# Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi

**Panji Novantara*[1], Opin Pasruli[2]**
[1,2]Fakultas Ilmu Komputer Universitas Kuningan
*[1]**panji@uniku.ac.id**,[2]Opin.Pasruli@yahoo.com

## ABSTRAK

Karya tulis ilmiah merupakan bagian dari tuntutan formal akademik di Perguruan Tinggi. Dilihat dari jenisnya, karya tulis ilmiah terdiri atas makalah, laporan, jurnal, skripsi dan lain-lain. Karya tulis ilmiah yang digunakan sebagai salah satu syarat mahasiswa untuk menyelesaikan program studi S1 yaitu skripsi. Dalam pembuatan skripsi hampir tidak dapat dihindari sering ditemukannya kecurangan, seperti mengambil ide maupun menjiplak karya tulis orang lain atau sering juga disebut dengan plagiarisme. Plagiarisme adalah tindakan menjiplak karya seseorang dan kemudian mengakuinya sebagai karya sendiri. Untuk itu perlu dilakukan upaya-upaya penekanan tindakan plagiarisme sejak dini. Maka diperlukan sisitem pendeteksi plagiarisme untuk mengetahui seberapa besar tingkat kemiripan dokumen skripsi. Agar bisa digunakan untuk memutuskan apakah skripsi ini merupakan plagiarisme atau bukan. Untuk mengetahui seberapa besar persentase kesamaan bisa menggunakan pendekatan string metrik, yaitu dengan melakukan perbandingan string menggunakan algoritma *Jaro-Winkler Distance* dengan tahapan melakukan proses menghitung panjang string, menentukan jumlah karakter yang sama, menghitung nilai transposisi, menghitung nilai *Jaro-Distance*, menghitung nilai Jaro-Winkler Distance, dan menghitung tingkat kemiripan (*similarity*). Pembuatan aplikasi menggunakan bahasa pemrograman Java dengan menggunakan NetBeans IDE 8.0. Sistem pendeteksi plagiarisme dengan menggunakan algoritma *Jaro-Winkler Distance* dapat digunakan untuk mendeteksi plagiarisme dokumen skripsi dengan cara melakukan perbandingan antara dokumen asli dan dokumen uji yang diinputkan untuk mengetahui tingkat kemiripan (*similarity*) dari dokumen skripsi yang diuji.

Kata Kunci : Skripsi, Plagiarisme, Algoritma *Jaro-Winkler Distance*, NetBeans IDE 8.0

## 1. PENDAHULUAN

Karya tulis ilmiah di Perguruan Tinggi merupakan bagian dari tuntutan formal akademik. Dilihat dari tujuan penulisannya, karya tulis ilmiah dibedakan menjadi dua, pertama untuk memenuhi tugas-tugas perkuliahan seperti makalah, laporan dan lain-lain. Kedua, karya tulis ilmiah yang digunakan sebagai salah satu syarat mahasiswa untuk menyelesaikan program studi, yaitu skripsi untuk S1, dan tugas akhir untuk D3.

Seperti di Perguruan Tinggi pada umumnya, kedudukan skripsi di Universitas Kuningan sangat penting dan merupakan bagian dari karya tulis ilmiah untuk menyelesaikan Program Sarjana (S1). Skripsi merupakan bukti kemampuan akademik mahasiswa dalam penelitian yang berhubungan dengan bidang keilmuannya. Melalui skripsi, mahasiswa mengungkapkan pikirannya secara sistematis sesuai dengan kaidah-kaidah keilmuan. Akan tetapi dengan perkembangan teknologi informasi ternyata membawa dampak negatif juga, hal ini dikarenakan semakin banyaknya informasi yang dapat digunakan sebagai referensi dalam pembuatan skripsi. Sehingga hampir tidak dapat dihindari sering ditemukannya kecurangan, seperti mengambil ide maupun menjiplak karya tulis orang lain atau sering juga disebut dengan plagiarisme.

Plagiarisme adalah tindakan menjiplak karya seseorang dan kemudian mengakuinya sebagai karya sendiri. Aksi plagiarisme merupakan tindakan yang sangat merugikan, tidak hanya merugikan orang lain yang karya tulisnya dijiplak tetapi merugikan juga orang yang melakukan tindakan plagiat itu sendiri. Dengan tindakan plagiarisme, secara tidak langsung dapat mematikan kreatifitas seseorang dan menimbulkan kecendrungan sikap malas dan tidak mau berfikir karena sudah terbiasa mengambil sesuatu yang bukan miliknya. Untuk itu perlu dilakukan upaya-upaya penekanan tindakan plagiarisme sejak dini.

Oleh karena itu dibutuhkan suatu sistem yang dapat digunakan untuk pendeteksian plagiarisme dokumen skripsi. Yaitu dengan menggunakan pendekatan string metrik, yang mana dengan melakukan perbandingan string dan memasukan kedalam fungsi matematis tertentu. Terdapat beberapa algoritma yang berdasarkan kepada string matric diantaranya yaitu *Levenshtein Distance*, *TF/IDF*, *Needleman-Wunsch Distance*, *Jaro-Winkler Distance*, dan lain sebagainya. Dari beberapa algoritma yang telah disebutkan penulis memilih algoritma *Jaro-Winkler Distance*, algoritma ini di pilih di karenakan dapat memeriksa kemiripan dengan hanya melewai sedikit proses sehingga meminimalkan waktu dan relatif cepat.

## 2. LANDASAN TEORI

### 2.1. Plagiarisme

Menurut Kamus Besar Bahasa Indonesia (KBBI) Plagiarisme adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri. (KBBI, 1997: 775).

Plagiat dapat dianggap sebagai tindak pidana karena mencuri hak cipta orang lain. Di dunia pendidikan, pelaku plagiarisme akan mendapat hukuman berat seperti dikeluarkan dari sekolah/universitas. Pelaku plagiat disebut sebagai plagiator (Eko Nugroho, 2011).

Beberapa tipe plagiarisme :

1. *Word-for-word* plagiarism adalah menyalin setiap kata secara langsung tanpa diubah sedikitpun.
2. *Plagiarism of authorship* adalah mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang yang sebenarnya.
3. *Plagiarism of ideas* adalah mengakui hasil pemikiran atau ide orang lain.

Metode pendeteksi plagiarisme dibagi menjadi tiga bagian yaitu metode perbandingan teks lengkap, metode dokumen *fingerprinting*, dan metode kesamaan kata kunci. (Eko Nugroho, 2011)

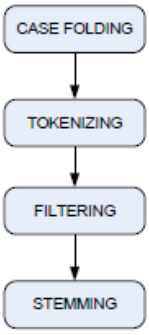

*Gambar 2.1 Metode Pendeteksi Plagiarisme (Stein dan Eissen, 2006).*

Klasifikasi berdasarkan proporsi atau persentasi kata, kalimat, paragraf yang dibajak (Sudigdo Sastroasmoro 2007) :

- Plagiarisme ringan : < 30%
- Plagiarisme sedang : 30 - 70%
- Plagiarisme besar : > 70%

### 2.2. Ekstraksi Dokumen

Teks yang akan dilakukan proses *teks mining*, pada umumnya memiliki beberapa karakteristik, diantaranya adalah memiliki dimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Cara yang dignakan dalam mempelajari suatu data teks, adalah dengan terlebih dahulu menentukan fitur-fitur yang mewakili setiap kata untuk setiapfitur ang ada pada dokumen. Sebelum menentukan fitur-fitur yang mewakili, diperlukan tahapan *preprocessing* yang dilakukan secara umum dalam *teks mining* pada dokumen, yaitu *case folding*, *tokenizing*, *filtering*, *stemming*, *tagging* dan *analizing*. Untuk tahapan dari *preprocessing* seperti terlihat pada gambar 2.2. (Triawati, 2009).



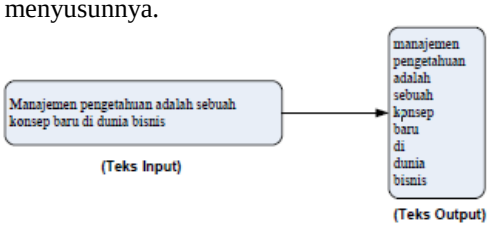

*Gambar 2.2. Tahapan Preprocessing*

Untuk penjelasan tahapan *preprocessing* seperti dibawah ini :

1. *Case Folding* adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima. Karakter

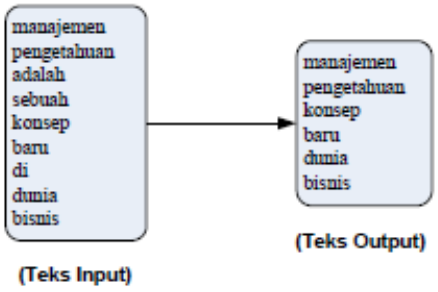
selain huruf dihilangkan dan dianggap *delimiter*

2. Tokenizing/parsing adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya.



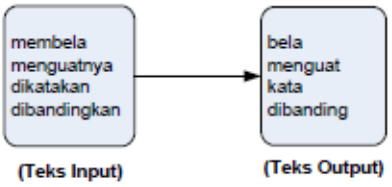

*Gambar 2.3. Tokenizing*

3. *Filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist/wordlist* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh *stopwords* adalah "yang", "dan", "di", "dari", dan seterusnya.




*Gambar 2.4. Filtering*

4. *Stemming* adalah tahap mencari *root* kata kunci dari setiap kata hasil *filtering*. Pada tahap ini dilakukan proses pengambilan berbagai bentuk kata ke dalam suatu representasi yang sama. Contoh dari tahapan ini pada teks adalah seperti pada gambar 2.5.




*Gambar 2.5. Stemming*

## 2.3. String Metrik

*String metric* atau *similarity metric* adalah kelas matrik berbasis tekstual yang dapat menghasilkan nilai kesamaan atau ketidaksamaan dari dua teks string untuk proses perbandingan dan penyamaan. *String metric* biasanya digunakan dalam deteksi kecurangan, analisa fingerprint, deteksi plagiarisme, ontology *merging*, analisa DNA, analisa RNA, analisa image, databse deduplication, data mining, web interfaces, dsb. Beberapa algoritma yang berdasarkan kepada string metric diantaranya adalah *Levenshtein distance*, TF/IDF, *Needleman-Wunsch distancei*, *Jaro-Winkler distance*, dsb. Dari algoritma yang telah disebutkan diatas *Jaro-Winkler distance* memiliki kemampuan yang baik di dalam pencocokan string yang relatif pendek. Algoritma *Jaro-Winkler distance* juga sering digunakan dalam pendeteksian duplikat (Kurniawati, Sulistyo dan Sazali 2010).

## 2.4. Algoritma Jaro-Winkler distance

Algoritma *Jaro-Winkler Distance* merupakan varian dari Jaro Distance metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua string, biasanya algoritma ini digunakan didalam pendeteksian duplikat. Semakin tinggi *Jaro-Winkler Distance* untuk dua string, semakin mirip dengan string tersebut. Nilai normalnya yaitu 0 untuk menandakan tidak ada kesamaan, dan 1 untuk menandakan adanya kesamaan. (Kurniawati, Sulistyo dan Sazali 2010).

Dasar dari algoritma ini sendiri memiliki tiga bagian :

1. Menghitung panjang string.
2. Menentukan jumlah karakter yang sama didalam dua string.
3. Menentukan jumlah transposisi.

Pada algoritma Jaro-Winkler Distance digunakan rumus untuk menghitung jarak ($d_j$) antara dua string yaitu $S_1$ dan $S_2$ adalah

$$d_j = \frac{1}{3} \times \left( \frac{m}{S_1} + \frac{m}{S_2} + \frac{m-t}{m} \right)$$

Dimana :

$m$ = Jumlah karakter yang sama persis
$|S_1|$ = Panjang string 1
$|S_2|$ = Panjang string 2
$t$ = Jumlah transposisi

Jarak teoritis dua buah karakter yang disamakan dapat dibenarkan jika tidak melebihi :

$$\left( \frac{\max(|S_1|, |S_2|)}{2} \right) - 1$$

Akan tetapi bila mengacu kepada nilai yang akan dihasilkan oleh algoritma *Jaro-Winkler Distance* maka nilai maksimalnya adalah 1, yang menendakan kesamaan string yang di bandingkan mencapai seratus persen atau sama persis. Algoritma *Jaro-Winkler Distance* menggunakan *prefix scale* (*p*), menurut Winkler nilai standar untuk konstanta ini adalah $p = 0.1$. Dan *prefix length* (*l*) yaitu untuk menyatakan panjang *prefix* atau panjang karakter yang sama sampai ditemukan ketidaksamaan (maksimum 4 karakter), maka untuk menghitung *Jaro-Winkler Distance* ($d_w$) adalah :

$$d_w = d_j + \left(l \times p\left(1 - d_j\right)\right)$$

Dimana :

$d_j$ = Jaro distance untuk string $S_1$ dan $S_2$

$l$ = Panjang *prefix* (panjang karakter yang sama sebelum ditemukan ketidaksamaan) nilai maksimum 4 karakter

$p$ = Konstanta *scaling factor* (Nilai standar untuk konstanta ini menurut Winkler adalah $p = 0.1$).

Berikut ini merupakan contoh perhitungan algoritma *Jaro-Winkler Distance*. Jika string $S_2$ MARTHA dan string $S_2$ MARHTA maka :

$m$ = 6

$S_1$ = 6

$S_2$ = 6

Karakter yang tertukar hanyalah *T* dan *H*. Maka $t$ = 1. Maka nilai Jaro Distance adalah :

$$d_j = \frac{1}{3} \times \left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944$$

Kemudian bila diperhatikan susunan $S_1$ dan $S_2$ dapat diketahui nilai $l$ = 3, dan dengan nilai konstan $p$ = 0.1. maka nilai Jaro-Winkler distance adalah :

$$d_w = 0.944 + \left(3 \times 0.1(1 - 0.944)\right) = 0.961$$

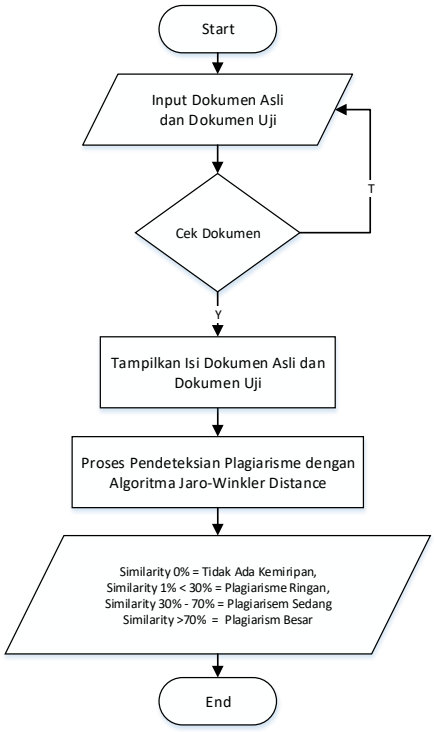
## 3. ANALISIS DAN PERANCANGAN

Tindakan plagiarisme dalam pembuatan tugas akhir atau skripsi sangatlah mungkin terjadi, hal ini merupakan salah satu dampak negatif dari perkembangan teknologi informasi. Seperti mengambil ide maupun menjiplak karya tulis orang lain.

Oleh karena itu dengan dibuatnya sistem pendeteksi plagiarisme ini di harapkan dapat membantu dalam melakukan pendeteksian plagiarisme dokumen skripsi.

### 3.1. Flowchart Sistem

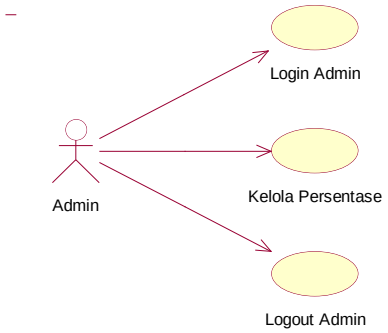
Untuk menggambarkan langkah-langkah yang dilalui dalam proses komputasi pada proses pendeteksian plagiarisme dokumen digunakan *flowchart* atau diagram alir. Untuk flowchart sistemnya dapat dilihat pada Gambar 3.1.




*Gambar 3.1. Flowchart Sistem*

### 3.2. *Use Case Diagram*

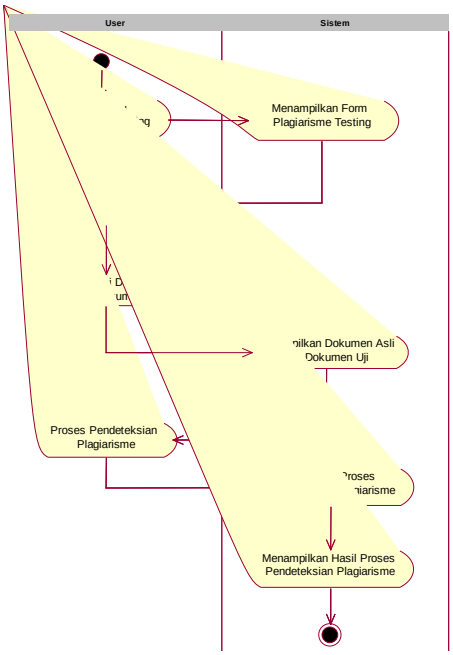
*Use Case Diagram* menggambarkan sistem dari perspektif pandangan pemakai akhir (end user). Dalam sistem ini terdiri dari 2 aktor yaitu Admin dan *User*. Admin dapat melakukan proses login, proses kelola persentase dan dapat melakukan proses logout. Sedangkan *User* dapat melakukan proses pengecekan plagiarisme dan dapat melihat hasil pengecekan. Untuk *Use Case Diagram* dapat dilihat pada Gambar 3.2.



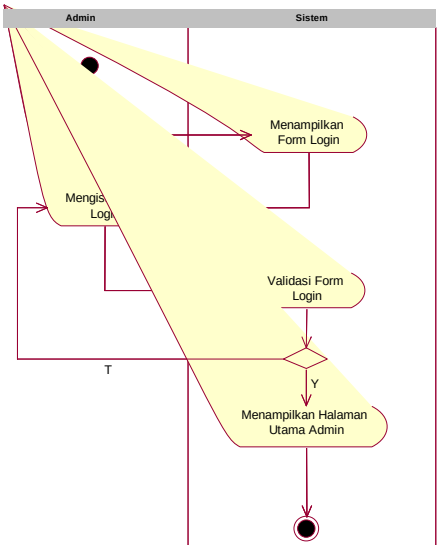

*Gambar 3.2. Use Case Diagram*

### 3.3. *Activity Diagram*

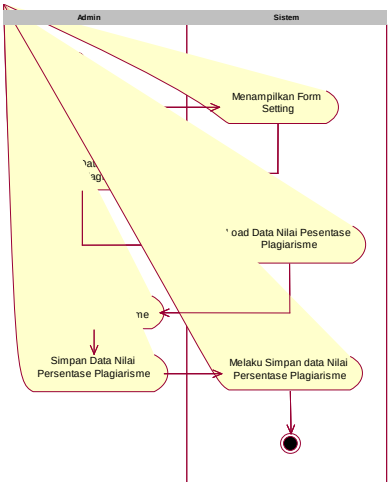
*Activity Diagram* adalah untuk menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Berikut ini akan digambarkan aktivitas-aktivitas diagram dari masing-masing aktor :




*Gambar 3.4. Activity Diagram Pengecekan Plagiarisme*



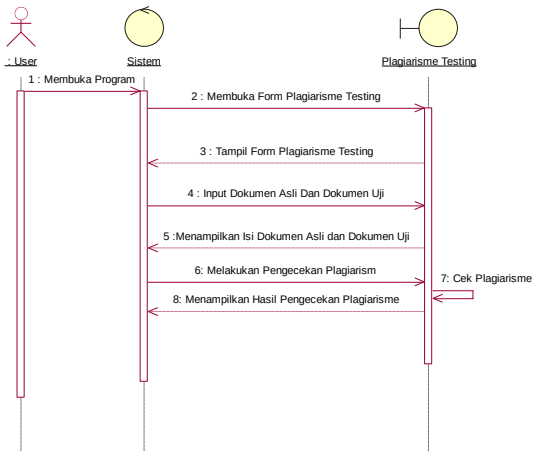

*Gambar 3.5. Activity Diagram Login Admin*



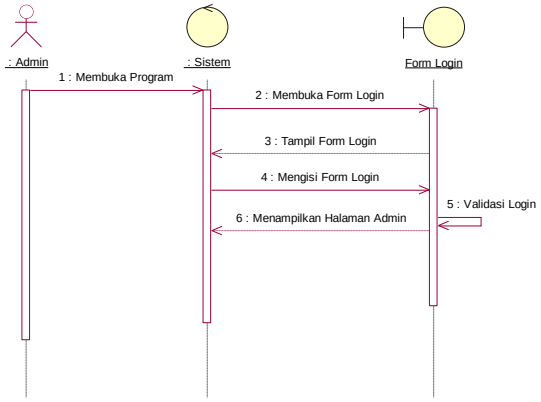

*Gambar 3.6. Activity Diagram Kelola Persentase*

### 3.4. *Sequence Diagram*

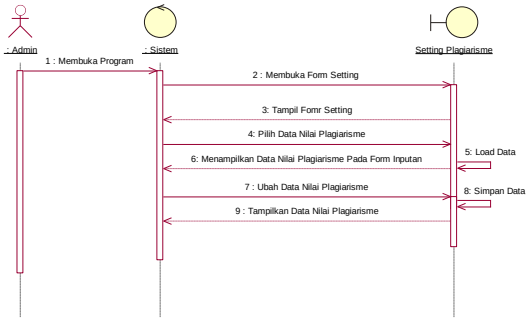
*Sequence Diagram* menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertical (waktu) dan dimensi horizontal (objek – objek yang terkait). Pada *Sequence Diagram* ini menggambarkan scenario atau rangkaian langkah langkah yang dilakukan pada sistem sebagai respon dari sebuah event untuk menghasilkan output tertentu.




*Gambar 3.7. Sequence Diagram Pengecekan Plagiarisme*




*Gambar 3.8. Sequence Diagram Login Admin*




*Gambar 3.9. Sequence Diagram Kelola Persentase*

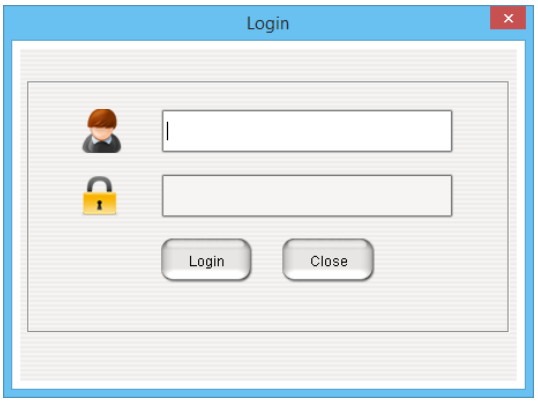
## 4. IMPLEMENTASI

Implementasi merupakan tahap untuk penerapan hasil dari analisis dan perancangan sistem yang telah di dilakukan sebelumnya yang bertujuan agar hasil dari analisis dan perancangan dapat digunakan sesuai dengan kebutuhan. Untuk tahap implementasi ini antara lain menerapkan algoritma kedalam perangkat lunak, menerapkan hasil perancangan *interface*, serta pembuatan sintak program dalam pembuatan komponen-komponen pokok dalam sebuah sistem berdasarkan desain yang telah dilakukan sebelumnya.

### 4.1. Implementasi *Interface* Admin

#### 1. Form Login

Form Login digunakan untuk melakukan proses login admin dengan menginputkan *Username* dan *Password* yang digunakan untuk mengakses halaman admin. Untuk tampilan form login seperti pada gambar 4.1.




*Gambar 4.1. Form Login*

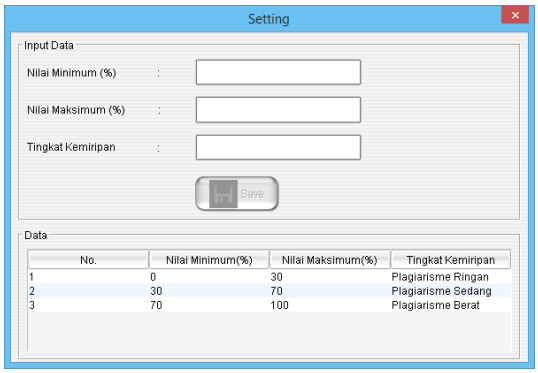
#### 2. Halaman Utama Admin

Halaman ini merupakan halaman utama yang akan tampil ketika proses login admin berhasil. Untuk tampilannya seperti pada gambar 4.2.


*Gambar 4.2. Halaman Utama Admin*

#### 3. Menu Setting

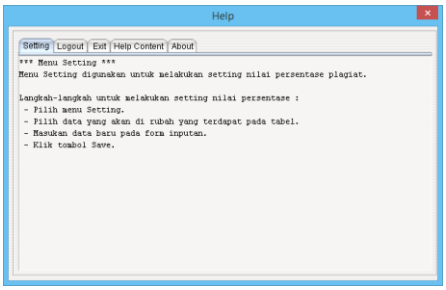
Menu Setting merupakan halaman untuk pengaturan nilai persentase plagiarisme. Untuk tampilannya seperti pada gambar 4.3.




*Gambar 4.3. Halaman Menu Setting*

**4. Menu Help**

Menu Help merupakan halaman yang berisi informasi tentang cara penggunaan aplikasi. Untuk tampilannya seperti pada gambar 4.4.




*Gambar 4.4. Halaman Menu Help*

**5. Menu About**

Menu About merupakan halaman yang berisi informasi tentang biodata pembuat program. Untuk tampilannya seperti pada gambar 4.5


*Gambar 4.5. Halaman Menu About*

**4.2. Implementasi Interface *User***

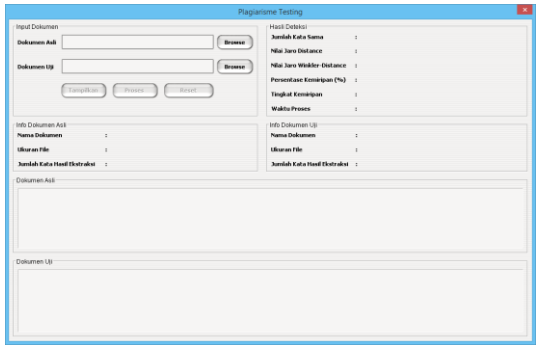
**1. Halaman Utama User**

Halaman utama user merupakan halaman yang pertama akan tampil ketika program dijalankan untuk tampilannya seperti pada gambar 4.6.


*Gambar 4.6. Halaman Utama*

**2. Menu Plagiarims Testing**

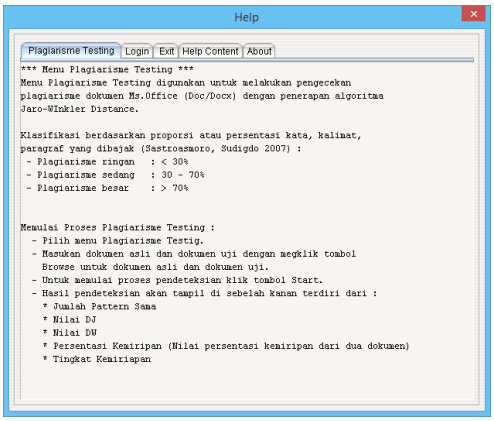
Menu plagiarisme testing merupakan halaman yang dapat di gunakan untuk melakukan proses pendeteksian pagiarisme. Untuk tampilan halaman menu plagiarisme testing seperti pada gambar 4.7.



*Gambar 4.7. Halaman Menu Plagiarisme Testing*

**3. Menu Help**

Menu Help merupakan halaman yang berisi informasi tentang cara penggunaan aplikasi. Untuk tampilannya seperti pada gambar 4.8.




### 4. Menu About

Menu About merupakan halaman yang berisi informasi tentang biodata pembuat program. Untuk tampilannya seperti pada gambar 4.5


*Gambar 4.5. Halaman Menu About*

## 5. KESIMPULAN

Berdasarkan hasil pembahasan pada bab sebelumnya maka dapat diambil kesimpulan sebagai berikut :

1. Sistem pendeteksi plagiarisme dengan menggunakan algoritma *Jaro-Winkler Distance* dapat digunakan untuk mendeteksi plagiarisme dokumen skripsi dengan cara melakukan perbandingan antara dokumen asli dan dokumen uji yang diinputkan untuk mengetahui tingkat kemiripan (*similarity*) dari dokumen skripsi yang diuji.
2. Sistem pendeteksi plagiarisme dengan algoritma *Jaro-Winkler Distance*, waktu proses yang dibutuhkan dalam melakukan pendeteksian plagiarisme bergantung pada banyaknya isi dokumen, dan ukuran file yang akan di proses.

## DAFTAR PUSTAKA

Anna Kurniawati, Sulistyo Puspitodjati, Sazali Rahman (2010), *Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia*. Depok : Universitas Gunadarma.

A.S Rosa & M. Shalahuddin. (2013), *Modul Pembelajaran Rekayasa Perangkat Lunak*, Bandung : Informatika.

Hartati, Sri. (2007). *Pemrograman GUI Swing Java dengan NetBeans 5/G. Sri Hartati, B. Herry Suharto dan M.Soesilo Wijono;-Ed.1*.Yogyakarta : Andi.

Kamus Besar Bahasa Indonesia (1997). Jakarta : Pusat Pembinaan dan Pengembangan Bahasa.

Kornain, Ahmad, Ferry Yansen (2010), *Penerapan Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Teks Berbahasa Indonesia*, Program Studi Teknik Informatika, STMIK GI MDP.

Nugroho, Adi (2005). *Rational Rose Untuk Pemodelan Berorientasi Objek*, Bandung : Informatika,.

Nugroho, Eko 2011. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks dengan Menggunakan Algoritma Rabin-Karp*, Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijaya, Malang.

Ridhatillah, Ardini. 2003. *Dealing with Plagiarism in the Information System Research Community: A Look at Factors that Drive Plagiarism and Ways to Address Them*, MIS Quarterly; Vol. 27, No. 4, p. 511-532/December 2003.

Sastroasmoro, Sudigdo 2007, *Beberapa Catatan tentang Plagiarisme*. Departemen Ilmu Kesehatan Anak, Fakultas Kedokteran Universitas Indonesia, Jakarta.

Sismoro, Heri (2005), *Pengantar Logika Informatika, Algoritma, dan Pemrograman Komputer*, Yogyakarta : Andi.

Triawati, Chandra. 2009. *Metode Pembobotan Statistical Concept Based untuk Klastering dan Kategorisasi Dokumen Berbahasa Indonesia*. Institut Teknologi Telkom. Bandung.