BAB 4
IMPLEMENTASI DAN PENGUJIAN

Bab ini akan membahas mengenai proses implementasi dan pengujian terhadap aplikasi pencarian dokumen. Dalam skripsi ini implementasi yang akan dijabarkan adalah mengenai deskripsi perangkat lunak, input dan output perangkat lunak, serta hasil tampilan pada perangkat lunak. Sedangkan untuk pengujian adalah melakukan testing terhadap hasil implementasi dengan menggunakan beberapa data dan akan dianalisis hasil pengujiannya.

4.1 Implementasi Aplikasi

Pada implementasi perangkat lunak ini akan dijelaskan lingkungan implementasi, batasan implementasi, data masukan dan data keluaran.

4.1.1 Lingkungan Implementasi

Spesifikasi lingkungan perangkat keras yang digunakan untuk pengembangan perangkat lunak ini adalah sebagai berikut :

a. Processor Intel(R) Core(TM) 2 Duo processor 2.5GHz
b. Memori RAM 3072MB
c. Harddisk 296GB
d. Perangkat keluaran berupa monitor 14’’
e. Perangkat masukkan berupa *mouse* dan *keyboard*

Spesifikasi lingkungan perangkat lunak yag digunakan untuk skripsi ini adalah sebagai berikut :

a. Sistem operasi Microsoft Windows 7 Ultimate

b. Java Standard Edition Development Kit 6.10
c. IDE NetBeans 6.8
d. MySQL server 5.1
e. Enterprise Architect 7.5 – UML Case Tool sebagai alat bantu penggambaran perancangan dengan UML
f. Power Designer 12 sebagai alat bantu pengembangan perancangan skema *database*
g. Web browser Mozilla Firefox 8.0.1

4.1.2 Batasan Implementasi

Batasan dalam implementasi perangkat lunak adalah sebagai berikut :

a. Perangkat lunak yang dibangun merupakan perangkat lunak yang dapat melakukan proses penentuan indeks dan pencarian *file*.
b. Perangkat lunak yang dibuat dijalankan secara *offline* pada komputer yang sekaligus menjadi *web server* dan *web client.*
c. *Fil*e yang digunakan sebagai masukan untuk koleksi *file* adalah *file text* berbahasa Inggris dengan ekstensi file *.txt dan *.doc, dimana setiap masukan *file* memiliki total *term* yang berbeda-beda.
d. Algoritma *stemming* yang digunakan dalam perangkat lunak ini adalah algoritma *Porter Stemmer*
e. Daftar kata *stop list* disimpan dalam file stop-list.txt.
f. Daftar *irregular verb* disimpan dalam file irregularverb-list.txt

4.1.3 Data Masukan

Data masukan yang diperlukan dalam perangkat lunak ini adalah sebagai berikut :

a. Data dokumen yang berupa judul, keterangan dan kategori
b. Koleksi *file* bertipe teks berbahasa Inggris dengan panjang jumlah *term* yang berbeda-beda.

4.1.4 Data Keluaran

Data keluaran dari perangkat lunak ini adalah *file* yang sesuai dengan *query* yang dimasukkan.

4.1.5 Implementasi Database

Pembuatan *database* dilakukan dengan menggunakan aplikasi *DBMS MySQL*. Implementasi *database* dalam bahasa *SQL* terlihat pada bagian dibawah ini :

Tabel 4.1 Implementasi database

```
--
-- Definition of table `user`
--

DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `user_id` varchar(36) NOT NULL,
  `user_address` varchar(2000) DEFAULT NULL,
  `user_displayname` varchar(50) NOT NULL,
  `user_email` varchar(50) NOT NULL,
  `user_fname` varchar(20) NOT NULL,
  `user_lastlogindate` datetime DEFAULT NULL,
  `user_lname` varchar(30) DEFAULT NULL,
  `user_password` varchar(20) DEFAULT NULL,
  `user_phonenumber` varchar(50) DEFAULT NULL,
  `user_photo` varchar(255) DEFAULT NULL,
```

```
  `user_status` int(11) DEFAULT NULL,
  `user_username` varchar(36) NOT NULL,
  `user_website` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


--
-- Definition of table `document`
--

DROP TABLE IF EXISTS `document`;
CREATE TABLE `document` (
  `document_id` varchar(36) NOT NULL,
  `document_datecreated` datetime DEFAULT NULL,
  `document_dateupdated` datetime DEFAULT NULL,
  `document_description` varchar(1000) DEFAULT NULL,
  `document_title` varchar(250) NOT NULL,
  `document_total_view` int(11) DEFAULT NULL,
  `document_last_updater_id` varchar(36) NOT NULL,
  `document_author_id` varchar(36) DEFAULT NULL,
  PRIMARY KEY (`document_id`),
  KEY `FK335CD11B3829C5DF` (`document_author_id`),
  CONSTRAINT `FK335CD11B3829C5DF` FOREIGN KEY
(`document_author_id`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


--
-- Definition of table `file`
--

DROP TABLE IF EXISTS `file`;
CREATE TABLE `file` (
  `file_id` varchar(36) NOT NULL,
  `file_date` datetime DEFAULT NULL,
  `file_ext` varchar(5) NOT NULL,
  `file_name` varchar(500) NOT NULL,
  `pure_name` varchar(500) NOT NULL,
  `file_size` bigint(20) DEFAULT NULL,
  `document_id` varchar(36) DEFAULT NULL,
  PRIMARY KEY (`file_id`),
  KEY `FK2FF57C797180DC` (`document_id`),
  CONSTRAINT `FK2FF57C797180DC` FOREIGN KEY (`document_id`)
REFERENCES `document` (`document_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Definition of table `dictionary`
--

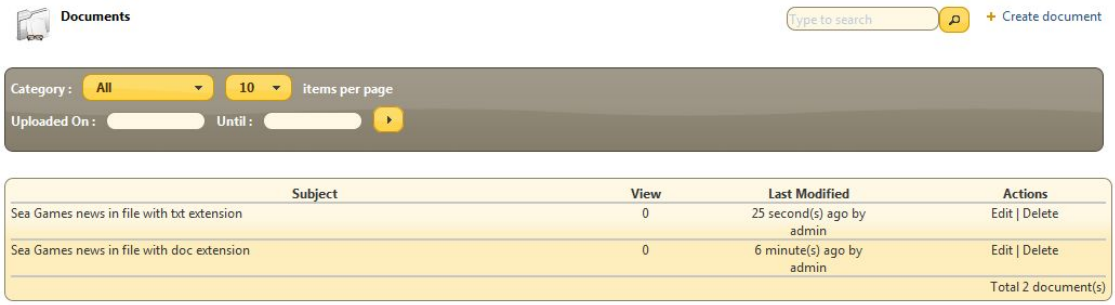
DROP TABLE IF EXISTS `dictionary`;
CREATE TABLE `dictionary` (
  `dictId` varchar(20) NOT NULL,
  PRIMARY KEY (`dictId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Definition of table `posting`
--

DROP TABLE IF EXISTS `posting`;
CREATE TABLE `posting` (
  `posting_id` varchar(36) NOT NULL,
  `frequency` int(11) DEFAULT NULL,
  `dict_id` varchar(20) DEFAULT NULL,
  `file_id` varchar(36) DEFAULT NULL,
  PRIMARY KEY (`posting_id`),
  KEY `FKE8AEBB42CCABEA3C` (`file_id`),
  KEY `FKE8AEBB423295E35C` (`dict_id`),
  CONSTRAINT `FKE8AEBB423295E35C` FOREIGN KEY (`dict_id`)
REFERENCES `dictionary` (`dictId`),
  CONSTRAINT `FKE8AEBB42CCABEA3C` FOREIGN KEY (`file_id`)
REFERENCES `file` (`file_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.1.6 Aplikasi Pengelolaan Dokumen

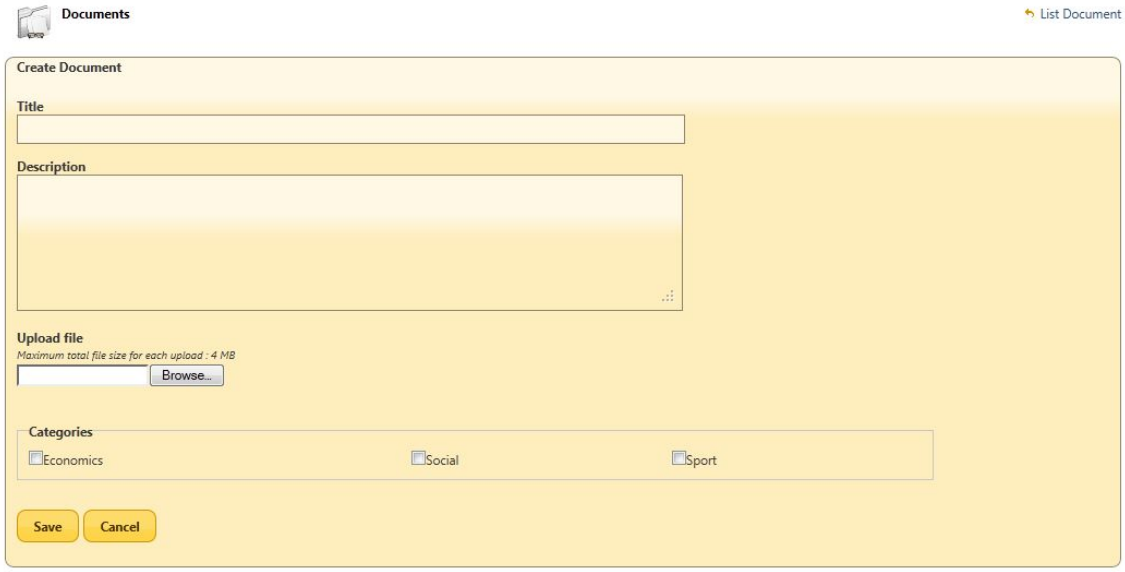
Aplikasi ini bertujuan untuk mengelola data dokumen dan juga mengindeks *file* ke dalam basis data. Aplikasi ini berbasis web. Pada tampilan awal aplikasi, terdapat beberapa komponen yaitu : daftar dokumen; *toolbar* untuk pencarian dokumen berdasarkan kategori, tanggal *upload* dokumen, dan kata kunci; menu-menu untuk mengelola dokumen seperti : *create*, *edit*, dan *delete*.



Gambar 4.1 Tampilan awal aplikasi pengelolaan dokumen

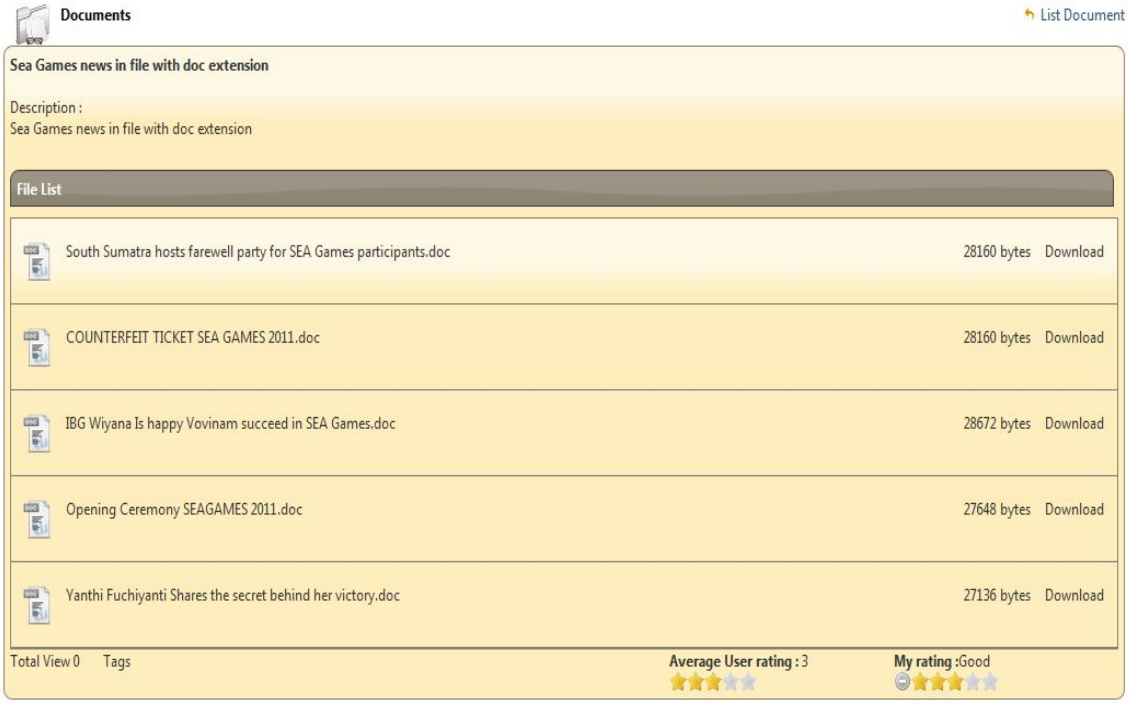
Pada saat pengguna ingin menambah dokumen baru, maka pengguna tinggal memilih menu Create Document pada halaman awal pengelolaan

dokumen. Selanjutnya pengguna akan diarahkan ke halaman *upload* dokumen. Pada halaman *upload* dokumen, pengguna harus mengisi kolom-kolom seperti : judul, keterangan, kategori, serta *file* yang akan diunggah.



Gambar 4.2 Tampilan halaman *upload* dokumen

Pada saat pengguna ingin melihat detail dokumen, maka pengguna pilih judul dokumen pada daftar dokumen di halaman awal pengelolaan dokumen. Selanjutnya pengguna akan diarahkan ke halaman detail dokumen. Pada halaman detail dokumen, pengguna mendapatkan informasi mengenai judul dokumen, keterangan dokumen, daftar *file* dalam dokumen tersebut, jumlah dokumen dilihat oleh pengguna, juga *rating* (penilaian) dari dokumen tersebut. Pada halaman detail dokumen ini, pengguna bisa mengunduh *file-file* yang terdapat dalam dokumen, serta memberikan penilaian terhadap dokumen.



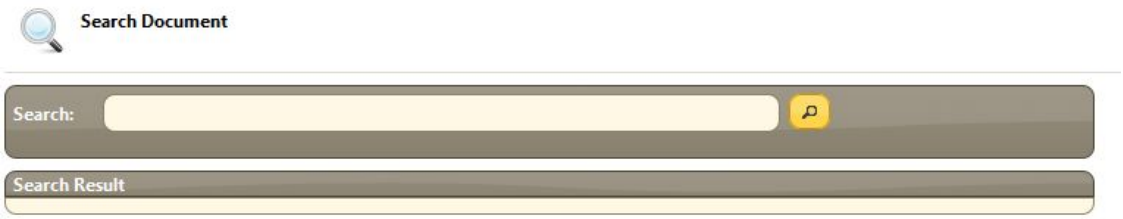
Gambar 4.3 Tampilan halaman detail dokumen

1.1.7. Aplikasi Pencarian Dokumen

Aplikasi yang digunakan pengguna untuk memasukkan kata kunci pencarian terhadap *file-file* yang telah terindeks ke dalam basis data. Aplikasi ini berbasis web. Tampilan awal aplikasi ini berupa inputan untuk melakukan pencarian *file*.

Terdapat *textbox* yang menunjukkan tempat untuk memasukkan kata kunci. Setelah mengetikkan kata kunci pencarian lalu pengguna dapat menekan tombol enter pada *keyboard* atau bisa juga dengan mengklik tombol search yang ada pada layar.

Setelah tahapan sebelumnya, maka selanjutnya pengguna akan di arahkan menuju halaman hasil pencarian.



Gambar 4.4 Tampilan halaman pencarian dokumen

Halaman dibawah ini merupakan halaman yang berupa hasil pencarian *file*. Setiap nama *file* bisa diklik sehingga pengguna dapat mengunduh *file* tersebut.



Gambar 4.5 Tampilan halaman hasil pencarian

4.2 Pengujian

Tahap pengujian perangkat lunak dilakukan terhadap fungsionalitas perangkat lunak. Pengujian dilakukan dengan data masukan seperti disebutkan pada sub bab 4.1.3. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

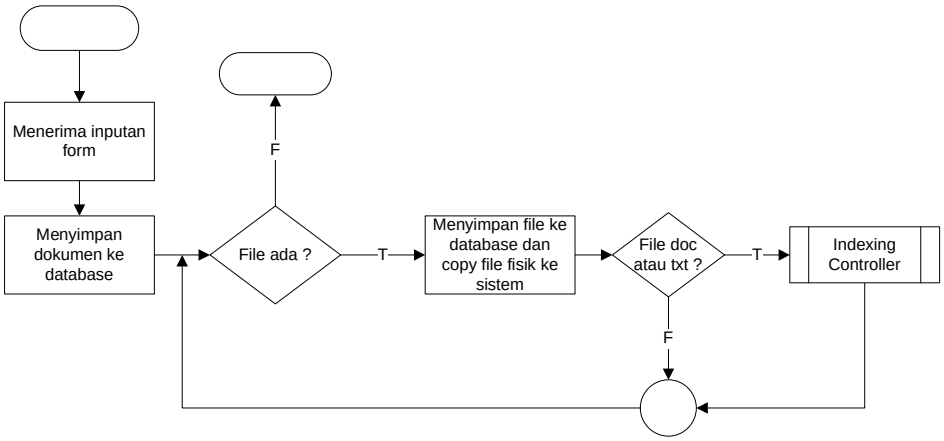
4.3 Pengujian Alpha

Pengujian alpha merupakan pengujian yang menitikberatkan pada hasil *output* dari kendali *input* yang dimasukan pada tampilan kendali *input* form tampilan, dimana pengujian dikatakan berhasil apabila *output* sesuai dengan kendali *input* yang dimasukan pada tiap tampilan.

1. Proses *upload file*

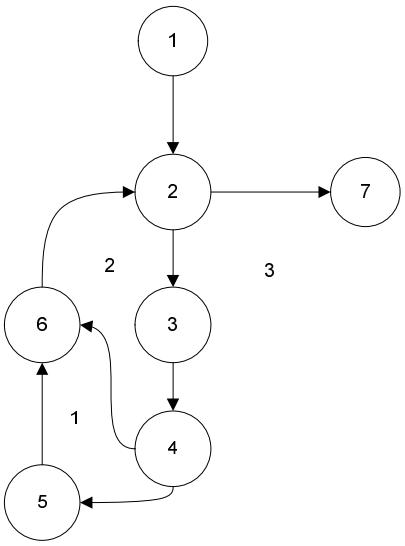
   Menguji apakah dokumen yang dibuat sudah disimpan ke database.

   a. *Flow Chart*



Gambar 4.6 *Flow chart* proses *upload file*

b. *Flow Graph*



Gambar 4.7 *Flow graph* proses *upload file*

c. *Cyclomatic Complexity* V(G)

E = 8, N = 7

V(G) = E – N + 2 = 8 – 7 + 2 = 3

V(G) = Region = 3

d. *Independent Paths*

Path 1 = 1–2–7

Path 2 = 1–2–3–4–6–2–7

Path 3 = 1–2–3–4–5–6–2–7

e. Hasil Uji

Tabel 4.2 Hasil uji *upload file*

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	Inputan tidak memiliki file	Data dokumen tersimpan di database	Data dokumen tersimpan di database	*Path* Dikerjakan
2	Inputan tidak memiliki file doc atau txt	Data dokumen dan file tersimpan di database	Data dokumen dan file tersimpan di database	*Path* Dikerjakan
3	Inputan memiliki file doc atau txt	Data dokumen dan file tersimpan di database. Melakukan proses penentuan indeks	Data dokumen dan file tersimpan di database. Melakukan proses penentuan indeks	*Path* Dikerjakan

2. Procedure indexingController

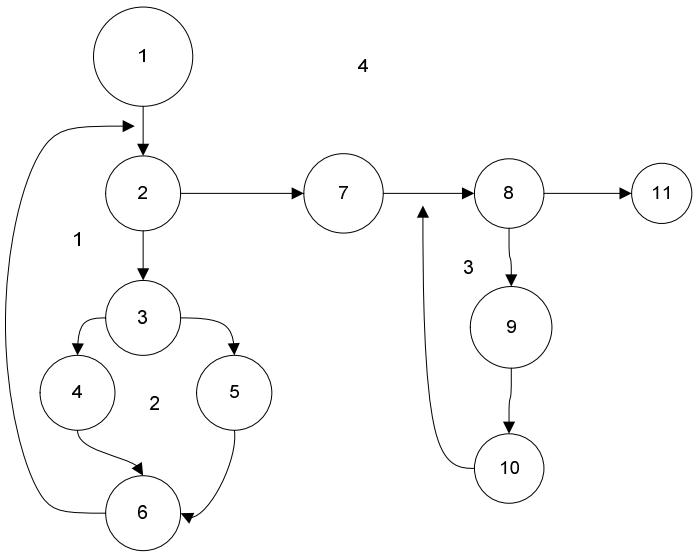
Menguji apakah proses penentuan index sudah sesuai dengan yang diharapkan.

a. *Source Code*

```
1. String directory = FileServiceUtil.userDocumentDirectory(file.getDocumentId().getId());
   java.io.File filePhysical = new java.io.File(FileServiceUtil.projectDirectory() + "/" + directory +
   file.getName() + "." + file.getExt());
   Extractor extractor = new Extractor(filePhysical);
```

```
        Tokenizer tokenizer = new Tokenizer();
        List<String> tokens = tokenizer.tokenize(extractor.getContent());
        List<String> non_stopwords = stop.stopWord(tokens);
        List<String> tagged = tag.tag(non_stopwords);
2.      for(String token:tagged){
3.              if(FileServiceUtil.isDictionaryWord(token))
4.        dictionaryList.add(token);
5.              else nonDictionaryList.add(token);
6.      }
7.      Stemmer stem = new Stemmer();
        List<String> stemmed = stem.stemming(nonDictionaryList);
        dictionaryList.addAll(stemmed);
        Map<String, Integer> term_freq = tokenizer.termWeighting(dictionaryList);
        Iterator<String> iter = term_freq.keySet().iterator();
8.      while (iter.hasNext()) {
9.              String temp = iter.next();
                Dictionary newDict = new Dictionary();
                newDict.setTerm(temp);
                FileServiceUtil.dictionarySave(newDict);
                Posting newPost = new Posting();
                 newPost.setId(java.util.UUID.randomUUID().toString());
                 newPost.setDictId(newDict);
                 newPost.setFileId(file);
                newPost.setFreq(term_freq.get(temp));
              FileServiceUtil.postSave(newPost);
10.     }
11.
```

b. *Flow Graph*



Gambar 4.8 *Flow graph* indexingController

c. *Cyclomatic Complexity* V(G)

E = 13, N = 11

V(G) = E – N + 2 = 13 – 11 + 2 = 4

V(G) = Region = 4

d. *Independent Paths*

Path 1 = 1–2–7–8–11

Path 2 = 1–2–7–8–9–10–11

Path 3 = 1–2–3–4–6–2–7–8–9–10–11

Path 4 = 1–2–3–5–6–2–7–8–9–10–11

e. Hasil Uji

Tabel 4.3 Hasil uji method indexingController

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	File tidak mengandung teks	Tidak ada posting yang disimpan ke database	Tidak ada posting yang disimpan ke database	*Path* Dikerjakan
3	File mengandung teks yang ada dalam kamus	Data posting disimpan ke database	Data posting disimpan ke database	*Path* Dikerjakan
4	File mengandung teks yang tidak ada dalam kamus	Data posting disimpan ke database	Data posting disimpan ke database	*Path* Dikerjakan

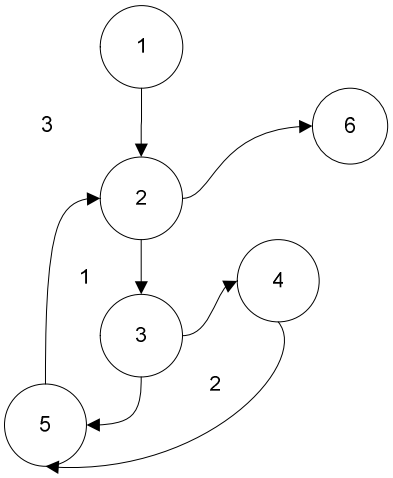
3. Procedure tokenize

Menguji apakah hasil *tokenizing* dari teks sudah sesuai yang diharapkan.

a. *Source Code*

```
1.  text = text.replace('.', ' ');
    text = text.replace(',', ' ');
    text = text.replace('?', ' ');
    text = text.replace('!', ' ');
    text = text.replace('"', ' ');
    text = text.replace(':', ' ');
    text = text.replace(';', ' ');
    text = text.replace('-', ' ');
    text = text.replace('\n', ' ');
    text = text.replace('(', ' ');
    text = text.replace(')', ' ');
    text = text.replace('[', ' ');
    text = text.replace(']', ' ');
    text = text.replace('-', ' ');
    text = text.replace('\n', ' ');
    text = text.replace("", " ");
    List<String> extracted = new ArrayList<String>();
    List<String> tokens = new ArrayList<String>();
    extracted.addAll(Arrays.asList(text.toLowerCase().split(" ")));
2.  for(String str:extracted){
3.          if(!str.trim().equals(""))
4.                  tokens.add(str.trim());
5.  }
6.  return tokens;
```

b. *Flow Graph*



Gambar 4.9 *Flow graph* proses *tokenizing*

c. *Cyclomatic Complexity* V(G)

E = 7, N = 6

V(G) = E – N + 2 = 7 – 6 + 2 = 3

V(G) = Region = 3

d. *Independent Paths*

Path 1 = 1–2–6

Path 2 = 1–2–3–5–2–6

Path 3 = 1–2–3–4–5–2–6

e. Hasil Uji

Tabel 4.4 Hasil uji proses tokenizing

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	Inputan teks kosong	Mengembalikan daftar kosong	Mengembalikan daftar kosong	*Path* Dikerjakan
2	Inputan teks mengandung spasi	Spasi tidak dimasukkan kedalam daftar	Spasi tidak dimasukkan kedalam daftar	*Path* Dikerjakan
3	Inputan teks berisi kata	Kata dimasukkan ke dalam daftar	Kata dimasukkan ke dalam daftar	*Path* Dikerjakan

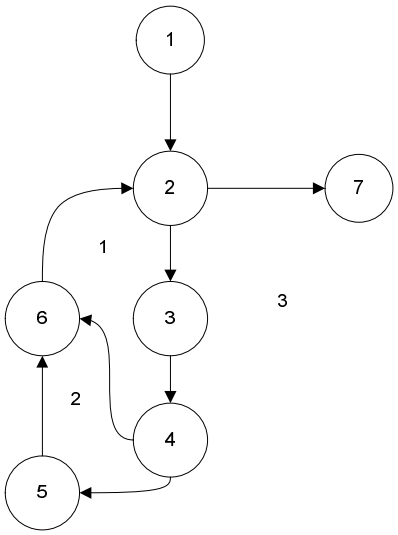
4. Procedure stopWord

Menguji apakah proses menghilangkan kata-kata yang tidak penting dalam daftar sudah sesuai dengan yang diharapkan.

a. *Source Code*

```
1.  List<String> non_stopwords = new ArrayList<String>();
    Iterator<String> iter = tokens.iterator();
2.  while(iter.hasNext()){
3.          String temp = iter.next();
4.          if(!checkStopwords(temp))
5.                  non_stopwords.add(temp);
6.  }
7.  return non_stopwords;
```

b. *Flow Graph*



Gambar 4.10 *Flow graph* proses stopWord

c. *Cyclomatic Complexity* V(G)

E = 8, N = 7

V(G) = E – N + 2 = 8 – 7 + 2 = 3

V(G) = Region = 3

d. *Independent Paths*

Path 1 = 1–2–7

Path 2 = 1–2–3–4–6–2–7

Path 3 = 1–2–3–4–5–6–2–7

e. Hasil Uji

Tabel 4.5 Hasil uji proses stopWord

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	Inputan daftar kosong	Mengembalikan daftar kosong	Mengembalikan daftar kosong	*Path* Dikerjakan
2	Inputan daftar kata mengandung yang tidak dibutuhkan	Kata yang tidak dibutuhkan tidak dimasukkan kedalam daftar	Kata yang tidak dibutuhkan tidak dimasukkan kedalam daftar	*Path* Dikerjakan
3	Inputan daftar kata mengandung yang dibutuhkan	Kata yang dibutuhkan dimasukkan kedalam daftar	Kata yang dibutuhkan dimasukkan kedalam daftar	*Path* Dikerjakan

5. Procedure stem

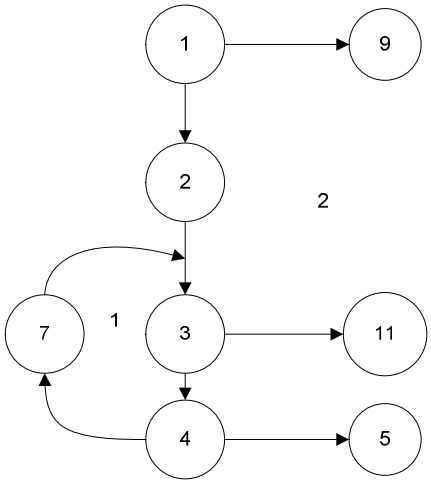
Menguji apakah proses *stemming* sudah sesuai dengan yang diharapkan.

a. *Source Code*

```
1.  if (str.length() > 0) {
2.          char[] c = str.toCharArray();
3.          for (int i = 0; i < c.length; i++) {
4.                  if (!Character.isLetter(c[i])) {
5.                          return null;
6.                  }
7.          }
8.  } else {
9.          return null;
10. }
11. str = step1a(str);
    str = step1b(str);
    str = step1c(str);
    str = step2(str);
```

```
str = step3(str);
str = step4(str);
str = step5a(str);
str = step5b(str);
return str;
```

b. *Flow Graph*



Gambar 4.11 *Flow graph* proses stem

c. *Cyclomatic Complexity* V(G)

E = 8, N = 8

V(G) = E – N + 2 = 8 – 8 + 2 = 2

V(G) = Region = 2

d. *Independent Paths*

Path 1 = 1–2–3–4–5

Path 2 = 1–2–3–4–7–11

e. Hasil Uji

Tabel 4.6 Hasil uji proses stem

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	Masukkan berupa angka	Mengembalikan null	Mengembalikan null	*Path* Dikerjakan
2	Masukkan berupa kata	Mengembalikan kata hasil proses *stemming*	Mengembalikan kata hasil proses *stemming*	*Path* Dikerjakan

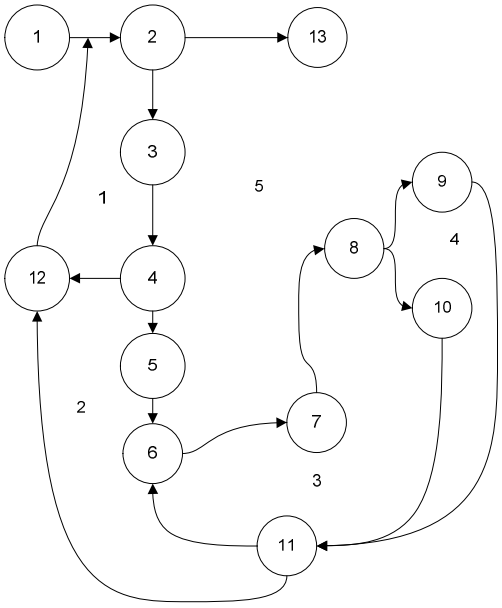
6. Proses menghitung tf-idf

Menguji apakah proses *stemming* sudah sesuai dengan yang diharapkan.

a. *Source Code*

```
1.  Iterator<String> iter = listTermStemmed.iterator();
    Map<File, Double> resultsMap = new HashMap<File, Double>();
    int N = searchingDao.totalFile();
2.  while (iter.hasNext()) {
3.      String token = iter.next();
        List<Posting> list = searchingDao.getPosting(token);
4.      if(!list.isEmpty()) {
5.          int df = list.size();
            Double idf = Math.log(N/df);
6.          for(Posting post : list){
7.              Double tfidf = post.getFreq() * idf;
8.              if(resultsMap.containsKey(post.getFileId())){
9.      resultsMap.put(post.getFileId(),resultsMap.get(post.getFileId()) + tfidf);
                }else{
10.                 resultsMap.put(post.getFileId(), tfidf);
                }
11.         }
        }
12. }
13. Map<File, Double> sortedMap =
    ValueComparator.sortByComparator(resultsMap);
```

b. *Flow Graph*



Gambar 4.12 *Flow graph* proses menghitung tf-idf

c. *Cyclomatic Complexity* V(G)

E = 16, N = 13

V(G) = E – N + 2 = 16 – 13 + 2 = 5

V(G) = Region = 5

d. *Independent Paths*

Path 1 = 1–2–13

Path 2 = 1–2–3–4–12–2–13

Path 3 = 1–2–3–4–5–6–7–8–9–11–12–2–13

Path 4 = 1–2–3–4–5–6–7–8–10–11–12–2–13

Path 5 = 1–2–3–4–5–6–7–8–9–11–6–7–8–10–11–12–2–13

e. Hasil Uji

Tabel 4.7 Hasil uji perhitungan tf-idf

Path No.	***Node* (n), Input Condition**	***Excpected Result***	***Test Result***	***Conclusion***
1	Kata kunci pencarian kosong	Daftar file kosong	Daftar file kosong	*Path* Dikerjakan
2	Memasukkan kata kunci yang tidak ada di database	Daftar file kosong	Daftar file kosong	*Path* Dikerjakan
3	Memasukkan beberapa kata kunci yang ada dalam file yang sama	Daftar file terisi	Daftar file terisi	*Path* Dikerjakan
4	Memasukkan beberapa kata kunci yang ada dalam file yang berbeda	Daftar file terisi	Daftar file terisi	*Path* Dikerjakan
5	Masukan gabungan dari path no 3 dan 4	Daftar file terisi	Daftar file terisi	*Path* Dikerjakan

4.3.4 Kesimpulan Pengujian Alpha

Berdasarkan hasil pengujian diatas ada beberapa kesimpulan sebagai berikut :

1. Aplikasi bisa melakukan proses *upload* dokumen sekaligus melakukan penentuan indeks terhadap *file-file* yang di*upload.*
2. Aplikasi bisa melakukan pencarian *file-file* yang ada, berdasarkan query pencarian yang dimasukkan.
3. Aplikasi bisa menampilkan hasil pencarian file yang telah diurutkan, sehingga bisa dihasilkan file yang lebih relevan dengan kebutuhan pengguna.

4.4 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif dimana diuji secara langsung oleh beberapa *user* dengan membuat kuisoner mengenai kepuasan *user* dengan kandungan poin syarat *user friendly* untuk selanjutnya dibagikan kepada *user* dengan mengambil *sample* sebanyak 10 orang. Dari hasil kuesioner tersebut akan dilakukan perhitungan untuk dapat diambil kesimpulan terhadap penelitian penerapan sistem yang baru.

4.4.1 Kuesioner Pengguna

Kuesioner disebarkan menggunakan teknik sampling yaitu *Sample Random Sampling* yang disebarkan kepada 10 orang pengguna. Dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap aplikasi yang dibuat. Kuesioner ini terdiri dari 5 pertanyaan untuk

pengguna (hasil kuesioner dari 10 pengguna dapat dilihat pada lampiran) dengan menggunakan skala likert dengan skala 1 sampai 10, dengan ketentuan skala untuk tiap pertanyaan berikut :

1. Apakah aplikasi ini mudah digunakan dan mudah dipelajari?

   A. Sangat Baik
   B. Baik
   C. Cukup Baik
   D. Kurang Baik
   E. Tidak Baik

2. Apakah aplikasi ini sudah dilengkapi dengan pesan – pesan kesalahan sehingga dapat meminimalisasi kesalahan inputan?

   A. Sangat Baik
   B. Baik
   C. Cukup Baik
   D. Kurang Baik
   E. Tidak Baik

3. Apakah aplikasi ini membantu dalam proses pencarian dokumen?

   A. Sangat Baik
   B. Baik
   C. Cukup Baik
   D. Kurang Baik
   E. Tidak Baik

4. Apakah penggunaan tema dan *layout* dari aplikasi ini sudah baik?

   A. Sangat Baik
   B. Baik
   C. Cukup Baik
   D. Kurang Baik
   E. Tidak Baik

5. Apakah aplikasi ini sudah layak digunakan?

   A. Sangat Baik
   B. Baik
   C. Cukup Baik
   D. Kurang Baik
   E. Tidak Baik

Bedasarkan data hasil kuisoner tersebut, dapat dicari prosentase masing-masing jawaban dengan menggunakan rumus : Y = P/Q * 100%

Keterangan :

P = Banyaknya jawaban responden tiap soal.

Q = Jumlah responden

Y = Nilai persentase

Berikut ini adalah hasil persentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus diatas. Kuisoner ini diujikan kepada 10 orang.

1. Apakah aplikasi ini mudah digunakan dan mudah dipelajari?

Tabel 4.8 Hasil Pengujian Kuesioner Soal No 1

Pertanyaan	No	Pilihan	Responden	Persentase
1	1	Sangat Baik	3	30%
	2	Baik	6	60%
	3	Cukup Baik	1	10%
	4	Tidak Baik	0	0%
	5	Kurang Baik	0	0%
Total			**10**	**100%**

Berdasarkan hasil prosentase diatas maka dapat disimpulkan bahwa sebanyak 3 orang atau 30% mengatakan sangat baik, 6 orang atau 60% mengatakan baik dan 1 orang atau 10% mengatakan cukup baik.

2. Apakah aplikasi ini sudah dilengkapi dengan pesan – pesan kesalahan sehingga dapat meminimalisasi kesalahan inputan?

Tabel 4.9 Hasil Pengujian Kuesioner Soal No 2

Pertanyaan	No	Pilihan	Responden	Persentase
2	1	Sangat Baik	0	0%
	2	Baik	8	80%
	3	Cukup Baik	2	20%
	4	Kurang Baik	0	0%
	5	Tidak Baik	0	0%
Total			**10**	**100%**

Berdasarkan hasil prosentase diatas maka dapat disimpulkan bahwa sebanyak 8 orang atau 80% mengatakan baik dan 2 orang atau 20% mengatakan cukup baik.

3. Apakah aplikasi ini membantu dalam proses pencarian dokumen?

Tabel 4.10 Hasil Pengujian Kuesioner Soal No 3

Pertanyaan	No	Pilihan	Responden	Persentase
3	1	Sangat Baik	4	40%
	2	Baik	5	50%
	3	Cukup Baik	1	10%
	4	Kurang Baik	0	0%
	5	Tidak Baik	0	0%
Total			**10**	**100%**

Berdasarkan hasil prosentase diatas maka dapat disimpulkan bahwa sebanyak 4 orang atau 40% mengatakan sangat baik, 5 orang atau 50% mengatakan baik dan 1 orang atau 10% mengatakan cukup baik.

4. Apakah penggunaan tema dan *layout* dari aplikasi ini sudah baik?

Tabel 4.11 Hasil Pengujian Kuesioner Soal No 4

Pertanyaan	No	Pilihan	Responden	Persentase
4	1	Sangat Baik	2	20%
	2	Baik	1	10%
	3	Cukup Baik	6	60%
	4	Kurang Baik	1	10%
	5	Tidak Baik	0	0%
Total			**10**	**100%**

Berdasarkan hasil prosentase diatas maka dapat disimpulkan bahwa sebanyak 2 orang atau 20% mengatakan sangat baik, 1 orang atau 10% mengatakan baik, 6 orang atau 60% mengatakan cukup baik dan 1 orang atau 10% mengatakan kurang baik.

5. Apakah aplikasi ini sudah layak digunakan?

Tabel 4.12 Hasil Pengujian Kuesioner Soal No 5

Pertanyaan	No	Pilihan	Responden	Persentase
5	1	Sangat Baik	2	20%
	2	Baik	6	60%
	3	Cukup Baik	2	20%
	4	Kurang Baik	0	0%
	5	Tidak Baik	0	0%
Total			**10**	**100%**

Berdasarkan hasil prosentase diatas maka dapat disimpulkan bahwa sebanyak dan 2 orang atau 20% mengatakan sangat baik dan 6 orang atau 60% mengatakan baik dan 2 orang atau 20% mengatakan cukup baik.

4.4.2 Kesimpulan Pengujian Beta

Berdasarkan hasil prosentase diatas yang didapatkan dari pengujian beta terhadap pengguna. Maka dapat disimpulkan bahwa penerapan aplikasi pencarian dokumen ini sudah baik karena bisa memberikan kemudahan dalam proses pencarian dokumen.