BAB III

ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Analisis sistem didefinisikan sebagai penguraian dari sistem utama ke dalam sub-sub sistem dengan tujuan untuk mengidentifikasi permasalahan-permasalahan yang ada dan kebutuhan-kebutuhan yang diperlukan agar dapat diusulkan dan diciptakan sistem baru yang lebih baik. Analisis terhadap sistem lama yang sedang berjalan perlu dilakukan sebagai dasar dari perancangan sistem baru, agar dapat dibuat sistem yang lebih efektif dan efisien.

III.1.1 Analisis Masalah

Tahapan analisis masalah dilakukan terlebih dahulu sebelum tahapan perancangan sistem. Hal ini dilakukan agar masalah-masalah yang dihadapi dapat diketahui dengan jelas, bagaimana penggunaan sistem yang sedang berjalan oleh pengguna, sampai solusi yang diajukan untuk permasalahan tersebut. Berdasarkan hasil penelitian, masalah-masalah yang dibahas dalam penelitian ini adalah sebagai berikut :

1. Studi kasus dilakukan di mesin pencari Google dan jejaring sosial Twitter.
2. Sumber data berasal dari hasil pencarian dokumen berdasarkan suatu kata kunci.
3. Metode yang digunakan adalah metode *Suffix Tree Clustering* (STC).
4. *Clustering* data dilakukan pada halaman hasil yang telah ditelusuri oleh mesin pencari Google dan Twitter.

III.1.2 Analisis Sumber Data

Sumber data yang digunakan dalam penelitian ini adalah data-data yang berasal dari hasil pencarian terhadap suatu kata kunci dalam mesin pencari atau jejaring sosial. Cara-cara utama untuk melakukan pencarian terhadap suatu kata kunci dalam mesin pencari Google adalah sebagai berikut :

1. Pengguna memasukkan kata kunci pencarian ke dalam antarmuka sistem.
2. Sistem kemudian akan meneruskan pencarian berdasarkan kata kunci tersebut. Sebelum pencarian dilakukan, aturan-aturan khusus untuk pencarian berdasarkan sistem yang dipilih akan diterapkan terhadap kata kunci tersebut.
3. Hasil pencarian yang didapat kemudian dikembalikan kepada pengguna.

Sedangkan dalam Twitter berlaku hal-hal sebagai berikut :

1. Pengguna memasukkan kata kunci pencarian ke dalam antarmuka sistem.
2. Sistem kemudian akan meneruskan pencarian berdasarkan kata kunci tersebut. Sebelum pencarian dilakukan, aturan-aturan khusus untuk pencarian berdasarkan sistem yang dipilih akan diterapkan terhadap kata kunci tersebut. Twitter memberikan beberapa fungsi khusus pencarian, seperti format #<kata kunci> untuk mencari *tweet* yang tergabung dalam bahasan kata kunci tersebut.
3. Hasil pencarian yang didapat kemudian dikembalikan kepada pengguna.

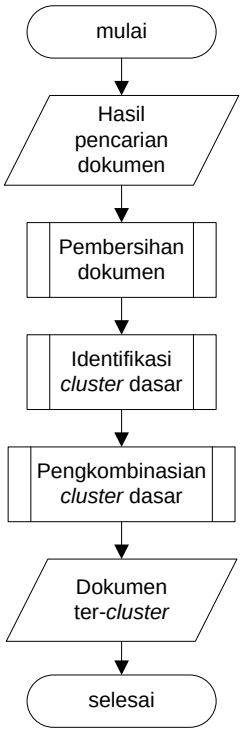
   Hasil pencarian yang diberikan dalam berupa *tweet* ataupun orang yang mengandung kata yang dicari.

Hasil pencarian pada nomor 3, baik dalam pencarian versi Google maupun Twitter, adalah data yang akan digunakan dalam penelitian ini. Data-data ini akan

diolah berdasarkan algoritma STC. Pengolahan data hasil pencarian dilakukan dengan melalui beberapa langkah dalam sub-bab 3.1.3. Keluaran dari algoritma ini adalah data hasil pencarian dan topik-topik utama yang terdapat dalam hasil pencarian tersebut. Topik-topik ini diekstrak dari dokumen-dokumen hasil pencarian.

III.1.3 Analisis Algoritma

Algoritma STC memiliki tiga proses utama, yaitu d*ocument preprocessing*, identifikasi cluster dasar pembangun menggunakan *suffix tree*, dan pengkombinasian cluster-cluster dasar ke dalam cluster yang lebih umum.



Gambar III-1 Langkah-langkah dalam algoritma STC

Penjelasan untuk langkah-langkah dalam algoritma STC adalah sebagai berikut :

III.1.3.1 *Document preprocessing*

Pembersihan dokumen merupakan langkah awal dalam algoritma STC. Dalam proses ini dokumen yang didapat berdasarkan pencarian suatu kata kunci dibersihkan dari karakter-karakter yang tidak diperlukan, proses penghapusan *stop-word*, dan proses *stemming*. Penghapusan karakter-karakter yang tidak diperlukan meliputi pembersihan tanda baca jika tanda tersebut bukan kata kunci yang dicari pengguna, penghapusan *stop-word* meliputi penghapusan kata-kata umum yang tidak mengacu kepada suatu hal, seperti kata depan, kata gabung, dan sebagainya. Sedangkan proses *stemming* adalah proses pengembalian suatu kata ke dalam bentuk awalnya, misalnya penghilangan imbuhan. Proses-proses pada langkah ini meliputi pembersihan *stop word* dan *stemming* kata. Penjelasan untuk tiap-tiap langkah di atas adalah sebagai berikut :

III.1.3.1.1 Pembersihan *stop word*

Stop word adalah kata-kata yang muncul dalam suatu pencarian, namun bukan merupakan kata-kata yang menjadi inti dari hasil pencarian tersebut. Kata-kata yang termasuk ke dalam anggota *stop word* [14] adalah sebagai berikut :

-	both	has	i've	off
a	but	hasn't	if	on
about	by	have	in	once
above	can't	haven't	into	only
after	cannot	having	is	or
again	could	he	isn't	other
against	couldn't	he'd	it	ought
all	did	he'll	it's	our
am	didn't	he's	its	ours
an	do	her	itself	ourselves
and	does	here	let's	out
any	doesn't	here's	me	over
are	doing	hers	more	own
aren't	don't	herself	most	same
as	down	him	much	shan't
at	during	himself	mustn't	she
be	each	his	my	she'd
because	few	how	myself	she'll
been	for	how's	no	she's
before	from	i	nor	should
being	further	i'd	not	shouldn't
below	had	i'll	new	so
between	hadn't	i'm	of	some

such	under	whom
than	until	why
that	up	why's
that's	very	with
the	was	won't
their	wasn't	would
theirs	we	wouldn't
them	we'd	you
themselves	we'll	you'd
then	we're	you'll
there	we've	you're
there's	were	you've
these	weren't	your
they	what	yours
they'd	what's	yourself
they'll	when	yourselves
they're	when's	
they've	where	
this	where's	
those	which	
through	while	
to	who	
too	who's	

Berikut ini adalah *pseudocode* dari prosedur penghapusan *stop word*

```
function hapus_stopWord(Input kata : String) → String
{I. S : masukan untuk pembanding antara daftar stop word terdefinisi}
{F. S : stop word tereliminasi}
Deklarasi :
Kumpulan_stopWord : array [1..100] of String {berisi dari kumpulan daftar stop word terdefinisi dengan jumlah anggota 100}
Algoritma :
If kata != Kumpulan_stopWord then
    Return kata
Endif
```

III.1.3.1.2 *Stemming* kata

Stemming dalam perangkat lunak ini mengikuti langkah-langkah dalam algoritma Porter Stemming. Dalam algoritma Porter Stemming, konsonan didefinisikan sebagai huruf selain A, E, I, O, atau U dan Y yang didahului oleh huruf selain yang disebutkan. Misalkan dalam kata TOY, konsonannya adalah T dan Y, dan dalam kata SYZYGY konsonannya adalah S, Z dan G. Dalam algoritma ini kumpulan dari konsonan dengan panjang lebih dari 0 dinotasikan dengan C, sedangkan kumpulan vokal dengan panjang lebih dari 0 dinotasikan dengan V. Oleh karena itu, setiap kata akan mengandung satu dari bentuk di bawah :

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

Bentuk di atas dapat dipresentasikan dengan :

[C] VCVC .. [V].

Dengan menggunakan komposisi (VC) {m} untuk menggambarkan VC yang diulang sebanyak m kali, maka rumus di atas dapat ditulis ulang menjadi :

[C] (VC) {m} [V]

Contoh kata dengan m yang berbeda adalah sebagai berikut :

m = 0 TR, EE, TREE, Y, BY

m = 1 TROUBLE, OATS, TREES, IVY

m = 2 TROUBLES, PRIVATE, OATEN, ORRERY

Kondisi untuk setiap kata dapat juga mengandung :

*S – kata berakhiran dengan S

v - kata mengandung satu huruf vokal

*d – kata berakhiran dengan konsonan ganda (misalkan –TT, -SS)

*o – kata berakhiran dengan pola cvc, dimana konsonan kedua bukan W, X, atau Y (misalkan –WIL, -HOP)

Selanjutnya untuk langkah-langkah dalam algoritma Porter Stemming adalah sebagai berikut :

Langkah 1a :

a. Jika kata berakhir dengan akhiran SSES, maka ubah akhiran tersebut menjadi SS. Contohnya *caresses* menjadi *caress*.
b. Jika kata berakhir dengan akhiran IES, maka ubah akhiran tersebut menjadi I. Contohnya *ponies* menjadi *poni*, *ties* menjadi *ti*.

c. Jika kata berakhir dengan akhiran SS, maka jangan ubah kata tersebut. Contohnya *caress* menjadi *caress*.
d. Jika kata berakhir dengan akhiran S, maka hilangkan akhiran S. Contohnya *cats* menjadi *cat*.

Langkah 1b :

a. Jika kata berakhiran EED, maka ubah akhiran tersebut menjadi EE. Contohnya *agreed* menjadi *agree*.
b. Jika kata berakhiran ED, maka hilangkan akhiran tersebut. Contohnya *plastered* menjadi *plaster*.
c. Jika kata berakhiran ING, maka hilangkan akhiran tersebut. Contohnya *motoring* menjadi *motor*.

Jika aturan kedua atau ketiga dalam langkah 1b terpenuhi, maka aturan berikut ini akan diterapkan :

a. Jika kata berakhiran AT, maka ubah akhiran tersebut menjadi ATE. Contohnya *conflat(ed)* menjadi *conflate*.
b. Jika kata berakhiran BL, maka ubah akhiran tersebut menjadi BLE. Contohnya *troubl(ed)* menjadi *trouble.*
c. Jika kata berakhiran IZ, maka ubah akhiran tersebut menjadi IZE. Contohnya *siz(ed)* menjadi *size.*
d. Jika kata berakhiran dengan konsonan ganda (misalnya –NN) dan bukan berakhiran L, S, atau Z, maka ubah konsonan ganda tersebut menjadi 1 karakter saja. Contohnya *hopp(ing)* menjadi *hop*, *tann(ed)* menjadi *tan*, *fall(ing)* menjadi *fall.*

e. Jika kata berakhiran dengan pola konsonan – vokal – konsonan (misalnya – WIL, -HOP), dan konsonan kedua bukan W, X, atau Y, maka ubah akhiran tersebut menjadi E. Contohnya *fil(ing)* menjadi *file*.

Langkah 1c :

a. Jika kata hanya memiliki 1 huruf vokal dan berakhiran Y, maka ubah akhiran tersebut menjadi I. Contohnya *happy* menjadi *happi.*

Langkah 2 :

a. Jika kata berakhiran dengan ATIONAL, maka ubah akhiran tersebut menjadi ATE. Contohnya *relational* menjadi *relate*.
b. Jika kata berakhiran dengan TIONAL, maka ubah akhiran tersebut menjadi TION. Contohnya *conditional* menjadi *condition*.
c. Jika kata berakhiran dengan ENCI, maka ubah akhiran tersebut menjadi ENCE. Contohnya *valenci* menjadi *valence*.
d. Jika kata berakhiran dengan ANCI, maka ubah akhiran tersebut menjadi ANCE. Contohnya *hesitanci* menjadi *hesitance*.
e. Jika kata berakhiran dengan IZER, maka ubah akhiran tersebut menjadi IZE. Contohnya *digitizer* menjadi *digitize*.
f. Jika kata berakhiran dengan ABLI, maka ubah akhiran tersebut menjadi ABLE. Contohnya *conformabli* menjadi *conformable*.
g. Jika kata berakhiran dengan ALLI, maka ubah akhiran tersebut menjadi AL. Contohnya *radicalli* menjadi *radical*.
h. Jika kata berakhiran dengan ENTLI, maka ubah akhiran tersebut menjadi ENT. Contohnya *differentli* menjadi *different*.

i. Jika kata berakhiran dengan ELI, maka ubah akhiran tersebut menjadi E. Contohnya *vileli* menjadi *vile*.

j. Jika kata berakhiran dengan OUSLI, maka ubah akhiran tersebut menjadi OUS. Contohnya *analogousli* menjadi *analogous*.

k. Jika kata berakhiran dengan IZATION, maka ubah akhiran tersebut menjadi IZE. Contohnya *vietnamization* menjadi *vietnamize*.

l. Jika kata berakhiran dengan ATION, maka ubah akhiran tersebut menjadi ATE. Contohnya *predication* menjadi *predicate*.

m. Jika kata berakhiran dengan ATOR, maka ubah akhiran tersebut menjadi ATE. Contohnya *operator* menjadi *operate*.

n. Jika kata berakhiran dengan ALISM, maka ubah akhiran tersebut menjadi AL. Contohnya *feudalism* menjadi *feudal*.

o. Jika kata berakhiran dengan IVENESS, maka ubah akhiran tersebut menjadi IVE. Contohnya *decisiveness* menjadi *decisive*.

p. Jika kata berakhiran dengan FULNESS, maka ubah akhiran tersebut menjadi FUL. Contohnya *hopefulness* menjadi *hopeful*.

q. Jika kata berakhiran dengan OUSNES, maka ubah akhiran tersebut menjadi OUS. Contohnya *callousness* menjadi *callous*.

r. Jika kata berakhiran dengan ALITI, maka ubah akhiran tersebut menjadi AL. Contohnya *formaliti* menjadi *formal*.

s. Jika kata berakhiran dengan IVITI, maka ubah akhiran tersebut menjadi IVE. Contohnya *sensitiviti* menjadi *sensitive*.

t. Jika kata berakhiran dengan BILITI, maka ubah akhiran tersebut menjadi BLE. Contohnya *sensibiliti* menjadi *sensible*.

Langkah 3 :

a. Jika kata berakhiran dengan ICATE, maka ubah akhiran tersebut menjadi IC. Contohnya *triplicate* menjadi *triplic*.
b. Jika kata berakhiran dengan ATIVE, maka hapus akhiran tersebut. Contohnya *formative* menjadi *form*.
c. Jika kata berakhiran dengan ALIZE, maka ubah akhiran tersebut menjadi AL. Contohnya *formalize* menjadi *formal*.
d. Jika kata berakhiran dengan ICITI, maka ubah akhiran tersebut menjadi IC. Contohnya *electriciti* menjadi *electric*.
e. Jika kata berakhiran dengan ICAL, maka ubah akhiran tersebut menjadi IC. Contohnya *electrical* menjadi *electric*.
f. Jika kata berakhiran dengan FUL, maka hapus akhiran tersebut. Contohnya *hopeful* menjadi *hope*.
g. Jika kata berakhiran dengan NESS, maka hapus akhiran tersebut. Contohnya *goodness* menjadi *good*.

Langkah 4 :

a. Jika kata berakhiran dengan AL, maka hapus akhiran tersebut. Contohnya *revival* menjadi *reviv*.
b. Jika kata berakhiran dengan ANCE, maka hapus akhiran tersebut. Contohnya *allowance* menjadi *allow*.

c. Jika kata berakhiran dengan ENCE, maka hapus akhiran tersebut. Contohnya *inference* menjadi *infer*.

d. Jika kata berakhiran dengan ER, maka hapus akhiran tersebut. Contohnya *airliner* menjadi *airlin*.

e. Jika kata berakhiran dengan IC, maka hapus akhiran tersebut. Contohnya *gyroscopic* menjadi *gyroscop*.

f. Jika kata berakhiran dengan ABLE, maka hapus akhiran tersebut. Contohnya *adjustable* menjadi *adjust*.

g. Jika kata berakhiran dengan IBLE, maka hapus akhiran tersebut. Contohnya *defensible* menjadi *defens*.

h. Jika kata berakhiran dengan ANT, maka hapus akhiran tersebut. Contohnya *irritant* menjadi *irrit*.

i. Jika kata berakhiran dengan EMENT, maka hapus akhiran tersebut. Contohnya *replacement* menjadi *replac*.

j. Jika kata berakhiran dengan MENT, maka hapus akhiran tersebut. Contohnya *adjustment* menjadi *adjust*.

k. Jika kata berakhiran dengan ENT, maka hapus akhiran tersebut. Contohnya *dependent* menjadi *depend*.

l. Jika kata berakhiran dengan S atau T lalu diikuti ION, maka hapus akhiran tersebut. Contohnya *adoption* menjadi *adopt*.

m. Jika kata berakhiran dengan OU, maka hapus akhiran tersebut. Contohnya *homologou* menjadi *homolog*.

n. Jika kata berakhiran dengan ISM, maka hapus akhiran tersebut. Contohnya *communism* menjadi *commun*.
o. Jika kata berakhiran dengan ATE, maka hapus akhiran tersebut. Contohnya *activate* menjadi *activ*.
p. Jika kata berakhiran dengan ITI, maka hapus akhiran tersebut. Contohnya *angulariti* menjadi *angular*.
q. Jika kata berakhiran dengan OUS, maka hapus akhiran tersebut. Contohnya *homologous* menjadi *homolog*.
r. Jika kata berakhiran dengan IVE, maka hapus akhiran tersebut. Contohnya *effective* menjadi *effect*.
s. Jika kata berakhiran dengan IZE, maka hapus akhiran tersebut. Contohnya *bowdlerize* menjadi *bowdler*.

Langkah 5a :

a. Jika kata berakhiran dengan E, maka hapus akhiran tersebut. Contohnya *probate* menjadi *probat*.
b. Jika kata berakhiran dengan pola konsonan - vokal – konsonan dan diikuti E, maka hapus akhiran tersebut. Contohnya *cease* menjadi *ceas.*

Langkah 5b :

a. Jika kata berakhiran dengan konsonan ganda lalu diikuti dengan L, maka hapus akhiran tersebut. Contohnya *controll* menjadi *control.*

Berikut ini adalah *pseudocode* dari prosedur *stemming* kata.

```
function stemming(Input kata : String) → String
{I.S : kata sebagai masukan merupakan hasil dari proses
penghapusan stop word}
{F. S : kata kembali dalam bentuk dasar}
Deklarasi:
{langkah 1-5 dalam proses stemming menggunakan algoritma porter
stemming terdefinisi}
Algoritma :
If length(kata) <= 2 then
    {length digunakan untuk mengetahui jumlah huruf dari kata}
    Return kata
Endif
If kata mengandung tanda baca then
    Hapus tanda baca dari kata
    Return kata
Endif
If kata mengandung bagian dari langkah 1a then
    Hapus tiap bagian langkah 1a dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 1b then
    Hapus tiap bagian langkah 1b dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 1c then
    Hapus tiap bagian langkah 1c dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 2 then
    Hapus tiap bagian langkah 2 dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 3 then
    Hapus tiap bagian langkah 3 dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 4 then
    Hapus tiap bagian langkah 4 dari kata
    Return kata
endif
If kata mengandung bagian dari langkah 5a then
    Hapus tiap bagian langkah 5a dari kata
    Return kata
If kata mengandung bagian dari langkah 5b then
    Hapus tiap bagian langkah 5a dari kata
    Return kata
endif
```

III.1.3.2 Identifikasi *Cluster* Dasar Pembangun

Tahap kedua dari algoritma STC adalah tahap identifikasi pembentukan *cluster* dasar. Pembentukan *cluster* dasar dilakukan dengan cara menemukan kesamaan frasa-frasa yang ditemukan dalam dokumen-dokumen yang diteliti dengan menggunakan struktur data *suffix tree*. Dengan cara ini maka setiap dokumen dapat direpresentasikan menjadi suatu kalimat. Setiap *cluster* dasar yang dibentuk memiliki suatu *score*. Penghitungan *score* tersebut berdasarkan jumlah dokumen yang masuk ke dalam anggota *cluster* dasar dan jumlah kata yang menyusun frasa dari *cluster* dasar. Fungsi untuk menghitung *score cluster* dasar ditunjukkan oleh persamaan (3.1).

$$S(B) = |B|.f(|P|) \tag{3.1}$$

Dimana :

S(B) = *score cluster*,

|B| = jumlah dokumen yang memiliki frasa yang muncul di dalam *cluster* dasar, dan

f(|P|) = jumlah kata yang menyusun frasa tersebut.

Contoh perhitungannya adalah sebagai berikut :

Misalkan pencarian dilakukan terhadap kata *school holiday* [Lampiran C]. Hasil pencarian terhadap kata ini menunjuk kepada beberapa dokumen dan kata-kata lain selain masukan di atas, diantaranya berupa :

Tabel III-1 Hasil pencarian dengan kata kunci *cat eat cheese*

Kata yang dihasilkan (cluster dasar)	Muncul dalam dokumen ke-

Kata yang dihasilkan (cluster dasar)	Muncul dalam dokumen ke-
School	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Holidai	1, 2, 3, 4, 5, 6, 7, 9, 10
Term	2, 4, 6, 7, 8, 10
Public	6, 7, 9, 10
Date	1, 4, 8, 10
2012	3, 7, 9
State	2, 4, 6
Govern	4, 7, 8

Maka *score* untuk kata yang dihasilkan dari pencarian adalah sebagai berikut :

Tabel III-2 Penghitungan *score cluster*

Kata yang dihasilkan	|B| (jumlah dokumen)	f(|P|) (jumlah kata penyusun frasa)	S(B) (*score cluster)*
School	10	1	10
Holidai	9	1	9
Term	6	1	6
Public	4	1	4
Date	4	1	4
2012	3	1	3
State	3	1	3
Govern	3	1	3

Dalam perangkat lunak yang dibangun, *cluster* yang ditampilkan hanyalah *cluster* yang memiliki *score* lebih dari sama dengan tiga, dengan asumsi bahwa *score* yang lebih sedikit dihasilkan oleh *outlier* yang tidak memiliki arti khusus bagi pencarian.

Berikut ini adalah *pseudocode* dari prosedur identifikasi *cluster* dasar.

```
Procedure cluster(IN kata : String) → String
{I.S : kata sebagai masukan merupakan hasil dari proses stemming}
{F. S : skor kata didapat}
Deklarasi :
i : integer
Algoritma :
For i ← 1 to count(kata) do
    Simpan dan hitung kemunculan kata ke dalam database
Endfor
```

III.1.3.3 Pengkombinasian *Cluster* Dasar

Tahap ini dilakukan unutk menangani *cluster-cluster* yang *overlap*. Pengkombinasian dapat dilakukan dengan menghitung nilai *similarity* antar *cluster* dasar yang didasarkan pada jumlah dokumen yang *overlap*. *Overlapping* dokumen didasarkan karena dokumen dapat memiliki lebih dari satu topik.

Penghitungan nilai *similarity* menggunakan nilai biner. Rumus untuk menghitung nilai *similarity* antar *cluster* dasar dapat dilihat pada persamaan (3.2) dan (3.3).

$$\frac{|Bm \cap Bn|}{|Bm|} > 0{,}5 \tag{3.2}$$

$$\frac{|Bm \cap Bn|}{|Bn|} > 0{,}5 \tag{3.3}$$

Dimana :

|Bm ∩ Bn| = jumlah dokumen yang mengandung kedua kata <Bm> dan <Bn>.

|Bm| = jumlah dokumen yang mengandung kata <Bm>.

|Bn| = jumlah dokumen yang mengandung kata <Bn>.

Dalam penggunaan di atas, penggunaan nilai ambang 0,5 digunakan karena nilai tersebut merupakan nilai tengah antara 0 sampai 1. Jika persamaan (3.2) dan (3.3) bernilai benar (lebih dari 0,5) maka *similarity* antar dua *cluster* dasar akan bernilai satu sehingga kedua *cluster* tersebut akan terhubung. Jika hanya salah satu dari persamaan (3.2) dan (3.3) bernilai benar atau keduanya salah maka nilai

similarity akan bernilai 0 sehingga antara kedua *cluster* dasar tersebut tidak terhubung.

Contoh perhitungannya adalah sebagai berikut :

1. Kata *school* dan *holidai* berdasarkan Tabel III-1 terdapat dalam dokumen no 1, 2, 3, 4, 5, 6, 7, 9, 10, maka untuk *school* dan *holidai* :

   |Bm ∩ Bn| = jumlah dokumen yang mengandung kata *school* dan *holidai* = 9 dokumen,

   |Bm| = jumlah dokumen yang hanya mengandung kata *school* = 10 dokumen,

   |Bn| = jumlah dokumen yang hanya mengandung kata *holidai* = 9 dokumen,

   Maka nilai *similarity* :

$$\frac{|Bm \cap Bn|}{|Bm|} = \frac{9}{10} = 0{,}9, \qquad 0{,}9 > 0{,}5$$

$$\frac{|Bm \cap Bn|}{|Bn|} = \frac{9}{9} = 1, \qquad 1 > 0{,}5$$

   Berdasarkan perhitungan di atas, nilai *similarity* antara *school* dan *holidai* masing-masing adalah 0,9 dan 1. Dengan batas nilai ambang 0,5, maka dapat dilihat bahwa antara kata *school* dan *holidai* terhubung dan kedua anggota *cluster* akan digabungkan.

2. Kata *school* dan *govern* berdasarkan Tabel III-1 dihubungkan oleh dokumen no 4, 7, 8, maka untuk *school* dan *govern* :

   |Bm ∩ Bn| = jumlah dokumen yang mengandung kata *school* dan *govern* = 3 dokumen,

   |Bm| = jumlah dokumen yang hanya mengandung kata *school* = 10,

|Bn| = jumlah dokumen yang hanya mengandung kata *govern* = 3,

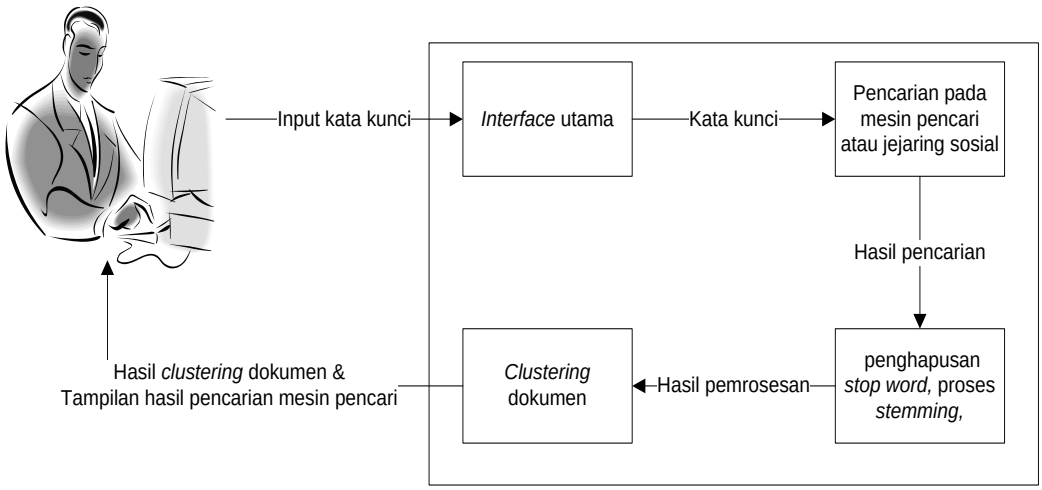
$$\frac{|Bm \cap Bn|}{|Bm|} = \frac{3}{10} = 0{,}3, \qquad 0{,}3 \leq 0{,}5$$

$$\frac{|Bm \cap Bn|}{|Bn|} = \frac{3}{3} = 1, \qquad 1 > 0{,}5$$

Berdasarkan perhitungan di atas, nilai *similarity* antara *school* dan *govern* masing-masing adalah 0,3 dan 1. Dengan batas nilai ambang 0,5, maka dapat dilihat bahwa antara kata *school* dan *govern* tidak terhubung.

Namun, karena perangkat lunak yang dibangun hanya bekerja dengan set dokumen yang sangat kecil, yang berjumlah antara 10 sampai dengan 15, maka langkah pengkombinasian *cluster* dasar tidak dilakukan, dengan asumsi bahwa *cluster* dasar yang terbentuk sudah cukup mewakili inti dari dokumen hasil pencarian.

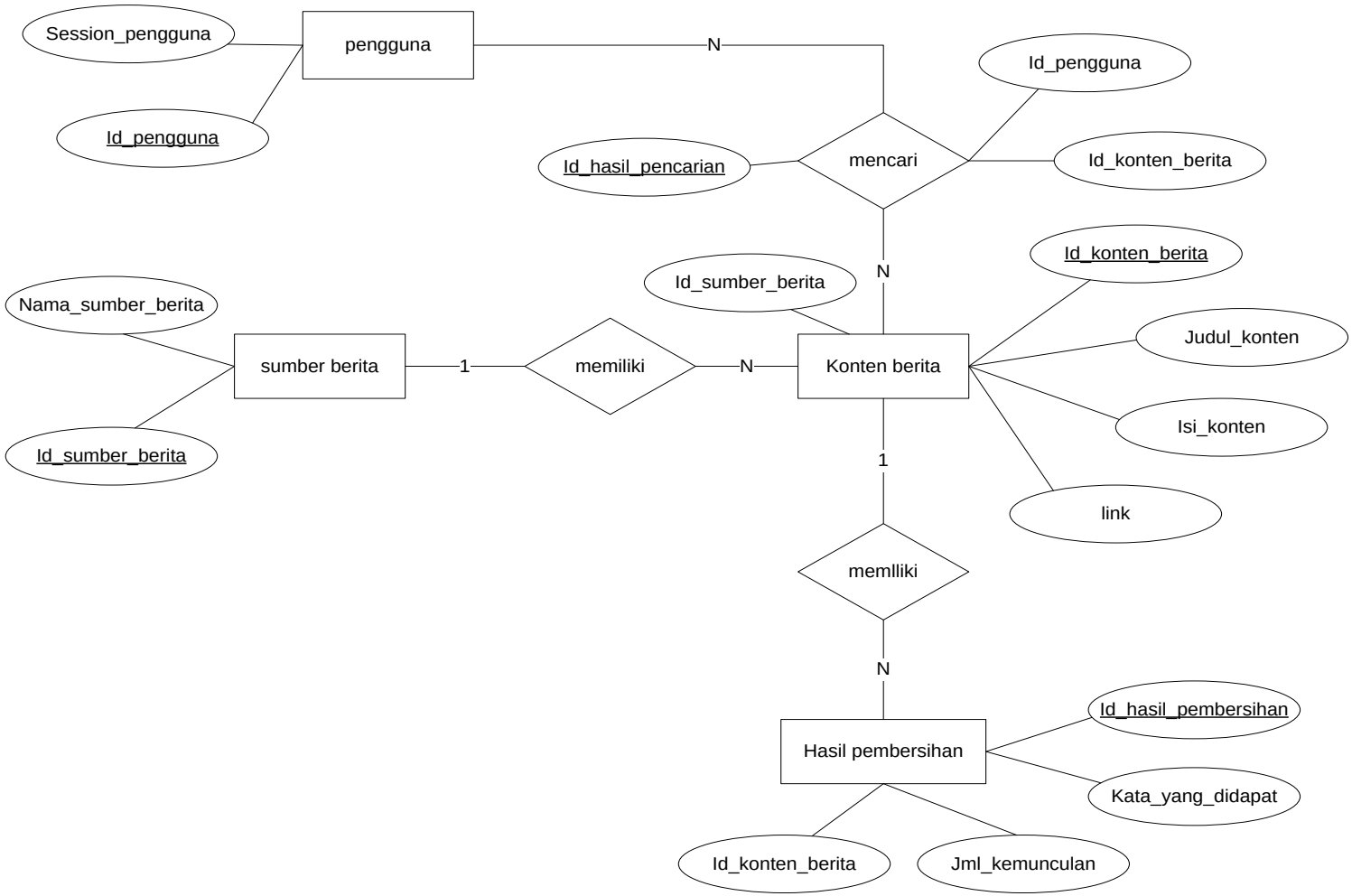
Deskripsi umum sistem yang akan dibangun dapat dilihat pada Gambar 3.2 di bawah ini.



Gambar III-2 Deskripsi umum sistem

III.1.4 Analisis Data

Pemodelan entitas dan komponen data dan hubungan masing-masing entitas tersebut dalam penelitian ini menggunakan *Entity Relationship Diagram* (ERD). ERD untuk penelitian ini adalah sebagai berikut :



Gambar III-3 ERD perangkat lunak yang dibangun

Adapun kamus data untuk ERD di atas adalah sebagai berikut :

Tabel III-3 Kamus data

No	Entitas	Atribut
1	Sumber_berita	a. Id_sumber_berita b. Nama_sumber_berita
2	Konten_berita	a. Id_konten_berita b. Id_sumber_berita c. Judul_konten d. Isi_konten e. link
3	Hasil_pembersihan	a. Id_hasil_konten b. Id_konten_berita c. Kata_yang didapat d. Jml_kemunculan
4	Pengguna	a. Id_pengguna b. Session_pengguna
5	Hasil_pencarian	a. Id_hasil_pencarian b. Id _pengguna c. Id_konten_berita

III.1.5 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak yang akan dibangun berdasarkan pada kebutuhan pengguna pada jurnal dan artikel yang ada. Spesifikasi kebutuhan perangkat lunak akan dibagi ke dalam dua bagian, yaitu SKPL-F (spesifikasi kebutuhan perangkat lunak fungsional) dan SKPL-NF (spesifikasi kebutuhan perangkat lunak non-fungsional). Berikut ini adalah tabel dari spesifikasi kebutuhan perangkat lunak dalam penelitian ini.

Tabel III-4 Spesifikasi kebutuhan perangkat lunak fungsional

Kode	Kebutuhan
SKPL-F001	Sistem harus dapat mengambil dokumen hasil pencarian yang diberikan mesin pencari Google dan Twitter
SKPL-F002	Sistem harus dapat melakukan pembersihan elemen-elemen html dari hasil pencarian
SKPL-F003	Sistem harus dapat melakukan penghapusan *stop-words* dari hasil pencarian
SKPL-F004	Sistem harus dapat melakukan *stemming* kata terhadap hasil

Kode	Kebutuhan
	pencarian
SKPL-F005	Sistem harus dapat melakukan *clustering* hasil pencarian

Spesifikasi kebutuhan perangkat lunak non-fungsional untuk perangkat lunak yang dibangun adalah sebagai berikut :

Tabel III-5 Spesifikasi kebutuhan perangkat lunak non-fungsional

Kode	Kebutuhan
SKPL-NF001	Perangkat lunak yang dibangun berbasis web
SKPL-NF002	Wajib tersedia jaringan internet untuk menggunakan perangkat lunak
SKPL-NF003	Prosesor yang digunakan minimal Pentium 4 atau setara
SKPL-NF004	Memori komputer yang tersedia minimal 256 Mb
SKPL-NF005	Sistem operasi yang digunakan adalah Microsoft Windows XP
SKPL-NF006	Pembangunan perangkat lunak menggunakan Adobe Dreamweaver CS3
SKPL-NF007	Pengguna diharapkan mengerti cara menggunakan komputer, internet, serta *browser* untuk menjelajahi internet

III.1.6 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan fungsional digunakan untuk memastikan bahwa perangkat lunak yang dibuat sesuai dengan kebutuhan pengguna dan layak untuk dikembangkan. Tahapan analisis kebutuhan fungsional dibagi menjadi beberapa tahap yaitu analisis kebutuhan perangkat keras, perangkat lunak, dan perangkat pikir. Tahapan-tahapan ini diperlukan agar perangkat lunak yang dibangun dapat berhasil secara keseluruhan.

III.1.6.1 Analisis Kebutuhan Perangkat Keras

Perangkat lunak yang dibangun adalah perangkat lunak yang akan menjadi perangkat lunak publik. Oleh sebab itu, spesifikasi umum perangkat lunak agar dapat menjalankan aplikasi ini harus memenuhi kriteria minimal sebagai berikut:

1. Prosesor minimal pentium 4 atau setara.
2. Memori komputer minimal 256 Mb.

III.1.6.2 Analisis Kebutuhan Perangkat Lunak

Spesifikasi minimal perangkat lunak yang dapat menjalankan aplikasi ini adalah sistem operasi Microsoft Windows XP dengan pembangunan perangkat lunak menggunakan Adobe Dreamweaver CS3.

III.1.6.3 Analisis Kebutuhan Perangkat Pikir

Aplikasi ini diarahkan untuk menjadi aplikasi yang akan digunakan oleh orang-orang yang harus memenuhi kriteria sebagai berikut:

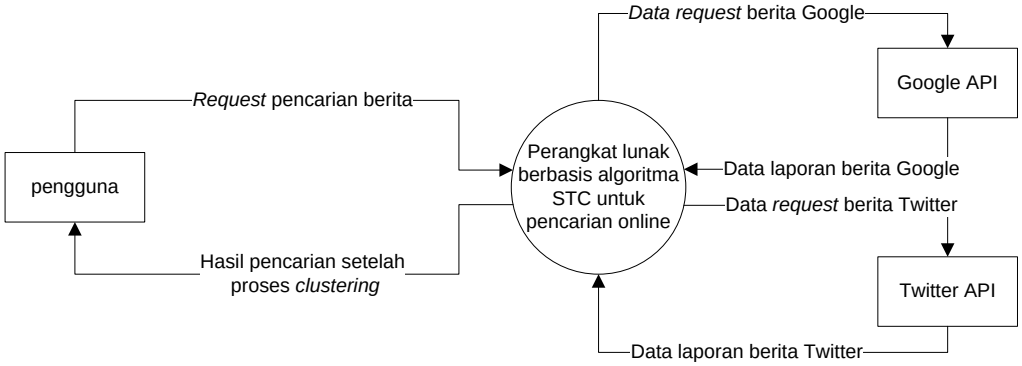
1. Mengerti cara mengoperasikan komputer.
2. Mengerti cara menggunakan internet.
3. Mengerti cara menggunakan *browser*.
4. Berusia minimal 16 tahun.
5. Memahami cara melakukan pencarian dalam Google dan Twitter.

III.1.7 Analisis Kebutuhan Fungsional

Kebutuhan fungsional dianalisis dengan memodelkan sistem. Pemodelan yang digunakan untuk pembangunan perangkat lunak ini adalah pemodelan analisis terstruktur menggunakan *Data Flow Diagram* (DFD), sedangkan alat bantu pemodelan perangkat lunak yang digunakan adalah Microsoft Office Visio 2007.

III.1.7.1 Diagram Konteks

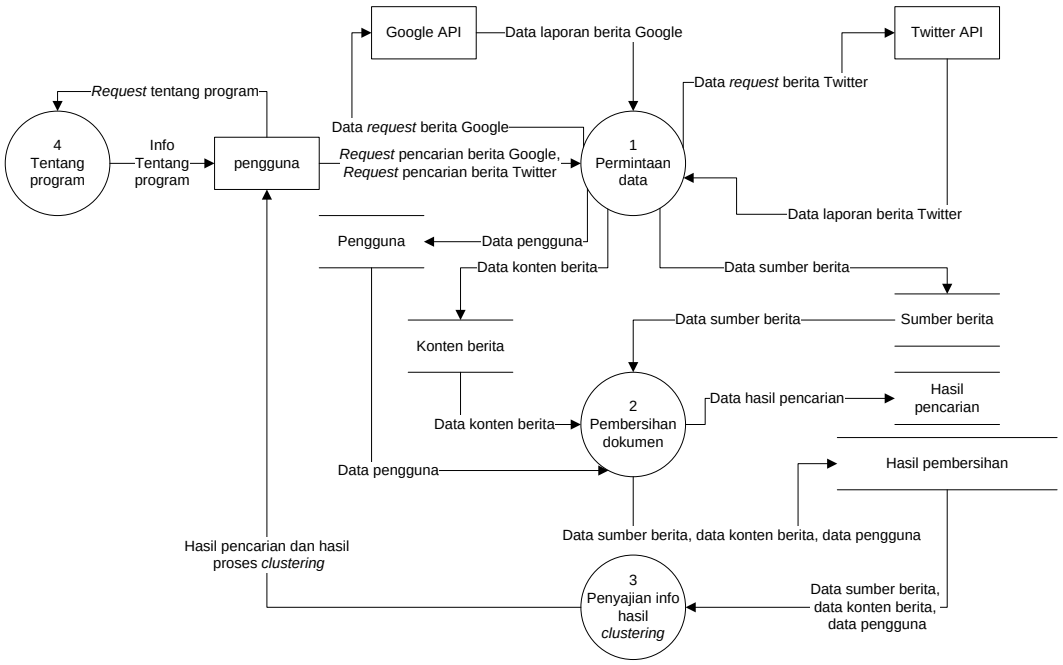
Diagram konteks adalah diagram yang menggambarkan masukan dan keluaran yang terjadi pada sistem. Diagram konteks untuk aplikasi ini digambarkan dalam Gambar III-4.



Gambar III-4 Diagram konteks

III.1.7.2 DFD Level 1

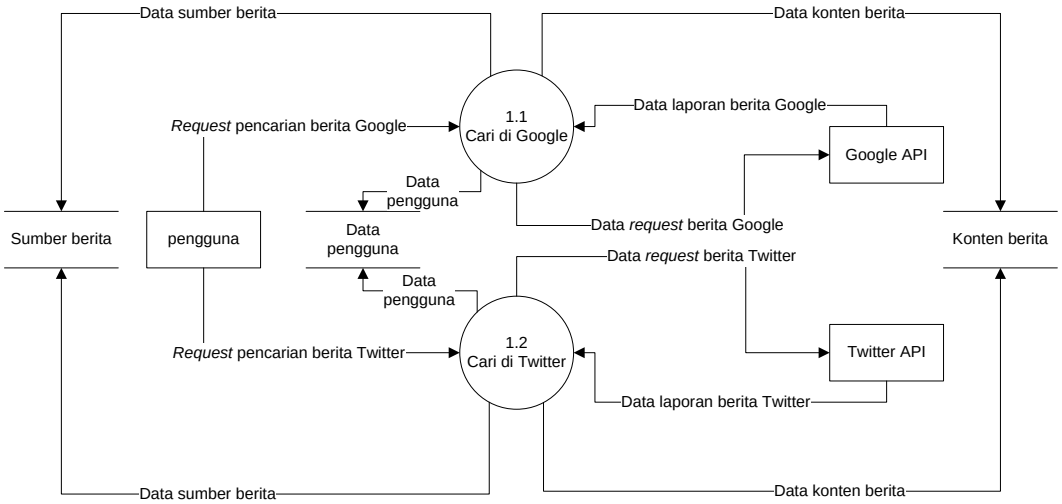
Dalam DFD level 1 terdapat 4 proses, yaitu proses permintaan data, pembersihan dokumen, clustering, dan tentang program. Proses pertama merupakan tempat interaksi pengguna terhadap perangkat lunak serta perangkat lunak terhadap *API* dari sumber berita yang dipilih untuk melakukan pencarian informasi. Proses ini kemudian akan menyimpan sumber data yang dipilih beserta konten yang dihasilkan dari pencarian ke dalam basis data. Penyimpanan ini diperlukan untuk proses kedua, yaitu pembersihan dokumen. Proses ini akan melakukan *clustering* terhadap hasil pencarian yang telah disimpan dalam basis data, dan kemudian hasil *clustering* dan pencarian akan diberikan kepada pengguna oleh proses ketiga agar dapat ditelusuri lebih lanjut.



Gambar III-5 DFD Level 1

III.1.7.3 DFD Level 2 Proses 1

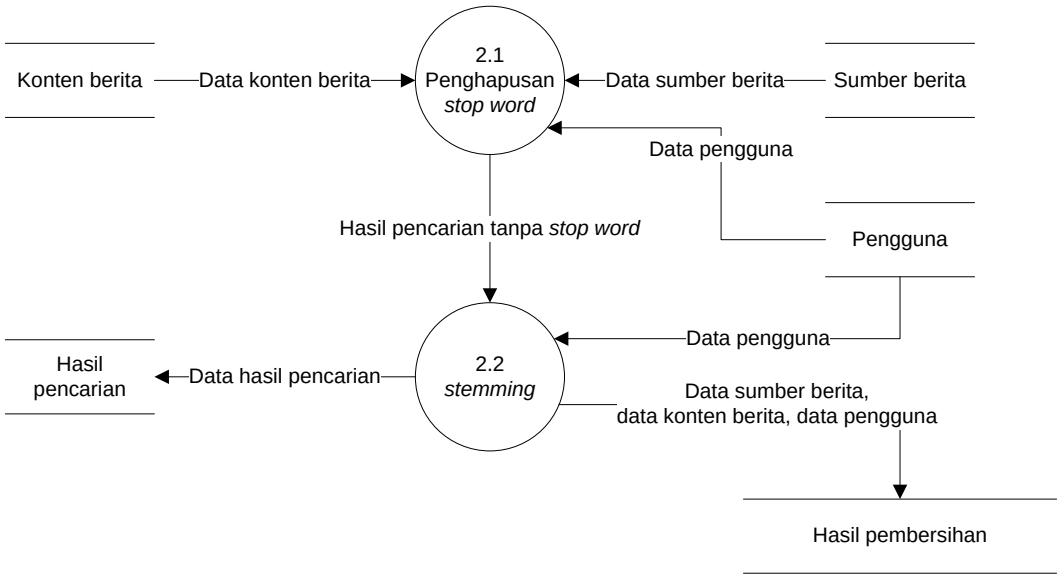
Dalam proses ini, permintaan dari pengguna akan diteruskan kepada API yang sesuai. API yang ada dalam perangkat lunak yang dibangun hanyalah API Google dan Twitter. Jika permintaan pengguna sesuai dengan aturan dari API yang terpilih, maka permintaan tersebut akan diproses namun hasilnya tidak akan langsung diberikan kepada pengguna. Hasil dari API ini akan disimpan di dalam *database* terlebih dahulu sebelum menjadi masukan untuk proses selanjutnya.



Gambar III-6 DFD level 2 proses 1

III.1.7.4 DFD Level 2 Proses 2

DFD level 2 ini merupakan spesifikasi dari proses Pembersihan Dokumen. Pada level ini, dokumen hasil pencarian berupa sumber berita beserta isinya yang telah disimpan dalam basis data diproses agar dapat dilakukan *clustering* dalam proses 3 level 1. Dokumen-dokumen yang telah disimpan harus melalui proses penghapusan *stop word*, proses *stemming* serta proses detail pembersihan. Dokumen kemudian akan disimpan kembali dalam basis data hasil pembersihan dokumen dan kemudian hasilnya dapat digunakan untuk proses 3 level 1.



Gambar III-7 DFD Level 2 proses 2

III.1.8 Spesifikasi Proses

Spesifikasi proses digunakan untuk menggambarkan aliran proses yang terdapat pada DFD. Spesifikasi proses dari gambaran DFD di atas akan dijelaskan pada tabel III-3 di bawah ini.

Tabel III-6 Spesifikasi proses

No	Proses	Keterangan
1	No. Proses	1
	Nama Proses	Permintaan data
	Sumber	a. Pengguna b. Google API c. Twitter API
	Data Masukan	a. Request pencarian berita Google b. Request pencarian berita Twitter c. Data laporan berita Google d. Data laporan berita Twitter
	Data Keluaran	a. Data sumber berita b. Data *request* berita Google c. Data *request* berita Twitter d. Data konten berita e. Data pengguna
	Tujuan	a. Google API b. Twitter API

No	Proses	Keterangan
	Logika Proses	1. Pengguna melakukan *request* pencarian berita kepada perangkat lunak. 2. Perangkat lunak kemudian meneruskan *request* kepada Google API atau Twitter API. 3. Google API atau Twitter API akan memeriksa apakah *request* tersebut sesuai aturan yang diterapkan atau tidak. Jika ya, tampilkan laporan data berita. Jika tidak, ulangi pencarian.
No	**Proses**	**Keterangan**
2	No. Proses	2
	Nama Proses	Pembersihan dokumen
	Sumber	a. Sumber berita b. Konten berita c. Pengguna
	Data Masukan	a. Data sumber berita b. Data konten berita c. Data pengguna
	Data Keluaran	a. Data sumber berita b. Data konten berita c. Data pengguna d. Data hasil pencarian
	Tujuan	a. Hasil pencarian b. Hasil pembersihan
	Logika Proses	1. Masukan dari proses ini adalah Isi sumber berita dan isi berita hasil pencarian pengguna. 2. Pembersihan dokumen kemudian dilakukan terhadap data masukan. Pembersihan dokumen berupa penghapusan *stop word* serta *stemming* kata. 3. Data keluaran yang dihasilkan dari proses ini adalah dokumen hasil *clustering*.
No	**Proses**	**Keterangan**
3	No. Proses	3
	Nama Proses	Penyajian info hasil *clustering*
	Sumber	Hasil pembersihan
	Data Masukan	a. Data sumber berita b. Data konten berita c. Data pengguna
	Data Keluaran	Hasil pencarian dan hasil proses *clustering*
	Tujuan	pengguna
	Logika Proses	1. Masukan dari proses ini adalah isi berita hasil proses pembersihan dokumen. Hasil dari pembersihan dokumen berupa dokumen tanpa *stop-words*, serta *stemming* terhadap kata-kata yang lolos dari proses sebelumnya. 2. Selanjutnya sistem melakukan *clustering* terhadap masukan ini. 3. Keluaran yang dihasilkan dari proses ini adalah

No	Proses	Keterangan
		hasil pencarian awal serta hasil proses *clustering* yang diberikan kepada pengguna.
No	**Proses**	**Keterangan**
4	No. Proses	4
	Nama Proses	Tentang program
	Sumber	Pengguna
	Data Masukan	*Request* tentang program
	Data Keluaran	Info tentang program
	Tujuan	Pengguna
	Logika Proses	1. Masukan dari proses ini adalah data *request* informasi tentang perangkat lunak oleh pengguna. 2. Keluaran yang dihasilkan adalah informasi tentang perangkat lunak.
No	**Proses**	**Keterangan**
5	No. Proses	1.1
	Nama Proses	Cari di Google
	Sumber	a. Pengguna b. Google API c. Twitter API
	Data Masukan	a. Request pencarian berita Google b. Data laporan berita Google
	Data Keluaran	a. Data sumber berita b. Data *request* berita Google c. Data konten berita d. Data pengguna
	Tujuan	a. Google API b. Sumber berita c. Konten berita d. Data pengguna
	Logika Proses	1. Masukan dari proses ini adalah data pencarian berdasarkan kata kunci yang dimasukan oleh pengguna. 2. Masukan ini kemudian diproses untuk diteruskan kepada API Google. 3. Keluaran yang dihasilkan adalah data pengguna, data sumber berita, dan data konten berita hasil pencarian Google yang digunakan untuk proses pembersihan dokumen.
No	**Proses**	**Keterangan**
6	No. Proses	1.2
	Nama Proses	Cari di Twitter
	Sumber	a. Pengguna b. Google API c. Twitter API
	Data Masukan	a. Request pencarian berita Twitter d. Data laporan berita Twitter

	Data Keluaran	a. Data sumber berita b. Data *request* berita Twitter c. Data konten berita d. Data pengguna
	Tujuan	a. Twitter API b. Sumber berita c. Konten berita d. Data pengguna
	Logika Proses	1. Masukan dari proses ini adalah data pencarian berdasarkan kata kunci yang dimasukan oleh pengguna. 2. Masukan ini kemudian diproses untuk diteruskan kepada API Twitter. 3. Keluaran yang dihasilkan adalah data pengguna, data sumber berita, dan data konten berita yang digunakan untuk proses selanjutnya.
No	**Proses**	**Keterangan**
7	No. Proses	2.1
	Nama Proses	Penghapusan *stop word*
	Sumber	a. Sumber berita b. Konten berita c. Pengguna
	Data Masukan	a. Data konten berita b. Data sumber berita
	Data Keluaran	Hasil pencarian tanpa *stop word*
	Tujuan	*Stemming*
	Logika Proses	1. Masukan dari proses ini adalah data sumber berita, data konten berita, dan data pengguna. 2. Masukan ini kemudian diproses untuk dilakukan penghapusan terhadap *stop word* yang dimilikinya. Penghapusan dilakukan dengan menelusuri dan memeriksa setiap kata dalam data masukan. Kata-kata yang termasuk ke dalam *stop word* terdedifinisi sebelumnya. 3. Keluaran yang dihasilkan adalah hasil pencarian tanpa *stop-words* untuk proses 2.2.
No	**Proses**	**Keterangan**
8	No. Proses	2.2
	Nama Proses	*Stemming*
	Sumber	Penghapusan *stop word*
	Data Masukan	a. Hasil pencarian tanpa *stop word* b. Data pengguna
	Data Keluaran	a. Data sumber berita b. Data konten berita c. Data pengguna d. Data hasil pencarian
	Tujuan	clustering
	Logika Proses	1. Proses *stemming* terhadap kata-kata hasil dari proses 2.1 dilakukan dalam proses ini. 2. *Stemming* dilakukan dengan aturan-aturan yang

No	Proses	Keterangan
		telah terdefinisi. 3. Keluaran dari proses ini adalah dokumen-dokumen ter-*cluster* yang akan ditampilkan kepada pengguna dalam proses 3.

III.1.9 Kamus Data

Kamus data adalah penjelasan tertulis tentang suatu data yang berada di dalam basis data. Kamus data untuk perangkat lunak ini diuraikan sebagai beikut :

Tabel III-7 Kamus Data

Nama Aliran Data	*Request* pencarian berita Google
Where used / how used	Proses 1, proses 1.1
Deskripsi	Berisi data pencarian pengguna
Struktur data	Data pencarian
Data_pencarian	{Karakter ASCII}
Nama Aliran Data	*Request* pencarian berita Twitter
Where used / how used	Proses 1, proses 1.2
Deskripsi	Berisi data pencarian pengguna
Struktur data	Data pencarian
Data_pencarian	{Karakter ASCII}
Nama Aliran Data	Data laporan berita Google
Where used / how used	Proses 1, proses 1.1
Deskripsi	Berisi data hasil pencarian Google berdasarkan permintaan pengguna
Struktur data	Data hasil pencarian Google
Data_hasil_pencarian_Google	{Karakter ASCII}
Nama Aliran Data	Data laporan data berita Twitter
Where used / how used	Proses 1, proses 1.1
Deskripsi	Berisi data hasil pencarian Twitter berdasarkan permintaan pengguna
Struktur data	Data hasil pencarian Twitter
Data_hasil_pencarian_Twitter	{Karakter ASCII}
Nama Aliran Data	Data pengguna
Where used / how used	Proses 2, proses 2.1, proses 2.2
Deskripsi	Berisi data pengguna
Struktur data	a. Id pengguna b. Session pengguna
Id_pengguna	[0..9]
Session pengguna	[0..9]
Nama Aliran Data	Data sumber berita
Where used / how used	Proses 2, proses 2.1
Deskripsi	Berisi data sumber berita dari hasil pencarian

Struktur data	c. Id sumber berita d. Nama sumber berita
Id_sumber_berita	[0..9]
Nama_sumber_berita	[A..Z|a..z]
Nama Aliran Data	Data konten berita
Where used / how used	Proses 2, proses 2.1
Deskripsi	Berisi data hasil pencarian berita
Struktur data	a. Id konten berita b. Id sumber berita c. Judul konten d. Isi konten e. Link
Id_konten_berita	[0..9]
Id_sumber_berita	[0..9]
Judul_konten	[0..9|A..Z|a..z]
Isi_konten	[0..9|A..Z|a..z]
Link	[0..9|A..Z|a..z]

III.2 Perancangan Sistem

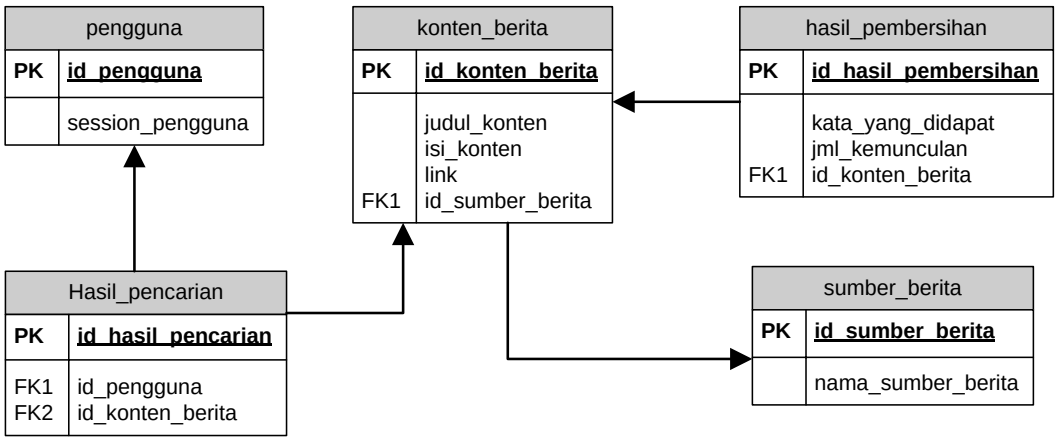
Perancangan sistem dilakukan setelah tahap analisis sistem selesai dilakukan. Perancangan dapat didefinisikan sebagai penggambaran, perencanaan dan pembuatan sketsa atau aturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh.

III.2.1 Perancangan Basis Data

Perancangan basis data merupakan tahapan untuk memetakan model konseptual ke model basis data yang akan dipakai. Perancangan basis data terbagi menjadi dua, yaitu skema relasi dan perancangan struktur data.

III.2.1.1 Skema Relasi

Skema relasi adalah relasi yang memiliki nama, didefinisikan oleh himpunan pasangan atribut dan domainnya.



Gambar III-8 Skema relasi

III.2.1.2 Struktur Tabel

Struktur tabel merupakan isi data yang berada dalam satu *record* pada satu tabel. Struktur tabel dalam perangkat lunak yang akan dibangun adalah sebagai berikut :

Tabel sumber berita adalah tabel yang akan menampung data dari sumber pencarian berita yang dipilih pengguna, dalam hal ini Google dan Twitter. Tabel ini memiliki dua *field*, yaitu id_sumber_berita dan nama_sumber_berita. Berikut ini adalah struktur tabel sumber berita dari perangkat lunak yang dibangun.

Tabel III-8 Struktur tabel sumber berita

Nama tabel	Sumber_berita			
Primary key	Id_sumber_berita			
Nama field	Tipe data	ukuran	kunci	Keterangan
Id_sumber_berita	Int	2	PK	
Nama_sumber_berita	Varchar	10		*Not Null*

Tabel konten berita adalah tabel yang digunakan untuk menyimpan data hasil pencarian berdasarkan kata kunci pengguna terhadap sumber berita yang dipilih. Tabel ini akan menyimpan beberapa informasi, seperti id_konten_berita,

id_sumber_berita, judul_konten, isi_konten, dan *link* dari konten yang didapat dari hasil pencarian. Berikut ini adalah struktur tabel konten berita.

Tabel III-9 Struktur tabel konten berita

Nama tabel	Konten_berita			
Primary key	Id_konten_berita			
Nama field	Tipe data	ukuran	kunci	Keterangan
Id_konten_berita	Int	11	PK	
Id_sumber_berita	Varchar	10	FK	*Reference* dari tabel sumber_berita
Judul_konten	Varchar	500		*Not Null*
Isi_konten	Varchar	500		*Not Null*
Link	Varchar	500		*Not Null*

Tabel hasil pembersihan dokumen akan menyimpan hasil dari proses pembersihan dokumen. Informasi yang disimpan adalah id_hasil_pembersihan, id_konten_berita, kata_yang_didapat dari keluaran proses *stemming*, serta jumlah_kemunculan yang berguna untuk *scoring* dokumen. Berikut ini adalah struktur tabel hasil pembersihan dokumen.

Tabel III-10 Struktur tabel hasil pembersihan dokumen

Nama tabel	Hasil_pembersihan			
Primary key	Id_hasil_pembersihan			
Nama field	Tipe data	ukuran	kunci	Keterangan
Id_hasil_pembersihan	Int	11	PK	
Id_konten_berita	Int	11	FK	*Reference* dari tabel konten_berita
Kata_yang_didapat	Varchar	30		*Not Null*
Jml_kemunculan	Int	11		*Not Null*

Tabel hasil pencarian digunakan untuk menyimpan informasi berupa id_hasil_pencarian, id_konten_berita, dan id_pengguna. Tabel ini muncul berdasarkan relasi antara pengguna dan konten berita. Berikut ini adalah struktur tabel hasil pencarian.

Tabel III-11 Struktur tabel hasil pencarian

Nama tabel	Hasil_pencarian			
Primary key	Id_hasil_pencarian			
Nama field	Tipe data	ukuran	kunci	Keterangan
Id_hasil_pencarian	Int	11	PK	
Id_konten_berita	Int	11	FK	*Reference* dari tabel konten_berita
Id_pengguna	Int	11	FK	*Reference* dari tabel pengguna

Tabel pengguna digunakan untuk menyimpan informasi dari pengguna yang melakukan pencarian, yang disimpan *field* id_pengguna dan session_pengguna. Berikut ini adalah struktur tabel pengguna.

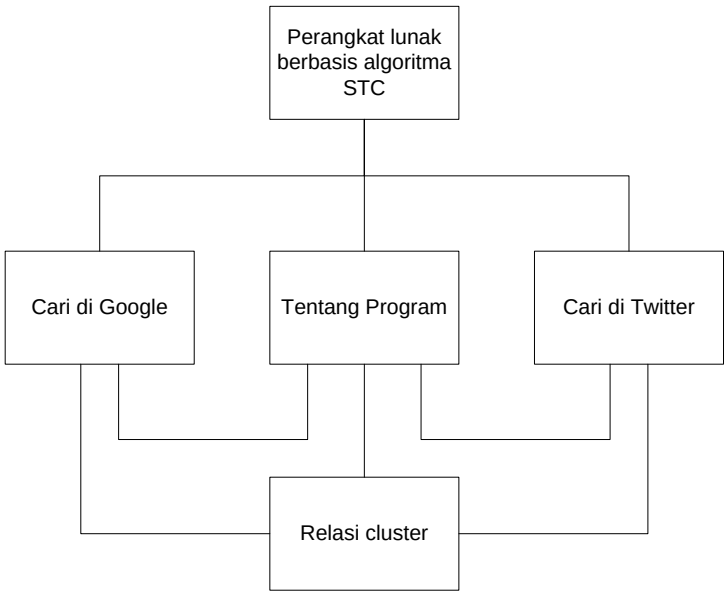
Tabel III-12 struktur tabel pengguna

Nama tabel	Pengguna			
Primary key	Id_pengguna			
Nama field	Tipe data	ukuran	kunci	Keterangan
Id_pengguna	Int	11	PK	
Session_pengguna	Int	11		*Not Null*

III.2.2 Perancangan Arsitektur Perangkat Lunak

III.2.2.1 Perancangan Struktur Menu

Struktur menu dari perangkat lunak yang dibangun adalah sebagai berikut :



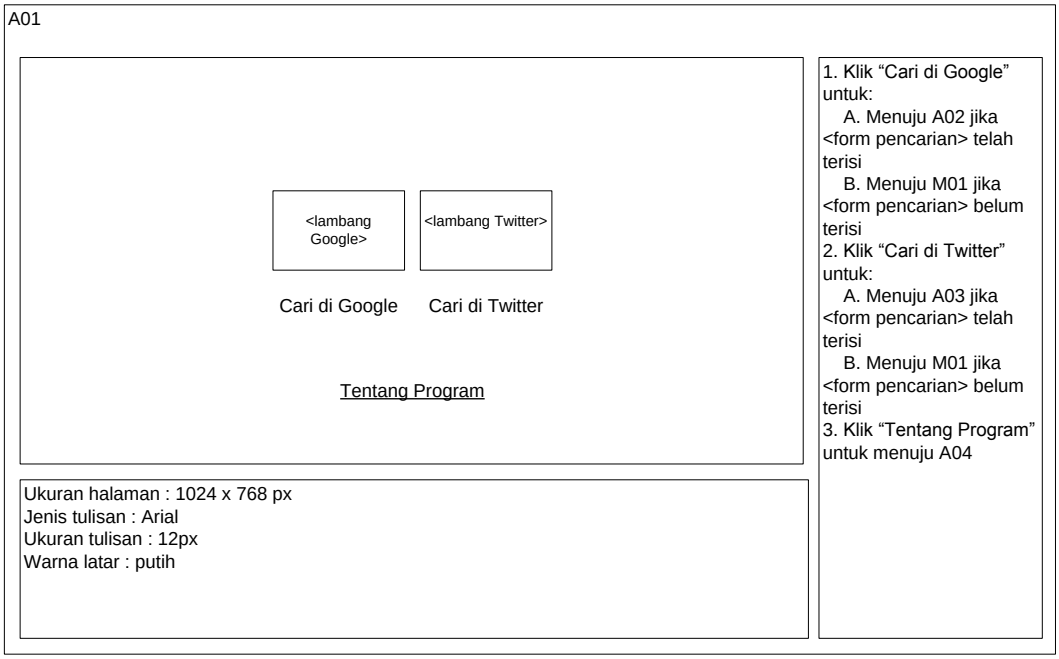
Gambar III-9 Struktur menu

III.2.3 Perancangan Antarmuka

Antarmuka merupakan tampilan dari suatu perangkat lunak yang berperan sebagai media komunikasi yang digunakan sebagai sarana antara program dan pengguna. Perancangan ini merupakan sebuah penggambaran, perencanaan dan pengaturan dari beberapa elemen terpisah ke dalam suatu kesatuan utuh. Sistem yang dibangun diharapkan menyediakan suatu tampilan antarmuka yang mudah dipahami dan digunakan oleh pengguna. Berikut ini adalah tampilan antarmuka dalam perangkat lunak yang dibangun.

III.2.3.1 Perancangan Antarmuka Halaman Utama

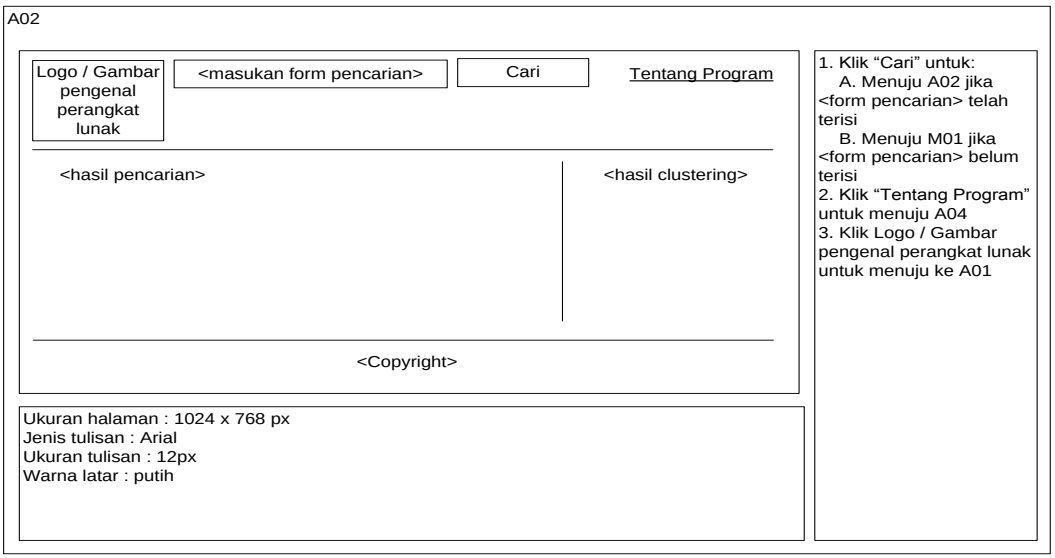
Halaman utama adalah halaman yang akan pertama kali muncul saat pengguna mengakses aplikasi. Berikut ini adalah perancangan antarmuka halaman utama.



Gambar III-10 Perancangan antarmuka halaman utama

III.2.3.2 Perancangan Antarmuka Hasil Pencarian Google

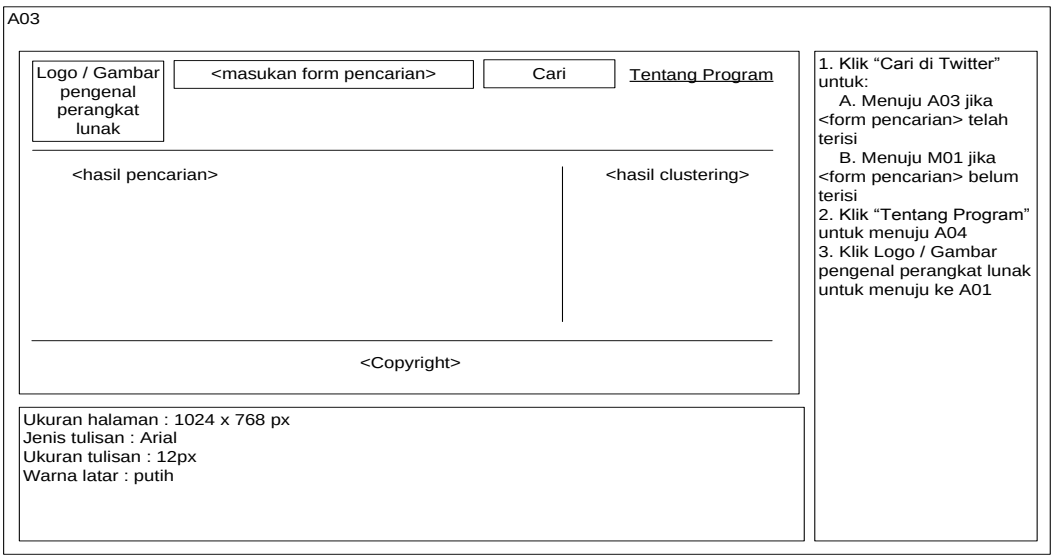
Antarmuka ini akan muncul saat pengguna memilih untuk melakukan pencarian di dalam Google. Berikut ini adalah perancangan antarmuka hasil pencarian di Google.



Gambar III-11 Perancangan antarmuka hasil pencarian Google

III.2.3.3 Perancangan Antarmuka Hasil Pencarian Twitter

Antarmuka ini akan muncul saat pengguna memilih untuk melakukan pencarian di dalam Twitter. Berikut ini adalah perancangan antarmuka hasil pencarian di Twitter.



Gambar III-12 Perancangan antarmuka hasil pencarian Twitter

III.2.3.4 Perancangan Antarmuka Hasil *Clustering*

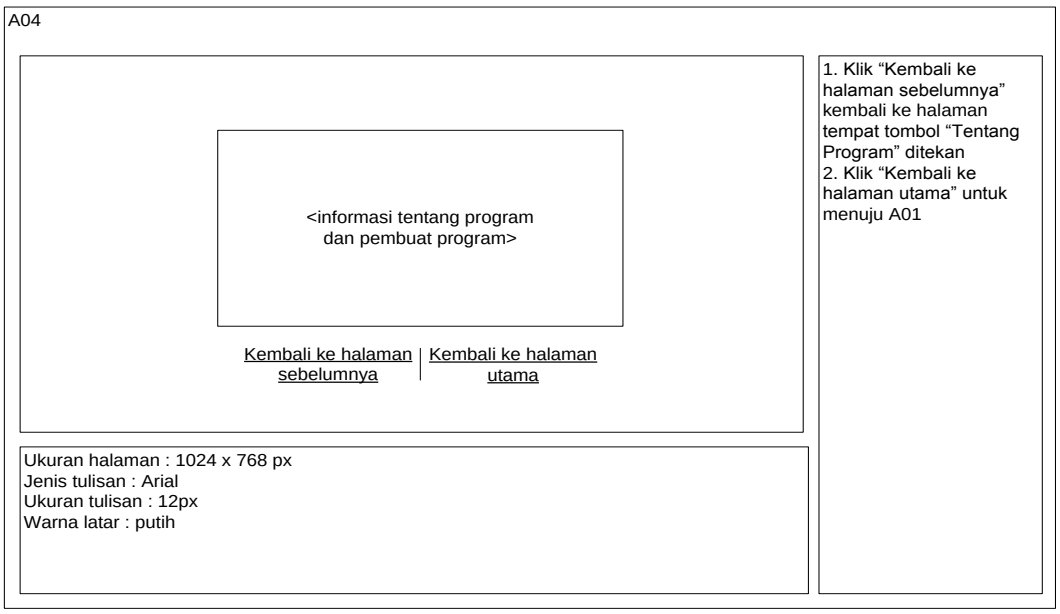
Halaman ini dapat digunakan untuk melihat hasil *clustering* berdasarkan *cluster* yang dipilih. Perancangan antarmuka hasil *clustering* dapat dilihat pada Gambar III-14.

III.2.3.5 Perancangan Antarmuka Relasi *Cluster*

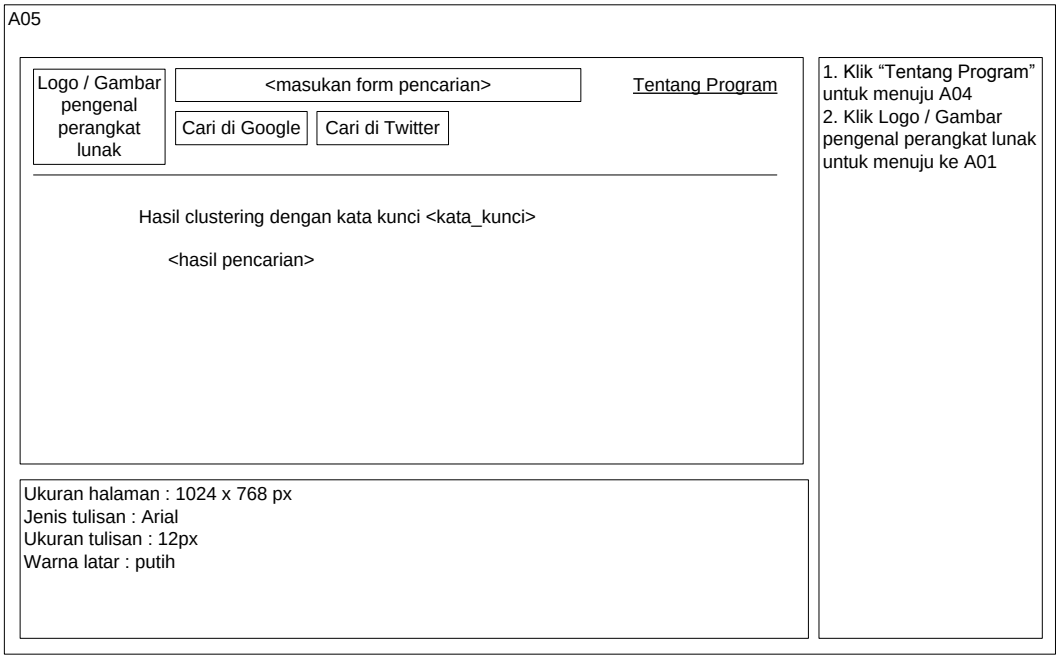
Pada antarmuka ini terdapat informasi mengenai relasi antara 2 *cluster*. Kedua *cluster* dipilih berdasarkan pengguna. Antarmuka ini dapat dilihat pada Gambar III-15.

III.2.3.6 Perancangan Antarmuka Tentang Program

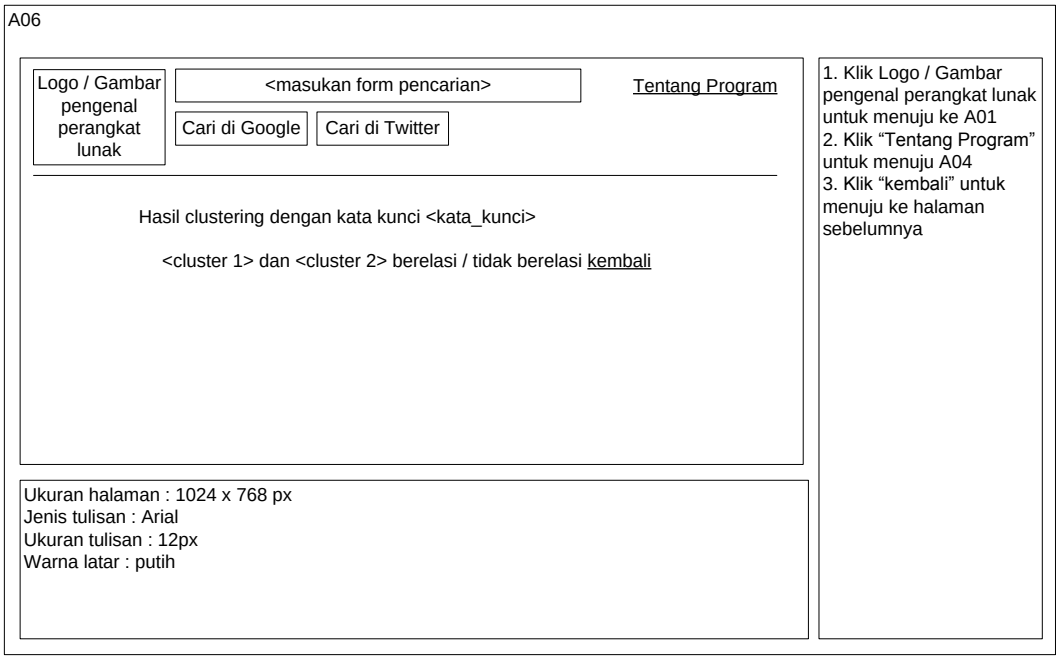
Antarmuka Tentang Program digunakan utnuk menampilkan informasi tentang perangkat lunak yang dibangun serta pembangunnya. Perancangan antarmuka tentang program dapat dilihat pada Gambar III-13.



Gambar III-13 Perancangan antarmuka tentang program



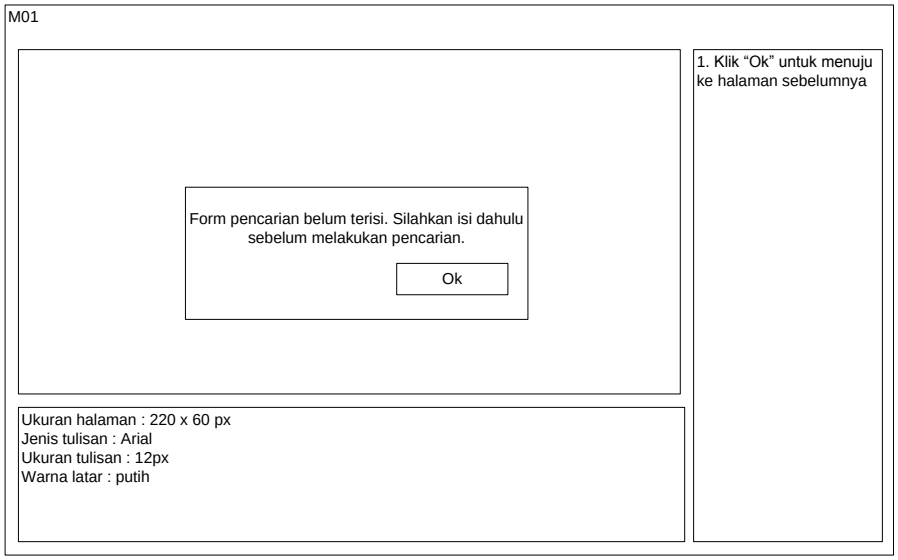
Gambar III-14 Perancangan antarmuka hasil *clustering*



Gambar III-15 Perancangan antarmuka relasi *clustering*

III.2.3.7 Perancangan Antarmuka Pesan Kesalahan

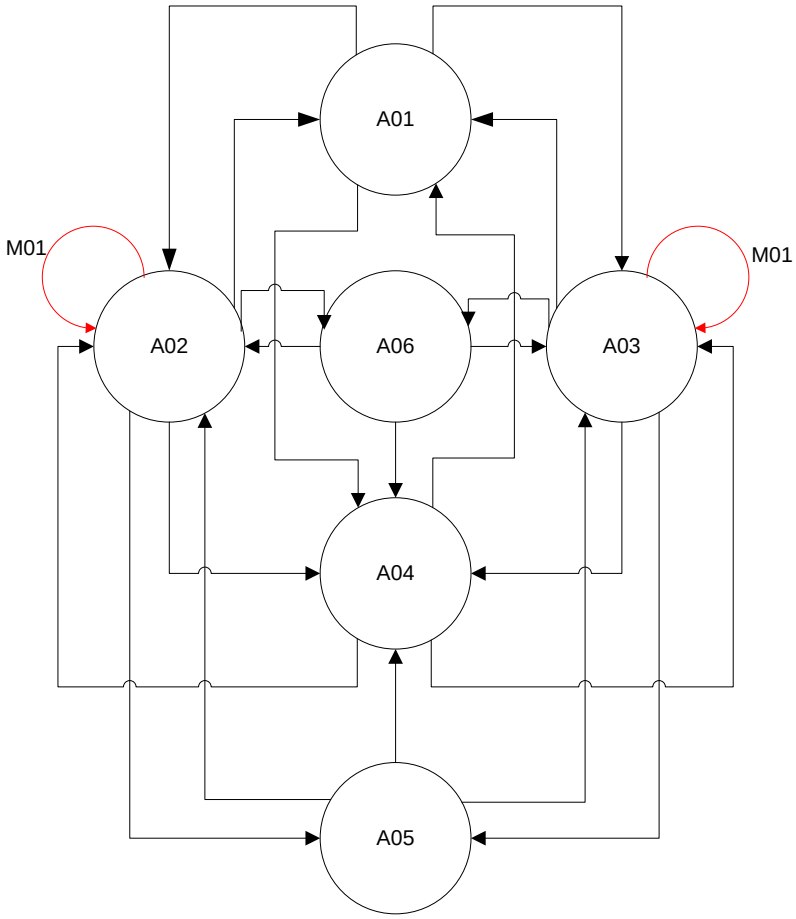
Pesan kesalahan akan muncul pada saat pengguna menekan salah satu tombol untuk melakukan pencarian namun belum mengisi *form* pencarian. Berikut ini adalah perancangan antarmuka pesan kesalahan dalam perangkat lunak yang dibangun.



Gambar III-16 Perancangan antarmuka pesan kesalahan

III.2.4 Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam pebuah perangkat lunak. Berikut ini adalah jaringan semantik dari perangkat lunak yang dibangun.

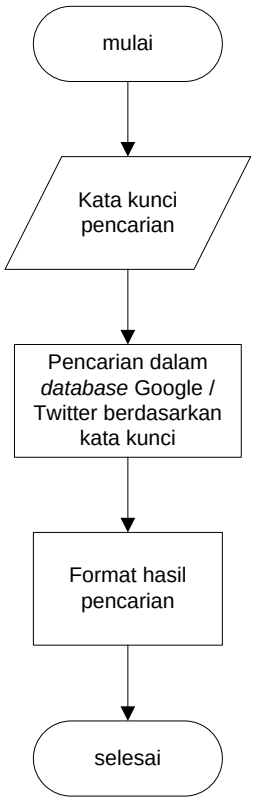


Gambar III-17 Jaringan Semantik

III.2.5 Perancangan Prosedural

Perancangan prosedural digunakan sebagai algoritma dasar yang akan diimplementasikan ke dalam sistem. Perancangan prosedural dari perangkat lunak yang dibangun untuk proses pencarian adalah sebagai berikut :

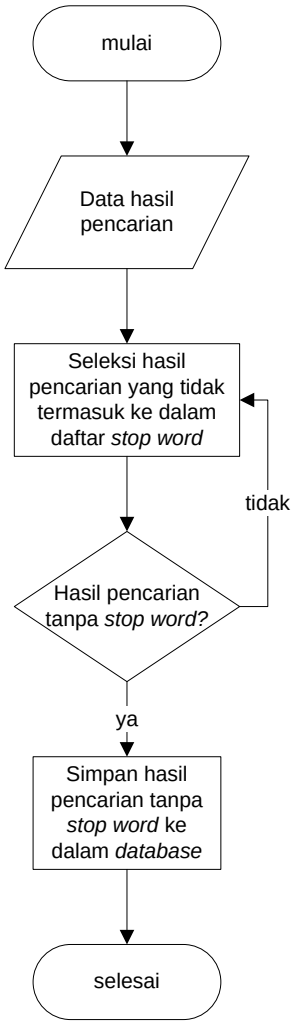
1. Pengguna memasukkan kata kunci pencarian ke dalam sistem.
2. Sistem melakukan pencarian di dalam database Google atau Twitter.
3. Hasil pencarian kemudian akan di-*format* untuk disiapkan sebelum ditampilkan kepada pengguna



Gambar III-18 Perancangan prosoedural permintaan data

Berikut ini adalah langkah-langkah yang dilakukan untuk prosedural penghapusan *stop word* :

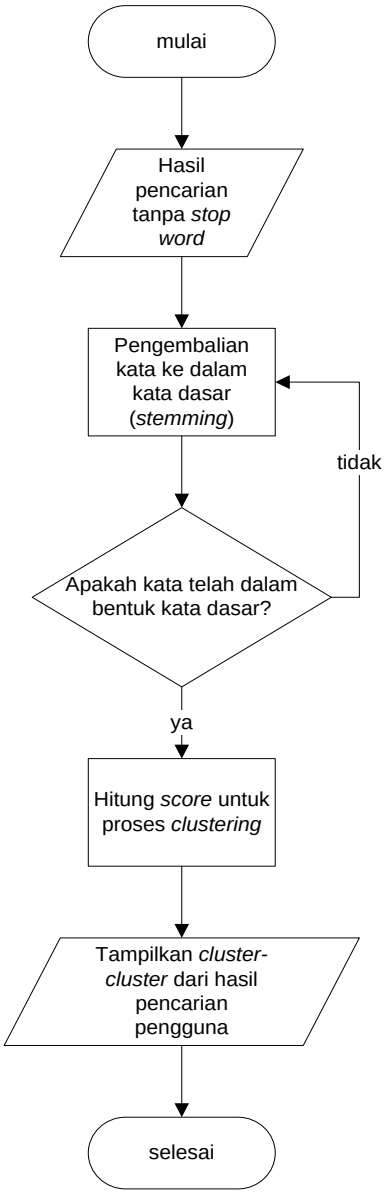
1. Masukan dari prosedural ini adalah data hasil pencarian pengguna.
2. Setiap kata dari hasil pencarian kemudian diseleksi untuk menemukan apakah terdapat *stop word* atau tidak.
3. Jika ya, maka jangan proses kata tersebut.
4. Jika tidak, masukkan kata tersebut untuk disimpan dan diproses pada langkah selanjutnya.



Gambar III-19 Perancangan prosedural penghapusan *stop word*

Langkah-langkah yang digunakan untuk proses stemming adalah sebagai berikut :

1. Kata hasil seleksi dari proses *stop word* dijadikan masukan untuk proses ini.
2. Seleksi kata tersebut berdasarkan aturan *stemming* yang telah ditetapkan.
3. Proses kata tersebut untuk dijadikan *cluster* dasar dan kemudian tampilkan kepada pengguna



Gambar III-20 Perancangan procedural *clustering* dokumen