BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Katalog adalah daftar koleksi buku, aplikasi katolog bisa disusun berdasarkan judul buku. Katalog merupakan kumpulan buku-buku yang sudah masuk kedalam sistem, aplikasi katalog sendiri mempunyai sebuah fitur pencarian yang dimana didalam fitur ini bekerja melalui query yang diinputkan oleh pengguna untuk menemukan buku yang sesuai, ketika pencarian buku telah menemukan hasil yang sesuai dengan yang dicari oleh pengguna maka sistem akan memberikan sebuah refrensi buku yang lain tetapi masi berkaitan dengan buku yang dicari oleh pengguna dalam hal memberikan sebuah refrensi buku terkait aplikasi ini menggunakan sebuah metode Vector Space Model.

Vector Space Model (VSM) adalah suatu model yang digunakan untuk mengukur kemiripan antara suatu dokumen dengan suatu query. Query dan dokumen dianggap sebagai vektor-vektor pada ruang n-dimensi, dimana t adalah jumlah dari seluruh term yang ada dalam leksikon. Leksikon adalah daftar semua term yang ada dalam indeks. Selanjutnya akan dihitung nilai cosinus sudut dari dua vektor. Aplikasi pencarian buku berdasarkan keyword yang diteliti menggunakan metode *cosine similarity* untuk perhitungan nilai kemiripan deskripsi. Sedangkan untuk tahapan *text preprocessing* deskripsi menggunakan *text mining* dengan beberapa tahapan seperti *case folding, stopword, tagging, stemming, filtering* dan *analyzing.* Algoritma nazief adriani digunakan dalam proses *stemming* untuk mengembalikan kata berimbuhan menjadi kata dasar. Dalam melakukan perhitungan nilai kemiripan deskripsi kemudian dilakukan suatu pembobotan kata pada deskripsi dalam hal ini menggunakan pembobotan *binary term* untuk setiap kata pada deskripsi. Pembobotan kata merupakan tahapan awal dalam menentukan nilai kemiripan dari sebuah deskripsi dengan melakukan *skoring* dan sangat berpengaruh terhadap nilai *similarity* yang didapatkan. Penelitian ini menggunakan *binary term* atau disebut tipe *booelan* yang bernilai 0 dan 1. Setiap kata direpresentasikan dengan nilai 0 jika terdapat

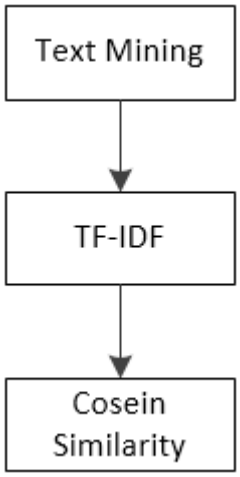
kata yang tidak ada pada deskripsi tersebut dan bernilai 1 jika terdapat kata yang sama pada deskripsi tersebut. Dari *skoring* kata pada masing deskripsi akan di akumulasikan untuk dilakukan perhitungan *similarity* dengan metode *cosine similarity.*

3.1.1 Hasil Analisis Sistem

Aplikasi pencarian buku berdasarkan keyword adalah suatu aplikasi yang nantinya akan mencari nilai *similarity* dari beberapa deskripsi dan juga mencari letak kemiripan deskripsi berdasarkan sebuah kata. Proses dalam menentukan kemiripan deskripsi dilakukan dengan menggunakan sebuah deskripsi yang berisi teks dengan beberapa ketentuan meliputi :

- Teks berbahasa Indonesia.
- Teks memiliki aturan kata sesuai EYD.
- Deskripsi yang akan diolah berupa teks

3.1.2 Vector Space Model

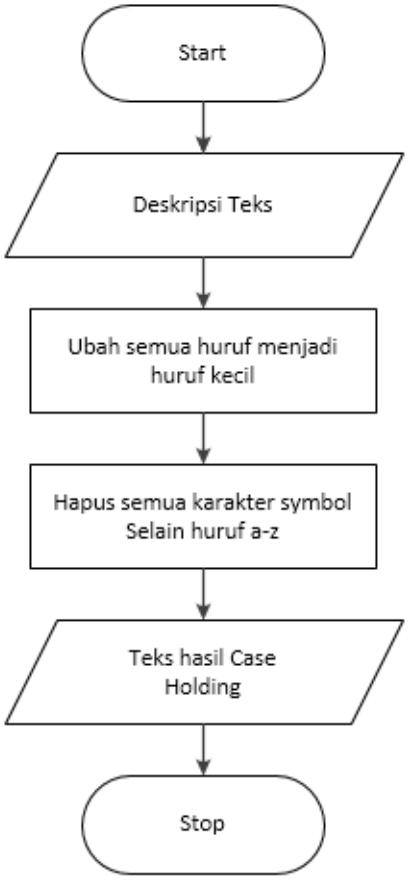


Gambar 3.1 Alur metode Vector Space Model

Text Mining dengan beberapa tahapan sebagai berikut :

1. Case Folding dan Tokenizing.

   Pengertian *Case Folding* dan *Tokenizing* sudah dijelaksan pada bab sebelumnya dapat dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *case folding* dan tokenizing pada gambar 3.2



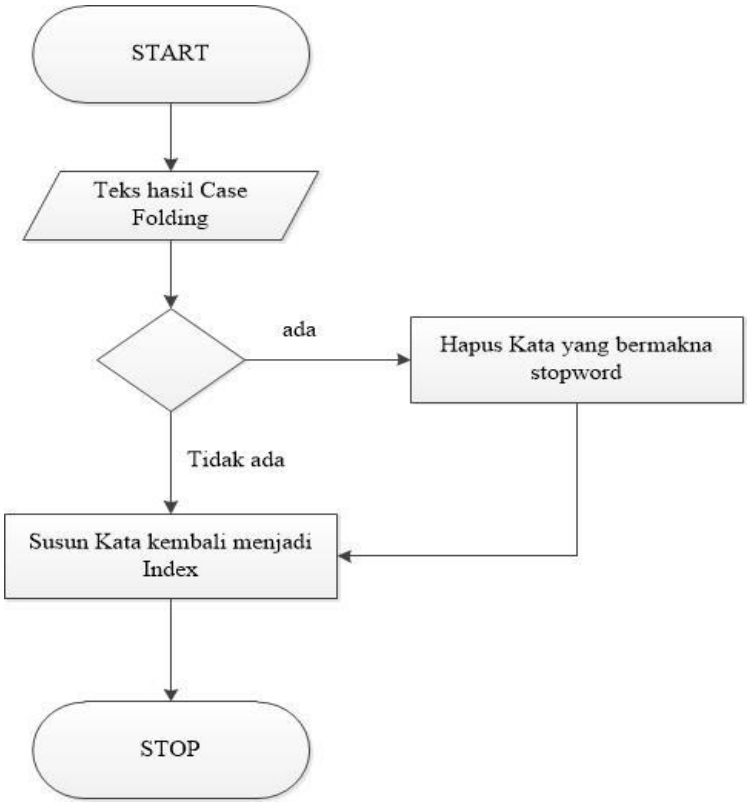
Gambar 3.2 Alur Flow Case Folding dan Tokenizing

Keterangan :

- Memulai program.
- Memasukan deskripsi teks
- Melakukan penyeragaman kata dengan mengubah huruf menjadi huruf kecil.
- Menghapus semua karakter symbol dan lain sebagainya kecuali huruf a-z.
- Output dari *case folding* dan *tokeninzing.*
- Selesai

2. Filtering

   Pengertian *Filtering* sudah dijelaksan pada bab sebelumnya bisa dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *filtering* dapat dilihat pada Gambar 3.3



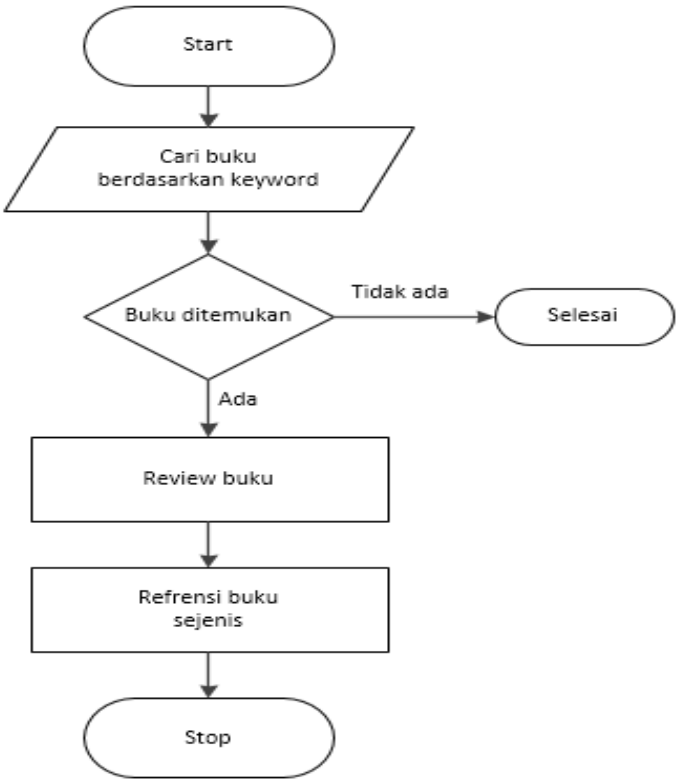
Gambar 3.3 Alur Flow Filtering

Keterangan :

- Memulai program.
- Output dari *case folding* akan diolah kembali untuk dicek kata-kata yang terdapat makna *stopword.*
- Cek kata stopword jika ditemukan hapus kata tersebut jika tidak susun kembali kata menjadi sebuah index.
- Selesai.

3. Stemming Algoritma Nazief & Adriani

Stemming Nazief dan Adriani dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (prefix), sisipan (infix), akhiran (suffix) dan gabungan awalan akhiran (confixes). Algoritma ini menggunakan kamus kata dasar dan mendukung recoding, yakni penyusunan kembali kata-kata yang mengalami proses stemming berlebih**.** Untuk lebih jelasnya mengenai *stemming* sudah dijelaksan pada bab sebelumnya dapat dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *stemming* dengan algoritma nazief dan adriani dapat dilihat pada Gambar 3.4



Gambar 3.4 Stemming dengan Algortitma Nazief dan Adriani

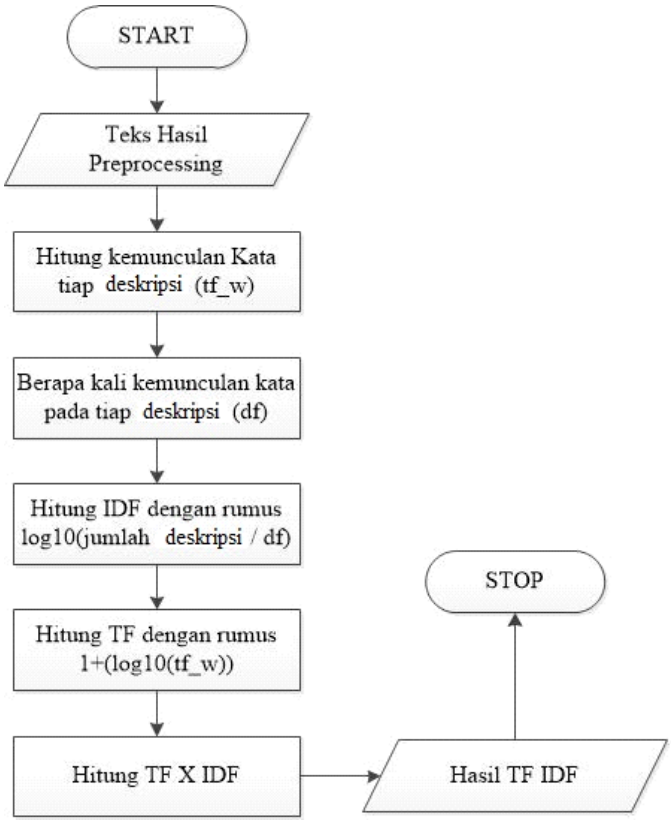
Keterangan :

1. Cari kata dalam kamus jika ditemukan maka diasumsikan bahwa kata tersebut adalah kata dasar. Algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 2.

2. Hilangkan *inflectional suffixes* bila ada. Dimulai dari *inflectional particle* (“-lah”, “-kah”, “-tah” dan “-pun”), kemudian *possessive pronoun* (“-ku”, “-mu” dan “-nya”). Cari kata pada kamus jika ditemukan algoritma berhenti, jika kata tidak tidak ditemukan dalam kamus lakukan langkah 3.
3. Hilangkan *derivation suffixes* (“-an”, “-i” dan “-kan”). Jika akhiran “-an” dihapus dan ditemukan akhiran “-k”, maka akhiran “-k” dihapus. Cari kata pada kamus jika ditemukan algoritma berhenti, jika kata tidak tidak ditemukan maka lakukan langkah 4.
4. Pada langkah 4 terdapat tiga iterasi.
   1) Iterasi berhenti jika :
      a Ditemukannya kombinasi awalan yang tidak diizikan berdasarkan awalan
      b Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      c Tiga awalan telah dihilangkan.
   2) Identifikasikasikan tipe awalan dan hilangkan. Awalan terdiri dari dua tipe:
5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recording* dilakukan dengan mengacu pada aturan tabel 2.4. *Recording* dilakukan dengan menambahkan karakter *recording* di awal kata yang dipenggal. Pada tabel 2.4, karakter *recording* adalah huruf kecil setelah tanda hubung (‘-‘) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 15) pada tabel 2.4, setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka *recording* dilakukan dan menghasilkan kata “tangkap”.
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini di anggap sebagai kata dasar.

4. Pembobotan TF-IDF

TF-IDF (Term Frequency-Inversed Document Frequency) merupakan salah satu algoritma pembobotan sebuah kata seperti yang dijelaskan pada bab sebelumya bisa dilihat pada bab 2 landasan teori mengenai pengertian serta rumus dalam pembobotan TF-IDF. Dalam hal ini pembobotan TF-IDF memiliki alur pembobotan dapat dilihat pada Gambar 3.5



Gambar 3.5 Alur Flow Pembobotan TF-IDF

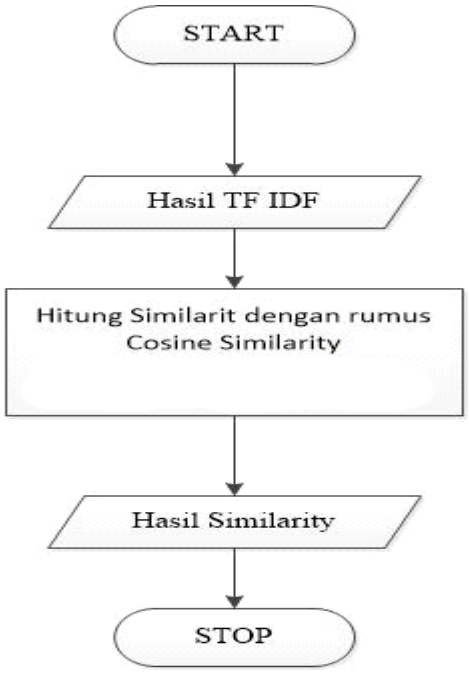
Keterangan :

- Mulai.
- Ambil kata hasil *text preprocessing* dalam *database.*
- Hitung jumlah kemunculan kata tiap deskripsi dengan disimbolkan sebagai (*tf_w*).
- Hitung berapa kali kemunculan kata yang sama pada tiap deskripsi dengan disimbolkan sebagai (*df).*

- Hitung *idf* dengan rumus *log10(* jumlah deskripsi / *df).*
- Hitung *tf* dengan rumus 1+*log10(tf_w).*
- Hitung *tf x idf.*
- Hasil nilai
- Berhenti

5. Metode Cosine Similarity

Cosine Similarity merupakan salah satu metode untuk mencari suatu nilai *similarity* pada suatu dokumen dengan menggunakan pemodelan perhitungan vektor yang pada bab sebelumnya sudah dijelaskan bisa dilihat pada bab 2 landasan teori mengenasi pengertian metode *cosine similarity* beserta rumusnya dapat dilihat pada bab 2 sub bab 2.5 *Cosine Similarity* sedangkan untuk bab ini akan dijelaskan alur flow Metode *cosine similarity* seperti terlihat pada Gambar 3.6



Gambar 3.6 Alur Flow Metode Cosine Similarity

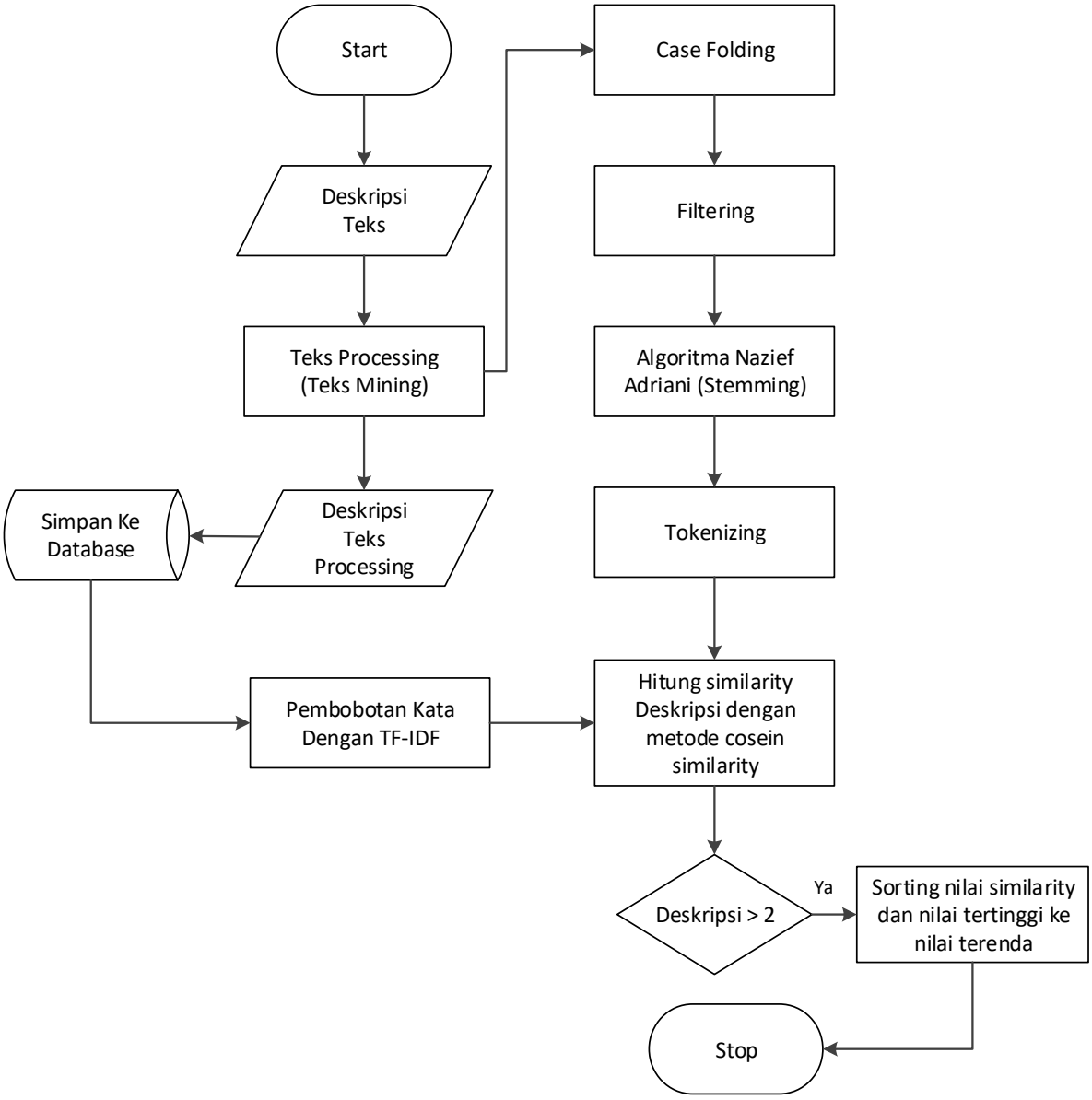
Keterangan Gambar 3.6 :

1. Memulai program.
2. Mengambil hasil nilai pembobotan *tf-idf.*
3. Lakukan perhitugan sesuai dengan rumus.
4. Hasil perhitungan.
5. Selesai.

Sistem ini dapat digunakan untuk semua pihak yang membutuhkan informasi letak kemiripan kata dari deskripsi. Berdasarkan hasil analisa sistem dapat diketahui kebutuhan fungsional dari aplikasi deteksi kemiripan deskripsi yang akan dibuat nantinya antara lain sebagai berikut :

1. Melakukan penyeragaman kata dengan mengubah seluruh kata menjadi huruf kecil biasa disebut *case-folding.*
2. Melakukan *filtering* terhadap kata-kata yang tidak penting dalam hal ini biasa disebut *stopword*
3. Mengubah kata yang memiliki imbuhan dalam Bahasa Indonesia menjadi sebuah kata dasar istilahnya adalah *stemming* menggunakan Algoritma Nazief dan Adriani.
4. Melakukan pembobotan kata dari deskripsi dengan TF-IDF.
5. Melakukan sebuah penilaian similarity deskripsi dengan *cosine similarity.*
6. Mencari letak kemiripan kata dari deskripsi yang sedang dibandigkan.

Sistem yang akan dibangun merupakan sebuah aplikasi atau tool untuk mendeteksi kemiripan deskripsi dengan memanfaatkan suatu ilmu yang mengacu pada *text preprocessing* atau biasa disebut dengan *text mining.* Dalam aplikasi ini data yang akan diolah adalah sebuah text dengan alur data seperti terlihat pada Gambar 3.7



Gambar 3.7 Diagram Alir Data Aplikasi Deteksi Kemirpan Deskripsi

Berdasarkan Gambar 3.7 Diagram Alir Data Aplikasi Deteksi kemiripan Deskripsi didapatkan alir data sebagai berikut :

1. Start atau memulai program

   Pada tahap ini merupakan permulaan dengan membuka aplikasi

2. Input Deskripsi Teks

   Pada tahap ini pengguna melakukan input deskripsi teks berdasarkan buku yang mau dipilih.

3. *Text preprocessing* atau *text mining*

   Text preprocessing atau *text mining* merupakan tahapan awal dalam mengolah teks untuk dapat dilakukan pencarian buku.

4. *Case Folding*

   Case Folding merupakan salah satu tahapan dalam *text mining* yang berfungsi melakukan penyeragaman kata dengan mengubah semua huruf menjadi huruf kecil.

5. *Filtering*

   Filtering merupakan salah satu tahapan dalam *text mining* yang berfungsi melakukan *filter* kata yang bermakna *stopword. Stopword* merupakan kata hanya sebagai penghubung antar kata seperti halnya : dan, ke, atau, yang, atau, sebagai, dari, adalah, kemana, dimana, tetapi, tersebut, walupun, dan lain sebagainya dan kata yang bermkana *stopword* akan dihapus. Untuk lebih jelasnya mengenai *stopword* dapat dilihat pada bab 2 landasan teori sub bab 2.2 stop word.

6. *Stemming* dengan Algoritma Nazief dan Adriani

   Stemming merupakan salah satu tahapan dalam *text mining* yang berfungsi merubah kata menjadi kata dasar tanpa imbuhan untuk lebih jelasnya mengenai *stemming* dan Algoritma Nazief dan Adriani dapat dilihat pada bab 2 landasan teori sub bab 2.3 Algoritma Nazief dan Adriani

7. *Tokenizing*

   Tokenizing merupakan suatu tahapan pemotongan *string* kata berdasrakan penyusunan kata tersebut.

8. Deskripsi *text preprocessing*

   Pada tahap ini akan didapatkan keluaran kata hasil dari *text preprocessing* setelah melalui beberapa tahapan dalam *text mining* seperti *case folding, filtering, stemming* dan *tokenizing.*

9. Pembobotan kata dengan TF-IDF

   Pembobotan kata adalah tahapan yang dilakukan selanjutnya dalam mendeteksi kemiripan deskripsi dengan memberi bobot nilai pada kata hasil dari *text preprocessing* dengan rumus sebagai berikut :

$$tf\_wt_{t,d} = \begin{cases} 1 + \log_{10}(tf_{t,d}) & Jika\ \ tf_{t,d} > 0 \\ 0 & jika\ tf_{t,d} \leq 0 \end{cases}$$

   Untuk lebih jelasnya mengenai perhitungan pembobotan tf-idf dapat dilihat pada bab 2 landasan teori sub bab 2.4 TF-IDF

10. Hitung *similarity* dokumen dengan Metode *cosine similarity*

    Pada tahap ini dilakukan suatu perhitungan *similarity* menggunakan metode *cosine similarity* dengan menggunakan bobot nilai yang sudah diberikan sebelumnya menggunakan pembobotan TF-IDF. Perhitungan *cosine similarity* dapat dilakukan dengan rumus sebagai berikut :

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\ \vec{V}(d_2)}{\left|\vec{V}(d_1)\right|\ \ |\vec{V}(d_2)|}$$

    Untuk lebih jelasnya mengenai *cosine similarity* dapat dilihat pada bab 2 landasan teori sub bab 2.5 *cosine similarity.*

11. *Sorting* nilai similarity deskripsi dari nilai tertinggi hingga terendah.

    Pada tahap ini dilakukan sebuah sorting jika deskripsi yang dibandingkan lebih dari dua deskripsi dan jika tidak langsung tampilkan nilai similarity deskripi. Nilai similarity ini menjadi acuan untuk melihat data memiliki kemiripan atau tidak yang nantinya bisa dibuktikan dengan melihat informasi letak kemiripan kata yang sama dari deskripsi tersebut.

12. Stop.

    Berdasarkan uraian diatas dapat dilakukan hipotesa dengan menggunakan representasi model data sebagai berikut:

3.1.3 Representasi Model

Table 3.1 Data Uji

Kode	Deskripsi
Q	Buku java digital database aplikasi
D1	Buku Java untuk Kriptografi memberikan pemahaman dasar pada anda tentang bagaimana kriptografi dilakukan dalam Java. Buku ini mencakup pembahasan tentang JCE dan juga JCA, enskripsi kunci simetris dan asimetris dalam Java, otentikasi pesan, sertifikat digital, bagaimana menciptakan implementasi-implementasi Java dengan AVI yang disediakan oleh Provider Bouncy Castle, dengan banyak Contoh
D2	Buku ini mengungkap secara komprehensif; kelas, control dan antarmuka, koleksi, dan database SQL server, dan aplikasi database. Setiap berorientasi contoh langkah demi langkah, memberikan kesempatan kepada setiap pembaca untuk belajar Visual Basic mulai dari nol sampai benar-benar menguasai.

Berdasarkan Table 3.1 dilakukan sebuah representasi model dengan data uji kedua yang sama dengan data tersebut dilakukan sebuah pembelajaran menggunakan sistem dan pembobotan yang berbeda sesuai dengan yang digambarkan pada Gambar 3.1 Diagram Alir Data Aplikasi Toko Buku Online.

Case folding

Table 3.2 Case folding Deskripsi Pembanding atau (D1)

Teks Asli	Teks Hasil Case Folding
Buku Java untuk Kriptografi memberikan pemahaman dasar pada anda tentang bagaimana kriptografi dilakukan dalam Java. Buku ini mencakup pembahasan tentang JCE dan juga JCA, enskripsi kunci simetris dan asimetris dalam Java, otentikasi pesan, sertifikat digital, bagaimana menciptakan implementasi-implementasi Java dengan AVI yang disediakan oleh Provider Bouncy Castle, dengan banyak Contoh	buku java kriptografi memberikan pemahaman dasar tentang bagaimana kriptografi dilakukan dalam java. buku ini mencakup pembahasan tentang jce dan juga jca, enskripsi kunci simetris dan asimetris dalam java, otentikasi pesan, sertifikat digital, bagaimana menciptakan implementasi-implementasi java dengan avi yang disediakan oleh provider bouncy castle, dengan banyak contoh

Table 3.3 Case Folding Deskripsi sumber (D2)

Teks Asli	Teks Hasil Case Folding
Buku ini mengungkap secara kompre hensif; kelas, control dan antarmuka, koleksi, dan database SQL server, dan aplikasi database. Setiap berorientasi contoh langkah demi langkah, memberikan kesempatan kepada setiap pembaca untuk belajar Visual Basic mulai dari nol sampai benar-benar menguasai.	buku ini mengungkap secara kompre hensif; kelas, control dan antarmuka, koleksi, dan database sql server, dan aplikasi database. setiap berorientasi contoh langkah demi langkah, memberikan kesempatan kepada setiap pembaca untuk belajar visual basic mulai dari nol sampai benar-benar menguasai.

Berdasarkan Table 3.2 dan 3.3 deskripsi data teks asli dilakukan penyeragaman kata menjadi huruf kecil semua hasilnya perbedaannya dapat dilihat pada masing-masing table 3.2 dan 3.3 kolom teks asli dan teks hasil case folding.

Filtering

Table 3.4 Filtering Deskripsi pembanding (D1)

Teks Hasil Case Folding	Teks Hasil Filtering
buku java untuk kriptografi memberikan pemahaman dasar pada anda tentang bagaimana kriptografi dilakukan dalam java. buku ini mencakup pembahasan tentang jce dan juga jca, enskripsi kunci simetris dan asimetris dalam java, otentikasi pesan, sertifikat digital, bagaimana menciptakan implementasi-implementasi java dengan avi yang disediakan oleh provider bouncy castle, dengan banyak contoh	buku java kriptografi pemahaman dasar kriptografi java buku mencakup pembahasan jce jca enskripsi kunci simetris asimetris java otentikasi pesan sertifikat digital menciptakan implementasi-implementasi java avi disediakan provider bouncy castle

Table 3.5 Filtering Deskripsi sumber (D2)

Teks Hasil Case Folding	Teks Hasil Filtering
buku ini mengungkap secara kompre hensif; kelas, control dan antarmuka, koleksi, dan database sql	buku mengungkap secara kompre hensif kelas control antarmuka koleksi database sql server aplikasi

server, dan aplikasi database. setiap berorientasi contoh langkah demi langkah, memberikan kesempatan kepada setiap pembaca untuk belajar visual basic mulai dari nol sampai benar-benar menguasai.	database berorientasi belajar visual basic nol menguasai.

Berdasarkan table 3.4 dan 3.5 teks hasil case folding kemudian diteruskan ke tahapan proses *text mining* berikutnya yaitu *filtering* dengan dilakukan menghapus kata yang bermakna stopword atau kata penghubung seperti kata : yang, dalam, manusia, dan sehingga, bagi, pada, hamper, dapat. Kata yang mengandung stopword akan dihapus dan hasilnya perbedaannya dapat dilihat pada table 3.4 dan table 3.5 kolom teks hasil case folding dan kolom teks hasil filtering.

Stemming

Table 3.6 Stemming Deskripsi pembanding (D1)

Teks Hasil Case Folding	Teks Hasil Filtering
buku java kriptografi pemahaman dasar kriptografi java buku mencakup pembahasan jce jca enskripsi kunci simetris asimetris java otentikasi pesan sertifikat digital menciptakan implementasi-implementasi java avi disediakan provider bouncy castle	buku java kriptografi pemahaman dasar kriptografi java buku cakup bahasan jce jca enskripsi kunci simetris asimetris java otentikasi pesan sertifikat digital ciptakan implementasi-implementasi java avi sedia provider bouncy castle

Table 3.7 Stemming Deskripsi Sumber (D2)

Teks Hasil Filtering	Teks Hasil Stemming
buku mengungkap secara kompre hensif kelas control antarmuka koleksi database sql server aplikasi database berorientasi belajar visual basic nol menguasai.	buku ungkap secara kompre hensif kelas control antarmuka koleksi database sql server aplikasi database orientasi belajar visual basic menguasai.

Berdasarkan table 3.6 dan 3.7 teks hasil filtering akan diproses ke tahapan berikutnya yang terdapat dalam *text mining* yaitu *stemming* dengan cara

mengubah kata menjadi kata dasar tanpa imbuhan seperti kata imbuhan : di, pen, ke, per, mem dan lain sebagianya sesuai dengan aturan Algoritma Nazief dan Adriani sehingga didapatkan hasil perbedaan teks yang dapat dilihat pada masing table 3.6 dan 3.7 kolom teks hasil *filtering* dan teks hasil *stemming.* Dari hasil teks *stemming* kemudian dilakukan *tokenizing* pengurutan kata yang membentuknya sesuai teks hasilnya dapat dilihat pada table 3.8 *tokenizing.*

Tokenizing

Table 3.8 Tokenizing

Kata hasil Stemming	Type Dokumen
buku	1
java	1
kriptografi	1
pemahaman	1
dasar	1
kriptografi	1
java	1
buku	1
cangkup	1
bahasan	1
jce	1
jca	1
enkripsi	1
kunci	1
simetris	1
asimetris	1
java	1
otentifikasi	1
pesan	1
sertifikat	1
digital	1
ciptakan	1
implementasi	1

implementasi	1
java	1
avi	1
sedia	1
provider	1
bouncy	1
castle	1
buku	2
ungkap	2
secara	2
kompre	2
hensif	2
kelas	2
control	2
antarmuka	2
koleksi	2
database	2
sql	2
server	2
aplikasi	2
database	2
orientasi	2
belajar	2
visual	2
Basic	2
menguasai	2

Table 3.9 Pembobotan TF-IDF

NO	Kata	Dokumen			Df	idf	Nilai TF-IDF		
		Q	D1	D2			Q	tf_idf 1	tf_idf 2
1	buku	1	2	1	3	1	**1**	**2**	**1**
2	java	1	4	0	2	0.176	**0,176**	**0.704**	**0**
3	kriptografi	0	2	0	1	0.477	**0**	**0.954**	**0**
4	pemahaman	0	1	0	1	0.477	**0**	**0.477**	**0**
5	dasar	0	1	0	1	0.477	**0**	**0.477**	**0**
6	cangkup	0	1	0	1	0.477	**0**	**0.477**	**0**
7	bahasan	0	1	0	1	0.477	**0**	**0.477**	**0**
8	jce	0	1	0	1	0.477	**0**	**0.477**	**0**
9	jca	0	1	0	1	0.477	**0**	**0.477**	**0**
10	enkripsi	0	1	0	1	0.477	**0**	**0.477**	**0**
11	kunci	0	1	0	1	0.477	**0**	**0.477**	**0**
12	simetris	0	1	0	1	0.477	**0**	**0.477**	**0**
13	asimetris	0	1	0	1	0.477	**0**	**0.477**	**0**
14	otentifikasi	0	1	0	1	0.477	**0**	**0.477**	**0**
15	pesan	0	1	0	1	0.477	**0**	**0.477**	**0**
16	sertifikat	0	1	0	1	0.477	**0**	**0.477**	**0**
17	digital	1	1	0	2	0.176	**0,176**	**0.477**	**0**
18	ciptakan	0	1	0	1	0.477	**0**	**0.477**	**0**
19	implementasi	0	2	0	1	0.477	**0**	**0.954**	**0**
20	avi	0	1	0	1	0.477	**0**	**0.477**	**0**
21	sedia	0	1	0	1	0.477	**0**	**0.477**	**0**
22	provider	0	1	0	1	0.477	**0**	**0.477**	**0**
23	bouncy	0	1	0	1	0.477	**0**	**0.477**	**0**
24	castle	0	1	0	1	0.477	**0**	**0.477**	**0**
25	ungkap	0	0	1	1	0.477	**0**	**0**	**0.477**
26	secara	0	0	1	1	0.477	**0**	**0**	**0.477**
27	kompre	0	0	1	1	0.477	**0**	**0**	**0.477**

28	hensif	0	0	1	1	0.477	**0**	**0**	**0.477**
29	kelas	0	0	1	1	0.477	**0**	**0**	**0.477**
30	control	0	0	1	1	0.477	**0**	**0**	**0.477**
31	antarmuka	0	0	1	1	0.477	**0**	**0**	**0.477**
32	koleksi	0	0	1	1	0.477	**0**	**0**	**0.477**
33	database	1	0	2	2	0.176	**0,176**	**0**	**0.352**
34	sql	0	0	1	1	0.477	**0**	**0**	**0.477**
35	server	0	0	1	1	0.477	**0**	**0**	**0.477**
36	aplikasi	1	0	1	2	0.176	**0,176**	**0**	**0.176**

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\ \vec{V}(d_2)}{|\vec{V}(d_1)|\ \ |\vec{V}(d_2)|}$$

$$\vec{V}(Q) = \frac{1\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0{,}176}{\sqrt{1^2 + 0{,}176^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2} + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0{,}176^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0{,}176^2 + 0^2 + 0^2 + 0{,}176^2}$$

$$\vec{V}(Q) = \frac{1\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0{,}176}{\sqrt{1.123904}}$$

$$\vec{V}(Q) = \frac{1\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}176\ 0\ 0\ 0{,}176}{1.06014}$$

Q= {0,94327} {0,16601} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0,16601} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0,16601} {0} {0} {0,16601}

$$\vec{V}(D_1) = \frac{2\ 0{,}704\ \ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}{\sqrt{2^2 + 0{,}704^2 + 0{,}954^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2\ 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}954^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2}}$$

$$\vec{V}(D_1) = \frac{2\ 0{,}704\ \ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}{\sqrt{10.866428}}$$

$$\vec{V}(D_1) = \frac{2\ 0{,}704\ \ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}954\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}{3.29642}$$

D1= {0,60671} {0,21356} {0,28940} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0,28940} {0,14470} {0,14470} {0,14470} {0,14470} {0,14470} {0} {0} {0} {0} {0} {0} {0} {0} {0}

$$\vec{V}(D_2) = \frac{1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}352\ 0{,}477\ 0{,}477\ 0{,}176}{\sqrt{1^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}477^2 + 0{,}352^2 + 0{,}477^2 + 0{,}477^2 + 0{,}176^2}}$$

$$\vec{V}(D_2) = \frac{1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}352\ 0{,}477\ 0{,}477\ 0{,}176}{\sqrt{3.43017}}$$

$$\vec{V}(D_2) = \frac{1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}477\ 0{,}352\ 0{,}477\ 0{,}477\ 0{,}176}{1.85207}$$

D2={0,53993}{0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0} {0,25754} {0,25754} {0,25754} {0,25754} {0,25754} {0,25754} {0,25754} {0,25754} {0,19005} {0,25754} {0,25754} {0,09502}

Sim(Q,D1) = $(Q * D1_1) + (Q * D2_2) + \cdots + (D1_N * D2_N)$

Table 3.10 Hasil perhitungan dengan threshold diatas diperlihatkan tabel berikut :

D1	D2
0,96378	**0,55661**

Urutkan hasil perhitungan kemiripan, diperoleh

1	2
D1	**D2**

3.2 Perancangan system

Berdasarkan dari alur flow pada deskripsi sistem dapat dimodelkan sebuah desain sistem yang sesuai dengan urutan proses yang telah ditetapkan, yaitu *Context Diagram*, Diagram Jenjang, Diagram Alir Data (*Data Flow Diagram*), Desain Basis Data (*Database*), Ektrasi Dokumen, Desain Antarmuka (*Interface*).

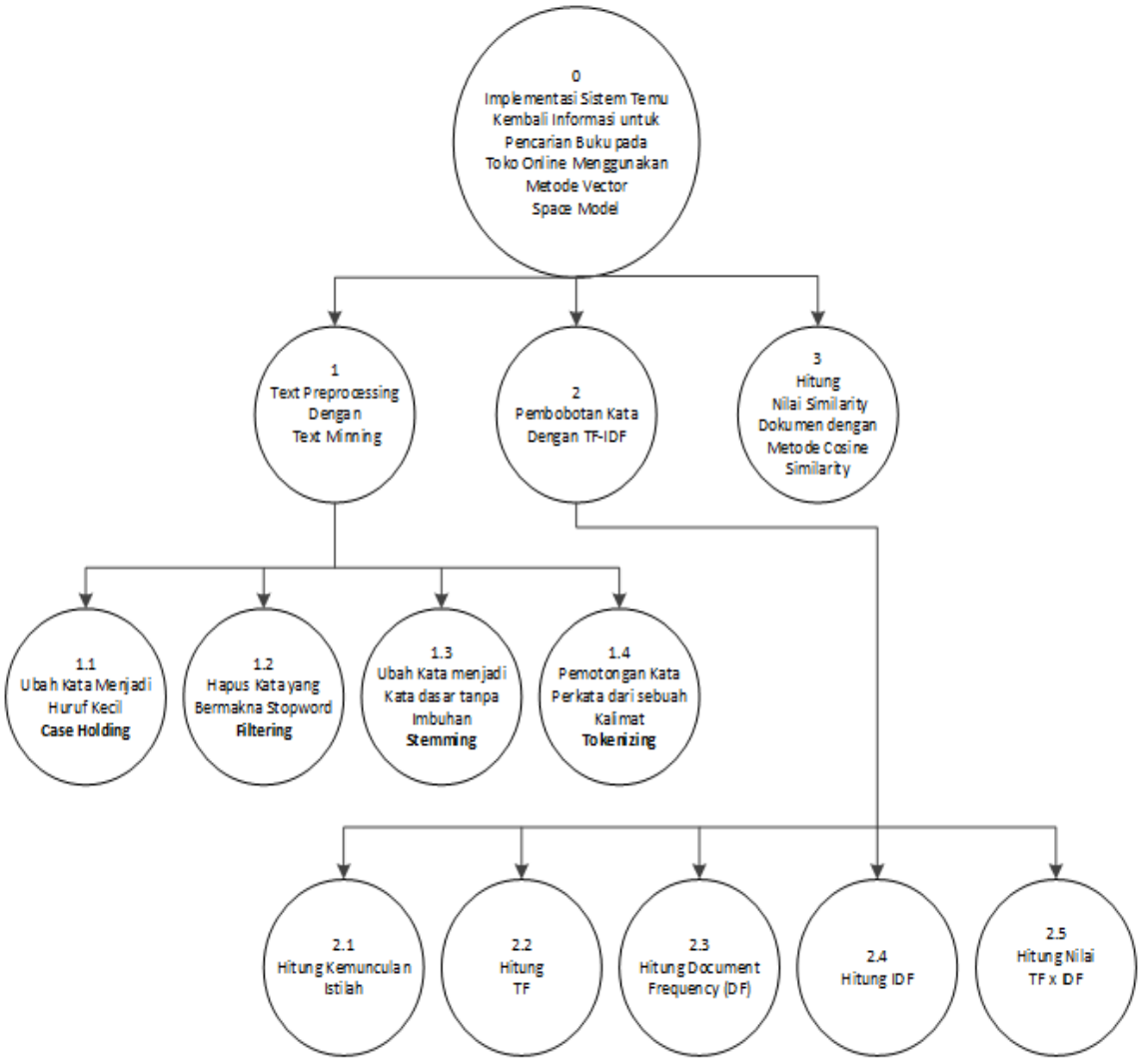
3.2.1 Diagram konteks

Gambar 3.8 Diagram Konteks

Berdasarkan Gambar 3.8 pengguna memasukan deskripsi teks kedalam toko buku online dan pengguna akan mendapat keluaran dari sistem yaitu sebuah informasi mengetahui letak kemiripan kata yang sedang dicari *similaritynya.*

3.2.2 Diagram Berjenjang.

Diagram berjenjang merupakan pemecahan proses dari sebuah sistem dimana pada aplikasi deteksi kemiripan dibagi beberapa proses anatar lain: simpan dokumen, text preprocessing pembobotan TF-IDF, perhitungan nilai *similarity* dengan metode cosine similarity sedangkan untuk pemecahan proses



Gambar 3.9 Diagram Berjenjang

Berdasarkan Gambar 3.9 didapatkan beberapa pemecahan proses yang terbagi dalam beberapa level sebagai berikut :

Level 0 Aplikasi Detksi Kemiripan dokumen

Level 1.

1. Teks Preprocessing dengan *Text Mining*

   Teks Preprocessing merupakan tahapan awal dalam mengolah deskripsi teks yang diunggah oleh pengguna kedalam sistem dan diproses menggunakan teknik *text mining.*

2. Pembobotan kata dengan TF-IDF

   Pembobotan Kata dengan TF-IDF dilakukan untuk memberikan nilai bobot kata dari masing deskripsi.

3. Hitung Nilai Similarity dengan Metode *Cosine Similarity*

   Hitung Nilai similarity dilakukan untuk mencari nilai similarity deskripsi dengan menggunakan metode cosine similarity dari bobot kata yang telah diperoleh dengan TF-IDF. Kemudian dari nilai bobot tersebut kita akumulasikan terhadap rumus *cosine similarity* seperti terlihat pada persamaan ke 4

Level 2 proses 1

1.0 *Case Folding*

Case Foldng merupakan salah satu sub proses yang terdapat dalam tahapn *text mining* digunakan untuk penyeragaman kata dari huruf besar menjadi huruf kecil.

1.1 *Filtering*

Filtering merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* digunakan untuk menghapus kata yang bermakna *stopword*

1.2 Algortitma Nazief dan Adriani *Stemming*

Stemming merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* digunakan untuk mengubah kata menjadi kata dasar tanpa imbuhan

1.3 *Tokenizing*

Tokenizing merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* sebagai pengurutan kata dari kata yang membentuk pada suatu dokumen

Level 2 proses 2

2.0 Hitung kemunculan istilah

Proses ini dilakukan suatu perhitungan kemunculan istilah kata pada masing masing dokumen dari index kata.

2.1 Hitung TF (Term Frequency)

Proses ini dilakukan suatu perhitungan TF dari tiap kata dari masing masing dokumen dengan rumus seperti terlihat pada persamaan ke 1

2.2 Hitung DF (Document Frequency)

Proses ini dilakukan suatu perhitungan kemunculan jumlah kata pada tiap dokumen.

2.3 Hitung IDF (Inverse Document Frequency)

Proses ini dilakukan suatu perhitungan inverse dokumen frequency pada masing kata pada tiap deskripsi dengan rumus seperti terlihat pada persamaan ke 2.

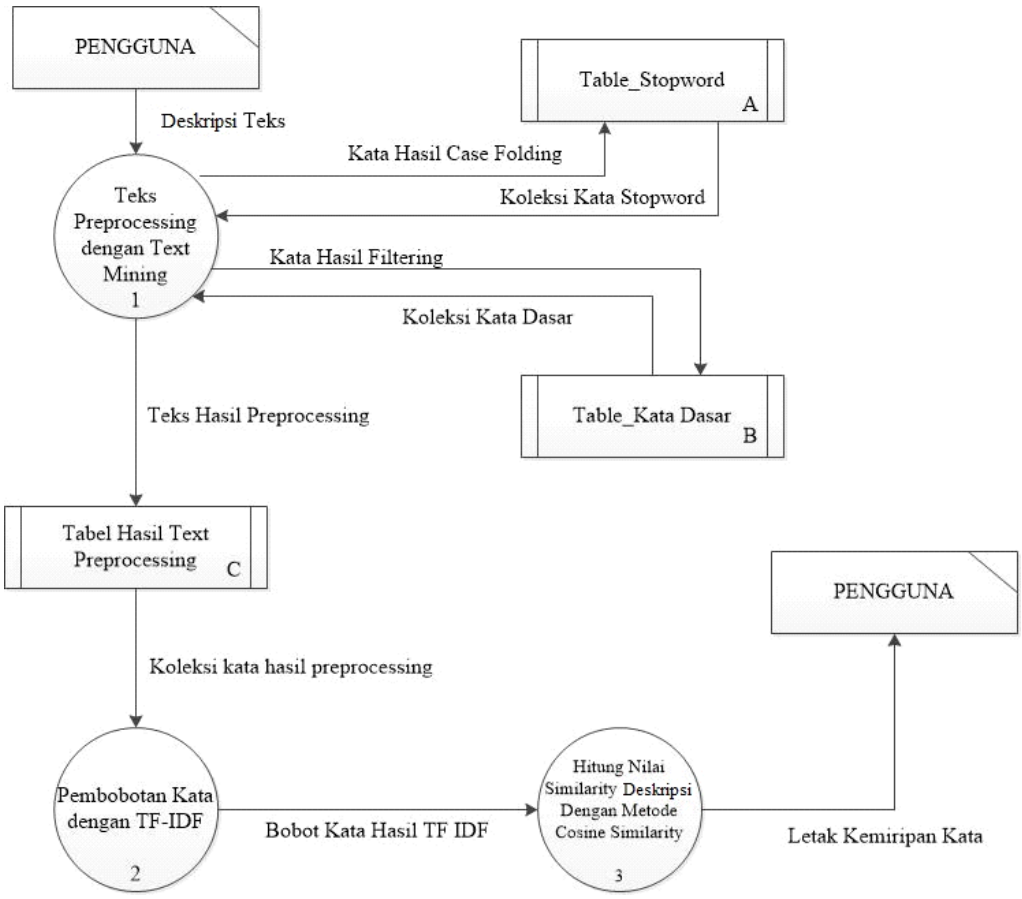
2.4 Hitung TF X IDF

Proses ini dilakukan akumulasi terkahir dengan menghitung nilai TF X IDF seperti terlihat pada persamaan ke 3 untuk rumus keseluruhan dari pembobotan kata dengan menggunakan TF IDF untuk melihat hasil perhitungan dapat dilihat pada table 3.9

3.2.3 Data Flow Diagram

Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem, yang penggunaannya sangat membantu untuk memahami sistem secara logika, tersruktur dan jelas. DFD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan logis sedangkan untuk

aplikasi deteksi kemiripan dokumen memiliki arus dari data sistem yang digambarkan sebagai berikut:



Gambar 3.10 DFD Level 1

Berdasarkan Gambar 3.10 didapatkan data flow diagram sebagai berikut :

4. Pada diagram tersebut dimulai dari pengguna dengan menuliskan sebuah deskripsi kedalam aplikasi.
5. Kemudian dari data yang masuk akan disimpan ke dalam *database*
6. Dari deskripsi tersebut akan diolah kembali dengan *text mining* dan hasil dari text mining akan dimasukan ke dalam *database* dengan *table* hasil *text preprocessing.*
7. Dalam tahapan text mining terdapat tahapan *case folding* atau penyeragaman kata menjadi huruf kecil.

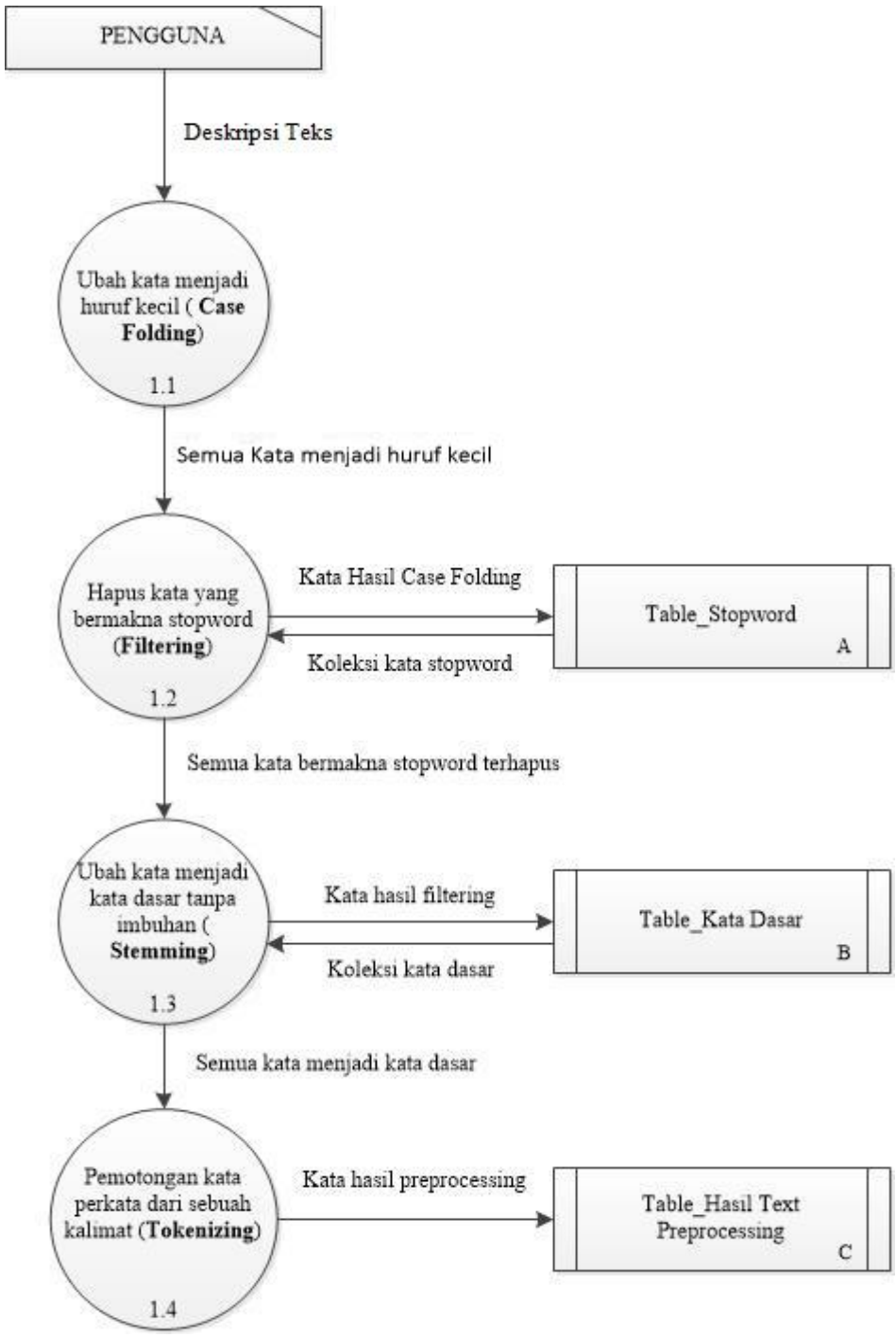
8. Dari hasil *case folding* akan dilanjutkan ke tahapan *text mining* yaitu *filtering* dengan menghapus kata yang bermakna *stopword* atau biasa disebut kata penghubung.
9. Hasil dari *filtering* kemudian dilanjutkan ke tahapan berikutnya yaitu *stemming* dengan mengubah kata menjadi kata dasar tanpa imbuhan dalam kosakata Bahasa Indonesia dan hasilnya ditampilkan dalam bentuk *Tokenizing* atau dipotong sesuai kata perkata dari sebuah kalimat dan disimpan dalam *storage text* hasil *preprocessing*.
10. Setelah itu data hasil *text preprocessing* akan diolah kembali untuk diberikan nilai bobot dari kata-kata pada tiap dokumen dengan TF-IDF.
11. Selanjutnya dari pembobotan yang telah dilakukan data akan diambil untuk dilakukan perhitungan nilai *similarity* dengan metode *cosine similarity* dengan rumus dan dapat dilihat seperti pada persamaan ke 4
12. Hasil nilai similarity akan ditampilkan pada pengguna aplikasi.
13. Hasil nilai similarity akan menjadi acuan untuk mengetahui kemiripan deskripsi dan nantinya di buktikan dengan melihat letak dari kemiripan kata dari deskripsi tersebut.

DFD Level 2 Proses 1

Berdasarkan Gambar 3.11 didapatkan data flow sebagai berikut :

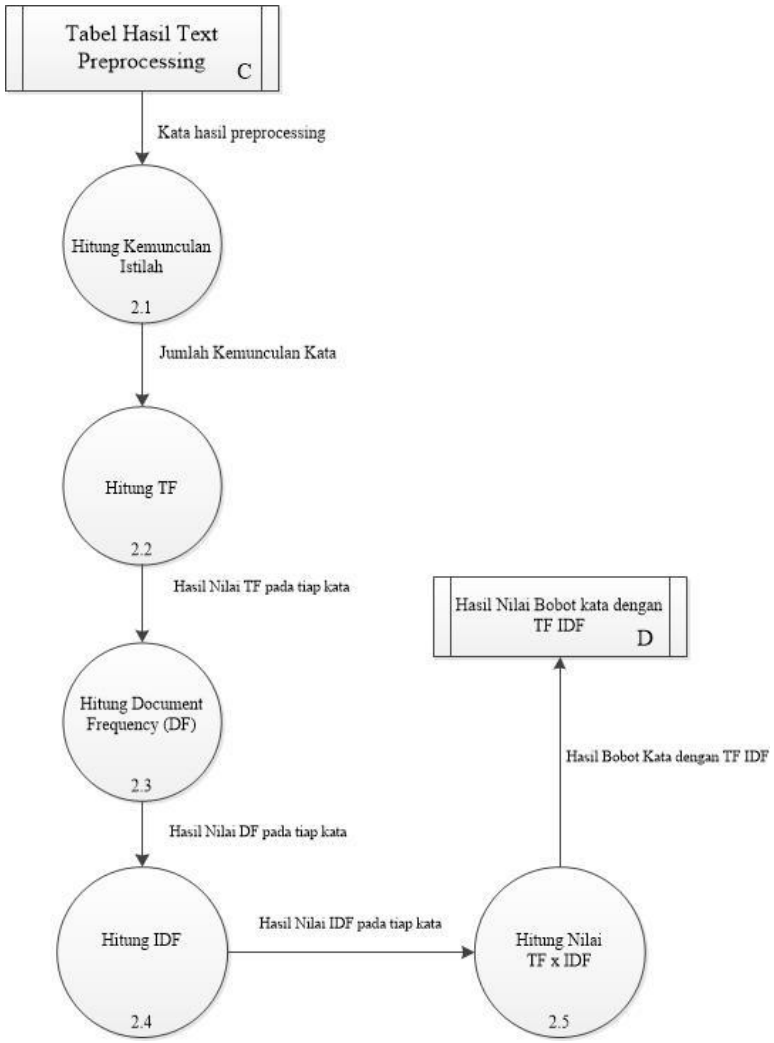
14. Pada diagram tersebut dimulai dari pengguna dengan memasukan sebuah deskripsi berisi sebuah teks berbahasa Indonesia.
15. Kemudian deskripsi tersebut akan masuk tahap awal *text preprocessing* atau *text mining* yaitu *case folding* dengan melakukan penyeragaman kata menjadi huruf kecil
16. Setelah itu hasil dari *case folding* akan diteruskan ke proses *filtering* dengan menghapus kata-kata yang bermakna *stopword*.
17. Selanjutnya kata-kata tersebut akan diubah ke bentuk kata dasar sebelum mendapatkan kata-kata imbuhan proses ini biasa disebut *stemming*. Dengan mengguakan algoritma nazief & adriani.

18. Hasil dari keseluruhan proses akan dilakukan sebuah pemotongan kata perkata (*tokenizing)* yang nantinya akan dimasukkan ke dalam table hasil text preprocessing



Gambar 3.11 DFD Level 2 Proses 1

DFD Level 2 Proses 2



Gambar 3.12 DFD Level 2 Proses 2

Berdasarkan gambar 3.12 didapatkan sebuah alur flow data dari proses pembobotan dengan TF IDF sebagai berikut :

19. Pengguna akan memasukkan kata hasil text preprocessing untuk diolah oleh sistem untuk dilakukan suatu pembobotan kata.
20. Kata hasil text preprocessing akan dilakukan perhitungan jumlah istilah kemunculan kata sebagai acuan awal dari pembobotan TF IDF.
21. Kata hasil text preprocessing kemudian dilakukan perhitungan nilai TF dari tiap kata dengan rumus dan dapat dilihat seperti persamaan ke 1

22. Hitung Jumlah kata yang sama dan muncul di masing dokumen perhitungan ini digunakan untuk menghitung DF (Document Frequency).
23. Kata hasil preprocessing kemudian dilakukan perhitungan nilai IDF dari tiap kata dengan rumus dan dapat dilihat seperti persamaan ke 2
24. Hasil TF dan IDF diakumulasikan dengan perkalian untuk mendapatkan hasil nilai bobot kata dengan TF IDF
25. Selesai dan di simpan ke dalam table hasil nilai bobot kata dengan TF IDF.

3.3 Desain database

26. Tabel kata dasar

Tabel kata dasar berisi kumpulan kata dasar dari sebuah kata dengan berbahasa Indonesia yang nantiny akan digunakan pada proses *stemming* dengan struktur table seperti terlihat pada table 3.11

Tabel 3.11 Kata dasar

Nama Field	Type	Keterangan
Id_katadasar	Int Auto Increment	Primary Key
Kata_dasar	Varchar (255)	
Tipe_keterangan	Varchar (255)	

27. Tabel stopword

Tabel *stopword* adalah kumpulan kata yang bermakna *stopword* yang nantinya akan digunakan pada proses *filtering* dan kata yang sama dengan kata *stopword* akan dihilangkan dengan struktur table seperti terlihat pada table 3.12

Tabel 3.12 Stopword

Nama Field	Type	Keterangan
Id_stopword	Int Auto Increment	Primary Key
Stopword	Varchar (255)	

28. Tabel Deskripsi sumber

Tabel dokumen sumber merupakan kumpulan kata yang telah melalui tahapan *text preprocessing* dan menjadi dokumen sumber yang nanti akan menjadi pembanding untuk dokumen pembanding dengan struktur table seperti terlihat pada table 3.14

Tabel 3.14 Deskripsi sumber

Nama Field	Type	Keterangan
Id_sumber	Int Auto Increment	Primary Key
Kata	Varchar (255)	
Dok_sumber	Int (20)	
Id_banding	Int	Foreign Key

29. Tabel Kata Inputan

Tabel dokumen pembanding merupakan kumpulan kata yang telah melalui tahapan *text preprocessing* dan menjadi dokumen pembanding untuk dicari nilai similarity dengan dokumen sumber dengan struktur table terlihat pada table 3.15

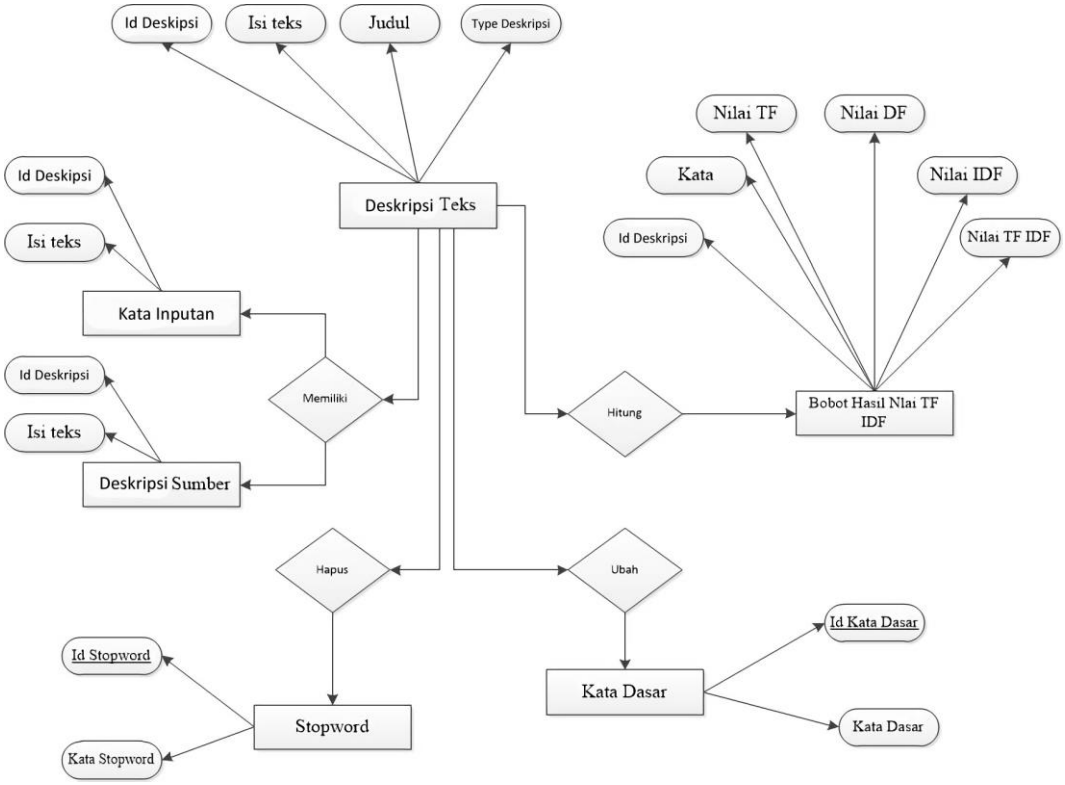
Tabel 3.15 Kata Inputan

Nama Field	Type	Keterangan
Id_Inputan	Int Auto Increment	Primary Key
Kata	Varchar (255)	
Des_Inputan	Int (20)	

3.3.1 Entity Relation Diagram (ERD)

Konsep data model merupakan bentuk data yang masih di konsep untuk direalisasikan dengan tabel-tabel yang lain dan data ini bukan merupakan tabel pada keadaan yang sebenarnya karena masih perlu dilakukan proses *generic* untuk menjadi tabel yang sesuai dengan sebenarnya. Karena masih konsep maka kunci-

kunci relasi dari tabel yang lain belum di masukkan diagram ERD *database* yang dirancang dapat dilihat pada Gambar 3.13.



Gambar 3.13 Entity Relation Diagram

Berdasarkan gambar 3.13 didapatkan entity relation diagram sebagai berikut:

1. Dokumen teks yang akan diupload memiliki dua tipe dokumen yaitu dokumen pembanding dan dokumen sumber yang akan menjadi sumber dokumen yang akan dibandingkan dengan dokumen pembanding.
2. Dokumen teks sudah diupload akan diproses untuk dicari kata yang bermakna stopword dan kemudian kata tersebut akan dihapus secara otomatis.
3. Dokumen teks sudah diupload akan diubah menjadi kata dasar tanpa imbuhan sesuai EYD Bahasa Indonesia.
4. Dokumen teks yang sudah melalui tahap *text preprocessing* akan dilakukan perhitungan bobot kata TF IDF.

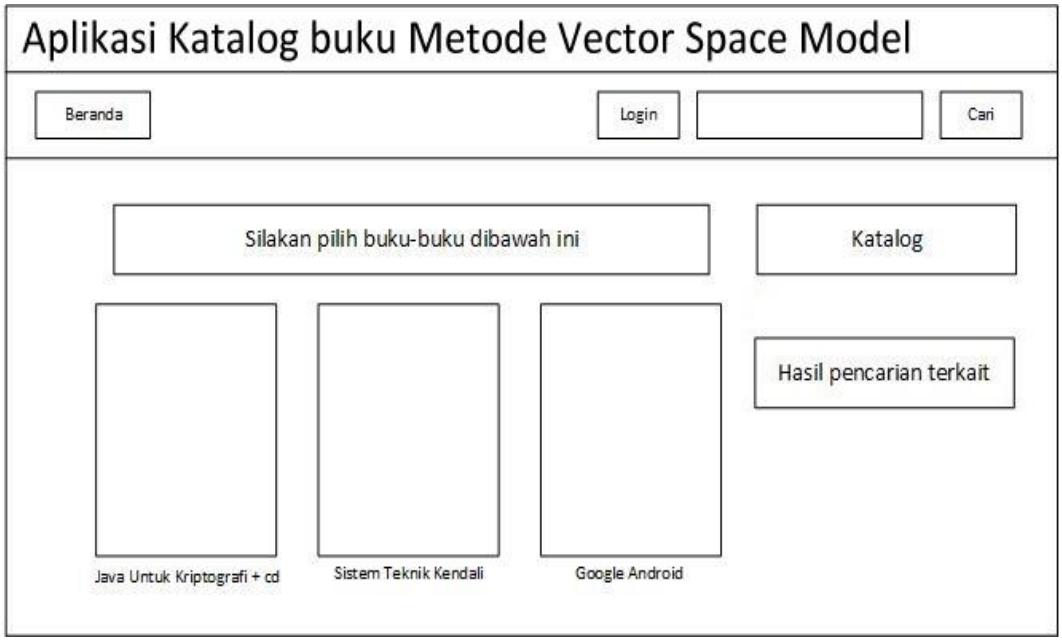
3.4 Desain Interface

3.4.1 Desain Tampilan Antar Muka (Beranda)

Halaman Antar Muka merupakan tampilan awal dari aplikasi pencarian buku pada toko online yang menampilkan deskripsi dari aplikasi tersebut dengan beberapa fitur menu seperti : tutorial yang berisi tentang cara pemakaian aplikasi, pencarian buku dengan beberapa deskripsi merupakan fitur menu untuk membandingkan beberapa deskripsi, stopword merupakan menu tampilan kata-kata yang bermakna stopword dalam Bahasa Indonesia dan menu tentang merupakan tentang pembuatan aplikasi dan lain sebagainya yang berhubungan dengan pembuatan aplikasi. Berikut merupakan tampilan dari beberapa menu aplikasi tampilan antar muka dapat dilihat pada gambar dibawah ini.

3.4.2 Desain Tampilan Antar Muka

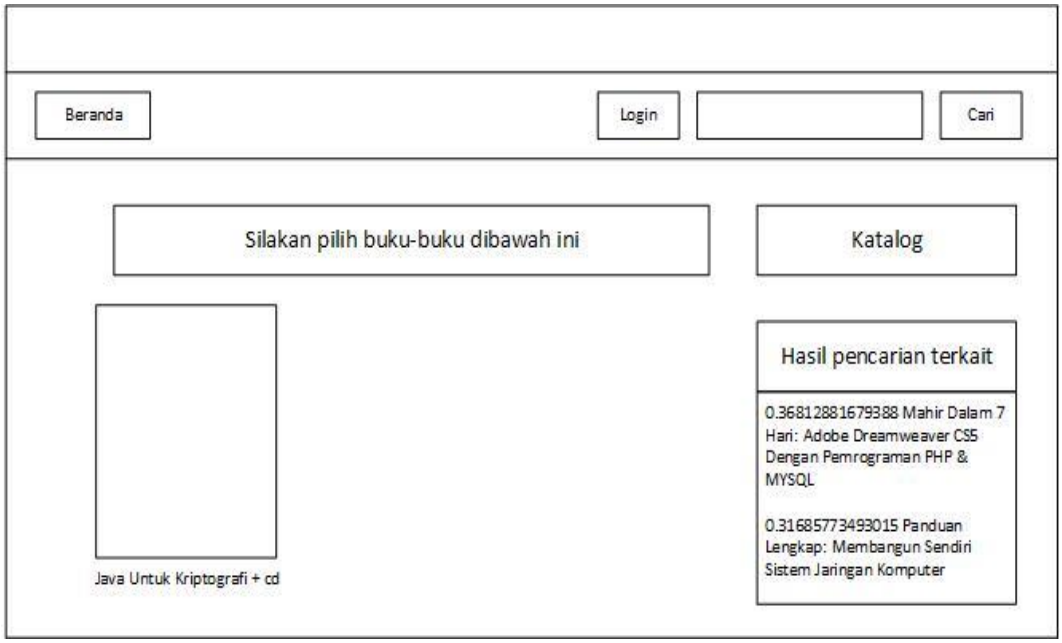
Desain tampilan antar muka memiliki fitur pencarian yang bersumber pada buku yang dicari oleh pengguna terdapat sebuah kolom seach untuk mencari buku seperti pada Gambar 3.14



Gambar 3.14 Desain Tampilan Antar Muka

3.4.3 Desain Tampilan Hasil Pencarian Buku

Desain tampilan hasil pencarian buku memiliki fitur buku yang sudah dipilih oleh pengguna terdapat sebuah hasil pencarian terkait beserta judul buku dan perhitungannya seperti pada Gambar 3.15



Gambar 3.15 Desain Tampilan Hasil Pencarian Buku

3.5 Kebutuhan Pembuatan Sistem

3.5.1 Perangkat Lunak

1. Sistem Operasi windows 8

   Sistem operasi digunakan untuk mengimplementasi Aplikasi deteksi kemiripan dokumen

2. PHP 5.6.3 dan Apache 2.4.3

   PHP adalah salah satu Bahasa pemrograman yang digunakan untuk membangun aplikasi deteksi kemripan dokumen dan apache sebagai server untuk menjalankan program yang dibuat dengan Bahasa pemrograman PHP

3. MySQL Server 5.5.27 sebagai *database* server

   Mysql adalah salah satu databse server yang digunakan untuk menampung dokumen dari Aplikasi deteksi yang akan dibangun nantinya.
4. Mozilla Firefox 38.0.1 (*browser*)

   Mozila firefox adalah salah satu browser untuk menampilkan output atau keluaran dari program yang telah dibuat
5. Edit Plus dan Microsoft Visio

   Edit plus adalah *software* text editor untuk menulisakn barisan perintah-perintah code sedangkan Microsoft Visio adalah *software* untuk membuat *paper flow diagram, DFD.*

3.5.2 Kebutuhan perangkat keras

Sistem perangkat keras (*Hardware*) adalah komponen-komponen pendukung kinerja dari sistem komputer. Komponen-komponen yang dapat dipakai untuk menjalankan aplikasi deteksi kemiripan dokumen adalah sebagai berikut:

1. Prosesor Intel Core^{tm} i3-380M
2. Memory RAM 512 MB atau lebih
3. Monitor VGA atau SVGA dengan resolusi 800x 600 atau lebih
4. Hardisk minimal 40 GB atau lebih
5. Mouse
6. Keyboard
7. Printer

3.6 Skenario Pengujian

Pada penelitian ini, untuk mengukur evaluasi kinerja sistem temu kembali informasi digunakan pengujian *recall, precision, accuracy*, dan *F-Measure*. *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. *Precision* adalah tingkat ketepatan antara

informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem.

Tabel 3.16 Parameter menghitung precision dan recall

Keterangan	**Relevan**	**Tidak Relevan**
Terambil	True positive (tp)	False positive (fp)
Tidak terambil	False negative (fn)	True negative (tn)

Rumus untuk menghitung Precision :

$$Precision = \frac{tp}{tp+fp} \quad (3.1)$$

Rumus untuk menghitung Recall :

$$Recall = \frac{tp}{tp+ fn} \quad (3.2)$$

Rumus untuk menghitung Accuracy :

$$Accuracy = \frac{tp+tn}{tp+fp+tn+fn} \quad (3.3)$$

Rumus untuk menghitung F-Measure :

$$F - measure = \frac{2\times precision\times recall}{precision+recall} \quad (3.4)$$

Misalnya dari pengujian menggunakan 100 ikan salmon dan 900 ikan lain ternyata mesin hanya memisahkan 1 ikan salmon, dan setelah dicek oleh manusia, 1 ikan tersebut benar merupakan ikan salmon. Pengujian ini dapat kita tuliskan sebagai berikut :

		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	1	0
	FALSE	99	900

$$precision = \frac{1}{1+0} = \frac{1}{1} = 1 = 100\%$$

$$recall = \frac{1}{1+99} = \frac{1}{100} = 0.01 = 1\%$$

$$accuracy = \frac{1+900}{1+900+0+99} = \frac{901}{1000} = 0.901 = 90.1\%$$

Dari hasil perhitungan kita dapatkan precision sebesar 100% dan accuracy sebesar 90.1%. Sekilas tampak baik, namun perhatikan nilai recall yang hanya

sebesar 1%. Hal ini menunjukkan bahwa sistem hanya dapat memisahkan ikan salmon dalam jumlah yang sedikit sekali dan masih banyak ikan-ikan salmon yang lolos dari pemisahan. Nilai *precision*, *recall*, dan *accuracy* dinyatakan dalam persen. Semakin tinggi ketiga nilai tersebut menunjukkan semakin baiknya kinerja aplikasi. Evaluasi yang akan dilakukan dalam penelitian ini adalah menghitung nilai dari *precision*, *recall*, *accuracy* dan *f-measure* berdasarkan dokumen yang berhasil ditemukan kembali oleh sistem aplikasi yang dibuat. Sedangkan untuk menentukan nilai dari *precision*, *recall*, *accuracy*, dan *f-measure* harus didapatkan jumlah dokumen yang relevan terhadap suatu topik dokumen abstrak.