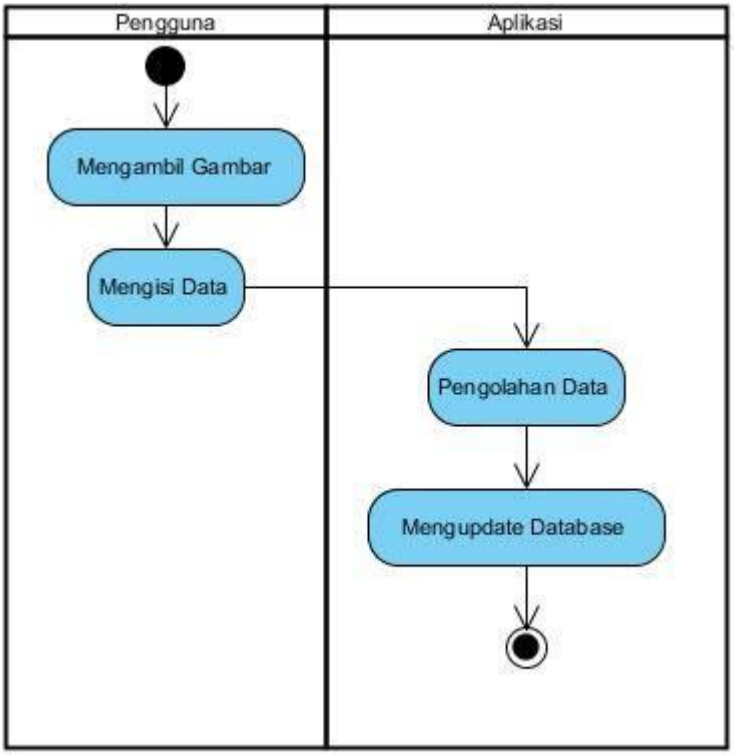
	4. Melakukan penyimpanan
5. Melakukan proses edit data, memilih data yang akan diedit	
	6. Menampilkan data yang diedit
	7. Memproses editing data
8. Melakukan proses hapus data, memilih data yang akan dihapus	
	9. Menampilkan data yang akan dihapus
	10. Memproses penghapusan data yang dimaksud

3.13 Activity Diagram

Diagram aktifitas menggambarkan berbagai alir aktivitas dalam sistem yang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. Penggambaran *activity diagram* memodelkan *event-event* yang terjadi pada *Use Case* dan digunakan untuk pemodelan aspek dinamis dari aplikasi. Diagram aktifitas dimodelkan seperti gambar sebagai berikut :



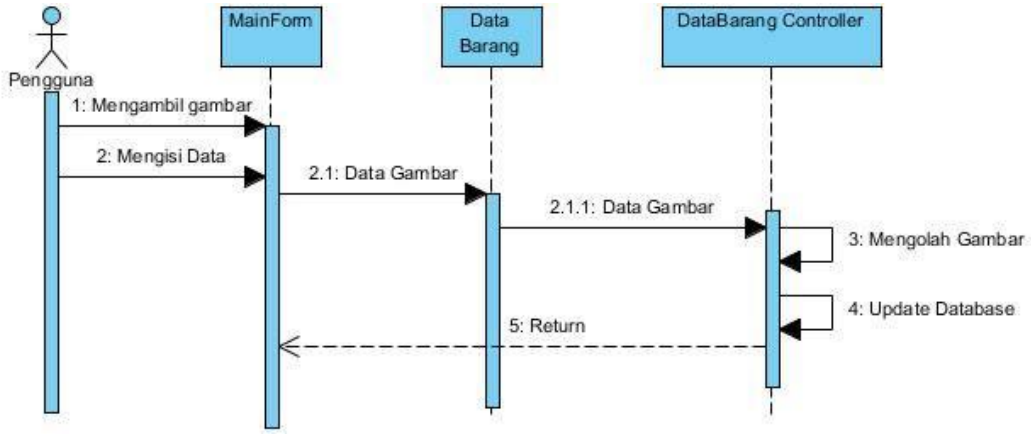
Gambar 3.13 Diagram Aktifitas Melakukan Pencarian



Gambar 3.14 Diagram Aktifitas Mengelola Data

3.14 Sequence Diagram

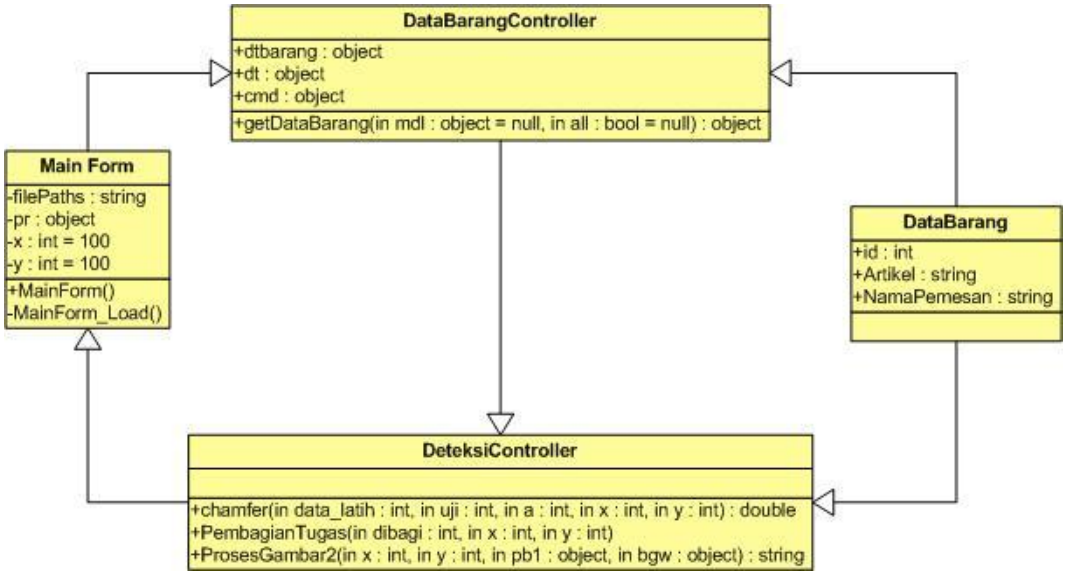
Sebuah diagram yang menggambarkan interaksi antar objek di dalam sebuah system. Interaksi tersebut berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri dari dimensi horizontal (objek-objek) dan dimensi vertikal (waktu). Dimensi vertikal adalah poros waktu, dimana waktu berjalan ke arah bawah, sedangkan dimensi horizontal merepresentasikan objek-objek individual. Tiap objek (termasuk aktor) tersebut mempunya waktu aktif yang direpresentasikan dengan kolom vertikalyang disebut dengan *lifeline.* Adapun *sequence diagram* untuk menggambarkan aplikasi dapat dilihat pada gambar sebagai berikut :



Gambar 3.15 Sequence Diagram

3.15 Class Diagram

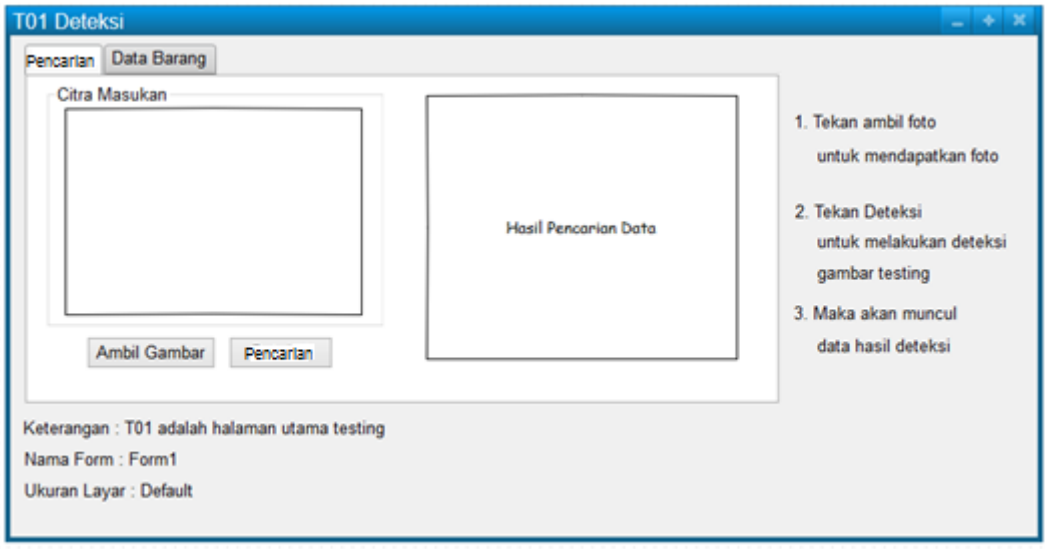
Class diagram digunakan untuk memberi gambaran suatu aplikasi. Dapat dilihat dari kelas-kelas yang ada dan cara kerja relasinya. Class diagram yang dibuat dapat dilihat pada gambar sebagai berikut :



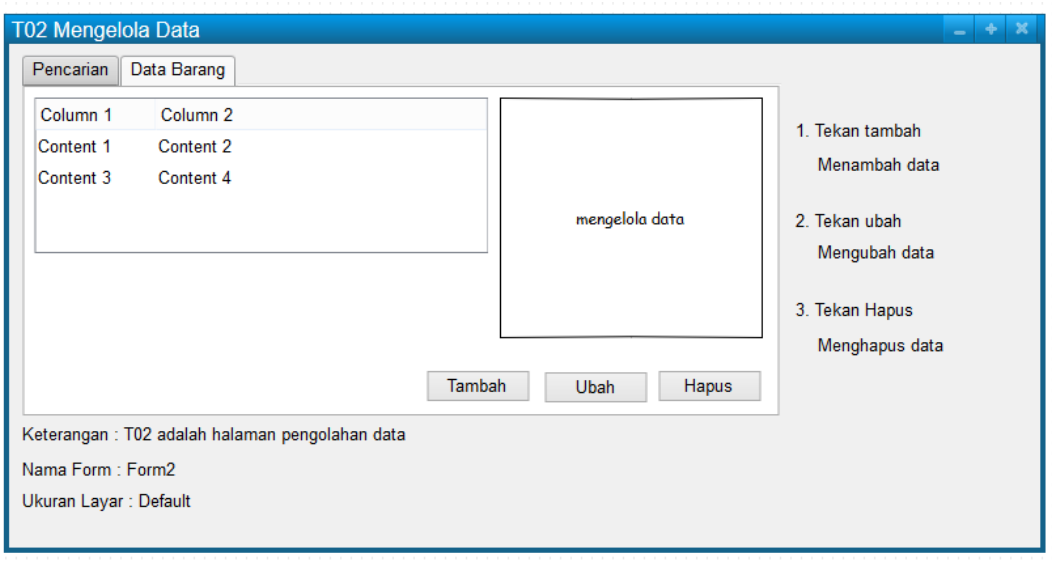
Gambar 3.16 Class Diagram

3.16 Perancangan Antarmuka

Perancangan antarmuka merupakan sebuah penggambaran, perancagnan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh dan berfungsi. Pada perancangan antarmuka penelitian ini menggunakan aplikasi “pencil”, dengan desain antar muka yang menyerupai tampilan real. Adapun perancagan antarmuka dari aplikasi yang akan dibangun terdiri dua menu yaitu tampilan menu utama pencarian dan menu mengelola data, seperti terlihat dalam gambar berikut :



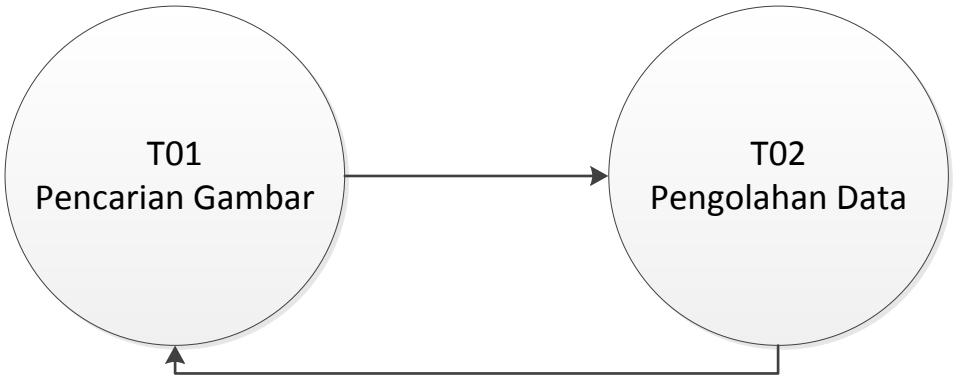
Gambar 3.17 Antarmuka T01



Gambar 3.18 Antarmuka T02

3.17 Jaringan Semantik

Jaringan semantik aplikasi yang akan dibangun dapat dilihat pada gambar dibawah ini :



Gambar 3.19 Jaringan Semantik