BAB 3
ANALISIS DAN KEBUTUHAN ALGORITMA

3.1 Analisis Masalah

Beberapa hal dari penjabaran latar belakang yang telah diuraikan pada bab 1, maka didapat analisis masalah yang terjadi dalam penelitian ini adalah sebagai berikut. Dengan semakin banyaknya dokumen teks yang akan diakses serta meningkatnya pencarian informasi, pengguna semakin kesulitan untuk menemukan dokumen yang relevan sesuai dnegan query yang diinputkan serta adanya kebutuhan untuk mendapatkan metode *stemming* yang memiliki performansi paling baik untuk menemukan kata dasar bahasa Indonesia antara Algoritma Idris dan Algoritma ECS *Stemmer.*

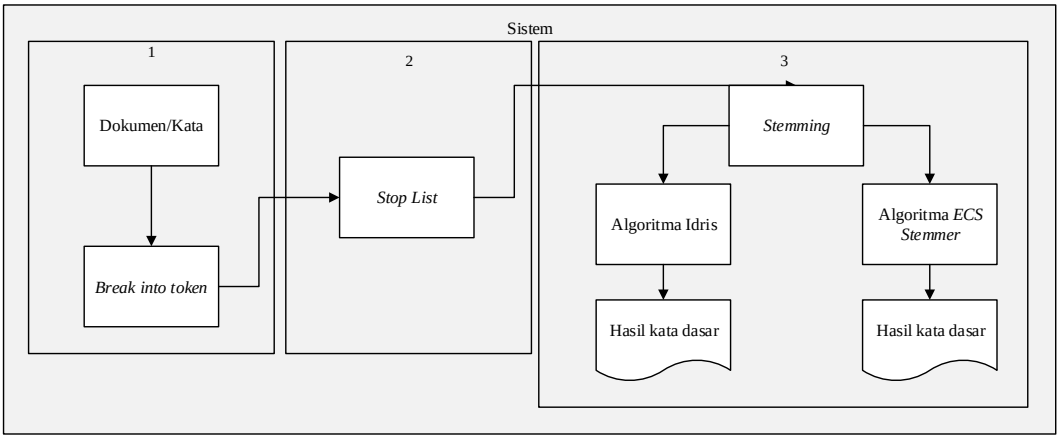
3.2 Analisis Sistem

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 diciptakan suatu simulator yang baik. Dalam mengimplementasikan algoritma Idris dan Algoritma *Enhanced Confix Stripping Stemmer* akan dibangun sebuah simulator dengan gambaran sistem yang akan tertera pada gambar 3.1.

Simulator yang akan dibangun memiliki 3 (tiga) tahap yaitu *tokenizing*, *filtering* dan *stemming*. Untuk penjelasannya setiap tahapan dapat dilihat sebagi berikut ini:

1. Tahap pertama adalah tahap dimana akan dilakukan penghapusan tanda baca, angka, serta pengubahan huruf besar menjadi huruf kecil. Kemudian akan dilakukan pemecah kalimat menjadi kata – kata, karena tanda baca sudah dihilangkan sebelumnya maka pemisahan kata – kata dalam kalimat ini dilakukan berdasarkan adanya spasi.
2. Tahapan yang kedua adalah melakukan penghapusan kata – kata umum sesuai dengan kamus *stopword,* misalnya dalam, dan, karena, mana, untuk dan lainnya.

3. Tahapan ketiga yaitu proses *stemming* merupakan pengelompokan kata secara morfologis sesuai dengan kelompoknya. Pada tahapan ini akan dilakukan perubahan kata – kata yang berimbuhan menjadi kata dasar dengan cara menghilangkan imbuhan depan, sisipan maupun akhiran. Pada tahapan *stemming* akan dilakukan dengan dua algoritma yaitu algoritma Idris dan algoritma *Enhanced Confix Stripping Stemmer*. Hasil *stemming* merupakan kata dasar dari dua algoritma tersebut yang hasilnya akan dibandingkan guna mengetahui algoritma mana yang paling baik performansinya untuk dokumen teks bahasa Indonesia.



Gambar 3.1 Gambaran Umum Sistem

3.3 Analisis Data Masukkan

Data yang digunakan dalam *stemming* sebagi data masukkan merupakan kata dasar yang dapat berbentuk kalimat atau paragraf. Jumlah dokumen yang diinputkan minimal memiliki 100 kata, tiap kata tersebut akan diolah untuk mendapatkan hasil akhir berupa kata dasar.

Dalam penelitian tugas akhir ini data pengolahan kata yang akan digunakan untuk analisis data adalah sebagai berikut :

1. Data masukkan berupa sebuah kata yang dapat berdiri sendiri yang diinputkan langsung kedalam simulator.
2. Data masukkan dapat berupa sebuah kalimat yang terdiri dari beberapa kata, atau sebuah paragraf yang terdiri dari beberapa kalimat.

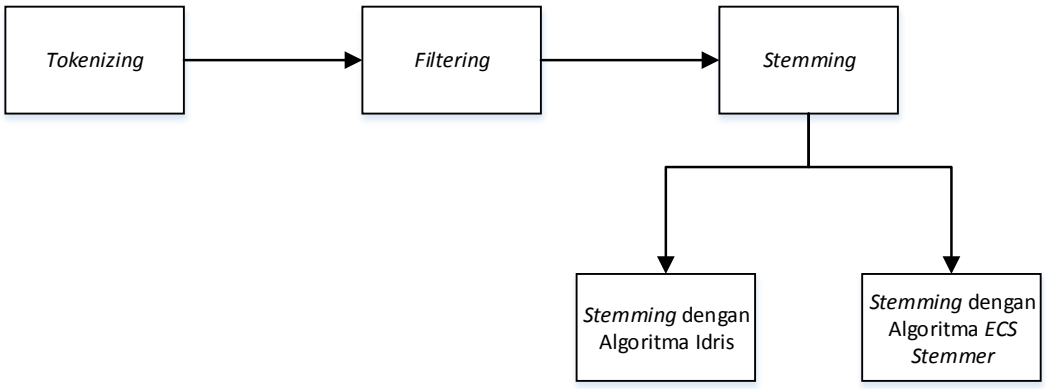
3. Kalimat merupakan sebuah gabungan dari kata-kata yang berawalan huruf besar dan diakhir kalimat diakhiri dengan tanda titik (.).
4. Data dari teks atau dokumen yang diinputkan berekstensi .txt.
5. Kemudian data yang selain kata pada dokumen yang diinputkan akan dihilangkan dengan menggunakan tahapan *tokenizing*, *filtering* dan kamus sebagai pembanding kata yang akan distemming.

3.4 Analisis Metode

Analisis metode menjelaskan bagaimana *stemming* itu terjadi dan menjelaskan tentang tahap-tahap yang dilakukan untuk melakukan proses *stemming.* Secara umum sistem ini terdiri dari beberapa proses, proses – proses tersebut adalah sebagai berikut :

1. Proses *Tokenizing.*
2. Proses *Filtering.*
3. Proses *Stemming* yaitu terdiri dari *stemming* dengan algoritma Idris dan *stemming* algoritma ECS *Stemmer.*

Tahapan dari analisis metode dapat dilihat pada gambar 3.2 berikut.



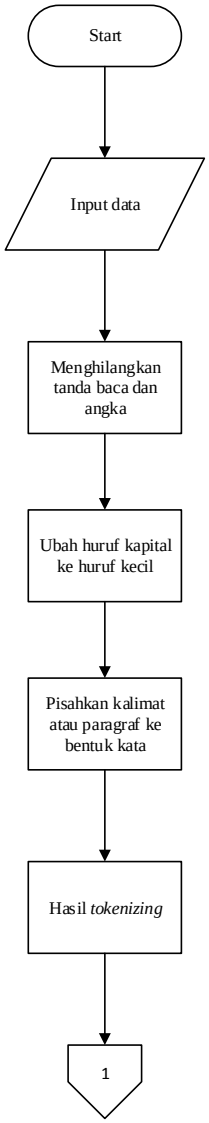
Gambar 3.2 Tahapan analisis metode dalam *stemming*

Berikut pemaparan yang lebih jelas dari tahapan-tahapan *stemming* pada gambar 3.2 sebagai berikut :

1. Proses *Tokenizing*

Dalam proses dilakukan penghapusan terhadap tanda baca, angka serta pengubahan huruf besar menjadi huruf kecil. Kemudian memecah dokumen menjadi *token-*

token dengan memotongnya menjadi *term* (kata). Berikut adalah langkah-langkah untuk melakukan *tokenizing* pada sebuah kalimat dapat dilihat pada gambar 3.3.



Gambar 3.3 *Flowchart Tokenizing*

Contoh kalimat yang akan melewati tahapan *tokenizing* :

"Sebuah bunyi letusan keras yang menggemakan memecahkan keheningan."

Penjelasan mengenai proses *tokenizing* dapat dilihat pada tabel 3.1.

a. "Sebuah bunyi letusan keras yang menggemakan memecahkan keheningan."
b. sebuah bunyi letusan keras yang menggemakan memecahkan keheningan

Tabel 3.1 Proses *Tokenizing*

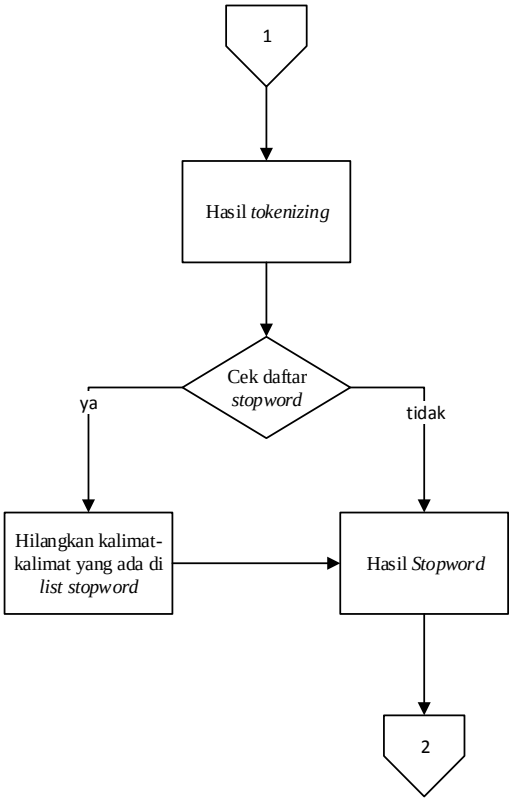
c. |sebuah|bunyi|letusan|keras|yang|menggemakan|memecahkan|keheningan|

Keterangan dari tabel 3.1 diatas adalah sebagai berikut :

a. Pada tahap ini *tokenizing* akan menghilangkan tanda baca, angka dan merubah huruf kapital menjadi huruf kecil.
b. Setelah itu proses terakhir dari *tokenizing* adalah memecah kalimat menjadi *term* (kata).

2. Proses *Filtering*

Proses *filtering* merupakan proses mengambil kata-kata penting yang terdapat dari hasil *tokenizing.* Untuk melakukan *filtering* bisa menggunakan *stoplist* atau *word list* atau *stopword.* Data hasil *tokenizing* akan dibandingkan dengan kamus. Jika tidak terdapat terdapat dalam kamus maka kata tersebut akan dihapus. Kata-kata yang tersisa merupakan kata yang dianggap penting. *flowchart* dari proses *filtering* dapat dilihat pada gambar 3.4 dibawah ini.



Gambar 3.4 *Flowchart Filtering*

Untuk lebih jelasnya tahapan proses *filtering* adalah sebagai berikut :

a. Kata hasil proses *tokenizing* dibandingkan dengan tabel *stopword*. Pada tabel *stopword* peneliti memilih 57 kata yang sering muncul dan dianggap tidak penting, *stopword* ini akan dimasukkan ke program dalam bentuk *class.*
b. Jika data hasil *tokenizing* sama dengan kata di tabel *stopword* maka akan dihapus.
c. Jika tidak sama dengan tabel *stopword* maka kata tersebut akan disimpan.

Berikut adalah contoh dari proses *filtering*:

Daftar *stopword* : yang, akan, pada, itu. Contoh langkah dari *filtering* dapat dilihat pada tabel 3.2 berikut.

Tabel 3.2 Proses *Filtering*

a.|sebuah|bunyi|letusan|keras|yang|menggemakan|memecahkan|keheningan
b.|sebuah|bunyi|letusan|keras|menggemakan|memecahkan|keheningan|

Keterangan dari tabel 3.2 diatas adalah sebagai berikut :

a. Pada baris a merupakan kalimat yang telah melewati tahapan *tokenizing.*
b. Selanjutnya dilakukan *filtering* yaitu menghapus kalimat – kalimat umum atau disebut *stopword.*

3. Proses *Stemming*

Proses *stemming* merupakan pengelolaan kata hasil *filtering* menjadi kata dasar yaitu dengan cara menghilangkan imbuhan yang diantaranya adalah ”di”, “ke”, “me”, “meng”, “-an”, “-kan” untuk lebih jelasnya tahapan dari proses *stemming* adalah sebagai berikut:

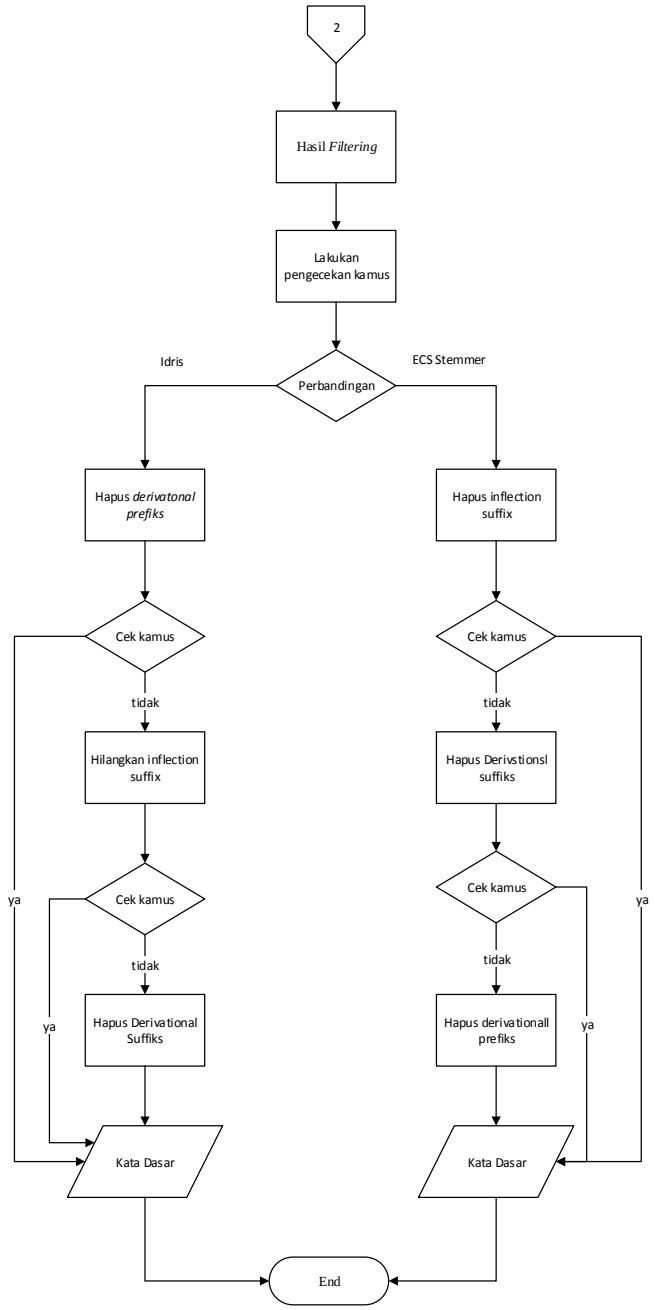
a. Data hasil *filtering* diperiksa apakah mengandung imbuhan atau tidak. Jika terdapat imbuhan maka akan dilakukan pembuangan imbuhan,terus diulang sampai kata tersebut tidak mengandung imbuhan.
b. Jika sudah tidak mengandung imbuhan maka akan disimpan.

Contoh dari proses *stemming* dapat dilihat pada tabel 3.3 berikut.

Tabel 3.3 Kata yang siap melewati proses *stemming*

|sebuah|bunyi|letusan|keras|menggemakan|memecahkan|keheningan|

Adapun *flowchart* dari proses *stemming* dapat dilihat pada gambar 3.5.



Gambar 3.5 *Flowchart Stemming*

3.4.1 *Stemming* dengan Algoritma Idris

Setelah melalui tahapan *tokenizing* dan *filtering* langkah selanjutnya adalah proses *stemming*. Berikut proses *stemming* dengan algoritma Idris.

a) Langkah 1

Melanjutkan proses *stemming* yang dilakukan seperti tabel 3.3 diatas maka selanjutnya adalah melakukan pengecekan atau pencarian kedalam kamus umum. Kata – kata yang valid dengan kamus umum akan dikeluarkan dari algoritma, karena dianggap sudah merupakan kata dasar. Maka kata – kata tersebut akan menjadi seperti tabel 3.4 dibawah ini.

Tabel 3.4 Mengeluarkan kata dasar

a.|sebuah|bunyi|letusan|keras|menggemakan|memecahkan|keheningan|
b.|sebuah|letusan|menggemakan|memecahkan|keheningan|

b) Langkah 2

Langkah berikutnya adalah melakukan cek kata pada aturan prefiks yaitu menghilangkan imbuhan – imbuhan seperti di-, men -, ber-. Maka kata – kata tersebut akan menjadi seperti tabel 3.5 dibawah ini.

Tabel 3.5 Menghapus awalan (prefiks)

b.|sebuah|letusan|menggemakan|memecahkan|keheningan|
c.|buah|letusan|gemakan|fecahkan|heningan|

c) Langkah 3

Langkah selanjutnya mengecek dikamus apabila sudah sesuai dengan kamus maka keluarkan dari proses, selanjutnya adalah menghapus suffiks (akhiran) yaitu menghilangkan akhiran seperti –an, -nya, -lah, -kah dan lainnya. Maka kata akan tampil seperti tabel 3.6 dibawah ini.

Tabel 3.6 Menghapus akhiran (suffiks)

c.|buah|letusan|gemakan|ecahkan|heningan|
c. |letus|gema|fecah|hening|

d) Langkah 4

Langkah berikutnya adalah melakukan pengecekan dikamus apabila cocok maka keluarkan dari kamus. Dalam contoh ini terdapat contoh kata *rule* 2 yaitu "memecahkan" dimana setelah pemotongan imbuhan mem- akan ditambahkan huruf "f" atau "p", yang pertama dilakukan adalah menambahkan huruf "f" dan akan dilakukan pengecekan dengan kamus apa bila cocok maka sudah benar

penambahan hurufnya bila tidak langkah selanjutnya adalah mengganti dengan huruf “p”. Setelah dilakukan pengecekan dikamus tidak terdapat kata dasar “fecah” maka akan diganti dengan “pecah” dan terdapat dalam kamus maka kata tersebut merupakan kata dasar. Apabila semua kata sudah valid dengan kamus umum maka kata – kata tersebut merupakan kata dasar dan telah berhasil di *stemming* dengan algoritma Idris akan seperti tabel 3.7 dibawah ini.

Tabel 3.7 Hasil *stemming* dengan menggunakan Algoritma Idris

a. |letus|gema|fecah|hening|
b. |fecah|
c. |pecah|
d.|buah|bunyi|letus|keras|gema|pecah|hening|

Perhitungan kekuatan *stemmer* ICF = (N-S)/N = (8-7)/8 = 0,125

3.4.2 *Stemming* dengan Algoritma *Enhanced Confix Stripping Stemmer*

Setelah melalui tahapan *tokenizing* dan *filtering* langkah selanjutnya adalah proses *stemming*. Berikut proses *stemming* dengan algoritma *Enhanced Confix Stripping Stemmer*.

a. Langkah 1

Melanjutkan proses *stemming* yang dilakukan seperti tabel 3.3 diatas maka selanjutnya adalah melakukan pengecekan atau pencarian kedalam kamus umum. Kata – kata yang valid dengan kamus umum akan dikeluarkan dari algoritma, karena dianggap sudah merupakan kata dasar. Maka kata – kata tersebut akan menjadi seperti tabel 3.8 dibawah ini.

Tabel 3.8 Mengeluarkan kata dasar

a.|sebuah|bunyi|letusan|keras|menggemakan|memecahkan|keheningan|
b.|sebuah|letusan|menggemakan|memecahkan|keheningan|

b. Langkah 2

Jika masih terdapat kata – kata yang tidak sesuai bukan kata dasar langkah selanjutnya adalah hapus *inflection suffixes* yaitu akhiran pertama dalam kata yang termasuk *particles* dihilangkan seperti –lah, -kah, -tah, -pun. Setelah itu

menghilangkan akhiran yang termasuk *possesive pronouns* seperti –nya, -mu, -ku. Pada contoh kaliamt ini tidak terdapat *particles* maupun *possesive pronoun.* Maka kata-kata tersebut akan menjadi seperti tabel 3.9 dibawah ini.

Tabel 3.9 Menghapus akhiran(suffiks)

a.|sebuah|letusan|menggemakan|memecahkan|keheningan|

c. Langkah 3

Dilakukan pengecekan apa bila sudah valid dengan kamus maka kata tersebut dikeluarkan dari algortima, bila masih terdapat kata – kata yang belum valid dengan kamus. Langkah selanjutnya dalah menghapus derivation suffixes yaitu ,-i,-kan, atau, -an. Selanjutnya dilakukan pengecekan didalam sebuah kamus apabila sudah sesuai dengan kamus maka akan dikeluarkan dari proses. Berikutnya hapus derivation prefixes yaitu di-, ke-, me-, atau pe-. Maka kata – kata tersebut akan menjadi seperti pada tabel 3.10 berikut.

Tabel 3.10 Menghapus awalan (prefiks)

a.|sebuah|letusan|menggemakan|memecahkan|keheningan|
b.|sebuah|letus|menggema|memecah|kehening|
b.|buah| gema|pecah|hening|

d. Langkah 4

Langkah selanjutnya adalah melakukan pengecekan terhadap kata dengan kamus dan kata tersebut valid maka proses dihentikan dan kata tersebut merupakan kata dasar dan telah berhasil di *stemming* dengan algortima *Enhanced Confix Stripping Stemmer* seperti pada tabel 3.11 dibawah ini.

Tabel 3.11 Hasil *stemming* dengan algortima *Enhanced Confix Stripping Stemmer*

|buah|bunyi|letus|keras|gema|pecah|hening|

Perhitungan kekuatan *stemmer* ICF = (N-S)/N = (8-7)/8 = 0,125

3.5 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 dapat dilihat pada tabel 3.12 dibawah ini.

Tabel 3.12 Spesifikasi kebutuhan perangkat lunak fungsional

Kode	Kebutuhan
SKPL – F001	Sistem harus dapat mengambil dokumen dari direktori komputer pengguna.
SKPL – F002	Sistem harus dapat memproses kata yang diinputkan
SKPL – F003	Sistem harus dapat melakukan *stemming* kata dengan Algoritma Idris.
SKPL – F004	Sistem harus dapat melakukan *stemming* kata dengan Algoritma *ECS Stemmer*.
SKPL – F005	Sistem harus dapat menambahkan dokumen.
SKPL – F005	Sistem harus dapat menghapus dokumen.

Spesifikasi kebutuhan perangkat lunak *non* fungsional untuk perangkat lunak yang akan dibangun dapat dilihat pada tabel 3.13 berikut.

Tabel 3.13 Spesifikasi kebutuhan perangkat lunak *non* fungsional

Kode	Kebutuhan
SKPL – NF001	Perangkat lunak yang dibangun berbasis desktop.
SKPL – NF002	Prosessor yang digunakan minimal intel core i3.
SKPL – NF003	Memori komputer yang tersedia minimal 2GB.
SKPL – NF004	Sistem operasi yang digunakan minimal Win XP/7
SKPL – NF005	Pembangunan perangkat lunak menggunakan Microsoft Visual Studio 2012

3.6 Analisis Kebutuhan *Non* Fungsional

Analisis kebutuhan *non* fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk simulator. Spesifikasi kebutuhan melibatkan komponen-komponen yang dibutuhkan simulator sampai dengan simulator diimplementasikan. Analisis ini juga menentukan spesifikasi masukan yang diperlukan simulator, proses yang dibutuhkan untuk mengolah masukan, dan

keluaran yang dihasilkan oleh simulator. Kebutuhan *non* fungsional meliputi analisis perangkat keras dan analisis perangkat lunak.

3.6.1 Kebutuhan Perangkat Keras

Perangkat keras merupakan bagian dari komputer yang menerima instruksi dari perangkat lunak untuk menjalankan instruksi agar dapat menjalankan suatu sistem didalamnya. Adapun minimum kebutuhan perangkat keras yang digunakan untuk membangun simulator ini adalah sebagai berikut.

a. Prosessor : Intel Core i3 @1.8 GHz
b. Monitor : 14”
c. *Harddisk* : 160 GB
d. VGA : 512 MB
e. *Memory* : 2 GB
f. *Mouse*
g. *Keyboard*

3.6.2 Kebutuhan Perangkat Lunak

Perangkat lunak merupakan bagian dari komputer yang memberikan instruksi kepada perangkat keras untuk menjalankan instruksi agar dapat menjalankan suatu sistem didalamnya. Adapun minimum kebutuhan perangkat lunak yang digunakan untuk membangun simulator ini adalah sebagai berikut.

a. Sistem Operasi : Windows XP/7
b. Microsoft Visual Studio 2010
c. WAMP *Server*
d. *Notepad* ++
e. MySQL

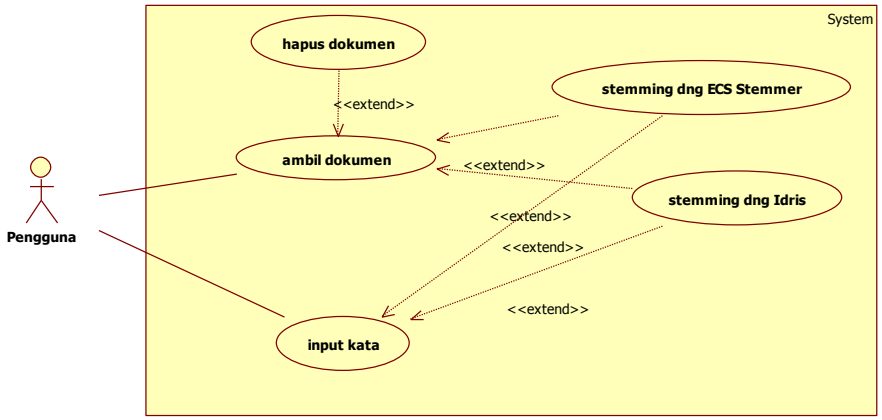
3.7 Analisis Kebutuhan Fungsional

Spesifikasi kebutuhan fungsional adalah spesifikasi tentang hal – hal yang akan dilakukan simulator ketika diimplementasikan. Analisis kebutuhan ini diperlukan untuk menentukan keluaran yang akan dihasilkan simulator, masukkan yang diperlukan simulator, lingkup proses yang digunakan untuk mengolah

masukan menjadi keluaran diantaranya *Usecase diagram, diagram scenario, activity diagram, sequence diagram, collaboration diagram* dan *class diagram.*

3.7.1 *Usecase Diagram*

Usecase diagram ini memperlihatkan pada kita hubungan antara aktor dengan *usecase* dalam simulator. *Usecase* diagram pada simulator yang akan dibangun dapat dilihat pada gambar 3.6 berikut.



Gambar 3.6 *Usecase Diagram*

3.7.1.1 Definisi Aktor

Bagian ini diisi dengan daftar aktor dan deksripsi *role* untuk aktor tersebut. Deksripsi *role* harus menjelaskan wewenang *role* tersebut dalam perangkat lunak. Bisa dibuat dalam bentuk tabel seperti pada tabel 3.14 dibawah ini:

Tabel 3.14 Definisi aktor

No	Aktor	Deskripsi
1	Pengguna	Pengguna menembahkan dokumen, pengguna menambahkan kata secara manual, pengguna me*refresh* dokumen dan pengguna melakukan perbandingan hasil algoritma Idris dan algoritma *Enhanced Confix Stripping Stemmer*.

3.7.1.2 Definisi *Usecase*

Bagian ini diisi dengan daftar *usecase* dan deskripsi singkat mengenai *usecase* pada simulator yang akan dibangun seperti pada tabel 3.15 dibawah ini.

Tabel 3.15 Deskripsi *usecase*

No	*Usecase*	Deskripsi
1	Ambil dokumen	Fungsionalitas yang digunakan pengguna untuk menambahkan dokumen kata didalam simulator.
2	Menginputkan kata	Fungsionalitas yang digunakan pengguna untuk menginputkan kata ke dalam simulator.
3	*Stemming* dengan Idris	Fungsionalitas yang digunakan pengguna untuk melakukan pengujian kata guna perbandingan dengan menggunakan algortima Idris.
4	*Stemming* dengan ECS *Stemmer*	Fungsionalitas yang digunakan pengguna untuk melakukan pengujian kata guna perbandingan dengan menggunakan algortima ECS *Stemmer*.
5	Menghapus dokumen	Fungsionalitas yang digunakan pengguna untuk menghapus dokumen kata yang sudah terdapat didalam simulator.

3.7.2 Diagram Skenario

Bagian ini diisi dengan skenario untuk setiap *usecase* pada rancangan yang menggambarkan urutan interaksi aktor dengan simulator seperti pada tabel 3.16, tabel 3.17, tabel 3.18, tabel 3.19 dan tabel 3.20 dibawah ini.

Tabel 3.16 Tambah dokumen

Use case Name	**Ambil dokumen**	
Related Requirements	Dokumen sudah tersedia dalam bentuk .txt atau .doc	
Goal In Context	Mengambil file berekstensi .txt atau .doc	
Preconditions	Menampilkan cari file	
Succsessful End Condition	Teks berhasil diproses	
Failed End Condition	Teks tidak berhasil diproses	
Primary Actors	Pengguna	
	Trigger	Aplikasi aktif
Main Flow	Step	Action

		Pengguna	Simulator
	1	Pengguna memilih dokumen yang akan diproses.	
	2		Mengambil dokumen berekstensi .txt atau .doc ke *directori* komputer
	3		Setelah dokumen berhasil dimasukkan maka simulator akan menghitung jumlah kata dan karakter dari dokumen
	4		sistem mengaktifkan pilihan untuk *stemming*
Extensions	Step	Branching Action	
	3	Memilih jenis dokumen yang tersedia	

Tabel 3.17 Menginputkan kata

Use case Name	**Menginputkan kata**		
Related Requirements	Pengguna menginputkan kata secara *manual*		
Goal In Context	Kata berhasil diinputkan		
Preconditions	Masuk ke menu kata		
Succsessful End Condition	Berhasil melakukan *stemming*		
Failed End Condition	Gagal melakukan *stemming*		
Primary Actors	Pengguna		
	Trigger	Aplikasi aktif	
Main Flow	Step	Action	
		Pengguna	Simulator
	1	Pengguna memasukkan kata yang ingin distemming	
	2		simulator menghitung jumlah karakter dan jumlah kata yang dimasukkan
	3		sistem mengaktifkan pilihan untuk *stemming*
Extensions	Step	Branching Action	

Tabel 3.18 *Stemming* dengan Idris

Use case Name	***Stemming* dengan Idris**		
Related Requirements	dokumen atau kata telah diinputkan		
Goal In Context	Dapat merubah suatu kata berimbuhan menjadi kata dasar sesuai dengan kamus KBBI		
Preconditions	Kata dasar telah tersedia didatabase		
Succsessful End Condition	Menampilkan kata – kata dasar		
Failed End Condition	tidak dapat menampilkan kata dasar		
Primary Actors	Pengguna		
	Trigger	Aplikasi aktif	
Main Flow	Step	Action	
		Pengguna	Simulator
	1	Pengguna memilih menstemming dokumen/ kata dengan algoritma ECS *Stemmer*	
	2		sistem memproses dokumen/kata dengan *stemming* algoritma ECS *Stemmer*.
	4		Melakukan proses *preprocessing.*
			Melakukan proses *tokenizing*
			Melakukan proses *stemming*
			kemudian melakukan perhitungan bobot terhadap kata - kata
			Simulator menampilkan hasil
Extensions	Step	Branching Action	

Tabel 3.19 *Stemming* dengan ECS *Stemmer*

Use case Name	***Stemming* dengan ECS *Stemmer***		
Related Requirements	dokumen atau kata telah diinputkan		
Goal In Context	Dapat merubah suatu kata berimbuhan menjadi kata dasar sesuai dengan kamus KBBI		
Preconditions	Kata dasar telah tersedia didatabase		
Succsessful End Condition	Menampilkan kata – kata dasar		
Failed End Condition	tidak dapat menampilkan kata dasar		
Primary Actors	Pengguna		
	Trigger	Aplikasi aktif	
Main Flow	Step	Action	
		Pengguna	Simulator
	1	Pengguna memilih menstemming dokumen/ kata dengan algoritma ECS *Stemmer*	
	2		sistem memproses dokumen/kata dengan *stemming* algoritma ECS *Stemmer.*
	3		Melakukan proses *preprocessing.*
			Melakukan proses *tokenizing*
			Melakukan proses *stemming*
			kemudian melakukan perhitungan bobot terhadap kata - kata
			Simulator menampilkan hasil
Extensions	Step	Branching Action	

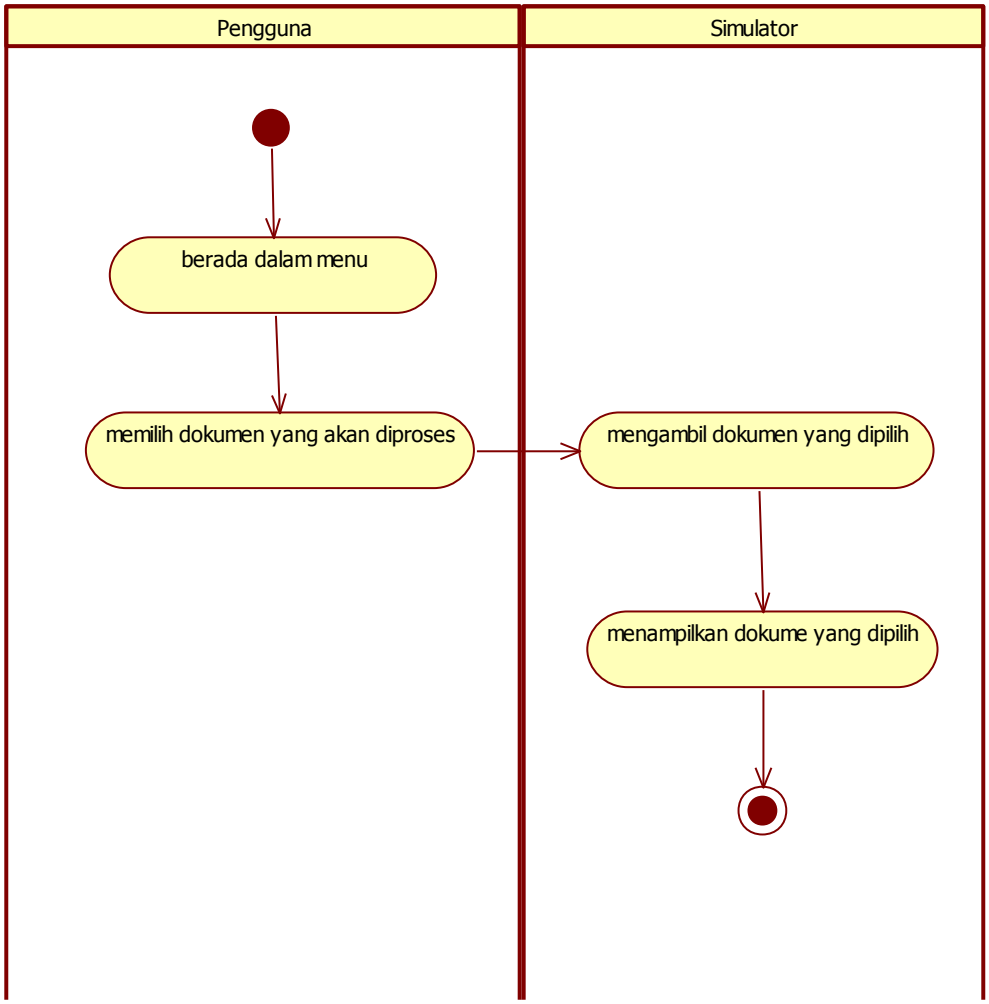
Tabel 3.20 Hapus Dokumen

Use case Name	**Hapus Dokumen**		
Related Requirements	Dokumen yang akan dihapus sudah tersedia disimulator		
Goal In Context	Melakukan proses hapus dari dokumen inputan		
Preconditions	Proses hapus berhasil dilakukan		
Succsessful End Condition	Proses hapus berhasil dilakukan		
Failed End Condition	Proses hapus gagal dilakukan		
Primary Actors	Pengguna		
	Trigger	Aplikasi aktif	
Main Flow	Step	Action	
		Pengguna	Simulator
	1	pengguna memilih hapus	
	2		simulator menghapus data masukkan
Extensions	Step	Branching Action	
	1.1	Menampilkan kata yang telah distemming	
	2.1	Data tidak berhasil terhapus	

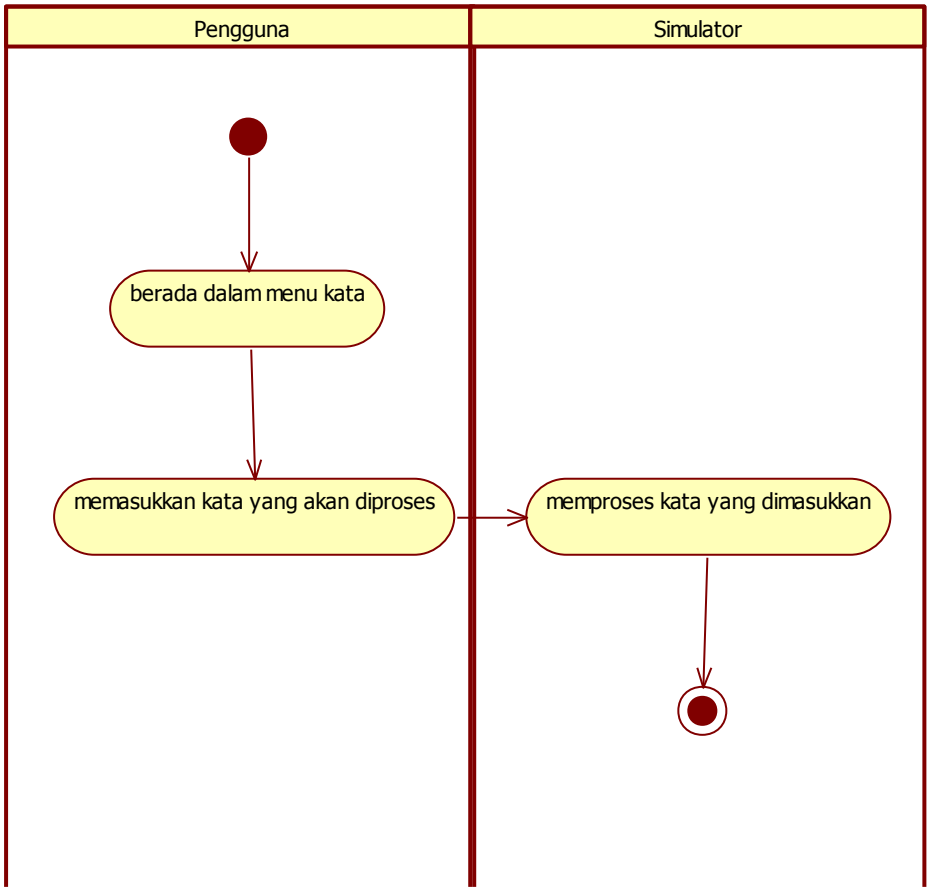
3.7.3 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. *Activity diagram* merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*).

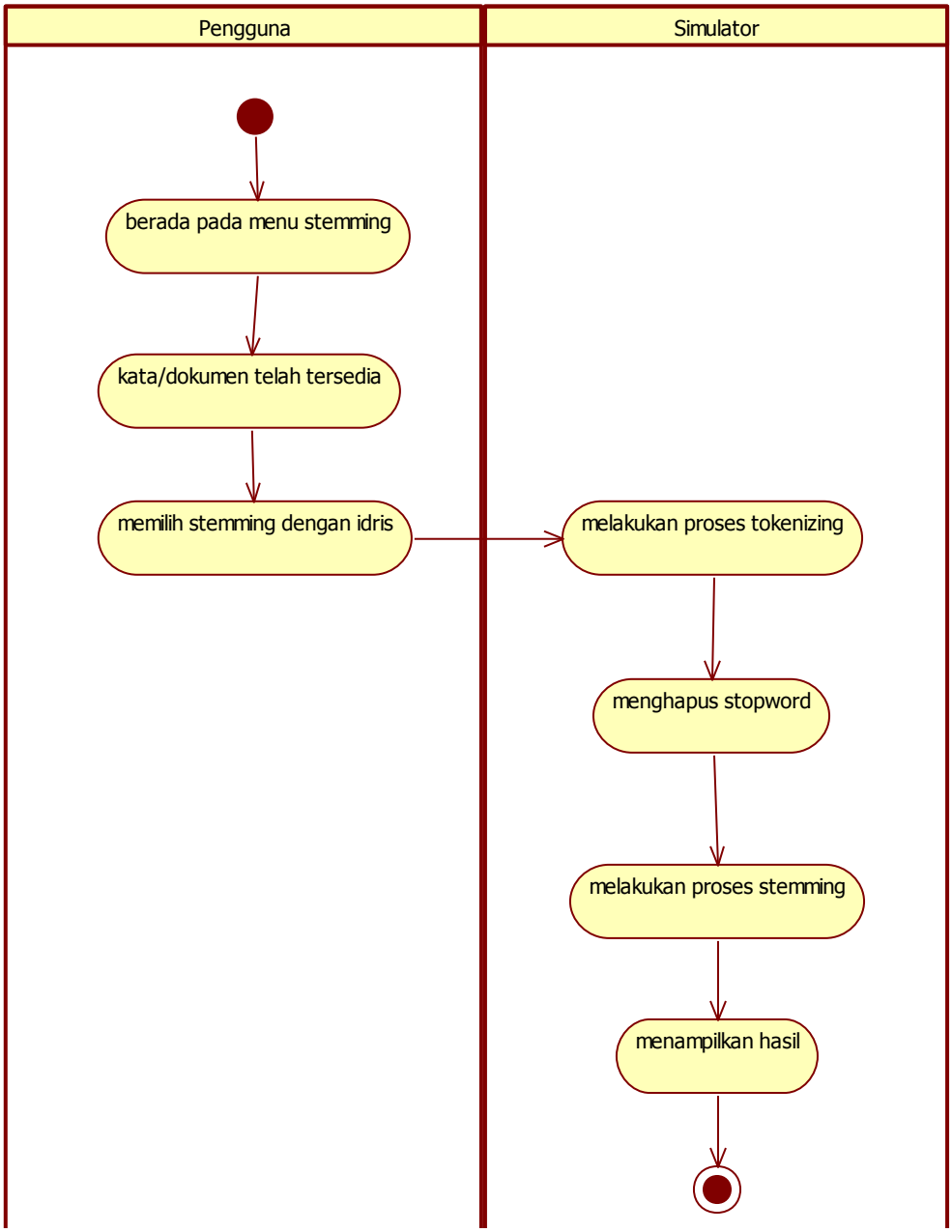
Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Adapun diagram aktivitas pada simulator yang akan dibangun dapat dilihat pada gambar 3.7, gambar 3.8, gambar 3.9, gambar 3.10 dan gambar 3.111 sebagai berikut.



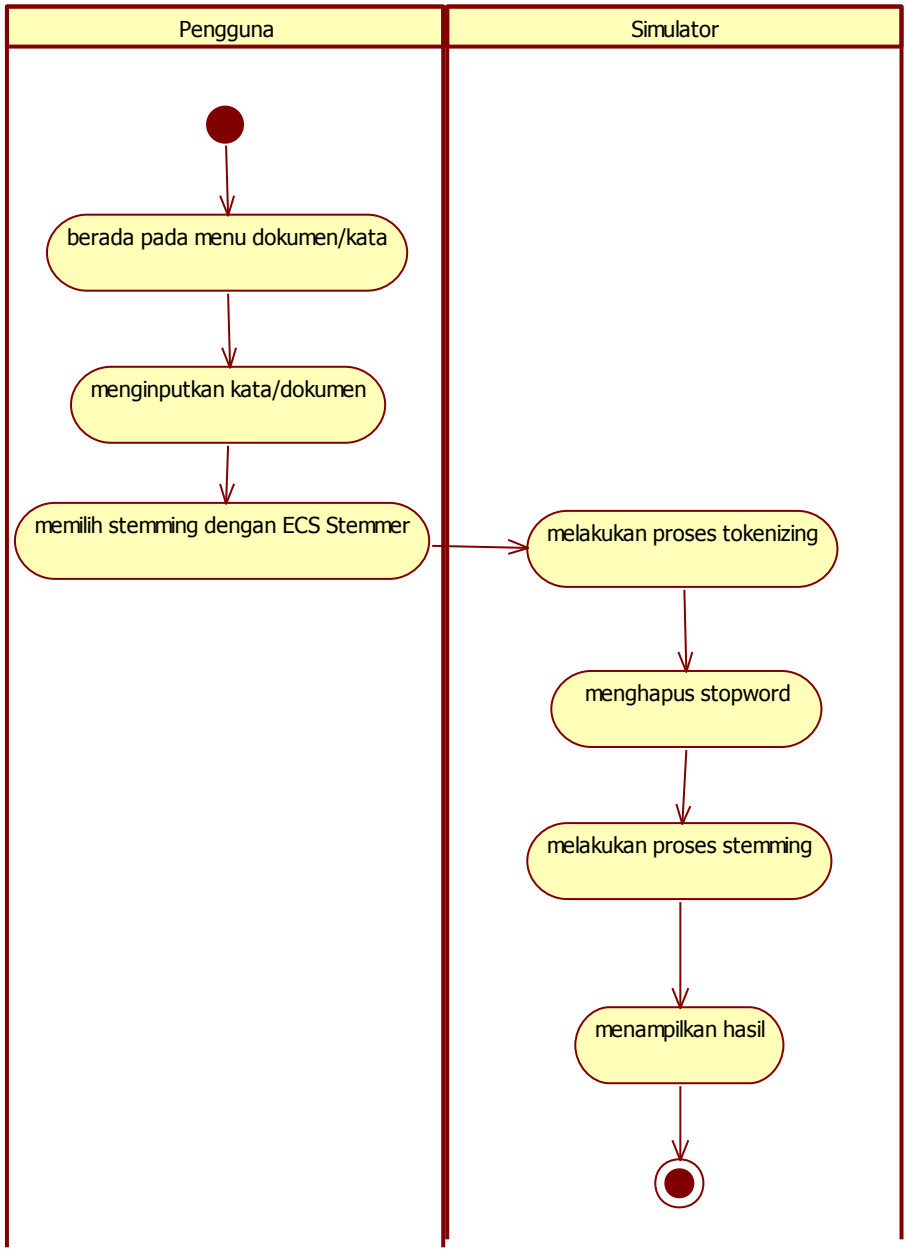
Gambar 3.7 *Activity diagram* tambah dokumen



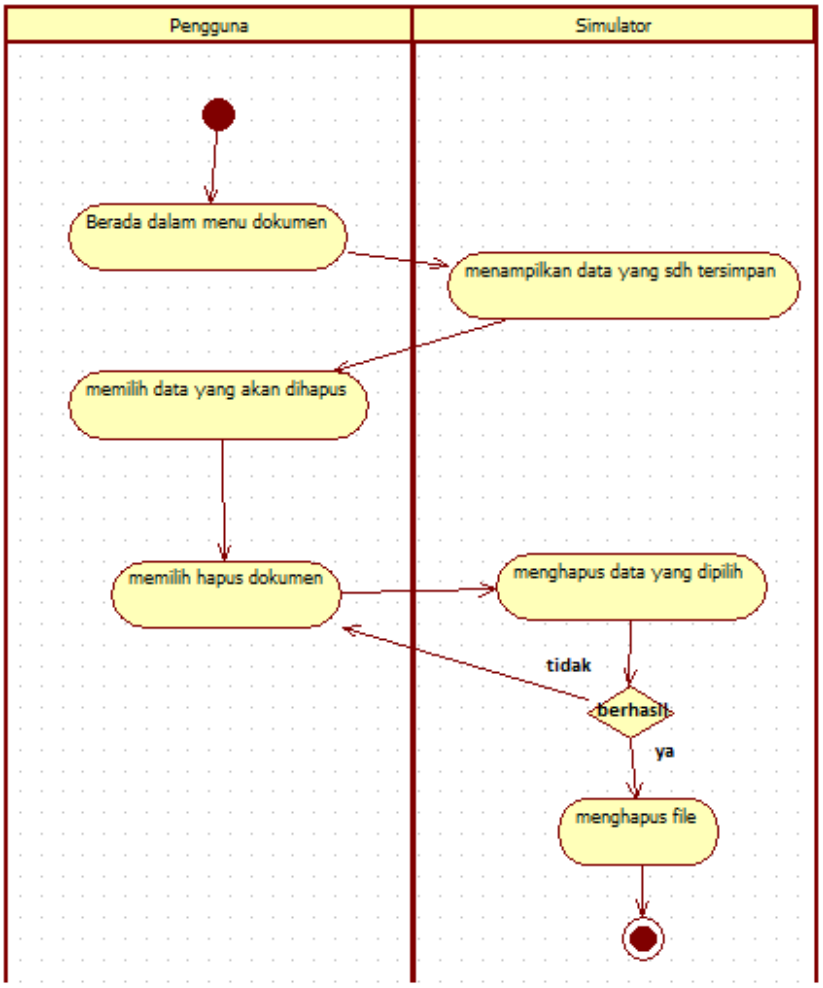
Gambar 3.8 *Activity diagram* menginputkan kata



Gambar 3.9 *Activity diagram* algortima Idris



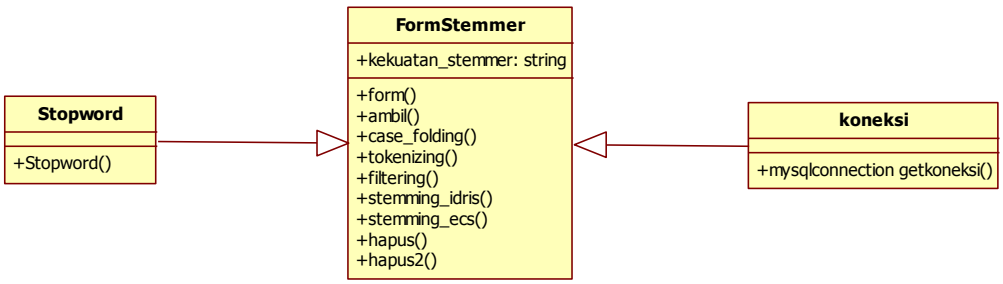
Gambar 3.10 *Activity diagram* algortima ECS *Stemmer*



Gambar 3.11 *Activity diagram* hapus dokumen

3.7.4 *Class Diagram*

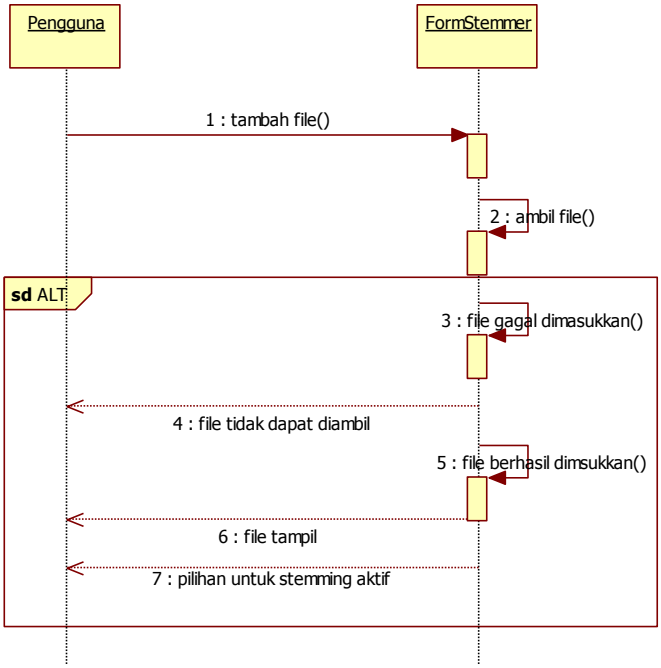
Class diagram adalah diagam yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita gunakan. *Class diagram* memberi kita gambaran (diagram statis) tentang sistem/perangkat lunak dan relas-relasi yang ada didalamnya. Berikut adalah *class diagram* dari simulator yang dibangun dapat dilihat pada gambar 3.12.



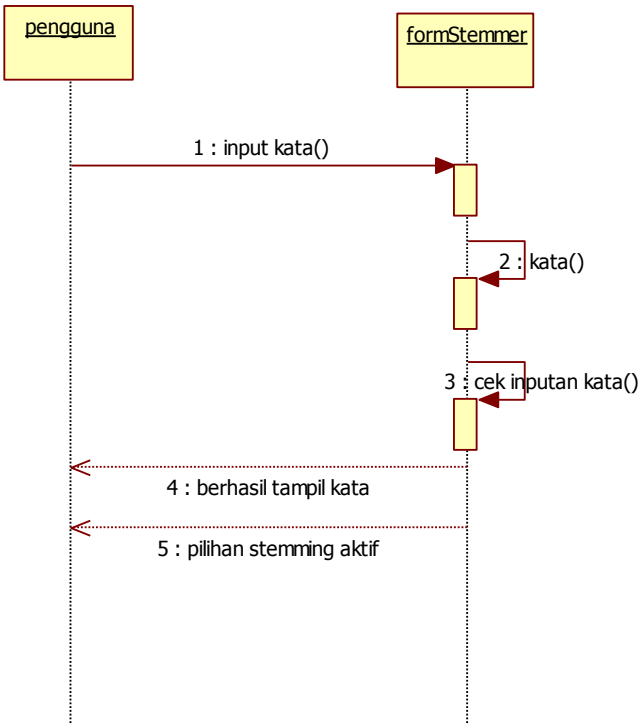
Gambar 3.12 *Class diagram*

3.7.5 *Sequance Diagram*

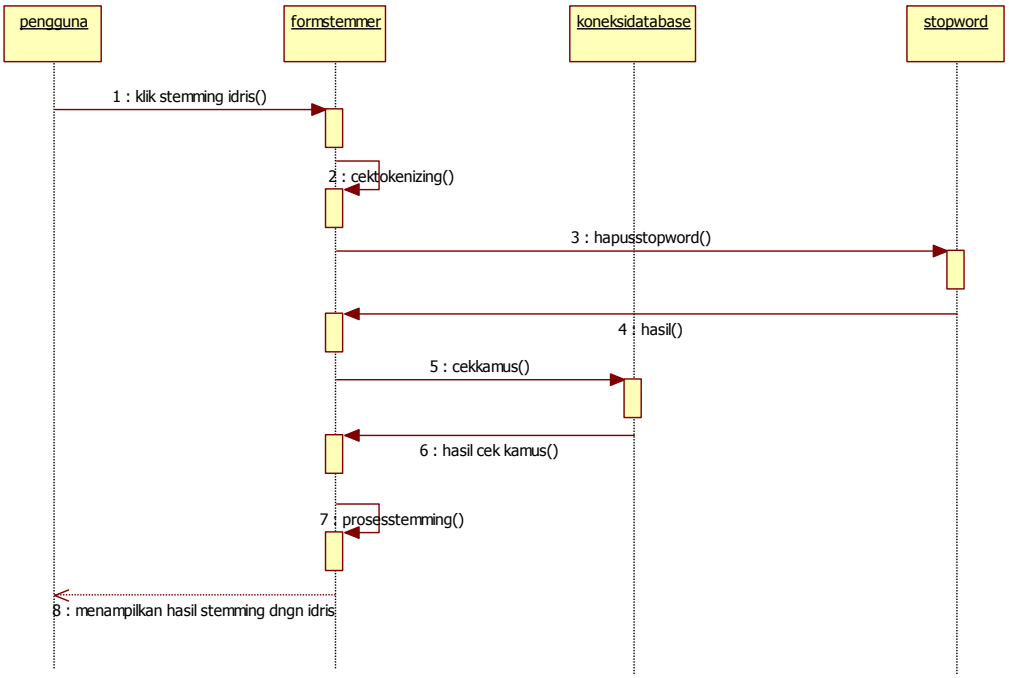
Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram. Berikut merupakan *Sequence diagram* dari simulator yang akan dibangun dapat dilihat pada gambar 3.13, gambar 3.14, gambar 3.15, gambar 3.16 dan gambar 3.17.



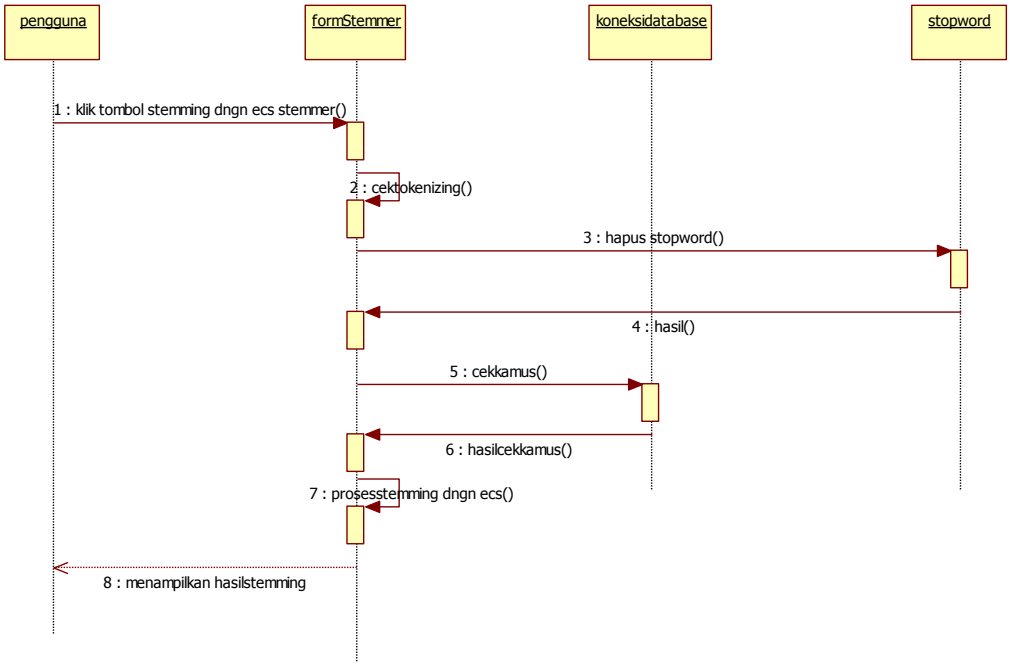
Gambar 3.13 *Sequence diagram* ambil dokumen



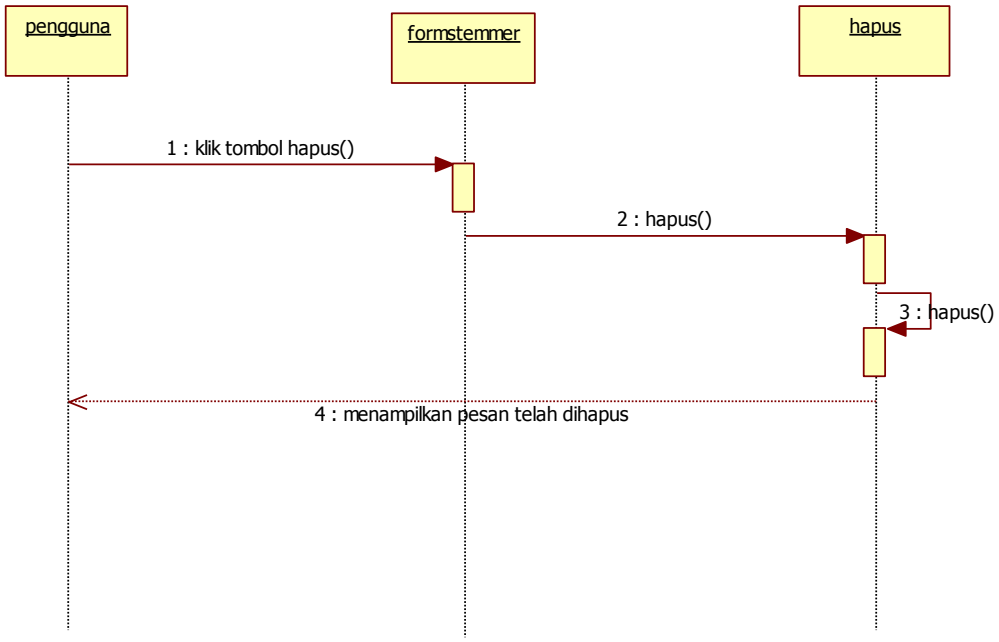
Gambar 3.14 *Sequence diagram input* kata



Gambar 3.15 *Sequence diagram stemming* dengan algoritma Idris



Gambar 3.16 *Sequence diagram stemming* dengan algortima ECS *Stemmer*



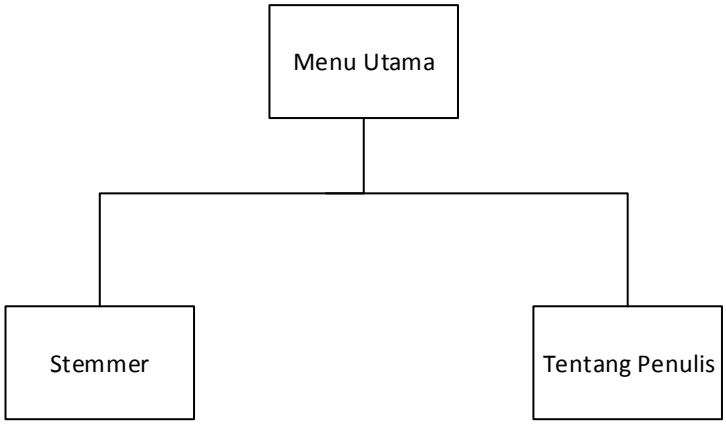
Gambar 3.17 *Sequence diagram* hapus

3.8 Perancangan Sistem

Perancangan sistem adalah rancangan – rancangan mengenai sistem yang akan dibangun yang terdiri dari rancangan menu, rancangan antarmuka sistem, rancangan antarmuka pesan dan jaringan semantik.

3.8.1 Rancangan Menu

Rancangan menu dari simulator yang akan dibangun adalah sebagai berikut ditunjukkan pada gambar 3.18 dibawah ini.



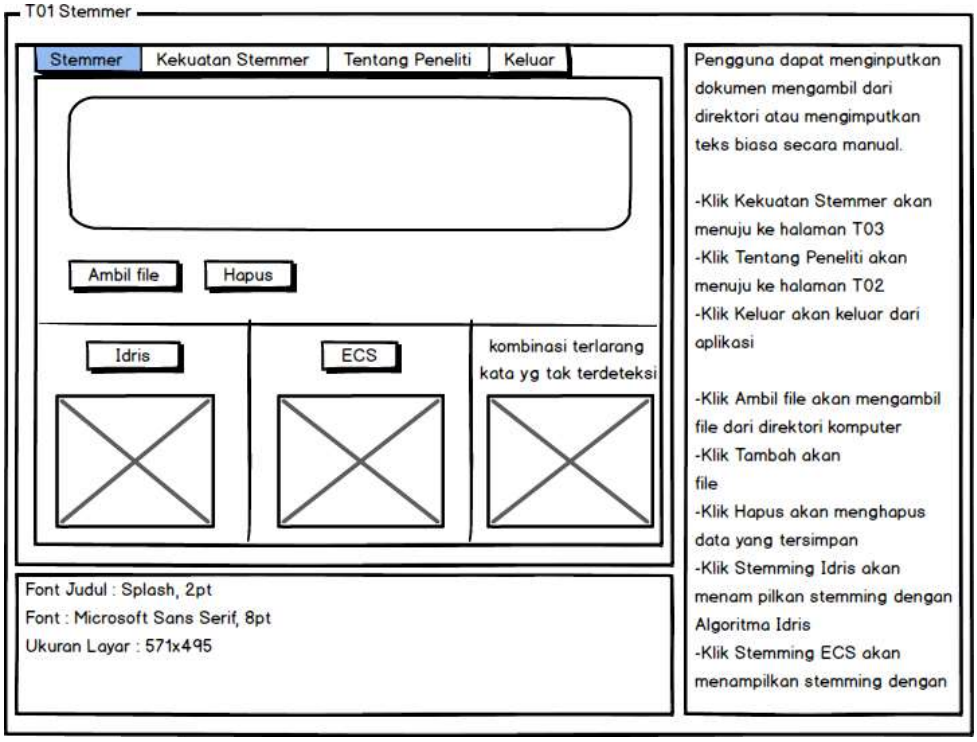
Gambar 3.18 Rancangan Menu

3.8.2 Rancangan Antarmuka

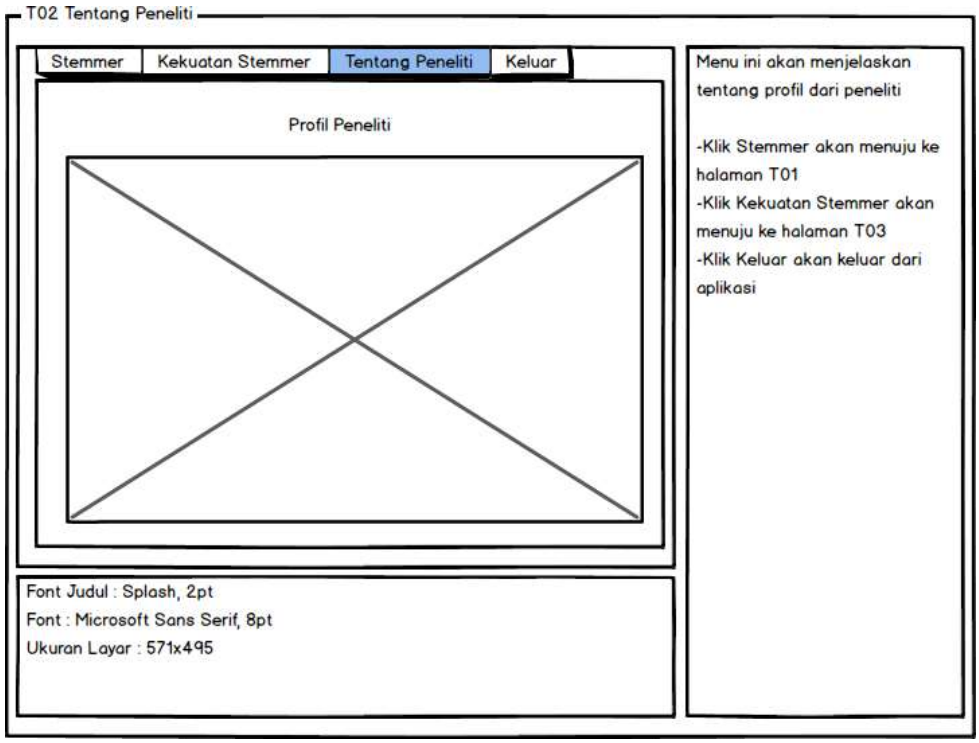
Rancangan antarmuka dari simulator yang akan dibangun adalah sebagai berikut dapat dilihat pada gambar 3.19, gambar 3.20, gambar 3.21 dan gambar 3.22.



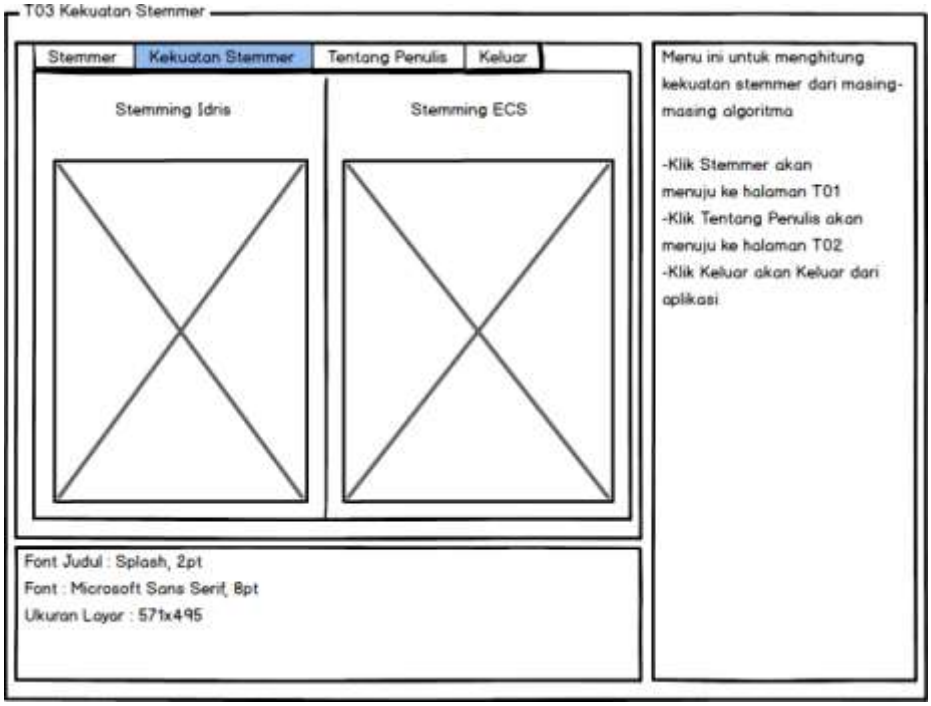
Gambar 3.19 Tampilan Menu Utama (T00)



Gambar 3.20 Tampilan Menu *Stemmer* (T01)



Gambar 3.21 Tampilan Menu Tentang Peneliti (T02)



Gambar 3.22 Tampilan menu Kekuatan *Stemmer* (T03)

3.8.3 Rancangan Antarmuka Pesan

Antarmuka pesan dari simulator yang akan dibangun dapat dilihat pada gambar 3.23, gambar 3.24, gambar 3.25 dan gambar 3.26 dibawah ini



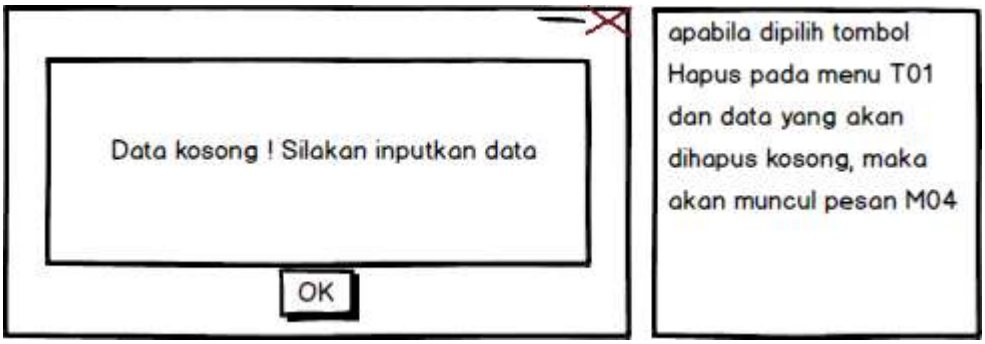
Gambar 3.23 Pesan akan menghapus data



Gambar 3.24 Pesan data berhasil dihapus



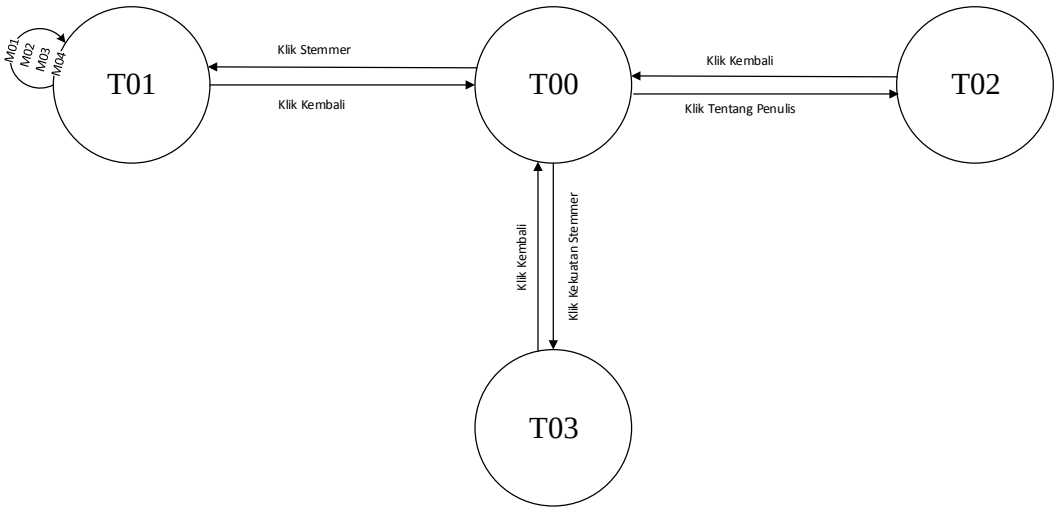
Gambar 3.25 Pesan gagal mengambil *file* dari direktori komputer



Gambar 3.26 Pesan data yang diinputkan kosong

3.9 Jaringan Semantik

Jaringan semantik merupakan pengetahuan secara grafis yang menunjukkan hubungan antara berbagai objek. Jaringan semantik dari simulator yang akan dibangun pada gambar 3.27 dibawah ini.



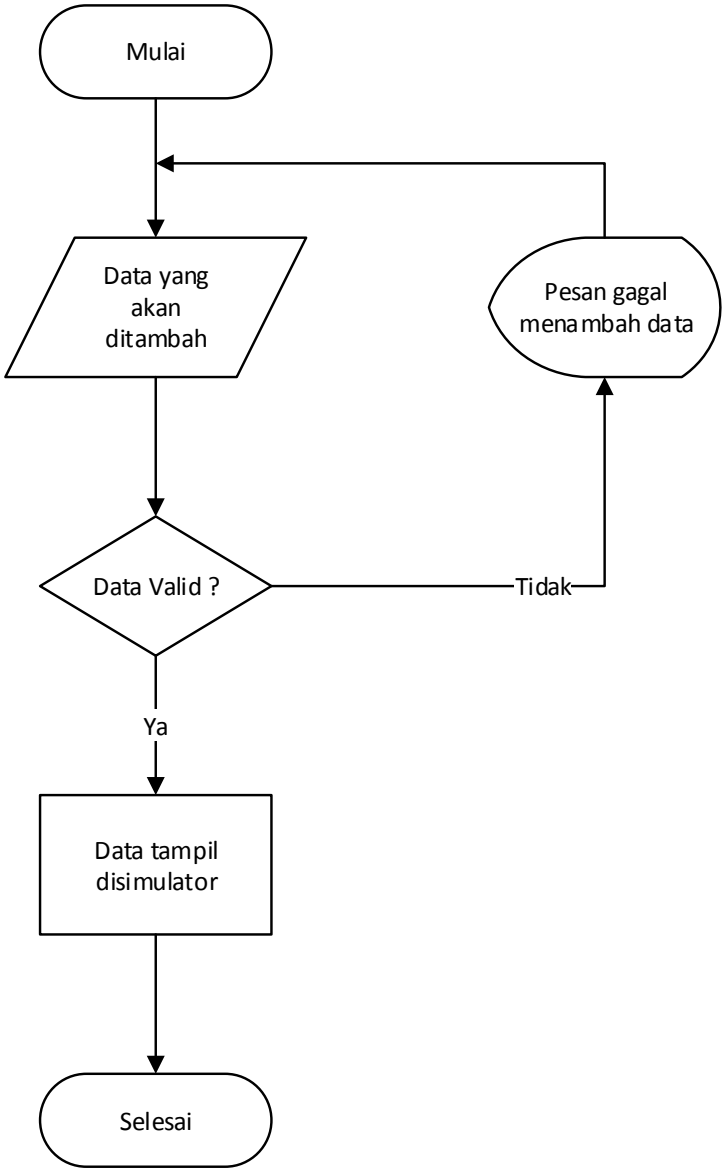
Gambar 3.27 Jaringan Semantik

3.10 Perancangan *Method*

Perancangan *Method* merupakan kumpulan metode yang ada pada simulator yang dirancang dan berfungsi untuk menjalankan simulator. Berikut adalah perancangan method dari simulator yang akan dibangun dapat dilihat pada gambar 3.28, gambar 3.29, gambar 3.30, gambar 3.31, gambar 3.32 dan gambar 3.33.

1. Perancangan *Method* Tambah

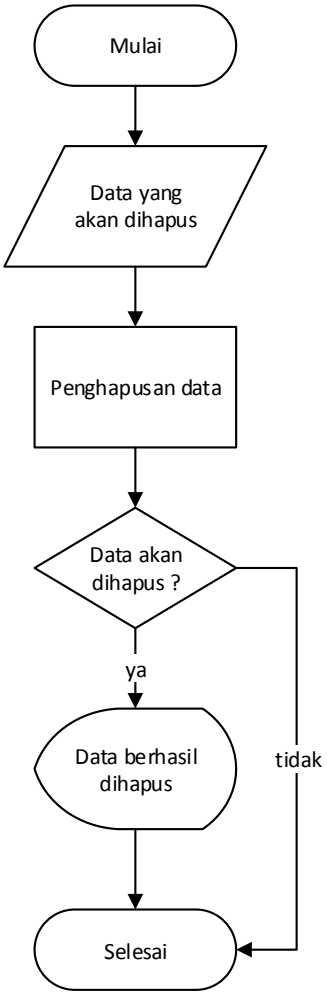
   Method ini dijalankan untuk dapat menambahkan *file* atau menambahkan kata pada simulator. Untuk lebih jelasnya dapat dilihat pada gambar 3.28 berikut.



Gambar 3.28 Perancangan *method* Tambah

2. Perancangan *Method* Hapus

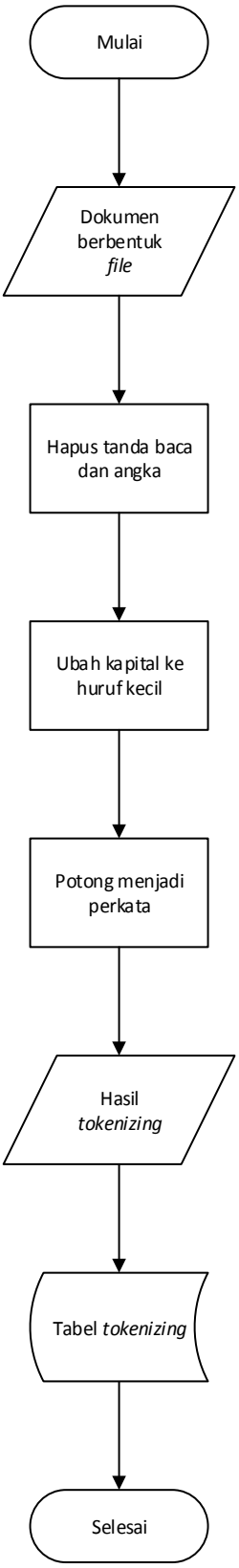
   Method ini dijalankan untuk dapat menghapus *file* pada simulator. Untuk lebih jelasnya dapat dilihat pada gambar 3.29.



Gambar 3.29 Perancangan *method* Hapus

3. Perancangan *Method Tokenizing*

   Method ini dijalankan untuk melakukan proses *tokenizing* atau penghapusan tanda baca yang tedapat pada *file* atau kata yang diinputkan. Untuk lebih jelasnya dapat dilihat pada gambar 3.30 berikut.

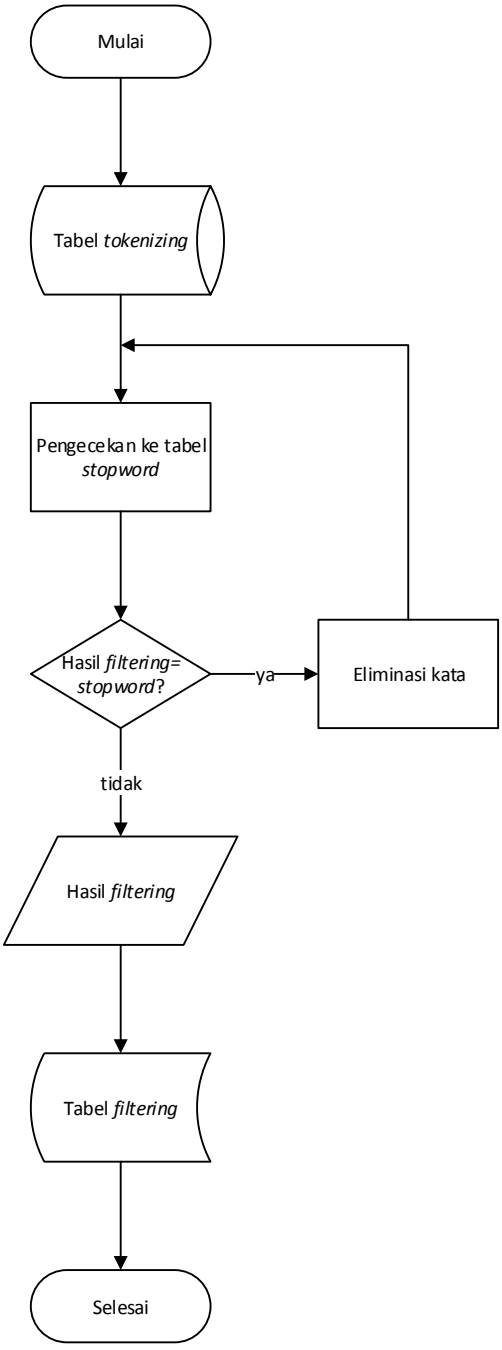


Gambar 3.30 Perancangan *method Tokenizing*

4. Perancangan *Method Filtering*

   Method ini dijalankan untuk melakukan proses *filtering* atau penghapusan *stopword* yang tedapat pada *file* atau kata yang diinputkan. Untuk lebih jelasnya dapat dilihat pada gambar 3.31 berikut.

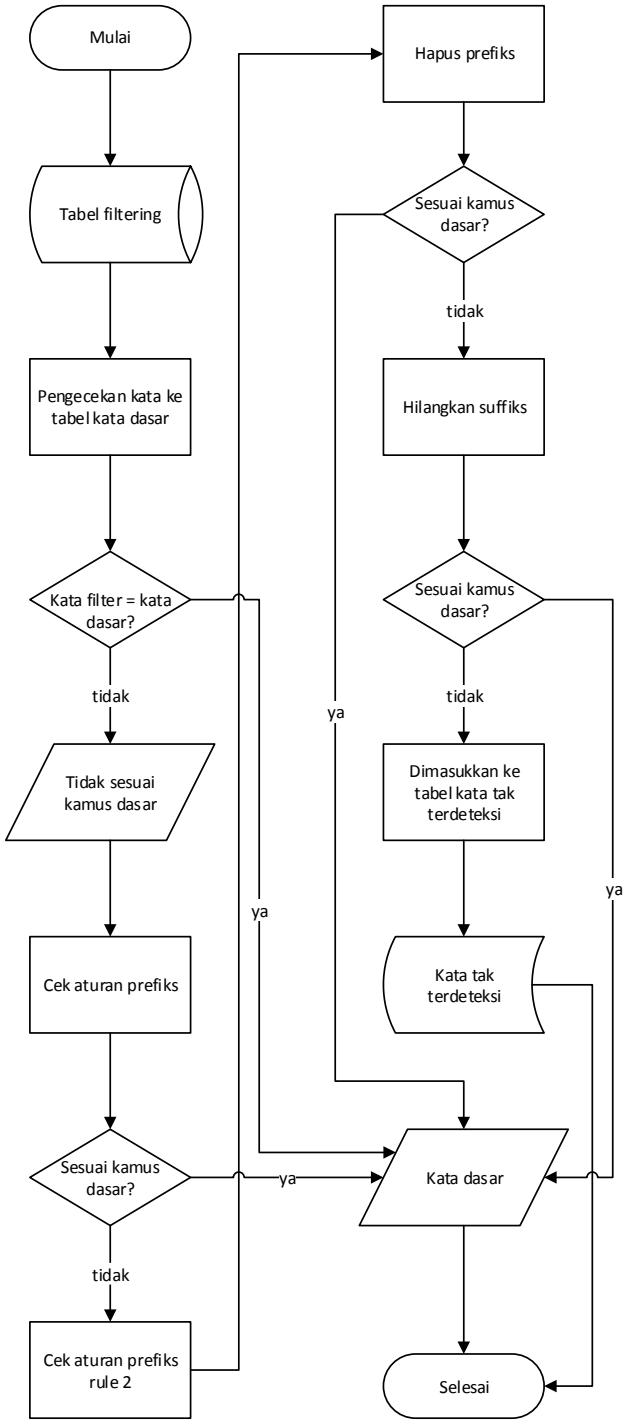
   Contoh *stopword* : yang, dan, atau.



Gambar 3.31 Perancangan *method Filtering*

5. Perancangan *Method Stemming* dengan Algortima Idris

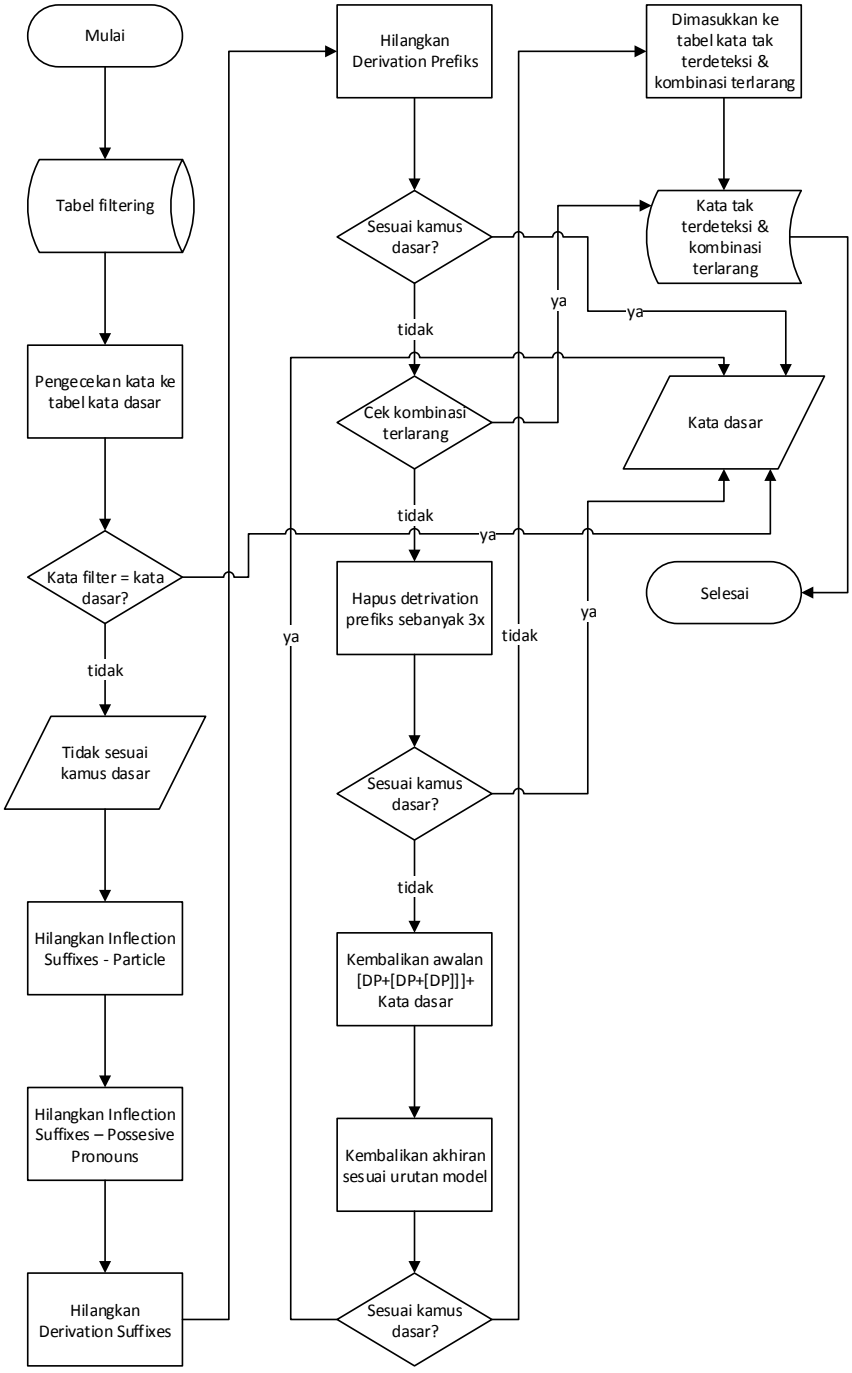
   Method ini dijalankan untuk melakukan proses *stemming* yaitu pemenggalan kata yang berimbuhan menjadi kata dasar. Untuk lebih jelasnya dapat dilihat pada gambar 3.32 berikut.



Gambar 3.32 Perancangan *method stemming* dengan Algoritma Idris

6. Perancangan *Method Stemming* dengan Algoritma *ECS Stemmer*

   Method ini dijalankan untuk melakukan proses *stemming* yaitu pemenggalan kata yang berimbuhan menjadi kata dasar. Untuk lebih jelasnya dapat dilihat pada gambar 3.31 berikut.



Gambar 3.33 Perancangan *method stemming* dengan Algoritma ECS *Stemmer*